AIX เวอร์ชัน 7.2

AIX โกลบอลไลเซชัน

AIX เวอร์ชัน 7.2

AIX โกลบอลไลเซชัน

หมายเหตุ

ก่อนใช้ข้อมูลนี้ รวมถึงผลิตภัณฑ์ที่สนับสนุน โปรดอ่าน ข้อมูลใน “คำประกาศ” ในหน้า 279

This edition applies to AIX Version 7.2 and to all subsequent releases and modifications until otherwise indicated in new editions.

สารบัญ

เกี่ยวกับเอกสารนี้

เอกสารนี้ให้ข้อมูลที่สมบูรณ์สำหรับโปรแกรมเมอร์แอ็พพลิเคชันเกี่ยวกับการเปิดใช้งานแอ็พพลิเคชันสำหรับการสนับสนุน multicultural สำหรับระบบปฏิบัติการ AIX® นอกจากนี้ ยังให้ข้อมูลที่สมบูรณ์แก่ผู้ดูแลระบบเกี่ยวกับ การเปิดใช้งานสภาพแวดล้อมเครือข่ายสำหรับการใช้คุณลักษณะ globalization ในระบบปฏิบัติการ AIX โปรแกรมเมอร์และผู้ดูแลระบบสามารถใช้เอกสารนี้เพื่อศึกษาหาความรู้ เกี่ยวกับแนวทางและหลักการเกี่ยวกับ globalization หัวข้อที่มีอยู่คือ โลแคล ชุดโค้ด วิธีการอินพุต รูทีนย่อย ตัวแปลง การแม็พอักขระ ข้อมูลเฉพาะวัฒนธรรม และฟังก์ชันข้อความ

การไฮไลต์

วิธีการไฮไลต์ต่อไปนี้ถูกนำมาใช้ในคู่มือนี้:

ตัวหนา	ระบุคำสั่ง รูทีนย่อย คีย์เวิร์ด ไฟล์ โครงสร้าง ไดเร็กทอรี และไอเท็มอื่นๆ ซึ่งมีชื่อ ที่กำหนดไว้แล้วโดยระบบ นอกจากนี้ ยังระบุอ็อบเจ็กต์กราฟิก เช่น ปุ่ม ป้ายชื่อ และไอคอนที่ผู้ใช้เลือกด้วย
ตัวเอน	ระบุพารามิเตอร์ที่เป็นเจ้าของชื่อจริง หรือค่าที่ผู้ใช้จะระบุ
เว้นวรรคเดียว	ระบุตัวอย่างของค่าข้อมูลเฉพาะ ตัวอย่างของข้อความที่คล้ายกับข้อความที่คุณจะเห็นเมื่อแสดงขึ้น ตัวอย่าง ของส่วนของโค้ดโปรแกรมซึ่งคล้ายกับที่คุณจะบันทึกในฐานะโปรแกรมเมอร์ ข้อความจากระบบ หรือข้อมูลที่คุณควรจะพิมพ์จริง

การตรงตามตัวพิมพ์ใน AIX

ทุกสิ่งในระบบปฏิบัติการ AIX เป็น แบบตรงตามตัวพิมพ์ ซึ่งหมายความว่ามีการแยกแยะความแตกต่างระหว่าง ตัวอักษรพิมพ์ใหญ่และพิมพ์เล็ก ตัวอย่างเช่น คุณสามารถใช้คำสั่ง **ls** เพื่อ แสดงรายการไฟล์ ถ้าคุณพิมพ์ LS ระบบจะตอบกลับว่า ไม่พบคำสั่ง ในลักษณะคล้ายกัน **FILEA**, **FiLea**, และ **filea** เป็นชื่อไฟล์ที่แตกต่างกันสามไฟล์ แม้ว่า จะอยู่ในไดเร็กทอรีเดียวกัน เพื่อหลีกเลี่ยงการดำเนินการที่ไม่ต้องการ ควรตรวจสอบให้แน่ใจว่าคุณใช้ตัวพิมพ์ที่ถูกต้อง

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

การทำให้ AIX เป็นโกลบอล

Globalization จัดเตรียมคำสั่งแบะรูทีนย่อย Standard C Library สำหรับฐานระบบเดียวทั่วโลก ระบบที่ทำให้เป็นโกลบอล ไม่มีสมมติฐานหรือการขึ้นต่อกันในตัวสำหรับระเบียบเฉพาะภาษา หรือ เฉพาะวัฒนธรรมเช่น:

- ชุดโค้ด
- การจัดประเภทอักขระ
- กฎการเปรียบเทียบอักขระ
- ลำดับการจัดเรียงอักขระ
- การจัดรูปแบบตัวเลขและเงิน
- การจัดรูปแบบวันที่และเวลา
- ภาษาข้อความตัวอักษร

ข้อมูลทั้งหมดที่เกี่ยวกับระเบียบวัฒนธรรมและภาษา จะได้รับที่รันไทม์กระบวนการ

คุณลักษณะ globalization จัดเตรียมความสามารถต่อไปนี้เพื่อให้ระบบสามารถรันได้ในสภาวะแวดล้อมสากล:

- “การแยกข้อความออกจากโปรแกรม” ในหน้า 2
- “การแปลงระหว่างชุดโค้ด” ในหน้า 2

มีอะไรใหม่ในการทำ AIX ให้เป็นโกลบอล

อ่านเกี่ยวกับข้อมูลใหม่ หรือที่เปลี่ยนแปลงอย่างมากสำหรับ คอลเล็กชันหัวข้อการทำ AIX ให้เป็น โกลบอล

วิธีดูสิ่งใหม่ หรือที่เปลี่ยนแปลง

ในไฟล์ PDF นี้ คุณอาจเห็นแถบการแก้ไข (|) ในขอบด้านซ้ายเพื่อระบุข้อมูลใหม่ และที่เปลี่ยนแปลง

October 2016

ข้อมูลต่อไปนี้เป็นข้อมูลสรุป ของการอัพเดตที่มีในคอลเล็กชันหัวข้อนี้:

- เพิ่มภาษาต่างๆ ไปยังตาราง *ภาษาและโลแคลที่เข้ารหัส Unicode ไว้ในตาราง AIX* ในหัวข้อ “ภาษาและโลแคลที่ได้รับการสนับสนุน” ในหน้า 195

December 2015

ข้อมูลต่อไปนี้เป็นข้อมูลสรุป ของการอัพเดตที่มีในคอลเล็กชันหัวข้อนี้:

- เพิ่มภาษาต่างๆ ไปยังตาราง *ภาษาและโลแคลที่เข้ารหัส Unicode ไว้ในตาราง AIX* ในหัวข้อ “ภาษาและโลแคลที่ได้รับการสนับสนุน” ในหน้า 195
- โลแคล `@euro` และ `@preeuro` ไม่ได้รับการสนับสนุนอีกต่อไป
- โลแคล `Ja_JP.IBM-932` และคุณลักษณะที่เกี่ยวข้องล้าสมัยแล้วและถูกแทนที่ด้วยโลแคล `Ja_JP.IBM-943`

การแยกข้อความออกจากโปรแกรม

คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ ซึ่งจะช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความที่ถูกแปลพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้

หลักการที่เกี่ยวข้อง:

"ฟังก์ชันข้อความ" ในหน้า 173

คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ การจัดระเบียบนี้ช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้ เพื่อช่วยในภารกิจนี้ ฟังก์ชันข้อความจึงนำเสนอคำสั่งและรูทีนย่อย ต่างๆ

การแปลงระหว่างชุดโค้ด

อักขระคือสัญลักษณ์ใดๆ ที่ใช้สำหรับการจัดระเบียบ ควบคุม หรือการแสดงถึงข้อมูล กลุ่มของสัญลักษณ์ดังกล่าวที่ใช้ เพื่ออธิบายภาษาเฉพาะประกอบกันขึ้นเป็นชุดอักขระ ชุดโค้ดมี ค่าการเข้ารหัสสำหรับชุดอักขระ ค่าการเข้ารหัส ในชุดโค้ดทำหน้าที่เป็นอินเตอร์เฟสระหว่างระบบและอุปกรณ์อินพุต และเอาต์พุตของระบบนั้น ตัวแปลงการจัดส่ง การสนับสนุนหลายวัฒนธรรมที่สอดคล้องกับค่าการเข้ารหัสอักขระ ซึ่ง พบในชุดโค้ดที่แตกต่างอื่น

ในอดีต มีความพยายามเข้ารหัสตัวอักษรภาษาอังกฤษโดยตรง การใช้วิธีการเข้ารหัส 7 บิตสำหรับวัตถุประสงค์นี้จึงถือว่าเพียงพอแล้ว เนื่องจากจำนวนของอักขระภาษาอังกฤษมีอยู่ไม่มากนัก เพื่อสนับสนุน ตัวอักษรให้ได้มากขึ้น เช่น ภาษาในภูมิภาคเอเชีย อาทิภาษาจีน ภาษาญี่ปุ่น และภาษาเกาหลี จึงมีการพัฒนาชุดโค้ดเพิ่มเติมซึ่งมีการเข้ารหัสแบบ หลายไบต์ ตอนนี้ มีการใช้ Unicode ซึ่งเป็นชุดอักขระสำหรับสนับสนุน การประมวลผลข้อมูลทั่วโลก เป็นรูปแบบการแลกเปลี่ยนพื้นฐาน ในระดับระบบปฏิบัติการ ชุดโค้ด UTF-8, UTF-16 และ UTF-32 คือแบบแผนการเข้ารหัส Unicode ที่สำคัญสำหรับแอ็พพลิเคชันระบบ

โปรแกรมที่ทำให้เป็นโกลบอลต้องอ่านข้อมูลที่สร้างในสภาวะแวดล้อม ชุดโค้ดที่แตกต่างอื่นอย่างถูกต้อง และประมวลผลข้อมูลอย่างถูกต้อง การทราบชุดโค้ดปัจจุบันสามารถช่วยในการแปลงชุดโค้ดได้ คุณสามารถ ใช้รูทีนย่อย **nl_langinfo (CODESET)** เพื่อให้ทราบ ชุดโค้ดปัจจุบันในกระบวนการ ค่าที่ส่งคืนคือตัวชี้ `char` ซึ่งเป็น ชื่อของชุดโค้ดในระบบ

หลักการที่เกี่ยวข้อง:

"ภาพรวมของตัวแปลงสำหรับการเขียนโปรแกรม" ในหน้า 93

การสนับสนุนหลายวัฒนธรรมจะมีพื้นฐานสำหรับการทำให้เป็นโกลบอล ซึ่งข้อมูลมักถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีกโค้ดหนึ่ง มีการสนับสนุนตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

การสนับสนุนวิธีการอินพุต

อินพุตของอักขระมีความซับซ้อนมากขึ้นสำหรับภาษา ที่มีชุดอักขระจำนวนมาก ตัวอย่างเช่น ในภาษาจีน ภาษาเกาหลี และภาษาญี่ปุ่น ที่มีอักขระเป็นจำนวนมากและไม่สามารถแม็พคีย์แบบหนึ่งต่อหนึ่ง สำหรับ keystroke ของอักขระ อย่างไรก็ตาม วิธีการอินพุตพิเศษช่วยให้ผู้ใช้สามารถป้อนอักขระการออกเสียง หรือ stroke และแปลงอักขระนั้นเป็นอักขระภาษา แม่

แม็พคีย์บอร์ดที่เชื่อมโยงกับแต่ละคีย์บอร์ดจะตรงกับลำดับของ keystrokes ที่มีการเข้ารหัสที่เหมาะสมหนึ่งรายการขึ้นไป

หลักการที่เกี่ยวข้อง:

“วิธีอินพุต” ในหน้า 141
สำหรับแอ็พพลิเคชันที่จะรันในสภาวะแวดล้อมระหว่างประเทศที่ globalization เป็นพื้นฐาน จำเป็นต้องใช้วิธีการอินพุต วิธีการอินพุตคืออินเตอร์เฟสการเขียนโปรแกรมแอ็พพลิเคชัน (API) ที่ช่วยให้คุณสามารถจัดทำภาษา คีย์บอร์ด หรือชุดโค้ดเฉพาะที่ เป็นอิสระจากแอ็พพลิเคชัน

ภาพรวมของตัวแปลง

การทำให้เป็นโกลบอลมีพื้นฐานสำหรับการทำให้เป็นโกลบอลเพื่อให้ ข้อมูลถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีกโค้ดหนึ่ง คุณอาจ ต้องการแปลงไฟล์ข้อความหรือแค็ตตาล็อกข้อความ มีตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

เมื่อโปรแกรมส่งข้อมูลไปยังโปรแกรมอื่นที่อยู่บนรีโมตโฮสต์ ข้อมูลนั้นอาจต้องการการแปลงจากชุดโค้ดของเครื่องต้นทาง เป็น ชุดโค้ดของเครื่องรับ ตัวอย่างเช่น เมื่อสื่อสารกับระบบ IBM® VM ระบบจะแปลงข้อมูล ISO8859-1 เป็น EBCDIC ชุดโค้ดเป็นตัวกำหนดอักขระและการกำหนดฟังก์ชันควบคุมให้กับจุดโค้ด อักขระที่เข้ารหัสเหล่านี้ต้องได้รับการแปลงเมื่อโปรแกรมได้รับ ข้อมูลในชุดโค้ดหนึ่งแต่แสดงข้อมูลนั้นในอีกชุดโค้ดหนึ่ง

หลักการที่เกี่ยวข้อง:

“ภาพรวมของตัวแปลงสำหรับการเขียนโปรแกรม” ในหน้า 93
การสนับสนุนหลายวัฒนธรรมจะมีพื้นฐานสำหรับการทำให้เป็นโกลบอล ซึ่งข้อมูลมักถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีกโค้ดหนึ่ง มีการสนับสนุนตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

การใช้ฟังก์ชันข้อความ

เพื่อช่วยในการแปลข้อความเป็นภาษาต่างๆ และทำให้โปรแกรมสามารถใช้ข้อความตามโลแคลของผู้ใช้ คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมและแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงได้เมื่อรันไทม์ เพื่อช่วยในภารกิจนี้ ฟังก์ชันข้อความจึงนำเสนอคำสั่งและรูทีนย่อย ต่างๆ ไฟล์ต้นฉบับข้อความที่มีข้อความแอ็พพลิเคชันมีการสร้างขึ้น โดยโปรแกรมเมอร์และถูกแปลงเป็นแค็ตตาล็อกข้อความ แอ็พพลิเคชัน ใช้แค็ตตาล็อกเหล่านี้เพื่อดึงข้อมูลและแสดงข้อความ ตามความต้องการ ไฟล์ต้นฉบับข้อความสามารถแปลเป็นภาษาอื่น แล้วแปลง เป็นแค็ตตาล็อกข้อความได้โดยไม่ต้องเปลี่ยนและรีคอมไพล์โปรแกรม

โปรแกรมอำนวยความสะดวกข้อความประกอบด้วยคำสั่งต่อไปนี้สำหรับการแสดง ข้อความด้วยเชลล์สคริปต์ หรือจากบรรทัดรับคำสั่ง

คำสั่ง	คำอธิบาย
dspcat	แสดงแค็ตตาล็อกข้อความทั้งหมดหรือบางส่วน
dspmsg	แสดงข้อความที่เลือกจากแค็ตตาล็อกข้อความ

คำสั่งเหล่านี้ใช้ตัวแปรสภาพแวดล้อม NLSPATH เพื่อระบุตำแหน่งแค็ตตาล็อกข้อความที่ระบุ ตัวแปรสภาพแวดล้อม NLSPATH แสดงรายการไดเร็กทอรีที่มีแค็ตตาล็อกข้อความ มีการค้นหา ไดเร็กทอรีเหล่านี้ตามลำดับที่แสดงรายการ ตัว อย่างเช่น:

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/prime/%N
```

ตัวแปรพิเศษ %L และ %N มีการกำหนด ดังนี้:

ตัวแปรพิเศษ	คำอธิบาย
	ระบุไดเร็กทอรีเฉพาะโลแคลที่มี แค็ตตาล็อกข้อความ ค่าของหมวดหมู่ **LC_MESSAGES** หรือตัวแปรสภาพแวดล้อม **LANG** ใช้สำหรับชื่อไดเร็กทอรี ผู้ใช้สามารถตั้งค่าตัวแปรสภาพแวดล้อม **LANG, LC_ALL,** หรือ **LC_MESSAGES** ให้กับโลแคลสำหรับ
%L	แค็ตตาล็อกข้อความ
%N	ระบุชื่อของแค็ตตาล็อกที่จะเปิด

ถ้าคำสั่ง **dspcat** ไม่พบข้อความ ข้อความดีฟอลต์ จะแสดงขึ้น คุณต้องใส่ข้อความดีฟอลต์ไว้ในเครื่องหมายคำพูดขีดเดียว ถ้าข้อความดีฟอลต์มีสตริงรูปแบบ *%n*$ ถ้าคำสั่ง **dspcat** ไม่พบข้อความและคุณไม่ได้ ระบุข้อความดีฟอลต์ ข้อความแสดงข้อผิดพลาดที่ระบบสร้างขึ้น จะแสดงขึ้น

ตัวอย่างต่อไปนี้ใช้คำสั่ง **dspcat** เพื่อแสดง ข้อความทั้งหมดในแค็ตตาล็อกข้อความ `msgerrs.cat` ที่มีอยู่:

```
/usr/lib/nls/msg/$LANG/msgerrs.cat:
dspcat msgerrs.cat
```

เอาต์พุตที่แสดงขึ้นมีดังต่อไปนี้:

```
1:1 ไม่สามารถเปิดแค็ตตาล็อกข้อความ %s
จำนวนสูงสุดของแค็ตตาล็อกที่เปิดแล้ว
1:2 ไฟล์ %s ดำเนินการไม่ได้
2:1 ไม่พบข้อความ %d, ชุด %d
```

โดยการแสดงเนื้อหาของแค็ตตาล็อกข้อความในลักษณะนี้ คุณจึงสามารถค้นหาหมายเลข ID ข้อความที่กำหนดให้กับไฟล์ต้นฉบับข้อความ `msgerrs` โดยคำสั่ง **mkcatdefs** เพื่อแทนที่ตัวระบุที่เป็น สัญลักษณ์ได้ ตัวระบุที่เป็นสัญลักษณ์ไม่พร้อมที่จะใช้เป็นข้อมูลอ้างอิงสำหรับคำสั่ง **dspmsg** แต่การใช้คำสั่ง **dspcat** ดังที่แสดงไว้ช่วยให้คุณได้หมายเลข ID ที่จำเป็น

ข้อมูลต่อไปนี้คือ shell script แบบง่ายที่เรียกว่า **runtest** ซึ่ง จะแสดงวิธีการใช้คำสั่ง **dspmsg**:

```
if [ - x ./test ]
    ./test;
else
    dspmsg  msgerrs.cat -s 1 2 '%s NOT EXECUTABLE \n' "test";
    exit;
```

หมายเหตุ: ถ้าคุณไม่ได้ใช้ชื่อพาธเต็ม ดังเช่นในตัวอย่างก่อนหน้านี้ ให้ระมัดระวังในการตั้งค่าตัวแปรสภาพแวดล้อม **NLSPATH** เพื่อให้คำสั่ง **dspcat** ค้นหาไดเร็กทอรีที่ถูกต้องสำหรับแค็ตตาล็อก หมวดหมู่ **LC_MESSAGES** หรือค่าของตัวแปรสภาพแวดล้อม **LANG** ยังกระทบต่อพาธการค้นหาไดเร็กทอรีด้วย

การตั้งค่าการสนับสนุน multicultural สำหรับอุปกรณ์

การสนับสนุน multicultural ใช้การตั้งค่าโลแคลเพื่อกำหนดสภาวะแวดล้อม การตั้งค่าโลแคลจะขึ้นอยู่กับความต้องการของผู้ใช้สำหรับการประมวลผลข้อมูลและภาษาที่กำหนดความต้องการอุปกรณ์อินพุตและเอาต์พุต ผู้ดูแลระบบรับผิดชอบในการตั้งค่าคอนฟิกอุปกรณ์ที่ สอดคล้องกับโลแคลผู้ใช้

เทอร์มินัล (อุปกรณ์ tty)

ใช้คำสั่ง **setmaps** เพื่อตั้งค่าเทอร์มินัลและ แม็พชุดโค้ดสำหรับ tty หรือ pty ที่กำหนด รูปแบบไฟล์ **setmaps** จะกำหนดข้อความของไฟล์แม็พชุดโค้ดและไฟล์แม็พเทอร์มินัล

ข้อความของไฟล์แม็พชุดโค้ดคือคำอธิบายของชุดโค้ด รวมถึงชนิด (ไบต์เดียวหรือหลายไบต์) หน่วยความจำและความกว้างของจอภาพ (สำหรับชุดโค้ดหลายไบต์) และโมดูลตัวแปลงที่เลือกกำหนดได้ ซึ่งจะออกใช้บนสตรีม ไฟล์แม็พชุดโค้ดอยู่ในไดเร็กทอรี **/usr/lib/nls/csmap** และมีชื่อเหมือนกับชุดโค้ด หากต้องการข้อมูลเพิ่มเติม ให้ดูโมดูลตัวแปลง ใน *General Programming Concepts: Writing and Debugging Programs*

เครื่องพิมพ์

เครื่องพิมพ์เสมือนสืบทอดชุดโค้ดดีฟอลต์ของงานที่เข้ามา จากรายการ **LANG** ในไฟล์ **/etc/environment** ระบบย่อยเครื่องพิมพ์สามารถสนับสนุนเครื่องพิมพ์เสมือนได้หลายเครื่อง ถ้ามีการสนับสนุน เครื่องพิมพ์เสมือนมากกว่าหนึ่งเครื่อง แต่ละเครื่องอาจมีชุดโค้ดที่แตกต่างกัน ได้ สถานการณ์จำลองของระบบย่อยเครื่องพิมพ์ที่แนะนำมีดังนี้:

- สถานการณ์จำลองแรกเกี่ยวข้องกับหลายคิว หลายเครื่องพิมพ์เสมือน และหนึ่งเครื่องพิมพ์ทางกายภาพ เครื่องพิมพ์เสมือนแต่ละเครื่องมีชุดโค้ดของตนเอง คำสั่งพิมพ์ระบุคิวที่จะใช้ จากนั้น คิวระบุ เครื่องพิมพ์เสมือนซึ่งมีชุดโค้ดที่เหมาะสม ในสถานการณ์ จำลองนี้ ผู้ใช้ต้องทราบว่าคิวใดถูกแนบกับเครื่องพิมพ์เสมือนใด และชุดโค้ดที่เชื่อมโยงกับแต่ละเครื่อง
- สถานการณ์จำลองที่สองคล้ายกับสถานการณ์แรก แต่เครื่องพิมพ์เสมือน แต่ละเครื่องถูกแนบกับเครื่องพิมพ์ที่แตกต่างกัน
- สถานการณ์จำลองที่สามเกี่ยวข้องกับการใช้คำสั่ง **qprt** เพื่อระบุ ชุดโค้ด ในอ็อพชันนี้ มีคิวอยู่หลายคิวและมีเครื่องพิมพ์เสมือน หนึ่งเครื่อง เครื่องพิมพ์เสมือนจะใช้ชุดโค้ดดีฟอลต์ ที่สืบทอดมา

ใช้คำสั่ง **qprt** ที่มีแฟล็ก **–P–x** เพื่อระบุ คิวและชุดโค้ด ถ้าไม่ได้ระบุแฟล็ก **–P** ระบบจะใช้ คิวดีฟอลต์ ถ้าไม่ได้ใช้แฟล็ก **–x** ระบบจะใช้ ชุดโค้ดดีฟอลต์สำหรับเครื่องพิมพ์เสมือน

เทอร์มินัลฟังก์ชันต่ำ

เทอร์มินัลฟังก์ชันต่ำ (LFTs) สนับสนุนภาษาชุดโค้ดไบต์เดียว โดยใช้คีย์แม็พ คีย์แม็พ LFT จะแปล key stroke เป็น สตริงอักขระในชุดโค้ด รายการของคีย์แม็พที่มีอยู่ทั้งหมด มีอยู่ในไดเร็กทอรี **/usr/lib/nls/loc** LFT ไม่สนับสนุน ภาษาที่ต้องการชุดโค้ดหลายไบต์

การตั้งค่าคีย์บอร์ด LFT ดีฟอลต์และการตั้งค่าฟอนต์ที่เชื่อมโยงใช้ ภาษาที่เลือกในระหว่างการติดตั้ง ชุดโค้ดดีฟอลต์ ที่เป็นไปได้มีดังนี้:

- ISO8859-1
- ISO8859-2
- ISO8859-4
- ISO8859-5
- ISO8859-6
- ISO8859-7
- ISO8859-8
- ISO8859-9
- ISO8859-15

คุณสามารถเปลี่ยนการตั้งค่าดีฟอลต์ในวิธีดังต่อไปนี้:

- เมื่อต้องการเปลี่ยนฟอนต์ดีฟอลต์สำหรับรีบูตครั้งถัดไป ให้ใช้คำสั่ง **chfont** ที่มีแฟล็ก **–n**
- เมื่อต้องการเปลี่ยนคีย์บอร์ดดีฟอลต์สำหรับรีบูตครั้งถัดไป ให้ใช้คำสั่ง **chkbd** ที่มีแฟล็ก **–n**

คำสั่ง **lsfont** และ **lskbd** จะแสดงรายการแม็พฟอนต์ และคีย์บอร์ดทั้งหมดที่มีอยู่ในปัจจุบันสำหรับ LFT

ไลบรารีฟอนต์ LFT สำหรับชุดโค้ดที่ได้รับการสนับสนุนทั้งหมดอยู่ในไดเร็กทอรี **/usr/lpp/fonts**

การเปลี่ยนสภาพแวดล้อมภาษา

การดำเนินงานของระบบหลายอย่างได้รับผลกระทบจากสภาพแวดล้อม ภาษา การดำเนินงานดังกล่าวบางอย่างได้แก่ การเรียงหน้า การแสดงเวลาของวัน และวันที่ การแสดงตัวเลข การแสดงทางการเงิน และการแปลข้อความ สภาพแวดล้อมภาษามีการกำหนดโดย ค่าของตัวแปรสภาพแวดล้อม **LANG** และคุณสามารถเปลี่ยน ค่านั้นได้โดยใช้คำสั่ง **chlang** คำสั่ง **chlang** สามารถรันได้จากบรรทัดคำสั่งหรือจาก SMIT

เมื่อต้องการใช้พาธด่วน SMIT เพื่อเปลี่ยนสภาพแวดล้อมภาษา ให้พิมพ์ `smit chlang` บนบรรทัดคำสั่ง

การเปลี่ยนแม็พคีย์บอร์ดดีฟอลต์

คุณสามารถระบุคีย์บอร์ดที่ถูกต้องสำหรับภาษาที่ คุณต้องการใช้งาน ระบบปฏิบัติการแสดงหมายเลข ของแม็พคีย์บอร์ดสำหรับวัตถุประสงค์นี้ คุณสามารถเปลี่ยนการแม็พคีย์บอร์ดดีฟอลต์ สำหรับเทอร์มินัล LFT โดยใช้พาธด่วนของ SMIT **smit chkbd** หรือ คำสั่ง **chkbd** การเปลี่ยนแปลงไม่มีผลใช้งานจนกว่า คุณรีสตาร์ทระบบ

หมายเหตุ: อย่าสันนิษฐานว่ามีการใช้คีย์บอร์ดฟิสิคัลใดอยู่ให้ใช้วิธีการอินพุตตามค่าติดตั้งโลแคลเพื่อจัดการ คีย์บอร์ดอินพุต

ไลบรารี ICU4C

คอมโพเนนต์ทั่วไป ICU4C หมายถึงคลาสไลบรารี International Components for Unicode for C/C++ ซึ่งมียูทิลีตีการทำให้เป็นโกลบอล สำหรับการเขียนโกลบอลแอ็พพลิเคชันในภาษาโปรแกรม C/C++ ไลบรารี ICU4C ใช้โดยผลิตภัณฑ์จำนวนมากที่กำลังรันบนระบบปฏิบัติการ AIX

โลแคล

ระบบที่ทำให้เป็นโกลบอลไม่มีสมมติฐานในตัว หรือการพึ่งพา ชุดอักขระ การจัดประเภทอักขระ กฎการเปรียบเทียบอักขระ ลำดับการตรวจเทียบอักขระ การจัดรูปแบบเงิน การคั่นตัวเลข การจัดรูปแบบวันที่และเวลา หรือตัวอักษรของข้อความ *โลแคล* มีการ กำหนดโดยระเบียบภาษาและวัฒนธรรมเหล่านี้ ระบบที่ทำให้เป็นโกลบอล ประมวลผลข้อมูลอย่างถูกต้องสำหรับตำแหน่งที่ต่างกัน ตัวอย่างเช่น ในสหรัฐอเมริกา รูปแบบวันที่คือ 9/6/2015 ถูกตีความให้มี ความหมายถึงวันที่หกเดือนเก้าปี 2015 สหราชอาณาจักรตีความรูปแบบวันที่เดียวกัน ซึ่งหมายถึงวันที่เก้าเดือนหกปี 2015 การจัดรูปแบบของข้อมูลตัวเลขและการเงิน เป็นลักษณะเฉพาะประเทศเช่นกัน ตัวอย่างเช่น ดอลลาร์สหรัฐฯ และ ปอนด์สหราชอาณาจักร

โปรแกรมต้องสามารถเข้าถึงข้อมูลโลแคลทั้งหมดที่รันไทม์ เพื่อให้ข้อมูลมีการประมวลผลและแสดงขึ้นอย่างถูกต้องสำหรับระเบียบตาม วัฒนธรรมและภาษาของคุณ กระบวนการนี้เรียกว่าการสนับสนุนหลายวัฒนธรรม การสนับสนุนหลายวัฒนธรรมประกอบด้วยการจัดทำฐานข้อมูลที่มี กฎเฉพาะโลแคลสำหรับการจัดรูปแบบข้อมูล และอินเตอร์เฟสเพื่อให้ได้ รับกฎ

การทำความเข้าใจกับโลแคล

โลแคลประกอบด้วยชุดข้อมูลภาษา เขตแดน และชุดโค้ด ที่ใช้เพื่อระบุชุดของแบบแผนภาษา แบบแผน เหล่านี้มีข้อมูลเกี่ยวกับการจัดเรียง การแปลงตัวพิมพ์ และ การจัดประเภทอักขระ ภาษาของแค็ตตาล็อกข้อความ การแสดงวันที่และเวลา สัญลักษณ์การเงิน และการแสดงตัวเลข

ข้อมูลโลแคลที่มีอยู่ในไฟล์ต้นฉบับ **คำนิยามโลแคล** ต้องถูกแปลง เข้าในฐานข้อมูลโลแคลก่อน โดยใช้คำสั่ง **localedef** จากนั้น รูทีนย่อย **setlocale** สามารถเข้าถึงข้อมูลนี้และ ตั้งค่าข้อมูลโลแคลสำหรับแอ็พพลิเคชันได้ เพื่อจัดการกับข้อมูลโลแคล ในลักษณะเชิงตรรกะ จึงมีการแบ่งไฟล์ต้นฉบับคำนิยามโลแคลออกเป็น หกหมวดหมู่ แต่ละหมวดหมู่มีแง่มุมเฉพาะของข้อมูลโลแคล ตัวแปรสภาพแวดล้อม **LC_*** และตัวแปรสภาพแวดล้อม LANG สามารถใช้เพื่อระบุโลแคลที่ต้องการได้

หลักการที่เกี่ยวข้อง:

"การทำความเข้าใจกับหมวดหมู่โลแคล" ในหน้า 8

หมวดหมู่โลแคล คือการจัดกลุ่มเฉพาะของข้อมูลเฉพาะภาษาและ ข้อมูลเฉพาะแบบแผนวัฒนธรรม ตัวอย่างเช่น ข้อมูลที่อ้างอิง การจัดรูปแบบวันที่และเวลา ชื่อวันของสัปดาห์ ชื่อ ของเดือน และข้อมูลเฉพาะเวลาอื่นๆ จะมีการจัดกลุ่มเข้าใน หมวดหมู่ **LC_TIME** แต่ละหมวดหมู่ใช้ชุด ของคีย์เวิร์ดที่อธิบายชุดย่อยของโลแคลนั้น

สถานการณ์ของผู้ใช้ทั่วไป

คุณอาจพบสถานการณ์จำลองที่เกี่ยวข้องกับสภาพวัฒนธรรมที่หลายหลาย ในระบบ ส่วนนี้แสดงรายการสถานการณ์จำลองทั่วไปเกี่ยวกับการดำเนินการที่แนะนำ ให้ใช้

- ผู้ใช้ยังคงใช้ชุดโค้ดดีฟอลต์

  คุณอาจพอใจกับ ชุดโค้ดดีฟอลต์สำหรับการผสานภาษา-เขตแดน แม้ว่ามีการสนับสนุน มากกว่าหนึ่งชุดโค้ด คุณยังคงใช้ชุดโค้ดดีฟอลต์ต่อไป แม้ว่าสภาวะแวดล้อมของผู้ใช้ปัจจุบันจะใช้ชุดโค้ดนั้น หรือแม้ว่าคุณ เป็นผู้ใช้ใหม่ และยังไม่มีการกำหนดค่าตามความชอบของชุดโค้ด

  ภาษา-เขตแดน ที่ถูกเลือกในเวลาที่ติดตั้งระบบจะถูกกำหนดเป็นดีฟอลต์ให้กับโลแคลที่เหมาะสม ตามชุดโค้ดดีฟอลต์ การแม็พคีย์บอร์ดดีฟอลต์ ฟอนต์ดีฟอลต์ และแค็ตตาล็อกข้อความ ล้วนจัดทำขึ้นตามข้อมูล ชุดโค้ดดีฟอลต์ สถานการณ์จำลองนี้ไม่ต้องการการดำเนินการพิเศษจากคุณ
- ผู้ใช้เปลี่ยนชุดโค้ดจากชุดโค้ดดีฟอลต์

  ผู้ใช้ที่ใช้ โลแคล ภาษาลาติน-1 หรือภาษาญี่ปุ่นอาจต้องการโอนย้ายข้อมูล และสภาวะแวดล้อมการสนับสนุน multicultural ไปยังชุดโค้ดอื่น (ที่ไม่ใช่ชุดโค้ดดีฟอลต์) การโอนย้ายนี้ สามารถทำได้ด้วยวิธีการต่อไปนี้:
  - เมื่อคุณมีข้อมูลเดิมที่ต้องการแปลง

    ไฟล์ข้อความแบบ Flat ที่จำเป็นต้องแปลงให้เป็นชุดโค้ดที่ต้องการสามารถ แปลงได้โดยใช้เมนู SMIT **จัดการสภาวะแวดล้อมภาษา**, หรือยูทิลีตี **iconv** ไฟล์โครงสร้างที่ผู้ใช้กำหนดเอง จำเป็นต้องแปลงผ่านเครื่องมือการแปลงที่ผู้ใช้เขียนขึ้นเอง ซึ่งใช้ฟังก์ชันไลบรารี **iconv** เพื่อแปลง ฟิลด์ข้อความที่จำเป็นภายในไฟล์โครงสร้าง
  - เมื่อคุณต้องการเปลี่ยนเป็นชุดโค้ดอื่น

    เมื่อมีการสนับสนุนชุดโค้ด มากกว่าหนึ่งชุดสำหรับชุดภาษา-เขตแดน คุณสามารถเปลี่ยนโลแคลที่ไม่ใช่ดีฟอลต์ได้โดยใช้อ็อพชันต่อไปนี้:
    - เมนู SMIT **จัดการสภาวะแวดล้อมภาษา**
    - คำสั่ง **chlang**, **chkbd**, และ **chfont**

แนวทางการตั้งชื่อโลคัล

แต่ละโลแคลมีการตั้งชื่อโดยใช้ชื่อของไฟล์ต้นฉบับคำนิยาม โลแคล ไฟล์เหล่านี้มีการตั้งชื่อเพื่ออธิบายถึงข้อมูลภาษา เขตแดน และ ชุดโค้ด รูปแบบที่ใช้สำหรับการตั้งชื่อ ไฟล์คำนิยามโลแคลมีดังต่อไปนี้:

```
language[_territory][.codeset][@modifier]
```

ตัวอย่างเช่น โลแคลสำหรับภาษาเดนิชที่พูดกันในประเทศเดนมาร์ก ที่ใช้ชุดโค้ด ISO8859-1 คือ da_DK.ISO8859-1 da ย่อมาจากภาษาเดนิชและ DK ย่อมาจาก เดนมาร์ก รูปแบบย่อ da_DK เพียงพอที่จะ บ่งชี้โลแคลนี้ได้ ภาษาและเขตแดนเดียวกันที่ใช้ชุดโค้ด ISO8859-15 มีการบ่งชี้โดย da_DK.8859-15

ไฟล์คำนิยามโลแคลที่ระบบกำหนดมีการนำเสนอเพื่อแสดง รูปแบบของหมวดหมู่โลแคลและคีย์เวิร์ด ไดเร็กทอรี **/usr/lib/nls/loc** มีไฟล์คำนิยามโลแคลสำหรับโลแคลที่ระบบกำหนด โลแคล C หรือ POSIX กำหนดโลแคลมาตรฐานที่ ANSI C กำหนดไว้ซึ่งสืบทอด โดยกระบวนการทั้งหมด ณ เวลาสตาร์ทอัพ เมื่อต้องการได้รับรายชื่อของไฟล์ต้นฉบับคำนิยามโลแคล ที่ระบบกำหนด ให้ป้อนข้อมูลต่อไปนี้บนบรรทัด คำสั่ง:

```
/usr/lib/nls/lsmle -c
```

โลคัลดีฟอลต์การติดตั้ง

โลแคลดีฟอลต์การติดตั้งหมายถึงโลแคลที่เลือกเมื่อ ติดตั้ง ตัวอย่างเช่น เมื่อพร้อมต์ ผู้ใช้สามารถระบุภาษาฝรั่งเศสตามที่พูดกันในประเทศแคนาดาในระหว่างกระบวนการติดตั้ง ชุดโค้ดจะดีฟอลต์เป็นชุดโค้ด ISO8859-1 โดยอัตโนมัติ

ด้วยข้อมูลนี้ ระบบจะตั้งค่าของโลแคลดีฟอลต์ ที่ระบุโดยตัวแปรสภาพแวดล้อม **LANG** เป็น **fr_CA** (**fr** หมายถึง ภาษาฝรั่งเศส ISO8859-1 และ **CA** หมายถึงแคนาดา) ทุกกระบวนการจะใช้โลแคลนี้ ยกเว้นว่ามีการดัดแปลงตัวแปรสภาพแวดล้อม **LC_*** หรือ **LANG** โลแคลดีฟอลต์สามารถเปลี่ยนได้โดยใช้เมนู **จัดการ สภาวะแวดล้อมภาษา** ใน SMIT

เมื่อการสำรองระบบถูกสร้างขึ้นและติดตั้งอีกครั้ง ค่าโลแคลดีฟอลต์จะถูกใช้ ในไฟล์ /bosinst.data หากมี และในไฟล์ /var/adm/ras/bosinst.data ไฟล์สองไฟล์นี้ไม่ถูกอัพเดตโดยอัตโนมัติ เมื่อคุณเปลี่ยนค่าโลแคลโดยใช้คำสั่ง **smit mlang** ในสถานการณ์นี้ เมื่อต้องการให้ตรงกับค่าโลแคลของระบบที่รันอยู่ คุณต้องเปลี่ยน stanza ในไฟล์ /bosinst.data หากมีและในไฟล์ /var/adm/ras/bosinst.data

ข้อมูลที่เกี่ยวข้อง:

อินเตอร์เฟสการจัดการระบบที่มี

โลแคล C หรือ POSIX

โลแคลนี้อ้างอิงมาตรฐานที่ ANSI C หรือ POSIX กำหนดไว้ สำหรับโลแคลซึ่งกระบวนการทั้งหมดได้รับสืบทอดมา ณ เวลาสตาร์ทอัพ โลแคล C หรือ POSIX สมมุติชุดอักขระ 7 บิต ASCII และกำหนด ข้อมูลสำหรับหกหมวดหมู่ก่อนหน้านี้

การทำความเข้าใจกับหมวดหมู่โลแคล

หมวดหมู่โลแคล คือการจัดกลุ่มเฉพาะของข้อมูลเฉพาะภาษาและ ข้อมูลเฉพาะแบบแผนวัฒนธรรม ตัวอย่างเช่น ข้อมูลที่อ้างอิง การจัดรูปแบบวันที่และเวลา ชื่อวันของสัปดาห์ ชื่อ ของเดือน และข้อมูลเฉพาะเวลาอื่นๆ จะมีการจัดกลุ่มเข้าใน หมวดหมู่ **LC_TIME** แต่ละหมวดหมู่ใช้ชุด ของคีย์เวิร์ดที่อธิบายชุดย่อยของโลแคลนั้น

หมวดหมู่มาตรฐานที่สามารถกำหนดไว้ในไฟล์ต้นฉบับคำนิยามโลแคล มีดังต่อไปนี้:

LC_COLLATE
: กำหนดข้อมูลการจัดเรียงอักขระหรือการจัดเรียงสตริง

LC_CTYPE
: กำหนดการจัดประเภทอักขระ การแปลงตัวพิมพ์ และแอ็ตทริบิวต์อักขระ อื่นๆ

LC_MESSAGES
: กำหนดรูปแบบของการตอบรับและการตอบปฏิเสธ

LC_MONETARY
กำหนดกฎและสัญลักษณ์สำหรับการจัดรูปแบบข้อมูลตัวเลขทางการเงิน

LC_NUMERIC
กำหนดกฎและสัญลักษณ์สำหรับการจัดรูปแบบข้อมูลตัวเลขที่ไม่ใช่การเงิน

LC_TIME
กำหนดรายการของกฎและสัญลักษณ์สำหรับการจัดรูปแบบข้อมูลเวลาและ วันที่

หมายเหตุ: หมวดหมู่โลแคลสามารถดัดแปลงได้โดยการแก้ไขไฟล์ต้นฉบับคำนิยามโลแคล เท่านั้น อย่าสับสนกับตัวแปรสภาพแวดล้อมที่มีชื่อเหมือนกัน ซึ่งสามารถตั้งค่าได้จากบรรทัดคำสั่ง

หลักการที่เกี่ยวข้อง:

"การทำความเข้าใจกับโลแคล" ในหน้า 6
โลแคลประกอบด้วยชุดข้อมูลภาษา เขตแดน และชุดโค้ด ที่ใช้เพื่อระบุชุดของแบบแผนภาษา แบบแผน เหล่านี้มีข้อมูลเกี่ยวกับการจัดเรียง การแปลงตัวพิมพ์ และ การจัดประเภทอักขระ ภาษาของแค็ตตาล็อกข้อความ การแสดงวันที่และเวลา สัญลักษณ์การเงิน และการแสดงตัวเลข

การทำความเข้าใจกับตัวแปรสภาพแวดล้อมโลแคล

การสนับสนุน Multicultural ใช้ตัวแปรสภาวะแวดล้อมหลายตัวเพื่อให้มีผลต่อการเลือกโลแคล คุณสามารถตั้งค่า ของตัวแปรเหล่านี้เพื่อเปลี่ยนพาธการค้นหาสำหรับข้อมูลโลแคลได้ดังนี้:

LANG ระบุโลแคลดีฟอลต์การตั้งค่า

หมายเหตุ: ค่าตัวแปรสภาพแวดล้อม **LANG** จัดสร้างขึ้นเมื่อติดตั้ง (นี่เป็นโลแคล ที่ทุกกระบวนการจะใช้ ยกเว้นในกรณีที่มีการตั้งค่าตัวแปรสภาพแวดล้อม **LC_***) ตัวแปรสภาวะแวดล้อม **LANG** สามารถเปลี่ยนได้โดยใช้เมนู จัดการสภาพแวดล้อมภาษา Language Environment® ใน SMIT หากต้องการข้อมูลเพิ่มเติม เกี่ยวกับการใช้ SMIT ให้ดู อินเตอร์เฟสการจัดการระบบที่มีอยู่ ใน *Operating system and device management* โลแคล C และ POSIX ได้รับการออกแบบมาเพื่อนำเสนอประสิทธิภาพการทำงานที่ดีที่สุด

LC_ALL
บันทึกทับค่าของตัวแปรสภาพแวดล้อม **LANG** และ ค่าของตัวแปรสภาพแวดล้อม **LC_*** อื่นใด

LC_COLLATE
ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_COLLATE** หมวดหมู่ **LC_COLLATE** กำหนด กฎการจัดเรียงอักขระ หรือการจัดเรียงสตริงที่ควบคุม พฤติกรรมของช่วง คลาสที่เท่าเทียม และองค์ประกอบการจัดเรียง หลายอักขระ

LC_CTYPE
ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_CTYPE** หมวดหมู่ **LC_CTYPE** กำหนดกฎการจัดการอักขระ ที่ควบคุมการตีความลำดับของไบต์ ของอักขระข้อมูลที่เป็นข้อความ (นั่นคืออักขระไบต์เดียวและหลายไบต์) การจัดประเภทของอักขระ (ตัวอย่างเช่น alpha, ตัวเลข และ อื่นๆ) และพฤติกรรมของคลาสอักขระ

LC__FASTMSG
ระบุข้อความดีฟอลต์ที่จะใช้สำหรับโลแคล C และ POSIX และที่ `NLSPATH` จะถูกละเว้นเมื่อมีการตั้งค่า **LC__FASTMSG** เป็น `จริง` มิฉะนั้น จะทำการจัดการข้อความ POSIX compliant ค่าดีฟอลต์จะเป็น `LC__FASTMSG=true` ใน ไฟล์ **/etc/environment**

LC_MESSAGES

ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_MESSAGES** หมวดหมู่ **LC_MESSAGES** กำหนด กฎที่ควบคุมการตอบรับและการตอบปฏิเสธ และ โลแคล (ภาษา) สำหรับข้อความและเมนูต่างๆ

ผู้พัฒนาแอ็พพลิเคชัน ที่บันทึกแอ็พพลิเคชันซึ่งไม่แสดงอักขระหลายไบต์บนเทอร์มินัล ควรตรวจสอบให้แน่ใจว่าค่า **LC_MESSAGES** ไม่มีการตั้งค่า เป็น `C@lft` ถ้าจำเป็น ปิดใช้งานการตั้งค่าโดยใช้รูทีนย่อย **putenv** (**"LC_MESSAGES="**) ผลคือเอาต์พุต ที่ใช้แค็ตตาล็อกข้อความที่แปลง `C@lft` ปิดใช้งานได้ โดยล็อกอินเซสชันที่สามารถแสดงอักขระหลายไบต์ กระบวนการ ที่ออกใช้ซึ่งใช้ **cron** หรือ **inittab** จะเก็บรักษาค่า `C@lft` **LC_MESSAGES** และใช้รูทีนย่อย **setlocale( )** เพื่อสร้างสภาพแวดล้อมภาษา สำหรับข้อความดีฟอลต์

LC_MONETARY

ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_MONETARY** หมวดหมู่ **LC_MONETARY** กำหนด กฎที่ควบคุมการจัดรูปแบบที่เกี่ยวข้องกับการเงิน

LC_NUMERIC

ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_NUMERIC** หมวดหมู่ **LC_NUMERIC** กำหนด กฎที่ควบคุมการจัดรูปแบบตัวเลขที่ไม่ใช่การเงิน

LC_TIME

ระบุโลแคลที่จะใช้สำหรับข้อมูลหมวดหมู่ **LC_TIME** หมวดหมู่ **LC_TIME** กำหนด กฎที่ควบคุมการจัดรูปแบบวันที่และเวลา

LOCPATH

ระบุพาธการค้นหาสำหรับข้อมูลที่โลคัลไลซ์ รวมถึง ไฟล์ไบนารีโลแคล วิธีการอินพุต และตัวแปลงชุดโค้ด

หมายเหตุ: โปรแกรม **setuid** และ **setgid** ทั้งหมด ละเว้นตัวแปรสภาพแวดล้อม **LOCPATH**

NLSPATH

ระบุพาธการค้นหาสำหรับการระบุตำแหน่งไฟล์แค็ตตาล็อกข้อความ ตัวแปรสภาวะแวดล้อมนี้ใช้โดยคอมโพเนนต์ Message Facility ของระบบย่อยการสนับสนุน multicultural ให้ดูรูทีนย่อย **catopen** สำหรับข้อมูลเพิ่มเติม เกี่ยวกับรูปแบบที่คาดไว้ของตัวแปร **NLSPATH**

ตัวแปรสภาพแวดล้อมที่มีผลต่อการเลือกโลแคลสามารถจัดกลุ่มเป็นคลาส ระดับความสำคัญได้สามคลาสดังนี้: สูง ปานกลาง และต่ำ ตัวแปรสภาพแวดล้อม ในคลาสระดับความสำคัญสูงมีดังนี้:

- **LC_ALL**
- **LC_COLLATE**
- **LC_CTYPE**

ตัวแปรสภาพแวดล้อมในคลาสระดับความสำคัญปานกลางมีดังนี้:

- **LC_MESSAGES**
- **LC_MONETARY**
- **LC_NUMERIC**
- **LC_TIME**

ตัวแปรสภาพแวดล้อมในคลาสระดับความสำคัญต่ำมีดังนี้:

- **LANG**

เมื่อโลแคลมีการร้องขอโดยใช้รูทีนย่อย **setlocale** สำหรับหมวดหมู่ เฉพาะหรือสำหรับหมวดหมู่ทั้งหมด การตั้งค่าตัวแปรสภาพแวดล้อม จะมีการเคียวรี่โดยใช้ระดับของระดับความสำคัญในลักษณะ ต่อไปนี้:

- ถ้ามีการตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** หมวดหมู่ทั้งหกหมวด จะใช้โลแคลที่ตัวแปรนั้นระบุ ตัวอย่างเช่น ถ้าตัวแปรสภาพแวดล้อม **LC_ALL** เท่ากับ en_US และตัวแปรสภาพแวดล้อม **LANG** เท่ากับ fr_FR การเรียกไปยังรูทีนย่อย **setlocale** จะตั้งค่าแต่ละหมวดหมู่ของหกหมวดหมู่เป็นโลแคล en_US
- ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** แต่ละหมวดหมู่ จะใช้โลแคลที่ระบุโดยตัวแปรสภาพแวดล้อมที่สอดคล้องกันของ หมวดหมู่ ตัวอย่างเช่น ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** ตัวแปรสภาพแวดล้อม **LC_COLLATE** มีการตั้งค่าเป็น de_DE และตัวแปรสภาพแวดล้อม **LC_TIME** มีการตั้งค่าเป็น fr_CA ผลคือการเรียกไปยังรูทีนย่อย **setlocale** จะตั้งค่าหมวดหมู่ **LC_COLLATE** เป็น de_DE และหมวดหมู่ **LC_TIME** เป็น fr_CA ตัวแปรสภาพแวดล้อมทั้งสองไม่มีความสำคัญเหนือกว่ากันในสถานการณ์ นี้
- ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** และไม่ได้ตั้งค่า สำหรับตัวแปรสภาพแวดล้อม **LC_*** โดยเฉพาะ ค่าของตัวแปรสภาพแวดล้อม **LANG** จะกำหนดการตั้งค่าสำหรับ หมวดหมู่เฉพาะนั้น ตัวอย่างเช่น ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** ตัวแปรสภาพแวดล้อม **LC_CTYPE** มีการตั้งค่า เป็น en_US ไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_NUMERIC** และตัวแปรสภาพแวดล้อม **LANG** มีการตั้งค่าเป็น is_IS ผลคือการเรียกไปยังรูทีนย่อย **setlocale** จะตั้งค่าหมวดหมู่ **LC_CTYPE** เป็น en_US และหมวดหมู่ **LC_NUMERIC** เป็น is_IS ตัวแปรสภาพแวดล้อม **LANG** จะระบุโลแคลสำหรับหมวดหมู่ที่ยังไม่ได้กำหนดก่อนหน้านี้โดยตัวแปรสภาพแวดล้อม LC_* เท่านั้น
- ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** ไม่ได้ตั้งค่า ตัวแปรสภาพแวดล้อม **LC_*** เฉพาะ และไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LANG** โลแคลสำหรับหมวดหมู่เฉพาะนั้น จะมีค่าดีฟอลต์เป็นโลแคล C ตัวอย่างเช่น ถ้าไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL** ตัวแปรสภาพแวดล้อม **LC_MONETARY** มีการตั้งค่าเป็น sv_SE ไม่ได้ตั้งค่าตัวแปรสภาพแวดล้อม **LC_TIME** และไม่ได้ตั้งค่าตัวแปร สภาพแวดล้อม **LANG** ผลคือการเรียกไปยังรูทีนย่อย **setlocale** จะตั้งค่าหมวดหมู่ **LC_MONETARY** เป็น sv_SE และ หมวดหมู่ **LC_TIME** เป็น C

ตัวอย่างของตัวแปรสภาพแวดล้อมก่อนหน้านี้

ตารางต่อไปนี้แสดงการตั้งค่าปัจจุบันของตัวแปรสภาพแวดล้อม และผลกระทบจากการเรียก **setlocale**(**LC_ALL**,"") คอลัมน์สุดท้ายบ่งชี้การตั้งค่าโลแคลหลังจากการเรียก **setlocale**(**LC_ALL**,"")

ตารางที่ 1. ตัวอย่างของตัวแปรสภาพแวดล้อมก่อนหน้านี้

ตัวแปรสภาพแวดล้อมและชื่อหมวดหมู่	ค่าของตัวแปรสภาพแวดล้อม	ค่าของหมวดหมู่หลังเรียกใช้ setlocale (LC_ALL,"")
LC_COLLATE	de_DE	de_DE
LC_CTYPE	de_DE	de_DE
LC_MONETARY	en_US	en_US
LC_NUMERIC	(ไม่ได้กำหนด)	da_DK
LC_TIME	(ไม่ได้กำหนด)	da_DK
LC_MESSAGES	(ไม่ได้กำหนด)	da_DK
LC_ALL	(ไม่ได้กำหนด)	(ใช้ไม่ได้)
LANG	da_DK	(ใช้ไม่ได้)

การทำความเข้าใจกับไฟล์ต้นฉบับคำนิยามโลแคล

ไม่เหมือนกับตัวแปรสภาพแวดล้อมซึ่งสามารถตั้งค่าจาก บรรทัดคำสั่ง แต่โลแคลสามารถดัดแปลงโดยการแก้ไขและการคอมไพล์ไฟล์ต้นฉบับคำนิยามโลแคลเท่านั้น

ถ้าโลแคลที่ต้องการไม่ได้เป็นส่วนหนึ่งของไลบรารี เวอร์ชันไบนารี ของโลแคลสามารถคอมไพล์ได้โดยใช้คำสั่ง **localedef** พฤติกรรมโลแคลของโปรแกรมไม่ได้รับผลกระทบจากไฟล์ต้นฉบับคำนิยาม โลแคล ยกเว้นว่าไฟล์จะถูกแปลงโดยคำสั่ง **localedef** ก่อน และอ็อบเจ็กต์โลแคลพร้อมใช้งานสำหรับโปรแกรม คำสั่ง **localedef** จะแปลง ไฟล์ต้นฉบับที่มีคำนิยามของโลแคลเป็นรูปแบบรันไทม์ และจะคัดลอกเวอร์ชันรันไทม์ของไฟล์ที่ระบุบน บรรทัดคำสั่ง ซึ่งโดยปกติ เป็นชื่อโลแคล คำสั่งที่ถูกทำให้เป็นโกลบอล และรูทีนย่อยสามารถเข้าถึงข้อมูลโลแคล หากต้องการข้อมูล เกี่ยวกับการจัดเตรียมไฟล์ต้นฉบับที่จะแปลงโดยคำสั่ง **localedef** ให้ดู **รูปแบบไฟล์ต้นฉบับคำนิยามโลแคล** ใน *Files Reference*

รูทีนย่อยหลายไบต์

รูทีนย่อยหลายไบต์ประมวลผลอักขระในรูปแบบโค้ดไฟล์ โดยปกติ ชื่อของรูทีนย่อยเหล่านี้ขึ้นต้นด้วยคำเสริมหน้า **mb** อย่างไรก็ตาม รูทีนย่อยหลายไบต์บางรายการไม่มีคำเสริมหน้านี้ ตัว อย่างเช่น รูทีนย่อย **strcoll** และ **strxfrm** ประมวลผลอักขระ ในรูปแบบหลายไบต์แต่ไม่มีคำเสริมหน้า **mb** รูทีนย่อยมาตรฐาน C ต่อไปนี้ดำเนินงานบนไบต์ต่างๆ และสามารถใช้ เพื่อจัดการข้อมูลหลายไบต์ได้: **strcmp**, **strcpy**, **strncmp**, **strncpy**, **strcat**, และ **strncat** รูทีนย่อยการค้นหามาตรฐาน C **strchr**, **strrchr**, **strpbrk**, **strcspn**, **strrchr**, **strspn**, **strstr**, และ **strtok** สามารถใช้ในกรณีดังต่อไปนี้:

- การค้นหาหรือการสแกนอักขระในชุดโค้ดไบต์เดียว
- การค้นหาหรือการสแกนอักขระช่วงจุดโค้ดเฉพาะใน สตริงหลายไบต์

หลักการที่เกี่ยวข้อง:

"ชุดโค้ดสำหรับการสนับสนุน multicultural" ในหน้า 15
ส่วนนี้แนะนำโปรแกรมเมอร์ในการใช้รูทีนย่อยเมื่อ พัฒนาโปรแกรมที่ทำให้เป็นโกลบอลแบบเคลื่อนย้ายได้ ใช้ฟังก์ชัน Open Group, ISO/ANSI C, และ POSIX มาตรฐานเพื่อให้สามารถใช้ได้หลายระบบสูงสุด

รูทีนย่อยอักขระ wide

รูทีนย่อยอักขระ wide ประมวลผลอักขระในรูปแบบโค้ด กระบวนการ โดยปกติ รูทีนย่อยอักขระ wide ขึ้นต้นด้วยคำเสริมหน้า **wc** อย่างไรก็ตาม มีข้อยกเว้นสำหรับกฎนี้ ตัวอย่างเช่น ฟังก์ชัน การจัดประเภทอักขระ wide ใช้คำเสริมหน้า **isw** เพื่อกำหนดว่ารูทีนย่อยเป็นรูทีนย่อยอักขระ wide หรือไม่ ให้ตรวจสอบว่า prototype รูทีนย่อยกำหนดอักขระเป็นชนิดข้อมูล **wchar_t** หรือตัวชี้ข้อมูล **wchar_t** หรือตรวจสอบว่ารูทีนย่อยส่งคืนชนิดข้อมูล **wchar_t** หรือไม่ มีข้อยกเว้นบางอย่างสำหรับกฎนี้ ตัวอย่างเช่น รูทีนย่อย การจัดประเภทอักขระ wide ยอมรับค่าชนิดข้อมูล **wint_t**

หลักการที่เกี่ยวข้อง:

"ชุดโค้ดสำหรับการสนับสนุน multicultural" ในหน้า 15
ส่วนนี้แนะนำโปรแกรมเมอร์ในการใช้รูทีนย่อยเมื่อ พัฒนาโปรแกรมที่ทำให้เป็นโกลบอลแบบเคลื่อนย้ายได้ ใช้ฟังก์ชัน Open Group, ISO/ANSI C, และ POSIX มาตรฐานเพื่อให้สามารถใช้ได้หลายระบบสูงสุด

ความเป็นสองทิศทางและการจัดรูปทรงอักขระ

โปรแกรมที่เป็นโกลบอลอาจต้องจัดการข้อความแบบ สองทิศทาง และการจัดรูปทรงอักขระ

ความเป็นสองทิศทาง (BIDI) เกิดขึ้นเมื่อข้อความที่มีการจัดวางในทิศทาง ที่แตกต่างกันปรากฏขึ้นพร้อมกัน ตัวอย่างเช่น ข้อความภาษาอังกฤษมีการอ่านจาก ซ้ายไปขวา ข้อความภาษาฮิบรูมีการอ่านจากขวาไปซ้าย ถ้าทั้งข้อความภาษาอังกฤษ และข้อความภาษาฮิบรูปรากฏขึ้นบนบรรทัดเดียวกัน ข้อความจะเป็นแบบสองทิศทาง

การจัดรูปทรงอักขระ เกิดขึ้นเมื่อรูปทรงของอักขระ ขึ้นอยู่กับตำแหน่งของอักขระในบรรทัดข้อความ ในบางภาษา เช่น อารบิก อักขระมีรูปทรงที่แตกต่างกันขึ้นอยู่กับ ตำแหน่งของอักขระนั้นในสตริงและอักขระรอบข้าง

หลักการที่เกี่ยวข้อง:

"ภาพรวมของโครงร่าง (การจัดรูปแบบข้อความและอักขระสองทิศทาง)" ในหน้า 191
ข้อความสองทิศทาง (BIDI) เกิดขึ้นเมื่อข้อความที่มีการจัดวางทิศทาง แตกต่างกันปรากฏขึ้นพร้อมกัน ตัวอย่างเช่น ข้อความภาษาอังกฤษมีการอ่านจาก ซ้ายไปขวา ข้อความภาษาอารบิกและฮิบรูมีการอ่านจากขวา ไปซ้าย ถ้าทั้งข้อความภาษาอังกฤษ และข้อความภาษาฮิบรูปรากฏขึ้นบนบรรทัดเดียวกัน ข้อความจะเป็นแบบสองทิศทาง

ความเป็นอิสระของชุดโค้ด

ระบบต้องใช้ข้อมูลบางอย่างเกี่ยวกับชุดโค้ดเพื่อ สื่อสารกับสภาพแวดล้อมภายนอก ข้อมูลนี้ถูกซ่อนไว้โดยรูทีนย่อยไลบรารีที่เป็นอิสระจากชุดโค้ด (ไลบรารี globalization) รูทีนย่อย เหล่านี้ส่งผ่านข้อมูลไปยังฟังก์ชันที่พึ่งพาชุดโค้ด เนื่องจากรูทีนย่อย multicultural จัดการข้อมูลชุดโค้ดที่จำเป็น ดังนั้นคุณจึงไม่ต้องมีความรู้เกี่ยวกับชุดโค้ดอย่างชัดแจ้งเมื่อคุณเขียนโปรแกรมที่ประมวลผลอักขระ เทคนิคการเขียนโปรแกรมนี้เรียกว่า *ความ เป็นอิสระของชุดโค้ด*

หลักการที่เกี่ยวข้อง:

"ตัวอย่างโปรแกรมการสนับสนุน Multicultural" ในหน้า 268
ส่วนนี้มีส่วนของโปรแกรมตัวอย่าง `my_example.c` ซึ่งแสดงการทำให้เป็นโกลบอลผ่านโปรแกรมมิงที่ไม่ขึ้นกับชุดโค้ด

การกำหนดจำนวนไบต์สูงสุดในชุดโค้ด

คุณสามารถใช้แมโคร **MB_CUR_MAX** เพื่อกำหนดจำนวนไบต์ สูงสุดในอักขระหลายไบต์สำหรับชุดโค้ด ในโลแคลปัจจุบัน ค่าของแมโครนี้ขึ้นอยู่กับ การตั้งค่าปัจจุบันของหมวดหมู่ **LC_CTYPE** เนื่องจากโลแคล อาจแตกต่างกันระหว่างกระบวนการต่างๆ ดังนั้นการรันแมโคร **MB_CUR_MAX** ในกระบวนการที่แตกต่างกันหรือในเวลาที่แตกต่างกันอาจทำให้ได้ผลลัพธ์ ที่แตกต่างกันได้ แมโคร **MB_CUR_MAX** มีการกำหนดไว้ในไฟล์ส่วนหัว **stdlib.h**

คุณสามารถใช้แมโคร **MB_LEN_MAX** เพื่อกำหนดจำนวนไบต์สูงสุด ในชุดโค้ดใดๆ ที่ระบบสนับสนุน แมโครนี้มีการกำหนดไว้ในไฟล์ส่วนหัว **limits.h**

การกำหนดความกว้างในการแสดงอักขระและสตริง

แมโคร **_max_disp_width** เป็นแมโครเฉพาะระบบปฏิบัติการ และควรหลีกเลี่ยงการใช้แมโครนั้นในแอ็พพลิเคชันที่ใช้ได้หลายระบบ ถ้าการใช้ได้หลายระบบ ไม่ใช่สิ่งสำคัญ คุณสามารถใช้แมโคร **_max_disp_width** เพื่อ กำหนดจำนวนสูงสุดของคอลัมน์ที่แสดงผลซึ่งเป็นที่ต้องการโดยอักขระ เดี่ยวในชุดโค้ดในโลแคลปัจจุบัน ค่าของแมโครนี้ขึ้นอยู่กับ การตั้งค่าปัจจุบันของหมวดหมู่ **LC_CTYPE** ถ้าค่าของแมโครนี้เป็น 1 (หนึ่ง) อักขระทั้งหมดในชุดโค้ด ปัจจุบันจะต้องการความกว้างของคอลัมน์การแสดงผลเอาต์พุตเพียงคอลัมน์เดียวเท่านั้น

เมื่อทั้ง **MB_CUR_MAX** และ **_max_disp_width** มีการตั้งค่า เป็น 1 (หนึ่ง) คุณสามารถใช้รูทีนย่อย **strlen** เพื่อกำหนด ความกว้างของคอลัมน์แสดงผลที่ต้องใช้สำหรับสตริงได้ เมื่อ **MB_CUR_MAX** มากกว่าหนึ่ง ให้ใช้รูทีนย่อย **wcswidth** เพื่อค้นหาความกว้างของคอลัมน์แสดงผลของสตริง

รูทีนย่อยความกว้างการแสดงผลอักขระ wide **wcswidth** และ **wcwidth** ไม่มีฟังก์ชันหลายไบต์ที่สอดคล้องกัน รูทีนย่อย **wcswidth** ไม่ได้บ่งชี้จำนวนอักขระที่สามารถ แสดงขึ้นได้ในพื้นที่ว่างซึ่งมีอยู่บนจอแสดงผล รูทีนย่อย **wcwidth** มีประโยชน์สำหรับวัตถุประสงค์นี้ รูทีนย่อยนี้ต้องมีการเรียกซ้ำ บนสตริงอักขระ wide เพื่อค้นหาจำนวนอักขระ ที่สามารถแสดงขึ้นได้ในตำแหน่งซึ่งมีอยู่บนจอแสดงผล

ความรู้เกี่ยวกับข้อยกเว้นของชุดโค้ด: ช่วงจุดโค้ดเฉพาะ

เนื่องจากวิธีการจัดระเบียบของชุดโค้ดที่สนับสนุน จึงมีประกาศข้อยกเว้นหนึ่งว่า: "ไม่มีความรู้เกี่ยวกับชุดโค้ดที่เน้น ใดสามารถสมมุติในโปรแกรมได้"

เมื่อค้นหาสตริงอักขระหลายไบต์สำหรับอักขระใดๆ ภายในช่วงจุดโค้ดเฉพาะ (ตัวอย่างเช่น อักขระ . (จุด)) ไม่จำเป็นต้องแปลงสตริงเป็นรูปแบบโค้ดกระบวนการ ซึ่ง เพียงพอต่อการค้นหาอักขระ (.) โดยตรวจสอบไบต์ แต่ละไบต์ ข้อยกเว้นนี้ช่วยให้เคอร์นัลและยูทิลิตี้สามารถค้นหา อักขระพิเศษ และ / ในขณะที่แจงชื่อไฟล์ได้ ถ้าโปรแกรม จะค้นหาอักขระใดๆ ในช่วงจุดโค้ดเฉพาะ ควรจะใช้ฟังก์ชันสตริงมาตรฐานที่ดำเนินงานบนไบต์ (เช่น รูทีนย่อย **strchr**) สำหรับรายการของอักขระในช่วงจุดโค้ดเฉพาะ ให้ดู "อักขระ ASCII" ในหน้า 59

การจับคู่ชื่อไฟล์

POSIX.2 กำหนดรูทีนย่อย **fnmatch** ที่จะใช้สำหรับ การจับคู่ชื่อไฟล์ แอ็พพลิเคชันสามารถใช้รูทีนย่อย **fnmatch** เพื่ออ่านไดเร็กทอรีและประยุกต์ใช้รูปแบบกับแต่ละรายการ ตัวอย่างเช่น ยูทิลิตี้ **ค้นหา** สามารถใช้รูทีนย่อย **fnmatch** ได้ ยูทิลิตี้ **pax** สามารถ ใช้รูทีนย่อย **fnmatch** เพื่อประมวลผลตัวถูกดำเนินการของรูปแบบ แอ็พพลิเคชันที่ต้องจับคู่สตริงในลักษณะคล้ายกันสามารถใช้ รูทีนย่อย **fnmatch** ได้

การจัดการอักขระ radix

หมายเหตุว่าอักขระ radix ดังที่ได้รับโดย **nl_langinfo**(RADIXCHAR) คือ ตัวชี้ไปยังสตริง เป็นไปได้ที่โลแคลอาจระบุอักขระนี้เป็นอักขระหลายไบต์หรือเป็นสตริงของอักขระ อย่างไรก็ตาม ใน AIX มีการกำหนดสมมุติฐาน แบบง่ายว่า RADIXCHAR เป็นอักขระไบต์เดียว

โมเดลการเขียนโปรแกรม

โปรแกรมมิงโมเดลที่แสดงในที่นี้ไฮไลต์การเปลี่ยนแปลง ที่คุณต้องทำเมื่อโปรแกรมที่มีอยู่นั้นถูกทำให้เป็นโกลบอล หรือเมื่อมีโปรแกรมใหม่ถูกปรับใช้:

- ระบุการทำให้เป็นโกลบอลโดยสมบูรณ์ อย่าสมมุติว่าอักขระ มีคุณสมบัติเฉพาะใดๆ กำหนดคุณสมบัติแบบไดนามิก โดยใช้อินเตอร์เฟสที่เหมาะสม อย่าสมมุติคุณสมบัติของ ชุดโค้ด ยกเว้นสำหรับอักขระ ASCII ที่มีจุดโค้ดอยู่ใน ช่วงจุดโค้ดเฉพาะ
- จัดทำความเป็นอิสระจากชุดโค้ดของโปรแกรม โปรแกรมไม่ควรสมมุติ การเข้ารหัสไบต์เดียว สองไบต์ หรือหลายไบต์ของการเรียงลำดับใดๆ ข้อมูล สามารถถูกประมวลผลในรูปแบบโค้ดกระบวนการหรือโค้ดไฟล์ อย่างใดอย่างหนึ่ง โดยใช้รูทีนย่อยที่เหมาะสม
- นำเสนอการโต้ตอบกับเคอร์นัลในรูปแบบโค้ดไฟล์อย่างเดียว เคอร์นัล ไม่ได้จัดการโค้ดกระบวนการ
- ไลบรารีรูทีนย่อยการสนับสนุน multicultural สามารถจัดการกับการประมวลผลที่ใช้โค้ดไฟล์ และการประมวลผลที่ใช้โค้ดกระบวนการ

  หมายเหตุ: รูทีนย่อยหลายรายการ ที่ใช้รูปแบบโค้ดกระบวนการไม่มีรูทีนย่อยที่ใช้รูปแบบรูปแบบโค้ดไฟล์ ซึ่งสอดคล้องกัน เนื่องจากอสมมาตรนี้ จึงจำเป็นต้องแปลงสตริง เป็นรูปแบบโค้ดกระบวนการและเรียกใช้รูทีนย่อยโค้ดกระบวนการที่เหมาะสม

- บางไลบรารีอาจไม่มีการประมวลผลในรูปแบบโค้ดกระบวนการ แอ็พพลิเคชันที่ต้องการไลบรารีเหล่านี้ต้องใช้โค้ดไฟล์เมื่อเรียกใช้ ฟังก์ชันจากไลบรารีนั้น
- โปรแกรมสามารถเข้าถึงอักขระในรูปแบบโค้ดกระบวนการหรือ รูปแบบโค้ดไฟล์ อย่างใดอย่างหนึ่ง สามารถบันทึกโปรแกรมที่เป็นอิสระจากชุดโค้ด โดยใช้ได้ทั้งสองวิธีการ

ชุดโค้ดสำหรับการสนับสนุน multicultural

ส่วนนี้แนะนำโปรแกรมเมอร์ในการใช้รูทีนย่อยเมื่อ พัฒนาโปรแกรมที่ทำให้เป็นโกลบอลแบบเคลื่อนย้ายได้ ใช้ฟังก์ชัน Open Group, ISO/ANSI C, และ POSIX มาตรฐานเพื่อให้สามารถใช้ได้หลายระบบสูงสุด

หมายเหตุ: อย่าใช้รูทีนย่อยโครงร่างในไลบรารี **libi18n.a** ยกเว้นว่าแอ็พพลิเคชันกำลังทำการแสดงชนิดการบริการ แอ็พพลิเคชัน ส่วนใหญ่จัดการกับข้อความที่จัดลำดับตามตรรกะ

หลักการที่เกี่ยวข้อง:

"รูทีนย่อยหลายไบต์" ในหน้า 12

รูทีนย่อยหลายไบต์ประมวลผลอักขระในรูปแบบโค้ดไฟล์ โดยปกติ ชื่อของรูทีนย่อยเหล่านี้ขึ้นต้นด้วยคำเสริมหน้า **mb** อย่างไรก็ตาม รูทีนย่อยหลายไบต์บางรายการไม่มีคำเสริมหน้านี้ ตัว อย่างเช่น รูทีนย่อย **strcoll** และ **strxfrm** ประมวลผลอักขระ ในรูปแบบหลายไบต์แต่ไม่มีคำเสริมหน้า **mb** รูทีนย่อยมาตรฐาน C ต่อไปนี้ดำเนินงานบนไบต์ต่างๆ และสามารถใช้ เพื่อจัดการข้อมูลหลายไบต์ได้: **strcmp**, **strcpy**, **strncmp**, **strncpy**, **strcat**, และ **strncat** รูทีนย่อยการค้นหามาตรฐาน C **strchr**, **strrchr**, **strpbrk**, **strcspn**, **strrchr**, **strspn**, **strstr**, และ **strtok** สามารถใช้ในกรณีดังต่อไปนี้:

"รูทีนย่อยอักขระ wide" ในหน้า 12

รูทีนย่อยอักขระ wide ประมวลผลอักขระในรูปแบบโค้ด กระบวนการ โดยปกติ รูทีนย่อยอักขระ wide ขึ้นต้นด้วยคำเสริมหน้า **wc** อย่างไรก็ตาม มีข้อยกเว้นสำหรับกฎนี้ ตัวอย่างเช่น ฟังก์ชัน การจัดประเภทอักขระ wide ใช้คำเสริมหน้า **isw** เพื่อกำหนดว่ารูทีนย่อยเป็นรูทีนย่อยอักขระ wide หรือไม่ ให้ตรวจสอบว่า prototype รูทีนย่อยกำหนดอักขระเป็นชนิดข้อมูล **wchar_t** หรือตัวชี้ข้อมูล **wchar_t** หรือตรวจสอบว่ารูทีนย่อยส่งคืนชนิดข้อมูล **wchar_t** หรือไม่ มีข้อยกเว้นบางอย่างสำหรับกฎนี้ ตัวอย่างเช่น รูทีนย่อย การจัดประเภทอักขระ wide ยอมรับค่าชนิดข้อมูล **wint_t**

"รายการของรูทีนการสนับสนุน multicultural" ในหน้า 214

รูทีนย่อยการสนับสนุน multicultural ใช้สำหรับการจัดการข้อมูลเฉพาะโลแคล ดำเนินการกับอักขระแบบกว้าง และอักขระหลายไบต์ และการใช้นิพจน์ธรรมดา

"รายการของรูทีนย่อยนิพจน์ธรรมดา" ในหน้า 218

รูทีนย่อยโลแคล

โปรแกรมที่ทำการประมวลผลโดยอาศัยโลแคล รวมถึง ข้อความผู้ใช้ ต้องเรียกรูทีนย่อย **setlocale** ที่ตอนต้น ของโปรแกรม การเรียกนี้เป็นคำสั่งแรกที่ดำเนินการได้ในโปรแกรม **หลัก** โปรแกรมที่ไม่ได้เรียกรูทีนย่อย **setlocale** ในวิธีนี้ จะสืบทอดโลแคล C หรือ POSIX โปรแกรมดังกล่าวดำเนินการดังเช่นในโลแคล C โดยไม่คำนึงถึงการตั้งค่าของตัวแปรสภาพแวดล้อม **LC_*** และ **LANG**

มีการนำเสนอรูทีนย่อยอื่นๆ เพื่อกำหนดการตั้งค่าปัจจุบัน สำหรับการจัดรูปแบบข้อมูลโลแคล

โลแคลของกระบวนการจะกำหนดวิธีจัดการจัดเรียงอักขระ การจัดประเภทอักขระ การจัดรูปแบบวันที่และเวลา การวรรคตัวเลข การวรรคการเงิน และเอาต์พุตข้อความ ส่วนต่อไปนี้อธิบายวิธีการตั้งค่าและเข้าถึงข้อมูลเกี่ยวกับโลแคลปัจจุบันในโปรแกรม โดยใช้การสนับสนุน multicultural

หลักการที่เกี่ยวข้อง:

"การตั้งค่าโลแคล"

โปรแกรมที่ทำให้เป็นโกลบอลทุกโปรแกรมต้องตั้งค่าโลแคลปัจจุบันโดยใช้ รูทีนย่อย **setlocale** รูทีนย่อยนี้ช่วยให้กระบวนการสามารถเปลี่ยนหรือเคียวรี่โลแคลปัจจุบันโดยการเข้าถึงฐานข้อมูลโลแคล

การตั้งค่าโลแคล

โปรแกรมที่ทำให้เป็นโกลบอลทุกโปรแกรมต้องตั้งค่าโลแคลปัจจุบันโดยใช้ รูทีนย่อย **setlocale** รูทีนย่อยนี้ช่วยให้กระบวนการสามารถเปลี่ยนหรือเคียวรี่โลแคลปัจจุบันโดยการเข้าถึงฐานข้อมูลโลแคล

เมื่อกระบวนการเริ่มต้นขึ้น โลแคลปัจจุบันของกระบวนการมีการตั้งค่าเป็นโลแคล C หรือ POSIX โปรแกรมที่พึ่งพาข้อมูลโลแคลซึ่งไม่มีการกำหนดไว้ในโลแคล C หรือ POSIX ต้องเรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้ก่อนการใช้ข้อมูลเฉพาะโลแคลใดๆ:

```
setlocale(LC_ALL, "");
```

หลักการที่เกี่ยวข้อง:

"รูทีนย่อยโลแคล" ในหน้า 15

โปรแกรมที่ทำการประมวลผลโดยอาศัยโลแคล รวมถึง ข้อความผู้ใช้ ต้องเรียกรูทีนย่อย **setlocale** ที่ตอนต้น ของโปรแกรม การเรียกนี้เป็นคำสั่งแรกที่ดำเนินการได้ในโปรแกรม **หลัก** โปรแกรมที่ไม่ได้เรียกรูทีนย่อย **setlocale** ในวิธีนี้ จะสืบทอดโลแคล C หรือ POSIX โปรแกรมดังกล่าวดำเนินการดังเช่นในโลแคล C โดยไม่คำนึงถึงการตั้งค่าของตัวแปรสภาพแวดล้อม **LC_*** และ **LANG**

"รายการตรวจสอบการดำเนินงานโปรแกรม" ในหน้า 207

การเข้าถึงข้อมูลโลแคล

รูทีนย่อยต่อไปนี้ช่วยให้สามารถเข้าถึงข้อมูล ที่กำหนดไว้ในโลแคลปัจจุบันตามที่กำหนดโดยการเรียกล่าสุด ไปยังรูทีนย่อย **setlocale**:

localeconv

ช่วยให้เข้าถึงข้อมูลโลแคลที่กำหนดไว้ในหมวดหมู่ **LC_MONETARY** และ **LC_NUMERIC** ของโลแคลปัจจุบัน รูทีนย่อย **localeconv** จะดึงข้อมูล เกี่ยวกับหมวดหมู่เหล่านี้ วางข้อมูลนั้นไว้ในโครงสร้าง ชนิด **lconv** ตามที่กำหนดไว้ในไฟล์ **locale.h** และจะส่งคืน ตัวชี้ไปยังโครงสร้างนี้

nl_langinfo

ส่งคืนตัวชี้ไปยังสตริงที่ยุติด้วย null ซึ่งมีข้อมูล ที่กำหนดไว้ในหมวดหมู่ **LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC,** และ **LC_TIME** ของโลแคลปัจจุบัน

rpmatch

ทดสอบการตอบสนองเชิงบวกและเชิงลบ ซึ่งมีการระบุ อยู่ในหมวดหมู่ **LC_MESSAGES** ของโลแคลปัจจุบัน การตอบสนอง อาจเป็นนิพจน์ปกติและสตริงแบบง่าย เนื่องจากรูทีนย่อย **rpmatch** ไม่ใช่รูทีนย่อยตามมาตรฐานอุตสาหกรรม ดังนั้นแอ็พพลิเคชันที่ใช้ได้หลายระบบ จึงอาจไม่มีรูทีนย่อยนี้

รูทีนย่อย **localeconv** และ **nl_langinfo** ไม่ได้ช่วยให้ เข้าถึงหมวดหมู่ **LC_*** ทั้งหมด

สามารถดูการตั้งค่าโลแคลปัจจุบันได้จาก: **setlocale**(*หมวดหมู่*, (char*)0) ค่าที่ส่งคืนคือสตริงที่ระบุโลแคลปัจจุบันสำหรับ *หมวดหมู่* ตัวอย่างต่อไปนี้กำหนดการตั้งค่าโลแคลปัจจุบันสำหรับหมวดหมู่ **LC_CTYPE**:

```
char *ctype_locale; ctype_locale = setlocale(LC_CTYPE, (char*)0);
```

ตัวอย่าง

ส่วนนี้มีตัวอย่างของรูทีนย่อย

- ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **setlocale** เพื่อเปลี่ยนโลแคล จากโลแคล C ดีฟอลต์เป็นโลแคลที่ระบุโดยตัวแปรสภาพแวดล้อม เพื่อให้สอดคล้องกับลำดับชั้นของตัวแปรสภาพแวดล้อม โลแคล:

```
#include <locale.h>
main()
{

    char *p;

    p = setlocale(LC_ALL, "");

   /*
    **  โปรแกรมจะมีโลแคลตามที่ตั้งค่าโดยตัวแปร
    **  LC_* และ LANG
    */
 }
```

- ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **setlocale** เพื่อให้ได้การตั้งค่าโลแคลปัจจุบัน สำหรับหมวดหมู่ **LC_COLLATE**:

```
#include <stdio.h>
#include <locale.h>

main()
{
    char *p;

    /*  ตั้งค่าโลแคลปัจจุบันสำหรับสิ่งที่ระบุ  */
    p = setlocale(LC_ALL, "");
    /*  การตั้งค่าโลแคลปัจจุบันสำหรับหมวดหมู่
    **  ทั้งหมดมีการบ่งชี้โดย p
    */

    /*
    **  ค้นหาการตั้งค่าปัจจุบันสำหรับ
    **  หมวดหมู่ LC_COLLATE
    */
    p = setlocale(LC_COLLATE, NULL);
    /*
    **  p ชี้ไปยังสตริงที่มีการตั้งค่าโลแคลปัจจุบัน
    **  สำหรับหมวดหมู่ LC_COLLATE
    */

 }
```

- ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย **setlocale** เพื่อให้ได้การตั้งค่าโลแคลปัจจุบัน และบันทึกไว้สำหรับการใช้งานในภายหลัง การดำเนินการนี้ทำให้โปรแกรม สามารถเปลี่ยนโลแคลเป็นโลแคลใหม่ชั่วคราวได้ หลังจากการประมวลผล เสร็จสมบูรณ์แล้ว โลแคลสามารถกลับไปยังสถานะดั้งเดิมได้

```
#include <stdio.h>
#include <locale.h>
#include <string.h>

#define NEW_LOCALE "MY_LOCALE"

main()
{
    char *p, *save_locale;

    p = setlocale(LC_ALL, "");
    /*
    **  เริ่มต้นโลแคล p ชี้ไปยังการตั้งค่าโลแคลปัจจุบัน
    **  สำหรับหมวดหมู่ทั้งหมด
    */

    save_locale = (char *)malloc(strlen(p) +1);
    strcpy(save_locale, p);
        /*  บันทึกการตั้งค่าโลแคลปัจจุบัน */
    p = setlocale(LC_ALL, NEW_LOCALE);
        /* เปลี่ยนเป็นโลแคลใหม่ */

    /*
    **  กำลังประมวลผล  ...
    */

     /*  เปลี่ยนกลับเป็นโลแคลเก่า  */
    p = setlocale(LC_ALL, save_locale);  /* คืนสภาพโลแคลเก่า */

     free(save_locale);    /* ทำให้หน่วยความจำว่าง */
}
```

- **ตัวอย่างต่อไปนี้ใช้รูทีนย่อย setlocale เพื่อตั้งค่าหมวดหมู่ LC_MESSAGES ให้กับโลแคลที่กำหนด โดยตัวแปรสภาพแวดล้อม หมวดหมู่อื่นทั้งหมดยังคงมีการตั้งค่าให้กับโลแคล C**

```
#include <locale.h>

main()
{
    char *p;

    /*
    **  โปรแกรมจะเริ่มต้นในโลแคล  C  สำหรับหมวดหมู่ทั้งหมด
    */

    p = setlocale(LC_MESSAGES, "");

     /*
    **  ณ เวลานี้ LC_COLLATE, LC_CTYPE, LC_NUMERIC,
    **  LC_MONETARY, LC_TIME จะอยู่ในไลแคล C
    **  LC_MESSAGES จะถูกตั้งค่าเป็นการตั้งค่าโลแคลปัจจุบัน
    **  ตามที่กำหนดโดยตัวแปรสภาพแวดล้อม
    */
}
```

- **ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย localeconv เพื่อให้ได้การตั้งค่าจุดทศนิยม สำหรับโลแคลปัจจุบัน:**

```
#include <locale.h>

main()
{
        struct lconv *ptr;
        char *decimal;

        (void)setlocale(LC_ALL, "");
        ptr = localeconv();
        /*
        ** เข้าถึงข้อมูลที่ได้รับ ตัวอย่างเช่น
        ** ได้รับการตั้งค่าจุดทศนิยมปัจจุบัน
        */
        decimal = ptr->decimal_point;
}
```

- ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย **nl_langinfo** เพื่อให้ได้รูปแบบวันที่ และเวลาสำหรับโลแคลปัจจุบัน:

```
#include <langinfo.h>
#include <locale.h>
main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");
    ptr = nl_langinfo(D_T_FMT);
}
```

- ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย **nl_langinfo** เพื่อให้ได้อักขระ radix สำหรับโลแคลปัจจุบัน:

```
#include <langinfo.h>
#include <locale.h>

main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");  /* ตั้งค่าโลแคลของโปรแกรม  */
    ptr = nl_langinfo(RADIXCHAR); /* ได้รับอักขระ radix*/
}
```

- ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย **nl_langinfo** เพื่อให้ได้การตั้งค่า ของสัญลักษณ์สกุลเงินสำหรับโลแคลปัจจุบัน:

```
#include <langinfo.h>
#include <locale.h>

main()
{
    char *ptr;
    (void)setlocale(LC_ALL, "");  /* ตั้งค่าโลแคลของโปรแกรม  */
    ptr = nl_langinfo(CRNCYSTR);  /* ได้รับสตริงสกุลเงิน*/
    /* สตริงสกุลเงินจะเป็น "-$" ในโลแคลสหรัฐฯ */
}
```

- ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย rpmatch เพื่อให้ได้การตั้งค่าสตริงการตอบรับและการตอบปฏิเสธ สำหรับโลแคลปัจจุบัน:

  การตอบรับ และการตอบปฏิเสธดังระบุอยู่ในฐานข้อมูลโลแคล จะไม่ใช่สตริงแบบง่ายอีกต่อไป โดยอาจเป็นนิพจน์ธรรมดา ตัว อย่างเช่น yesexpr อาจเป็นนิพจน์ธรรมดาต่อไปนี้ ซึ่งจะ ยอมรับตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กของ y ตามด้วย เลขศูนย์หรืออักขระตัวอักษรเพิ่มเติม หรืออักขระ 0 ตามด้วย K ดังนั้น yesexpr อาจเป็นนิพจน์ธรรมดา ต่อไปนี้:

```
([yY][:alpha:]*|OK)
```

มาตรฐาน ไม่มีรูทีนย่อยที่จะดึงข้อมูลและเปรียบเทียบข้อมูล นี้ คุณสามารถใช้รูทีนย่อย AIX-specific **rpmatch(const char *response)**

```
#include <stdio.h>
#include <langinfo.h>
#include <locale.h>
#include <regex.h>

int rpmatch(const char *);
    /*
    **  ส่งคืน 1 ถ้าการตอบกลับเป็น ใช่, 0 ถ้าการตอบกลับเป็น ไม่ใช่
    ** มิฉะนั้นจะส่งคืน -1
    */

main()
{
    int  ret;
    char *resp;

(void)setlocale(LC_ALL, "");

    do {
        /*
        ** ได้รับการตอบกลับเคียวรี่เป็น ใช่/ไม่มีสตริง
        ** ตัวชี้สตริง resp ชี้ไปยังการตอบกลับนี้
        ** ตรวจสอบว่าสตริงเป็น ใช่ หรือไม่
        */
    ret = rpmatch(resp);

    if(ret == 1){
        /* การตอบกลับเป็นใช่ */
        /* ประมวลผลตามนั้น */
        }else if(ret == 0){
        /* การตอบกลับเป็นไม่ใช่ */
        /* ประมวลผลการตอบปฏิเสธ */
        }else if(ret<0){
        /* ไม่ตรงกับใช่/ไม่ใช่ */
        continue;
        }
    }while(ret <0);
}
```

- ตัวอย่างต่อไปนี้แสดงวิธีดำเนินการรูทีนย่อย **rpmatch** หมายเหตุว่าแอ็พพลิเคชันส่วนใหญ่ควรจะใช้รูทีนย่อย **rpmatch** ใน **libc** การดำเนินการต่อไปนี้ของรูทีนย่อย **rpmatch** ใช้เพื่อ การสาธิตเท่านั้น

  หมายเหตุว่า **nl_langinfo(YESEXPR)** และ **nl_langinfo(NOEXPR)** ใช้เพื่อให้ได้นิพจน์ธรรมดาสำหรับการตอบรับและการตอบปฏิเสธ ตามลำดับ

```
#include <langinfo.h>
#include <regex.h>
/*
** rpmatch() ทำการเปรียบเทียบสตริงกับนิพจน์ธรรมดา
** โดยใช้ POSIX.2 ที่กำหนดฟังก์ชันการคอมไพล์และการจับคู่นิพจน์ธรรมดา
** อาร์กิวเมนต์แรกคือการตอบกลับจากผู้ใช้และ
** สตริงที่สองคือการตั้งค่าโลแคลปัจจุบันของนิพจน์ธรรมดา
*/
```

```
int rpmatch( const char *string)



{
     int status;
     int retval;
     regex_t re;
     char *pattern;

     pattern = nl_langinfo(YESEXPR);
     /* คอมไพล์นิพจน์ธรรมดาที่ชี้โดยรูปแบบ */
     if( ( status = regcomp( &re, pattern, REG_EXTENDED | REG_NOSUB )) != 0 ){
          retval = -2; /*-2 บ่งชี้ข้อผิดพลาดการคอมไพล์นิพจน์ yes */
          return(retval);
     }
     /* จับคู่สตริงกับนิพจน์ธรรมดาที่คอมไพล์ */
     status = regexec( &re, string, (size_t)0, (regmatch_t *)NULL, 0);
     if(status == 0){
          retval = 1;   /* พบการตรงกับ Yes */
     }else{    /* ตรวจสอบการตอบปฏิเสธ */
          pattern = nl_langinfo(NOEXPR);
          if( ( status = regcomp( &re, pattern,
               REG_EXTENDED | REG_NOSUB )) != 0 ){
                    retval = -3;/*-3 บ่งชี้ว่าไม่มีข้อผิดพลาดการคอมไพล์ */
                    return(retval);
               }
          status = regexec( &re, string, (size_t)0,
                         (regmatch_t *)NULL, 0);
          if(status == 0)
               retval = 0;/* พบการตรงกับการตอบปฏิเสธ */
     }else
               retval = -1; /* สตริงไม่ตรงกับการตอบกลับ ใช่ หรือ
                              ไม่ใช่ */
     regfree(&re);
     return(retval);
}
```

รูทีนย่อยการจัดรูปแบบเวลา

โปรแกรมที่ต้องจัดรูปแบบเวลาเป็นสตริงโค้ดอักขระ wide สามารถใช้รูทีนย่อย **wcsftime** ได้ โปรแกรมที่ต้องแปลงสตริงแบบมัลติไบต์เป็นรูปแบบเวลาภายในสามารถใช้รูทีนย่อย **strptime** ได้

นอกเหนือจากรูทีนย่อย **strftime** ที่กำหนดในมาตรฐานภาษาการเขียนโปรแกรม C แล้ว X/Open Portability Guide Issue 4 กำหนดรูทีนย่อยการจัดรูปแบบเวลาดังต่อไปนี้:

wcsftime
: จัดรูปแบบเวลาเป็นสตริงโค้ดอักขระ wide

strptime
: แปลงสตริงแบบมัลติไบต์เป็นรูปแบบเวลาภายใน

ตัวอย่าง

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcsftime** เพื่อจัดรูปแบบเวลาเป็นสตริงอักขระ wide:

มักจะเริ่มต้นด้วยคำนิยามที่ตอบคำถาม "สิ่งนี้คือ อะไร?"

```
#include <stdio.h>
#include <langinfo.h>
#include <locale.h>
#include <time.h>

main()
{
        wchar_t timebuf[BUFSIZE];
        time_t clock = time( (time_t*) NULL);
        struct tim *tmptr = localetime(&clock);

        (void)setlocale(LC_ALL, "");

        wcsftime(
                timebuf,        /* บัฟเฟอร์เอาต์พุตสตริงเวลา */
                BUFSIZ,         /*ขนาดสูงสุดของเอาต์พุตสตริง */
                nl_langinfo(D_T_FMT),       /* รูปแบบวันที่/เวลา */
                tmptr           /* ตัวชี้ไปยังโครงสร้าง tm */
        );

        printf("%S\n", timebuf);
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย strptime เพื่อแปลง สตริงเวลาที่จัดรูปแบบเป็นรูปแบบภายใน:

```
#include <langinfo.h>
#include <locale.h>
#include <time.h>

main(int argc, char **argv)
{
        struct tm tm;

        (void)setlocale(LC_ALL, "");

        if (argc != 2) {
                ...                 /* Error handling */
        }
        if (strptime(
                argv[1],            /* สตริงเวลาที่จัดรูปแบบ */
                nl_langinfo(D_T_FMT),       /* รูปแบบวันที่/เวลา */
                &tm                 /* ที่อยู่ของโครงสร้าง tm */
        ) == NULL) {
                ...                 /* Error handling */
        }
        else {
                ...                /* Other Processing */
        }
}
```

รูทีนย่อยการจัดรูปแบบเงิน

โปรแกรมที่ต้องระบุ หรือเข้าถึงปริมาณทางการเงิน สามารถเรียกรูทีนย่อย strfmon

แม้ว่ามาตรฐานการเขียนโปรแกรมภาษา C ด้วย POSIX จัดเตรียม สื่อสำหรับการระบุและการเข้าถึงข้อมูลด้านการเงิน มาตรฐานเหล่านี้ ไม่ได้กำหนดรูทีนย่อยที่จะจัดรูปแบบปริมาณทางการเงิน รูทีนย่อย XPG4 strfmon นำเสนอฟังก์ชันการจัดรูปแบบปริมาณทางการเงิน ไม่มีรูทีนย่อยที่กำหนดใดจะแปลงสตริงการเงินที่จัดรูปแบบเป็นปริมาณตัวเลขที่เหมาะสมสำหรับการคำนวณ แอ็พพลิเคชันที่ต้องคำนวณปริมาณทางการเงิน สามารถดำเนินการได้หลังจากการประมวลผลสตริงทางการเงินที่ขึ้นกับโลแคล ให้เป็นตัวเลข ข้อมูลการจัดรูปแบบเงินเฉพาะวัฒนธรรม มีการระบุโดยหมวดหมู่ **LC_MONETARY** แอ็พพลิเคชันสามารถรับข้อมูลที่เกี่ยวข้องกับรูปแบบทางการเงิน และสัญลักษณ์สกุลเงินโดยการเรียกรูทีนย่อย localeconv

หมายเหตุ: โมดิฟายเออร์ @euro และ @preeuro ไม่จำเป็นสำหรับโลแคล POSIX

ตัวอย่าง

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **strfmon** และยอมรับข้อมูลจำเพาะเกี่ยวกับรูปแบบและค่าอินพุต ค่าอินพุต มีการจัดรูปแบบตามข้อมูลจำเพาะเกี่ยวกับรูปแบบอินพุต

```
#include <monetary.h>
#include <locale.h>
#include <stdio.h>

main(int argc, char **argv)
{
        char bfr[256], format[256];
        int match; ssize_t size;
        float value;

        (void) setlocale(LC_ALL, "");

        if (argc != 3){
                ...        /* Error handling */
        }
        match = sscanf(argv[1], "%f", &value);
        if (!match) {
                ...        /* Error handling */
        }
        match = sscanf(argv[2], "%s", format);
        if (!match) {
                ...        /*Error handling */
        }
        size = strfmon(bfr, 256, format, value);
        if (size == -1) {
                ...        /* Error handling */
        }
        printf ("Formatted monetary value is: %s\n", bfr);
}
```

ตารางต่อไปนี้จัดเตรียมตัวอย่างของข้อมูลจำเพาะการแปลงที่เป็นไปได้ และเอาต์พุตสำหรับ 12345.67 และ -12345.67 ในโลแคลภาษาอังกฤษ สหรัฐอเมริกา:

ข้อมูลจำเพาะเกี่ยวกับการแปลง	เอาต์พุต	เอาต์พุต
%n	$12,345.67 -$12,345.67$12	การจัดรูปแบบดีฟอลต์
%15n	$12,345.67 -$12,345.67	จัดชิดด้านขวาภายในฟิลด์ที่มี 15 อักขระ
%#6n	$ 12,345.67 -$ 12,345.67	จัดวางคอลัมน์สำหรับค่าสูงสุดถึง 999,999
%=*#8n	$****12,345.67 -$****12,345.67	ระบุอักขระที่กรอกข้อมูล
%=#8n	$000012,345.67 -$000012,345.67	อักขระที่กรอกข้อมูลไม่ใช้การแบ่งกลุ่ม
%^#6n	$ 12345.67 -$ 12345.67	ปิดใช้งานตัวแบ่งหลักพัน
%^#6.0n	12346 -$ 12346	ปัดเศษลงเป็นหน่วยทั้งหมด
%^#6.3n	$ 12345.670 -$ 12345.670	เพิ่มความแม่นยำ
%(#6n	$ 12,345.67 ($ 12,345.67)	ใช้สไตล์การเปลี่ยนเป็นค่าบวกหรือค่าลบได้
%!(#6n	12,345.67 (12,345.67)	ปิดใช้งานสัญลักษณ์สกุลเงิน

ตัวอย่างต่อไปนี้แปลงค่าเงินเป็นค่า ตัวเลข สตริงการเงินมีการบ่งชี้โดยใช้ อินพุต และผลลัพธ์ของการแปลงสตริงดังกล่าวเป็นรูปแบบตัวเลขจะถูกจัดเก็บไว้ในสตริง ที่มีการบ่งชี้โดยใช้ เอาต์พุต สมมุติว่า อินพุต และ เอาต์พุต มีการเริ่มต้นขึ้น

```
char *input;   /* สตริงอินพุตแบบหลายไบต์ที่มีสตริงการเงิน */
char *output;  /* สตริงตัวเลขที่ได้จากสตริงอินพุต */
wchar_t src_string[SIZE], dest_string[SIZE];
wchar_t *monetary, *numeric;
wchar_t mon_decimal_point, radixchar;
wchar_t wc;
localeconv *lc;

/* เริ่มต้นอินพุตและเอาต์พุตเพื่อชี้ถึงบัฟเฟอร์ที่ถูกต้องตามความเหมาะสม */
/* แปลงสตริงอินพุตเป็นรูปแบบโค้ดกระบวนการ*/
retval = mbstowcs(src_string, input, SIZE);
/* จัดการการส่งคืนข้อผิดพลาด */

monetary = src_string;
numeric = dest_string;
lc = localeconv();
        /* ได้ข้อมูล LC_MONETARY และ LC_NUMERIC */

/* แปลงจุดทศนิยมทางการเงินเป็นรูปแบบ wide char */
retval = mbtowc( &mon_decimal_point, lc->mon_decimal_point,
                 MB_CUR_MAX);
/* จัดการข้อผิดพลาดเกี่ยวกับตัวพิมพ์ใดๆ */

/* แปลงจุดทศนิยมทางตัวเลขเป็นรูปแบบ wide char */
retval = mbtowc( &radixchar, lc->decimal_point, MB_CUR_MAX);
/* จัดการข้อผิดพลาดเกี่ยวกับตัวพิมพ์ */
/* สมมุติว่าสตริงถูกแปลงเป็น wide character ก่อนอื่น
** รูปแบบโค้ดผ่านทาง mbstowcs, จุดทางการเงินเป็นสตริงนี้
*/
/* เลือกข้อมูลตัวเลขจาก wide character
** สตริง และคัดลอกไปยังบัฟเฟอร์ temp
*/
```

```
        while(wc = *monetary++){
                if(iswdigit(wc))
                        *numeric++ = wc;
                else if( wc == mon_decimal_point)
                        *numeric++=radixchar;
        }
        *numeric = 0;
/* dest_string มีค่าตัวเลขของปริมาณทางการเงิน */
/* แปลงปริมาณทางตัวเลขเป็นรูปแบบหลายไบต์ */
retval = wcstombs( output, dest_string, SIZE);
/* จัดการการส่งคืนข้อผิดพลาดใดๆ */
/* เอาต์พุตมีค่าตัวเลขที่เหมาะสมสำหรับการแปลง atof */
```

รูทีนย่อยอักขระหลายไบต์และอักขระ wide

การแสดงถึงข้อมูลแบบภายนอกเรียกว่าการแสดงถึง *โค้ด ไฟล์* ของอักขระ เมื่อข้อมูลโค้ดไฟล์ถูกสร้างขึ้น ในไฟล์หรือแปลงระหว่างคอมพิวเตอร์และอุปกรณ์ I/O อักขระเดียวอาจมีการแสดงถึงโดยใช้หนึ่งหรือหลายไบต์ สำหรับการประมวลผล สตริงของอักขระดังกล่าว การแปลงโค้ดเหล่านี้เป็นการแสดงถึง ความยาวเดียวกันเป็นสิ่งที่มีประสิทธิภาพมากกว่า รูปแบบที่แปลงนี้มีไว้สำหรับการประมวลผลอักขระเป็นการภายในเท่านั้น การแสดงถึงข้อมูล แบบภายในเรียกว่าการแสดงถึง *โค้ดกระบวนการ* หรือ *โค้ดอักขระ wide* ของอักขระ

การสนับสนุนหลายวัฒนธรรมของโปรแกรมเป็นการผสมของรูทีนย่อยอักขระ แบบมัลติไบต์ และแบบกว้าง รูทีนย่อยแบบ *มัลติไบต์* ใช้ชุดอักขระแบบมัลติไบต์ รูทีนย่อย *อักขระ wide* ใช้ชุดอักขระ wide รูทีนย่อยหลายไบต์มีคำเสริมหน้า **mb** รูทีนย่อยอักขระ wide มีคำเสริมหน้า wc รูทีนย่อย การจัดการสตริงที่สอดคล้องกันมีการบ่งชี้โดยคำเสริมหน้า **mbs** และ **wcs** ตามลำดับ การตัดสินใจว่าจะใช้รูทีนย่อยอักขระแบบมัลติไบต์หรืออักขระ wide เมื่อใดสามารถทำได้หลังจากการวิเคราะห์อย่างรอบคอบแล้วเท่านั้น

รูทีนย่อยการแปลงโค้ดแบบมัลติไบต์และโค้ดอักขระ

สภาวะแวดล้อมที่เป็นโกลบอลของการสนับสนุนหลายวัฒนธรรมผสม รูทีนย่อยอักขระแบบมัลติไบต์ และ wide การตัดสินใจว่าจะใช้รูทีนย่อยอักขระ wide หรืออักขระ แบบมัลติไบต์เมื่อใดสามารถทำได้หลังจากการวิเคราะห์อย่างรอบคอบแล้วเท่านั้น

ถ้าโปรแกรมใช้รูทีนย่อยแบบมัลติไบต์เป็นหลัก อาจจำเป็นต้อง แปลงโค้ดอักขระแบบมัลติไบต์เป็นโค้ดอักขระ wide ก่อนจะสามารถใช้ รูทีนย่อยโค้ด wide บางรายการได้ ถ้าโปรแกรม ใช้รูทีนย่อยอักขระ wide ข้อมูลอาจต้องมีการแปลงเป็นรูปแบบมัลติไบต์เมื่อเรียกใช้รูทีนย่อย ทั้งสองวิธีมีข้อดีข้อเสีย ขึ้นอยู่กับโปรแกรมที่ใช้อยู่และการมีอยู่ของรูทีนย่อยมาตรฐาน ที่ทำการประมวลผลที่ต้องการ ตัวอย่างเช่น รูทีนย่อยความกว้างคอลัมน์ในการแสดงผลอักขระ wide ไม่มีรูทีนย่อยแบบมัลติไบต์ มาตรฐานที่สอดคล้องกัน

ถ้าโปรแกรมสามารถประมวลผลอักขระของโปรแกรมในรูปแบบมัลติไบต์ได้ ควร จะใช้วิธีการนี้แทนการแปลงอักขระเป็นรูปแบบอักขระ wide

ข้อควรสนใจ: การแปลงระหว่างโค้ดอักขระแบบมัลติเป็นอักขระ wide ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน อย่าแลกเปลี่ยนโค้ดอักขระ wide ระหว่างสองกระบวนการ ยกเว้นคุณมีความรู้ว่า แต่ละโลแคลที่จะใช้จัดการโค้ดอักขระ wide ในลักษณะที่สอดคล้องกัน ด้วยข้อยกเว้นของโลแคลตามชุดโค้ด IBM-eucTW โลแคล AIX ใช้ค่าอักขระ Unicode เป็นโค้ดอักขระ wide

รูทีนย่อยการแปลงอักขระหลายไบต์เป็นโค้ดอักขระ wide:

รูทีนย่อยต่อไปนี้ใช้เมื่อแปลงจากโค้ดแบบมัลติไบต์เป็นโค้ดอักขระ wide:

mblen กำหนดความยาวของอักขระแบบมัลติไบต์ อย่าใช้ p++ เพื่อเพิ่มตัวชี้ในสตริงแบบมัลติไบต์ ใช้รูทีนย่อย **mblen** เพื่อกำหนดจำนวนของ ไบต์ที่เรียบเรียงอักขระ

mbstowcs
แปลงสตริงแบบมัลติไบต์เป็นสตริงอักขระ wide

mbtowc
แปลงอักขระแบบมัลติไบต์เป็นอักขระ wide

รูทีนย่อยการแปลงโค้ดอักขระ wide เป็นโค้ดแบบมัลติไบต์:

รูทีนย่อยต่อไปนี้ใช้เมื่อแปลงจากโค้ดอักขระ wide เป็นโค้ดอักขระแบบมัลติไบต์:

wcslen กำหนดจำนวนของอักขระ wide ในสตริงอักขระ wide

wcstombs
แปลงสตริงอักขระ wide เป็นสตริงอักขระแบบมัลติไบต์

wctomb
แปลงอักขระ wide เป็นอักขระแบบมัลติไบต์

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **mbtowc** เพื่อแปลงอักขระในโค้ดอักขระหลายไบต์เป็นโค้ดอักขระ wide:

```
main()
{
    char     *s;
    wchar_t  wc;
    int      n;

    (void)setlocale(LC_ALL,"");

    /*
    **  s ชี้ไปยังสตริงอักขระที่จะถูก
    **  แปลงเป็นอักขระ wide ซึ่งจะจัดเก็บไว้ใน wc
    */
    n = mbtowc(&wc, s, MB_CUR_MAX);

    if (n == -1){
        /*  การจัดการข้อผิดพลาด  */
    }
    if (n == 0){
        /*  กรณีของชื่อที่ชี้ไปยัง null  */
    }

    /*
    **  wc มีโค้ดกระบวนการสำหรับอักขระหลายไบต์
    **  ที่ชี้โดย s
    */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wctomb** เพื่อแปลงอักขระ ในโค้ดอักขระ wide เป็นโค้ดอักขระหลายไบต์:

```
main()
{
    char     *s;
    wchar_t  wc;
    int      n;

    (void)setlocale(LC_ALL,"");

    /*
    **  s ชี้ไปยังสตริงอักขระที่จะถูก
    **  แปลงเป็นอักขระ wide ซึ่งจะจัดเก็บไว้ใน wc
    */
    n = mbtowc(&wc, s, MB_CUR_MAX);

    if (n == -1){
        /*  การจัดการข้อผิดพลาด  */
    }
    if (n == 0){
        /*  ตัวพิมพ์ของชื่อที่ชี้ไปยัง null  */
    }

    /*
    **  wc มีโค้ดกระบวนการสำหรับอักขระหลายไบต์
    **  ที่ชี้โดย s
    */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย mblen เพื่อ ค้นหาความยาวไบต์ของอักขระในโค้ดอักขระหลายไบต์:

```
#include <stdlib.h>
#include <locale.h>

main
{
    char *name = "h";
    int  n;

    (void)setlocale(LC_ALL,"");

    n = mblen(name, MB_CUR_MAX);
    /*
    **  จำนวนนับที่ส่งคืนใน n คือความยาวหลายไบต์
    **  ค่านี้น้อยกว่าหรือเท่ากับค่าของ
    **  MB_CUR_MAX ใน stdlib.h เสมอ
    */
    if (n == -1){
        /* การจัดการข้อผิดพลาด */
    }
}
```

ตัวอย่าง ต่อไปนี้มีตำแหน่งอักขระก่อนหน้านี้ในสตริง หลายไบต์ ถ้าคุณต้องการกำหนดตำแหน่งอักขระก่อนหน้านี้โดยเริ่มต้นจากตำแหน่งอักขระปัจจุบัน (ไม่ใช่ตำแหน่งไบต์แบบสุ่ม) ไปยังบัฟเฟอร์ที่เริ่มต้นเมื่อเริ่มทำงาน ใช้รูทีนย่อย mblen จนกว่าจะถึงตำแหน่งอักขระปัจจุบัน และบันทึกตำแหน่งอักขระ ก่อนหน้านี้เพื่อให้ได้ตำแหน่งอักขระที่ต้องการ

```
char buf[];     /* มีสตริงหลายไบต์ */
char *cur,      /* ชี้ไปยังตำแหน่งอักขระปัจจุบัน */
char *prev,     /* ชี้ไปยังอักขระหลายไบต์ก่อนหน้านี้ */
char *p;        /* ตัวชี้ที่เคลื่อนที่ */

/* เริ่มต้นบัฟเฟอร์และตัวชี้ตามต้องการ */
/* ลูปผ่านบัฟเฟอร์จนกว่าตัวชี้ที่เคลื่อนที่จะไปถึง
** ตำแหน่งอักขระปัจจุบันในบัฟเฟอร์ โดยบันทึก
** ตำแหน่งอักขระล่าสุดไว้ในตัวชี้ prev เสมอ */
p = prev = buf;

/* cur ชี้ไปยังอักขระที่ถูกต้องที่ไหนสักแห่งใน buf */
while(p< cur){
        prev = p;
        if( (i=mblen(p, mbcurmax))<=0){
                /* อักขระหลายไบต์ที่ไม่ถูกต้องหรือ null */
                /* คุณอาจมีกลยุทธ์ในการจัดการข้อผิดพลาดที่แตกต่าง
                ** กันได้ */
                p++;    /* ข้ามไป */
        }else {
                p += i;
        }
}
/* prev จะชี้ไปยังตำแหน่งอักขระก่อนหน้านี้ */

/* หมายเหตุว่าถ้า ( prev == cur) แสดงว่า
** ไม่มีอักขระก่อนหน้านี้ และถ้าไบต์ทั้งหมดจนถึง
** อักขระปัจจุบันไม่ถูกต้อง ระบบจะจัดการเป็น
** อักขระไบต์เดียวที่ถูกต้องทั้งหมด และนี่ไม่ใช่สิ่งที่
** คุณต้องการ ผู้ใช้อาจเปลี่ยนกรณีนี้เพื่อจัดการกู้คืนข้อผิดพลาด
** ด้วยวิธีการอื่น */
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย mbstowcs เพื่อแปลงสตริงหลายไบต์เป็นสตริงอักขระ wide:

```
#include <stdlib.h>
#include <locale.h>

main()
{
     char    *s;
     wchar_t *pwcs;
     size_t  retval, n;

     (void)setlocale(LC_ALL, "");

     n = strlen(s) + 1;          /* ความยาวสตริง + การยุติ null */

     /*  จัดสรรอาร์เรย์ wchar ที่ต้องการ    */
     pwcs = (wchar_t *)malloc(n * sizeof(wchar_t) );
     retval = mbstowcs(pwcs, s, n);
     if(retval == -1){

     /*  การจัดการข้อผิดพลาด  */
          }
```

```
        /*
        ** pwcs มีสตริงอักขระ wide
        */
}
```

ตัวอย่างต่อไปนี้สาธิตปัญหาเกี่ยวกับ การใช้รูทีนย่อย **mbstowcs** บนบล็อกข้อมูลขนาดใหญ่ สำหรับการแปลงเป็นรูปแบบอักขระ wide เมื่อระบบพบหลายไบต์ ที่ไม่ถูกต้อง รูทีนย่อย **mbstowcs** จะส่งคืนค่า -1 แต่ไม่ได้ระบุตำแหน่งที่ข้อผิดพลาดเกิดขึ้น ดังนั้นจึงต้องใช้รูทีนย่อย **mbtowc** ซ้ำเพื่อ แปลงอักขระเป็นโค้ดอักขระ wide ทีละหนึ่งตัว

หมายเหตุ: การประมวลผลในลักษณะนี้อาจทำให้ประสิทธิภาพของโปรแกรมช้าลงอย่างมาก

ในระหว่างการแปลงชุดโค้ดไบต์เดียว เป็นไปไม่ได้ที่จะแปลง หลายไบต์เป็นบางส่วน อย่างไรก็ตาม ในระหว่างการแปลงชุดโค้ดหลายไบต์ หลายไบต์บางส่วนจะถูกบันทึกไว้ในบัฟเฟอร์ที่ปลอดภัย ในระหว่าง การเรียกไปยังรูทีนย่อย **read** ครั้งถัดไป หลายไบต์บางส่วน จะมีการเสริมหน้าในส่วนที่เหลือของลำดับไบต์

หมายเหตุ: และได้สตริงอักขระ wide ที่ยุติ null สามารถทำ การจัดการข้อผิดพลาดที่เป็นอ็อพชันได้ถ้าพบอินสแตนซ์ของลำดับไบต์ ที่ไม่ถูกต้อง

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main(int argc, char *argv[])
{

    char     *curp, *cure;
    int      bytesread, bytestoconvert, leftover;
    int      invalid_multibyte, mbcnt, wcnt;
    wchar_t  *pwcs;
    wchar_t  wbuf[BUFSIZ+1];
    char     buf[BUFSIZ+1];
    char     savebuf[MB_LEN_MAX];
    size_t   mb_cur_max;
    int      fd;
        /*
        **  MB_LEN_MAX ระบุค่าคงที่ wide ระบบสำหรับ
        **  จำนวนไบต์สูงสุดในอักขระหลายไบต์
        */

    (void)setlocale(LC_ALL, "");
    mb_cur_max = MB_CUR_MAX;

    fd = open(argv[1], 0);
    if(fd < 0){
       /*  การจัดการข้อผิดพลาด  */
    }

    leftover = 0;
    if(mb_cur_max==1){    /*  ตัวพิมพ์ชุดโค้ดไบต์เดียว  */
        for(;;){
            bytesread = read(fd, buf, BUSIZ);
            if(bytesread <= 0)
                break;
```

```
            mbstowcs(wbuf, buf, bytesread+1);
            /*  กระบวนการที่ใช้บัฟเฟอร์อักขระ wide   */
        }
                        /*  ไฟล์ที่ประมวลผลแล้ว ...   */
        exit(0);        /*  สิ้นสุดโปรแกรม       */

    }else{              /*  ชุดโค้ดหลายไบต์  */
        leftover = 0;

        for(;;) {
            if(leftover)
                strncpy(buf, savebuf ,leftover);
            bytesread=read(fd,buf+leftover, BUFSIZ-leftover);
            if(bytesread <= 0)
                break;

            buf[leftover+bytesread] = '\0';
                     /* สตริงที่ยุติ Null */
            invalid_multibyte = 0;
            bytestoconvert = leftover+bytesread;
            cure= buf+bytestoconvert;
            leftover=0;
            pwcs = wbuf;
                /* หยุดการประมวลผลเมื่อพบ mbyte ที่ไม่ถูกต้อง */
            curp= buf;

            for(;curp<cure;){
                mbcnt = mbtowc(pwcs,curp, mb_cur_max);
                if(mbcnt>0){
                    curp += mbcnt;
                    pwcs++;
                    continue;

                }else{
                    /* ต้องการข้อมูลเพิ่มเติมในการอ่านครั้งถัดไป*/
                    if ( cure-curp<mb_cur_max){
                        leftover=cure-curp;
                        strncpy(savebuf,curp,leftover);
                        /* Null ยุติก่อนหลายไบต์บางส่วน */
                        *curp=0;
                        break;

                    }else{
                            /*พบหลายไบต์ที่ไม่ถูกต้อง */
                        invalid_multibyte =1;
                        break;
                    }
                }
            }
            if(invalid_multibyte){          /*การจัดการข้อผิดพลาด */
            }
            /* ประมวลผลบัฟเฟอร์อักขระ wide */
        }
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcstombs** และ **wcslen** เพื่อแปลงสตริงอักขระ wide เป็นรูปแบบหลายไบต์:

```
#include <stdlib.h>
#include <locale.h>

main()
{
    wchar_t *pwcs; /* สตริงอักขระ wide ต้นฉบับ */
    char *s;       /* สตริงอักขระหลายไบต์ปลายทาง */
    size_t n;
    size_t retval;

    (void)setlocale(LC_ALL, "");
    /*
    ** คำนวณจำนวนไบต์สูงสุดที่ต้องใช้เพื่อ
    ** จัดเก็บบัฟเฟอร์อักขระ wide ในรูปแบบหลายไบต์ในหน้า
    ** โค้ดปัจจุบันและ malloc() หน่วยจัดเก็บที่เหมาะสม
    ** รวมถึง null ที่ยุติ
    */
    s = (char *) malloc( wcslen(pwcs) * MB_CUR_MAX + 1 );
    retval= wcstombs( s, pwcs, n);
    if( retval == -1) {
        /*  การจัดการข้อผิดพลาด  */
        /* s ชี้ไปยังสตริงอักขระหลายไบต์ */
}
```

รูทีนย่อยการจัดประเภทอักขระ wide

รูทีนย่อยการจัดประเภทอักขระ wide ส่วนใหญ่ คล้ายกับรูทีนย่อยการจัดประเภทอักขระดั้งเดิม ยกเว้นว่ารูทีนย่อยการจัดประเภทอักขระ wide ดำเนินงานอาร์กิวเมนต์ชนิดข้อมูล **wchar_t** ที่ส่งผ่านเป็นอาร์กิวเมนต์ชนิดข้อมูล **wint_t**

หลักการที่เกี่ยวข้อง:

“รายการตรวจสอบการดำเนินงานโปรแกรม” ในหน้า 207

รูทีนย่อยการจัดประเภทอักขระ wide ทั่วไป:

ในสภาวะแวดล้อมความเป็นสากล คุณต้องมีความสามารถสร้างคุณสมบัติคลาสอักขระใหม่ ตัวอย่างเช่น มีการกำหนดคุณสมบัติหลายอย่างสำหรับอักขระภาษาญี่ปุ่น ซึ่งใช้ไม่ได้กับภาษาอังกฤษ เมื่อมีการสนับสนุน ภาษามากขึ้น เฟรมเวิร์คที่ช่วยให้แอ็พพลิเคชันสามารถจัดการกับคุณสมบัติอักขระ จำนวนมากที่แตกต่างกันได้นับเป็นสิ่งที่จำเป็น

รูทีนย่อย **wctype** และ **iswctype** ช่วยในการจัดการ คลาสอักขระในลักษณะทั่วไป รูทีนย่อยเหล่านี้ใช้ เพื่อทำให้ใช้คลาสอักขระที่ผู้ใช้กำหนดและคลาสอักขระเฉพาะภาษาได้

เมื่อต้องการสร้างการจัดประเภทอักขระใหม่สำหรับใช้กับรูทีนย่อย **wctype** และ **iswctype** ให้สร้างคลาสอักขระใหม่ในหมวดหมู่ **LC_CTYPE** และสร้างโลแคลโดยใช้คำสั่ง **localedef** แอ็พพลิเคชันผู้ใช้จะได้รับข้อมูลโลแคลนี้โดยใช้รูทีนย่อย **setlocale** จากนั้น โปรแกรม สามารถเข้าถึงรูทีนย่อยการจัดประเภทใหม่ได้โดยใช้รูทีนย่อย **wctype** เพื่อเรียกใช้การจัดการคุณสมบัติ **wctype_t** จากนั้นผ่านไปยังรูทีนย่อย **iswctype** ซึ่งจะทดสอบการจัดการคุณสมบัติและโค้ดอักขระ wide ของ อักขระ

รูทีนย่อยต่อไปนี้ใช้สำหรับการจัดประเภทอักขระ wide:

wctype ได้รับการจัดการสำหรับการจัดประเภทคุณสมบัติอักขระ

iswctype

ทดสอบคุณสมบัติอักขระ

รูทีนย่อยการจัดประเภทอักขระ wide มาตรฐาน:

รูทีนย่อย **isw*** กำหนดแง่มุมหลากหลายด้าน ของการจัดประเภทอักขระ wide มาตรฐาน รูทีนย่อย **isw*** ยังทำงานกับชุดโค้ดไบต์เดียวด้วย ใช้รูทีนย่อย **isw*** ก่อนหน้ารูทีนย่อย **wctype** และ **iswctype** ใช้รูทีนย่อย **wctype** และ **iswctype** เฉพาะสำหรับคุณสมบัติ คลาสอักขระแบบขยายเท่านั้น (ตัวอย่างเช่น คุณสมบัติภาษาญี่ปุ่น)

เมื่อใช้ฟังก์ชันอักขระ wide เพื่อแปลงตัวอักษรใน หลายบล็อกของข้อมูล แอ็พพลิเคชันต้องแปลงอักขระจาก รูปแบบโค้ดแบบมัลติไบต์เป็นอักขระ wide เนื่องจากการทำเช่นนี้อาจส่งผลกระทบต่อประสิทธิภาพ ในโลแคลชุดโค้ดไบต์เดียว ให้พิจารณาการใช้พาธการแปลงสองพาธ ในแอ็พพลิเคชันของคุณ พาธดั้งเดิมสำหรับโลแคลชุดโค้ดไบต์เดียว จะแปลงตัวอักษรโดยใช้รูทีนย่อย **isupper**,**islower**, **toupper**, และ **tolower** พาธอื่นสำหรับโลแคลชุดโค้ดแบบมัลติไบต์ จะแปลงอักขระหลายไบต์เป็นรูปแบบโค้ดอักขระ wide และแปลงตัวอักษรโดยใช้รูทีนย่อย **iswupper**, **iswlower**, **towupper** และ **towlower** เมื่อแปลงอักขระแบบมัลติไบต์เป็นรูปแบบโค้ดอักขระ wide แอ็พพลิเคชันต้องจัดการตัวอักษรพิเศษ ซึ่งอักขระแบบมัลติไบต์อาจแบ่งบนบล็อกที่ต่อเนื่องกัน

ข้อมูลต่อไปนี้เป็นรายการของรูทีนย่อยการจัดประเภทอักขระ wide มาตรฐาน:

iswalnum

ทดสอบการจัดประเภท อักขระตัวอักษรและตัวเลข

iswcntrl

ทดสอบการจัดประเภท อักขระตัวอักษร

iswalpha

ทดสอบการจัดประเภท อักขระควบคุม

iswdigit

ทดสอบการจัดประเภท อักขระตัวเลข

iswgraph

ทดสอบการจัดประเภท อักขระกราฟิก

iswlower

ทดสอบการจัดประเภท อักขระตัวพิมพ์เล็ก

iswprint

ทดสอบการจัดประเภท อักขระที่พิมพ์ได้

iswpunct

ทดสอบการจัดประเภท อักขระวรรคตอน

iswspace

ทดสอบการจัดประเภทอักขระพื้นที่ว่าง

iswupper

ทดสอบการจัดประเภท อักขระตัวพิมพ์ใหญ่

iswxdigit

ทดสอบการจัดประเภท อักขระตัวเลขฐานสิบหก

รูทีนย่อยการแปลงตัวพิมพ์อักขระ wide:

รูทีนย่อยต่อไปนี้ใช้สำหรับการแปลงตัวพิมพ์สำหรับอักขระ wide การดำเนินการของการแปลงตัวพิมพ์อักขระ wide ขึ้นอยู่กับคำนิยามในหมวดหมู่ **LC_CTYPE** ของโลแคล ปัจจุบัน

towlower

แปลงอักขระ wide ตัวพิมพ์ใหญ่เป็นอักขระ wide ตัวพิมพ์เล็ก

towupper

แปลงอักขระ wide ตัวพิมพ์เล็กเป็นอักขระ wide ตัวพิมพ์ใหญ่

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wctype** เพื่อทดสอบ การจัดประเภทอักขระ **NEW_CLASS**:

```
#include <ctype.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t    wc;
    int       retval;
    wctype_t  chandle;

    (void)setlocale(LC_ALL,"");
    /*
    ** ได้รับการจัดการคุณสมบัติอักขระสำหรับคุณสมบัติ NEW_CLASS
    **
    */
    chandle = wctype("NEW_CLASS") ;
    if(chandle == (wctype_t)0){
        /* คุณสมบัติไม่ถูกต้อง การจัดการข้อผิดพลาด */
    }
    /* ให้ wc เป็นโค้ดอักขระ wide สำหรับอักขระ */
    /* ทดสอบว่า wc มีคุณสมบัติ NEW_CLASS หรือไม่ */
    retval = iswctype( wc, chandle );
    if( retval > 0 ) {
        /*
        ** wc มีคุณสมบัติ NEW_CLASS
        */
    }else if(retval == 0) {
        /*
        ** อักขระที่แสดงโดย wc ไม่มี
        ** คุณสมบัติ NEW_CLASS
        */
    }
}
```

รูทีนย่อยความกว้างคอลัมน์ในการแสดงผลอักขระ wide

เมื่อแสดงหรือพิมพ์อักขระ จำนวนของคอลัมน์ที่ครอบครอง โดยอักขระหนึ่งอาจแตกต่างกัน ตัวอย่างเช่น อักขระ Kanji (ภาษาญี่ปุ่น) ตัวหนึ่งอาจครอบครองตำแหน่งมากกว่าหนึ่งคอลัมน์ จำนวนของคอลัมน์ที่แสดงผลที่ต้องการโดยแต่ละอักขระเป็นส่วนหนึ่ง ของฐานข้อมูลโลแคลการสนับสนุน multicultural หมวดหมู่ **LC_CTYPE** กำหนด จำนวนของคอลัมน์ที่ต้องใช้ในการแสดงอักขระ

ไม่มีรูทีนย่อยความกว้างคอลัมน์ในการแสดงผลหลายไบต์มาตรฐาน เพื่อให้ใช้ได้หลายระบบ ให้แปลงโค้ดแบบมัลติไบต์เป็นโค้ดอักขระ wide และใช้รูทีนย่อยความกว้างในการแสดงอักขระ wide ที่ต้องการ อย่างไรก็ตาม ถ้าแมโคร **__max_disp_width** (กำหนดไว้ในไฟล์ **stdlib.h**) มีการตั้งค่าเป็น 1 และใช้ชุดโค้ดไบต์เดียวอยู่ ผลคือความกว้างคอลัมน์ในการแสดงผล ของอักขระทั้งหมด (ยกเว้น tabs) ในชุดโค้ดจะเหมือนกัน และเท่ากับ 1 ในกรณีนี้ รูทีนย่อย **strlen (***string***)** จะให้ความกว้างคอลัมน์ในการแสดงผลของสตริงที่ระบุ ดังแสดง ในตัวอย่างต่อไปนี้ :

```
#include <stdlib.h>
        int display_column_width;  /* จำนวนของคอลัมน์ที่แสดง  */
        char  *s;                  /*  สตริงอักขระ            */
        ....
        if((MB_CUR_MAX  ==  1)  &&  (__max_disp_width  ==  1)){
                display_column_width  =  strlen(s);
                                   /*  s  เป็นตัวชี้สตริง    */
        }
```

รูทีนย่อยต่อไปนี้ค้นหาความกว้างในการแสดงผลสำหรับสตริงอักขระ wide:

wcswidth
กำหนดความกว้างในการแสดงผลของสตริงอักขระ wide

wcwidth
กำหนดความกว้างในการแสดงผลของอักขระ wide

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcwidth** เพื่อ ค้นหาความกว้างคอลัมน์ในการแสดงผลของอักขระ wide:

```
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wint_t  wc;
    int     retval;

    (void)setlocale(LC_ALL,  "");

    /*
    **    ให้ wc เป็นอักขระ wide ที่เป็นเจ้าของความกว้างในการแสดงผลที่
    **    จะพบ
    */
    retval  =  wcwidth(wc);
    if(retval  ==  -1){
        /*
        **  การจัดการข้อผิดพลาด อักขระ wide
```

```
        **  ที่ไม่ถูกต้องหรือพิมพ์ไม่ได้ใน wc
        */
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcswidth** เพื่อ ค้นหาความกว้างคอลัมน์ในการแสดงผลของสตริงอักขระ wide:

```
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wchar_t  *pwcs;
    int      retval;
    size_t   n;

    (void)setlocale(LC_ALL,  "");
    /*
    **    ให้ pwcs ชี้ไปยังสตริงอักขระ wide
    **    ที่ยุติด้วย null
    **    ให้ n เป็นจำนวนของอักขระ wide
    **    ที่เป็นเจ้าของความกว้างคอลัมน์ในการแสดงผลที่จะกำหนด
    */
    retval  =  wcswidth(pwcs,  n);
    if(retval  ==  -1){
        /*
        **  การจัดการข้อผิดพลาด พบโค้ดอักขระ wide ที่ไม่ถูกต้องหรือพิมพ์ไม่ได้
        **  ในสตริงอักขระ wide
        **  pwcs
        */
    }
}
```

รูทีนย่อยการจัดเรียงสตริงอักขระหลายไบต์และอักขระ wide

สตริงสามารถเปรียบเทียบได้ในวิธีต่อไปนี้:

- การใช้ค่าจัดลำดับ (ไบนารี) ของอักขระ
- การใช้น้ำหนักที่เชื่อมโยงกับอักขระของแต่ละโลแคล ตามที่กำหนดโดยหมวดหมู่ **LC_COLLATE**

การสนับสนุน Multicultural ใช้วิธีที่สอง

การจัดเรียงเป็นคุณสมบัติเฉพาะโลแคลของอักขระ น้ำหนัก มีการกำหนดให้กับแต่ละอักขระเพื่อบ่งชี้ลำดับความสัมพันธ์ของอักขระนั้นสำหรับ การเรียงลำดับ อักขระหนึ่งอาจได้รับการกำหนดมากกว่าหนึ่งน้ำหนักได้ น้ำหนัก มีการจัดระดับความสำคัญเป็นหลัก รอง ที่สาม และที่สี่ ระบบจะกำหนด จำนวนสูงสุดของน้ำหนักที่สามารถกำหนดให้กับแต่ละอักขระ

กระบวนการสืบทอดโลแคล C หรือโลแคล POSIX ณ เวลาสตาร์ท อัพ เมื่อเรียกรูทีนย่อย **setlocale (LC_ALL, " ")** กระบวนการจะได้รับโลแคลตามข้อมูลตัวแปรสภาพแวดล้อม **LC_*** และ **LANG** รูทีนย่อยต่อไปนี้ได้รับผลกระทบจากหมวดหมู่ **LC_COLLATE** และกำหนดวิธีการเรียงลำดับสองสตริงในโลแคลที่กำหนดใดๆ

หมายเหตุ: การเปรียบเทียบสตริงโดยใช้การจัดเรียงต้องใช้เวลานาน เนื่องจาก การประมวลผลเกี่ยวข้องกับการได้รับค่าการจัดเรียง ดังนั้นจึงควร ทำการเปรียบเทียบดังกล่าวเมื่อจำเป็นเท่านั้น ถ้าคุณต้องการกำหนดว่าสตริงอักขระ wide สองรายการเท่ากันหรือไม่ อย่าใช้รูทีนย่อย **wcscoll** และ **wcsxfrm** ให้ใช้รูทีนย่อย **wcscmp** แทน

รูทีนย่อยสำหรับการเปรียบเทียบสตริงอักขระหลายไบต์มีดังต่อไปนี้:

strcoll เปรียบเทียบน้ำหนักการจัดเรียงของสตริงอักขระหลายไบต์

strxfrm
แปลงสตริงอักขระหลายไบต์เป็นค่าที่แสดงถึงน้ำหนักการจัดเรียง อักขระ

รูทีนย่อยสำหรับการเปรียบเทียบสตริงอักขระ wide มีดังต่อไปนี้:

wcscoll เปรียบเทียบน้ำหนักการจัดเรียงของสตริงอักขระ wide

wcsxfrm
แปลงสตริงอักขระ wide เป็นค่าที่แสดงถึงน้ำหนักการจัดเรียง อักขระ

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcscoll** เพื่อเปรียบเทียบสตริงอักขระ wide สองรายการโดยใช้น้ำหนักการจัดเรียง:

```
#include  <stdio.h>
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

extern  int  errno;

main()
{
    wchar_t  *pwcs1, *pwcs2;
    size_t   n;

    (void)setlocale(LC_ALL,  "");

    /*    ตั้งค่าเป็นศูนย์สำหรับการตรวจสอบข้อผิดพลาดบน wcscoll    */
    errno  =  0;
    /*
    **    ให้ pwcs1 และ pwcs2 เป็นสองสตริงอักขระ wide
    **    ที่จะเปรียบเทียบ
    */
    n  =  wcscoll(pwcs1, pwcs2);
        /*
        **    ถ้ามีการตั้งค่า errno แสดงว่ามี
        **    ข้อผิดพลาดการจัดเรียงบางอย่าง
        */
    if(errno  !=  0){
        /*  เกิดข้อผิดพลาดขึ้น...ข้อผิดพลาดการจัดการ ...*/
    }
}
```

ตัวอย่าง ต่อไปนี้ใช้รูทีนย่อย **wcsxfrm** เพื่อเปรียบเทียบสตริงอักขระ wide สองรายการโดยใช้น้ำหนักการจัดเรียง:

หมายเหตุ: การกำหนดขนาด *n* (โดยที่ *n* คือตัวเลข) ของ สตริงที่แปลงเมื่อใช้รูทีนย่อย **wcsxfrm** สามารถทำได้โดยใช้วิธีอย่างใดอย่างหนึ่งต่อไปนี้:

- สำหรับแต่ละอักขระในสตริงอักขระ wide จำนวนของไบต์สำหรับ ค่าการจัดเรียงที่ใช้ได้ต้องไม่เกินกว่าค่า **COLL_WEIGHTS_MAX * sizeof(wchar_t)** เมื่อนำค่านี้ไปคูณกับจำนวนของโค้ดอักขระ wide จะทำให้ได้ค่าความยาวบัฟเฟอร์ที่ต้องการ ให้บวก 1 ในความยาวบัฟเฟอร์สำหรับ null การยุติอักขระ wide กลยุทธ์นี้ อาจทำให้ประสิทธิภาพการทำงานช้าลง
- ประเมินความยาวไบต์ที่ต้องการ ถ้าค่าที่ได้รับก่อนหน้านี้ไม่เพียงพอ ให้เพิ่มค่า ซึ่งอาจไม่ตอบสนองสตริงทั้งหมดแต่ให้ประสิทธิภาพสูงสุด
- เรียกรูทีนย่อย **wcsxfrm** สองครั้ง: ครั้งแรกเพื่อค้นหาค่าของ *n* และครั้งที่สองเพื่อแปลงสตริงโดยใช้ค่า *n* นี้ กลยุทธ์นี้ทำให้ประสิทธิภาพการทำงานช้าลงเนื่องจากมีการเรียกรูทีนย่อย **wcsxfrm** สองครั้ง อย่างไรก็ตาม วิธีการนี้ให้ค่าที่แม่นยำของขนาดบัฟเฟอร์ ที่ต้องการสำหรับการจัดเก็บสตริงที่แปลง
-

วิธีการที่คุณเลือกขึ้นอยู่กับลักษณะของสตริงที่ใช้ในโปรแกรมและวัตถุประสงค์ประสิทธิภาพของโปรแกรม

```
#include  <stdio.h>
#include  <string.h>
#include  <locale.h>
#include  <stdlib.h>

main()
{
    wchar_t  *pwcs1, *pwcs2, *pwcs3, *pwcs4;
    size_t   n, retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้สตริงที่ชี้โดย pwcs1 และ pwcs3 เป็น
    **  อาร์เรย์อักขระ wide ที่จัดเก็บสตริงอักขระ wide
    **  ที่แปลง ให้สตริงที่ชี้โดย pwcs2
    **  และ pwcs4 เป็นสตริงอักขระ wide ที่จะเปรียบเทียบโดยใช้
    **  ค่าการจัดเรียงของอักขระ wide ในสตริง
    **  เหล่านี้
    **  ให้ n ใหญ่เพียงพอ (นั่นคือ BUFSIZ) ที่จะแปลงสอง
    **  สตริงอักขระ wide ซึ่งระบุโดย pwcs2 และ pwcs4
    **
    **  หมายเหตุ:
    **  ในทางปฏิบัติ สิ่งที่ดีที่สุดคือการเรียก wcsxfrm ถ้าสตริงอักขระ wide
    **  จะถูกเปรียบเทียบกับสตริงอักขระ wide ที่แตกต่างอื่น
    **  หลายครั้ง
    */

    do {
        retval = wcsxfrm(pwcs1, pwcs2, n);
        if(retval == (size_t)-1){
            /*  เกิดข้อผิดพลาดขึ้น  */
            /*  ประมวลผลข้อผิดพลาดถ้าจำเป็น  */
            break;
        }

        if(retval >= n ){
```

```
        /*
        ** เพิ่มค่าของ n และใช้บัฟเฟอร์ที่ใหญ่ขึ้น pwcs1
        */
        }
    }while (retval >= n);

    do {
        retval = wcsxfrm(pwcs3, pwcs4, n);
        if (retval == (size_t)-1){
            /*  เกิดข้อผิดพลาดขึ้น  */
            /*  ประมวลผลข้อผิดพลาดถ้าจำเป็น  */
            break;

        if(retval >= n){
        /*เพิ่มค่าของ n และใช้บัฟเฟอร์ที่ใหญ่ขึ้น pwcs3*/
        }
    }while (retval >= n);
    retval = wcscmp(pwcs1, pwcs3);
    /*  retval มีผลลัพธ์  */
}
```

รูทีนย่อยการเปรียบเทียบสตริงอักขระหลายไบต์และอักขระ wide

รูทีนย่อย **strcmp** และ **strncmp** กำหนดว่า เนื้อหาของสองสตริงหลายไบต์เท่าเทียมกันหรือไม่ ถ้าแอ็พพลิเคชัน ของคุณต้องการทราบว่าสองสตริงมีศัพท์ที่แตกต่างกันอย่างไร ให้ใช้รูทีนย่อย การจัดเรียงสตริงอักขระหลายไบต์และอักขระ wide

รูทีนย่อยการสนับสนุน multicultural ต่อไปนี้สำหรับเปรียบเทียบสตริงอักขระแบบกว้าง:

รูทีนย่อย	คำอธิบาย
wcscmp	เปรียบเทียบสตริงอักขระ wide สองรายการ
wcsncmp	เปรียบเทียบจำนวนที่ระบุของสตริงอักขระ wide

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcscmp** เพื่อเปรียบเทียบสตริงอักขระ wide สองรายการ:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    int retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  pwcs1 และ pwcs2 ชี้ไปยังสตริงอักขระ wide สองรายการ
    **  ที่จะเปรียบเทียบ
    */
    retval = wcscmp(pwcs1, pwcs2);
```

```
    /*  pwcs1 มีสำเนาของสตริงอักขระ wide
    **  ใน pwcs2
    */
}
```

รูทีนย่อยการแปลงสตริงอักขระ wide

รูทีนย่อยการสนับสนุน multicultural ต่อไปนี้แปลงสตริงอักขระแบบกว้าง เป็นดับเบิล จำนวนเต็มแบบยาว และจำนวนเต็มแบบยาวที่ไม่มีเครื่องหมาย:

รูทีนย่อย	คำอธิบาย
wcstod	แปลงสตริงอักขระ wide เป็นจุดลอยตัว ความแม่นยำดับเบิล
wcstol	แปลงสตริงอักขระ wide เป็นจำนวนเต็ม แบบยาวที่มีเครื่องหมาย
wcstoul	แปลงสตริงอักขระ wide เป็นจำนวนเต็มแบบยาว ที่ไม่มีเครื่องหมาย

ก่อนการเรียกรูทีนย่อย **wcstod, wcstoul,** หรือ **wcstol** ต้องตั้งค่าตัวแปรสากล **errno** เป็น 0 เพื่อให้สามารถจัดการกับข้อผิดพลาดใดๆ ที่เกิดขึ้นอันเป็นผลมาจากการเรียกรูทีนย่อยเหล่านี้ได้อย่างถูกต้อง

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcstod** เพื่อแปลงสตริงอักขระ wide เป็นจุดลอยตัวความแม่นยำ สองเท่า:

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>

extern int errno;

main()
{
    wchar_t  *pwcs, *endptr;
    double   retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้ pwcs ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ซึ่งมีค่าจุดลอยตัว
    */
    errno = 0;   /*  ตั้งค่า errno เป็นศูนย์  */
    retval = wcstod(pwcs, &endptr);

    if(errno != 0){
        /*  errno มีการเปลี่ยนแปลง ดังนั้นจึงเกิดข้อผิดพลาดขึ้น */

        if(errno == ERANGE){
            /*  ค่าที่ถูกต้องคือค่าภายนอกช่วงของ
            **  ค่าที่แสดงได้ ในกรณีของข้อผิดพลาด
            **  โอเวอร์โฟลว์
            */

             if((retval == HUGE_VAL) ||
```

```
                (retval == -HUGE_VAL)){
                /*  ในกรณีของข้อผิดพลาด ให้จัดการตามนั้น  */
            }else if(retval == 0){
                /*  ค่าที่ถูกต้องทำให้เกิดอันเดอร์โฟลว์   */
                /*  จัดการอย่างเหมาะสม             */
            }
        }
    }
    /*  retval มีดับเบิล  */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย wcstol เพื่อแปลง สตริงอักขระ wide **เป็นจำนวนเต็มแบบยาวที่มีเครื่องหมาย:**

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>
#include <stdio.h>

extern int errno;

main()
{
    wchar_t   *pwcs, *endptr;
    long int  retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้ pwcs ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ซึ่งมีค่าจำนวนเต็มแบบยาวที่มีเครื่องหมาย
    */
    errno = 0;   /*  ตั้งค่า errno เป็นศูนย์  */
    retval = wcstol(pwcs, &endptr, 0);

    if(errno != 0){
        /*  errno มีการเปลี่ยนแปลง ดังนั้นจึงเกิดข้อผิดพลาดขึ้น */

        if(errno == ERANGE){
            /*  ค่าที่ถูกต้องคือค่าภายนอกช่วงของ
            **  ค่าที่แสดงได้ ในกรณีของข้อผิดพลาด
            **  โอเวอร์โฟลว์
            */

             if((retval == LONG_MAX) || (retval == LONG_MIN)){
                /*  ในกรณีของข้อผิดพลาด ให้จัดการตามนั้น */
            }else if(errno == EINVAL){
                /*  ค่าของพื้นฐานไม่ได้รับการสนับสนุน  */
                /*  จัดการอย่างเหมาะสม             */
            }
        }
    }
    /*  retval มีจำนวนเต็มแบบยาว  */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย wcstoul เพื่อแปลง สตริงอักขระ wide **เป็นจำนวนเต็มแบบยาวที่ไม่มีเครื่องหมาย:**

```
#include <stdlib.h>
#include <locale.h>
#include <errno.h>

extern int errno;

main()
{
    wchar_t    *pwcs, *endptr;
    unsigned long int  retval;

    (void)setlocale(LC_ALL, "");

    /*
    **  ให้ pwcs ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ซึ่งมีค่าจำนวนเต็มแบบยาวที่ไม่มีเครื่องหมาย
    */
    errno = 0;   /*  ตั้งค่า errno เป็นศูนย์  */
    retval = wcstoul(pwcs, &endptr, 0);

    if(errno != 0){
        /*  เกิดข้อผิดพลาดขึ้น */
        if(retval == ULONG_MAX || errno == ERANGE){
            /*
            **  ค่าที่ถูกต้องคือค่าภายนอกช่วงของ
            **  ค่าที่แสดงได้ จัดการอย่างเหมาะสม
            */
        }else if(errno == EINVAL){
            /*  ค่าของพื้นฐานไม่สามารถแสดงได้  */
            /*  จัดการอย่างเหมาะสม             */
        }
    }
    /*  retval มีจำนวนเต็มแบบยาวที่ไม่มีเครื่องหมาย  */
}
```

รูทีนย่อยการคัดลอกสตริงอักขระ wide

รูทีนย่อยการสนับสนุน multicultural ต่อไปนี้สำหรับการคัดลอกสตริงอักขระแบบกว้าง:

รูทีนย่อย	คำอธิบาย
wcscpy	คัดลอกสตริงอักขระ wide ไปยังสตริงอักขระ wide อื่น
wcsncpy	คัดลอกจำนวนเฉพาะของอักขระจากสตริงอักขระ wide ไปยังสตริงอักขระ wide อื่น
wcscat	ผนวกสตริงอักขระ wide เข้ากับสตริงอักขระ wide อื่น
wcsncat	ผนวกจำนวนเฉพาะของอักขระจากสตริงอักขระ wide เข้ากับสตริงอักขระ wide อื่น

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcscpy** เพื่อคัดลอกสตริงอักขระ wide เข้าในอาร์เรย์อักขระ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>
```

```
main()
{
    wchar_t *pwcs1, *pwcs2;
    size_t  n;

    (void)setlocale(LC_ALL, "");
    /*
    **  จัดสรรอาร์เรย์อักขระ wide ที่ต้องการ
    */
    pwcs1 = (wchar_t *)malloc( (wcslen(pwcs2) +1)*sizeof(wchar_t));
    wcscpy(pwcs1, pwcs2);
    /*
    **  pwcs1 มีสำเนาของสตริงอักขระ wide ใน pwcs2
    */
}
```

รูทีนย่อยการค้นหาสตริงอักขระ wide

รูทีนย่อยการสนับสนุน multicultural ต่อไปนี้ใช้เพื่อค้นหาสตริงอักขระแบบกว้าง:

รูทีนย่อย	คำอธิบาย
wcschr	ค้นหาการเกิดครั้งแรกของอักขระ wide ในสตริง อักขระ wide
wcsrchr	ค้นหาการเกิดครั้งล่าสุดของอักขระ wide ในสตริง อักขระ wide
wcspbrk	ค้นหาการเกิดครั้งแรกของหลายอักขระ wide ในสตริง อักขระ wide
wcsspn	กำหนดจำนวนของอักขระ wide ใน เซกเมนต์แรกเริ่มของสตริงอักขระ wide
wcscspn	ค้นหาสตริงอักขระ wide
wcswcs	ค้นหาการเกิดครั้งแรกของสตริงอักขระ wide ภายในสตริง อักขระ wide อื่น
wcstok	แบ่งสตริงอักขระ wide เป็นลำดับของสตริงอักขระ wide ที่แบ่งแยก

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcschr** เพื่อระบุตำแหน่งการเกิดขึ้นครั้งแรกของอักขระ wide ในสตริงอักขระ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, wc, *pws;
    int     retval;

    (void)setlocale(LC_ALL, "");

    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ wc ชี้ไปยังอักขระ wide ที่จะค้นหา
    **
    */
    pws = wcschr(pwcs1, wc);
```

```
    if (pws == (wchar_t )NULL ){
        /*  wc ไม่ได้เกิดขึ้นใน pwcs1  */
    }else{
        /*  pws ชี้ไปยังที่ตั้งที่พบ wc  */
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcsrchr** เพื่อระบุตำแหน่ง การเกิดขึ้นครั้งล่าสุดของอักขระ wide ในสตริงอักขระ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, wc, *pws;
    int     retval;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ wc ชี้ไปยังอักขระ wide ที่จะค้นหา
    **
    */
    pws = wcsrchr(pwcs1, wc);
    if (pws == (wchar_t )NULL ){
        /*  wc ไม่ได้เกิดขึ้นใน pwcs1  */
    }else{
        /*  pws ชี้ไปยังที่ตั้งที่พบ wc  */
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcspbrk** เพื่อระบุตำแหน่งการเกิดขึ้นครั้งแรกของหลายอักขระ wide ในสตริงอักขระ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2, *pws;

    (void)setlocale(LC_ALL, "");

    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ pwcs2 มีการเริ่มต้นเป็นสตริงอักขระ wide
    **  ซึ่งมีอักขระ wide ที่จะค้นหา
    */
    pws = wcspbrk(pwcs1, pwcs2);

    if (pws == (wchar_t )NULL ){
        /* ไม่พบอักขระ wide จาก pwcs2 ใน pwcs1    */
```

```
    }else{
        /* pws ชี้ไปยังที่ตั้งซึ่งพบข้อมูลที่ตรงกัน */
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย wcsspn เพื่อกำหนด จำนวนของอักขระ wide **ในเซกเมนต์แรกเริ่มของเซกเมนต์สตริงอักขระ**ำ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    size_t count;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ pwcs2 มีการเริ่มต้นเป็นสตริงอักขระ wide
    **  ซึ่งมีอักขระ wide ที่จะค้นหา
    */
    count = wcsspn(pwcs1, pwcs2);
    /*
    **  จำนวนมีความยาวของเซกเมนต์
    */
 }
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย wcscspn **เพื่อกำหนด** จำนวนของอักขระ wide **ที่ไม่ได้อยู่ในเซกเมนต์สตริงอักขระ** wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2;
    size_t count;

    (void)setlocale(LC_ALL, "");

    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ pwcs2 มีการเริ่มต้นเป็นสตริงอักขระ wide
    **  ซึ่งมีอักขระ wide ที่จะค้นหา
    */
    count = wcscspn(pwcs1, pwcs2);
    /*
    **  จำนวนมีความยาวของเซกเมนต์ที่ประกอบด้วย
    **  อักขระซึ่งไม่มีอยู่ใน pwcs2
    */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcswcs** เพื่อระบุตำแหน่ง การเกิดขึ้นครั้งแรกของสตริงอักขระ wide ภายในสตริงอักขระ wide อื่น:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1, *pwcs2, *pws;

    (void)setlocale(LC_ALL, "");
    /*
    **  ให้ pwcs1 ชี้ไปยังสตริงที่ยุติด้วย null ของอักขระ wide
    **  ให้ pwcs2 มีการเริ่มต้นเป็นสตริงอักขระ wide
    **  ซึ่งมีลำดับอักขระ wide ที่จะระบุตำแหน่ง
    */
    pws = wcswcs(pwcs1, pwcs2);
    if (pws == (wchar_t)NULL){
       /* ไม่พบลำดับอักขระ wide จาก pwcs2 ใน pwcs1    */
    }else{
        /*
        **  pws ชี้ไปยังการเกิดขึ้นครั้งแรกของลำดับ
        **  ที่ระบุโดย pwcs2 ใน pwcs1
        */
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **wcstok** เพื่อ tokenize สตริงอักขระ wide:

```
#include <string.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wchar_t *pwcs1 = L"?a???b,,,#c";
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");
    pwcs = wcstok(pwcs1, L"?");
    /*  pws ชี้ไปยัง token:  L"a"  */
    pwcs = wcstok((wchar_t *)NULL, L",");
    /*  pws ชี้ไปยัง token:  L"??b"  */
    pwcs = wcstok((wchar_t *)NULL, L"#,");
    /*  pws ชี้ไปยัง token:  L"c"  */
}
```

รูทีนย่อยอินพุต/เอาต์พุตอักขระ wide

การสนับสนุน Multicultural มีรูทีนย่อยสำหรับทั้ง I/O ที่จัดรูปแบบและ ที่ไม่จัดรูปแบบ

I/O อักขระ wide ที่จัดรูปแบบ:

รูทีนย่อย **printf** และ **scanf** ช่วยในการจัดรูปแบบ อักขระ wide รูทีนย่อย **printf** และ **scanf** มีตัวระบุรูปแบบเพิ่มเติมสองตัว สำหรับการจัดการอักขระ wide ซึ่งได้แก่: %C และ %S ตัวระบุรูปแบบ %C และ %S ช่วยให้มี I/O บน อักขระ wide และสตริง อักขระ wide ตามลำดับ ตัวระบุรูปแบบ ทั้งสองนี้คล้ายกับตัวระบุรูปแบบ %c และ %s ซึ่งช่วยให้มี I/O บน อักขระและสตริง หลายไบต์

รูทีนย่อย multibyte ยอมรับอาร์เรย์หลายไบต์และแสดงเอาต์พุตอาร์เรย์ หลายไบต์ เมื่อต้องการแปลงเอาต์พุตหลายไบต์จากรูทีนย่อย multibyte เป็น สตริงอักขระ wide ให้ใช้รูทีนย่อย **mbstowcs**

I/O อักขระ wide ที่ไม่ได้จัดรูปแบบ:

รูทีนย่อย I/O อักขระ wide ที่ไม่ได้จัดรูปแบบจะใช้เมื่อ โปรแกรมต้องการ I/O ที่เป็นอิสระจากชุดโค้ดสำหรับอักขระจากชุดโค้ด หลายไบต์ ตัวอย่างเช่น ใช้รูทีนย่อย **fgetwc** หรือ **getwc** เพื่ออินพุต อักขระหลายไบต์ ถ้าโปรแกรมใช้รูทีนย่อย **getc** เพื่ออินพุตอักขระหลายไบต์ โปรแกรมต้องเรียกรูทีนย่อย **getc** หนึ่งครั้ง สำหรับแต่ละไบต์ในอักขระหลายไบต์

รูทีนย่อยอินพุตอักขระ wide อ่านอักขระหลายไบต์จาก สตรีมและแปลงอักขระหลายไบต์นั้นเป็นอักขระ wide การแปลงทำเหมือนกับว่ารูทีนย่อยเรียกรูทีนย่อย **mbtowc** และ **mbstowcs**

รูทีนย่อยเอาต์พุตอักขระ wide แปลงอักขระ wide เป็นอักขระหลายไบต์ และบันทึกอักขระหลายไบต์ที่ได้ลงในสตรีม การแปลงทำ เหมือนกับว่ารูทีนย่อยเรียกรูทีนย่อย **wctomb** และ **wcstombs**

หมวดหมู่ **LC_CTYPE** ของ โลแคลปัจจุบันมีผลกระทบต่อพฤติกรรมของรูทีนย่อย I/O อักขระ wide

การอ่านและการประมวลผลทั้งไฟล์:

ถ้าโปรแกรมต้องดำเนินการในไฟล์ทั้งไฟล์ซึ่งต้อง จัดการในรูปแบบโค้ดอักขระ wide ให้ใช้วิธีอย่างใดอย่างหนึ่งดังต่อไปนี้:

- ในกรณีของอักขระหลายไบต์ ให้ใช้รูทีนย่อย **อ่าน** หรือ **fread** เพื่อแปลงบล็อกของข้อมูลข้อความเข้าในบัฟเฟอร์ แปลงทีละหนึ่งอักขระ ในบัฟเฟอร์นี้โดยใช้รูทีนย่อย **mbtowc** จัดการ ตัวพิมพ์พิเศษของอักขระหลายไบต์บนขอบเขตบล็อก สำหรับชุดโค้ดหลายไบต์ อย่าใช้รูทีนย่อย **mbstowcs** บนบัฟเฟอร์นี้ บนลำดับอักขระหลายไบต์ที่ไม่ถูกต้องหรือแบบบางส่วน รูทีนย่อย **mbstowcs** จะส่งคืน -1 โดยไม่บ่งชี้ว่าแปลงข้อมูลสำเร็จ ไปมากเพียงใดแล้ว คุณสามารถใช้รูทีนย่อย **mbstowcs** กับชุดโค้ด ไบต์เดียว เนื่องจากคุณจะไม่รันจนเกิดปัญหาลำดับไบต์บางส่วน เมื่อใช้ชุดโค้ดไบต์เดียว
- ใช้รูทีนย่อย **fgetws** เพื่อให้ได้รับบรรทัดจากไฟล์ ถ้าสตริงอักขระ wide ที่ส่งคืนมีอักขระ wide <บรรทัดใหม่> ผลคือจะได้รับบรรทัดทั้งหมด ถ้าไม่มีอักขระ wide <บรรทัดใหม่> บรรทัดจะยาวกว่าที่คาดไว้ และต้องใช้การเรียกเพิ่มเติมไปยังรูทีนย่อย **fgetws** เพื่อให้ได้รับบรรทัดทั้งหมด ถ้าโปรแกรมสามารถประมวลอย่างมีประสิทธิภาพ ได้ทีละหนึ่งบรรทัด ขอแนะนำให้ใช้แนวทางนี้
- ถ้าใช้รูทีนย่อย **fgets** ในการอ่านไฟล์หลายไบต์ เพื่อให้ได้ข้อมูลทีละหนึ่งบรรทัด อาจส่งผลให้เกิดอักขระหลายไบต์ที่แบ่งจัดการสภาพนี้เช่นเดียวกับในกรณีของรูทีนย่อย **อ่าน** ที่แบ่งอักขระหลายไบต์บนการอ่านต่อเนื่อง ถ้าคุณ สามารถรับประกันได้ว่าความยาวบรรทัดอินพุตจะไม่มากกว่าขีดจำกัดที่ตั้งค่าไว้ คุณสามารถใช้บัฟเฟอร์ของขนาดนั้น (บวก 1 สำหรับ null) เพื่อหลีกเลี่ยง โอกาสของการแบ่งอักขระหลายไบต์ ถ้าโปรแกรมสามารถ ประมวลอย่างมีประสิทธิภาพได้ทีละหนึ่งบรรทัด สามารถใช้แนวทางนี้ได้ เนื่องจากโอกาสของการแบ่งไบต์ในบัฟเฟอร์ ให้ใช้รูทีนย่อย **fgetws** ก่อนหน้ารูทีนย่อย **fgets** สำหรับอักขระหลายไบต์
- ใช้รูทีนย่อย **fgetwc** ในไฟล์ที่จะอ่านทีละหนึ่งโค้ดอักขระ wide ถ้าไฟล์ใหญ่ ค่าใช้จ่ายในการเรียกฟังก์ชันจะสูง และลดค่าของวิธีการนี้

การตัดสินใจว่าควรจะใช้วิธีการใดเหล่านี้ควรพิจารณาแยกเฉพาะ สำหรับแต่ละโปรแกรม ขอแนะนำอ็อพชันรูทีนย่อย **fgetsw** เนื่องจากทำให้ได้ประสิทธิภาพการทำงานสูงสุดและโปรแกรมไม่มี การจัดการตัวพิมพ์พิเศษ

รูทีนย่อยอินพุต:

ชนิดข้อมูล **wint_t** เป็นสิ่งที่จำเป็นสำหรับการแสดงถึงค่าโค้ดอักขระ wide และมาร์กเกอร์ end-of-file (EOF) ตัวอย่างเช่น พิจารณาตัวพิมพ์ของรูทีนย่อย **fgetwc** ซึ่งจะส่งคืนค่าโค้ดอักขระ wide:

การประกาศค่าส่งคืนของรูทีนย่อย	คำอธิบาย
wchar_t fgetwc();	ถ้าชนิดข้อมูล **wchar_t** มีการกำหนดเป็นค่า **char** คุณจะไม่สามารถแยกความแตกต่างระหว่างสัญลักษณ์ y-umlaut และมาร์กเกอร์ end-of-file (EOF) ในชุดโค้ด ISO8859-1 จุดโค้ด 0xFF เป็นอักขระที่ถูกต้อง (y umlaut) ดังนั้น ค่าที่ส่งคืน จึงไม่สามารถเป็นชนิดข้อมูล **wchar_t** ชนิดข้อมูลที่ต้องการ คือชนิดที่สามารถจัดเก็บทั้งมาร์กเกอร์ EOF และจุดโค้ดทั้งหมดไว้ในชุดโค้ด เดียวได้
int fgetwc();	ในบางเครื่อง ชนิดข้อมูล **int** มีการ กำหนดเป็น 16 บิต ถ้าชนิดข้อมูล **wchar_t** ใหญ่กว่า 16 บิต ค่า **int** จะไม่สามารถ แสดงถึงค่าที่ส่งคืน ทั้งหมดได้

ดังนั้นจึงต้องใช้ชนิดข้อมูล **wint_t** เพื่อแสดงถึงค่าที่ส่งคืนของรูทีนย่อย **fgetwc** ชนิดข้อมูล **wint_t** มีการกำหนดไว้ในไฟล์ **wchar.h**

ค่าส่งคืนของรูทีนย่อย	คำอธิบาย
fgetwc	เรียกใช้อักขระ wide ถัดไปจากสตรีม
fgetws	เรียกใช้สตริงของอักขระ wide จากสตรีม
getwc	เรียกใช้อักขระ wide ถัดไปจากสตรีม
getwchar	เรียกใช้อักขระ wide ถัดไปจากอินพุตมาตรฐาน
getws	เรียกใช้สตริงของอักขระ wide จากอินพุตมาตรฐาน
ungetwc	ออกใช้อักขระ wide บนสตรีม

รูทีนย่อยเอาต์พุต:

รูทีนย่อยต่อไปนี้ใช้สำหรับเอาต์พุตอักขระ wide:

รูทีนย่อย	คำอธิบาย
fputwc	บันทึกอักขระ wide ลงในเอาต์พุตสตรีม
fputws	บันทึกสตริงอักขระ wide ลงในเอาต์พุต สตรีม
putwc	บันทึกอักขระ wide ลงในเอาต์พุตสตรีม
putwchar	บันทึกอักขระ wide ลงในเอาต์พุตมาตรฐาน
putws	บันทึกสตริงอักขระ wide ลงในเอาต์พุตมาตรฐาน

ตัวอย่าง:

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย **fgetwc** เพื่อ อ่านโค้ดอักขระ wide จากไฟล์:

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;
```

```
    (void)setlocale(LC_ALL, "");

    /*
    **  เปิดสตรีม
    */
    fp = fopen("file", "r");

    /*
    **  การจัดการข้อผิดพลาดถ้า fopen ไม่สำเร็จ
    */
    if(fp == NULL){
        /*  ตัวจัดการข้อผิดพลาด  */
    }else{
        /*
        **  pwcs ชี้ไปยังบัฟเฟอร์อักขระ wide ของ BUFSIZ
        */
        while((retval = fgetwc(fp)) != WEOF){
            *pwcs++ = (wchar_t)retval;
               /*  หยุดเมื่อบัฟเฟอร์เต็ม  */
        }
    }
    /*  ประมวลผลอักขระ wide ในบัฟเฟอร์  */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย getwchar เพื่อ อ่านอักขระ wide จากอินพุตมาตรฐาน:

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

 main()
{
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    index = 0;
    while((retval = getwchar()) != WEOF){
        /*  pwcs ชี้ไปยังบัฟเฟอร์อักขระ wide ของ BUFSIZ  */
        *pwcs++ = (wchar_t)retval;
        /*  หยุดเมื่อบัฟเฟอร์เต็ม  */
    }
    /*  ประมวลผลอักขระ wide ในบัฟเฟอร์  */
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย ungetwc เพื่อออกใช้อักขระ wide บนอินพุตสตรีม:

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    wint_t  retval;
```

```
    FILE    *fp;

    (void)setlocale(LC_ALL, "");
    /*
    **  เปิดสตรีม
    */
    fp = fopen("file", "r");

    /*
    **  การจัดการข้อผิดพลาดถ้า fopen ไม่สำเร็จ
    */
    if(fp == NULL){
        /*  ตัวจัดการข้อผิดพลาด  */

    else{
        retval = fgetwc(fp);
        if(retval != WEOF){
            /*
            **  มองเห็นที่อักขระและส่งคืนไปยังสตรีม
            */
            retval = ungetwc(retval, fp);
            if(retval == EOF){
                /*  ข้อผิดพลาดใน ungetwc  */
            }
        }
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย fgetws เพื่ออ่าน ไฟล์ทีละหนึ่งบรรทัด:

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  เปิดสตรีม
    */
    fp = fopen("file", "r");

    /*
    **  การจัดการข้อผิดพลาดถ้า fopen ไม่สำเร็จ
    */
    if(fp == NULL){
        /*  ตัวจัดการข้อผิดพลาด  */
    }else{
        /*  pwcs ชี้ไปยังบัฟเฟอร์อักขระ wide ของ BUFSIZ  */
        while(fgetws(pwcs, BUFSIZ, fp) != (wchar_t *)NULL){
            /*
            ** pwcs มีอักขระ wide ที่ยุติด้วย
```

```
            **  null
            */
        }
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย fputwc เพื่อ บันทึกอักขระ wide **ลงในเอาต์พุตสตรีม:**

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    int     index, len;
    wint_t  retval;
    FILE    *fp;
    wchar_t *pwcs;

    (void)setlocale(LC_ALL, "");

    /*
    **  เปิดสตรีม
    */
    fp = fopen("file", "w");

    /*
    **  การจัดการข้อผิดพลาดถ้า fopen ไม่สำเร็จ
    */
    if(fp == NULL){
        /*  ตัวจัดการข้อผิดพลาด  */
    }else{
        /*  ให้ len บ่งชี้จำนวนของอักขระ wide ที่จะแสดงเอาต์พุต
        **  pwcs ชี้ไปยังบัฟเฟอร์อักขระ wide ของ BUFSIZ
        */
        for(index=0; index < len; index++){
            retval = fputwc(*pwcs++, fp);
            if(retval == WEOF)
                break;  /*  เกิดข้อผิดพลาดการบันทึก  */
                        /*  errno มีการตั้งค่าเพื่อบ่งชี้ข้อผิดพลาด */
        }
    }
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย fputws เพื่อบันทึกสตริงอักขระ wide **ลงในไฟล์:**

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>

main()
{
    int     retval;
    FILE    *fp;
    wchar_t *pwcs;
```

```
    (void)setlocale(LC_ALL, "");

    /*
    **  เปิดสตรีม
    */
    fp = fopen("file", "w");

    /*
    **  การจัดการข้อผิดพลาดถ้า fopen ไม่สำเร็จ
    */
    if(fp == NULL){
        /*  ตัวจัดการข้อผิดพลาด  */

    }else{
        /*
        **  pwcs ชี้ไปยังสตริงอักขระ wide
        **  ที่จะแสดงเอาต์พุตไปยัง fp
        */
        retval = fputws(pwcs, fp);
        if(retval == -1){
            /*  เกิดข้อผิดพลาดการบันทึก                */
            /*  errno มีการตั้งค่าเพื่อบ่งชี้ข้อผิดพลาด  */
        }
    }
}
```

การทำงานกับค่าคงที่อักขระ wide

ใช้ค่าคงที่ **L** สำหรับอักขระ ASCII เท่านั้น สำหรับอักขระ ASCII ค่าคงที่ **L** เป็นตัวเลขเดียวกันกับ ค่าจุดรหัสของอักขระ ตัวอย่างเช่น **L'a'** เหมือนกับ a ค่าคงที่ **L** มีค่า **wchar_t** ของอักขระ ASCII ที่ใช้สำหรับการกำหนด ค่าคงที่อักขระ wide มีการระบุโดยตัวระบุ **L** ตัวอย่างเช่น:

```
wchar_t wc = L'x' ;
```

โค้ดอักขระ wide ที่สอดคล้องกับอักขระ x มีการ จัดเก็บไว้ใน wc คอมไพเลอร์ C แปลงอักขระ x โดยใช้ รูทีนย่อย **mbtowc** หรือ **mbstowcs** ตามความเหมาะสม การแปลง เป็นอักขระ wide นี้ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน ณ เวลาคอมไพล์ เนื่องจากอักขระ ASCII เป็นส่วนประกอบหนึ่งของชุดโค้ดที่ได้รับการ สนับสนุนทั้งหมด และการแสดงอักขระ wide ของอักขระ ASCII ทั้งหมด เหมือนกันในทุกโลแคล L'x' จึงให้ผลลัพธ์ค่าเดียวกัน บนชุดโค้ดทั้งหมด อย่างไรก็ตาม ถ้าอักขระ x ไม่ใช่ ASCII โปรแกรมอาจไม่ทำงานเมื่อโปรแกรมรันบนชุดโค้ดอื่น ที่ไม่ใช่ชุดโค้ดที่ใช้ ณ เวลาคอมไพล์ ข้อจำกัดนี้กระทบบางโปรแกรมที่ใช้คำสั่งเปลี่ยนซึ่งใช้การแสดงค่าคงที่อักขระ wide

ไฟล์ส่วนหัว wchar.h

ไฟล์ส่วนหัว **wchar.h** แสดงข้อมูลที่จำเป็นสำหรับ การเขียนโปรแกรมโดยใช้รูทีนย่อยอักขระหลายไบต์และอักขระ wide ไฟล์ส่วนหัว **wchar.h** แสดงข้อมูลชนิด **wchar_t, wctype_t,** และ **wint_t** และฟังก์ชันหลายอย่างสำหรับการทดสอบอักขระ wide เนื่องจากจำนวนของอักขระที่ดำเนินการเป็นอักขระ wide มากเกินกว่าจำนวนของอักขระพื้นฐาน จึงไม่สามารถจัดประเภทอักขระ wide ทั้งหมดเข้าในคลาสที่มีอยู่ซึ่งใช้สำหรับอักขระพื้นฐาน ได้ ด้วยเหตุนี้ จึงจำเป็นต้องนำเสนอวิธีการกำหนด คลาสเพิ่มเติมเฉพาะสำหรับบางโลแคล การดำเนินการของรูทีนย่อยเหล่านี้ ได้รับผลกระทบจากโลแคลปัจจุบัน

ไฟล์ส่วนหัว **wchar.h** ยังแสดงรูทีนย่อย สำหรับการจัดดำเนินการสตริงอักขระ wide (นั่นคืออาร์เรย์ชนิดข้อมูล **wchar_t**) ด้วย ความยาวอาร์เรย์มีการกำหนดในหน่วยของจำนวนองค์ประกอบ **wchar_t** ในอาร์เรย์เสมอ โค้ดอักขระ wide null ใช้เพื่อจบ อาร์เรย์ ตัวชี้ไปยังชนิดข้อมูล **wchar_t** หรือ void array จะชี้ไปยังอิลิเมนต์แรกเริ่มของอาร์เรย์เสมอ

หมายเหตุ: ถ้าจำนวนของอิลิเมนต์ **wchar_t** ในอาร์เรย์เกินกว่าความยาวอาร์เรย์ที่กำหนดไว้ อาจเกิดผลที่คาดไม่ถึง

รูทีนย่อยนิพจน์ทั่วไปที่ทำให้เป็นโกลบอล

โปรแกรมที่มีนิพจน์ทั่วไปที่ทำให้เป็นโกลบอลสามารถ ใช้รูทีนย่อย **regcomp, regexec, regerror, regfree** และ **fnmatch**

รูทีนย่อยต่อไปนี้มีให้สำหรับใช้กับนิพจน์ทั่วไป ที่ทำให้เป็นโกลบอล

regcomp
คอมไพล์นิพจน์ธรรมดาพื้นฐานหรือแบบขยายที่ระบุเข้าใน สตริงที่ดำเนินการได้

regexec เปรียบเทียบสตริงที่ยุติด้วย null กับนิพจน์ธรรมดาพื้นฐานหรือแบบขยายที่ คอมไพล์ ซึ่งต้องมีการคอมไพล์ก่อนหน้านี้โดยการเรียกไปยังรูทีนย่อย **regcomp**

regerror
แสดงการแม็พจากโค้ดระบุความผิดพลาดที่ส่งคืนโดยรูทีนย่อย **regcomp** และ **regexec** ไปยังสตริงที่พิมพ์ได้

regfree ทำให้หน่วยความจำที่จัดสรรโดยรูทีนย่อย **regcomp** ซึ่งเชื่อมโยง กับนิพจน์ธรรมดาพื้นฐานหรือแบบขยายที่คอมไพล์ว่างลง นิพจน์จะไม่ถูกจัดการ เป็นนิพจน์ธรรมดาพื้นฐานหรือแบบขยายที่คอมไพล์อีกต่อไป หลังจากมีการกำหนดนิพจน์ให้กับรูทีนย่อย **regfree**

fnmatch
ตรวจสอบสตริงที่ระบุเพื่อดูว่าสตริงนั้นตรงกับรูปแบบที่ระบุหรือไม่ คุณสามารถใช้รูทีนย่อย **fnmatch** ในแอ็พพลิเคชันที่จะอ่าน พจนานุกรมเพื่อค้นหาว่ารายการใดตรงกับรูปแบบที่กำหนด คุณยังสามารถ ใช้รูทีนย่อย **fnmatch** เพื่อจับคู่ชื่อพาธกับรูปแบบได้ด้วย

ตัวอย่าง

ตัวอย่างต่อไปนี้คอมไพล์นิพจน์ทั่วไปที่เป็นโกลบอล และจับคู่สตริงโดยใช้นิพจน์ที่คอมไพล์นี้ พบ ข้อมูลที่ตรงกันสำหรับรูปแบบแรก แต่ไม่พบข้อมูลที่ตรงกันสำหรับ รูปแบบที่สอง

```
#include                <locale.h>
#include                <regex.h>

#define                 BUFSIZE    256


main()
{
        char    *p;

        char    *pattern[] = {
                "hello[0-9]*",
                "1234"
        };


        char     *string = "this is a test string hello112 and this is test";
```

```
                /* นี่เป็นสตริงต้นฉบับสำหรับการจับคู่ */

        int     retval;
        regex_t   re;
        char    buf[BUFSIZE];

        int     i;

        setlocale(LC_ALL, "");

        for(i = 0;i <2; i++){
                retval = match(string, pattern[i], &re);
                if(retval == 0){
                        printf("Match found \n");
                }else{
                        regerror(retval, &re, buf, BUFSIZE);
                        printf("error = %s\n", buf);
                }
        }
        regfree( &re);
}


int match(char *string, char *pattern, regex_t *re)
{
        int     status;

        if((status=regcomp( re, pattern, REG_EXTENDED))!= 0)
                return(status);
        status = regexec( re, string, 0, NULL, 0);
        return(status);
}
```

ตัวอย่างต่อไปนี้ค้นหาสตริงย่อยทั้งหมดในบรรทัดที่ตรงกับ รูปแบบ หมายเลข 11 และ 2001 ตรง กัน ตัวเลขทุกตัวที่ตรงกันจะนับเป็นหนึ่งข้อมูลที่ตรงกัน มีข้อมูลที่ตรงกัน ดังกล่าวหกรายการซึ่งสอดคล้องกับตัวเลขหกตัวที่ระบุในสตริง

```
#include         <locale.h>
#include                <regex.h>

#define                BUFSIZE   256


main()
{
        char    *p;

        char    *pattern = "[0-9]";
        char    *string = "Today is 11 Feb 2001 ";

        int     retval;
        regex_t re;
        char    buf[BUFSIZE];
        regmatch_t pmatch[100];
        int      status;
        char     *ps;
```

```
        int      eflag;

        setlocale(LC_ALL, "");

        /* คอมไพล์รูปแบบ */

        if((status = regcomp( &re, pattern, REG_EXTENDED))!= 0){
                regerror(status, &re, buf, 120);
                exit(2);
        }

        ps = string;
        printf("String to match=%s\n", ps);
        eflag = 0;

        /* แยกข้อมูลที่ตรงกันทั้งหมด */
        while( status = regexec( &re, ps, 1, pmatch, eflag)== 0){
                printf("match found at: %d, string=%s\n",
                        pmatch[0].rm_so, ps +pmatch[0].rm_so);
                ps += pmatch[0].rm_eo;
                printf("\nNEXTString to match=%s\n", ps);
                eflag = REG_NOTBOL;

        }
        regfree( &re);
}
```

ตัวอย่างต่อไปนี้ใช้รูทีนย่อย fnmatch เพื่ออ่านไดเร็กทอรีและจับคู่ชื่อไฟล์กับรูปแบบ

```
#include                <locale.h>
#include                <fnmatch.h>
#include                <sys/dir.h>

main(int argc, char *argv[] )
{
        char    *pattern;
        DIR     *dir;
        struct dirent    *entry;
        int     ret;


                setlocale(LC_ALL, "");

        dir = opendir(".");

        pattern = argv[1];

        if(dir != NULL){
                while( (entry = readdir(dir)) != NULL){
                        ret = fnmatch(pattern, entry->d_name,
                                FNM_PATHNAME|FNM_PERIOD);
                        if(ret == 0){
                                printf("%s\n", entry->d_name);
                        }else if(ret == FNM_NOMATCH){
                                continue ;
```

```
                        }else{
                                printf("error file=%s\n",
                                        entry->d_name);
                        }
                }
                closedir(dir);
        }
}
```

ชุดโค้ดสำหนับการสนับสนุน multicultural

การทำให้เป็นโกลบอลของ AIX อิง ตามสมมติฐานว่าชุดโค้ดทั้งหมดสามาถรแบ่งออกเป็น ชุดอักขระจำนวนหนึ่ง

เพื่อทำความเข้าใจกับชุดโค้ด คุณจำเป็นต้องเข้าใจชุดอักขระก่อน *ชุดอักขระ* เป็นชุดของอักขระที่กำหนดไว้แล้วตามความต้องการเฉพาะของภาษาหนึ่งภาษาขึ้นไป โดยไม่คำนึงถึงค่าที่เข้ารหัสที่ใช้เพื่อแสดงอักขระ ตัวเลือก ของชุดโค้ดที่จะใช้ขึ้นอยู่กับความต้องการการประมวลผลข้อมูลของผู้ใช้ ชุดอักขระเฉพาะสามารถเข้ารหัสได้โดยใช้แบบแผนการเข้ารหัส ที่แตกต่างกัน ตัวอย่างเช่น ชุดอักขระ ASCII จะกำหนดชุดของอักขระ ที่พบในภาษาอังกฤษ ชุดอักขระ Japanese Industrial Standard (JIS) จะกำหนดชุดของอักขระที่ใช้ในภาษาญี่ปุ่น ชุดอักขระทั้งภาษาอังกฤษและภาษาญี่ปุ่นสามารถเข้ารหัสได้โดยใช้ ชุดโค้ดที่แตกต่างกัน

โค้ดเพจ คล้ายกับชุดโค้ดที่มีข้อจำกัดว่า ข้อมูลจำเพาะเกี่ยวกับโค้ดเพจสร้างขึ้นบนเมตริกซ์ 16 คอลัมน์คูณ 16 แถว จุดตัดระหว่างแต่ละคอลัมน์และแถวเป็นตัวกำหนดอักขระที่เข้ารหัส

ให้พิจารณาข้อมูลดังต่อไปนี้เมื่อทำงานกับชุดโค้ด:

- อย่าสมมุติว่าขนาดของอักขระทั้งหมดจะเป็น 8 บิต หรือ 1 ไบต์ อักขระอาจเป็น 1, 2, 3, 4 ไบต์หรือมากกว่านั้นก็ได้
- อย่าสมมุติการเข้ารหัสของชุดโค้ดใดๆ
- อย่า hard code ชื่อของชุดโค้ด โลแคล หรือฟอนต์ เนื่องจาก อาจส่งผลกระทบต่อความสามารถในการใช้ได้หลายระบบ

ชุดโค้ดที่ได้รับการสนับสนุนมีดังต่อไปนี้:

- มีการสนับสนุนชุดโค้ดตามมาตรฐานอุตสาหกรรม ชุดโค้ดในตระกูล **ISO8859** ให้การสนับสนุนชุดโค้ดไบต์เดียวซึ่ง รวมถึง:
  - ลาติน-1
  - ลาติน-2
  - ลาติน-4
  - ซิริลลิก
  - อารบิก
  - กรีก
  - ฮิบรู
  - เตอร์กิช

  ชุดโค้ดตามมาตรฐานอุตสาหกรรมที่มีอยู่มีดังต่อไปนี้:
  - ชุดโค้ด IBM-eucJP คือชุดโค้ดตามมาตรฐานอุตสาหกรรมที่ใช้ เพื่อสนับสนุนโลแคลภาษาญี่ปุ่น
  - ชุดโค้ด IBM-eucKR คือชุดโค้ดตามมาตรฐานอุตสาหกรรมที่ใช้ เพื่อสนับสนุนประเทศที่ใช้ภาษาเกาหลี

  - ชุดโค้ด IBM-eucTW คือชุดโค้ดตามมาตรฐานอุตสาหกรรมที่ใช้ เพื่อสนับสนุนประเทศที่ใช้ภาษาจีนดั้งเดิม
  - ชุดโค้ด IBM-eucCN คือชุดโค้ดตามมาตรฐานอุตสาหกรรมที่ใช้ เพื่อสนับสนุนประเทศที่ใช้ภาษาจีนประยุกต์
  - ชุดโค้ด UTF-8 คือ Universal Transformation Format ของ Unicode/ISO10646 ที่ใช้เพื่อสนับสนุนหลายภาษาพร้อมกัน (ได้แก่ ภาษาจีนประยุกต์ ภาษาจีนดั้งเดิม และอักขระภาษาจีนที่ใช้ในภาษาญี่ปุ่นและภาษาเกาหลี)
- ชุดโค้ดมาตรฐาน **ISO8859-15** คือมาตรฐานที่ใช้แทน ชุดโค้ด ISO8859-1 ที่มีอยู่ ซึ่งปัจจุบันมีการใช้โดย โลแคลภาษายุโรปตะวันตก สหรัฐอเมริกา และแคนาดา ความต้องการ ชุดโค้ดอื่นเป็นผลมาจากการออกใช้หน่วยสกุลเงินยูโร และความต้องการของประเทศในยุโรปที่ต้องการทำธุรกรรมทางธุรกิจ โดยใช้เงินยูโร นอกจากนั้น ISO8859-15 ยังมีอักขระเพิ่มเติม 7 แบบ สำหรับภาษาฝรั่งเศสและฟินนิช
- ส่วนสนับสนุนถูกจัดเตรียมไว้สำหรับคอมพิวเตอร์ส่วนบุคคล (PC) อ้างอิงชุดรหัส IBM-856, IBM-943 และ IBM-1046 IBM-856 คือชุดโค้ดไบต์เดียวที่ใช้เพื่อสนับสนุนประเทศที่ใช้ภาษาฮิบรู IBM-943 เป็นชุดรหัสแบบมัลติไบต์ที่ใช้เพื่อสนับสนุนโลแคลภาษาญี่ปุ่น IBM-1046 คือชุดโค้ดไบต์เดียวที่ใช้เพื่อสนับสนุนประเทศที่ใช้ ภาษาอารบิก
- **IBM-1129** คือชุดโค้ดไบต์เดียวที่ใช้เพื่อสนับสนุนภาษาเวียดนาม
- **TIS-620** คือชุดโค้ดไบต์เดียวที่ใช้เพื่อสนับสนุนภาษาไทย
- **IBM-1124** คือชุดโค้ดไบต์เดียวที่ใช้เพื่อสนับสนุนภาษายูเครเนียน
- ชุดโค้ด **UTF-8** ให้การสนับสนุนสำหรับ Full Unicode สำหรับ *ทุก* ภาษาและเขตแดนที่ AIX สนับสนุน ชุดรหัส UTF-8 คือ Universal Transformation Format ของ Unicode/ISO10646 ที่ใช้เพื่อสนับสนุนหลายภาษาพร้อมกัน ชุดรหัส UTF-8 นับเป็น โซลูชันที่สมบูรณ์แบบที่สุดสำหรับการใช้งานในสภาพแวดล้อมซึ่งต้อง ประมวลผลหลายภาษาและตัวอักษรหลายแบบ ชุดโค้ด Unicode/UTF-8 ยังมีการสนับสนุนครบถ้วนสำหรับสกุลเงิน ยูโรทั่วไป (euro)
- มีการสนับสนุนชุดรหัส **IBM-1252** เป็นอ็อพชันความเข้ากันได้ สำหรับผู้ใช้ที่ต้องการสภาพแวดล้อมชุดรหัสไบต์เดียวซึ่งมี สัญลักษณ์สกุลเงินยูโรอยู่ด้วย โครงสร้างของชุดโค้ด IBM-1252 เหมือน กับชุดโค้ดมาตรบานอุตสาหกรรม ISO8859-1 ยกเว้นว่า อักขระกราฟิกเพิ่มเติมจะถูกเพิ่มในช่วงอักขระควบคุม ISO จาก 0x80 ถึง 0x9F สัญลักษณ์สกุลเงินยูโรอยู่ ที่ค่าเลขฐานสิบหก valTXx80 ในชุดโค้ด IBM-1252

หลักการที่เกี่ยวข้อง:

"อักขระ ASCII" ในหน้า 59
ASCII คือชุดโค้ดที่ประกอบด้วยจุดโค้ด 128 จุด (0x00 ถึง 0x7F) ชุดอักขระ ASCII ประกอบด้วยอักขระควบคุม เครื่องหมาย วรรคตอน ตัวเลข และตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก ชุดโค้ด 8 บิตหลายชุดใช้ ASCII เป็นชุดย่อยที่เหมาะสม อย่างไรก็ตาม ตลอดเอกสารนี้ อักขระ ASCII หมายถึงชุดโค้ด 7 บิตเท่านั้น เพื่อ เน้นถึงข้อมูลนี้ จึงมีการอ้างอิงเป็น 7 บิต ASCII ชุดโค้ด 7 บิต ASCII เป็นชุดย่อยที่เหมาะสมของชุดโค้ดที่สนับสนุนทั้งหมด และมีการอ้างอิง เป็น *ชุดอักขระที่ใช้ได้หลายระบบ*

"รายการของตัวแปลง" ในหน้า 101
ตัวแปลงเปลี่ยนข้อมูลจากชุดโค้ดหนึ่งเป็นชุดโค้ดอื่น ชุด ของตัวแปลงที่ได้รับการสนับสนุนในไลบรารี iconv มีการแสดงรายการอยู่ใน ส่วนต่อไปนี้

ชุดโค้ดไบต์เดียวและหลายไบต์

วิธีการเข้ารหัสไบต์เดียวเพียงพอสำหรับการแสดงถึง ชุดอักขระภาษาอังกฤษ เนื่องจากจำนวนของอักขระไม่ มากนัก เพื่อสนับสนุนตัวอักษรให้ได้มากขึ้น เช่น ภาษาญี่ปุ่นและภาษาจีน จึงจำเป็นต้องใช้ชุดโค้ดเพิ่มเติมซึ่งมีการเข้ารหัสแบบมัลติไบต์ ชุดโค้ดไบต์เดียวและมัลติไบต์ที่ได้รับการสนับสนุนทั้งหมด มีชุดอักขระ ASCII ไบต์เดียว ด้วยเหตุนี้ โปรแกรมที่จัดการชุดโค้ดแบบมัลติไบต์ ต้องจัดการการเข้ารหัสอักขระของไบต์จำนวนหนึ่งไบต์ขึ้นไป

ตัวอย่างของชุดโค้ดไบต์เดียวคือชุดโค้ดตระกูล ISO 8859 ตัวอย่างของชุดอักขระหลายไบต์คือชุดโค้ด IBM-eucJP และ IBM-943 ชุดโค้ดไบต์เดียวมีอักขระมากที่สุด 256 อักขระ และชุดโค้ดแบบมัลติไบต์มีมากกว่า 256 (โดยไม่มี ขีดจำกัดในทางทฤษฎี)

ช่วงจุดโค้ดเฉพาะ

ไม่มีชุดโค้ดที่ได้รับการสนับสนุนใดมีไบต์ 0x00 ถึง 0x3F ในไบต์ใดๆ ของอักขระหลายไบต์ จุดโค้ดกลุ่มนี้ เรียกว่า *ช่วงจุดโค้ดเฉพาะ*

จุดโค้ดเหล่านี้อ้างอิงอักขระเดียวกันกับที่ระบุสำหรับ 7 บิต ASCII เสมอ นี่เป็นคุณสมบัติพิเศษที่ควบคุมชุดโค้ดที่สนับสนุนทั้งหมด อักขระ ASCII ในช่วงจุดโค้ดเฉพาะ (“อักขระ ASCII” ในหน้า 59) แสดงรายการอักขระต่างๆ ในช่วงจุดโค้ดเฉพาะ

การแทนค่าข้อมูล

เนื่องจากการเข้ารหัสสำหรับอักขระบางตัวต้องใช้ไบต์มากกว่าหนึ่งไบต์ อักขระตัวเดียวจึงอาจมีการแสดงถึงโดยใช้หนึ่งหรือหลายไบต์ เมื่อมีการสร้างข้อมูลขึ้นในไฟล์หรือเมื่อโอนย้ายข้อมูลระหว่างคอมพิวเตอร์และอุปกรณ์ I/O การแสดงถึงข้อมูลแบบภายนอกนี้เรียกว่าการแสดงถึงอักขระ โค้ดไฟล์ หรือ *โค้ดอักขระหลายไบต์*

สำหรับการประมวลผลสตริงของอักขระดังกล่าว การแปลงโค้ดไฟล์ เป็นการแสดงถึงรูปแบบเดียวกันนับเป็นสิ่งที่คุ้มค่ามากกว่า รูปแบบที่แปลงนี้มีไว้สำหรับการประมวลผลอักขระเป็นการภายในเท่านั้น การแสดงถึงข้อมูลแบบภายในนี้เรียกว่าการแสดงถึงอักขระ *โค้ดกระบวนการ* หรือ *โค้ดอักขระ wide* การทำความเข้าใจ อักขระหลายไบต์ และชุดอักขระ wide เป็นสิ่งจำเป็นต่อ กลยุทธ์การทำให้เป็นโกลบอลโดยรวม

การแสดงข้อมูลโค้ดอักขระแบบมัลติไบต์

โค้ดอักขระแบบมัลติไบต์เป็นการแสดงถึงข้อมูลแบบภายนอก โดยไม่คำนึงว่าข้อมูลนั้นเป็นอินพุตอักขระจากคีย์บอร์ด หรือไฟล์บนดิสก์ ภายในชุดโค้ดเดียวกัน จำนวนของไบต์ ที่แสดงถึงโค้ดแบบมัลติไบต์ของอักขระอาจแตกต่างกัน คุณต้องใช้ฟังก์ชันการสนับสนุน multicultural สำหรับการประมวลผลอักขระเพื่อให้มั่นใจถึงความเป็นอิสระของชุดโค้ด

ตัวอย่างเช่น ชุดโค้ดอาจระบุอักขระที่เข้ารหัสต่อไปนี้:

```
C  = 0x43
*  = 0x81 0x43
*C = 0x81 0x43& 0x43
```

การค้นหาโปรแกรม `C` ที่ไม่ได้พิจารณาถึง อักขระแบบมัลติไบต์ จะค้นหาไบต์ที่สองของสตริง `*C` และสมมุติว่าพบ `C` เมื่อในข้อเท็จจริง การค้นหาพบ ไบต์ที่สองของอักขระ * (เครื่องหมายดอกจัน)

การแสดงข้อมูลอักขระ wide

โค้ดอักขระ wide มีการพัฒนาขึ้นเพื่อให้สามารถประมวลผลอักขระหลายไบต์ แบบภายในได้อย่างมีประสิทธิภาพมากขึ้นในระบบ การแสดงอักขระหลายไบต์จะถูกแปลงเป็นการแสดงภายใน ที่เหมือนกัน (โค้ดอักขระ wide) เพื่อให้อักขระทั้งหมดมีความยาวเท่ากัน ภายในระบบ ด้วยการใช้รูปแบบภายในนี้ การประมวลผลอักขระ จึงสามารถทำในลักษณะที่เป็นอิสระจากชุดโค้ด โค้ดอักขระ wide อ้างอิงถึงการแสดงภายในของอักขระแบบนี้

ชนิดข้อมูล **wchar_t** ใช้เพื่อแสดงถึงโค้ดอักขระ wide ของอักขระ ขนาดของชนิดข้อมูล **wchar_t** เป็นขนาดเฉพาะ สำหรับแต่ละการดำเนินการ ขนาดเป็นคำนิยาม **typedef** และสามารถพบได้ใน ไฟล์ **ctype.h**, **stddef.h**, และ **stdlib.h** ไม่มีโปรแกรมใดต้องสมมติขนาดเฉพาะสำหรับชนิดข้อมูล **wchar_t** เพื่อให้โปรแกรมสามารถรันภายใต้การดำเนินการที่ใช้ขนาดที่แตกต่างอื่นสำหรับชนิดข้อมูล **wchar_t**

บนระบบปฏิบัติการ AIX ชนิดข้อมูล **wchar_t** เป็น 32 บิตในสภาวะแวดล้อม 64 บิต และเป็น 16 บิตในสภาวะแวดล้อม 32 บิต เมธอดโลแคลมี ความเป็นมาตรฐานในโลแคลส่วนใหญ่ ดังนั้น ค่าที่จัดเก็บไว้ใน **wchar_t** สำหรับอักขระเฉพาะจึงเป็นค่าข้อมูล Unicode ของโลแคลนั้นเสมอ สำหรับแอ็พพลิเคชันซึ่งต้องการรันบน AIX เท่านั้น ค่านี้จะอนุญาตให้ แอ็พพลิเคชันจัดการชนิดข้อมูล **wchar_t** ในรูปแบบที่สอดคล้องกัน แม้ว่าจะ ไม่รู้จักชุดอักขระที่ดำเนินการ โลแคลทั้งหมดใช้ Unicode สำหรับค่าโค้ดอักขระ wide (โค้ด กระบวนการ) ยกเว้นชุดโค้ด IBM-eucTW ชุดโค้ด IBM-eucTW (LANG =**zh_TW**) มีอักขระจำนวนมากที่ไม่อยู่ใน มาตรฐาน Unicode ส่งผลให้ ไม่สามารถแสดงถึงอักขระเหล่านี้ด้วยค่าอักขระ Unicode-wide แอ็พพลิเคชันที่ต้องมีข้อมูล **wchar_t** แบบ Unicode สำหรับภาษาจีนดั้งเดิมต้องใช้โลแคล **Zh_TW** (ชุดโค้ด big5) แทน

อย่าสมมุติว่าชนิดข้อมูล **char** มีเครื่องหมายหรือ ไม่มีเครื่องหมาย นี่เป็นข้อมูลเฉพาะแพล็ตฟอร์ม ถ้าระบบเฉพาะที่ใช้ กำหนด **char** เป็น **มีเครื่องหมาย** การเปรียบเทียบกับ ปริมาณ 8 บิตทั้งหมดจะให้ผลลัพธ์ที่ไม่ถูกต้อง เนื่องจาก 8 บิตทั้งหมด จะถูกใช้ในการเข้ารหัสอักขระ จึงควรตรวจสอบให้แน่ใจว่ามีการกำหนด **char** เป็น **unsigned char** ในทุกที่ที่จำเป็น นอกจากนี้ ถ้าค่า **signed char** มีการใช้ในการจัดทำดัชนีอาร์เรย์ ค่าอาจให้ผลลัพธ์ที่ไม่ถูกต้อง เพื่อทำให้โปรแกรม สามารถใช้ได้หลายระบบให้กำหนดอักขระ 8 บิตเป็น **unsigned char**

คุณสมบัติอักขระ

อักขระทุกตัวมีแอ็ตทริบิวต์หรือคุณสมบัติที่พึ่งพาภาษา หลายอย่าง คุณสมบัติเหล่านี้เรียกว่า *คุณสมบัติคลาส* ตัวอย่างเช่น ตัวอักษรพิมพ์เล็ก a ในภาษาอังกฤษอเมริกา มีคุณสมบัติดังต่อไปนี้:

- ตัวอักษร
- ตัวเลขฐานสิบหก
- พิมพ์ได้
- ตัวพิมพ์เล็ก
- กราฟิก

คุณสมบัติคลาสอักขระมีการระบุโดยหมวดหมู่ **LC_CTYPE**

คุณสมบัติการจัดเรียง-การจัดลำดับ

การจัดลำดับอักขระหรือการจัดเรียงหมายถึงการจัดลำดับเฉพาะวัฒนธรรม ของอักขระ การจัดลำดับนี้แตกต่างจากการจัดลำดับตาม ค่าลำดับของอักขระในชุดโค้ด

การจัดลำดับอักขระ หรือ *การจัดเรียง* หมายถึง การจัดลำดับเฉพาะวัฒนธรรมของอักขระ การจัดลำดับนี้แตกต่างจากการจัดลำดับตาม ค่าลำดับของอักขระในชุดโค้ด การจัดลำดับตามการจัดเรียงขึ้นอยู่กับแต่ละภาษา การจัดเรียง อักขระมีการระบุโดยหมวดหมู่ LC_COLLATE คำว่า *องค์ประกอบการจัดเรียง* หมายถึง อักขระหนึ่งตัวขึ้นไปที่มีค่าการจัดเรียงในโลแคล เฉพาะ อักขระ Spanish ll เป็นตัวอย่างขององค์ประกอบการจัดเรียงแบบ หลายอักขระ

เพื่อ เรียงลำดับอักขระในภาษาที่กำหนดในลำดับที่ถูกต้อง จึงมีการกำหนด น้ำหนักให้กับแต่ละอักขระเพื่อให้อักขระเรียงลำดับตามที่คาดไว้ อย่างไรก็ตาม ค่าการเรียงลำดับของอักขระและค่าจุดโค้ดไม่จำเป็นต้อง เกี่ยวข้องกัน

น้ำหนักเพียงชุดเดียวอาจไม่เพียงพอสำหรับการเรียงลำดับสตริงของทุกภาษา ตัวอย่างเช่น ในตัวพิมพ์ของคำภาษาเยอรมัน b<*a-umlaut*>ch และ bane ถ้ามีน้ำหนักเพียงชุดเดียว และน้ำหนักของตัวอักษร a น้อยกว่าน้ำหนักของ <*a-umlaut*> ผลคือ bane จะเรียง ลำดับอยู่ก่อนหน้า b<*a-umlaut*>ch ทั้งที่จริงแล้วต้องอยู่ในลำดับ ตามหลังถึงจะถูกต้อง เพื่อให้เป็นไปตามความต้องการของตัวอย่างนี้ จึงมีการกำหนด น้ำหนักสองชุดคือ น้ำหนักหลักและน้ำหนักรอง ให้กับ อักขระแต่ละตัวในภาษานั้น ในตัวพิมพ์ของอักขระ a และ <*a-umlaut*> ทั้งสองมีน้ำหนักหลักเหมือนกัน แต่มีน้ำหนักรอง แตกต่างกัน ในโลแคลภาษาเยอรมัน น้ำหนักรองของ a น้อยกว่าน้ำหนักรองของ <*a-umlaut*>

ขั้นตอนวิธีการเรียงลำดับจะเปรียบเทียบสตริงสองรายการโดยพิจารณาจาก น้ำหนักหลักของแต่ละอักขระเป็นอันดับแรก ถ้าค่าน้ำหนักหลัก เหมือนกัน จะมีการเปรียบเทียบสตริงสองรายการนั้นอีกครั้งหนึ่งโดยพิจารณาจาก น้ำหนักรองของสตริงนั้น ในตัวอย่างนี้ น้ำหนักหลักของอักขระสองตัวแรก ba และ b<*a-umlaut*> เหมือนกัน แต่น้ำหนักหลักของอักขระที่ตามมา (c และ *n* ตามลำดับ) แตกต่างกัน ผลของการเปรียบเทียบนี้คือ b<*a-umlaut*>ch มี การเรียงลำดับอยู่ก่อนหน้า bane

ในตัวอย่างนี้ ไม่ต้องใช้น้ำหนักรองในการจัดเรียงสตริง อย่างไรก็ตาม ในตัวพิมพ์ของสตริง bach และ b<*a-umlaut*>ch ต้องใช้น้ำหนักรองเพื่อให้ได้ลำดับที่ถูกต้อง เมื่อเปรียบเทียบ โดยใช้ค่าน้ำหนักหลัก พบว่าสตริงสองรายการนี้เท่าเทียมกัน เพื่อเป็นการตัดสิน จึงใช้น้ำหนักรองของ a และ <*a-umlaut*> เนื่องจากน้ำหนักรองของ a น้อยกว่า น้ำหนักรองของ <*a-umlaut*> สตริง bach จึงเรียงลำดับ อยู่ก่อนหน้า b<*a-umlaut*>ch

อักขระ ที่มีน้ำหนักหลักเหมือนกันจะเป็นสมาชิกของ *คลาสความเท่าเทียม* เดียวกัน ในตัวอย่างนี้ อาจพูดได้ว่าอักขระ a และ <*a-umlaut*> เป็นสมาชิกของ คลาสความเท่าเทียมเดียวกัน

ในการจัดเรียง สตริง สตริงแต่ละคู่จะมีการเปรียบเทียบโดยพิจารณาจากน้ำหนักหลัก ก่อน ถ้าสตริงทั้งสองเท่ากัน จะมีการเปรียบเทียบอีกครั้งหนึ่งโดยพิจารณา จากน้ำหนักรอง ถ้ายังคงเท่ากันอีก จะมีการเปรียบเทียบอีกครั้งหนึ่ง โดยพิจาณาจากน้ำหนักชั้นที่สามจนถึงขีดจำกัดที่กำหนดโดยขีดจำกัดน้ำหนักการจัดเรียง **COLL_WEIGHTS_MAX** ซึ่งระบุอยู่ในไฟล์ **sys/limits.h**

ความกว้างชุดโค้ด

ความกว้างชุดโค้ด หมายถึงจำนวนไบต์สูงสุด ที่จำเป็นเพื่อแสดงอักขระเป็นโค้ดไฟล์ ข้อมูลนี้ มีการระบุโดยหมวดหมู่ **LC_CTYPE**

ความกว้างการแสดงชุดโค้ด

ความกว้างการแสดงชุดโค้ด อ้างอิงจำนวนคอลัมน์ สูงสุดที่จะแสดงอักขระบนเทอร์มินัล ข้อมูลนี้ มีการระบุโดยหมวดหมู่ **LC_CTYPE**

อักขระ ASCII

ASCII คือชุดโค้ดที่ประกอบด้วยจุดโค้ด 128 จุด (0x00 ถึง 0x7F) ชุดอักขระ ASCII ประกอบด้วยอักขระควบคุม เครื่องหมาย วรรคตอน ตัวเลข และตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก ชุดโค้ด 8 บิตหลายชุดใช้ ASCII เป็นชุดย่อยที่เหมาะสม อย่างไรก็ตาม ตลอดเอกสารนี้ อักขระ ASCII หมายถึงชุดโค้ด 7 บิตเท่านั้น เพื่อ เน้นถึงข้อมูลนี้ จึงมีการอ้างอิงเป็น 7 บิต ASCII ชุดโค้ด 7 บิต ASCII เป็นชุดย่อยที่เหมาะสมของชุดโค้ดที่สนับสนุนทั้งหมด และมีการอ้างอิง เป็น *ชุดอักขระที่ใช้ได้หลายระบบ*

หลักการที่เกี่ยวข้อง:

“ชุดโค้ดสำหนับการสนับสนุน multicultural” ในหน้า 55

การทำให้เป็นโกลบอลของ AIX อิง ตามสมมติฐานว่าชุดโค้ดทั้งหมดสามาถรแบ่งออกเป็น ชุดอักขระจำนวนหนึ่ง

อักขระ ASCII ในช่วงจุดโค้ดเฉพาะ

ตารางต่อไปนี้แสดงรายการอักขระ ASCII ในช่วงจุดโค้ด เฉพาะ อักขระเหล่านี้อยู่ในช่วง 0x00 ถึง 0x3F

ตารางที่ 2. อักขระ ASCII ในช่วงจุดโค้ดเฉพาะ

ชื่อสัญลักษณ์	ค่า Hex	Glyph	ชื่อสัญลักษณ์	ค่า Hex	Glyph
nul	00		พื้นที่ว่าง	20	ว่างเปล่า
soh	01		เครื่องหมายอุทาน	21	!
stx	02		เครื่องหมายคำพูด	22	″
etx	03		เครื่องหมายตัวเลข	23	#
eot	04		เครื่องหมายดอลลาร์	24	$
enq	05		เปอร์เซ็นต์	25	%
ack	06		เครื่องหมาย ″และ″	26	&
การแจ้งเตือน	07		เครื่องหมายละ	27	’
แป้นถอยหลัง	08		เครื่องหมายวงเล็บซ้าย	28	(
แท็บ	09		เครื่องหมายวงเล็บขวา	29	)
newline	0A		เครื่องหมายดอกจัน	2A	*
แท็บแนวตั้ง	0B		เครื่องหมายบวก	2B	+
ป้อนกระดาษ	0C		คอมมา	2C	,
แป้นปัดแคร่	0D		ยัติภังค์	2D	-
so	0E		จุด	2E	.
si	0F		เครื่องหมายขีด	2F	/
dle	10		ศูนย์	30	0
dc1	11		หนึ่ง	31	1
dc2	12		สอง	32	2
dc3	13		สาม	33	3
dc4	14		สี่	34	4
nak	15		ห้า	35	5
syn	16		หก	36	6
etb	17		เจ็ด	37	7
can	18		แปด	38	8
em	19		เก้า	39	9
sub	1A		โคลอน	3A	:
esc	1B		เซมิโคลอน	3B	;
is1	1C		น้อยกว่า	3C	<

ตารางที่ 2. อักขระ ASCII ในช่วงจุดโค้ดเฉพาะ (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex	Glyph	ชื่อสัญลักษณ์	ค่า Hex	Glyph
is2	1D		เครื่องหมายเท่ากับ	3D	=
is3	1E		มากกว่า	3E	>
is4	1F		เครื่องหมายคำถาม	3F	?

อักขระ ASCII อื่นๆ

ตารางต่อไปนี้แสดงรายการอักขระ 7 บิต ASCII ที่ ไม่ได้อยู่ในช่วงจุดโค้ดเฉพาะ อักขระเหล่านี้อยู่ในช่วง 0x40 ถึง 0x7F

ตารางที่ 3. อักขระ ASCII อื่นๆ

ชื่อสัญลักษณ์	ค่า Hex	Glyph	ชื่อสัญลักษณ์	ค่า Hex	Glyph
commercial-at	40	@	การเน้น grave	60	`
A	41	A	a	61	a
B	42	B	b	62	b
C	43	C	c	63	c
D	44	D	d	64	d
E	45	E	e	65	e
F	46	F	f	66	f
G	47	G	g	67	g
H	48	H	h	68	h
I	49	I	i	69	i
J	4A	J	j	6A	j
K	4B	K	k	6B	k
L	4C	L	l	6C	l
M	4D	M	m	6D	m
N	4E	N	n	6E	n
O	4F	O	o	6F	o
P	50	P	p	70	p
Q	51	Q	q	71	q
R	52	R	r	72	r
S	53	S	s	73	s
T	54	T	t	74	t
U	55	U	u	75	u
V	56	V	v	76	v

ตารางที่ 3. อักขระ ASCII อื่นๆ (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex	Glyph	ชื่อสัญลักษณ์	ค่า Hex	Glyph
W	57	W	w	77	w
X	58	X	x	78	x
Y	59	Y	y	79	y
Z	5A	Z	z	7A	z
วงเล็บซ้าย	5B	[	ปีกกาซ้าย	7B	{
backslash	5C	\	เส้นแนวตั้ง	7C	|
วงเล็บขวา	5D	]	ปีกกาขวา	7D	}
circumflex	5E	^	tilde	7E	~
ขีดเส้นใต้	5F	_	del	7F	

กลยุทธ์ของชุดโค้ด

แต่ละโลแคลในระบบจะกำหนดชุดโค้ดที่โลแคลจะใช้ และวิธีการจัดดำเนินการอักขระภายในชุดโค้ด เนื่องจาก สามารถติดตั้งได้หลายโลแคลบนระบบ ดังนั้นผู้ใช้ที่แตกต่างกันบนระบบ จึงสามารถใช้ชุดโค้ดได้หลายชุด แม้ว่าคุณสามารถตั้งค่าคอนฟิกระบบให้โลแคลใช้ชุดโค้ดที่แตกต่างกันได้ แต่ยูทิลิตี้ระบบทั้งหมดจะ สมมุติว่าระบบกำลังรันอยู่ภายใต้ชุดโค้ดเดียว

คำสั่งส่วนใหญ่ไม่ทราบถึงชุดโค้ดแท้จริงที่โลแคล กำลังใช้งานอยู่ ข้อมูลเกี่ยวกับชุดโค้ดถูกซ่อนไว้โดยรูทีนย่อยไลบรารีที่เป็นอิสระ จากชุดโค้ด (ไลบรารี globalization) ซึ่งส่งผ่านข้อมูล ไปยังรูทีนย่อยที่พึ่งพาชุดโค้ด

เนื่องจากโปรแกรมจำนวนมากอาศัย ASCII ชุดโค้ดทั้งหมดรวมถึงชุดโค้ด 7 บิต ASCII จึงนับเป็นชุดย่อยที่เหมาะสม เนื่องจากชุดโค้ด 7 บิต ASCII เป็นชุดย่อยที่เหมาะสมสำหรับชุดโค้ดที่สนับสนุนทั้งหมด ดังนั้นในบางครั้ง อักขระใน ชุดจึงมีการอ้างอิงเป็น *ชุดอักขระที่ใช้ได้หลายระบบ* ชุดโค้ด 7 บิต ASCII สร้างขึ้นตามคำนิยาม ISO646 และมีอักขระควบคุม อักขระวรรคตอน ตัวเลข (0-9), และตัวอักษรภาษา อังกฤษในตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก

โครงสร้างชุดโค้ด

ชุดโค้ดแต่ละชุดแบ่งออกเป็นพื้นที่สำคัญได้ดังต่อไปนี้:

- Graphic Left (GL): Columns 0-7
- Graphic Right (GR): Columns 8-F

สองคอลัมน์แรกของแต่ละชุดโค้ดถูกสงวนไว้ตามมาตรฐาน International Organization for Standardization (ISO) สำหรับอักขระควบคุม มีการใช้คำว่า C0 และ C1 เพื่อแสดงถึงอักขระควบคุมสำหรับพื้นที่ กราฟิกซ้ายและกราฟิกขวา ตามลำดับ

หมายเหตุ: ชุดโค้ด IBM PC ใช้พื้นที่ควบคุม C1 เพื่อเข้ารหัสอักขระกราฟิก

อีกหกคอลัมน์ที่เหลือใช้เพื่อเข้ารหัสอักขระกราฟิก อักขระกราฟิกจัดว่าเป็นอักขระที่พิมพ์ได้ ในขณะที่ อักขระควบคุมมีการใช้โดยอุปกรณ์และแอ็พพลิเคชันต่างๆ เพื่อบ่งชี้ ฟังก์ชันพิเศษบางอย่าง

อักขระควบคุม

ชื่อ	ค่า	คำอธิบาย
NUL	00	Null
SOH	01	เริ่มต้นส่วนหัว
STX	02	เริ่มต้นข้อความ
ETX	03	สิ้นสุดข้อความ
EOT	04	สิ้นสุดการส่งผ่าน
ENQ	05	การสอบถาม
ACK	06	การตอบรับ
BEL	07	Bell
BS	08	Backspace
HT	09	แท็บแนวนอน
LF	0A	ป้อนบรรทัด
VT	0B	แท็บแนวตั้ง
FF	0C	ป้อนกระดาษ
CR	0D	ปัดแคร่
SO	0E	Shift Out
SI	0F	Shift In
DLE	10	ออกจากลิงก์ข้อมูล
DC1	11	การควบคุมอุปกรณ์ 1
DC2	12	การควบคุมอุปกรณ์ 2
DC3	13	การควบคุมอุปกรณ์ 3
DC4	14	การควบคุมอุปกรณ์ 4
NAK	15	ไม่ตอบรับ
SYN	16	Synchrous idle
ETB	17	สิ้นสุดบล็อกการส่งผ่าน
CAN	18	ยกเลิก
EM	1	สิ้นสุดสื่อ
SUB	1A	อักขระทดแทน
ESC	1B	อักขระ Escape
IS4	1C	ตัวแบ่งข้อมูลสี่
IS3	1D	ตัวแบ่งข้อมูลสาม
IS2	1E	ตัวแบ่งข้อมูลสอง

ชื่อ	ค่า	คำอธิบาย
IS1	1F	ตัวแบ่งข้อมูลหนึ่ง

อักขระกราฟิก

ชุดโค้ดแต่ละชุดสามารถแบ่งออกเป็นชุดอักขระได้หนึ่งชุดขึ้นไป โดยอักขระแต่ละตัวจะมีค่าที่เข้ารหัส ไม่ซ้ำกัน มาตรฐาน ISO สงวนพื้นที่ไว้หกคอลัมน์สำหรับอักขระที่เข้ารหัส และไม่อนุญาตให้เข้าโค้ดอักขระกราฟิกในคอลัมน์ อักขระควบคุม

ชุดโค้ดไบต์เดียวและหลายไบต์

ชุดโค้ดที่ใช้ครบทั้ง 8 บิตของไบต์สามารถสนับสนุนภาษายุโรป ตะวันออกกลาง และภาษาตัวอักษรอื่นๆ ชุดโค้ดดังกล่าวเรียกว่า ชุดโค้ดไบต์เดียว ชุดโค้ดไบต์เดียวมีขีดจำกัดของ อักขระที่เข้ารหัสจำนวน 191 อักขระ ไม่รวมอักขระควบคุม

ภาษาที่ต้องใช้มากกว่า 191 อักขระจะใช้ อักขระไบต์เดียว (8 บิต) และอักขระหลายไบต์ (มากกว่า 8 บิต) ผสมกัน ระบบสามารถสนับสนุนจำนวนบิตในการเข้ารหัสอักขระได้ไม่จำกัด จำนวน

ชุดโค้ด ISO

ชุดโค้ดที่แสดงรายการในหัวข้อต่อไปนี้สร้างขึ้นตามข้อมูล คำนิยามที่กำหนดโดย International Organization for Standardization (ISO)

ISO646-IRV

ชุดโค้ด ISO646-IRV ข้างล่างจะกำหนดชุดโค้ด ที่ใช้สำหรับการประมวลผลข้อมูลโดยใช้การเข้ารหัส 7 บิต ชุดอักขระ ที่เชื่อมโยงกับชุดโค้ดนี้ได้มาจากอักขระ ASCII

Second Hexadecimal Digit	First Hexadecimal Digit															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	BLANK (SPACE)	0	@	P	`	p								
1	SOH	DC1	!	1	A	Q	a	q								
2	STX	DC2	"	2	B	R	b	r								
3	ETX	DC3	#	3	C	S	c	s								
4	EOT	DC4	$	4	D	T	d	t								
5	ENQ	NAK	%	5	E	U	e	u								
6	ACK	SYN	&	6	F	V	f	v								
7	BEL	ETB	'	7	G	W	g	w								
8	BS	CAN	(	8	H	X	h	x								
9	HT	EM	)	9	I	Y	i	y								
A	LF	SUB	*	:	J	Z	j	z								
B	VT	ESC	+	;	K	[	k	{								
C	FF	IS4	,	<	L	\	l	|								
D	CR	IS3	-	=	M	]	m	}								
E	SO	IS2	.	>	N	^	n	~								
F	SI	IS1	/	?	O	_	o	△								

ตระกูล ISO8859

ISO8859 คือตระกูลที่ใช้การเข้ารหัสไบต์เดียว และเข้ากันได้กับเทคนิคการขยายโค้ด ISO อื่นๆ, American National Standards Institute (ANSI), และ European Computer Manufacturer's Association (ECMA) การเข้ารหัส ISO8859 กำหนดตระกูลของชุดโค้ด โดยที่แต่ละสมาชิกมีชุดอักขระเฉพาะของตนเอง ชุดโค้ด 7 บิต ASCII เป็นชุดย่อยที่เหมาะสมของแต่ละชุดโค้ดในตระกูล ISO8859

ในขณะที่ชุดโค้ด ASCII จะกำหนดลำดับของตัวอักษรภาษาอังกฤษ อักขระ Graphic Right (GR) ไม่มีการจัดลำดับตามภาษาเฉพาะ ใดๆ โลแคลจะกำหนดการจัดลำดับเฉพาะภาษา

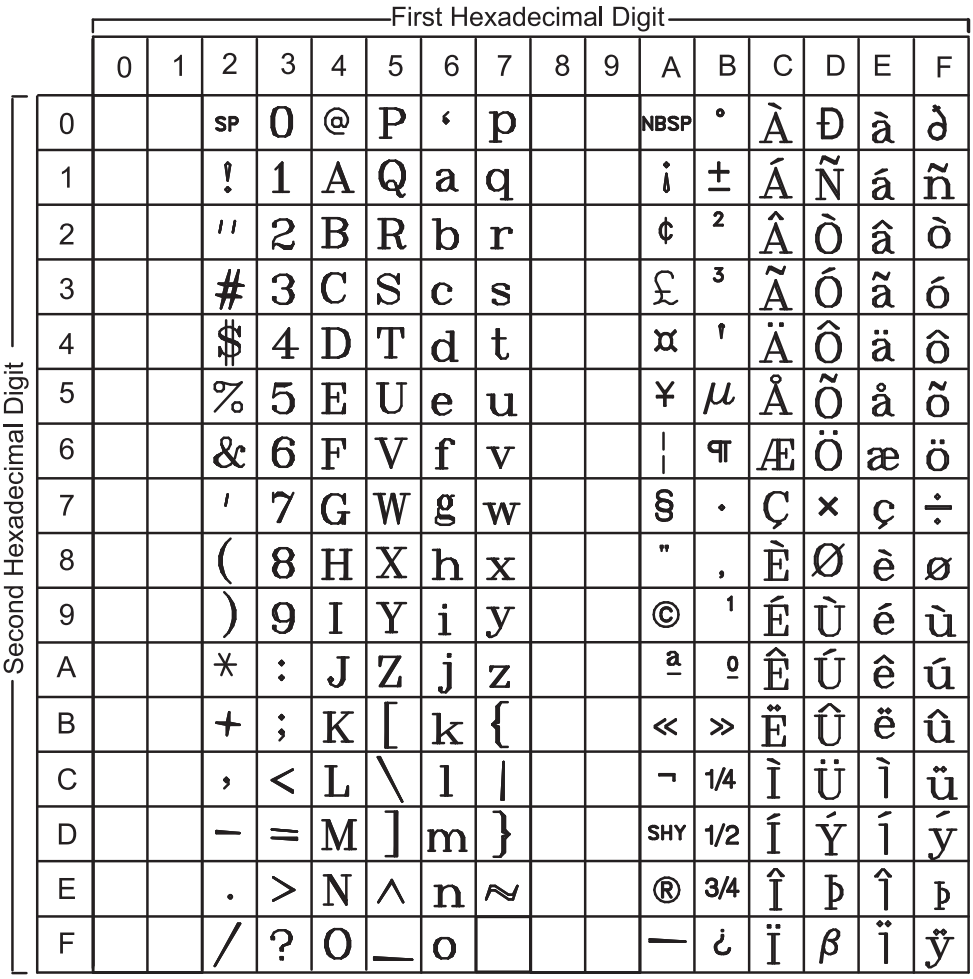
ชุดโค้ดแต่ละชุดประกอบด้วยชุดอักขระ ASCII บวกกับชุดอักขระเฉพาะของตนเอง รูปภาพการเข้ารหัส ISO8859 แสดงแบบแผนการเข้ารหัสทั่วไปของ ISO8859

อักขระที่เข้ารหัส	จุดโค้ด	คำอธิบาย	จำนวน
000xxxxx	00–1F	ตัวควบคุม	32
00100000	20	พื้นที่ว่าง	1
0xxxxxxx	21–7E	7 บิต	94
01111111	7F	ลบ	1
100xxxxx	80–9F	ตัวควบคุม	32
10100000	A0	พื้นที่ว่างต่อเนื่อง	1
1xxxxxxx	A1–F	8 บิต	96

ชุดโค้ด ISO8859-1

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-1 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด ISO8859-1 สำหรับการแสดงข้อความของชุดโค้ดนี้ให้ดู “ISO8859-1” ในหน้า 219



ชุดโค้ด ISO8859-2

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-2 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด ISO8859-2 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “ISO8859-2” ในหน้า 222

Second Hexadecimal Digit \ First Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	°	Ŕ	Đ	ŕ	đ
1			!	1	A	Q	a	q			Ą	ą	Á	Ń	á	ń
2			”	2	B	R	b	r			˘	˛	Â	Ň	â	ň
3			#	3	C	S	c	s			Ł	ł	Ă	Ó	ă	ó
4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5			%	5	E	U	e	u			Ľ	ľ	Ĺ	Ő	Í	ő
6			&	6	F	V	f	v			Ś	ś	Ć	Ö	ć	ö
7			’	7	G	W	g	w			§	ˇ	Ç	×	ç	÷
8			(	8	H	X	h	x			¨	¸	Č	Ř	č	ř
9			)	9	I	Y	i	y			Š	š	É	Ů	é	ů
A			*	:	J	Z	j	z			Ş	ş	Ę	Ú	ę	ú
B			+	;	K	[	k	{			Ť	ť	Ë	Ű	ë	ű
C			,	<	L	\	l	|		Ź	ź	Ě	Ü	ě	ü
D			−	=	M	]	m	}			SHY	˝	Í	Ý	í	ý
E			.	>	N	^	n	~			Ž	ž	Î	Ţ	î	ţ
F			/	?	O	_	o				Ż	ż	Ď	ß	ď	˙

ชุดโค้ด ISO8859-4

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-4 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด ISO8859-4 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “ISO8859-4” ในหน้า 225

Second Hexadecimal Digit \ First Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	°	Ā	Đ	ā	đ
1			!	1	A	Q	a	q			Ą	ą	Á	Ŋ	á	ŋ
2			"	2	B	R	b	r			ĸ	˛	Â	Ō	â	ō
3			#	3	C	S	c	s			Ŗ	ŗ	Ã	Ķ	ã	ķ
4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5			%	5	E	U	e	u			Ĩ	ĩ	Å	Õ	å	õ
6			&	6	F	V	f	v			Ļ	ļ	Æ	Ö	æ	ö
7			'	7	G	W	g	w			§	ˇ	Į	×	į	÷
8			(	8	H	X	h	x			¨	¸	Č	Ø	č	ø
9			)	9	I	Y	i	y			Š	š	É	Ų	é	ų
A			*	:	J	Z	j	z			Ē	ē	Ę	Ú	ę	ú
B			+	;	K	[	k	{			Ģ	ģ	Ë	Û	ë	û
C			,	<	L	\	l	|		Ŧ	ŧ	Ė	Ü	ė	ü
D			-	=	M	]	m	}			SHY	Ŋ	Í	Ũ	í	ũ
E			.	>	N	^	n	≈			Ž	ž	Î	Ū	î	ū
F			/	?	O	_	o				¯	ŋ	Ī	ß	ī	•

ชุดโค้ด ISO8859-5

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-5 มีการรวมไว้ในส่วนนี้

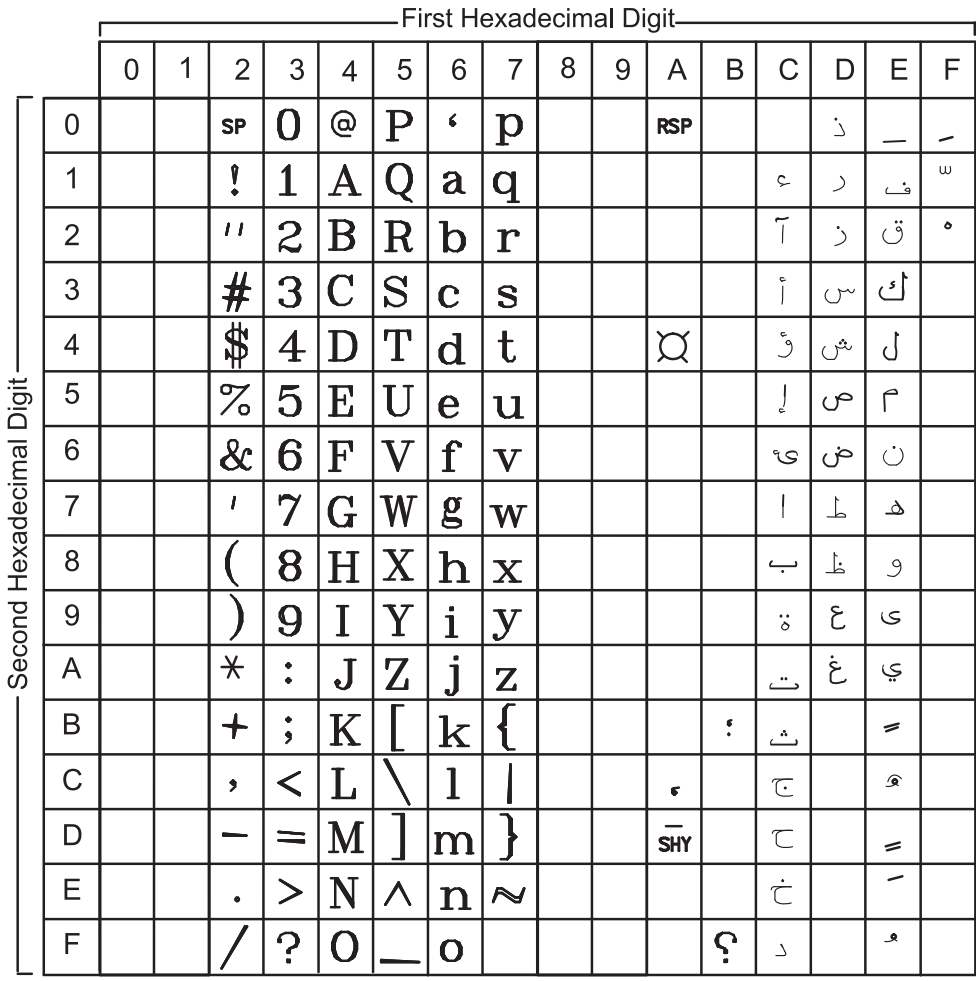
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด ISO8859-5 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ให้ดู “ISO8859-5” ในหน้า 228

Second Hexadecimal Digit \ First Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	А	Р	а	р	№
1			!	1	A	Q	a	q			Ё	Б	С	б	с	ё
2			"	2	B	R	b	r			Ђ	В	Т	в	т	ђ
3			#	3	C	S	c	s			Ѓ	Г	У	г	у	ѓ
4			$	4	D	T	d	t			Є	Д	Ф	д	ф	є
5			%	5	E	U	e	u			S	Е	Х	е	х	s
6			&	6	F	V	f	v			І	Ж	Ц	ж	ц	і
7			'	7	G	W	g	w			Ї	З	Ч	з	ч	ї
8			(	8	H	X	h	x			Ј	И	Ш	и	ш	ј
9			)	9	I	Y	i	y			Љ	Й	Щ	й	щ	љ
A			*	:	J	Z	j	z			Њ	К	Ъ	к	ъ	њ
B			+	;	K	[	k	{			Ћ	Л	Ы	л	ы	ћ
C			,	<	L	\	l	|		Ќ	М	Ь	м	ь	ќ
D			-	=	M	]	m	}			SHY	Н	Э	н	э	§
E			.	>	N	^	n	~			Ў	О	Ю	о	ю	ў
F			/	?	O	_	o				Џ	П	Я	п	я	џ

ชุดโค้ด ISO8859-6

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-6 มีการรวมไว้ในส่วนนี้

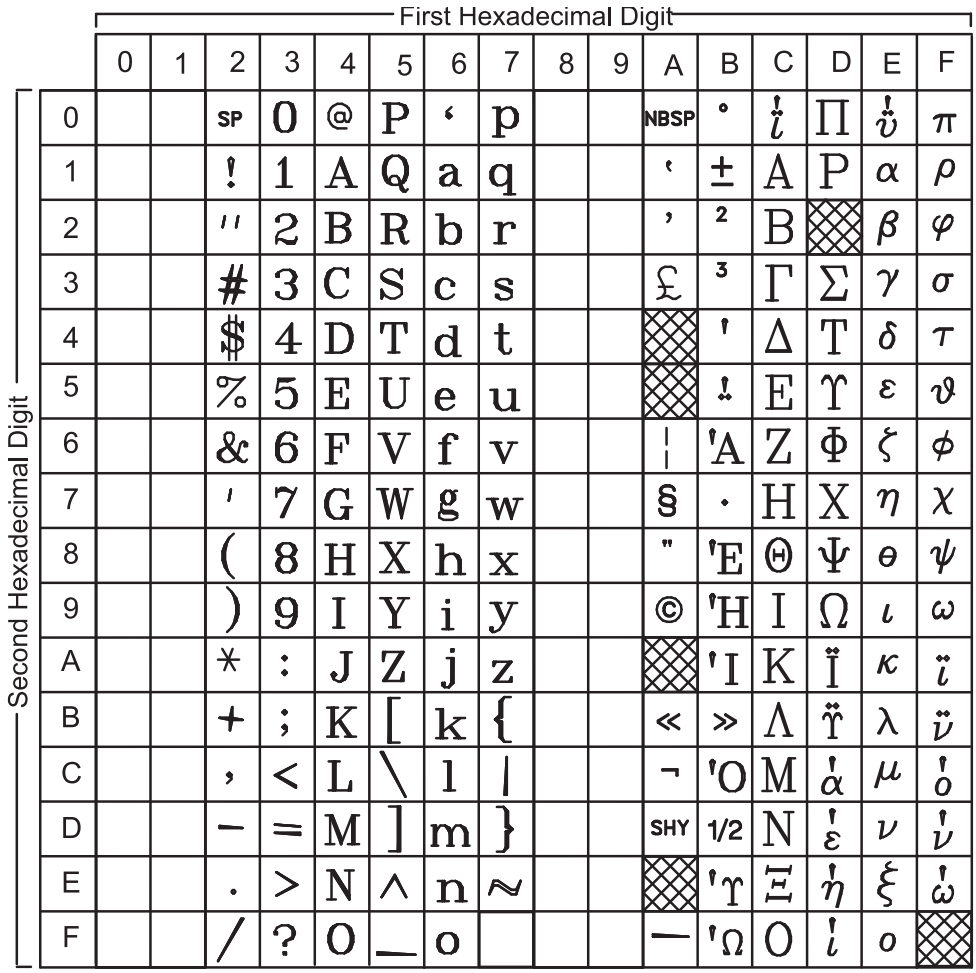
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด ISO8859-6 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “ISO8859-6” ในหน้า 232



ชุดโค้ด ISO8859-7

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-7 มีการรวมไว้ในส่วนนี้

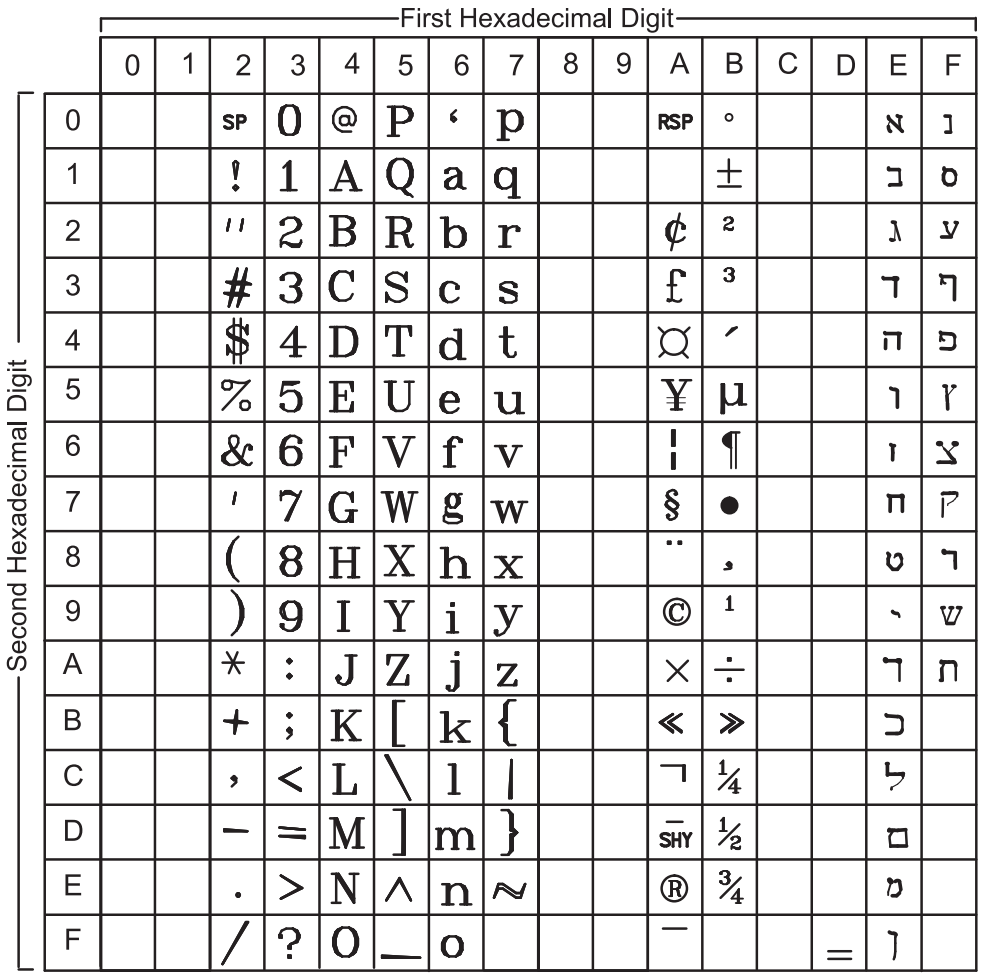
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด ISO8859-7 ชุดโค้ดนี้ประกอบขึ้นด้วยชุดอักขระ ASCII บวกกับชุดอักขระเฉพาะของตนเอง สำหรับการแสดงข้อความของ ชุดโค้ดนี้ให้ดู “ISO8859-7” ในหน้า 233



ชุดโค้ด ISO8859-8

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-8 มีการรวมไว้ในส่วนนี้

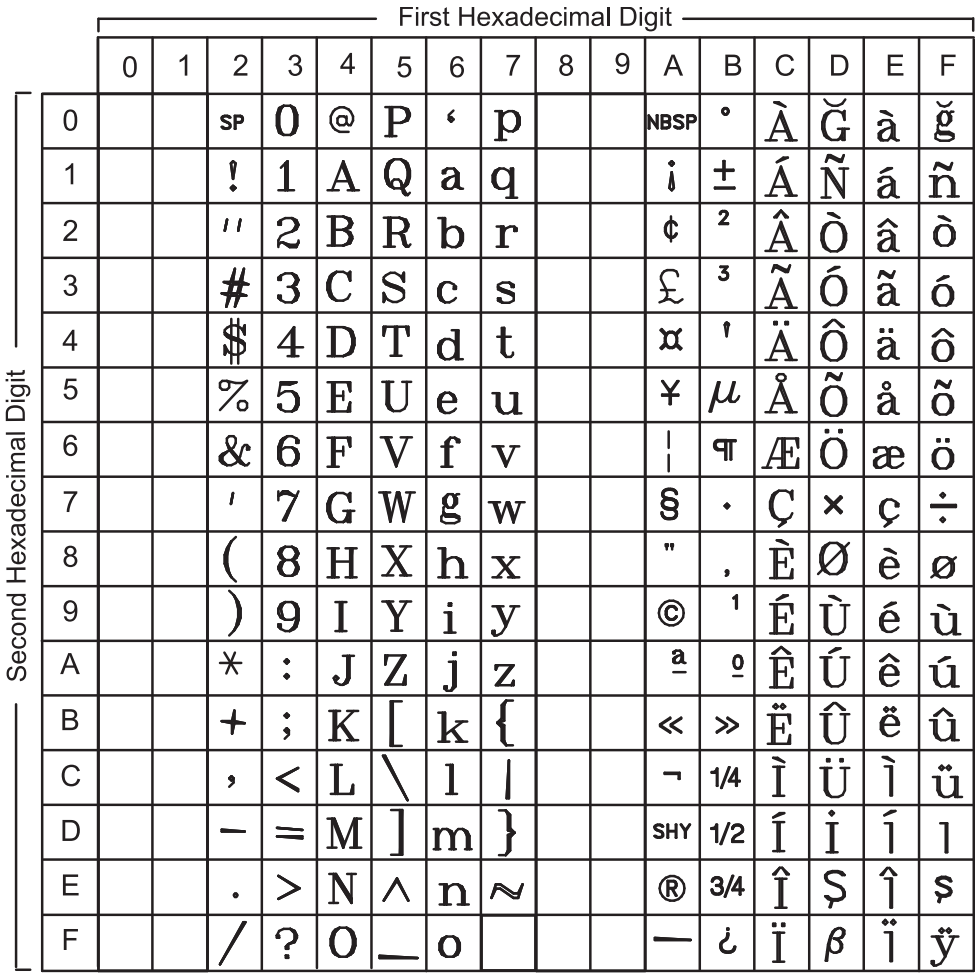
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด ISO8859-8 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “ISO8859-8” ในหน้า 237



ชุดโค้ด ISO8859-9

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-8 มีการรวมไว้ในส่วนนี้

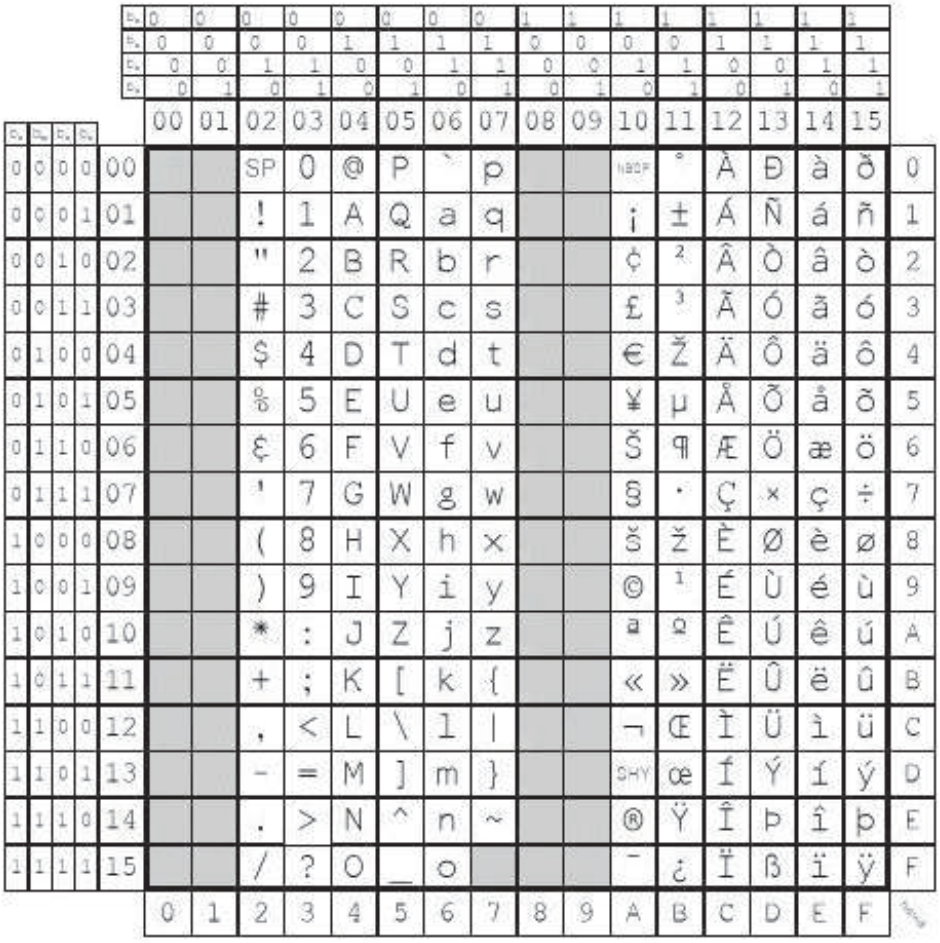
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด ISO8859-9 สำหรับการแสดงข้อความของชุดโค้ดนี้ ให้ดู “ISO8859-9” ในหน้า 239



ชุดโค้ด ISO8859-15

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด ISO8859-15 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด ISO8859-15 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “ISO8859-15” ในหน้า 242



แบบแผนการเข้ารหัส Extended UNIX code (EUC)

แบบแผนการเข้ารหัส EUC กำหนดชุดของกฎการเข้ารหัส ที่สามารถสนับสนุนชุดอักขระได้หนึ่งถึงสี่ชุด กฎการเข้ารหัส ทำงานโดยใช้คำนิยาม ISO2022 สำหรับการเข้ารหัสข้อมูล 7 บิตและ 8 บิต แบบแผนการเข้ารหัส EUC ใช้อักขระควบคุมเพื่อระบุชุดอักขระบางชุด ตารางต่อไปนี้แสดงโครงสร้างพื้นฐาน ของการเข้ารหัส EUC ทั้งหมด

EUC	อักขระที่เข้ารหัส
CS0	0xxxxxxx
CS1	1xxxxxxx 1xxxxxxx 1xxxxxxxx 1xxxxxxx 1xxxxxxxx 1xxxxxxx ...
CS2	10001110 1xxxxxxx 10001110 1xxxxxxx 1xxxxxxxx 10001110 1xxxxxxx 1xxxxxxxx 1xxxxxxxx ...

EUC	อักขระที่เข้ารหัส
CS3	10001111 1xxxxxxx 10001111 1xxxxxxx 1xxxxxxx 10001111 1xxxxxxx 1xxxxxxx 1xxxxxxx ...

คำว่า *EUC* หมายถึงกฎการเข้ารหัสทั่วไปเหล่านี้ ชุดโค้ดที่ใช้ EUC จะสอดคล้องกับกฎการเข้ารหัส EUC แต่ยังระบุชุดอักขระเฉพาะที่เชื่อมโยงกับอินสแตนซ์เฉพาะด้วย ตัวอย่างเช่น IBM-eucJP สำหรับภาษาญี่ปุ่นจะอ้างอิงการเข้ารหัสของอักขระ Japanese Industrial Standard ตามกฎการเข้ารหัส EUC

ชุดแรก (CS0) ประกอบด้วยชุดอักขระ ISO646 เสมอ ชุดอื่น ทั้งหมดต้องมีบิตที่สำคัญที่สุด (MSB) ซึ่งตั้งค่าเป็น 1 และสามารถใช้ไบต์จำนวนเท่าไรก็ได้เพื่อเข้ารหัสอักขระ นอกจากนี้ อักขระทุกตัวภายในชุดต้องมีลักษณะดังต่อไปนี้:

- จำนวนไบต์เท่ากันสำหรับการเข้ารหัสอักขระทั้งหมด
- ความกว้างในการแสดงคอลัมน์เท่ากัน (จำนวนของคอลัมน์บนเทอร์มินัลความกว้าง คงที่)

อักขระทั้งหมดในชุดที่สาม (CS2) จะมีอักขระควบคุม SS2 (single-shift 2, 0x8e) นำหน้าเสมอ ชุดโค้ดที่สอดคล้องกับ EUC ไม่ได้ใช้อักขระควบคุม SS2 นอกเหนือจากการใช้เพื่อระบุ ชุดที่สาม

อักขระทั้งหมดในชุดที่สี่ (CS3) จะมีอักขระควบคุม SS3 (single-shift 3, 0x8f) นำหน้าเสมอ ชุดโค้ดที่สอดคล้องกับ EUC ไม่ได้ใช้อักขระควบคุม SS3 นอกเหนือจากการใช้เพื่อระบุ ชุดที่สี่

IBM-eucJP

EUC สำหรับภาษาญี่ปุ่นเป็นการเข้ารหัสที่ประกอบด้วย อักขระไบต์เดียวและอักขระหลายไบต์ การเข้ารหัสดำเนินการตามข้อมูล ISO2022, Japanese Industrial Standard (JIS), และคำนิยาม EUC

ชุดโค้ด IBM-eucJP ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
JISCII	ชุดอักขระ JISX0201 กราฟิกซ้าย
JISX0201.1976	ชุดอักขระ Katakana/Hiragana กราฟิกขวา
JISX0208.1983	ชุดอักขระ Kanji ระดับ 1 และ 2
IBM-udcJP	อักขระที่ผู้ใช้ IBM กำหนดได้

ชุดอักขระ IBM-eucJP ยังสามารถสนับสนุนข้อมูลดังต่อไปนี้ ด้วย:

ชุดอักขระ	คำอธิบาย
JISX0212.1990	Supplemental Kanji

ชุดโค้ด IBM-eucJP มีการเข้ารหัสดังนี้:

- CS0 จะแม็พอักขระ JISX0201 กราฟิกซ้าย โดยเริ่มต้นที่ตำแหน่ง 0x00
- CS1 จะแม็พชุดอักขระ JISX0208 โดยเริ่มต้นที่ตำแหน่ง 0xa1xa1 ตำแหน่ง 0xf5a1 ถึง 0xfefe (940 อักขระ) ใน CS1 ถูกสงวนไว้ เป็นพื้นที่อักขระที่ผู้ใช้กำหนดหลัก
- CS2 จะแม็พ JISX0201 กราฟิกขวา โดยเริ่มต้นที่ตำแหน่ง 0x8ea1

- CS3 สามารถแม็พ JISX0212 โดยเริ่มต้นที่ตำแหน่ง 0x8fa1a1 ตำแหน่ง 0x8ff5a1 ถึง 0x8ffefe ใน CS3 (940 อักขระ) ถูกสงวนไว้ เป็นพื้นที่อักขระที่ผู้ใช้กำหนดรอง ตำแหน่ง 0x8feea1 ถึง 0x8ff4fe ใน CS3 (658 อักขระ) ถูกสงวนไว้สำหรับการใช้ของ ระบบในอนาคต ดังนั้น ผู้ใช้จึงไม่ควรใช้พื้นที่นี้

IBM-eucCN

EUC สำหรับภาษาจีนแบบง่ายคือการเข้ารหัสที่ประกอบด้วย อักขระที่มี 1 หรือ 2 ไบต์ การเข้ารหัส EUC ดำเนินการ ตามข้อมูล ISO2022, GB2312 ตามที่กำหนดโดยสาธารณรัฐประชาชนจีน และคำนิยามอักขระหลายไบต์เฉพาะของผู้ผลิต

GB2312 ปัจจุบันกำหนดอักขระภาษาจีนแบบง่ายจำนวน 6,763 อักขระ และ 682 สัญลักษณ์ IBM-eucCN สร้างขึ้นจากหลักการระนาบเดียว ที่มีข้อมูลมากถึง 94x94 อักขระ ค่าที่เข้ารหัสของอักขระเหล่านี้ อยู่ในช่วงตั้งแต่ 0xa1a1 ถึง 0xfefe

GB2312 มีการแม็พเข้าใน CS1 ของ EUC โดยเฉพาะอย่างยิ่ง IBM-eucCN ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
ISO0646-IRV	ชุดอักขระ 7 บิต ASCII กราฟิกซ้าย
GB2312.1980	มี 7445 อักขระ อยู่ในตำแหน่ง 0xa1a1 ถึง 0xfedf (อักขระที่ผู้ใช้กำหนดบางตัวกระจายอยู่ใน 0xa1a1 ถึง 0xfedf)
IBM-udcCN	กระจายอยู่ใน GB อยู่ในตำแหน่ง Oxa1a1 ถึง Oxfedf ค่าที่แท้จริงคือ: a2a1 -- a2b0 a1e3 -- a2e4 a1ef -- a2f0 a2fd -- a1fe a4f4 -- a4fe a5f7 -- a5fe a6b9 -- a6c0 a6d9 -- a6fe a7c2 -- a7d0 a7f2 -- a7fe a8bb -- a8c4 a8ea -- a9a3 a9f0 -- affe a7fa -- d7fe f8a1 -- fedf
IBM-sbdCN	กระจายอยู่ใน GB อยู่ในตำแหน่ง 0xfee0 ถึง 0xfefe

GB18030

GBK ย่อมาจาก Guo (ประจำชาติ) Biao (มาตรฐาน) Kuo (ส่วนขยาย) GB18030 ขยายคำนิยาม "Industry GB" ประจำชาติ เพื่อรวมอักขระ 20, 902 Han ทั้งหมดที่กำหนดไว้ในรูปแบบ Unicode และ สัญลักษณ์ DBCS เพิ่มเติมซึ่งกำหนดในโค้ด Big-5 (มาตรฐาน PC defacto ภาษาจีนดั้งเดิม) GB18030 กำหนดอักขระ DBCS และสัญลักษณ์ทั้งหมดที่ใช้ในประเทศจีน แผ่นดินใหญ่และในไต้หวัน

โลแคล	ชุดโค้ด	คำอธิบาย
Zh_CN	GB18030	ภาษาจีนแบบง่าย โลแคล GB18030

ช่วงโค้ด	คำ	เครื่องหมาย
A1A1-A9FE	846	GB2312, GB12345 (GBK/1)
A840-A9A0	192	Big5, สัญลักษณ์ (GBK/5)
B0A1-F7FE	6768	GB2312 (GBK/2)
8140-A0FE	6080	GB13000 (GBK/3)
AA40-FEA0	8160	GB13000 (GBK/4)
AAA1-AFFE	564	ผู้ใช้กำหนด 1
F8A1-FEFE	658	ผู้ใช้กำหนด 2
A140-A7A0	672	ผู้ใช้กำหนด 3

IBM-eucTW

EUC สำหรับภาษาจีนดั้งเดิมคือการเข้ารหัสที่ประกอบด้วย อักขระที่มี 1, 2 และ 4 ไบต์ การเข้ารหัส EUC ดำเนินการ ตามข้อมูล ISO2022, Chinese National Standard (CNS) ตามที่กำหนดโดย ไต้หวัน และคำนิยามอักขระหลายไบต์เฉพาะของผู้ผลิต

CNS ปัจจุบันกำหนดอักขระภาษาจีนจำนวน 13,501 อักขระและ 684 สัญลักษณ์ IBM-eucTW สร้างขึ้นตามหลักการ 15 ระนาบ โดยแต่ละระนาบมีอักขระมากถึง 8836 (94x94) อักขระ ค่าที่เข้ารหัสของอักขระเหล่านี้ อยู่ในช่วงตั้งแต่ 0xa1a1 ถึง 0xfefe ในปัจจุบัน มีการกำหนดอักขระสำหรับ 4 ระนาบเท่านั้น ส่วนระนาบที่เหลือถูกสงวนไว้สำหรับ การขยายในอนาคต

15 ระนาบจะมีการแม็พเข้าใน CS1 และ CS2 ของ EUC โดยที่ CS2 ของ EUC ประกอบด้วย 14 ระนาบ โดยเฉพาะอย่างยิ่ง IBM-eucTW ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
ISO646-IRV	ชุดอักขระ 7 บิต ASCII กราฟิกซ้าย
CNS11643.1986-1	ระนาบ 1 มี 6085 อักขระ (5401+684) ระนาบนี้ใช้ตำแหน่ง 0ax1a1-0xc2c1 และ 0xc4a1-0xfdcb
CNS11643.1986-2	ระนาบ 2 มี 7650 อักขระ ระนาบนี้ใช้ตำแหน่ง 0x8ea2a1a1-0x8ea2f2c4
CNS11643.1992-3	ระนาบ 4 มี 7298 อักขระ ระนาบนี้ใช้ตำแหน่ง 0x8ea4a1a1-0x8ea4eedc
IBM-udcTW	ระนาบ 12 มี 6204 อักขระ ระนาบนี้ถูกสงวนไว้ สำหรับพื้นที่อักขระที่ผู้ใช้กำหนด (udc) อยู่ในตำแหน่ง 0x8eaca1a1-0x8ea2f2c4
IBM-sbdTW	ระนาบ 13 มี 325 อักขระ ระนาบนี้ถูกสงวนไว้ สำหรับสัญลักษณ์เฉพาะของผู้ผลิต อยู่ในตำแหน่ง 0xeada1a1-0x8eada4cb

ระนาบ 3-11 คาดว่าจะอยู่ในตำแหน่ง 0x8ea3xxxx ถึง 0x8eabxxxx ระนาบ 14-15 คาดว่าจะอยู่ในตำแหน่ง 0x8eaexxxx ถึง 0x8eafxxxx

Big5

ชุดโค้ดโลแคล Traditional Chinese big5, **Zh_TW**, เป็นชุดโค้ดที่ใช้กันมากที่สุดในฟิลด์ PC ซึ่งใช้ เพื่อสนับสนุนประเทศที่ใช้ภาษาจีนดั้งเดิม

ชุดโค้ด Big5 กำหนด 13056 อักขระและ 1004 สัญลักษณ์ซึ่งรวมถึงสัญลักษณ์ 684 ใน CNS11643.192 รวมถึงสัญลักษณ์เฉพาะ 325 ใน IBM

โลแคล	ชุดโค้ด	คำอธิบาย
Zh_TW	Big5 (IBM-950)	โลแคล Traditional Chinese, Big5

ช่วงโค้ดสำหรับโลแคล Big5:

ช่วงโค้ดสำหรับโลแคล Big5 มีการกำหนดไว้ในตาราง ต่อไปนี้:

แผน	ช่วงโค้ด	คำอธิบาย
1	A140H - A3E0H	สัญลักษณ์และโค้ดควบคุมภาษาจีน
1	A440H - C67EH	อักขระที่ใช้ทั่วไป
2	C940H - F9D5H	อักขระที่ใช้ทั่วไปน้อย
UDF	FA40H - FEFE	อักขระที่ผู้ใช้กำหนด
	8E40H - A0FEH	อักขระที่ผู้ใช้กำหนด
	8140H - 8DFEH	อักขระที่ผู้ใช้กำหนด
	8181H - 8C82H	อักขระที่ผู้ใช้กำหนด
	F9D6H - F9F1H	อักขระที่ผู้ใช้กำหนด

ชุดโค้ด	คำ	ช่วงโค้ด	เครื่องหมาย
พื้นที่ที่ใช้ทั่วไป	5841	A140-C67E	
พื้นที่ที่ใช้ทั่วไปน้อย	7652	C940-F9D5	
พื้นที่เฉพาะ ET (1)	308	C6A1-C878	
พื้นที่เฉพาะ ET (2)	7	C8CD-C8D3	
พื้นที่เฉพาะของ IBM	251	F286-F9A0	ช่วงไบต์ต่ำ 81-A0
พื้นที่ที่ผู้ใช้กำหนด (1)	785	FA40-FEFE	
พื้นที่ที่ผู้ใช้กำหนด (2)	2983	8E40-A0FE	
พื้นที่ที่ผู้ใช้กำหนด (3)	2041	8140-8DFE	
พื้นที่ที่ผู้ใช้กำหนด (4)	354	8181-8C82	ช่วงไบต์ต่ำ 81-AQ
พื้นที่ที่ผู้ใช้กำหนด (5)	41	F9D6-F9FE	

IBM-eucKR

EUC สำหรับภาษาเกาหลีเป็นการเข้ารหัสที่ประกอบด้วย อักขระไบต์เดียวและอักขระหลายไบต์ การเข้ารหัสดำเนินการตามข้อมูล ISO2022, ชุดโค้ด มาตรฐานเกาหลี และคำนิยาม EUC

ชุดโค้ด EUC ภาษาเกาหลีประกอบด้วยกลุ่มอักขระหลัก ดังต่อไปนี้:

- ASCII (ภาษาอังกฤษ)
- Hangul (อักขระภาษาเกาหลี)

ชุดโค้ด Hangul ประกอบด้วยอักขระ Hangul และ Hanja (ภาษาจีน) อักขระ Hangul หนึ่งตัวอาจประกอบด้วยพยัญชนะและสระหลายตัว อย่างไรก็ตาม คำ Hangul ส่วนใหญ่สามารถระบุใน Hanja ได้ อักขระ Hanja แต่ละตัว มีความหมายเป็นของตัวเองและมีลักษณะเฉพาะมากกว่า Hangul

ชุดโค้ด IBM-eucKR ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
ISO646-IRV	ชุดอักขระ 7 บิต ASCII กราฟิกซ้าย
KSC5601.1987-0	ชุดอักขระกราฟิกเกาหลี กราฟิกขวา

ชุดโค้ด IBM PC

ชุดโค้ด IBM PC เป็นชุดโค้ดที่ได้รับการสนับสนุนตั้งแต่ดั้งเดิมบนระบบ IBM PC และ AIX ชุดโค้ด IBM PC จะกำหนดอักขระกราฟิกให้กับพื้นที่ควบคุม Control One (C1) แอ็พพลิเคชันที่ พึ่งพาอักขระควบคุมเหล่านี้ไม่สามารถสนับสนุนชุดโค้ดเหล่านี้ได้

อักขระ ASCII จะมีการเข้ารหัสโดยมีเลขศูนย์ most significant bit (MSB) อยู่ในตำแหน่ง 0x20-0x7e ละตินแบบขยาย 1 ที่รวมเข้ากับชุดอักขระเฉพาะ IBM PC ประกอบขึ้นเป็นชุดอักขระแบบขยายที่มีการเข้ารหัสในตำแหน่ง 0x80-0xfe ตารางต่อไปนี้แสดงที่ตั้งของอักขระควบคุม, ASCII, และอักขระแบบขยาย สำหรับชุดโค้ด IBM-850

อักขระที่เข้ารหัส	จุดโค้ด	คำอธิบาย	จำนวน
000xxxxx	00-1F	ตัวควบคุม	32
00100000	20	พื้นที่ว่าง	1
0xxxxxxx	21-7E	7 บิต	94
01111111	7F	ลบ	1
1xxxxxxx	80-FE	8 บิต	17
11111111	FF	พร้อมกัน	1

ชุดอักขระเฉพาะของ IBM PC รวมถึงต่อไปนี้:

ตารางที่ 4. ชุดอักขระเฉพาะของ IBM PC

สัญลักษณ์	โค้ดส่งคืน
เครื่องหมาย Florin	0x9f
Quarter-hashed	0xb0
Half-hashed	0xb1
Full-hashed	0xb2
แท่งแนวตั้ง	0xb3
กึ่งกลางด้านขวา	0xb4
กึ่งกลางด้านขวาคู่	0xb9
แท่งแนวตั้งคู่	0xba
กล่องมุมด้านขวาบนคู่	0xbb
กล่องมุมด้านขวาล่างคู่	0xbc
กล่องมุมด้านขวาบน	0xbf
กล่องมุมด้านซ้ายล่าง	0xc0

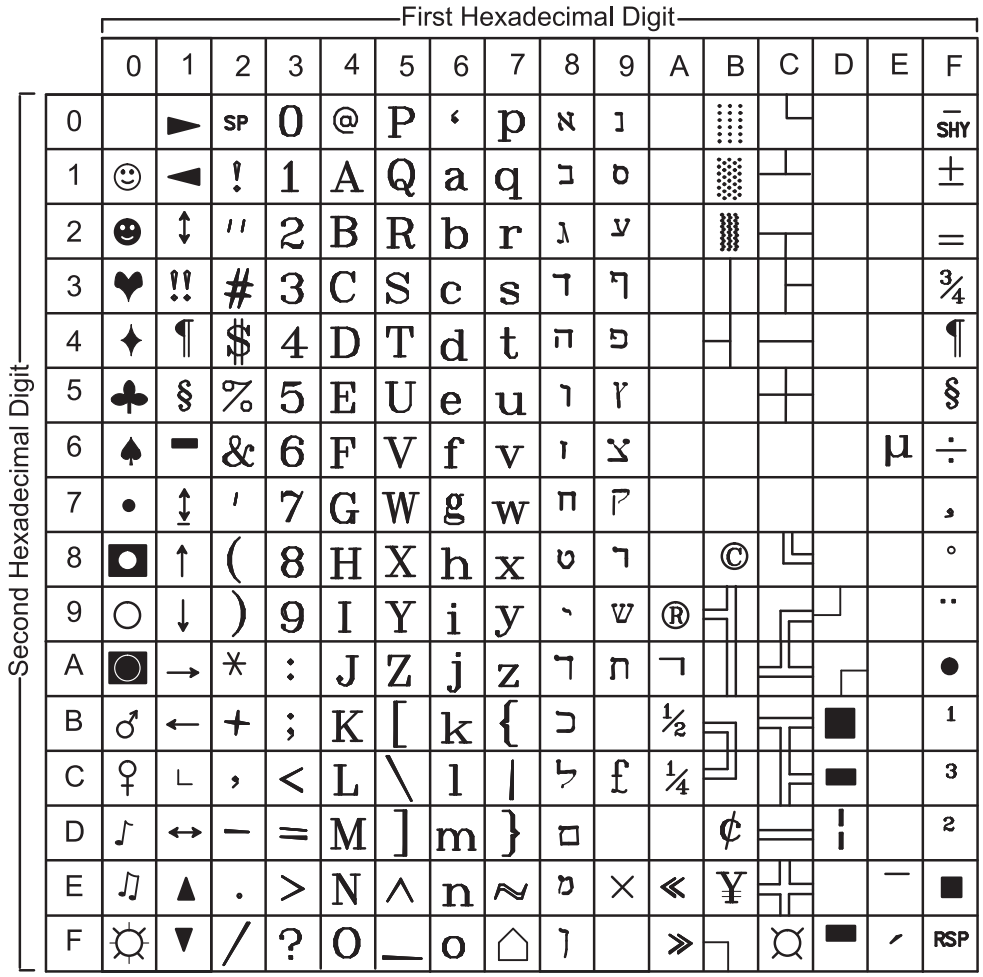
ตารางที่ 4. ชุดอักขระเฉพาะของ IBM PC (ต่อ)

สัญลักษณ์	โค้ดส่งคืน
กึ่งกลางด้านล่าง	0xc1
กึ่งกลางด้านบน	0xc2
กึ่งกลางด้านซ้าย	0xc3
แท่งกล่องศูนย์กลาง	0xc4
จุดตัด	0xc5
กล่องมุมด้านซ้ายล่างคู่	0xc8
กล่องมุมด้านซ้ายบนคู่	0xc9
กึ่งกลางด้านล่างคู่	0xca
กึ่งกลางด้านบนคู่	0xcb
กึ่งกลางด้านซ้ายคู่	0xcc
แท่งกล่องศูนย์กลางคู่	0xcd
จุดตัดคู่	0xce
ไม่มีจุด i ขนาดเล็ก	0xd5
กล่องมุมด้านขวาล่าง	0xd9
กล่องมุมด้านซ้ายบน	0xda
เซลล์อักขระ bright	0xdb
เซลล์อักขระ bright – ครึ่งด้านล่าง	0xde
เซลล์อักขระ bright – ครึ่งด้านบน	0xdf
Overbar	0xee
จุดกึ่งกลาง จุดผลิตภัณฑ์	0xfa
สี่เหลี่ยมทึบแนวตั้ง	0xfe

IBM-856

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM-856 มีการรวมไว้ในส่วนนี้

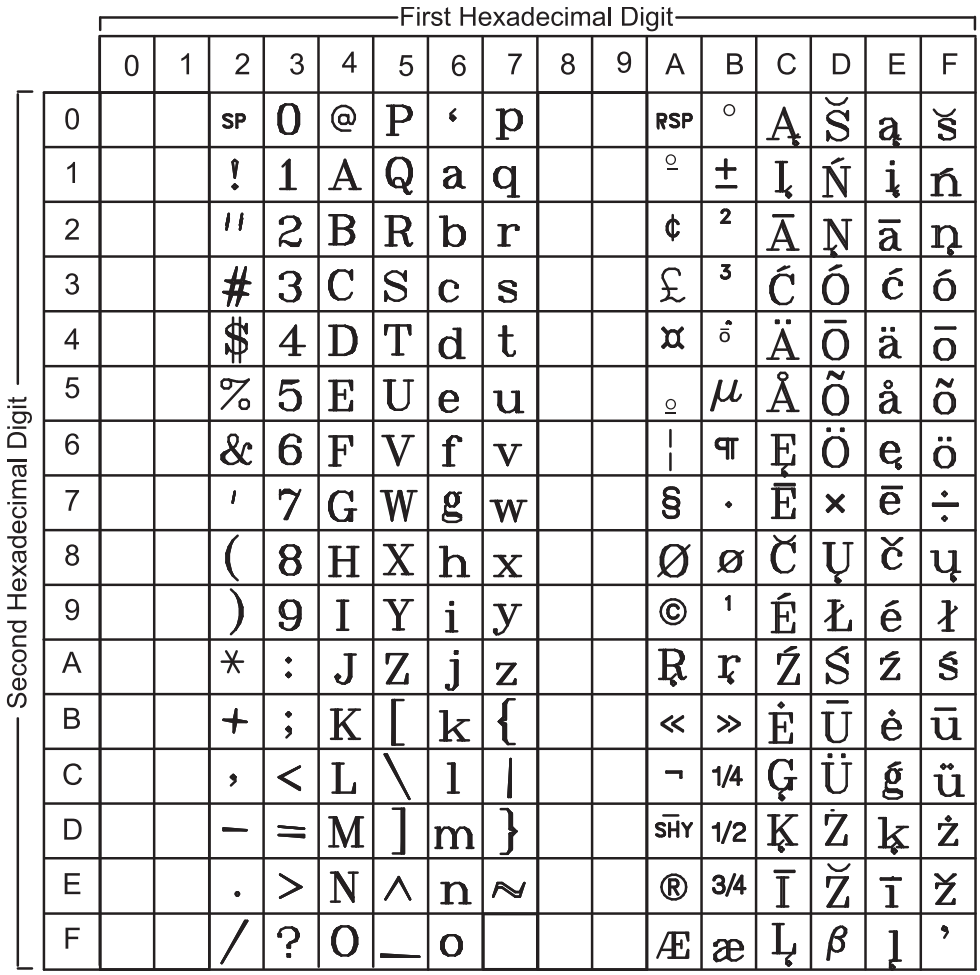
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด IBM-856 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “IBM-856” ในหน้า 245



IBM-921

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM-921 มีการรวมไว้ในส่วนนี้

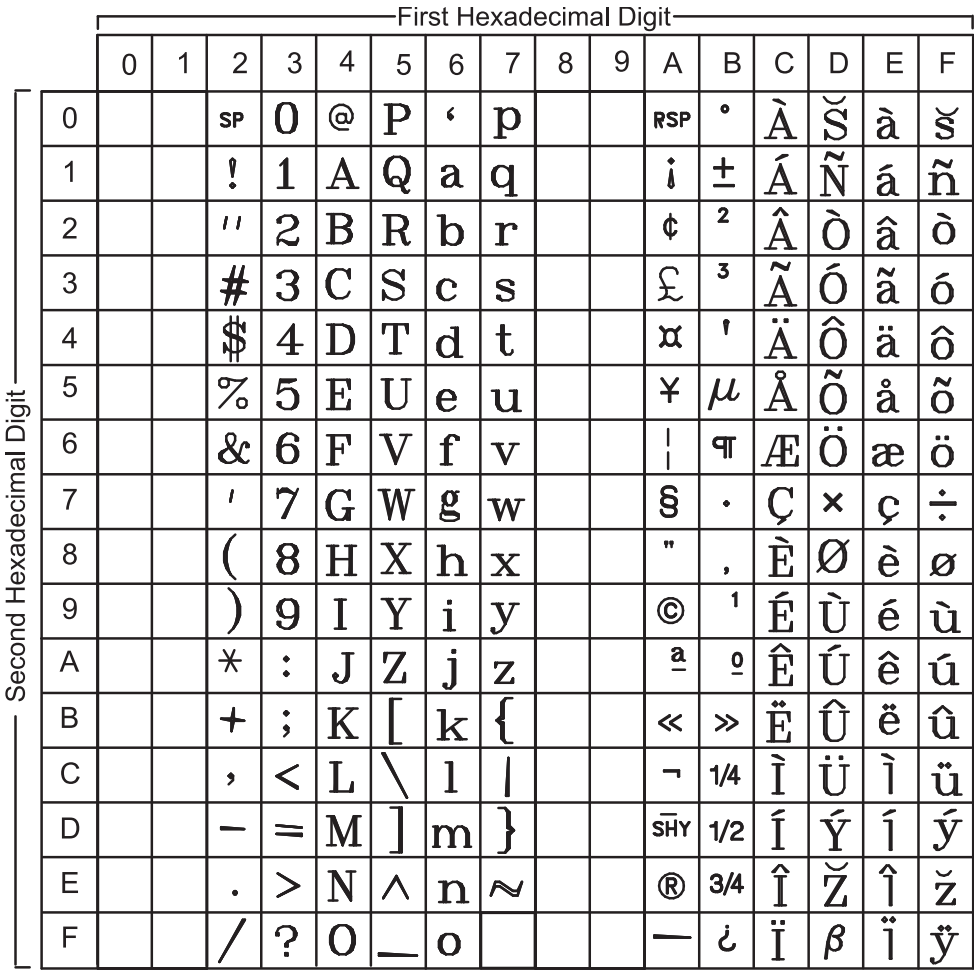
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด IBM-921 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “IBM-921” ในหน้า 248



IBM-922

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM-922 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด IBM-922 สำหรับการแสดงข้อความของชุดโค้ดนี้ ให้ดู “IBM-922” ในหน้า 251



Second Hexadecimal Digit \ First Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	°	À	Š	à	š
1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4			$	4	D	T	d	t			¤	'	Ä	Ô	ä	ô
5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8			(	8	H	X	h	x			"	¸	È	Ø	è	ø
9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C			,	<	L	\	l	|			¬	1/4	Ì	Ü	ì	ü
D			−	=	M	]	m	}			SHY	1/2	Í	Ý	í	ý
E			.	>	N	^	n	~			®	3/4	Î	Ž	î	ž
F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

IBM-943 และ IBM-932

ชุดโค้ด IBM PC ภาษาญี่ปุ่นแต่ละชุดเป็นการเข้ารหัสที่ประกอบด้วยอักขระที่เข้ารหัสแบบไบต์เดียวและแบบหลายไบต์ การเข้ารหัสจะเป็นไปตามชุดโค้ด IBM PCและวางอักขระ JIS ไว้ในตำแหน่งที่ถูกเลื่อนออกไป ซึ่งมีการอ้างอิงเป็น *Shift-JIS* หรือ SJIS

IBM-943 คือชุดโค้ดที่ใหม่กว่า IBM-932 สำหรับโลแคลภาษาญี่ปุ่น IBM-943 เป็นชุดโค้ดที่เข้ากันได้สำหรับสภาวะแวดล้อม Microsoft Windows ภาษาญี่ปุ่น ชุดโค้ดนี้รู้จักกันในชื่อว่า 1983 ordered shift-JIS ความแตกต่างระหว่าง IBM-932 และ IBM-943 มีดังนี้:

- ลำดับ JIS ก่อนหน้านี้ (ลำดับ 1978) ใช้สำหรับ IBM-932 ในขณะที่ ลำดับ JIS ที่ใหม่กว่า (ลำดับ 1983) ใช้สำหรับ IBM-943
- อักขระที่ NEC เลือกมีการเพิ่มลงใน IBM-943
- อักขระที่ IBM เลือก NEC มีการเพิ่มลงใน IBM-943

ชุดโค้ด IBM-932 ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
JISCII	ชุดอักขระ JISX0201 กราฟิกซ้าย
JISX0201.1976	ชุดอักขระ Katakana/Hiragana กราฟิกขวา
JISX0208.1983	ชุดอักขระ Kanji ระดับ 1 และ 2
IBM-udcJP	อักขระที่ผู้ใช้ IBM สามารถกำหนดได้

ชุดโค้ด IBM-943 ประกอบด้วยชุดอักขระดังต่อไปนี้:

ชุดอักขระ	คำอธิบาย
JISCII	ชุดอักขระ JISX0201 กราฟิกซ้าย
JISX0201.1976	ชุดอักขระ Katakana/Hiragana กราฟิกขวา
JISX0208.1990	ชุดอักขระ Kanji ระดับ 1 และ 2
IBM-udcJP	อักขระที่ผู้ใช้ IBM สามารถกำหนดได้ และ อักขระที่ IBM เลือกของ NEC และอักขระที่ NEC เลือก

ไบต์แรกของแต่ละอักขระใช้เพื่อกำหนดจำนวนไบต์ สำหรับอักขระที่กำหนด ค่า 0x20-0x7e และ 0xa1-oxdf ใช้เพื่อเข้ารหัสอักขระ JISX0201 โดยมีข้อยกเว้น ตำแหน่ง 0x81-0x9f และ 0xe0-0xfc ถูกสงวนไว้สำหรับใช้เป็นไบต์แรกของ อักขระแบบหลายไบต์ อักขระ JISX0208 จะมีการแม็พกับค่าหลายไบต์ ซึ่งเริ่มต้นที่ 0x8140 ไบต์ที่สองของอักขระหลายไบต์ อาจมีค่าใดๆ ก็ได้ ตาราง Shift-JIS แสดงตำแหน่งที่ตั้งของอักขระเหล่านี้ ในชุดโค้ด

อักขระที่เข้ารหัส	จุดโค้ด	คำอธิบาย	จำนวน
000xxxxx	00-1f	ตัวควบคุม	32
00100000	20	พื้นที่ว่าง	1
0xxxxxxx	21-7E	7 บิต ASCII	94
01111111	7F	ลบ	1
10000000	80	ไม่ได้กำหนด	1
100xxxxx 01xxxxxx	[81-9F] [40-7E]	ไบต์คู่	1953
100xxxxx 1xxxxxxx	[81-9F] [80-FC]	ไบต์คู่	3975
10100000	A0	ไม่ได้กำหนด	1
1xxxxxxx	A1-DF	7 บิตไบต์เดียว	63
111xxxxx 01xxxxxx	[E0-FC] [40-7E]	ไบต์คู่	1827
111xxxxx 1xxxxxxx	[E0-FC] [80-FC]	ไบต์คู่	3625
11111101	FD	ไม่ได้กำหนด	1
11111110	FE	ไม่ได้กำหนด	1
11111111	FF	ไม่ได้กำหนด	1

ตารางต่อไปนี้แสดงส่วน DBCS ของ IBM-943

จุดโค้ด	คำอธิบาย
[81–84] [40–7E] และ [81–84] [80–F0]	JIS X 0208 (Non–Kanji)
[87] [40–7E] และ [87] [80–F0]	อักขระที่ NEC เลือก
[89–98] [40–7E] และ [88] [9F–F0], [89–97] [80–F0], [98] [80–9F]	JIS X0208 (Kanji ระดับ 1)
[99–9F] [40–7E] และ [98] [9F–F0], [99–9F] [80–F0]	JIS X0208 (Kanji ระดับ 2)
[E0–EA] [40–7E] และ [E0–EA] [80–F0]	JIS X0208 (Kanji ระดับ 2)
[ED–EE] [40–7E] และ [ED–EE] [80–F0]	อักขระที่ IBM เลือกของ NEC
[F0–F9] [40–7E] และ [F0–F9] [80–F0]	อักขระที่ผู้ใช้กำหนด
[FA] [40–5C]	อักขระที่ IBM เลือก (ที่ไม่ใช่ Kanji)
[FA] [5C–7E], [FB–FC] [40–7E] และ [FA–FC] [80–F0]	อักขระที่ IBM เลือก (Kanji)

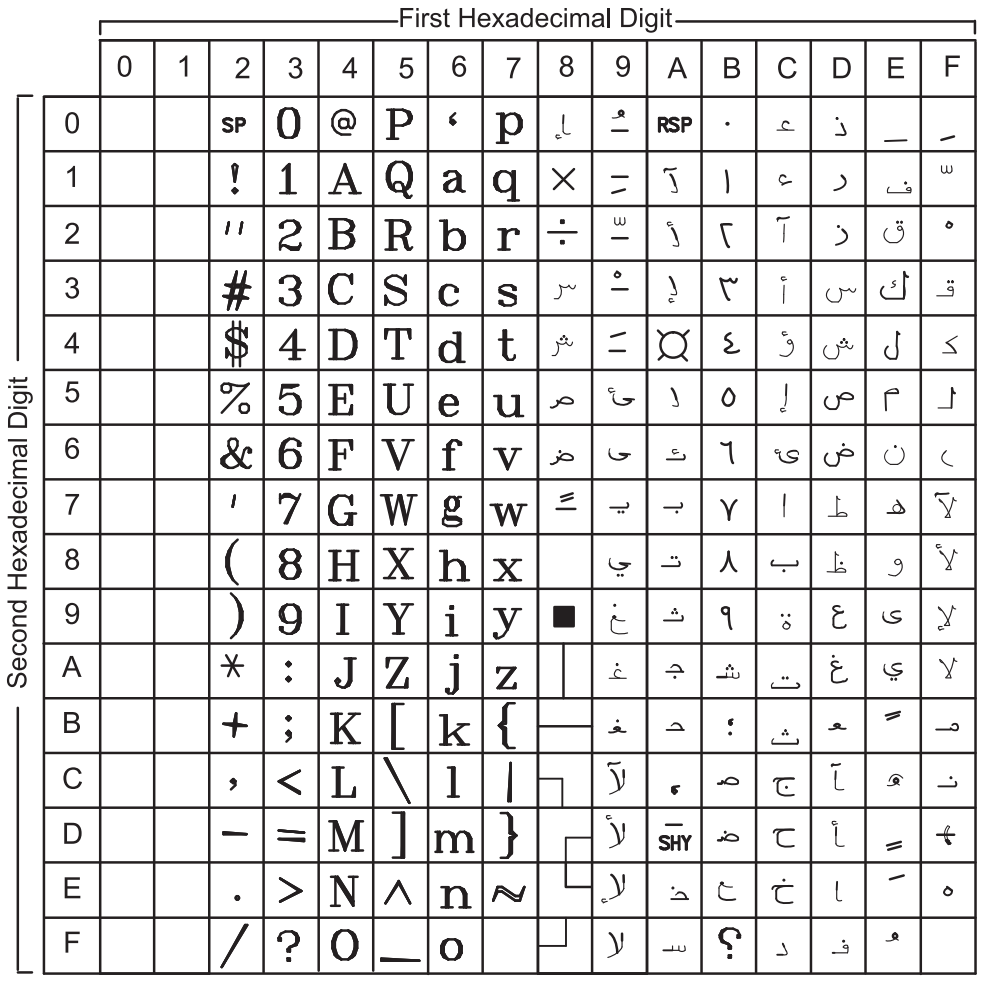
ตารางต่อไปนี้แสดงส่วน DBCS ของ IBM–932

จุดโค้ด	คำอธิบาย
[81–98] [40–7E] และ [81–97] [80–FC], [98] [80–9F]	JIS X 0208 (Kanji ระดับ 1)
[99–9F] [40–7E] และ [98] [9F–FC], [99–9F] [80–FC]	JIS X 0208 (Kanji ระดับ 2)
[E0–EF] [40–7E] และ [E0–EF] [80–FC]	JIS X 0208 (Kanji ระดับ 2)
[F0–F9] [40–7E] และ [F0–F9] [80–FC]	อักขระที่ผู้ใช้กำหนด
[FA–FC] [40–7E] และ [FA–FC] [80–FC]	อักขระที่ IBM เลือก

IBM-1046

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM–1046 มีการรวมไว้ในส่วนนี้

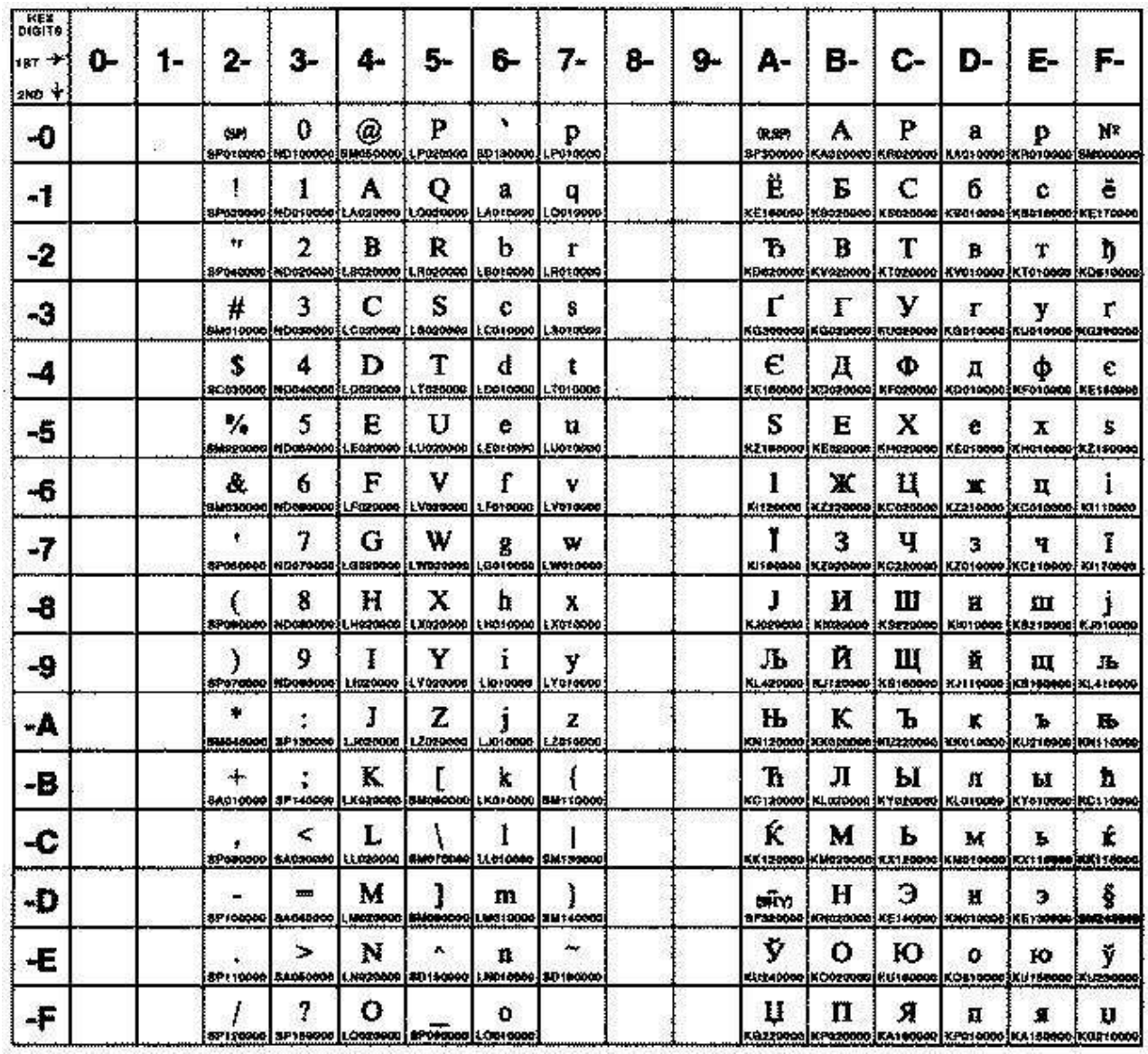
รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด IBM–1046 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “IBM–1046” ในหน้า 255



IBM-1124

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM-1124 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด IBM-1124 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ให้ดู “IBM-1124” ในหน้า 259

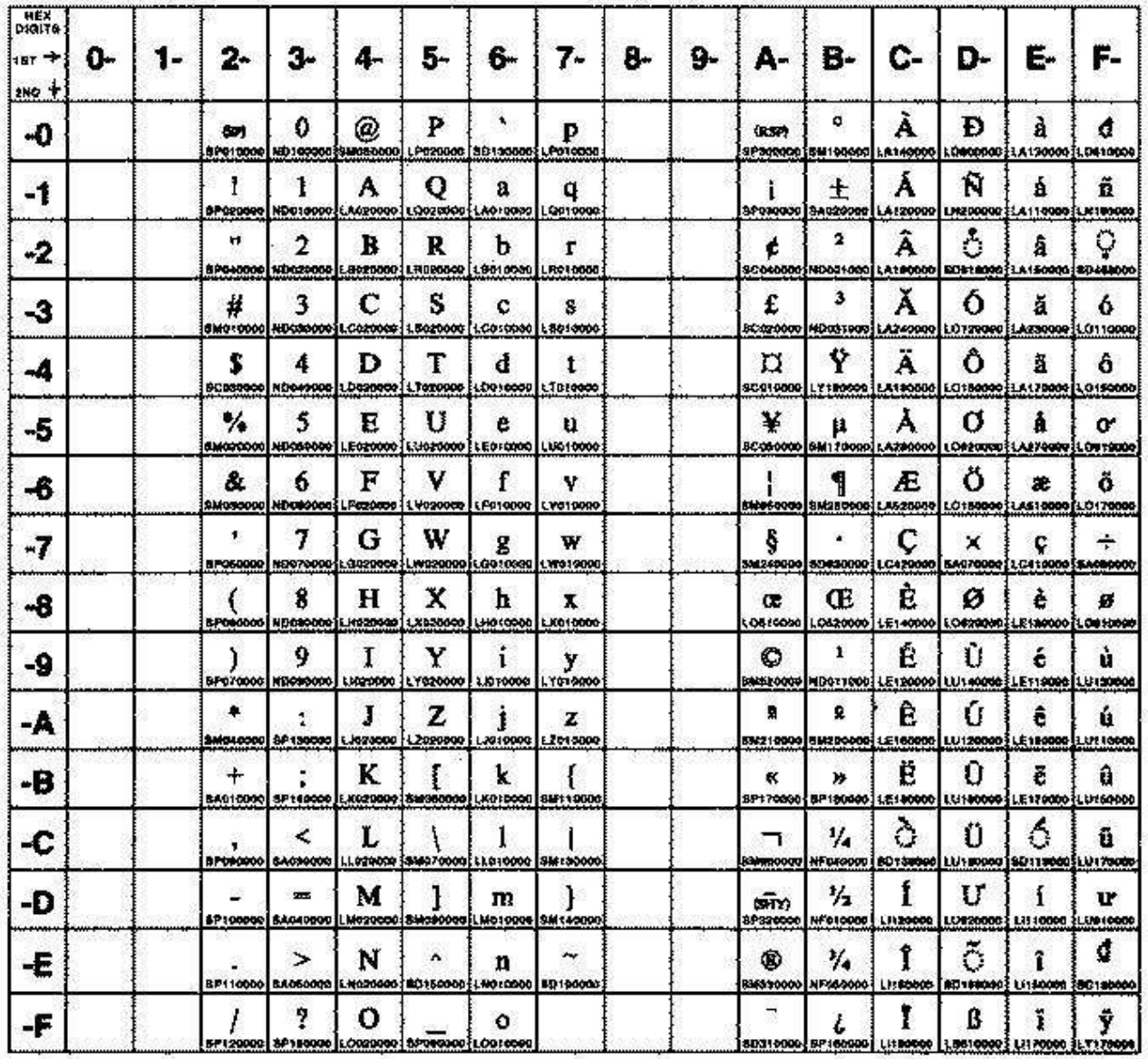


HEX DIGIT0 1ST → 2ND ↓	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			(SP)	0	@	P	`	p			(RSP)	А	Р	а	р	№
-1			!	1	A	Q	a	q			Ё	Б	С	б	с	ё
-2			"	2	B	R	b	r			Ђ	В	Т	в	т	ђ
-3			#	3	C	S	c	s			Ѓ	Г	У	г	у	ѓ
-4			$	4	D	T	d	t			Є	Д	Ф	д	ф	є
-5			%	5	E	U	e	u			S	Е	Х	е	х	s
-6			&	6	F	V	f	v			І	Ж	Ц	ж	ц	і
-7			'	7	G	W	g	w			Ї	З	Ч	з	ч	ї
-8			(	8	H	X	h	x			Ј	И	Ш	и	ш	ј
-9			)	9	I	Y	i	y			Љ	Й	Щ	й	щ	љ
-A			*	:	J	Z	j	z			Њ	К	Ъ	к	ъ	њ
-B			+	;	K	[	k	{			Ћ	Л	Ы	л	ы	ћ
-C			,	<	L	\	l	|		Ќ	М	Ь	м	ь	ќ
-D			-	=	M	]	m	}			(SHY)	Н	Э	н	э	§
-E			.	>	N	^	n	~			Ў	О	Ю	о	ю	ў
-F			/	?	O	_	o				Џ	П	Я	п	я	џ

IBM-1129

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด IBM-1129 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดง โครงร่างของชุดโค้ด IBM-1129 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “IBM-1129” ในหน้า 262

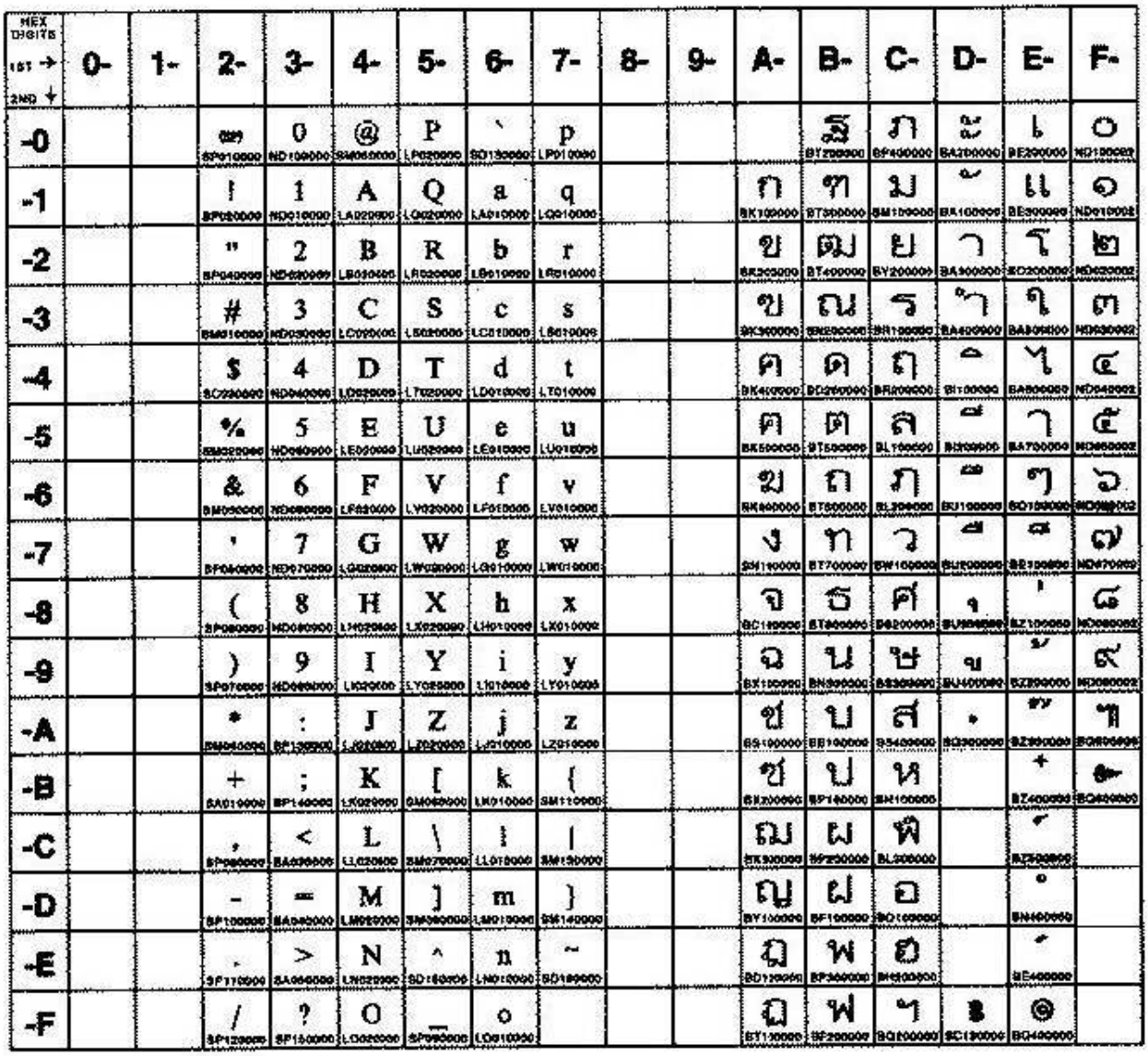


HEX DIGITS 1ST → 2ND ↓	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			(sp)	0	@	P	`	p			(NSP)	°	À	Ð	à	ð
-1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
-2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
-3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
-4			$	4	D	T	d	t			¤	Ý	Ä	Ô	ä	ô
-5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
-6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
-7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
-8			(	8	H	X	h	x			¨	Œ	È	Ø	è	ø
-9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
-A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
-B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
-C			,	<	L	\	l	|		¬	¼	Ì	Ü	ì	ü
-D			-	=	M	]	m	}			(SHY)	½	Í	Ý	í	ý
-E			.	>	N	^	n	~			®	¾	Î	Õ	î	ð
-F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

TIS-620

สัญลักษณ์ที่มีอยู่และโครงร่างของชุดโค้ด TIS-620 มีการรวมไว้ในส่วนนี้

รูปภาพต่อไปนี้สรุปสัญลักษณ์ที่มีอยู่และแสดงโครงร่างของชุดโค้ด TIS-620 สำหรับการแสดงข้อความของ ชุดโค้ดนี้ ให้ดู “TIS-620” ในหน้า 266



UCS-2 และ UTF-8

AIX จะมีชุด ของชุดโค้ดที่กำหนดแอดเดรสของภาษาเฉพาะหรือ กลุ่มภาษา ไม่มีชุดโค้ดที่แสดงในตระกูล ISO8859 ของชุดโค้ด, ชุดโค้ด PC หรือชุดโค้ด Extended UNIX Code (EUC) อนุญาตให้มีการรวม อักขระจากสคริปต์ที่ต่างกัน ด้วย ISO8859-1 คุณจึงสามารถผสมและแสดงอักขระภาษาลาติน 1 (ภาษาที่พูดกันในสหรัฐฯ แคนาดา ยุโรปตะวันตก และลาตินอเมริกาเป็นหลัก) ISO8859-2 ครอบคลุมภาษายุโรปตะวันออก; ISO8859-5 ครอบคลุมภาษา Cyrillic, ISO8859-6 ครอบคลุมภาษาอารบิก, ISO8859-7 ครอบคลุมภาษากรีก, ISO8859-8 ครอบคลุมภาษาฮิบรู, ISO8859-9 ครอบคลุมภาษาเตอร์กิช, IBM-eucJP ครอบคลุม ภาษาญี่ปุ่น, IBM-eucKR ครอบคลุมภาษาเกาหลี, IBM-eucTW ครอบคลุมภาษาจีนดั้งเดิม ประเด็นคือไม่มี ชุดโค้ดใดๆ ด้านบนที่ครอบคลุมทุกภาษา

International Organization for Standardization (ISO) กำหนดวิธีแก้ไข ความครอบคลุมภาษาที่จำกัดของชุดโค้ดโดยการใช้ Unicode เป็น การเข้ารหัสสำหรับรูปแบบ 2-octet ของ ISO10646 Universal Multiple-Octet Coded Character Set (UCS-2) รูปแบบ 32 บิตของ ISO10646 เรียกกันว่า UCS-4 สำหรับรูปแบบ 4-octet AIX ใช้รูปแบบ 16 บิตของ ISO10646 และ ใช้เลเบลมาตรฐาน *UCS-2* เพื่ออธิบายการเข้ารหัสนี้

แม้ว่า UCS-2 เป็นแนวคิดสำหรับโค้ดกระบวนการภายใน แต่ไม่เหมาะสมสำหรับการเข้าโค้ดข้อความธรรมดาบนระบบที่เน้นไบต์แบบดั้งเดิม เช่น AIX ดังนั้น โค้ดไฟล์ภายนอกคือ File System Safe UCS Transformation Format (FSS-UTF) ของ Open Group การเข้ารหัสรูปแบบการแปลงนี้ เรียกอีกอย่างหนึ่งว่า UTF-8, และ *UTF-8* เป็นเลเบลที่ใช้สำหรับการเข้ารหัสนี้บน AIX

หลักการที่เกี่ยวข้อง:

"ตัวแปลง interchange UTF-8" ในหน้า 125
ส่วนนี้จะอธิบายการแปลงที่ใช้ได้ในทั้งสองทิศทางระหว่างชุดโค้ด แต่ละชุดและ UTF-8

ISO10646 UCS-2 (Unicode)

Universal Coded Character Set (UCS) คือชื่อของมาตรฐาน ISO10646 ที่กำหนดโค้ดเพียงโค้ดเดียวสำหรับการแสดงถึง, interchange, การประมวลผล การจัดเก็บ การป้อน และการนำเสนอของรูปแบบ ที่บันทึกไว้ของภาษาที่สำคัญทั้งหมดของโลก

ค่าโค้ดอักขระของ UCS-2 เหมือนกับค่าของมาตรฐานการเข้ารหัส อักขระ Unicode ที่เผยแพร่โดย Unicode Consortium UCS-2 กำหนดโค้ดสำหรับอักขระที่ใช้ในภาษาเขียนที่สำคัญทั้งหมด นอกเหนือจากชุดของสัญลักษณ์ทางวิทยาศาสตร์ คณิตศาสตร์ และการเผยแพร่แล้ว UCS-2 จะครอบคลุมสคริปต์ต่อไปนี้:

- อารบิก
- อาร์เมเนียน
- อาเซอร์ไบจานี
- Bengali
- Bopomofo
- Cyrillic
- Devanagari
- จอร์เจียน
- กรีก
- Gujarati
- Gurmukhi
- Hangul
- จีน Hanzi
- ฮิบรู
- Hiragana
- International Phonetic Alphabet (IPA)
- Katakana
- ญี่ปุ่น Kanji
- Kannada

- เกาหลี Hanja
- ลาว
- ลาติน
- Malayalam
- Maltese
- Oriya
- Tamil
- Telugu
- ไทย
- Tibetan
- Urdu
- Welsh

ความสามารถของ AIX ในการแสดงอักขระในสคริปต์ที่ระบุข้างบนถูกจำกัดด้วยฟอนต์ที่มีอยู่ AIX จัดเตรียมตัวอักษรบิตแม็พสำหรับภาษาหลักของโลก รวมถึงฟอนต์ TrueType ที่สามารถปรับขนาดได้ที่ใช้รูปแบบ Unicode

UCS-2 เข้ารหัสอักขระที่รวมจำนวนมาก ซึ่งรู้จักกันในอีกชื่อหนึ่งว่าเครื่องหมาย ความต่อเนื่องสำหรับ floating diacritics อักขระเหล่านี้เป็นสิ่งที่จำเป็นในสคริปต์หลายรายการ รวมถึง Indic, ไทย อารบิก และฮิบรู อักขระที่รวม ใช้สำหรับการสร้างอักขระในสคริปต์ลาติน, Cyrillic, และกรีก อย่างไรก็ตาม การใช้อักขระที่รวมอาจทำให้เกิดโอกาสการเข้ารหัสอื่น สำหรับข้อความเดียวกันได้ แม้ว่าการเข้ารหัสเป็นเรื่อง ที่คลุมเครือและมีการรักษาบูรณภาพข้อมูล แต่การประมวลผลข้อความ ซึ่งมีอักขระที่รวมเป็นเรื่องที่ซับซ้อนมากกว่า เพื่อให้สอดคล้องกับ แอ็พพลิเคชันที่เลือกจะไม่จัดการกับอักขระที่รวม ISO10646 กำหนดระดับการดำเนินการดังต่อไปนี้:

ระดับ 1
ไม่อนุญาตให้ใช้อักขระที่รวม

ระดับ 2
อนุญาตให้ใช้เครื่องหมายที่รวมจากสคริปต์ภาษาไทย, Indic, ฮิบรู และอารบิก

ระดับ 3
อนุญาตให้ใช้เครื่องหมายที่รวม รวมถึงเครื่องหมายสำหรับภาษาลาติน, Cyrillic, และกรีก

หมายเหตุ: ในระบบปฏิบัติการ AIX เลเบล ISO10646-1 อ้างอิงถึงการเข้ารหัส UCS-2 ป้ายชื่อนี้สามารถใช้เป็นชื่อย่อสำหรับ UCS-2 ได้

UCS-4 และ UTF-32

มาตรฐาน Unicode ใช้เพื่อกำหนดอักขระที่เข้ารหัสมาตรฐาน สำหรับภาษาส่วนใหญ่ที่ใช้กันโดยทั่วไปในโลก รูปแบบ 2-ไบต์ของมาตรฐานนี้เรียกกันโดยทั่วไปว่า *UCS-2* อย่างไรก็ตาม UCS-2 สามารถแสดงข้อมูลได้สูงสุด 65,536 อักขระเท่านั้นในรูปแบบ 2-ไบต์ รูปแบบ 4-ไบต์ของ Unicode เรียกกันว่า *UCS-4* หรือ *UTF-32* และสามารถใช้ในการกำหนดส่วนขยายทั้งหมดของ Unicode ซึ่งกำหนดอักขระเฉพาะได้สูงสุดถึงมากกว่า 1,000,000 อักขระ

UTF-8 (รูปแบบการแปลง UCS)

Open Group ได้พัฒนารูปแบบการแปลงสำหรับ UCS ที่ออกแบบมาสำหรับการใช้งานในระบบไฟล์ที่มีอยู่ วัตถุประสงค์คือ UCS จะเป็นโค้ดกระบวนการสำหรับรูปแบบการแปลง ซึ่งสามารถใช้ เป็นโค้ดไฟล์ได้

UTF-8 มีคุณสมบัติดังต่อไปนี้:

- เป็น superset ของ ASCII ซึ่งอักขระ ASCII มีการเข้ารหัสเป็น อักขระไบต์เดียวที่มีค่าตัวเลขเหมือนกัน
- ไม่มีค่าโค้ด ASCII ในอักขระแบบมัลติไบต์ นอกเหนือจาก ค่าที่แสดงถึงอักขระ ASCII
- ไบต์แรกของอักขระบ่งชี้จำนวนไบต์ที่จะตามมา ในลำดับอักขระแบบมัลติไบต์ และไม่สามารถเกิดขึ้นในที่อื่นใด ในลำดับได้

UTF-8 เข้ารหัสค่า UCS ในช่วง 0 ถึง 0x7FFFFFFF โดยใช้ อักขระแบบมัลติไบต์ที่มีความยาว 1, 2, 3, 4, 5, และ 6 ไบต์ อักขระไบต์เดียวถูกสงวนไว้สำหรับอักขระ ASCII ในช่วง 0 ถึง 0x7f อักขระทั้งหมดเหล่านี้มีการตั้งค่าบิตระดับสูง เป็น 0 สำหรับอักขระที่เข้ารหัสทั้งหมดซึ่งมากกว่าหนึ่งไบต์ ไบต์แรก จะกำหนดจำนวนไบต์ที่ใช้ และมีการตั้งค่าบิตระดับสูงใน แต่ละไบต์ ทุกไบต์ที่ไม่ได้ขึ้นต้นด้วยชุดบิต 10xxxxxx โดยที่ x แสดงถึงบิตที่อาจเป็น 0 หรือ 1 คือจุดเริ่มต้น ของลำดับอักขระ UCS ตารางต่อไปนี้แสดงโค้ดแบบมัลติไบต์ UTF-8:

ไบต์	บิต	Hex ต่ำสุด	Hex สูงสุด	ลำดับไบต์ในไบนารี
1	7	00000000	0000007F	0xxxxxxx
2	11	00000080	000007FF	110xxxxx 10xxxxxx
3	16	00000800	0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
4	21	00010000	001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
5	26	00200000	03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxx 10xxxxxx
6	31	04000000	7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

ค่า UCS เป็นเพียงการต่อกันของบิต x ใน การเข้ารหัสแบบมัลติไบต์ เมื่อมีการเข้ารหัสค่าหนึ่งได้หลายวิธี (ตัวอย่างเช่น UCS 0) จะใช้ได้เฉพาะการเข้ารหัสที่สั้นที่สุดเท่านั้น

ชุดย่อยของ UTF-8 ที่ใช้เพื่อเข้ารหัส UCS-2 มีดังต่อไปนี้:

ไบต์	บิต	Hex ต่ำสุด	Hex สูงสุด	ลำดับไบต์ในไบนารี
1	7	00000000	0000007F	0xxxxxxx
2	11	00000080	000007FF	110xxxxx 10xxxxxx
3	16	00000800	0000FFFF	1110xxxx 10xxxxxx 10xxxxxx

ชุดย่อยนี้ของ UTF-8 ต้องการไบต์สูงสุดสาม (3) ไบต์

UTF-16

UTF-16 คือรูปแบบการแปลง UCS สำหรับ 16 ระนาบ ของกลุ่ม 00 UTF-16 เป็นการเข้ารหัส ISO/IEC ที่เท่าเทียมกับ มาตรฐาน Unicode ซึ่งมีการใช้ surrogates ใน UTF-16 แต่ละค่าโค้ด UCS-2 แสดงถึงตัวค่าเอง ค่าโค้ด Non-BMP ของ ISO/EIC 10646 ในระนาบ 1..16 มีการแสดงถึงโดยใช้คู่ของโค้ดพิเศษ UTF-16 กำหนดการแปลงระหว่างตำแหน่งโค้ด UCS-4 ในระนาบ 1 ถึง 16 ของกลุ่ม 00 และคู่ของโค้ดพิเศษ และเหมือนกับ การแปลงที่กำหนดในมาตรฐาน Unicode

---

ภาพรวมของตัวแปลงสำหรับการเขียนโปรแกรม

การสนับสนุนหลายวัฒนธรรมจะมีพื้นฐานสำหรับการทำให้เป็นโกลบอล ซึ่งข้อมูลมักถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีกโค้ดหนึ่ง มีการสนับสนุนตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

เมื่อโปรแกรมส่งข้อมูลไปยังโปรแกรมอื่นที่อยู่บนรีโมตโฮสต์ ข้อมูลนั้นอาจต้องการการแปลงจากชุดโค้ดของเครื่องต้นทางเป็น ชุดโค้ดของเครื่องรับ ตัวอย่างเช่น เมื่อสื่อสารกับระบบ VM เวิร์คสเตชันจะแปลงข้อมูล ISO8859-1 ของเวิร์คสเตชันเป็นรูปแบบ EBCDIC

ชุดโค้ดเป็นตัวกำหนดอักขระกราฟิกและการกำหนดอักขระควบคุมให้กับจุดโค้ด อักขระที่เข้ารหัสเหล่านี้ยังต้องได้รับการแปลงเมื่อโปรแกรมได้รับ ข้อมูลในชุดโค้ดหนึ่งแต่แสดงข้อมูลนั้นในอีกชุดโค้ดหนึ่ง

ระบบมีอินเตอร์เฟสการแปลงดังต่อไปนี้:

คำสั่ง iconv
: อนุญาตให้คุณร้องขอการแปลงเฉพาะได้โดยการระบุชื่อชุดโค้ด *FromCode* และ *ToCode*

ฟังก์ชัน libiconv
: อนุญาตให้แอ็พพลิเคชันร้องขอตัวแปลงได้โดยการระบุชื่อ

ระบบมีไลบรารีของตัวแปลงที่พร้อมใช้งาน ไลบรารีตัวแปลง มีอยู่ในไดเร็กทอรี **/usr/lib/nls/loc/iconv/*** และ **/usr/lib/nls/loc/iconvTable/*** ห้ามกำหนดตัวแปลงของคุณเองยกเว้นว่ามีความจำเป็นจริงๆ

นอกจากตัวแปลงชุดโค้ดแล้ว ไลบรารีตัวแปลงยังมี ชุดของตัวแปลง interchange บนเครือข่ายอีกด้วย ในสภาพแวดล้อมเครือข่าย ชุดโค้ดของระบบการสื่อสารและโปรโตคอลของการสื่อสาร เป็นตัวกำหนดวิธีการแปลงข้อมูล

ตัวแปลง Interchange ใช้เพื่อแปลงข้อมูลที่ส่งจากระบบหนึ่ง ไปยังระบบอื่น การแปลงจากชุดโค้ดภายในเป็นชุดโค้ดอื่น ต้องใช้ตัวแปลงชุดโค้ด เมื่อข้อมูลต้องได้รับการแปลงจาก ชุดโค้ดของผู้ส่งเป็นชุดโค้ดของผู้รับ หรือแปลงจากข้อมูล 8 บิตเป็นข้อมูล 7 บิต คุณต้องการอินเตอร์เฟสมาตรฐาน รูทีนย่อย **iconv** มี อินเตอร์เฟสนี้

หลักการที่เกี่ยวข้อง:

"การแปลงระหว่างชุดโค้ด" ในหน้า 2
อักขระคือสัญลักษณ์ใดๆ ที่ใช้สำหรับการจัดระเบียบ ควบคุม หรือการแสดงถึงข้อมูล กลุ่มของสัญลักษณ์ดังกล่าวที่ใช้ เพื่ออธิบายภาษาเฉพาะประกอบกันขึ้นเป็นชุดอักขระ ชุดโค้ดมี ค่าการเข้ารหัสสำหรับชุดอักขระ ค่าการเข้ารหัส ในชุดโค้ดทำหน้าที่เป็นอินเตอร์เฟสระหว่างระบบและอุปกรณ์อินพุต และเอาต์พุตของระบบนั้น ตัวแปลงการจัดส่ง การสนับสนุนหลายวัฒนธรรมที่สอดคล้องกับค่าการเข้ารหัสอักขระ ซึ่ง พบในชุดโค้ดที่แตกต่างอื่น

"ภาพรวมของตัวแปลง" ในหน้า 3
การทำให้เป็นโกลบอลมีพื้นฐานสำหรับการทำให้เป็นโกลบอลเพื่อให้ ข้อมูลถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีกโค้ดหนึ่ง คุณอาจ ต้องการแปลงไฟล์ข้อความหรือแค็ตตาล็อกข้อความ มีตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

“การทำความเข้าใจกับ libiconv”
ส่วนนี้จะครอบคลุมการแปลง **iconv** application programming interface (API)

การใช้คำสั่ง iconv

ตัวแปลงใดๆ ที่ติดตั้งในระบบสามารถใช้ได้โดยใช้คำสั่ง iconv ซึ่งจะใช้ไลบรารี iconv

คำสั่ง **iconv** ทำหน้าที่เป็นตัวกรองสำหรับการแปลงจากชุดโค้ดหนึ่งเป็นชุดโค้ดอื่น ตัวอย่างเช่น คำสั่งต่อไปนี้กรองข้อมูลจากโค้ด PC (IBM-850) เป็น ISO8859-1:

```
cat File | iconv -f IBM-850 -t ISO8859-1 | tftp -p - host /tmp/fo
```

คำสั่ง **iconv** แปลงการเข้ารหัสของอักขระที่อ่าน จากอินพุตมาตรฐานหรือไฟล์ที่ระบุ อย่างใดอย่างหนึ่ง จากนั้น บันทึก ผลที่ได้เป็นเอาต์พุตมาตรฐาน

หมายเหตุ: ในระบบปฏิบัติการ AIX เลเบล ISO10646-1 อ้างอิงถึงการเข้ารหัส UCS-2 ป้ายชื่อนี้สามารถใช้เป็นชื่อย่อสำหรับ UCS-2 ได้

การทำความเข้าใจกับ libiconv

ส่วนนี้จะครอบคลุมการแปลง **iconv** application programming interface (API)

ส่วนใหญ่แล้ว **iconv** application programming interface (API) ประกอบด้วยรูทีนย่อยที่ดำเนินการแปลงดังต่อไปนี้:

iconv_open
ทำการเริ่มต้นที่จำเป็นเพื่อแปลงอักขระจากชุดโค้ดที่ระบุโดย พารามิเตอร์ *FromCode* เป็นชุดโค้ดที่ระบุโดย พารามิเตอร์ *ToCode* สตริงที่ระบุ ขึ้นอยู่กับตัวแปลงที่ติดตั้งในระบบ ถ้าการเริ่มต้น สำเร็จ คำอธิบายตัวแปลง, **iconv_t**, จะถูกส่งคืน ไปในคำสั่งเริ่มต้น

iconv เรียกใช้ฟังก์ชันตัวแปลงที่ใช้คำอธิบายซึ่งได้รับ จากรูทีนย่อย **iconv_open** พารามิเตอร์ *inbuf* ชี้ ไปยังอักขระแรกในอินพุตบัฟเฟอร์ และพารามิเตอร์ *inbytesleft* บ่งชี้ จำนวนไบต์ที่ตอนท้ายของบัฟเฟอร์ที่กำลังจะแปลง พารามิเตอร์ *outbuf* ชี้ไปยังไบต์แรกที่มีอยู่ ในเอาต์พุตบัฟเฟอร์ และพารามิเตอร์ *outbytesleft* บ่งชี้ จำนวนไบต์ที่มีอยู่ที่ตอนท้ายของบัฟเฟอร์

สำหรับการเข้ารหัสที่ขึ้นอยู่กับ คำสั่ง รูทีนย่อยจะถูกวางไว้ในคำสั่งเริ่มต้นโดยการเรียก ซึ่งค่า *inbuf* เป็นตัวชี้ null การเรียกในเวลาต่อมา ด้วยพารามิเตอร์ *inbuf* เป็นบางสิ่งที่ไม่ใช่ตัวชี้ null จะทำให้คำสั่งภายในของฟังก์ชันถูกเปลี่ยนแปลงตามความจำเป็น

iconv_close
ปิดคำอธิบายการแปลงที่ระบุโดยตัวแปร *cd* และทำให้ใช้ได้อีกครั้งหนึ่ง

ในสภาพแวดล้อมเครือข่าย ปัจจัยต่อไปนี้ เป็นตัวกำหนดวิธีการแปลงข้อมูล:
- ชุดโค้ดของผู้ส่งและผู้รับ
- โปรโตคอลการสื่อสาร (ข้อมูล 8 บิตหรือ 7 บิต)

ตารางต่อไปนี้แสดงโครงร่างวิธีการแปลงและแนะนำ วิธีการแปลงข้อมูลในสถานการณ์ต่างๆ

ตารางที่ 5. การสื่อสารกับ ระบบที่ใช้ชุดโค้ดเหมือนกัน

เกณฑ์	โปรโตคอลสื่อสาร	โปรโตคอลสื่อสาร
เมธอดที่จะเลือก	7 บิตอย่างเดียว	8 บิต
เสมือน	ไม่ถูกต้อง	ตัวเลือกที่ดีที่สุด
fold7	OK	OK
fold8	ไม่ถูกต้อง	OK
uucode	ตัวเลือกที่ดีที่สุด	OK

ตารางที่ 6. การสื่อสารกับ ระบบที่ใช้ชุดโค้ดต่างกัน (หรือชุดโค้ดของผู้รับไม่เป็นที่รู้จัก)

เกณฑ์	โปรโตคอลสื่อสาร	โปรโตคอลสื่อสาร
เมธอดที่จะเลือก	7 บิตอย่างเดียว	8 บิต
เสมือน	ไม่ถูกต้อง	ไม่ถูกต้องถ้าชุดโค้ดรีโมตไม่เป็นที่รู้จัก
fold7	ตัวเลือกที่ดีที่สุด	OK
fold8	ไม่ถูกต้อง	ตัวเลือกที่ดีที่สุด
uucode	ไม่ถูกต้อง	ไม่ถูกต้อง

ถ้าผู้ส่งใช้ชุดโค้ดเหมือนกันกับผู้รับ สิ่งที่เป็นไปได้ มีดังต่อไปนี้:

- เมื่อโปรโตคอลอนุญาตให้ใช้ข้อมูล 8 บิต ข้อมูลสามารถส่งได้โดยไม่ต้องแปลง
- เมื่อโปรโตคอลอนุญาตให้ใช้ได้เฉพาะข้อมูล 7 บิต จุดโค้ด 8 บิต ต้องถูกแม็พกับค่า 7 บิต ใช้อินเตอร์เฟส **iconv** และวิธีการอย่างใดอย่างหนึ่งต่อไปนี้:

วิธีการ	คำอธิบาย
uucode	นำเสนอการแม็พเหมือนกันกับคำสั่ง **uuencode** และ **uudecode** นี่เป็นวิธีการที่แนะนำ
7 บิต	แปลงชุดโค้ดภายในที่ใช้ข้อมูล 7 บิต วิธีการนี้ จะส่งผ่าน ASCII โดยไม่มีการเปลี่ยนแปลงใดๆ

ถ้าผู้ส่งใช้ชุดโค้ดที่แตกต่างจากผู้รับ อาจเป็นไปได้สองอย่าง ดังนี้:

- เมื่อโปรโตคอลอนุญาตให้ใช้ได้เฉพาะข้อมูล 7 บิต ให้ใช้วิธีการ fold7
- เมื่อโปรโตคอลอนุญาตให้ใช้ข้อมูล 8 บิตได้และคุณทราบ ชุดโค้ดของผู้รับ ให้ใช้อินเตอร์เฟส iconv ในการแปลงข้อมูล ถ้าคุณไม่ทราบชุดโค้ดของผู้รับ ให้ใช้วิธีการต่อไปนี้:

วิธีการ	คำอธิบาย
8 บิต	แปลงชุดโค้ดภายในเป็นรูปแบบ interchange มาตรฐาน มีการส่งผ่านข้อมูล 8 บิต และข้อมูลถูกรักษาไว้ เพื่อให ผู้รับสามารถสร้างข้อมูลขึ้นใหม่ในชุดโค้ดของผู้รับได้

หลักการที่เกี่ยวข้อง:

“ภาพรวมของตัวแปลงสำหรับการเขียนโปรแกรม” ในหน้า 93

การสนับสนุนหลายวัฒนธรรมจะมีพื้นฐานสำหรับการทำให้เป็นโกลบอล ซึ่งข้อมูลมักถูกเปลี่ยนแปลงจากชุดโค้ดหนึ่งเป็นอีก

โค้ดหนึ่ง มีการสนับสนุนตัวแปลงมาตรฐาน อยู่หลายรายการสำหรับวัตถุประสงค์นี้

“ตัวแปลง interchange —7 บิต” ในหน้า 111
ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 7 บิต (fold7)

“ตัวแปลง interchange—8 บิต” ในหน้า 115
ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 8 บิต (fold8)

“ตัวแปลง interchange—uucode” ในหน้า 120
ตัวแปลงนี้นำเสนอการแม็พเหมือนกันกับคำสั่ง **uuencode** และ **uudecode**

การใช้รูทีนย่อย iconv_open

ส่วนนี้จะสาธิตวิธีการใช้รูทีนย่อย **iconv_open** ในสถานการณ์ต่างๆ

ตัวอย่างต่อไปนี้สาธิตวิธีการใช้รูทีนย่อย **iconv_open** ในสถานการณ์ต่างๆ:

- เมื่อผู้ส่งและผู้รับใช้ชุดโค้ดเหมือนกัน และถ้าโปรโตคอลอนุญาตให้ใช้ข้อมูล 8 บิต คุณสามารถส่งข้อมูลได้โดยไม่ต้องแปลง ถ้าโปรโตคอลอนุญาตให้ใช้ได้เฉพาะข้อมูล 7 บิต ให้ทำดังต่อไปนี้:

  ผู้ส่ง:
  ```
  cd = iconv_open("uucode", nl_langinfo(CODESET));
  ```

  ผู้รับ:
  ```
  cd = iconv_open(nl_langinfo(CODESET), "uucode");
  ```
- เมื่อผู้ส่งและผู้รับใช้ชุดโค้ดแตกต่างกัน และถ้าโปรโตคอลอนุญาตให้ใช้ข้อมูล 8 บิต และไม่ทราบชุดโค้ดของผู้รับ ให้ทำดังต่อไปนี้

  ผู้ส่ง:
  ```
  cd = iconv_open("fold8", nl_langinfo(CODESET));
  ```

  ผู้รับ:
  ```
  cd = iconv_open(nl_langinfo(CODESET),"fold8" );
  ```
  ถ้าโปรโตคอลอนุญาตให้ใช้ได้เฉพาะข้อมูล 7 บิต ให้ทำดังต่อไปนี้:

  ผู้ส่ง:
  ```
  cd = iconv_open("fold7", nl_langinfo(CODESET));
  ```

  ผู้รับ:
  ```
  cd = iconv_open(nl_langinfo(CODESET), "fold7" );
  ```

รูทีนย่อย **iconv_open** ใช้ตัวแปรสภาพแวดล้อม **LOCPATH** เพื่อค้นหา ตัวแปลงซึ่งมีชื่ออยู่ในรูปแบบดัง ต่อไปนี้:

`iconv/`*`FromCodeSet_ToCodeSet`*

สตริง *FromCodeSet* แสดงถึงชุดโค้ดของผู้ส่ง และสตริง *ToCodeSet* แสดงถึงชุดโค้ดของผู้รับ อักขระขีดเส้นใต้แบ่งออกเป็นสองสตริง

หมายเหตุ: โปรแกรม **setuid** และ **setgid** ทั้งหมด ละเว้นตัวแปรสภาพแวดล้อม **LOCPATH**

เนื่องจากตัวแปลง **iconv** เป็นโมดูลอ็อบเจ็กต์ที่โหลดได้ จึงต้องใช้อ็อบเจ็กต์ที่แตกต่างเมื่อรันในสภาพแวดล้อม 64 บิต ในสภาพแวดล้อม 64 บิต รูทีน **iconv_open** จะใช้ตัวแปรสภาพแวดล้อม **LOCPATH** เพื่อค้นหาตัวแปลงซึ่งมีชื่ออยู่ในรูปแบบดังต่อไปนี้:

```
iconv/FromCodeSet_ToCodeSet__64
```

ไลบรารี iconv เลือกโดยอัตโนมัติว่าจะโหลดอ็อบเจ็กต์ตัวแปลงมาตรฐาน หรืออ็อบเจ็กต์ตัวแปลง 64 บิต ถ้ารูทีนย่อย **iconv_open** ไม่ได้ค้นหาตัวแปลง รูทีนย่อยจะใช้คู่ **from,to** เพื่อค้นหาไฟล์ ซึ่งกำหนดการแปลงตามข้อมูลตาราง ไฟล์มี ตารางการแปลงที่สร้างขึ้นโดย **คำสั่ง gcenxlt**

ตัวแปลง iconvTable ใช้ตัวแปรสภาพแวดล้อม **LOCPATH** เพื่อค้นหาไฟล์ซึ่งมีชื่ออยู่ในรูปแบบดังต่อไปนี้:

```
iconvTable/FromCodeSet_ToCodeSet
```

ถ้าพบตัวแปลง ระบบจะดำเนินงานโหลดและ เริ่มต้น คำอธิบายตัวแปลง, **iconv_t**, จะถูกส่งคืน ไปยังคำสั่งเริ่มต้น

โปรแกรมตัวแปลงและตาราง

โปรแกรมตัวแปลงคือฟังก์ชันที่ดำเนินการได้ซึ่งแปลง ข้อมูลตามชุดของกฎ ตารางตัวแปลงคือตารางการแปลงไบต์เดียว ที่ดำเนินการแปลงแบบ stateless

โปรแกรมและตารางอยู่ในไดเร็กทอรีที่แยกต่างหากดังนี้:

ไดเร็กทอรี	คำอธิบาย
/usr/lib/nls/loc/iconv	โปรแกรมตัวแปลง
/usr/lib/nls/loc/iconvTable	ตารางตัวแปลง

หลังจากที่โปรแกรมตัวแปลงมีการคอมไพล์และลิงก์กับไลบรารี **libiconv.a** แล้ว โปรแกรมจะถูกวางไว้ในไดเร็กทอรี **/usr/lib/nls/loc/iconv**

เมื่อต้องการสร้างตัวแปลงตาราง ให้สร้างไฟล์ตารางตัวแปลงต้นฉบับ ใช้คำสั่ง **genxlt** เพื่อคอมไพล์ตารางการแปลไปเป็นรูปแบบที่เข้าใจได้โดยตัวแปลง จากนั้น ไฟล์เอาต์พุตจะถูกวางไว้ในไดเร็กทอรี **/usr/lib/nls/loc/iconvTable**

ตัวแปลง unicode และสากล

ส่วนนี้จะครอบคลุมตารางการแปลง unicode และโปรแกรมตัวแปลง สากล

ตารางการแปลง unicode (หรือ UCS-2) อยู่ใน:

```
$LOCPATH/uconvTable/*CodeSet*
```

โปรแกรมตัวแปลง **$LOCPATH/uconv/UCSTBL** ใช้เพื่อทำ การแปลงเป็นและแปลงจาก UCS-2 โดยใช้ยูทิลิตี้ **iconv**

มีโปรแกรมตัวแปลงสากลซึ่งสามารถใช้เพื่อแปลง ระหว่างสองชุดโค้ดที่มีการกำหนดการแปลงเป็นและแปลงจาก UCS-2 กำหนดตาราง uconv ต่อไปนี้:

```
X     -> UCS-2
UCS-2 -> Y
```

สามารถกำหนดการแปลงสากลที่แม็พดังต่อไปนี้:

```
X -> UCS-2 -> Y
```

โดยการใช้ **$LOCPATH/iconv/Universal_UCS_Conv**

ตัวแปลง UCS สากล

UCS-2 คือการเข้ารหัส 16 บิตสากลที่ใช้เป็นสื่อกลางการแลกเปลี่ยน เพื่อให้สามารถทำการแปลงระหว่างชุดโค้ดใดๆ แบบเสมือนได้

การแปลงสามารถทำได้โดยใช้ตัวแปลง UCS สากล ซึ่งจะแปลงระหว่างสองชุดโค้ด XXX และ YYY ดังนี้:

```
XXX <-> UTF-32 <-> YYY
```

การแปลง XXX และ YYY ต้องมีการรวมอยู่ในรายการของตัวแปลง UCS-2 Interchange ที่สนับสนุน และต้องมีการติดตั้งบนระบบด้วย

ตัวแปลงสากลมีการติดตั้งเป็นไฟล์ **/usr/lib/nls/loc/iconv/Universal_UCS_Conv**

การแปลงระหว่างโค้ดอักขระแบบมัลติไบต์และอักขระ wide ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน อย่าแลกเปลี่ยนโค้ดอักขระ wide ระหว่างสองกระบวนการ ยกเว้นคุณมีความรู้ว่า แต่ละโลแคลที่จะใช้จัดการโค้ดอักขระ wide ในลักษณะที่ สอดคล้องกัน โลแคลส่วนใหญ่สำหรับระบบปฏิบัติการนี้ใช้ค่าอักขระ Unicode เป็นโค้ดอักขระ wide ยกเว้นโลแคลที่อิงตามชุดโค้ด IBM-eucTW

การใช้ตัวแปลง

ส่วนนี้จะครอบคลุมรูทีนย่อยอินเตอร์เฟส iconv

อินเตอร์เฟส iconv คือชุดของรูทีนย่อยต่อไปนี้ที่ใช้ เพื่อเปิด ดำเนินการ และปิดการแปลง:

- **iconv_open**
- **iconv**
- **iconv_close**

ตัวอย่างตัวกรองการแปลงชุดโค้ด

ส่วนนี้จะครอบคลุมตัวกรองการแปลงชุดโค้ดที่ ยอมรับพารามิเตอร์ *ToCode* และ *FromCode*

ตัวอย่างต่อไปนี้แสดงให้เห็นว่าคุณสามารถใช้รูทีนย่อยเหล่านี้เพื่อสร้าง ตัวกรองการแปลงชุดโค้ดที่ยอมรับพารามิเตอร์ *ToCode* และ *FromCode* เป็นอินพุตอาร์กิวเมนต์ได้อย่างไร:

```
#include <stdio.h>
#include <nl_types.h>
#include <iconv.h>
#include <string.h>
#include <errno.h>
#include <locale.h>

#define ICONV_DONE() (r>=0)
#define ICONV_INVAL() (r<0) && (errno==EILSEQ))
#define ICONV_OVER() (r<0) && (errno==E2BIG))
#define ICONV_TRUNC() (r<0) && (errno==EINVAL))
```

```
#define USAGE 1
#define ERROR 2
#define INCOMP 3

char ibuf[BUFSIZ], obuf[BUFSIZ];

extern int errno;

main (argc,argv)
int argc;
char **argv;
{
 size_t  ileft,oleft;
 nl_catd catd;
 iconv_t cd;
 int r;
 char *ip,*op;

 setlocale(LC_ALL,"");
 catd = catopen (argv[0],0);

 if(argc!=3){
  fprintf(stderr,
   catgets (catd,NL_SETD,USAGE,"usage;conv fromcode tocode\n"));
  exit(1);
 }

 cd=iconv_open(argv[2],argv[1]);

ileft=0;

while(!feof(stdin)) {
 /*
 * หลังจากการดำเนินงานถัดไป ibuf จะ
 * มีข้อมูลใหม่บวกข้อมูลที่เหลือ
 * จากการอ่านครั้งก่อนหน้านี้ซึ่งถูกตัดให้สั้นลง
 */
 ileft+=fread(ibuf+ileft,1,BUFSIZ-ileft,stdin);
 do {
  ip=ibuf;
  op=obuf;
  oleft=BUFSIZ;

  r=iconv(cd,&ip,&ileft,&op,&oleft);

  if(ICONV_INVAL()){
   fprintf(stderr,
      catgets(catd,NL_SETD,ERROR,"invalid input\n"));
   exit(2);
 }

 fwrite(obuf,1,BUFSIZ-oleft,stdout);

 if(ICONV_TRUNC() || ICONV_OVER())
```

```
  /*
  *ข้อมูลที่ยังเหลืออยู่ในบัฟเฟอร์ ให้คัดลอก
  *ข้อมูลนั้นไปยังตอนต้น
  */

  memcpy(ibuf,ip,ileft);

  /*
  *ลูปจนกว่าอักขระทั้งหมดในอินพุต
  *บัฟเฟอร์ถูกแปลงหมดแล้ว
  */
 } while(ICONV_OVER());
}

 if(ileft!=0){
  /*
  *สิ่งนี้สามารถเกิดขึ้นได้เฉพาะถ้าการเรียกครั้งล่าสุด
  *ไปยัง iconv() ส่งคืน ICONV_TRUNC เท่านั้น ซึ่งหมายความว่า
  *ข้อมูลล่าสุดในอินพุตสตรีม
  *ไม่สมบูรณ์
  */
  fprintf(stderr,catgets(catd,NL_SETD,INCOMP,"input incomplete\n"));
 exit(3);
 }

 iconv_close(cd);
 exit(0);
}
```

ตัวแปลงการตั้งชื่อ

ส่วนนี้ครอบคลุมชื่อชุดโค้ด

ชื่อชุดโค้ดอยู่ในรูปแบบ *CodesetRegistry*-*CodesetEncoding* โดยที่:

ฟอร์มชุดโค้ด	คำอธิบาย
CodesetRegistry	ระบุหน่วยงานการลงทะเบียนสำหรับการเข้ารหัส *CodesetRegistry* ต้อง ประกอบด้วยอักขระจากชุดโค้ดที่ใช้ได้หลายระบบ (โดยทั่วไปคือ A-Z และ 0-9)
CodesetEncoding	ระบุชุดอักขระที่เข้ารหัสซึ่งกำหนดโดยหน่วยงาน ที่ลงทะเบียน

ตัวแปร from,to ที่ใช้โดยคำสั่ง **iconv** และรูทีนย่อย **iconv_open** ระบุไฟล์ที่ชื่อควรอยู่ในรูปแบบ */usr/lib/nls/loc/iconv/%f_%t* หรือ */usr/lib/nls/loc/iconvTable/%f_%t* โดย:

ตัวแปร	คำอธิบาย
%f	แสดงถึงชื่อชุด *FromCode*
%t	แสดงถึงชื่อชุด *ToCode*

รายการของตัวแปลง

ตัวแปลงเปลี่ยนข้อมูลจากชุดโค้ดหนึ่งเป็นชุดโค้ดอื่น ชุด ของตัวแปลงที่ได้รับการสนับสนุนในไลบรารี iconv มีการแสดงรายการอยู่ใน ส่วนต่อไปนี้

ตัวแปลงทั้งหมดที่จัดส่งพร้อมกับสภาพแวดล้อมรันไทม์ BOS มีอยู่ใน ไดเร็กทอรี **/usr/lib/nls/loc/iconv/*** หรือ **/usr/lib/nls/loc/iconvTable/***

ไดเร็กทอรีเหล่านี้ยังมีตัวแปลง *ส่วนตัว* ที่ใช้โดยตัวแปลงอื่นด้วย อย่างไรก็ตาม ผู้ใช้และโปรแกรม ควรจะใช้เฉพาะตัวแปลงในรายการต่อไปนี้เท่านั้น

ตัวแปลงใดๆ ที่จัดส่งพร้อมกับสภาพแวดล้อมรันไทม์ BOS และไม่มีอยู่ในรายการที่นี่ควรจะถือว่าเป็นตัวแปลงส่วนตัวและต้องเปลี่ยนหรือ ลบ ตัวแปลงที่จัดส่งโดยผลิตภัณฑ์อื่นสามารถวางไว้ในไดเร็กทอรี **/usr/lib/nls/loc/iconv/*** หรือ **/usr/lib/nls/loc/iconvTable/***

ควรส่งเสริมให้โปรแกรมเมอร์ใช้ชื่อชุดโค้ดที่ลงทะเบียนหรือ ชื่อชุดโค้ดที่เชื่อมโยงกับแอ็พพลิเคชัน X Consortium เก็บรักษาทะเบียนของชื่อชุดโค้ดไว้สำหรับการอ้างอิง

หลักการที่เกี่ยวข้อง:

"ชุดโค้ดสำหนับการสนับสนุน multicultural" ในหน้า 55
การทำให้เป็นโกลบอลของ AIX อิง ตามสมมติฐานว่าชุดโค้ดทั้งหมดสามาถรแบ่งออกเป็น ชุดอักขระจำนวนหนึ่ง

ตัวแปลงชุดโค้ด PC, ISO, และ EBCDIC

ตัวแปลงเหล่านี้ช่วยในการแปลงระหว่างชุดโค้ด stateless ไบต์เดียว PC, ISO, และ EBCDIC ชนิดของการแปลงที่ได้รับการสนับสนุน มีดังต่อไปนี้: PC ไปเป็น/จาก ISO, PC ไปเป็น/จาก EBCDIC, และ ISO ไปเป็น/จาก EBCDIC

มีการแปลงระหว่างชุดโค้ดที่เข้ากันได้ เช่น ลาติน-1 เป็นลาติน-1 และกรีกเป็นกรีก อย่างไรก็ตาม ไม่สนับสนุนการแปลงระหว่างชุดโค้ด ประจำชาติ EBCDIC ที่แตกต่างกัน สำหรับข้อมูลเกี่ยวกับ การแปลงระหว่างชุดอักขระที่เข้ากันไม่ได้

ตารางการแปลงในไดเร็กทอรี **iconvTable** สร้างขึ้น โดยคำสั่ง **genxlt**

หลักการที่เกี่ยวข้อง:

"ตัวแปลง interchange —7 บิต" ในหน้า 111
ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 7 บิต (fold7)

"ตัวแปลง interchange—8 บิต" ในหน้า 115
ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 8 บิต (fold8)

ข้อมูลที่เกี่ยวข้อง:

คำสั่ง genxlt

ชื่อชุดโค้ดที่เข้ากันได้

ตารางต่อไปนี้แสดงรายการชื่อชุดโค้ดที่เข้ากันได้ แต่ละบรรทัดกำหนดสตริงปลายทาง/ต้นทางที่จะใช้เมื่อร้องขอ ตัวแปลง

หมายเหตุ: ชุดโค้ด PC และ ISO ใช้ ASCII

ตารางที่ 7. ความเข้ากันได้ของชุดโค้ด

ชุดอักขระ	ภาษา	PC	ISO	EBCDIC
ลาติน-1	อังกฤษสำเนียงอเมริกา โปรตุกิส ฝรั่งเศสสำเนียงแคนาดา	N/A	ISO8859-1	IBM-037
ลาติน-1	เดนิช นอร์วีเจียน	N/A	ISO8859-1	IBM-277
ลาติน-1	ฟินนิช สวีดิช	N/A	ISO8859-1	IBM-278
ลาติน-1	อิตาเลียน	N/A	ISO8859-1	IBM-280
ลาติน-1	ญี่ปุ่น	N/A	ISO8859-1	IBM-281
ลาติน-1	สเปนนิช	N/A	ISO8859-1	IBM-284
ลาติน-1	สหราชอาณาจักร อังกฤษ	N/A	ISO8859-1	IBM-285
ลาติน-1	เยอรมัน	N/A	ISO8859-1	IBM-273
ลาติน-1	ฝรั่งเศส	N/A	ISO8859-1	IBM-297
ลาติน-1	เบลเยียน เยอรมันสำเนียงสวิส	N/A	ISO8859-1	IBM-500
ลาติน-2	โครเอเชียน เช็กโกสโลวาเกียน ฮังกาเรียน โปลิช โรมาเนียน ลาตินสำเนียง เซอร์เบียน สโลวัก สโลเวเน	IBM-852	ISO88859-2	IBM-870
Cyrillic	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน	IBM-855	ISO8859-5	IBM-880 IBM-1025
Cyrillic	รัสเซียน	IBM-866	ISO8859-5	IBM-1025
ฮิบรู	ฮิบรู	IBM-856 IBM-862	ISO8859-8	IBM-424 IBM-803
เตอร์กิช	เตอร์กิช	IBM-857	ISO8859-9	IBM-1026
อารบิก	อารบิก	IBM-864 IBM-1046	ISO8859-6	IBM-420
กรีก	กรีก	IBM-869	ISO8859-7	IBM-875
กรีก	กรีก	IBM-869	ISO8859-7	IBM-875
Baltic	ลิทัวเนียน ลัตเวียน เอสโทเนียน	IBM-921 IBM-922	ISO8859-4	IBM-1112 IBM-1122

หมายเหตุ: อักขระที่มีอยู่ในชุดโค้ดต้นฉบับแต่ไม่มีอยู่ใน ชุดโค้ดเป้าหมายจะถูกแปลงเป็นอักขระทดแทนซึ่งตัวแปลงกำหนดขึ้น

ไฟล์

ตารางนี้อธิบายตัวแปลง **inconvTable** ที่พบในไดเร็กทอรี **/usr/lib/nls/loc/iconvTable**

ตารางที่ 8. ตัวแปลง iconvTable

ตารางตัวแปลง	คำอธิบาย	ภาษา
IBM-037_IBM-850	IBM-037 เป็น IBM-850	อังกฤษสำเนียงสหรัฐฯ โปรตุกิส ฝรั่งเศสสำเนียงแคนาดา
IBM-273_IBM-850	IBM-273 เป็น IBM-850	เยอรมัน
IBM-277_IBM-850	IBM-277 เป็น IBM-850	เดนิช นอร์วีเจียน
IBM-278_IBM-850	IBM-278 เป็น IBM-850	ฟินนิช สวีดิช
IBM-280_IBM-850	IBM-280 เป็น IBM-850	อิตาเลียน
IBM-281_IBM-850	IBM-281 เป็น IBM-850	ญี่ปุ่น-ลาติน
IBM-284_IBM-850	IBM-284 เป็น IBM-850	สเปนนิช
IBM-285_IBM-850	IBM-285 เป็น IBM-850	สหราชอาณาจักร อังกฤษ
IBM-297_IBM-850	IBM-297 เป็น IBM-850	ฝรั่งเศส
IBM-420_IBM_1046	IBM-420 เป็น IBM-1046	อารบิก
IBM-424_IBM-856	IBM-424 เป็น IBM-856	ฮิบรู
IBM-424_IBM-862	IBM-424 เป็น IBM-862	ฮิบรู
IBM-500_IBM-850	IBM-500 เป็น IBM-850	เบลเยียน เยอรมันสำเนียงสวิส
IBM-803_IBM-856	IBM-803 เป็น IBM-856	ฮิบรู
IBM-803_IBM-862	IBM-803 เป็น IBM-862	ฮิบรู
IBM-850_IBM-037	IBM-850 เป็น IBM-037	อังกฤษสำเนียงสหรัฐฯ โปรตุกิส ฝรั่งเศสสำเนียงแคนาดา
IBM-850_IBM-273	IBM-850 เป็น IBM-273	เยอรมัน
IBM-850_IBM-277	IBM-850 เป็น IBM-277	เดนิช นอร์วีเจียน
IBM-850_IBM-278	IBM-850 เป็น IBM-278	ฟินนิช สวีดิช
IBM-850_IBM-280	IBM-850 เป็น IBM-280	อิตาเลียน
IBM-850_IBM-281	IBM-850 เป็น IBM-281	ญี่ปุ่น-ลาติน
IBM-850_IBM-284	IBM-850 เป็น IBM-284	สเปนนิช
IBM-850_IBM-285	IBM-850 เป็น IBM-285	สหราชอาณาจักร อังกฤษ
IBM-850_IBM-297	IBM-850 เป็น IBM-297	ฝรั่งเศส
IBM-850_IBM-500	IBM-850 เป็น IBM-500	เบลเยียน เยอรมันสำเนียงสวิส
IBM-856_IBM-424	IBM-856 เป็น IBM-424	ฮิบรู
IBM-856_IBM-803	IBM-856 เป็น IBM-803	ฮิบรู
IBM-856_IBM-862	IBM-856 เป็น IBM-862	ฮิบรู

ตารางที่ 8. ตัวแปลง iconvTable (ต่อ)

ตารางตัวแปลง	คำอธิบาย	ภาษา
IBM-862_IBM-424	IBM-862 เป็น IBM-424	ฮิบรู
IBM-862_IBM-803	IBM-862 เป็น IBM-803	ฮิบรู
IBM-862_IBM-856	IBM-862 เป็น IBM-856	ฮิบรู
IBM-864_IBM-1046	IBM-864 เป็น IBM-1046	อารบิก
IBM-921_IBM-1112	IBM-921 เป็น IBM-1112	ลิทัวเนียน ลัตเวียน
IBM-922_IBM-1122	IBM-922 เป็น IBM-1122	เอสโทเนียน
IBM-1112_IBM-921	IBM-1121 เป็น IBM-921	ลิทัวเนียน ลัตเวียน
IBM-1122_IBM-922	IBM-1122 เป็น IBM-922	เอสโทเนียน
IBM-1046_IBM-420	IBM-1046 เป็น IBM-420	อารบิก
IBM-1046_IBM-864	IBM-1046 เป็น IBM-864	อารบิก
IBM-037_ISO8859-1	IBM-037 เป็น ISO8859-1	อังกฤษสำเนียงอเมริกา โปรตุกิส ฝรั่งเศสสำเนียงแคนาดา
IBM-273_ISO8859-1	IBM-273 เป็น ISO8859-1	เยอรมัน
IBM-277_ISO8859-1	IBM-277 เป็น ISO8859-1	เดนิช นอร์วีเจียน
IBM-278_ISO8859-1	IBM-278 เป็น ISO8859-1	ฟินนิช สวีดิช
IBM-280_ISO8859-1	IBM-280 เป็น ISO8859-1	อิตาเลียน
IBM-281_ISO8859-1	IBM-281 เป็น ISO8859-1	ญี่ปุ่น-ลาติน
IBM-284_ISO8859-1	IBM-284 เป็น ISO8859-1	สเปนนิช
IBM-285_ISO8859-1	IBM-285 เป็น ISO8859-1	สหราชอาณาจักร อังกฤษ
IBM-297_ISO8859-1	IBM-297 เป็น ISO8859-1	ฝรั่งเศส
IBM-420_ISO8859-6	IBM-420 เป็น ISO8859-6	อารบิก
IBM-424_ISO8859-8	IBM-424 เป็น ISO8859-8	ฮิบรู
IBM-500_ISO8859-1	IBM-500 เป็น ISO8859-1	เบลเยียน เยอรมันสำเนียงสวิส
IBM-803_ISO8859-8	IBM-803 เป็น ISO8859-8	ฮิบรู
IBM-852_ISO8859-2	IBM-852 เป็น ISO8859-2	โครเอเชียน เช็กโกสโลวาเกียน ฮังกาเรียน โปลิช โรมาเนียน ลาตินสำเนียงเซอร์เบียน สโลวัก สโลเวเน
IBM-855_ISO8859-5	IBM-855 เป็น ISO8859-5	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
IBM-866_ISO8859-5	IBM-866 เป็น ISO8859-5	รัสเซียน
IBM-869_ISO8859-7	IBM-869 เป็น ISO8859-7	กรีก
IBM-875_ISO8859-7	IBM-875 เป็น ISO8859-7	กรีก

ตารางที่ 8. ตัวแปลง iconvTable (ต่อ)

ตารางตัวแปลง	คำอธิบาย	ภาษา
IBM-870_ISO8859-2	IBM-870 เป็น ISO8859-2	โครเอเชียน เช็กโกสโลวาเกียน ฮังกาเรียน โปลิช โรมาเนียน เซอร์เบียน สโลวัก สโลเวเน
IBM-880_ISO8859-5	IBM-880 เป็น ISO8859-5	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
IBM-1025_ISO8859-5	IBM-1025 เป็น ISO8859-5	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
IBM-857_ISO8859-9	IBM-857 เป็น ISO8859-9	เตอร์กิช
IBM-1026_ISO8859-9	IBM-1026 เป็น ISO8859-9	เตอร์กิช
IBM-850_ISO8859-1	IBM-850 เป็น ISO8859-1	ลาติน
IBM-856_ISO8859-8	IBM-856 เป็น ISO8859-8	ฮิบรู
IBM-862_ISO8859-8	IBM-862 เป็น ISO8859-8	ฮิบรู
IBM-864_ISO8859-6	IBM-864 เป็น ISO8859-6	อารบิก
IBM-1046_ISO8859-6	IBM-1046 เป็น ISO8859-6	อารบิก
ISO8859-1_IBM-850	ISO8859-1 เป็น IBM-850	ลาติน
ISO8859-6_IBM-864	ISO8859-6 เป็น IBM-864	อารบิก
ISO8859-6_IBM-1046	ISO8859-6 เป็น IBM-1046	อารบิก
ISO8859-8_IBM-856	ISO8859-8 เป็น IBM-856	ฮิบรู
ISO8859-8_IBM-862	ISO8859-8 เป็น IBM-862	ฮิบรู
ISO8859-1_IBM-037	ISO8859-1 เป็น IBM-037	อังกฤษสำเนียงอเมริกา โปรตุกิส ฝรั่งเศสสำเนียงแคนาดา
ISO8859-1_IBM-273	ISO8859-1 เป็น IBM-273	เยอรมัน
ISO8859-1_IBM-277	ISO8859-1 เป็น IBM-277	เดนิช นอร์วีเจียน
ISO8859-1_IBM-278	ISO8859-1 เป็น IBM-278	ฟินนิช สวีดิช
ISO8859-1_IBM-280	ISO8859-1 เป็น IBM-280	อิตาเลียน
ISO8859-1_IBM-281	ISO8859-1 เป็น IBM-281	ญี่ปุ่น-ลาติน
ISO8859-1_IBM-284	ISO8859-1 เป็น IBM-284	สเปนนิช
ISO8859-1_IBM-285	ISO8859-1 เป็น IBM-285	สหราชอาณาจักร อังกฤษ
ISO8859-1_IBM-297	ISO8859-1 เป็น IBM-297	ฝรั่งเศส
ISO8859-1_IBM-500	ISO8859-1 เป็น IBM-500	เบลเยียน เยอรมันสำเนียงสวิส
ISO8859-2_IBM-852	ISO8859-2 เป็น IBM-852	โครเอเชียน เช็กโกสโลวาเกียน ฮังกาเรียน โปลิช โรมาเนียน ลาตินสำเนียงเซอร์เบียน สโลวัก สโลเวเน

ตารางที่ 8. ตัวแปลง iconvTable (ต่อ)

ตารางตัวแปลง	คำอธิบาย	ภาษา
ISO8859-2_IBM-870	ISO8859-2 เป็น IBM-870	โครเอเชียน เช็กโกสโลวาเกียน ฮังกาเรียน โปลิช โรมาเนียน ลาตินสำเนียงเซอร์เบียน สโลวัก สโลเวเน
ISO8859-5_IBM-855	ISO8859-5 เป็น IBM-855	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
ISO8859-5_IBM-880	ISO8859-5 เป็น IBM-880	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
ISO8859-5_IBM-1025	ISO8859-5 เป็น IBM-1025	บัลกาเรียน มาเซโดเนียน, Cyrillic สำเนียงเซอร์เบียน รัสเซียน
ISO8859-6_IBM-420	ISO8859-6 เป็น IBM-420	อารบิก
ISO8859-5_IBM-866	ISO8859-5 เป็น IBM-866	รัสเซียน
ISO8859-7_IBM-869	ISO8859-7 เป็น IBM-869	กรีก
ISO8859-7_IBM-875	ISO8859-7 เป็น IBM-875	กรีก
ISO8859-8_IBM-424	ISO8859-8 เป็น IBM-424	ฮิบรู
ISO8859-8_IBM-803	ISO8859-8 เป็น IBM-803	ฮิบรู
ISO8859-9_IBM-857	ISO8859-9 เป็น IBM-857	เตอร์กิช
ISO8859-9_IBM-1026	ISO8859-9 เป็น IBM-1026	เตอร์กิช

ตัวแปลงชุดโค้ดหลายไบต์

ส่วนนี้อธิบายชุดโค้ดที่ใช้ตัวแปลงชุดโค้ดหลายไบต์ในการแปลง

ตัวแปลงชุดโค้ดหลายไบต์จะแปลงอักขระระหว่างชุดโค้ด ดังต่อไปนี้:

- ชุดโค้ดหลายไบต์ PC
- ชุดโค้ดหลายไบต์ EUC (บน ISO)
- ชุดโค้ดหลายไบต์ EBCDIC

ตารางต่อไปนี้แสดงรายการชื่อชุดโค้ดที่เข้ากันได้ แต่ละ บรรทัดกำหนดสตริงปลายทาง/ต้นทางที่จะใช้เมื่อร้องขอตัวแปลง

ตารางที่ 9. ความเข้ากันได้ของชุดโค้ด

ภาษา	PC	ISO	EBCDIC
ภาษาญี่ปุ่น	IBM-932	IBM-eucJP	IBM-930, IBM-939
ภาษาญี่ปุ่น (MS เข้ากันได้)	IBM-943	IBM-eucJP	IBM-930, IBM-939
ภาษาเกาหลี	IBM-934	IBM-eucKR	IBM-933
ภาษาจีนดั้งเดิม	IBM-938, big-5	IBM-eucTW	IBM-937
ภาษาจีนแบบง่าย	IBM-1381	IBM-eucCN	IBM-935

1. มีการแปลงระหว่างภาษาจีนแบบง่ายและภาษาจีนดั้งเดิม (IBM-eucTW <—> IBM-eucCN and big5 <—> IBM-eucCN)
2. UTF-8 เป็นชุดโค้ดเพิ่มเติม

หลักการที่เกี่ยวข้อง:

“ตัวแปลง interchange UTF-8” ในหน้า 125
ส่วนนี้จะอธิบายการแปลงที่ใช้ได้ในทั้งสองทิศทางระหว่างชุดโค้ด แต่ละชุดและ UTF-8

ไฟล์

รายการต่อไปนี้อธิบายตัวแปลง Multibyte Code Set ที่พบในไดเร็กทอรี **/usr/lib/nls/loc/iconv**

ตัวแปลง	คำอธิบาย
IBM-eucJP_IBM-932	IBM-eucJP เป็น IBM-932
IBM-eucJP_IBM-943	IBM-eucJP เป็น IBM-943
IBM-eucJP_IBM-930	IBM-eucJP เป็น IBM-930
IBM-eucCN_IBM-936(PC5550)	IBM-eucCN เป็น IBM-936(PC5550)
IBM-eucCN_IBM-935	IBM-eucCN เป็น IBM-935
IBM-eucJP_IBM-939	IBM-eucJP เป็น IBM-939
IBM-eucCN_IBM-1381	IBM-eucCN เป็น IBM-1381
IBM-943_IBM-932	IBM-943 เป็น IBM-932
IBM-932_IBM-943	IBM-932 เป็น IBM-943
IBM-930_IBM-932	IBM-930 เป็น IBM-932
IBM-930_IBM-943	IBM-930 เป็น IBM-943
IBM-930_IBM-eucJP	IBM-930 เป็น IBM-eucJP
IBM-932_IBM-eucJP	IBM-932 เป็น IBM-eucJP
IBM-932_IBM-930	IBM-932 เป็น IBM-930
IBM-943_IBM-eucJP	IBM-943 เป็น IBM-eucJP
IBM-943_IBM-930	IBM-943 เป็น IBM-930
IBM-936(PC5550)_IBM-935	IBM-936(PC5550) เป็น IBM-935
IBM-936_IBM-935	IBM-936 เป็น IBM-935
IBM-932_IBM-939	IBM-932 เป็น IBM-939
IBM-939_IBM-932	IBM-939 เป็น IBM-932
IBM-943_IBM-939	IBM-943 เป็น IBM-939
IBM-939_IBM-943	IBM-939 เป็น IBM-943
IBM-935_IBM-936(PC5550)	IBM-935 เป็น IBM-936(PC5550)
IBM-935_IBM-936	IBM-935 เป็น IBM-936

ตัวแปลง	คำอธิบาย
IBM-1381_IBM-935	IBM-1381 เป็น IBM-935
IBM-935_IBM-1381	IBM-935 เป็น IBM-1381
IBM-935_IBM-eucCN	IBM-935 เป็น IBM-eucCN
IBM-936(PC5550)_IBM-eucCN	IBM-936(PC5550) เป็น IBM-eucCN
IBM-eucTW_IBM-eucCN	IBM-eucTW เป็น IBM-eucCN
big5_IBM-eucCN	big5 เป็น IBM-eucCN
IBM-1381_IBM-eucCN	IBM-1381 เป็น IBM-eucCN
IBM-939_IBM-eucJP	IBM-939 เป็น IBM-eucJP
IBM-eucKR_IBM-934	IBM-eucKR เป็น IBM-934
IBM-934_IBM-eucKR	IBM-934 เป็น IBM-eucKR
IBM-eucKR_IBM-933	IBM-eucKR เป็น IBM-933
IBM-933_IBM-eucKR	IBM-933 เป็น IBM-eucKR
IBM-eucTW_IBM-937	IBM-eucTW เป็น IBM-937
IBM-938_IBM-937	IBM-938 เป็น IBM-937
big-5_IBM-937	big-5 เป็น IBM-937
IBM-eucCN_IBM-eucTW	IBM-eucCN เป็น IBM-eucTW
IBM-937_IBM-eucTW	IBM-937 เป็น IBM-eucTW
IBM-937_IBM-938	IBM-937 เป็น IBM-938
IBM-eucTW_IBM-938	IBM_eucTW เป็น IBM_938
IBM-eucCN_big5	IBM-eucCN เป็น big5
IBM-eucTW_big-5	IBM_eucTW เป็น big-5
IBM-937_big-5	IBM-937 เป็น big-5
CNS11643.1992-3_IBM-eucTW	CNS11643.1992-3 เป็น IBM_eucTW
CNS11643.1992-3-GL_IBM-eucTW	CNS11643.1992-3-GL เป็น IBM_eucTW
CNS11643.1992-3-GR_IBM-eucTW	CNS11643.1992-3-GR เป็น IBM_eucTW
CNS11643.1992-4_IBM-eucTW	CNS11643.1992-4 เป็น IBM_eucTW
CNS11643.1992-4-GL_IBM-eucTW	CNS11643.1992-4-GL เป็น IBM_eucTW
CNS11643.1992-4-GR_IBM-eucTW	CNS11643.1992-4-GR เป็น IBM_eucTW
IBM-eucTW_CNS11643.1992-3	IBM_eucTW เป็น CNS11643.1992-3
IBM-eucTW_CNS11643.1992-3-GL	IBM_eucTW เป็น CNS11643.1992-3-GL
IBM-eucTW_CNS11643.1992-3-GR	IBM_eucTW เป็น CNS11643.1992-3-GR
IBM-eucTW_CNS11643.1992-4	IBM_eucTW เป็น CNS11643.1992-4

ตัวแปลง	คำอธิบาย
IBM-eucTW_CNS11643.1992-4-GL	IBM_eucTW เป็น CNS11643.1992-4-GL
IBM-eucTW_CNS11643.1992-4-GR	IBM_eucTW เป็น CNS11643.1992-4-GR
IBM-eucCN_GB2312.1980-1	IBM-eucCN เป็น GB2312.1980-1
IBM-eucCN_GB2312.1980-1-GL	IBM-eucCN เป็น GB2312.1980-1-GL
IBM-eucCN_GB2312.1980-1-GR	IBM-eucCN เป็น GB2312.1980-1-GR
IBM-937_csic	IBM-937 เป็น csic
csic_IBM-937	csic เป็น IBM-937
IBM-938_csic	IBM-938 เป็น csic
csic_IBM-938	csic เป็น IBM-938
IBM-eucTW_ccdc	IBM-eucTW เป็น ccdc
ccdc_IBM-eucTW	ccdc เป็น IBM-eucTW
IBM-eucTW_cns	IBM-eucTW เป็น cns
cns_IBM-eucTW	cnd เป็น IBM-eucTW
IBM-eucTW_csic	IBM-eucTW เป็น csic
csic_IBM-eucTW	csic เป็น IBM-eucTW
IBM-eucTW_sops	IBM-ecuTW เป็น sops
sops_IBM-eucTW	sops เป็น IBM-eucTW
IBM-eucTW_tca	IBM-eucTW เป็น tca
tca_IBM-eucTW	tca เป็น IBM-eucTW
big5_cns	big5 เป็น cns
cns_big5	cns เป็น big5
big5_csic	big5 เป็น csic
csic_big5	csic เป็น big5
big5_ttc	big5 เป็น ttc
ttc_big5	ttc เป็น big5
big5_ttcmin	big5 เป็น ttcmin
ttcmin_big5	ttcmin เป็น big5
big5_unicode	big5 เป็น unicode
unicode_big5	unicode เป็น big5
big5_wang	big5 เป็น wang
wang_big5	wang เป็น big5
ccdc_csic	ccdc เป็น csic

ตัวแปลง	คำอธิบาย
csic_ccdc	csic เป็น_ccdc
csic_sops	csic เป็น sops
sops_csic	sops เป็น csic
CNS11643.1986-1_big5	CNS11643.1986-1 เป็น big5
big5_CNS11643.1986-1	big5 เป็น CNS11643.1986-1
CNS11643.1986-1-GR_big5	CNS11643.1986-1-GR เป็น big5
big5_CNS11643.1986-1-GR	big5 เป็น CNS11643.1986-1-GR
CNS11643.1986-2_big5	CNS11643.1986-2 เป็น big5
big5_CNS11643.1986-2	big5 เป็น CNS11643.1986-2
CNS11643.1986-2-GR_big5	CNS11643.1986-2-GR เป็น big5
big5_CNS11643.1986-2-GR	big5 เป็น CNS11643.1986-2-GR
CNS11643.CT-GR_big5	CNS11643.CT-GR เป็น big5
big5_CNS11643.CT-GR	big5 เป็น CNS11643.CT-GR
IBM-sbdTW-GR_big5	IBM-sbdTW-GR เป็น big5
big5_IBM-sbdTW-GR	big5 เป็น IBM-sbdTW-GR
IBM-sbdTW.CT-GR_big5	IBM-sbdTW.CT-GR เป็น big5
big5_IBM-sbdTW.CT-GR	big5 เป็น IBM-sbdTW.CT-GR
IBM-sbdTW_big5	IBM-sbdTW เป็น big5
big5_IBM-sbdTW	big5 เป็น IBM-sbdTW
IBM-udcTW-GR_big5	IBM-udcTW-GR เป็น big5
big5_IBM-udcTW-GR	big5 เป็น IBM-udcTW-GR
IBM-udcTW.CT-GR_big5	IBM-udcTW.CT-GR เป็น big5
big5_IBM-udcTW.CT-GR	big5 เป็น IBM-udcTW.CT-GR
ISO8859-1_big5	ISO8859 เป็น big5
big5_ISO8859-1	big5 เป็น ISO8859
IBM-sbdTW_big5	IBM-sbdTW เป็น big5
big5_IBM-sbdTW	big5 เป็น IBM-sbdTW
big5_ASCII-GR	big5 เป็น ASCII-GR
ASCII-GR_big5	ASCII-GR เป็น big5
GBK_big5	GBK เป็น big5
big5_GBK	big5 เป็น GBK
GBK_IBM-eucTW	GBK เป็น IBM-eucTW

ตัวแปลง	คำอธิบาย
IBM-eucTW_GBK	IBM-eucTW เป็น GBK
CNS11643.1986-1_GBK	CNS11643.1986-1 เป็น GBK
GBK_CNS11643.1986-1	GBK เป็น CNS11643.1986-1
CNS11643.1986-2_GBK	CNS11643.1986-2 เป็น GBK
GBK_CNS11643.1986-2	GBK เป็น CNS11643.1986-2
CNS11643.1986-1-GR_GBK	CNS11643.1986-1-GR เป็น GBK
GBK_CNS11643.1986-1-GR	GBK เป็น CNS11643.1986-1-GR
CNS11643.1986-2-GR_GBK	CNS11643.1986-2-GR เป็น GBK
GBK_CNS11643.1986-2-GR	GBK เป็น CNS11643.1986-2-GR
CNS11643.1986-1-GL_GBK	CNS11643.1986-1-GL เป็น GBK
GBK_CNS11643.1986-1-GL	GBK เป็น CNS11643.1986-1-GL
CNS11643.1986-2-GL_GBK	CNS11643.1986-2-GL เป็น GBK
GBK_CNS11643.1986-2-GL	GBK เป็น CNS11643.1986-2-GL
CNS11643.CT-GR_GBK	CNS11643.CT-GR เป็น GBK
GBK_CNS11643.CT-GR	GBK เป็น CNS11643.CT-GR
GB2312.1980.CT-GR_GBK	GB2312.1980.CT-GR เป็น GBK
GBK_GB2312.1980.CT-GR	GBK เป็น GB2312.1980.CT-GR
GB2312.1980-0_GBK	GBK2312.1980-0 เป็น GBK
GBK_GB2312.1980-0	GBK เป็น GBK2312.1980-0
GB2312.1980-0-GR_GBK	GB2312.1980-0-GR เป็น GBK
GBK_GB2312.1980-0-GR	GBK เป็น GB2312.1980-0-GR
GB2312.1980-0-GL_GBK	GB2312.1980-0-GL เป็น GBK
GBK_GB2312.1980-0-GL	GBK เป็น GB2312.1980-0-GL
ASCII-GR_GBK	ASCII-GR เป็น GBK
GBK_ASCII-GR	GBK เป็น ASCII-GR
ISO8859-1_GBK	ISO8859-1 เป็น GBK
GBK_ISO8859-1	GBK เป็น ISO8859-1
IBM-eucCN_GBK	IBM-eucCN เป็น GBK
GBK_IBM-eucCN	GBK เป็น IBM-eucCN

ตัวแปลง interchange — 7 บิต

ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 7 บิต (fold7)

ชื่อ fold7 ระบุการเข้ารหัสที่สามารถใช้เพื่อส่งผ่านข้อมูลที่เป็นข้อความ ผ่านทางโปรโตคอลเมล 7 บิต การเข้ารหัสใช้ข้อมูล ISO2022

ตัวแปลง fold7 จะแปลงอักขระจากชุดโค้ดเป็นการเข้ารหัส 7 บิตที่บัญญัติไว้ซึ่งระบุอักขระแต่ละตัว การแปลงชนิดนี้ มีประโยชน์ในเครือข่ายที่ไคลเอ็นต์สื่อสารกับชุดโค้ดที่แตกต่างกัน แต่ใช้ชุดอักขระเดียวกัน ตัวอย่างเช่น:

การแปลง ชุดโค้ด	คำอธิบาย
IBM-850 <—> ISO8859-1	อักขระ ภาษาลาตินทั่วไป
IBM-932 <—> IBM-eucJP	อักขระ ภาษาญี่ปุ่นทั่วไป

ลำดับ escape ที่กำหนดชุดโค้ดมาตรฐานมีดังต่อไปนี้:

Escape sequence	ชุดโค้ดมาตรฐาน
01/11 02/04 04/00	GL JIS X0208.1978-0.
01/11 02/04 02/08 04/01	GL ครึ่งซ้ายของ GB2312.1980-0
01/11 02/08 04/02	GL 7 บิต ASCII หรือครึ่งซ้ายของ ISO8859-1
01/11 02/14 04/01	GL ครึ่งขวาของ ISO8859-1
01/11 02/14 04/02	GL ครึ่งขวาของ ISO8859-2
01/11 02/14 04/03	GL ครึ่งขวาของ ISO8859-3
01/11 02/14 04/04	GL ครึ่งขวาของ ISO8859-4
01/11 02/14 04/06	GL ครึ่งขวาของ ISO8859-7
01/11 02/14 04/07	GL ครึ่งขวาของ ISO8859-6
01/11 02/14 04/08	GL ครึ่งขวาของ ISO8859-8
01/11 02/14 04/12	GL ครึ่งขวาของ ISO8859-5
01/11 02/14 04/13	GL ครึ่งขวาของ ISO8859-9
01/11 02/08 04/09	GL ครึ่งขวาของ JIS X0201.1976-0
01/11 02/08 04/10	GL ครึ่งซ้ายของ JIS X0201.1976
01/11 02/04 04/02	GL JIS X0208.1983-0
01/11 02/04 02/08 04/02	GL JIS X0208.1983-0
01/11 02/04 02/08 04/00	GL JISX0208.1978-0
01/11 02/05 02/15 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02	GL ครึ่งขวาของอักขระเฉพาะ IBM-850 อักขระทั่วไป ของ ISO8859-1 ไม่ได้ใช้ลำดับ escape นี้
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GL ภาษาญี่ปุ่น) IBM-udcJP) อักขระที่ผู้ใช้กำหนดได้
01/11 02/04 02/08 04/03	GL KSC5601-1987
01/11 02/04 02/09 03/00	GL CNS11643-1986-1

Escape sequence	ชุดโค้ดมาตรฐาน
01/11 02/04 02/10 03/01	GL CNS11643-1986-2
01/11 02/05 02/15 03/00 ML 05/05 05/04 04/06 02/13 03/07 00/02	UCS-2 ที่เข้ารหัสเป็น base64; ใช้เฉพาะสำหรับอักขระที่ไม่ได้เข้ารหัสโดยลำดับ escape 7 บิตอื่นใดที่แสดงรายการข้างบน

เมื่อแปลงจากชุดโค้ดเป็น fold7 จะมีการเลือกลำดับ escape ที่ใช้ในการกำหนด ชุดโค้ดตามลำดับที่แสดงรายการอยู่ ตัวอย่างเช่น อักขระ JISX0208.1983-0 ใช้ **01/11 01/04 04/02** เป็น การกำหนด

หลักการที่เกี่ยวข้อง:

"การทำความเข้าใจกับ libiconv" ในหน้า 94
ส่วนนี้จะครอบคลุมการแปลง **iconv** application programming interface (API)

"ตัวแปลงชุดโค้ด PC, ISO, และ EBCDIC" ในหน้า 101
ตัวแปลงเหล่านี้ช่วยในการแปลงระหว่างชุดโค้ด stateless ไบต์เดียว PC, ISO, และ EBCDIC ชนิดของการแปลงที่ได้รับการสนับสนุน มีดังต่อไปนี้: PC ไปเป็น/จาก ISO, PC ไปเป็น/จาก EBCDIC, และ ISO ไปเป็น/จาก EBCDIC

ไฟล์

รายการนี้จะอธิบายตัวแปลง fold7 ที่พบ ในไดเร็กทอรี **/usr/lib/nls/loc/iconv**

ตัวแปลง	คำอธิบาย
fold7_IBM-850	รูปแบบ interchange เป็น IBM-850
fold7_IBM-921	รูปแบบ interchange เป็น IBM-921
fold7_IBM-922	รูปแบบ interchange เป็น IBM-922
fold7_IBM-932	รูปแบบ interchange เป็น IBM-932
fold7_IBM-943	รูปแบบ interchange เป็น IBM-943
fold7_IBM_1124	รูปแบบ interchange เป็น IBM-1124
fold7_IBM_1129	รูปแบบ interchange เป็น IBM-1129
fold7_IBM_eucCN	รูปแบบ interchange เป็น IBM-eucCN
fold7_IBM-eucJP	รูปแบบ interchange เป็น IBM-eucJP
fold7_IBM-eucKR	รูปแบบ interchange เป็น IBM-eucKR
fold7_IBM-eucTW	รูปแบบ interchange เป็น IBM-eucTW
fold7_ISO8859-1	รูปแบบ interchange เป็น ISO8859-1
fold7_ISO8859-2	รูปแบบ interchange เป็น ISO8859-2
fold7_ISO8859-3	รูปแบบ interchange เป็น ISO8859-3
fold7_ISO8859-4	รูปแบบ interchange เป็น ISO8859-4
fold7_ISO8859-5	รูปแบบ interchange เป็น ISO8859-5
fold7_ISO8859-6	รูปแบบ interchange เป็น ISO8859-6
fold7_ISO8859-7	รูปแบบ interchange เป็น ISO8859-7

ตัวแปลง	คำอธิบาย
fold7_ISO8859-8	รูปแบบ interchange เป็น ISO8859-8
fold7_ISO8859-9	รูปแบบ interchange เป็น ISO8859-9
fold7_TIS-620	รูปแบบ interchange เป็น TIS-620
fold7_UTF-8	รูปแบบ interchange เป็น UTF-8
fold7_big5	รูปแบบ interchange เป็น big5
fold7_GBK	รูปแบบ interchange เป็น GBK
IBM-921_fold7	IBM-921 เป็นรูปแบบ interchange
IBM-922_fold7	IBM-922 เป็นรูปแบบ interchange
IBM-850_fold7	IBM-850 เป็นรูปแบบ interchange
IBM-932_fold7	IBM-932 เป็นรูปแบบ interchange
IBM-943_fold7	IBM-943 เป็นรูปแบบ interchange
IBM-1124_fold7	IBM-1124 เป็นรูปแบบ interchange
IBM-1129_fold7	IBM-1129 เป็นรูปแบบ interchange
IBM-eucCN_fold7	IBM-eucCN เป็นรูปแบบ interchange
IBM-eucJP_fold7	IBM-eucJP เป็นรูปแบบ interchange
IBM-eucKR_fold7	IBM-eucKR เป็นรูปแบบ interchange
IBM-eucTW_fold7	IBM-eucTW เป็นรูปแบบ interchange
ISO8859-1_fold7	ISO8859-1 เป็นรูปแบบ interchange
ISO8859-2_fold7	ISO8859-2 เป็นรูปแบบ interchange
ISO8859-3_fold7	ISO8859-3 เป็นรูปแบบ interchange
ISO8859-4_fold7	ISO8859-4 เป็นรูปแบบ interchange
ISO8859-5_fold7	ISO8859-5 เป็นรูปแบบ interchange
ISO8859-6_fold7	ISO8859-6 เป็นรูปแบบ interchange
ISO8859-7_fold7	ISO8859-7 เป็นรูปแบบ interchange
ISO8859-8_fold7	ISO8859-8 เป็นรูปแบบ interchange
ISO8859-9_fold7	ISO8859-9 เป็นรูปแบบ interchange
TIS-620_fold7	TIS-620 เป็นรูปแบบ interchange
UTF-8_fold7	UTF-8 เป็นรูปแบบ interchange
big5_fold7	big5 เป็นรูปแบบ interchange
GBK_fold7	GBK เป็นรูปแบบ interchange

ตัวแปลง interchange—8 บิต

ตัวแปลงนี้ช่วยในการแปลงระหว่างโค้ดภายใน และรูปแบบ interchange มาตรฐาน 8 บิต (fold8)

ชื่อ fold8 ระบุการเข้ารหัสที่สามารถใช้เพื่อส่งผ่านข้อมูลที่เป็นข้อความ ผ่านทางโปรโตคอลเมล์ 8 บิต การเข้ารหัสใช้ข้อมูล ISO2022

ตัวแปลง fold8 จะแปลงอักขระจากการเข้ารหัสชุดโค้ดเฉพาะเป็นการเข้ารหัส 8 บิตที่บัญญัติไว้ซึ่งระบุอักขระแต่ละตัว การแปลงชนิดนี้มีประโยชน์ในเครือข่ายที่ไคลเอ็นต์ สื่อสารกับชุดโค้ดที่แตกต่างกัน แต่ใช้ชุดอักขระเดียวกัน ตัวอย่างเช่น:

การแปลงชุดโค้ด	คำอธิบาย
IBM-850 <—> ISO8859-1	อักขระ ภาษาลาตินทั่วไป
IBM-932 <—> IBM-eucJP	อักขระ ภาษาญี่ปุ่นทั่วไป

ลำดับ escape กำหนดชุดโค้ดมาตรฐานดังต่อไปนี้

ลำดับ Escape	ชุดโค้ดมาตรฐาน
01/11 02/04 02/09 04/01	GR ครึ่งขวาของ GB2312.1980-0
01/11 02/13 04/01	GR ครึ่งขวาของ ISO8859-1
01/11 02/13 04/02	GR ครึ่งขวาของ ISO8859-2
01/11 02/13 04/03	GR ครึ่งขวาของ ISO8859-3
01/11 02/13 04/04	GR ครึ่งขวาของ ISO8859-4
01/11 02/13 04/06	GR ครึ่งขวาของ ISO8859-7
01/11 02/13 04/07	GR ครึ่งขวาของ ISO8859-6
01/11 02/13 04/08	GR ครึ่งขวาของ ISO8859-8
01/11 02/13 04/13	GR ครึ่งขวาของ ISO8859-5
01/11 02/13 04/13	GR ครึ่งขวาของ ISO8859-9
01/11 02/09 04/09	GR ครึ่งขวาของ JIS X0201.1976-1
01/11 02/04 02/09 04/02	GR JIS X0208.1983-1
01/11 02/04 02/09 04/00	GR JISX0208.1978-1
01/11 02/09 04/02	GR 7 บิต ASCII หรือครึ่งซ้ายของ ISO8859-1
01/11 02/05 02/15 03/01 ML 04/09 04/02 04/13 02/13 03/08 03/05 03/00 00/02	GR ครึ่งขวาของอักขระเฉพาะ IBM-850 อักขระทั่วไป ของ ISO8859-1 ไม่ควรใช้ลำดับ escape นี้
01/11 02/05 02/15 03/02 ML 04/09 04/02 04/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GR ครึ่งขวาของอักขระภาษาญี่ปุ่นที่ผู้ใช้กำหนดได้
01/11 02/08 04/02	GL 7 บิต ASCII หรือครึ่งซ้ายของ ISO8859-1
01/11 02/14 04/01	GL ครึ่งขวาของ ISO8859-1
01/11 02/14 04/02	GL ครึ่งขวาของ ISO8859-2

ลำดับ Escape	ชุดโค้ดมาตรฐาน
01/11 02/14 04/03	GL ครึ่งขวาของ ISO8859-3
01/11 02/14 04/04	GL ครึ่งขวาของ ISO8859-4
01/11 02/14 04/06	GL ครึ่งขวาของ ISO8859-7
01/11 02/14 04/07	GL ครึ่งขวาของ ISO8859-6
01/11 02/14 04/08	GL ครึ่งขวาของ ISO8859-8
01/11 02/14 04/12	GL ครึ่งขวาของ ISO8859-5
01/11 02/14 04/13	GL ครึ่งขวาของ ISO8859-9
01/11 02/08 04/09	GL ครึ่งขวาของ JIS X0201.1976-0
01/11 02/08 04/10	GL ครึ่งซ้ายของ JIS X0201.1976
01/11 02/04 02/08 04/02	GL JIS X0208.1983-0
01/11 02/04 04/02	GL JIS X0208.1983-0
01/11 02/04 04/00	GL JIS X0208.1978-0
01/11 02/05 02/15 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02	GL ครึ่งขวาของอักขระเฉพาะ IBM-850 อักขระทั่วไปของ ISO8859-1 ไม่ได้ใช้ลำดับ escape นี้
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02	GL ภาษาญี่ปุ่น (IBM-udcJP) อักขระที่ผู้ใช้กำหนดได้
01/11 02/04 02/09 04/03	GR KSC5601-1987
01/11 02/04 02/09 03/00	GR CNS11643-1986-1
01/11 02/04 02/10 03/01	GR CNS11643-1986-2
01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/05 06/04 06/03 05/05 05/08 00/02	GR ครึ่งขวาของอักขระภาษาจีนดั้งเดิมที่ผู้ใช้กำหนดได้
01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/03 06/02 06/04 05/05 05/08 00/02	GR ครึ่งขวาของสัญลักษณ์เฉพาะ IBM-850
01/11 02/04 02/08 04/03	GL KSC5601-1987
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 05/05 05/08 00/02	GL ภาษาจีนดั้งเดิม (IBM-udcTW) อักขระที่ผู้ใช้กำหนดได้
01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/03 06/02 06/04 05/05 05/08 00/02	GL ภาษาจีนดั้งเดิม IBM-850 สัญลักษณ์เฉพาะ (IBM-shdTW) อักขระที่ผู้ใช้กำหนดได้
01/11 02/05 02/15 03/00 M L 05/05 05/04 04/06 02/13 03/08 00/02	UCS-2 ที่เข้ารหัสเป็น UTF-8; ใช้เฉพาะสำหรับอักขระที่ไม่ได้เข้ารหัสโดยลำดับ escape ใดๆ ที่แสดงรายการข้างบน

เมื่อแปลงจากชุดโค้ดเป็น fold8 จะมีการเลือกลำดับ escape ที่ใช้ในการกำหนด ชุดโค้ดตามลำดับที่แสดงรายการอยู่ ตัวอย่างเช่น อักขระ JISX0208.1983-0 ใช้ **01/11 02/04 02/08 04/02** เป็นการกำหนด

หลักการที่เกี่ยวข้อง:

“การทำความเข้าใจกับ libiconv” ในหน้า 94

ส่วนนี้จะครอบคลุมการแปลง **iconv** application programming interface (API)

“ตัวแปลงชุดโค้ด PC, ISO, และ EBCDIC” ในหน้า 101
ตัวแปลงเหล่านี้ช่วยในการแปลงระหว่างชุดโค้ด stateless ไบต์เดียว PC, ISO, และ EBCDIC ชนิดของการแปลงที่ได้รับการสนับสนุน มีดังต่อไปนี้: PC ไปเป็น/จาก ISO, PC ไปเป็น/จาก EBCDIC, และ ISO ไปเป็น/จาก EBCDIC

ไฟล์

รายการนี้อธิบายตัวแปลง fold8 ที่พบในไดเร็กทอรี **/usr/lib/nls/loc/iconv**

ตัวแปลง	คำอธิบาย
fold8_IBM-850	รูปแบบ interchange เป็น IBM-850
fold8_IBM-921	รูปแบบ interchange เป็น IBM-921
fold8_IBM-922	รูปแบบ interchange เป็น IBM-922
fold8_IBM-932	รูปแบบ interchange เป็น IBM-932
fold8_IBM-943	รูปแบบ interchange เป็น IBM-943
fold8_IBM-1124	รูปแบบ interchange เป็น IBM-1124
fold8_IBM-1129	รูปแบบ interchange เป็น IBM-1129
fold8_IBM-eucCN	รูปแบบ interchange เป็น IBM-eucCN
fold8_IBM-eucJP	รูปแบบ interchange เป็น IBM-eucJP
fold8_IBM-eucKR	รูปแบบ interchange เป็น IBM-eucKR
fold8_IBM-eucTW	รูปแบบ interchange เป็น IBM-eucTW
fold8_IBM-eucCN	รูปแบบ interchange เป็น IBM-eucCN
fold8_ISO8859-1	รูปแบบ interchange เป็น ISO8859-1
fold8_ISO8859-2	รูปแบบ interchange เป็น ISO8859-2
fold8_ISO8859-3	รูปแบบ interchange เป็น ISO8859-3
fold8_ISO8859-4	รูปแบบ interchange เป็น ISO8859-4
fold8_ISO8859-5	รูปแบบ interchange เป็น ISO8859-5
fold8_ISO8859-6	รูปแบบ interchange เป็น ISO8859-6
fold8_ISO8859-7	รูปแบบ interchange เป็น ISO8859-7
fold8_ISO8859-8	รูปแบบ interchange เป็น ISO8859-8
fold8_ISO8859-9	รูปแบบ interchange เป็น ISO8859-9
fold8_TIS-620	รูปแบบ interchange เป็น TIS-620
fold8_UTF-8	รูปแบบ interchange เป็น UTF-8
fold8_big5	รูปแบบ interchange เป็น big5
fold8_GBK	รูปแบบ interchange เป็น GBK
IBM-921_fold8	IBM-921 เป็นรูปแบบ interchange
IBM-922_fold8	IBM-922 เป็นรูปแบบ interchange

ตัวแปลง	คำอธิบาย
IBM-850_fold8	IBM-850 เป็นรูปแบบ interchange
IBM-932_fold8	IBM-932 เป็นรูปแบบ interchange
IBM-943_fold8	IBM-943 เป็นรูปแบบ interchange
IBM-1124_fold8	IBM-1124 เป็นรูปแบบ interchange
IBM-1129_fold8	IBM-1129 เป็นรูปแบบ interchange
IBM-eucCN_fold8	IBM-eucCN เป็นรูปแบบ interchange
IBM-eucJP_fold8	IBM-eucJP เป็นรูปแบบ interchange
IBM-eucKR_fold8	IBM-eucKR เป็นรูปแบบ interchange
IBM-eucTW_fold8	IBM-eucTW เป็นรูปแบบ interchange
IBM-eucCN_fold8	IBM-eucCN เป็นรูปแบบ interchange
ISO8859-1_fold8	ISO8859-1 เป็นรูปแบบ interchange
ISO8859-2_fold8	ISO8859-2 เป็นรูปแบบ interchange
ISO8859-3_fold8	ISO8859-3 เป็นรูปแบบ interchange
ISO8859-4_fold8	ISO8859-4 เป็นรูปแบบ interchange
ISO8859-5_fold8	ISO8859-5 เป็นรูปแบบ interchange
ISO8859-6_fold8	ISO8859-6 เป็นรูปแบบ interchange
ISO8859-7_fold8	ISO8859-7 เป็นรูปแบบ interchange
ISO8859-8_fold8	ISO8859-8 เป็นรูปแบบ interchange
ISO8859-9_fold8	ISO8859-9 เป็นรูปแบบ interchange
TIS-620_fold8	TIS-620 เป็นรูปแบบ interchange
UTF-8_fold8	UTF-8 เป็นรูปแบบ interchange
big5_fold8	big5 เป็นรูปแบบ interchange
GBK_fold8	GBK เป็นรูปแบบ interchange

ตัวแปลง interchange—ข้อความผสม

ตัวแปลง interchange ข้อความผสมจะแปลงระหว่างข้อความ ผสมและชุดโค้ดภายใน

ข้อความผสมคือการเข้ารหัส interchange ที่กำหนดโดย X Consortium และใช้เพื่อสื่อสารข้อความระหว่างไคลเอ็นต์ X ข้อความผสมสร้างขึ้น ตามข้อมูล ISO2022 และสามารถเข้ารหัสชุดอักขระส่วนใหญ่โดยใช้ลำดับ escape มาตรฐาน และยังนำเสนอส่วนขยายสำหรับการเข้ารหัสชุดอักขระ ส่วนตัวด้วย ชุดโค้ดที่สนับสนุนมีตัวแปลงไปและ แปลงกลับจากข้อความผสม ชื่อที่ใช้ในการระบุการเข้ารหัสข้อความผสมคือ ct

ลำดับ escape ต่อไปนี้ใช้เพื่อกำหนดชุดโค้ดมาตรฐาน ในลำดับที่แสดงรายการข้างล่าง

01/11 02/05 02/15 03/01 M L 04/09 04/02 04/13 02/13 03/08 03/05 03/00 00/02
GR ครึ่งขวาของอักขระเฉพาะ IBM-850 อักขระทั่วไป ของ ISO8859-1 ไม่ควรใช้ลำดับ escape นี้

01/11 02/05 02/15 03/02 M L 04/09 04/02 04/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02
GR ครึ่งขวาของอักขระภาษาญี่ปุ่นที่ผู้ใช้กำหนดได้

01/11 02/05 02/15 03/01 M L 06/09 06/02 06/13 02/13 03/08 03/05 03/00 00/02
GL ครึ่งขวาของอักขระเฉพาะ IBM-850 อักขระทั่วไป ของ ISO8859-1 ไม่ได้ใช้ลำดับ escape นี้

01/11 02/05 02/15 03/02 M L 06/09 06/02 06/13 02/13 07/05 06/04 06/03 04/10 05/00 00/02
GL ภาษาญี่ปุ่น (IBM-udcJP) อักขระที่ผู้ใช้กำหนดได้

ไฟล์

รายการนี้อธิบายตัวแปลงข้อความผสมที่พบใน ไดเร็กทอรี **/usr/lib/nls/loc/iconv**

ตัวแปลง	คำอธิบาย
ct_IBM-850	รูปแบบ interchange เป็น IBM-850
ct_IBM-921	รูปแบบ interchange เป็น IBM-921
ct_IBM-922	รูปแบบ interchange เป็น IBM-922
ct_IBM-932	รูปแบบ interchange เป็น IBM-932
ct_IBM-943	รูปแบบ interchange เป็น IBM-943
ct_IBM-1124	รูปแบบ interchange เป็น IBM-1124
ct_IBM-1129	รูปแบบ interchange เป็น IBM-1129
ct_IBM-eucCN	รูปแบบ interchange เป็น IBM-eucCN
ct_IBM-eucJP	รูปแบบ interchange เป็น IBM-eucJP
ct_IBM-eucKR	รูปแบบ interchange เป็น IBM-eucKR
ct_IBM-eucTW	รูปแบบ interchange เป็น IBM-eucTW
ct_ISO8859-1	รูปแบบ interchange เป็น ISO8859-1
ct_ISO8859-2	รูปแบบ interchange เป็น ISO8859-2
ct_ISO8859-3	รูปแบบ interchange เป็น ISO8859-3
ct_ISO8859-4	รูปแบบ interchange เป็น ISO8859-4
ct_ISO8859-5	รูปแบบ interchange เป็น ISO8859-5
ct_ISO8859-6	รูปแบบ interchange เป็น ISO8859-6
ct_ISO8859-7	รูปแบบ interchange เป็น ISO8859-7
ct_ISO8859-8	รูปแบบ interchange เป็น ISO8859-8
ct_ISO8859-9	รูปแบบ interchange เป็น ISO8859-9
ct_TIS-620	รูปแบบ interchange เป็น TIS-620
ct_big5	รูปแบบ interchange เป็น big5

ตัวแปลง	คำอธิบาย
ct_GBK	รูปแบบ interchange เป็น GBK
IBM-850_ct	IBM-850 เป็นรูปแบบ interchange
IBM-921_ct	IBM-921 เป็นรูปแบบ interchange
IBM-922_ct	IBM-922 เป็นรูปแบบ interchange
IBM-932_ct	IBM-932 เป็นรูปแบบ interchange
IBM-943_ct	IBM-943 เป็นรูปแบบ interchange
IBM-1124_ct	IBM-1124 เป็นรูปแบบ interchange
IBM-1129_ct	IBM-1129 เป็นรูปแบบ interchange
IBM-eucCN_ct	IBM-eucCN เป็นรูปแบบ interchange
IBM-eucJP_ct	IBM-eucJP เป็นรูปแบบ interchange
IBM-eucKR_ct	IBM-eucKR เป็นรูปแบบ interchange
IBM-eucTW_ct	IBM-eucTW เป็นรูปแบบ interchange
ISO8859-1_ct	ISO8859-1 เป็นรูปแบบ interchange
ISO8859-2_ct	ISO8859-2 เป็นรูปแบบ interchange
ISO8859-3_ct	ISO8859-3 เป็นรูปแบบ interchange
ISO8859-4_ct	ISO8859-4 เป็นรูปแบบ interchange
ISO8859-5_ct	ISO8859-5 เป็นรูปแบบ interchange
ISO8859-6_ct	ISO8859-6 เป็นรูปแบบ interchange
ISO8859-7_ct	ISO8859-7 เป็นรูปแบบ interchange
ISO8859-8_ct	ISO8859-8 เป็นรูปแบบ interchange
ISO8859-9_ct	ISO8859-9 เป็นรูปแบบ interchange
TIS-620_ct	TIS-620 เป็นรูปแบบ interchange
big5_ct	big5 เป็นรูปแบบ interchange
GBK_ct	GBK เป็นรูปแบบ interchange

ตัวแปลง interchange—uucode

ตัวแปลงนี้นำเสนอการแม็พเหมือนกันกับคำสั่ง **uuencode** และ **uudecode**

ในระหว่างการแปลงจาก uucode จะมีการแปลงครั้งละ 62 ไบต์ (รวมถึง เร็กคอร์ดท้ายอักขระบรรทัดใหม่) และการสร้าง 45 ไบต์ใน *outbuf*

หลักการที่เกี่ยวข้อง:

"การทำความเข้าใจกับ libiconv" ในหน้า 94

ส่วนนี้จะครอบคลุมการแปลง **iconv** application programming interface (API)

ไฟล์

รายการนี้อธิบายตัวแปลง uucode ที่พบในไดเร็กทอรี **/usr/lib/nls/loc/iconv**

ตัวแปลง	คำอธิบาย
IBM-850_uucode	IBM-850 เป็น uucode
IBM-921_uucode	IBM-921 เป็น uucode
IBM-922_uucode	IBM-922 เป็น uucode
IBM-932_uucode	IBM-932 เป็น uucode
IBM-943_uucode	IBM-943 เป็น uucode
IBM-1124_uucode	IBM-1124 เป็น uucode
IBM-1129_uucode	IBM-1129 เป็น uucode
IBM-eucJP_uucode	IBM-eucJP เป็น uucode
IBM-eucKR_uucode	IBM-eucKR เป็น uucode
IBM-eucTW_uucode	IBM-eucTW เป็น uucode
IBM-eucCN_uucode	IBM-eucCN เป็น uucode
ISO8859-1_uucode	ISO8859-1 เป็น uucode
ISO8859-2_uucode	ISO8859-2 เป็น uucode
ISO8859-3_uucode	ISO8859-3 เป็น uucode
ISO8859-4_uucode	ISO8859-4 เป็น uucode
ISO8859-5_uucode	ISO8859-5 เป็น uucode
ISO8859-6_uucode	ISO8859-6 เป็น uucode
ISO8859-7_uucode	ISO8859-7 เป็น uucode
ISO8859-8_uucode	ISO8859-8 เป็น uucode
ISO8859-9_uucode	ISO8859-9 เป็น uucode
TIS-620_uucode	TIS-620 เป็น uucode
big5_uucode	big5 เป็น uucode
GBK_uucode	GBK เป็น uucode
uucode_IBM-850	uucode เป็น IBM-850
uucode_IBM-921	uucode เป็น IBM-921
uucode_IBM-922	uucode เป็น IBM-922
uucode_IBM-932	uucode เป็น IBM-932
uucode_IBM-943	uucode เป็น IBM-943
uucode_IBM-1124	uucode เป็น IBM-1124
uucode_IBM-1129	uucode เป็น IBM-1129

ตัวแปลง	คำอธิบาย
uucode_IBM-eucCN	uucode เป็น IBM-eucCN
uucode_IBM-eucJP	uucode เป็น IBM-eucJP
uucode_IBM-eucKR	uucode เป็น IBM-eucKR
uucode_IBM-eucTW	uucode เป็น IBM-eucTW
uucode_ISO8859-1	uucode เป็น ISO8859-1
uucode_ISO8859-2	uucode เป็น ISO8859-2
uucode_ISO8859-3	uucode เป็น ISO8859-3
uucode_ISO8859-4	uucode เป็น ISO8859-4
uucode_ISO8859-5	uucode เป็น ISO8859-5
uucode_ISO8859-6	uucode เป็น ISO8859-6
uucode_ISO8859-7	uucode เป็น ISO8859-7
uucode_ISO8859-8	uucode เป็น ISO8859-8
uucode_ISO8859-9	uucode เป็น ISO8859-9
uucode_TIS-1124	uucode เป็น TIS-1129
uucode_big5	uucode เป็น big5
uucode_GBK	uucode เป็น GBK

ตัวแปลง interchange UCS-2

UCS-2 ใช้การเข้ารหัส 16 บิตสากล การแปลงสำหรับ ชุดโค้ดแต่ละชุดสามารถทำได้ในทั้งสองทิศทาง ระหว่างชุดโค้ด และ UCS-2

ตัวแปลง UCS-2 สามารถพบได้ในไดเร็กทอรี **/usr/lib/nls/loc/uconvTable** และ **/usr/lib/nls/loc/uconv** คำสั่ง **uconvdef** ใช้เพื่อสร้างตัวแปลงใหม่หรือเพื่อ กำหนดตัวแปลง UCS-2 ที่มีอยู่เอง

ตัวแปลง	คำอธิบาย
ISO8859-1	UCS-2 <—> ISO ลาติน-1
ISO8859-2	UCS-2 <—> ISO ลาติน-2
ISO8859-3	UCS-2 <—> ISO ลาติน-3
ISO8859-4	UCS-2 <—> ISO บัลติก
ISO8859-5	UCS-2 <—> ISO Cyrillic
ISO8859-6	UCS-2 <—> ISO อารบิก
ISO8859-7	UCS-2 <—> ISO กรีก
ISO8859-8	UCS-2 <—> ISO ฮิบรู

ตัวแปลง	คำอธิบาย
ISO8859-9	UCS-2 <—> ISO เตอร์กิช
JISX0201.1976-0	UCS-2 <—> ญี่ปุ่น JISX0201-0
JISX0208.1983-0	UCS-2 <—> ญี่ปุ่น JISX0208-0
CNS11643.1986-1	UCS-2 <—> จีน CNS11643-1
CNS11643.1986-2	UCS-2 <—> จีน CNS11643-2
KSC5601.1987-0	UCS-2 <—> เกาหลี KSC5601-0
IBM-eucCN	UCS-2 <—> จีนแบบง่าย EUC
IBM-udcCN	UCS-2 <—> อักขระจีนแบบง่าย ที่ผู้ใช้กำหนด
IBM-sbdCN	UCS-2 <—> อักขระจีนแบบง่ายที่ IBM ระบุ
GB2312.1980-0	UCS-2 <—> จีนแบบง่าย GB
IBM-1381	UCS-2 <—> โค้ดข้อมูล PC จีนแบบง่าย
IBM-935	UCS-2 <—> จีนแบบง่าย EBCDIC
IBM-936	UCS-2 <—> จีนแบบง่าย PC5550
IBM-eucJP	UCS-2 <—> ญี่ปุ่น EUC
IBM-eucKR	UCS-2 <—> เกาหลี EUC
IBM-eucTW	UCS-2 <—> จีนดั้งเดิม EUC
IBM-udcJP	UCS-2 <—> อักขระญี่ปุ่น ที่ผู้ใช้กำหนด
IBM-udcTW	UCS-2 <—> อักขระจีนดั้งเดิม ที่ผู้ใช้กำหนด
IBM-sbdTW	UCS-2 <—> อักขระจีนดั้งเดิมที่ IBM ระบุ
UTF-8	UCS-2 <—> UTF-8
IBM-437	UCS-2 <—> โค้ดข้อมูล PC สหรัฐฯ
IBM-850	UCS-2 <—> โค้ดข้อมูล PC ลาติน 1
IBM-852	UCS-2 <—> โค้ดข้อมูล PC ลาติน 2
IBM-857	UCS-2 <—> โค้ดข้อมูล PC เตอร์กิช
IBM-860	UCS-2 <—> โค้ดข้อมูล PC โปรตุกิส
IBM-861	UCS-2 <—> โค้ดข้อมูล PC ไอซ์แลนดิก
IBM-863	UCS-2 <—> โค้ดข้อมูล PC ฝรั่งเศสสำเนียงแคนาดา
IBM-865	UCS-2 <—> โค้ดข้อมูล PC นอร์ดิก
IBM-869	UCS-2 <—> โค้ดข้อมูล PC กรีก
IBM-921	UCS-2 <—> โค้ดข้อมูล บัลติกหลายภาษา
IBM-922	UCS-2 <—> โค้ดข้อมูลเอสโทเนียน
IBM-932	UCS-2 <—> โค้ดข้อมูล PC ญี่ปุ่น

ตัวแปลง	คำอธิบาย
IBM-943	UCS-2 <—> โค้ดข้อมูล PC ญี่ปุ่น
IBM-934	UCS-2 <—> โค้ดข้อมูล PC เกาหลี
IBM-936	UCS-2 <—> โค้ดข้อมูล PC สาธารณรัฐประชาชนจีน
IBM-938	UCS-2 <—> โค้ดข้อมูล PC ไต้หวัน
IBM-942	UCS-2 <—> โค้ดข้อมูล PC ญี่ปุ่นแบบขยาย
IBM-944	UCS-2 <—> โค้ดข้อมูล PC เกาหลี
IBM-946	UCS-2 <—> โค้ดข้อมูล SAA สาธารณรัฐประชาชนจีน
IBM-948	UCS-2 <—> โค้ดข้อมูล PC จีนดั้งเดิม
IBM-1124	UCS-2 <—> โค้ดข้อมูล PC ยูเครเนียน
IBM-1129	UCS-2 <—> โค้ดข้อมูล PC เวียตนาม
TIS-620	UCS-2 <—> โค้ดข้อมูล PC ประเทศไทย
IBM-037	UCS-2 <—> สหรัฐ, แคนาดา EBCDIC
IBM-273	UCS-2 <—> เยอรมนี, ออสเตรีย EBCDIC
IBM-277	UCS-2 <—> เดนมาร์ก, นอร์เวย์ EBCDIC
IBM-278	UCS-2 <—> ฟินแลนด์, สวีเดน EBCDIC
IBM-280	UCS-2 <—> อิตาลี EBCDIC
IBM-284	UCS-2 <—> สเปน, ลาตินอเมริกา EBCDIC
IBM-285	UCS-2 <—> สหราชอาณาจักร EBCDIC
IBM-297	UCS-2 <—> ฝรั่งเศส EBCDIC
IBM-500	UCS-2 <—> EBCDIC สากล
IBM-875	UCS-2 <—> กรีก EBCDIC
IBM-930	UCS-2 <—> ญี่ปุ่น Katakana-Kanji EBCDIC
IBM-933	UCS-2 <—> เกาหลี EBCDIC
IBM-937	UCS-2 <—> จีนดั้งเดิม EBCDIC
IBM-939	UCS-2 <—> ญี่ปุ่น Latin-Kanji EBCDIC
IBM-1026	UCS-2 <—> เตอร์กิช EBCDIC
IBM-1112	UCS-2 <—> บัลติกหลายภาษา EBCDIC
IBM-1122	UCS-2 <—> เอสโทเนียน EBCDIC
IBM-1124	UCS-2 <—> ยูเครเนียน EBCDIC
IBM-1129	UCS-2 <—> เวียตนาม EBCDIC
TIS-620	UCS-2 <—>ประเทศไทย EBCDIC

หลักการที่เกี่ยวข้อง:

“ตัวแปลง interchange UTF-8”
ส่วนนี้จะอธิบายการแปลงที่ใช้ได้ในทั้งสองทิศทางระหว่างชุดโค้ด แต่ละชุดและ UTF-8

ตัวแปลง interchange UTF-8

ส่วนนี้จะอธิบายการแปลงที่ใช้ได้ในทั้งสองทิศทางระหว่างชุดโค้ด แต่ละชุดและ UTF-8

UTF-8 เป็นการเข้ารหัสแบบหลายไบต์สากล การแปลงสำหรับ ชุดโค้ดแต่ละชุดสามารถทำได้ในทั้งสองทิศทาง ระหว่างชุดโค้ดและ UTF-8

โดยปกติ การแปลง UTF-8 ทำโดยใช้ตัวแปลง Universal_UCS_Conv และ **/usr/lib/nls/loc/uconv/UTF-8**

ตัวแปลง	คำอธิบาย
ISO8859-1	UTF-8 <—> ISO ลาติน-1
ISO8859-2	UTF-8 <—> ISO ลาติน-2
ISO8859-3	UTF-8 <—> ISO ลาติน-3
ISO8859-4	UTF-8 <—> ISO บัลติก
ISO8859-5	UTF-8 <—> ISO Cyrillic
ISO8859-6	UTF-8 <—> ISO อารบิก
ISO8859-7	UTF-8 <—> ISO กรีก
ISO8859-8	UTF-8 <—> ISO ฮิบรู
ISO8859-9	UTF-8 <—> ISO เตอร์กิช
JISX0201.1976-0	UTF-8 <—> ญี่ปุ่น JISX0201-0
JISX0208.1983-0	UTF-8 <—> ญี่ปุ่น JISX0208-0
CNS11643.1986-1	UTF-8 <—> จีน CNS11643-1
CNS11643.1986-2	UTF-8 <—> จีน CNS11643-2
KSC5601.1987-0	UTF-8 <—> เกาหลี KSC5601-0
IBM-eucCN	UTF-8 <—> จีนแบบง่าย EUC
IBM-eucJP	UTF-8 <—> ญี่ปุ่น EUC
IBM-eucKR	UTF-8 <—> เกาหลี EUC
IBM-eucTW	UTF-8 <—> จีนดั้งเดิม EUC
IBM-udcJP	UTF-8 <—> อักขระญี่ปุ่น ที่ผู้ใช้กำหนด
IBM-udcTW	UTF-8 <—> อักขระจีนดั้งเดิม ที่ผู้ใช้กำหนด
IBM-sbdTW	UTF-8 <—> อักขระจีนดั้งเดิมที่ IBM ระบุ
UCS-2	UTF-8 <—> UCS-2
IBM-437	UTF-8 <—> โค้ดข้อมูล PC สหรัฐฯ
IBM-850	UTF-8 <—> โค้ดข้อมูล PC ลาติน 1

ตัวแปลง	คำอธิบาย
IBM-852	UTF-8 <—> โค้ดข้อมูล PC ลาติน 2
IBM-857	UTF-8 <—> โค้ดข้อมูล PC เตอร์กิช
IBM-860	UTF-8 <—> โค้ดข้อมูล PC โปรตุกิส
IBM-861	UTF-8 <—> โค้ดข้อมูล PC ไอซ์แลนดิก
IBM-863	UTF-8 <—> โค้ดข้อมูล PC ฝรั่งเศสสำเนียงแคนาดา
IBM-865	UTF-8 <—> โค้ดข้อมูล PC นอร์ดิก
IBM-868	UTF-8 <—> Urdu IBM-868
IBM-869	UTF-8 <—> โค้ดข้อมูล PC กรีก
IBM-918	UTF-8 <—> Urdu IBM-918
IBM-921	UTF-8 <—> โค้ดข้อมูล บัลติกหลายภาษา
IBM-922	UTF-8 <—> โค้ดข้อมูลเอสโทเนียน
IBM-932	UTF-8 <—> โค้ดข้อมูล PC ญี่ปุ่น
IBM-943	UTF-8 <—> โค้ดข้อมูล PC ญี่ปุ่น
IBM-934	UTF-8 <—> โค้ดข้อมูล PC เกาหลี
IBM-935	UTF-8 <—> จีนแบบง่าย EBCDIC
IBM-936	UTF-8 <—> โค้ดข้อมูล PC สาธารณรัฐประชาชนจีน
IBM-938	UTF-8 <—> โค้ดข้อมูล PC ไต้หวัน
IBM-942	UTF-8 <—> โค้ดข้อมูล PC ญี่ปุ่นแบบขยาย
IBM-944	UTF-8 <—> โค้ดข้อมูล PC เกาหลี
IBM-946	UTF-8 <—> โค้ดข้อมูล SAA สาธารณรัฐประชาชนจีน
IBM-948	UTF-8 <—> โค้ดข้อมูล PC จีนดั้งเดิม
IBM-1006	UTF-8 <—> Urdu IBM-1006
IBM-1124	UTF-8 <—> โค้ดข้อมูล PC ยูเครเนียน
IBM-1129	UTF-8 <—> โค้ดข้อมูล PC เวียตนาม
TIS-620	UTF-8 <—> โค้ดข้อมูล PC ประเทศไทย
IBM-037	UTF-8 <—> สหรัฐ, แคนาดา EBCDIC
IBM-273	UTF-8 <—> เยอรมนี, ออสเตรีย EBCDIC
IBM-277	UTF-8 <—> เดนมาร์ก, นอร์เวย์ EBCDIC
IBM-278	UTF-8 <—> ฟินแลนด์, สวีเดน EBCDIC
IBM-280	UTF-8 <—> อิตาลี EBCDIC
IBM-284	UTF-8 <—> สเปน, ลาตินอเมริกา EBCDIC
IBM-285	UTF-8 <—> สหราชอาณาจักร EBCDIC

ตัวแปลง	คำอธิบาย
IBM-297	UTF-8 <—> ฝรั่งเศส EBCDIC
IBM-500	UTF-8 <—> EBCDIC สากล
IBM-875	UTF-8 <—> กรีก EBCDIC
IBM-930	UTF-8 <—> ญี่ปุ่น Katakana-Kanji EBCDIC
IBM-933	UTF-8 <—> เกาหลี EBCDIC
IBM-937	UTF-8 <—> จีนดั้งเดิม EBCDIC
IBM-939	UTF-8 <—> ญี่ปุ่น Latin-Kanji EBCDIC
IBM-1026	UTF-8 <—> เตอร์กิช EBCDIC
IBM-1112	UTF-8 <—> บัลติกหลายภาษา EBCDIC
IBM-1122	UTF-8 <—> เอสโทเนียน EBCDIC
IBM-1124	UTF-8 <—> ยูเครเนียน EBCDIC
IBM-1129	UTF-8 <—> เวียตนาม EBCDIC
IBM-1381	UTF-8 <—> โค้ดข้อมูล PC จีนแบบง่าย
GB18030	UTF-8<—> จีนแบบง่าย
TIS-620	UTF-8 <—> ประเทศไทย EBCDIC

หลักการที่เกี่ยวข้อง:

“ตัวแปลงชุดโค้ดหลายไบต์” ในหน้า 106
ส่วนนี้อธิบายชุดโค้ดที่ใช้ตัวแปลงชุดโค้ดหลายไบต์ในการแปลง

“UCS-2 และ UTF-8” ในหน้า 89
AIX จะมีชุด ของชุดโค้ดที่กำหนดแอดเดรสของภาษาเฉพาะหรือ กลุ่มภาษา ไม่มีชุดโค้ดที่แสดงในตระกูล ISO8859 ของชุดโค้ด, ชุดโค้ด PC หรือชุดโค้ด Extended UNIX Code (EUC) อนุญาตให้มีการรวม อักขระจากสคริปต์ที่ต่างกัน ด้วย ISO8859-1 คุณจึงสามารถผสมและแสดงอักขระภาษาลาติน 1 (ภาษาที่พูดกันในสหรัฐฯ แคนาดา ยุโรปตะวันตก และลาตินอเมริกาเป็นหลัก) ISO8859-2 ครอบคลุมภาษายุโรปตะวันออก; ISO8859-5 ครอบคลุมภาษา Cyrillic, ISO8859-6 ครอบคลุมภาษาอารบิก, ISO8859-7 ครอบคลุมภาษากรีก, ISO8859-8 ครอบคลุมภาษาฮิบรู, ISO8859-9 ครอบคลุมภาษาเตอร์กิช, IBM-eucJP ครอบคลุม ภาษาญี่ปุ่น, IBM-eucKR ครอบคลุมภาษาเกาหลี, IBM-eucTW ครอบคลุมภาษาจีนดั้งเดิม ประเด็นคือไม่มี ชุดโค้ดใดๆ ด้านบนที่ครอบคลุมทุกภาษา

“ตัวแปลง interchange UCS-2” ในหน้า 122
UCS-2 ใช้การเข้ารหัส 16 บิตสากล การแปลงสำหรับ ชุดโค้ดแต่ละชุดสามารถทำได้ในทั้งสองทิศทาง ระหว่างชุดโค้ด และ UCS-2

ตัวแปลงเบ็ดเตล็ด

ชุดของตัวแปลงระดับต่ำที่ใช้โดยชุดโค้ดและ ตัวแปลง interchange ที่นำเสนอ ตัวแปลงเหล่านี้เรียกว่า *ตัวแปลง เบ็ดเตล็ด* ตัวแปลงระดับต่ำเหล่านี้อาจมีการใช้โดยตัวแปลง interchange บางตัว อย่างไรก็ตาม ไม่ส่งเสริมให้ใช้ตัวแปลงเหล่านี้ เนื่องจากตัวแปลงเหล่านี้มีไว้เพื่อสนับสนุนตัวแปลงอื่นๆ

ไฟล์

รายการนี้อธิบายตัวแปลงเบ็ดเตล็ดที่พบในไดเร็กทอรี **/usr/lib/nls/loc/iconv** และ **/usr/lib/nls/loc/iconvTable**

ตัวแปลง	คำอธิบาย
IBM-932_JISX0201.1976-0	IBM-932 เป็น JISX0201.1976-0
IBM-932_JISX0208.1983-0	IBM-932 เป็น JISX0208.1983-0
IBM-932_IBM-udcJP	IBM-932 เป็น IBM-udcJP (อักขระภาษาญี่ปุ่นที่ผู้ใช้กำหนด)
IBM-943_JISX0201.1976-0	IBM-943 เป็น JISX0201.1976-0
IBM-943_JISX0208.1983-0	IBM-943 เป็น JISX0208.1983-0
IBM-943_IBM-udcJP	IBM-943 เป็น IBM-udcJP (อักขระภาษาญี่ปุ่นที่ผู้ใช้กำหนด)
IBM-eucJP_JISX0201.1976-0	IBM-eucJP เป็น JISX0201.1976-0
IBM-eucJP_JISX0208.1983-0	IBM-eucJP เป็น JISX0208.1983-0
IBM-eucJP_IBM-udcJP	IBM-eucJP เป็น IBM-udcJP (อักขระภาษาญี่ปุ่นที่ผู้ใช้กำหนด)
IBM-eucKR_KSC5601.1987-0	IBM_eucKR เป็น KSC5601.1987-0
IBM-eucTW_CNS11643.1986-1	IBM-eucTW เป็น CNS11643.1986.1
IBM-eucTW_CNS11643.1986-2	IBM-eucTW เป็น CNS11643.1986-2
IBM-eucCN_GB2312.1980-0	IBM-eucCN เป็น GB2312.1980-0

การเขียนตัวแปลงโดยใช้อินเตอร์เฟส iconv

ส่วนนี้แสดงข้อมูลเกี่ยวกับรูทีนย่อย iconv และโครงสร้างในการเตรียมการสำหรับการเขียนตัวแปลงชุดโค้ด ข้อมูลที่รวม ในการอธิบายนี้คือ ภาพรวมของลำดับการควบคุมและลำดับซึ่ง เฟรมเวิร์คดำเนินการ รายละเอียดเกี่ยวกับการเขียนตัวแปลงชุดโค้ด และตัวอย่างรวมถึงโค้ด ไฟล์ส่วนหัว และไฟล์ที่จัดทำ ส่วนนี้ใช้กับเฟรมเวิร์ก **iconv** ภายใน AIX

ภายใต้เฟรมเวิร์คของรูทีนย่อย **iconv_open, iconv** และ **iconv_close** คุณสามารถสร้างและใช้ตัวแปลงได้หลายชนิด แอ็พพลิเคชัน สามารถเรียกรูทีนย่อยดังกล่าวเพื่อแปลงอักขระในชุดโค้ดหนึ่งเป็น อักขระในชุดโค้ดอื่นได้ การเข้าถึงและการใช้รูทีนย่อย **iconv_open, iconv** และ **iconv_close** มีลักษณะเหมือนกันโดย X/Open Portability Guide Issue 4

ชุดโค้ดและตัวแปลง

ชุดโค้ดสามารถจัดประเภทได้เป็นสองหมวดหมู่คือ: การเข้ารหัส stateful และการเข้ารหัส stateless

ชุดโค้ดและตัวแปลง

การเข้ารหัส stateful ใช้โค้ด shift-in และ shift-out เพื่อเปลี่ยนคำสั่ง Shift-out สามารถใช้เพื่อบ่งชี้การเริ่มต้นของโฮสต์ข้อมูลไบต์คู่ในสตรีมข้อมูลของอักขระ และ shift-in สามารถ ใช้เพื่อบ่งชี้การสิ้นสุดของข้อมูลอักขระไบต์คู่นี้ เมื่อปิด ข้อมูลไบต์คู่นั่นเป็นการส่งสัญญาณถึงการเริ่มต้นของข้อมูลอักขระ ไบต์เดียว ตัวอย่างของชุดโค้ด stateful ดังกล่าวคือ IBM-930 ที่ใช้บนเมนเฟรม (โฮสต์) เป็นหลัก

ตัวแปลงที่บันทึกเพื่อทำการแปลงการเข้ารหัส stateful เป็นชุดโค้ดอื่น ค่อนข้างซับซ้อนเนื่องจากต้องการการประมวลผล พิเศษ

ชุดโค้ดและตัวแปลง stateless

ส่วนนี้จะอธิบายชุดโค้ดและตัวแปลง stateless

ชุดโค้ด stateless คือชุดโค้ดที่สามารถจัดประเภทเป็นชนิดใดชนิดหนึ่ง ดังต่อไปนี้:

- ชุดโค้ดไบต์เดียว เช่น ตระกูล ISO8859 (ISO8859-1, ISO8859-2, และอื่นๆ)
- ชุดโค้ดหลายไบต์ เช่น IBM-eucJP (ภาษาญี่ปุ่น), IBM-932 (Shift-JIS)

หมายเหตุว่าการแปลงมีประโยชน์เฉพาะถ้าชุดโค้ดแสดงถึง อักขระเดียวกันเท่านั้น

ชนิดที่ง่ายที่สุดของการแปลงชุดโค้ดสามารถพบได้ในตัวแปลงชุดโค้ดไบต์เดียว เช่น ตัวแปลงจาก ISO8859-1 เป็น IBM-850 ตัวแปลงชุดโค้ดไบต์เดียวเหล่านี้ใช้การแปลงบนตาราง แบบง่าย โดยทั่วไป การแปลงของการเข้ารหัสอักขระแบบมัลติไบต์ เช่น IBM-eucJP เป็น IBM-932 ใช้ขั้นตอนวิธีการและไม่ได้ใช้ตาราง เนื่องจากตารางอาจมีความยาวมากเกินไป

ภาพรวมของโครงสร้างเฟรมเวิร์ค iconv

เฟรมเวิร์ค **iconv** ประกอบด้วยรูทีนย่อย **iconv_open, iconv** และ **iconv_close** และสร้างขึ้นตามข้อมูลโครงสร้างหลักทั่วไปซึ่งเป็นส่วนประกอบหนึ่งของตัวแปลงทุกตัว โครงสร้างหลักมีการเริ่มต้นเมื่อเวลาโหลดโมดูลอ็อบเจ็กต์ ตัวแปลง หลังจากการโหลดตัวแปลงเสร็จสมบูรณ์แล้ว จุดป้อนหลัก ซึ่งเป็นรูทีนย่อย **instantiate** เสมอ จะถูกเรียกใช้ ซึ่งจะเริ่มต้นโครงสร้างหลักและส่งคืนคำอธิบายตัวแปลง หลัก รูทีนย่อยนี้ยังมีการใช้ต่อไปในระหว่างการเรียกไปยังรูทีนย่อย **init** ที่นำเสนอโดยตัวแปลง เพื่อจัดสรรโครงสร้างเฉพาะตัวแปลง รูทีนย่อย **init** นี้ส่งคืนคำอธิบายตัวแปลงอื่น ซึ่งมีตัวชี้ไปยังคำอธิบายตัวแปลงหลัก รูทีนย่อย **init** จะจัดสรร หน่วยความจำตามที่ต้องการและอาจเรียกใช้ตัวแปลงอื่นๆ ถ้าจำเป็น รูทีนย่อย **init** คือสถานที่สำหรับการเริ่มต้นเฉพาะตัวแปลงใดๆ ในขณะที่รูทีนย่อย **instantiate** คือจุดป้อน ทั่วไป

หลังจากที่คำอธิบายตัวแปลงสำหรับตัวแปลงนี้มีการจัดสรรและ เริ่มต้นแล้ว ขั้นตอนถัดไปคือการนำเสนอโค้ดจริงที่ต้องการสำหรับส่วน **exec** ของฟังก์ชัน ถ้าตัวแปลงเป็นตัวแปลง บนตาราง สิ่งที่ต้องทำเพียงอย่างเดียวคือการระบุรูปแบบไฟล์ต้นฉบับที่สอดคล้องกับความต้องการอินพุตของยูทิลิตี้ **genxlt** ซึ่งจะใช้ตารางต้นฉบับนี้เป็นอินพุตและสร้างรูปแบบไฟล์เอาต์พุต ที่เฟรมเวิร์ค **iconv** สามารถใช้ได้

ไฟล์ iconv.h และโครงสร้าง

ส่วนนี้อธิบายไฟล์ iconv.h และโครงสร้าง

ไฟล์ **iconv.h** ใน **/usr/include** กำหนดโครงสร้าง ต่อไปนี้:

```
typedef struct __iconv_rec   iconv_rec, *iconv_t;
struct __iconv_rec   {
    _LC_object_t  hdr;
    iconv_t (*open)(const char *tocode, const char *fromcode);
    size_t (*exec)(iconv_t cd, char **inbuf, size_t *inbytesleft,
            char **outbuf, size_t *outbytesleft);
    void  (*close)(iconv_t cd);
};
```

โครงสร้างหลักทั่วไปเป็นดังนี้ (**/usr/include/iconv.h**):

```
typedef struct _LC_core_iconv_type      _LC_core_iconv_t;
struct _LC_core_iconv_type {
        _LC_object_t  hdr;
        /* การเริ่มต้นการดำเนินการ */
```

```
        _LC_core_iconv_t        *(*init)();
        size_t  (*exec)();
        void    (*close)();
};
```

ทุกตัวแปลงมีพื้นที่หน่วยความจำคงที่ ซึ่งมีโครงสร้าง **_LC_core_iconv_t** โครงสร้างนั้นมีการเริ่มต้นในรูทีนย่อย **instantiate** ที่จัดเตรียมให้เป็น ส่วนประกอบหนึ่งของโปรแกรมตัวแปลง

โฟลว์การควบคุม iconv

แอ็พพลิเคชันจะเรียกใช้ตัวแปลงชุดโค้ดโดยการเรียกดังต่อไปนี้:

```
iconv_open(char *to_codeset, char *from_codeset)
```

ชุดโค้ด *ปลายทาง* และ *ต้นทาง* ใช้ในการเลือกตัวแปลง โดยวิธีการค้นหาพาธที่กำหนดโดยตัวแปรสภาพแวดล้อม **LOCPATH** รูทีนย่อย **iconv_open** จะใช้รูทีนย่อย **_lc_load** เพื่อ โหลดอ็อบเจ็กต์โมดูลที่ระบุโดยชื่อชุด *ต้นทาง* และ *ปลายทาง* ที่ต่อกัน ไปยังรูทีนย่อย **iconv_open**

```
CONVERTER NAME= "from_codeset" + "_" +"to_codeset"
```

ถ้า *from_codeset* เป็น IBM-850 และ *to_codeset* เป็น ISO8859-1 ชื่อตัวแปลงจะเป็น IBM-850_ISO8859-1

หลังจากการโหลดตัวแปลง จุดป้อนของตัวแปลง จะถูกเรียกใช้โดยรูทีนย่อยตัวโหลด **_lc_load** นี่เป็นการเรียกครั้งแรก ไปยังตัวแปลง จากนั้น รูทีนย่อย instantiate จะเริ่มต้นโครงสร้างหลัก **_LC_core_iconv_t** จากนั้น รูทีนย่อย **iconv_open** จะเรียก รูทีนย่อย **init** ที่เชื่อมโยงกับโครงสร้างหลักที่จะ ส่งคืน รูทีนย่อย init จะจัดสรรโครงสร้างคำอธิบายเฉพาะตัวแปลง และเริ่มต้นโครงสร้างนั้นตามที่ตัวแปลงต้องการ รูทีนย่อย **iconv_open** ส่งคืนโครงสร้างเฉพาะตัวแปลงนี้ อย่างไรก็ตาม ค่าที่ส่งคืนมีการ typecast ไปยัง **iconv_t** ในแอ็พพลิเคชัน ของผู้ใช้ ดังนั้น แอ็พพลิเคชันจึงไม่ได้เห็นโครงสร้างเฉพาะตัวแปลงครบทั้งหมด แต่เห็นเฉพาะโครงสร้าง **iconv_t** สาธารณะเท่านั้น โค้ดตัวแปลงเอง จะใช้โครงสร้างตัวแปลงส่วนตัว แอ็พพลิเคชันที่ใช้ตัวแปลง **iconv** ไม่ควรเปลี่ยนคำอธิบายตัวแปลง แต่ควรจะใช้ คำอธิบายตัวแปลงเป็นโครงสร้าง opaque

จุดป้อน มีการประกาศอยู่ในทุกตัวแปลง เพื่อที่ว่าเมื่อเปิดตัวแปลง โดยการเรียกไปยังรูทีนย่อย *iconv_open* จุดป้อนจะถูกเรียกใช้โดยอัตโนมัติ จุดป้อนเป็นรูทีนย่อย *instantiate* ที่ควรจะมีอยู่ในตัวแปลงทุกตัว จุดป้อนมีการระบุใน makefile ดังนี้:

```
LDENTRY=-einstantiate
```

เมื่อโหลดตัวแปลงในการเรียกไปยังรูทีนย่อย **iconv_open** รูทีนย่อย **instantiate** จะถูกเรียกใช้ รูทีนย่อยนี้จะเริ่มต้น โครงสร้างคำอธิบายการแปลงหลักคงที่ **_LC_core_iconv_t cd**

คำอธิบายการแปลงหลัก **cd** มีตัวชี้ไปยังรูทีนย่อย **init**, **_iconv_exec,** และ _iconv_close ที่นำเสนอโดยตัวแปลงเฉพาะ รูทีนย่อย **instantiate** จะส่งคืน คำอธิบายการแปลงหลักที่จะใช้ในภายหลัง โครงสร้าง _LC_core_iconv_t มีการกำหนดไว้ใน **/usr/include/iconv.h**

เมื่อเรียกรูทีนย่อย **iconv_open** การดำเนินการต่อไปนี้ จะเกิดขึ้น:

1. พบตัวแปลงโดยใช้ตัวแปรสภาพแวดล้อม **LOCPATH** โหลดตัวแปลง จากนั้นเรียกใช้รูทีนย่อย **instantiate** เมื่อสำเร็จแล้ว รูทีนย่อยจะส่งคืนคำอธิบายการแปลงหลัก (**_LC_core_iconv_t *cd**) รูทีนย่อย **instantiate** ที่นำเสนอโดยตัวแปลง รับผิดชอบในการเริ่มต้นส่วนหัวในโครงสร้างหลัก

2. จากนั้น รูทีนย่อย **iconv_open** จะเรียกใช้รูทีนย่อย **init** ที่ระบุไว้ในคำอธิบายการแปลงหลัก รูทีนย่อย **init** ที่นำเสนอ โดยตัวแปลงรับผิดชอบในการจัดสรรหน่วยความจำที่ต้องใช้สำหรับการจัดเก็บ คำอธิบายตัวแปลงที่ต้องการสำหรับตัวแปลงเฉพาะนี้ ตัวอย่างเช่น ข้อมูลต่อไปนี้อาจเป็นโครงสร้างที่ตัวแปลง stateless ต้องการ:

```
typedef struct _LC_sample_iconv_rec {
                  LC_core_iconv_t        core;

            } _LC_sample_iconv_t;

เมื่อต้องการเริ่มต้นสิ่งนี้ ตัวแปลงต้องทำดังต่อไปนี้ในรูทีนย่อย
init:

static _LC_sample_iconv_t*
init (_LC_core_iconv_t *core_cd, char* toname,char* fromname)
{
   _LC_sample_iconv_t     *cd;    /* คำอธิบายตัวแปลง */

    /*
    **         จัดสรรคำอธิบายตัวแปลง
    **/
    if(!(cd = ( _LC_sample_iconv_t *) malloc (
                   sizeof(_LC_sample_iconv_t ))))
            return (NULL);

    /*
    ** คัดลอกส่วนหลักของคำอธิบายตัวแปลงซึ่ง
    ** ส่งเข้ามา
    */
    cd-&gt;core = *core_cd;
    /*
    **       ส่งคืนคำอธิบายตัวแปลง
    */
    return cd;
}
```

แอ็พพลิเคชันเรียกใช้รูทีนย่อย **iconv** เพื่อทำการแปลงชุดโค้ดที่ แท้จริง รูทีนย่อย **iconv** เรียกใช้รูทีนย่อย **exec** ในโครงสร้างหลัก

แอ็พพลิเคชันเรียกใช้รูทีนย่อย **iconv_close** เพื่อทำให้หน่วยความจำ ใดๆ ที่จัดสรรสำหรับการแปลงว่าง รูทีนย่อย **iconv_close** เรียกใช้ รูทีนย่อย **close** ในโครงสร้างหลัก

การเขียนตัวแปลงชุดโค้ด

ส่วนนี้แสดงข้อมูลเกี่ยวกับวิธีการเขียนตัวแปลง โดยใช้หลักการที่ได้อธิบายไปแล้ว

ทุกตัวแปลงควรกำหนดรูทีนย่อยดังต่อไปนี้:

- **instantiate**
- **init**
- **iconv_exec**
- **iconv_close**

โครงสร้างเฉพาะตัวแปลงควรมีโครงสร้าง **iconv** หลัก เป็นองค์ประกอบแรก ตัวอย่างเช่น:

```
typedef struct _LC_example_rec {
    /* หลักควรเป็นองค์ประกอบแรก */
        _LC_core_iconv_t        core;
    /* ส่วนที่เหลือเป็นข้อมูลเฉพาะตัวแปลง (เป็นอ็อพชัน) */
        iconv_t         curcd;
        iconv_t         sb_cd;
        iconv_t         db_cd;
        unsigned char   *cntl;
} _LC_example_iconv_t;
```

โครงสร้างตัวแปลงอื่น:

```
typedef struct _LC_sample_iconv_rec {
        _LC_core_iconv_t        core;
} _LC_sample_iconv_t;
```

ตัวแปลงแบบไม่มีสภาวะตามอัลกอริธึม

ตัวแปลงทุกตัวควรจะมีรูทีนย่อยที่ระบุไว้ ก่อนหน้านี้ มีการแสดงเฉพาะส่วนหัวรูทีนย่อยโดยไม่มีรายละเอียด ยกเว้นสำหรับรูทีนย่อย **instantiate** ซึ่งมีอยู่ทั่วไปสำหรับทุก ตัวแปลงและควรจะมีการเข้ารหัสในลักษณะเดียวกัน

ตัวอย่างต่อไปนี้ของตัวแปลง algorithm-based stateless เป็นตัวอย่างตัวแปลงจากชุดโค้ด IBM-850 เป็นชุดโค้ด ISO8859-1

```
#include <stdlib.h>
#include <iconv.h>
#include "850_88591.h"
/*
 *      Name :  _iconv_exec()
 *
 *      ข้อมูลนี้มีวิธีการแปลงจริง
 */
static size_t   _iconv_exec(_LC_sample_iconv_t *cd,
                        unsigned char** inbuf,
                        size_t *inbytesleft,
                        unsigned char** outbuf,
                        size_t *outbytesleft)
/*
 *      cd              : คำอธิบายตัวแปลง
 *      inbuf           : อินพุตบัฟเฟอร์
 *      outbuf          : เอาต์พุตบัฟเฟอร์
 *      inbytesleft     : จำนวนของข้อมูล (ในหน่วยไบต์) ในอินพุตบัฟเฟอร์
 *      outbytesleft    : จำนวนของข้อมูล (ในหน่วยไบต์) ในเอาต์พุตบัฟเฟอร์
 */

{
}

/*
 *      Name :   _iconv_close()
 *
 *      ทำให้คำอธิบายตัวแปลงที่จัดสรรว่าง
 */
static void      _iconv_close(iconv_t cd)
```

```
{
}

/*
 *      Name :  init()
 *
 *      ข้อมูลนี้จัดสรรและเริ่มต้นคำอธิบายตัวแปลง
 */
static _LC_sample_iconv_t        *init (_LC_core_iconv_t *core_cd,
                                  char* toname, char* fromname)
{
}

/*
 *       Name :  instantiate()
 *
 *       ส่วนหลักของคำอธิบายตัวแปลงมีการเริ่มต้นที่นี่
 */
_LC_core_iconv_t         *instantiate(void)
{
        static _LC_core_iconv_t   cd;

        /*
        * * การเริ่มต้น _LC_MAGIC และ _LC_VERSION มีการ
        ** กำหนดไว้ใน <lc_core.h> _LC_ICONV และ _LC_core_iconv_t
        ** มีการกำหนดไว้ใน <iconv.h>
         */
        cd.hdr.magic = _LC_MAGIC;
        cd.hdr.version = _LC_VERSION;
        cd.hdr.type_id = _LC_ICONV;
        cd.hdr.size = sizeof (_LC_core_iconv_t);

        /*
         *      ตั้งค่าตัวชี้ไปยังแต่ละวิธีการ
         */
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;

        /*
         *      กลับไปยังส่วนหลัก
         */
        return &cd;
}
```

ตัวแปลง Stateful

เนื่องจากตัวแปลง stateful ต้องการข้อมูลเพิ่มเติม ตัวแปลงเหล่านี้จึงให้ข้อมูลอิงตัวแปลงเพิ่มเติม ตัวอย่าง ต่อไปนี้ของตัวแปลง stateful เป็นตัวอย่างตัวแปลงจากชุดโค้ด IBM-930 เป็น IBM-932

ไฟล์ **host.h** มีโครงสร้างดังต่อไปนี้:

```
typedef struct _LC_host_iconv_rec {
                _LC_core_iconv_t        core;
                iconv_t         curcd;
```

```
                iconv_t         sb_cd;
                iconv_t         db_cd;
                unsigned char   *cntl;
        } _LC_host_iconv_t;

#include <stdlib.h&gt;
#include <sys/types.h&gt;
#include <iconv.h&gt;
#include "host.h"

/*
** รูทีนย่อย _iconv_exec ที่จะเรียกใช้ผ่านทาง cd-&gt;exec()
*/
static int      _iconv_exec(_LC_host_iconv_t *cd,
        unsigned char **inbuf, size_t *inbytesleft,
        unsigned char **outbuf, size_t *outbytesleft)
{
        unsigned char    *in, *out;
        int              ret_value;

    if (!cd){
        errno = EBADF; return NULL;
    }

    if (!inbuf) {
        cd-&gt;curcd = cd-&gt;sb_cd;
        return ICONV_DONE;
    }

    do {
    if ((ret_value = iconv(cd-&gt;curcd, inbuf, inbytesleft, outbuf,
            outbytesleft)) != ICONV_INVAL)
            return ret_value;
        in = *inbuf;
        out = *outbuf;
        if (in[0] == SO) {
            if (cd-&gt;curcd == cd-&gt;db_cd){
                errno = EILSEQ;
                return ICONV_INVAL;
            }
            cd-&gt;curcd = cd-&gt;db_cd;
        }
        else if (in[0] == SI) {
            if (cd-&gt;curcd == cd-&gt;sb_cd){
                errno = EILSEQ;
                return ICONV_INVAL;
            }
                cd-&gt;curcd = cd-&gt;sb_cd;
        }else if (in[0] <= 0x3f &&
            cd-&gt;curcd == cd-&gt;sb_cd) {
            if (*outbytesleft < 1){
                errno = E2BIG;
                return ICONV_OVER;
            }
                out[0] = cd-&gt;cntl[in[0]];
```

```
                *outbuf = ++out;
                (*outbytesleft)--;
        }
        else {
            errno = EILSEQ; return ICONV_INVAL;
        }
        *inbuf = ++in;
        (*inbytesleft)--;
    } while (1);
}

/*
** รูทีนย่อย iconv_close คือแมโครการเข้าถึงรูทีนย่อยนี้
** ตามที่ตั้งค่าไว้ในโครงสร้าง iconv หลัก
*/
static void   _iconv_close(_LC_host_iconv_t *cd)
{
    if (cd) {
        if (cd-&gt;sb_cd)
            iconv_close(cd-&gt;sb_cd);
        if (cd-&gt;db_cd)
            iconv_close(cd-&gt;db_cd);
        free(cd);
    }else{
        errno = EBADF;
    }
}

/*
** รูทีนย่อย init ที่จะเรียกใช้เมื่อเรียก iconv_open()
*/
static _LC_host_iconv_t   *init(_LC_core_iconv_t *core_cd,
               char* toname, char* fromname)
{
        _LC_host_iconv_t*  cd;
        int   i;

    for (i = 0; 1; i++) {
        if (!_iconv_host[i].local)
            return NULL;
        if (strcmp(toname, _iconv_host[i].local) == 0 &&
            strcmp(fromname, _iconv_host[i].host) == 0)
            break;
    }

        if (!(cd = (_LC_host_iconv_t *)
                        malloc(sizeof(_LC_host_iconv_t))))
                return (NULL);

    if (!(cd-&gt;sb_cd = iconv_open(toname, _iconv_host[i].sbcs))) {
        free(cd);
        return NULL;
    }
    if (!(cd-&gt;db_cd = iconv_open(toname, _iconv_host[i].dbcs))) {
        iconv_close(cd-&gt;sb_cd);
```

```
        free(cd);
        return NULL;
    }
            cd-&gt;core = *core_cd;
    cd-&gt;cntl = _iconv_host[i].fcntl;
    cd-&gt;curcd = cd-&gt;sb_cd;
    return cd;
}

/*
** วิธีการ instantiate() มีการเรียกเมื่อ iconv_open() โหลด
** ตัวแปลงโดยการเรียกไปยัง __lc_load()
*/
_LC_core_iconv_t         *instantiate(void)
{
        static _LC_core_iconv_t
cd;

        cd.hdr.magic = _LC_MAGIC;
        cd.hdr.version = _LC_VERSION;
        cd.hdr.type_id = _LC_ICONV;
        cd.hdr.size = sizeof (_LC_core_iconv_t);
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;
        return &cd;
}
```

ตัวอย่าง

ตัวอย่างนี้แสดงตัวอย่างโค้ดสำหรับตัวแปลง stateless ที่ดำเนินการแปลงตามขั้นตอนวิธีการจากชุดโค้ด IBM-850 เป็น ชุดโค้ด ISO8859-1

ชื่อไฟล์สำหรับตัวอย่างนี้คือ 850_88591.c

```
#include <stdlib.h>
#include <iconv.h>
#include "850_88591.h"

#define DONE     0

/*
 *  Name :  _iconv_exec()
 *
 *      ข้อมูลนี้มีวิธีการแปลงจริง
 */
static size_t  _iconv_exec(_LC_sample_iconv_t *cd,
      unsigned char** inbuf, size_t *inbytesleft,
      unsigned char** outbuf, size_t *outbytesleft)
/*
 *      cd              : คำอธิบายตัวแปลง
 *      inbuf           : อินพุตบัฟเฟอร์
 *      outbuf          : เอาต์พุตบัฟเฟอร์
 *      inbytesleft     : จำนวนของข้อมูล (ในหน่วยไบต์) ในอินพุตบัฟเฟอร์
 *      outbytesleft    : จำนวนของข้อมูล (ในหน่วยไบต์) ในเอาต์พุตบัฟเฟอร์
```

```
 */
{
        unsigned char   *in;   /* ชี้อินพุตบัฟเฟอร์ */
        unsigned char   *out;  /* ชี้เอาต์พุตบัฟเฟอร์ */
        unsigned char   *e_in; /* ชี้สิ้นสุดของอินพุตบัฟเฟอร์*/
        unsigned char   *e_out; /* ชี้สิ้นสุดของเอาต์พุตบัฟเฟอร์*/

    /*
     *   ถ้าคำอธิบายตัวแปลงที่กำหนดไม่ถูกต้อง
     *   ระบบจะกำหนด errno และส่งคืนจำนวน
     *   ไบต์ส่วนที่เหลือซึ่งจะแปลง
     */
        if (!cd) {
        errno = EBADF;
        return *inbytesleft;
        }

        /*
         *      ถ้าอินพุตบัฟเฟอร์ไม่มีอยู่หรือ
         *      ไม่มีอักขระที่จะแปลง ระบบจะส่งคืน
         *      0 (ไม่มีอักขระที่จะแปลง)
         */
        if (!inbuf || !(*inbytesleft))
                return DONE;

        /*
         *      ตั้งค่าตัวชี้และเริ่มต้นตัวแปรอื่น
         */
        e_in = (in = *inbuf) + *inbytesleft;
        e_out = (out = *outbuf) + *outbytesleft;

     /*
      *     ดำเนินการแปลงจุดโค้ดจนกว่าจะใช้
      *     อินพุตทั้งหมดแล้ว
      *     เมื่อเกิดข้อผิดพลาดขึ้น (เช่น บัฟเฟอร์โอเวอร์โฟลว์) จะมีการ
      *     กำหนดหมายเลขข้อผิดพลาดและออกจากลูปนี้
      */
         while (in < e_in) {

        /*
         *   ถ้าพื้นที่ว่างที่เหลืออยู่ในเอาต์พุตบัฟเฟอร์ไม่เพียงพอ
         *   สำหรับรองรับข้อมูลที่แปลง ระบบจะหยุดการแปลงและ
         *   กำหนด errno เป็น E2BIG
         */
        if (e_out <= out) {
            errno = E2BIG;
            break;
        }

        /*
         *   แปลงข้อมูลอินพุตและจัดเก็บไว้ในเอาต์พุต
         *   บัฟเฟอร์ จากนั้นเลื่อนตัวชี้ซึ่งชี้ไปยัง
         *   บัฟเฟอร์ไปข้างหน้า
         */
        *out++ = table[*in++];
```

```
    }   /* ในขณะที่ */

    /*
     *  อัพเดตตัวชี้ไปยังบัฟเฟอร์และ
     *  นับไบต์อินพุต /เอาต์พุต
     */
     *inbuf = in;
        *outbuf = out;
    *inbytesleft = e_in - in;
    *outbytesleft = e_out - out;

    /*
     *   ส่งคืนจำนวนไบต์ส่วนที่เหลือซึ่งจะแปลง
     *   (0 สำหรับการแปลงที่เสร็จสมบูรณ์แล้ว)
     */
    return *inbytesleft;
}

/*
 *  Name :  _iconv_close()
 *
 *      ทำให้คำอธิบายตัวแปลงที่จัดสรรว่าง
 */
static void _iconv_close(iconv_t cd)
{
    if (!cd)
        free(cd);
    else
        /*
         *  ถ้าตัวแปลงที่กำหนดไม่ถูกต้อง
         *  ระบบจะกำหนด errno เป็น EBADF
         */
        errno = EBADF;
}

/*
 *  Name :  init()
 *
 *      ข้อมูลนี้จัดสรรและเริ่มต้นคำอธิบายตัวแปลง
 */
static _LC_sample_iconv_t*
init (_LC_core_iconv_t *core_cd,  char* toname, char* fromname)
{
        _LC_sample_iconv_t *cd; /* คำอธิบายตัวแปลง */

    /*
     *   จัดสรรคำอธิบายตัวแปลง
     */
        if (!(cd = (_LC_sample_iconv_t *)
                        malloc(sizeof(_LC_sample_iconv_t))))
                return (NULL);

    /*
     *คัดลอกส่วนหลักของคำอธิบายตัวแปลงซึ่งมีการจัดส่ง        *ใน
```

```
     */
        cd->core = *core_cd;

    /*
     *   ส่งคืนคำอธิบายตัวแปลง
     */
        return cd;
}

/*
 *  Name :  instantiate()
 *
 *       ส่วนหลักของคำอธิบายตัวแปลงมีการเริ่มต้นที่นี่
 */
_LC_core_iconv_t*   instantiate(void)
{
        static _LC_core_iconv_t  cd;

    /*
     *  เริ่มต้น
     *  _LC_MAGIC และ _LC_VERSION มีการกำหนดไว้ใน <lc_core.h>
     *  _LC_ICONV และ _LC_core_iconv_t มีการกำหนดไว้ใน <iconv.h>
     */
    cd.hdr.magic = _LC_MAGIC;
    cd.hdr.version = _LC_VERSION;
    cd.hdr.type_id = _LC_ICONV;
    cd.hdr.size = sizeof (_LC_core_iconv_t);

    /*
     *      ตั้งค่าตัวชี้ไปยังแต่ละวิธีการ
     */
        cd.init = init;
        cd.exec = _iconv_exec;
        cd.close = _iconv_close;

    /*
     *  ส่งคืนส่วนหลัก
     */
        return &cd;
}
```

ตัวอย่างนี้มีตัวอย่างไฟล์ส่วนหัวที่ชื่อว่า 850_88591.h

```
#ifndef _ICONV_SAMPLE_H
#define _ICONV_SAMPLE_H

/*
 *      Define _LC_sample_iconv_t
 */
typedef struct _LC_sample_iconv_rec {
        _LC_core_iconv_t   core;
} _LC_sample_iconv_t;

static unsigned char     table[] = { /*
```

```
        _____________________________________
        |                                    |
        |      IBM-850              ISO8859-1  |
        |____________________________________|
        /*      0x00     */              0x00,
        /*      0x01     */              0x01,
        /*      0x02     */              0x02,
        /*      0x03     */              0x03,
        /*      0x04     */              0x04,
        /*      0x05     */              0x05,
        /*      0x06     */              0x06,
        /*      0x07     */              0x07,
        /*      0x08     */              0x08,
        /*      0x09     */              0x09,
        /*      0x0A     */              0x0A,
        /*      0x0B     */              0x0B,
        /*      0x0C     */              0x0C,
        /*      0x0D     */              0x0D,
        .
        .
        .
        /*      0xF3     */              0xBE,
        /*      0xF4     */              0xB6,
        /*      0xF5     */              0xA7,
        /*      0xF6     */              0xF7,
        /*      0xF7     */              0xB8,
        /*      0xF8     */              0xB0,
        /*      0xF9     */              0xA8,
        /*      0xFA     */              0xB7,
        /*      0xFB     */              0xB9,
        /*      0xFC     */              0xB3,
        /*      0xFD     */              0xB2,
        /*      0xFE     */              0x1A,
        /*      0xFF     */              0xA0,
};
#endif
```

ตัวอย่างนี้คือตัวอย่างไฟล์ที่จัดทำขึ้น

```
SHELL   = /bin/ksh
CFLAGS  = $(COMPOPT) $(INCLUDE) $(DEFINES)
INCLUDE = -I.
COMPOPT =
DEFINES = -D_POSIX_SOURCE -D_XOPEN_SOURCE
CC      = /bin/xlc
LD      = /bin/ld
RM      = /bin/rm

SRC     = 850_88591.c
TARGET  = 850_88591

ENTRY_POINT       = instantiate

$(TARGET) :
       cc -e $(ENTRY_POINT) -o $(TARGET) $(SRC) -l iconv
```

```
clean :
        $(RM) -f $(TARGET)
        $(RM) -f *.o
```

วิธีอินพุต

สำหรับแอ็พพลิเคชันที่จะรันในสภาวะแวดล้อมระหว่างประเทศที่ globalization เป็นพื้นฐาน จำเป็นต้องใช้วิธีการอินพุต วิธีการอินพุตคืออินเตอร์เฟสการเขียนโปรแกรมแอ็พพลิเคชัน (API) ที่ช่วยให้คุณสามารถจัดทำภาษา คีย์บอร์ด หรือชุดโค้ดเฉพาะที่ เป็นอิสระจากแอ็พพลิเคชัน

วิธีการอินพุตแต่ละชนิดมีคุณลักษณะดังต่อไปนี้:

ชนิดวิธีอินพุต	คุณลักษณะ
คีย์แม็พ	ชุดของคีย์แม็พวิธีการอินพุต (imkeymaps) ที่ทำงานกับ วิธีการอินพุตและกำหนดโลแคลที่สนับสนุน
Keysyms	ชุดของสัญลักษณ์คีย์ (keysyms) ที่วิธีการอินพุตสามารถจัดการได้
ตัวดัดแปลง	ชุดของตัวดัดแปลงหรือคำสั่ง แต่ละตัวมีค่า mask ซึ่งวิธีการอินพุต สนับสนุน

หลักการที่เกี่ยวข้อง:

"การสนับสนุนวิธีการอินพุต" ในหน้า 2
อินพุตของอักขระมีความซับซ้อนมากขึ้นสำหรับภาษา ที่มีชุดอักขระจำนวนมาก ตัวอย่างเช่น ในภาษาจีน ภาษาเกาหลี และภาษาญี่ปุ่น ที่มีอักขระเป็นจำนวนมากและไม่สามารถแม็พคีย์แบบหนึ่งต่อหนึ่ง สำหรับ keystroke ของอักขระ อย่างไรก็ตาม วิธีการอินพุตพิเศษช่วยให้ผู้ใช้สามารถป้อนอักขระการออกเสียง หรือ stroke และแปลงอักขระนั้นเป็นอักขระภาษา แม่

"Keysyms ที่สงวนไว้" ในหน้า 172
ส่วนนี้อธิบาย keysyms ที่สงวนไว้โดย วิธีการอินพุต

"AIXwindowsรายการตรวจสอบ" ในหน้า 212
ไอเท็มรายการตรวจสอบที่เหลือเป็นไอเท็มเฉพาะสำหรับระบบ AIXwindows

บทนำวิธีการอินพุต

วิธีการอินพุตคือชุดของฟังก์ชันที่แปล key strokes เป็นสตริงอักขระในชุดโค้ดที่ระบุโดย โลแคลของคุณ ฟังก์ชันวิธีการอินพุตมีการประมวลผลอินพุตเฉพาะโลแคล และตัวควบคุมบนคีย์บอร์ด (เช่น Ctrl, Alt, Shift, Lock, และ Alt-Graphic) วิธีการอินพุตอนุญาตให้ใช้อินพุตได้หลายชนิด แต่ในส่วนนี้จะอธิบายถึงเหตุการณ์คีย์บอร์ดเท่านั้น

โลแคลของคุณกำหนดวิธีการอินพุตที่ควรจะโหลด วิธีการรัน วิธีการอินพุต และอุปกรณ์ที่จะใช้ จากนั้น วิธีการอินพุต จะกำหนดคำสั่งและผลที่ได้ของคำสั่งนั้น

เมื่อวิธีการอินพุตแปล keystroke เป็นสตริงอักขระ กระบวนการแปลจะพิจารณาถึงคีย์บอร์ดและชุดโค้ด ที่คุณกำลังใช้อยู่ คุณสามารถบันทึกวิธีการอินพุตของคุณเองถ้าคุณไม่มี คีย์บอร์ดมาตรฐาน หรือถ้าคุณกำหนดชุดโค้ดของคุณเอง

ภาษาหลายภาษาใช้ชุดสัญลักษณ์หรือตัวอักษรจำนวนไม่มากนักเพื่อสร้างคำ เมื่อต้องการป้อนข้อความโดยใช้คีย์บอร์ด ให้คุณกดคีย์ที่สอดคล้องกับ สัญลักษณ์ของตัวอักษร ถ้าอักขระในตัวอักษรของคุณไม่มีอยู่ บนคีย์บอร์ด คุณต้องกดคีย์เป็นชุด วิธีการอินพุตนำเสนอขั้นตอนวิธีการที่ช่วยคุณเรียบเรียงอักขระดังกล่าว

บางภาษาใช้ระบบการบันทึกทางภูมิศาสตร์โดยการใช้สัญลักษณ์เฉพาะ แทนกลุ่มของตัวอักษร เพื่อแสดงถึงคำ ตัวอย่าง เช่น ชุดอักขระที่ใช้กันในประเทศจีนแผ่นดินใหญ่ ญี่ปุ่น เกาหลี และไต้หวันมีอักขระมากกว่า 5,000 ตัว ในเวลาต่อมา สามารถใช้มากกว่า หนึ่งไบต์เพื่อแสดงถึงอักขระหนึ่งตัว ยิ่งไปกว่านั้น คีย์บอร์ดหนึ่งตัว ไม่สามารถรวมสัญลักษณ์ทางภูมิศาสตร์ที่ต้องการได้ครบทั้งหมด คุณจึง ต้องใช้วิธีการอินพุตซึ่งสามารถเรียบเรียงอักขระหลายไบต์ได้

ไดเร็กทอรี **/usr/lib/nls/loc** มีวิธีการอินพุต ที่ติดตั้งไว้บนระบบของคุณ คุณสามารถแสดงรายการเนื้อหาของไดเร็กทอรีนี้เพื่อกำหนดวิธีการอินพุตที่คุณมีอยู่ ชื่อไฟล์วิธีการอินพุต มีรูปแบบ *Language_Territory*.**im** ตัวอย่าง เช่น ไฟล์ **fr_BE.im** คือไฟล์วิธีการอินพุตสำหรับภาษาฝรั่งเศส ตามที่ใช้กันในประเทศเบลเยียม

โดยใช้โปรโตคอลที่จัดโครงสร้างเป็นอย่างดี วิธีการอินพุตช่วยให้แอ็พพลิเคชัน สามารถสนับสนุนอินพุตที่แตกต่างอื่นได้โดยใช้การประมวลผลอินพุตเฉพาะโลแคล

ใน AIX วิธีการอินพุตจะมีอยู่ใน aixterm เมื่ออักขระ ที่พิมพ์จากอินเตอร์เฟส AIXwindows เข้าถึงเซิร์ฟเวอร์ อักขระ อยู่ในรูปแบบของรหัสคีย์ ตารางที่แสดงอยู่ในไคลเอ็นต์จะแปลงรหัสคีย์ เป็น *keysyms* ซึ่งเป็นชุดของโค้ดที่กำหนดไว้แล้ว รหัสคีย์ใดๆ ที่สร้างขึ้นโดยคีย์บอร์ดควรจะมี keysym Keysyms เหล่านี้มีการเก็บรักษาและจัดสรรโดย MIT X Consortium Keysyms ถูกส่งผ่านไปยังไคลเอ็นต์ aixterm terminal emulator ใน aixterm อินพุต keysyms จะถูกแปลงเป็นโค้ดไฟล์โดยวิธีการอินพุต จากนั้นจึงส่งไปยังแอ็พพลิเคชัน เซิร์ฟเวอร์ X ได้รับการออกแบบมาเพื่อ ทำงานกับอะแดปเตอร์จอแสดงผลที่มีอยู่ในฮาร์ดแวร์ระบบ เซิร์ฟเวอร์ X สื่อสารกับไคลเอ็นต์ X ผ่านทางซอกเก็ต ดังนั้น เซิร์ฟเวอร์และไคลเอ็นต์จึงสามารถอยู่บนระบบที่แตกต่างกันในเครือข่ายได้ หากว่าทั้งสองสามารถสื่อสารกันได้ ข้อมูลจากคีย์บอร์ด จะเข้าไปที่เซิร์ฟเวอร์ X และจากเซิร์ฟเวอร์ ข้อมูลจะถูกส่งผ่านไปยัง terminal emulator Terminal emulator จะส่งผ่านข้อมูลไปยัง แอ็พพลิเคชัน เมื่อข้อมูลมาจากแอ็พพลิเคชันไปยังอุปกรณ์แสดงผล ข้อมูลจะส่งผ่านผ่านทาง terminal emulator โดยซอกเก็ตไปยังเซิร์ฟเวอร์ และออกจากเซิร์ฟเวอร์ไปยังอุปกรณ์แสดงผล

ชื่อวิธีการอินพุต

ชุดของวิธีการอินพุตที่ใช้ได้ขึ้นอยู่กับว่ามีการติดตั้งบนโลแคลใด และโลแคลเหล่านั้นนำเสนอวิธีการอินพุตแบบใด โดยปกติแล้ว ชื่อของวิธีการอินพุตสอดคล้องกับโลแคล ตัว อย่างเช่น วิธีการอินพุตภาษากรีกมีการตั้งชื่อว่า el_GR ซึ่งเหมือนกับ โลแคลสำหรับภาษากรีกที่พูดกันในประเทศกรีซ

เมื่อมีวิธีการอินพุตมากกว่าหนึ่งวิธีบนโลแคล วิธีการอินพุตรองจะมีการระบุโดย ใช้ตัวดัดแปลงที่เป็นส่วนประกอบหนึ่งของชื่อโลแคล ตัวอย่างเช่น โลแคลภาษาฝรั่งเศสตามที่พูดกันในประเทศแคนาดา มีวิธีการ อินพุตอยู่สามวิธี คือวิธีการดีฟอลต์และวิธีการอื่นอีกสองวิธี ชื่อ วิธีการอินพุตมีดังนี้:

ชื่อวิธีการอินพุต	**คำอธิบาย**
fr_CA	วิธีการอินพุตดีฟอลต์
fr_CA@im=alt	วิธีการอินพุตอื่น
fr_CA.im__64	วิธีการอินพุต 64 บิต

ส่วน **fr** ของโลแคลแสดงถึงชื่อภาษา (ฝรั่งเศส) และ **CA** แสดงถึงชื่อเขตแดน (แคนาดา) สตริง **@im=alt** คือส่วนตัวดัดแปลงของโลแคลที่ ใช้เพื่อระบุวิธีการอินพุตอื่น สตริงตัวดัดแปลงทั้งหมด มีการระบุโดยใช้รูปแบบ **@im=***ตัวดัดแปลง*

เนื่องจากวิธีการอินพุตเป็นโมดูลอ็อบเจ็กต์ที่โหลดได้ จึงต้องใช้อ็อบเจ็กต์ ที่แตกต่างเมื่อรันในสภาพแวดล้อม 64 บิต ในสภาพแวดล้อม 64 บิต ไลบรารีวิธีการอินพุตจะผนวก **__64** เข้ากับ ชื่อโดยอัตโนมัติเมื่อค้นหาวิธีการอินพุต ในตัวอย่างก่อนหน้านี้ ชื่อของวิธีการอินพุตจะเป็น **fr_CA.im__64**

สามารถตั้งชื่อวิธีการอินพุตโดยไม่ใช้ชื่อโลแคลก็ได้ เนื่องจากไลบรารี **libIM** ไม่ได้จำกัดว่าชื่อต้องเป็นชื่อโลแคล แอ็พพลิเคชันที่เรียกต้องแน่ใจว่าชื่อที่ส่งผ่านไปยัง **libIM** สามารถ หาพบได้ อย่างไรก็ตาม แอ็พพลิเคชันควรจะร้องขอเฉพาะสตริงตัวดัดแปลง ในรูปแบบ **@im=***ตัวดัดแปลง* เท่านั้น และคำร้องขอของผู้ใช้นั้น ต้องต่อกันกับสตริงที่ส่งคืนจากรูทีนย่อย **setlocale** (**LC_CTYPE,NULL**)

พื้นที่วิธีการอินพุต

วิธีการอินพุตแบบซับซ้อนต้องการการโต้ตอบกับผู้ใช้โดยตรง ตัวอย่างเช่น วิธีการอินพุตภาษาญี่ปุ่นอาจต้องการแสดงเมนูของสตริงตัวเลือก ตามข้อมูลการออกเสียงที่ตรงกับคีย์ที่คุณ ป้อน

ผลป้อนกลับของ key strokes จะปรากฏขึ้นในพื้นที่หนึ่งพื้นที่ขึ้นไปบน จอแสดงผล พื้นที่วิธีการอินพุตมีดังนี้:

สถานะ ข้อมูลที่เป็นข้อความและบิตแม็พสามารถปรากฏขึ้นในพื้นที่สถานะ พื้นที่สถานะ เป็นส่วนขยายของ light-emitting diodes (LEDs) บนคีย์บอร์ด

การแก้ไขก่อนหน้า
ข้อความระหว่างกลางจะปรากฏขึ้นในพื้นที่การแก้ไขก่อนหน้าสำหรับภาษาที่ เรียบเรียง ก่อนที่ไคลเอ็นต์จะจัดการกับข้อมูล

คุณลักษณะทั่วไปของ วิธีการอินพุตคือ คุณกดชุดของคีย์เพื่อแสดงถึงอักขระ ตัวเดียวหรือชุดของอักขระ กระบวนการเรียบเรียง อักขระจาก keystrokes นี้เรียกว่า *การแก้ไขก่อนหน้า*

เสริม เมนูและกล่องโต้ตอบที่ช่วยคุณกำหนดวิธีการอินพุตเอง จะปรากฏขึ้นในพื้นที่เสริม คุณอาจมีพื้นที่เสริมหลายพื้นที่ ซึ่งจัดการโดยวิธีการอินพุตและไม่เกี่ยวข้องกับไคลเอ็นต์

การจัดการ พื้นที่วิธีการอินพุตขึ้นอยู่กับส่วนของความรับผิดชอบที่แบ่งกัน ระหว่างแอ็พพลิเคชัน (หรือ toolkit) และวิธีการอินพุต ส่วนของ ความรับผิดชอบมีดังนี้:

- แอ็พพลิเคชันรับผิดชอบขนาดและตำแหน่งของพื้นที่วิธีการ อินพุต
- วิธีการอินพุตรับผิดชอบเนื้อหาของพื้นที่อินพุต พื้นที่วิธีการอินพุตไม่สามารถแนะนำการจัดวางได้

คำสั่งวิธีการอินพุต

วิธีการอินพุตคือชุดของรูทีนย่อยที่แปล key strokes เป็นสตริงอักขระในชุดโค้ดที่ระบุโดย โลแคล รูทีนย่อยวิธีการอินพุตมีตรรกะสำหรับการประมวลผลอินพุตเฉพาะโลแคล และตัวควบคุมบนคีย์บอร์ด (Ctrl, Alt, Shift, Lock, Alt Graphic)

คำสั่งต่อไปนี้ช่วยในการกำหนดวิธีการอินพุตซึ่งแม็พ สำหรับการใช้ของรูทีนย่อยวิธีการอินพุตเอง:

keycomp
คอมไพล์ไฟล์การแม็พคีย์บอร์ดเข้าในไฟล์คีย์แม็พวิธีการอินพุต

การเขียนโปรแกรมวิธีการอินพุต

วิธีการอินพุตเป็นอินเตอร์เฟสการเขียนโปรแกรมที่ทำให้แอ็พพลิเคชันสามารถรันในสภาวะแวดล้อมสากลที่มีให้ โดยใช้การสนับสนุน multicultural

วิธีการอินพุตมีลักษณะดังต่อไปนี้:

- การสนับสนุนอินพุตท้องถิ่น (กำหนดโดยโลแคล)
- การสนับสนุนหลายคีย์บอร์ด

- การประมวลผลอินพุตอักขระหลายไบต์

หมายเหตุ: อย่าสมมุติว่ามีการใช้คีย์บอร์ดทางกายภาพใดๆ อยู่ใช้วิธีการอินพุตตามการตั้งค่าโลแคลเพื่อจัดการคีย์บอร์ดอินพุต

การเริ่มต้น

ส่วนนี้จะอธิบายรูทีนย่อยที่ใช้เพื่อกำหนด วิธีการอินพุต

คุณสามารถใช้รูทีนย่อย IMQueryLanguage เพื่อกำหนดว่า วิธีการอินพุตพร้อมใช้งานโดยไม่ต้องเริ่มต้นหรือไม่ แอ็พพลิเคชัน (toolkit) จะเริ่มต้นวิธีการอินพุตเฉพาะโลแคลโดยการเรียกรูทีนย่อย IMInitialize ซึ่งจะเริ่มต้น ตัวแก้ไขวิธีการอินพุตเฉพาะโลแคล (IMED) รูทีนย่อยใช้ตัวแปร สภาพแวดล้อม **LOCPATH** เพื่อค้นหาวิธีการอินพุต ที่ระบุชื่อโดยตัวแปรสภาพแวดล้อม **LANG** ตัวแปรสภาพแวดล้อม **LOCPATH** ระบุชุดของชื่อไดเร็กทอรีที่ใช้เพื่อค้นหาวิธีการ อินพุต

ถ้าพบวิธีการอินพุต รูทีนย่อย **IMInitialize** จะใช้ รูทีนย่อย **load** เพื่อโหลดวิธีการอินพุตและแนบไฟล์ **imkeymap** เมื่อเข้าถึงวิธีการอินพุต จะมีการส่งคืนอ็อบเจ็กต์ ชนิด **IMFep** (ตัวประมวลผล front-end ของวิธีการอินพุต) **IMFep** ควรมีการจัดการเป็นโครงสร้าง opaque

แต่ละ **IMFep** จะสืบทอดชุดโค้ดของโลแคลเมื่อมีการเรียกรูทีนย่อย IMInitialize ในเวลาต่อมา สตริงที่ส่งคืนโดยรูทีนย่อย **IMFilter** และ **IMLookupString** จะอยู่ในชุดโค้ด ของโลแคล การเปลี่ยนโลแคลหลังจากการเรียกรูทีนย่อย **IMInitialize** ไม่ส่งผลกระทบต่อชุดโค้ดของ **IMFep**

สำหรับแต่ละ **IMFep** แอ็พพลิเคชันสามารถใช้รูทีนย่อย **IMCreate** เพื่อสร้างอินสแตนซ์ **IMObject** หนึ่งรายการขึ้นไป **IMObject** จัดการ คำสั่งของตนเองและสามารถจัดการพื้นที่วิธีการอินพุตได้หลายพื้นที่ (ให้ดู “พื้นที่วิธีการอินพุต” ในหน้า 143) วิธีการที่แต่ละ **IMObject** ใช้กำหนดการประมวลผลอินพุตขึ้นอยู่กับชุดโค้ดและคีย์บอร์ดที่เชื่อมโยงกับ โลแคล ในกรณีที่ง่ายที่สุด ต้องการ **IMObject** เพียงรายการเดียว ถ้าแอ็พพลิเคชันจัดการโต้ตอบกับผู้ใช้เพียงรายการเดียว วิธีการ อินพุตยังสนับสนุนอินเตอร์เฟสผู้ใช้อื่นๆ ด้วยถ้าแอ็พพลิเคชัน อนุญาตให้โต้ตอบกับผู้ใช้หลายรายการได้ และแต่ละการโต้ตอบต้องการหนึ่ง **IMObject**

ความแตกต่างระหว่าง **IMFep** และ **IMObject** คือ **IMFep** คือการจัดการที่จะผูกแอ็พพลิเคชันไว้กับโค้ด ของวิธีการอินพุต ในขณะที่ **IMObject** คือการจัดการที่แสดงถึง อินสแตนซ์ของคำสั่งของอุปกรณ์อินพุต เช่น คีย์บอร์ด **IMFep** ไม่ได้ แสดงถึงคำสั่งของวิธีการอินพุต แต่ละ **IMObject** มีการ เริ่มต้นเป็นคำสั่งอินพุตเฉพาะ และมีการเปลี่ยนแปลงไปตาม ลำดับของเหตุการณ์ที่ได้รับ

หลังจากที่สร้าง **IMObject** แอ็พพลิเคชันสามารถประมวลผล เหตุการณ์ของคีย์ได้ แอ็พพลิเคชันควรส่งผ่านเหตุการณ์ของคีย์ไปยัง **IMObject** โดยใช้ รูทีนย่อย **IMFilter** และ **IMLookupString** รูทีนย่อยเหล่านี้ใช้เพื่อแยกกระบวนการภายในของ IMED ออกจากกระบวนการแม็พเหตุการณ์ของคีย์ที่กำหนดเอง

การจัดการวิธีการอินพุต

ส่วนนี้อธิบายรูทีนย่อยที่ใช้เพื่อวัตถุประสงค์ การดูแลรักษา

วิธีการอินพุตนี้มีรูทีนย่อยต่อไปนี้เพื่อวัตถุประสงค์การ ดูแลรักษา:

วิธีการอินพุต	คำอธิบาย
IMInitialize	เริ่มต้นวิธีการอินพุตมาตรฐานสำหรับภาษาที่ระบุ ส่งคืนการจัดการไปยัง IMED ที่เชื่อมโยงกับโลแคล โดย handle เป็นโครงสร้างที่ไม่ชัดเจนของชนิด IMFep
.IMQueryLanguage	ตรวจสอบว่าภาษาที่ระบุได้รับการสนับสนุนหรือไม่
IMCreate	สร้างอินสแตนซ์ของวิธีการอินพุตเฉพาะขึ้นหนึ่งรายการ รูทีนย่อยนี้ต้อง มีการเรียกก่อนทำการประมวลผลเหตุการณ์คีย์ใดๆ
IMClose	ปิดวิธีการอินพุต
IMDestroy	ทำลายอินสแตนซ์ของวิธีการอินพุต

การจัดการคีย์แม็พวิธีการอินพุต

วิธีการอินพุตนำเสนอรูทีนย่อยหลายอย่างเพื่อแม็พเหตุการณ์คีย์ เข้ากับสตริง การแม็พมีการเก็บรักษาไว้ในไฟล์ **imkeymap** ซึ่ง อยู่ในไดเร็กทอรี **LOCPATH**

รูทีนย่อยที่ใช้สำหรับการแม็พมีดังนี้:

รูทีนย่อย	คำอธิบาย
IMInitializeKeymap	เริ่มต้น imkeymap ที่เชื่อมโยงกับภาษาที่ระบุ
IMFreeKeymap	ทำให้รีซอร์สที่จัดสรรโดยรูทีนย่อย **IMInitializeKeymap** ว่าง
IMAIXMapping	แปลคู่ของสัญลักษณ์คีย์และพารามิเตอร์คำสั่งเป็นสตริง และส่งคืนตัวชี้ไปยังสตริงนั้น
IMSimpleMapping	แปลคู่ของสัญลักษณ์คีย์และพารามิเตอร์คำสั่งเป็นสตริง และส่งคืนตัวชี้ไปยังสตริงนั้น

การประมวลผลเหตุการณ์คีย์

การประมวลผลอินพุตเริ่มต้นขึ้นเมื่อคุณกดคีย์บนคีย์บอร์ด

แอ็พพลิเคชันต้องสร้าง **IMObject** ก่อนการเรียก ฟังก์ชันเหล่านี้:

ตารางที่ 10. การประมวลผลเหตุการณ์คีย์

ฟังก์ชัน	คำอธิบาย
IMFilter	ขอให้ IMED บ่งชี้ว่าเหตุการณ์คีย์มีการใช้เป็นการภายในหรือไม่ ถ้า IMED กำลังเรียบเรียงสตริงท้องถิ่น IMED จะแม็พเหตุการณ์คีย์ กับสตริงนั้น
IMLookupString	แม็พเหตุการณ์คีย์กับสตริงท้องถิ่น
IMProcessAuxiliary	แจ้งวิธีการอินพุตของอินพุตสำหรับพื้นที่เสริม
IMIoctl	ทำการควบคุมหรือเคียวรี่หลายอย่างเกี่ยวกับ วิธีการอินพุต

Callbacks

IMED สื่อสารกับผู้ใช้โดยตรงโดยใช้ Input Method-Callback (IM-CB) API เพื่อเข้าถึงฟังก์ชันที่พึ่งพากราฟิก (**callbacks**) ที่มีในแอ็พพลิเคชัน แอ็พพลิเคชัน จะแนบ callbacks ซึ่งดำเนินการฟังก์ชันเอาต์พุตและข้อมูลเคียวรี่ กับ **IMObject** ในระหว่างการเริ่มต้น แอ็พพลิเคชันยังคง จัดการอินพุตทั้งหมด

ผู้เรียกต้องระบุชุดของฟังก์ชัน callback ที่ IMED จะใช้เพื่อสื่อสาร กับผู้ใช้ ให้ดู “การใช้ callbacks” ในหน้า 148 สำหรับคำอธิบายของรูทีนย่อย ที่กำหนดโดย IM-CB API

โครงสร้างวิธีการอินพุต

ส่วนนี้อธิบายโครงสร้างวิธีการอินพุตที่สำคัญ

โครงสร้างหลักที่ใช้โดยวิธีการอินพุตมีดังนี้:

โครงสร้าง	คำอธิบาย
IMFepRec	มีข้อมูล front end
IMObjectRec	มีส่วนทั่วไปของอ็อบเจ็กต์วิธีการอินพุต
IMCallback	ลงทะเบียนรูทีนย่อย callback ใน **IMFep**
IMTextInfo	มีข้อมูลเกี่ยวกับพื้นที่ข้อความ ส่วนใหญ่เป็นสตริง การแก้ไขก่อนหน้า
IMAuxInfo	กำหนดเนื้อหาของพื้นที่เสริมและชนิดของ การประมวลผลที่ร้องขอ
IMIndicatorInfo	บ่งชี้ค่าปัจจุบันของตัวบ่งชี้
IMSTR	กำหนดสตริงที่ไม่ได้ยุติด้วย null
IMSTRATT	กำหนดสตริงที่ไม่ได้ยุติด้วย null และ แอ็ตทริบิวต์

การทำงานกับการแม็พคีย์บอร์ด

โมเดลต่อไปนี้แสดงวิธีการใช้วิธีการอินพุตโดย แอ็พพลิเคชัน ใช้ข้อมูลนี้เพื่อช่วยคุณกำหนดการแม็พ คีย์บอร์ดเอง

การประมวลผลอินพุตแบ่งออกเป็นสามขั้นตอนดังนี้:

1. **keycode/keystate(ดิบ) - > keysym/ตัวดัดแปลง(ใหม่)**
   ขั้นตอนนี้ขึ้นอยู่กับแอ็พพลิเคชันและสภาพแวดล้อม แอ็พพลิเคชัน รับผิดชอบการแม็พเหตุการณ์คีย์ดิบเข้าใน keysym/ตัวดัดแปลงสำหรับ อินพุตของวิธีการอินพุต
   ในสภาพแวดล้อม AIXwindows ไคลเอ็นต์จะใช้ตาราง keysym ของเซิร์ฟเวอร์, **xmodmap**, ซึ่งมีการติดตั้ง ที่เซิร์ฟเวอร์ เพื่อทำขั้นตอนนี้ **xmodmap** กำหนด การแม็พของคีย์ Shift, Lock, และ Alt-Graphic ไคลเอ็นต์จะใช้ **xmodmap** และตัวดัดแปลง Shift และ Lock จากเหตุการณ์ X เพื่อกำหนด keysym/ตัวดัดแปลงที่แสดงถึงโดยเหตุการณ์นี้
   ตัว อย่างเช่น ถ้าคุณกด **XK_a** keysym พร้อมกับตัวดัดแปลง Shift **xmodmap** จะแม็พตัวดัดแปลงนั้นกับ **XK_A** keysym เนื่องจากคุณ ใช้คีย์ Shift เพื่อแม็พรหัสคีย์กับ keysym แอ็พพลิเคชันจึง ควรจะ mask ตัวดัดแปลง Shift จากเหตุการณ์ X ดั้งเดิม ในเวลาต่อมา อินพุตของวิธีการอินพุตจะเป็น **XK_A** keysym และ ไม่มีตัวดัดแปลง
   ในสภาพแวดล้อมอื่น ถ้าอุปกรณ์ไม่มี ข้อมูลเพิ่มเติม วิธีการอินพุตจะได้รับ **XK_a** keysym พร้อมกับตัวดัดแปลง Shift วิธีการอินพุตควรทำการแม็พแบบเดียวกัน ในตัวพิมพ์ทั้งสองแบบและส่งคืนตัวอักษร A
2. **keysym/ตัวดัดแปลง(ใหม่) - > สตริงท้องถิ่น**
   ขั้นตอนนี้ขึ้นอยู่กับ IMED ท้องถิ่นและแตกต่างกันไปในแต่ละโลแคล ขั้นตอนจะแจ้ง ให้ IMED ทราบว่ามีเหตุการณ์คีย์เกิดขึ้นและขอการบ่งชี้ว่า IMED จะใช้เหตุการณ์คีย์เป็นการภายใน กรณีนี้เกิดขึ้นเมื่อแอ็พพลิเคชัน เรียกรูทีนย่อย **IMFilter**
   ถ้า IMED บ่งชี้ว่า เหตุการณ์คีย์ใช้สำหรับการประมวลผลภายใน แอ็พพลิเคชันจะ ละเว้นเหตุการณ์ เนื่องจาก IMED เป็นส่วนแรกที่เห็นเหตุการณ์ ขั้นตอนนี้จึงควรทำก่อนที่แอ็พพลิเคชันจะตีความเหตุการณ์ IMED จะใช้เฉพาะเหตุการณ์คีย์ที่จำเป็นเท่านั้น
   ถ้า IMED บ่งชี้ว่า เหตุการณ์คีย์ไม่ได้ใช้สำหรับการประมวลผลภายใน แอ็พพลิเคชันจะ ทำขั้นตอนถัดไป

3. **keysym/ตัวดัดแปลง(ใหม่) – > สตริงที่กำหนดเอง**

   ขั้นตอนนี้ เกิดขึ้นเมื่อแอ็พพลิเคชันเรียกรูทีนย่อย **IMLookupString** คีย์แม็พวิธีการอินพุต (ที่สร้างโดยคำสั่ง **keycomp**) จะกำหนด การแม็พสำหรับระยะนี้ นับเป็นความพยายามครั้งสุดท้ายในการแม็พ เหตุการณ์คีย์กับสตริง และอนุญาตให้ผู้ใช้กำหนดการแม็พเองได้

   ถ้ามีการกำหนด ชุด keysym/ตัวดัดแปลง (ใหม่) ในคีย์แม็พวิธีการอินพุต (imkeymap) ระบบจะส่งคืนสตริง มิฉะนั้น วิธีการอินพุตจะไม่รู้จัก เหตุการณ์คีย์

คีย์แม็พวิธีการอินพุต

วิธีการอินพุตนำเสนอการสนับสนุน imkeymaps ที่ผู้ใช้กำหนด ซึ่งช่วยให้คุณสามารถกำหนดการแม็พวิธีการอินพุตเองได้ วิธีการอินพุต สนับสนุน imkeymaps สำหรับแต่ละโลแคล ชื่อไฟล์สำหรับ imkeymaps คล้าย กับชื่อไฟล์ของวิธีการอินพุต ยกเว้นว่าคำเติมท้ายสำหรับไฟล์ imkeymap คือ **.imkeymap** แทน **.im**

ตัวอย่างนี้ใช้วิธีการอินพุตภาษาอิตาเลียนเพื่อสาธิตวิธีการกำหนดไฟล์ **imkeymap** ของคุณเอง:

1. คัดลอกไฟล์ต้นฉบับ **imkeymap** ดีฟอลต์ไปยังไดเร็กทอรี **$HOME** ของคุณโดยการพิมพ์ดังนี้:

   ```
   cd $HOME
   cp /usr/lib/nls/loc/it_IT.ISO8859-1.imkeymap.src
   ```

2. แก้ไขไฟล์ต้นฉบับ **imkeymap** ตามรูปแบบไฟล์ดีฟอลต์โดย การพิมพ์ดังนี้:

   ```
   vi it_IT.ISO8859-1.imkeymap.src
   ```

3. คอมไพล์ไฟล์ต้นฉบับ **imkeymap** โดยการพิมพ์ดังนี้:

   ```
   keycomp < it_IT.ISO8859-1.imkeymap.src > it_IT.ISO8859-1.imkeymap
   ```

4. ตรวจสอบให้แน่ใจว่าตัวแปรสภาพแวดล้อม **LOCPATH** ระบุ **$HOME** ก่อนหน้า **/usr/lib/nls/loc** โดย การพิมพ์ดังนี้:

   ```
   LOCPATH=$HOME:$LOCPATH
   ```

   หมายเหตุ: โปรแกรม **setuid** และ **setgid** ทั้งหมด ละเว้นตัวแปรสภาพแวดล้อม **LOCPATH**

การแม็พขาเข้าและขาออก

Imkeymaps แม็พสัญลักษณ์คีย์เข้ากับสตริงชุดโค้ดไฟล์ Imkeymaps ท้องถิ่นที่พบในไลบรารี **/usr/lib/nls/loc** ถูกกำหนดให้รวมการแม็พสำหรับคีย์ขาเข้าทั้งหมด

Imkeymaps มีชนิดการแม็พดังต่อไปนี้:

ชนิดการแม็พ	คำอธิบาย
การแม็พขาเข้า	การแม็พของ keysym หรือโมดิฟายเออร์ที่สร้าง สตริงปลายทางที่เข้ารหัสในชุดโค้ดของโลแคล
การแม็พขาออก	การแม็พของ keysym หรือโมดิฟายเออร์ที่ไม่สร้าง สตริงปลายทางที่รวมในชุดโค้ดของโลแคล

Imkeymap พิเศษ **/usr/lib/nls/loc/C@outbound.imkeymap** กำหนดการแม็พขาออกสำหรับคีย์บอร์ดทั้งหมดที่จัดทำโดยผู้ผลิตนี้ และมีไว้สำหรับให้ aixterm ใช้งานเป็นหลัก Imkeymap นี้มีการแม็พ คีย์ PF, ปุ่มเคอร์เซอร์ และคีย์พิเศษอื่นๆ ที่แอ็พพลิเคชันใช้กัน โดยทั่วไป แอ็พพลิเคชันที่ทำให้เป็นโกลบอลที่ใช้อินพุตมาตรฐานและ เอาต์พุตมาตรฐานควรจำกัดการขึ้นต่อกันของตนในการแม็พขาออก ซึ่งไม่ขึ้นกับคีย์บอร์ดที่แตกต่างกัน ตัวอย่างเช่น Alt-a มีการกำหนดในวิธีเดียวกันบนคีย์บอร์ดทั้งหมดที่จัดทำโดยผู้ผลิต รายนี้ แต่ Alt-tilde แตกต่างกันไป ขึ้นอยู่กับคีย์บอร์ดที่ใช้

Aixterm ดำเนินการแม็พขาออกบน **C@outbound** imkeymap แอ็พพลิเคชันที่ต้องการการแม็พเพิ่มเติมควรดัดแปลงต้นฉบับ imkeymap ท้องถิ่นเพื่อรวมคำนิยามที่จำเป็น

การใช้ callbacks

แอ็พพลิเคชันที่ใช้วิธีการอินพุตควรมีฟังก์ชัน callback เพื่อให้ตัวแก้ไขวิธีการอินพุต (IMED) สามารถสื่อสารกับ ผู้ใช้ได้ ชนิดของวิธีการอินพุตที่คุณใช้เป็นตัวกำหนดว่า callbacks เป็นสิ่งที่จำเป็นหรือไม่ ตัวอย่างเช่น วิธีการอินพุตไบต์เดียวไม่ต้องการ callbacks แต่วิธีการอินพุตภาษาญี่ปุ่นใช้ callbacks อย่างมากใน ฟังก์ชันการแก้ไขก่อนหน้า ฟังก์ชันการแก้ไขก่อนหน้าช่วยให้สามารถประมวลผลอักขระได้ก่อนที่ อักขระนั้นจะถูก committed ในแอ็พพลิเคชัน

เมื่อคุณใช้วิธีการอินพุต เฉพาะแอ็พพลิเคชันเท่านั้นที่สามารถแทรกหรือลบ ข้อมูลการแก้ไขก่อนหน้าและเลื่อนข้อความได้ ในเวลาต่อมา เสียงของ keystrokes มาถึงแอ็พพลิเคชันตามการร้องขอของตรรกะ วิธีการอินพุตผ่านทาง callbacks

เมื่อคุณป้อน keystroke แอ็พพลิเคชันจะเรียกรูทีนย่อย **IMFilter** ก่อนการส่งคืน วิธีการอินพุตสามารถเรียกฟังก์ชัน echoing callback เพื่อแทรก keystrokes ใหม่ หลังจากเรียบเรียงอักขระแล้ว รูทีนย่อย **IMFilter** จะส่งคืนฟังก์ชันดังกล่าว และลบ keystrokes

ในหลายกรณี ตรรกะวิธีการอินพุตต้อง call back ไคลเอ็นต์ แต่ละกรณีเหล่านี้ถูกกำหนดโดยการดำเนินการ callback ไคลเอ็นต์ระบุ callback ที่ควรจะเรียกสำหรับแต่ละการดำเนินการ

ชนิดของ callbacks มีการอธิบายดังนี้:

- ภาพวาดข้อความ

  IMED จะใช้ callbacks ข้อความเพื่อวาดข้อความการแก้ไขก่อนหน้า ใดๆ ที่มีการเรียบเรียงอยู่ในปัจจุบัน เมื่อต้องการ callbacks แอ็พพลิเคชันและ IMED จะแบ่งใช้บัฟเฟอร์บรรทัดเดียวที่ซึ่งทำ การแก้ไข IMED ยังแสดงข้อมูลเคอร์เซอร์ซึ่ง callbacks จะนำเสนอให้แก่ผู้ใช้ในเวลาต่อมา

  Callbacks ข้อความมีดังนี้:

รูทีนย่อย Callback	คำอธิบาย
IMTextDraw	ขอให้แอ็พพลิเคชันโปรแกรมวาดสตริงข้อความ
IMTextHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนพื้นที่ข้อความ
IMTextStart	แจ้งแอ็พพลิเคชันโปรแกรมให้ทราบถึงความยาวของพื้นที่ว่าง การแก้ไขก่อนหน้า
IMTextCursor	ขอให้แอ็พพลิเคชันโปรแกรมย้ายเคอร์เซอร์ข้อความ

- ตัวบ่งชี้ (สถานะ)

  IMED ใช้ callbacks ตัวบ่งชี้เพื่อร้องขอ สถานะภายใน รูทีนย่อย **IMIoctl** ทำงานร่วมกับคำสั่ง **IMQueryIndicatorString** เพื่อ ดึงข้อมูลสตริงข้อความที่แสดงสถานะภายใน Callbacks ตัวบ่งชี้คล้ายกับ callbacks ข้อความ ยกเว้นว่า callbacks ตัวบ่งชี้จะใช้ค่าสถานะ แทนการแบ่งใช้บัฟเฟอร์บรรทัดเดียว

  Callbacks ตัวบ่งชี้มีดังนี้:

รูทีนย่อย Callback	คำอธิบาย
IMIndicatorDraw	บอกให้แอ็พพลิเคชันโปรแกรมวาดตัวบ่งชี้สถานะ
IMIndicatorHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนตัวบ่งชี้สถานะ
IMBeep	บอกให้แอ็พพลิเคชันโปรแกรมส่งเสียงบีป

- เสริม

  IMED ใช้ callbacks เสริมเพื่อร้องขอการโต้ตอบที่ซับซ้อน กับผู้ใช้ ในเวลาต่อมา callbacks เหล่านี้มีความซับซ้อนมากกว่า callbacks ข้อความหรือตัวบ่งชี้

  Callbacks เสริมมีดังนี้:

รูทีนย่อย Callback	คำอธิบาย
IMAuxCreate	บอกให้แอ็พพลิเคชันโปรแกรมสร้างพื้นที่เสริม
IMAuxDraw	บอกให้แอ็พพลิเคชันโปรแกรมวาดพื้นที่เสริม
IMAuxHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนพื้นที่เสริม
IMAuxDestroy	บอกให้แอ็พพลิเคชันโปรแกรมทำลายพื้นที่เสริม

โครงสร้าง **IMAuxInfo** กำหนดการโต้ตอบที่ IMED ต้องการ
เนื้อหาของพื้นที่เสริมมีการกำหนดโดยโครงสร้าง **IMAuxInfo** ซึ่งอยู่ในไลบรารี **/usr/include/im.h**

เนื้อหารูทีนย่อย	คำอธิบาย
IMTitle	กำหนดชื่อของพื้นที่เสริม นี่ เป็นสตริงหลายไบต์ ถ้า `title.len` เป็น 0 จะไม่มีการแสดง ชื่อ
IMMessage	กำหนดรายการของข้อความที่จะแสดง จากมุมมองแอ็พพลิเคชัน โครงสร้าง **IMMessage** ควรถูกจัดการ เป็นข้อความเอาต์พุตอย่างเดียวที่ให้ข้อมูลเท่านั้น อย่างไรก็ตาม บาง วิธีการอินพุตใช้โครงสร้าง **IMMessage** เพื่อสร้างการโต้ตอบ กับผู้ใช้ซึ่งเหตุการณ์คีย์ที่ได้รับโดยวิธีรูทีนย่อย **IMFilter** หรือ **IMLookupString** มีการจัดการเป็นอินพุต ของวิธีการอินพุต ในกรณีเช่นนั้น วิธีการอินพุต อาจจัดการโครงสร้าง **IMMessage** เป็นรายการที่เลือกได้หรือพื้นที่พร้อมต์ อย่างใดอย่างหนึ่ง ในกรณีใดกรณีหนึ่ง แอ็พพลิเคชันจะแสดงเฉพาะ เนื้อหาข้อความ คุณไม่จำเป็นต้องเรียกรูทีนย่อย **IMProcessAuxiliary** ถ้า โครงสร้าง **IMSelection** ไม่มีโครงสร้าง **IMPanel** และฟิลด์ **IMButton** เป็น null `message.nline` บ่งชี้ จำนวนของข้อความที่มีอยู่ในโครงสร้าง **IMMessage** แต่ละข้อความถูกสมมุติว่าเป็นบรรทัดเดียว อักขระควบคุม เช่น \t, ไม่เป็นที่รู้จัก ข้อความของแต่ละข้อความมีการกำหนดโดยโครงสร้าง **IMSTRATT** ซึ่งประกอบด้วยทั้งสตริงหลายไบต์ และสตริงแอ็ตทริบิวต์ แต่ละแอ็ตทริบิวต์จะถูกแม็พแบบหนึ่งต่อหนึ่ง สำหรับแต่ละไบต์ในสตริงข้อความ ถ้า `message.cursor` เป็น True โครงสร้าง **IMMessage** จะกำหนดเคอร์เซอร์ข้อความที่ตำแหน่ง `message.cur_row`, `message.cur_col` ฟิลด์ `message.cur_col` มีการกำหนดในหน่วย ไบต์ ฟิลด์ `message.maxwidth` มีความกว้างสูงสุด ของข้อความทั้งหมดที่กำหนดในหน่วยคอลัมน์

เนื้อหารูทีนย่อย	คำอธิบาย
IMButton	บ่งชี้ปุ่มที่เป็นไปได้ซึ่งอาจแสดง ให้แก่ผู้ใช้ ฟิลด์ IMButton บอกแอ็พพลิเคชันให้ทราบว่า ควรนำเสนอ ตัวควบคุมอินเตอร์เฟสผู้ใช้ใดให้แก่ผู้ใช้ขั้นปลาย ปุ่ม เป็นชนิด int และอาจมี mask ดังต่อไปนี้: **IM_OK** แสดงปุ่ม OK **IM_CANCEL** แสดงปุ่ม CANCEL **IM_ENTER** แสดงปุ่ม ENTER **IM_RETRY** แสดงปุ่ม RETRY **IM_ABORT** แสดงปุ่ม ABORT **IM_YES** แสดงปุ่ม YES **IM_NO** แสดงปุ่ม NO **IM_HELP** แสดงปุ่ม HELP **IM_PREV** แสดงปุ่ม PREV **IM_NEXT** แสดงปุ่ม NEXT แอ็พพลิเคชันควรใช้รูทีนย่อย **IMProcessAuxiliary** เพื่อสื่อสารการเลือกปุ่ม
IMSelection	กำหนดรายการของไอเท็ม เช่น ตัวหนังสือทางภูมิศาสตร์ ที่ผู้ใช้ขั้นปลายสามารถเลือกได้ โครงสร้างนี้ใช้เมื่อวิธีการอินพุต ต้องการแสดงไอเท็มจำนวนมาก แต่ไม่ต้องการ ควบคุมว่าจะแสดงรายการให้แก่ผู้ใช้อย่างไร โครงสร้าง **IMSelection** มีการกำหนด เป็นรายการของโครงสร้าง **IMPanel** บางแอ็พพลิเคชันไม่ สนับสนุนโครงสร้าง **IMSelection** ภายในโครงสร้าง **IMAuxInfo** แอ็พพลิเคชันที่สนับสนุนโครงสร้าง **IMSelection** ควรดำเนินงาน **IM_SupportSelection** โดยใช้รูทีนย่อย **IMIoctl** ทันทีหลังจากการสร้าง **IMObject** นอกจากนี้ บางแอ็พพลิเคชันไม่สนับสนุนหลายโครงสร้าง **IMPanel** ดังนั้น ฟิลด์ `panel_row` และ `panel_col` จึงถูกจำกัดให้มีการตั้งค่าเป็น 1 โดยอินพุตวิธีการทั้งหมด แต่ละโครงสร้าง **IMPanel** ประกอบด้วยรายการของฟิลด์ `IMItem` ที่ควรถูกจัดการ เป็นรายการแถว/คอลัมน์แบบสองมิติ ซึ่งมีขนาดตามที่กำหนดเป็น `item_row` คูณ `item_col` ถ้า `item_col` เป็น 1 จะมีเพียงหนึ่งคอลัมน์เท่านั้น ขนาด ของโครงสร้าง **IMPanel** มีการกำหนดในหน่วยไบต์ แต่ละไอเท็ม ภายในโครงสร้าง **IMPanel** ยาวน้อยกว่าหรือเท่ากับ `panel->maxwidth` แอ็พพลิเคชัน ควรใช้รูทีนย่อย **IMProcessAuxiliary** เพื่อสื่อสาร การเลือกของผู้ใช้หนึ่งรายการขึ้นไป ค่า **IM_SELECTED** บ่งชี้ว่า ไอเท็มใดถูกเลือก ค่า **IM_CANCEL** บ่งชี้ว่า ผู้ใช้ต้องการยุติการโต้ตอบเสริม
คำใบ้	ใช้โดยวิธีการอินพุตเพื่อแสดงข้อมูล เกี่ยวกับบริบทของโครงสร้าง **IMAuxInfo** ค่าของ `IM_AtTheEvent` บ่งชี้ว่าโครงสร้าง **IMAuxInfo** เชื่อมโยงกับเหตุการณ์ล่าสุดที่ส่งผ่าน ไปยังวิธีการอินพุตโดยใช้รูทีนย่อย **IMFilter** หรือ **IMLookupString** คำใบ้อื่นๆ ใช้เพื่อแยกแยะความแตกต่างเมื่อโครงสร้าง **IMAuxInfo** มีการแสดงขึ้นหลายโครงสร้าง

เนื้อหารูทีนย่อย	คำอธิบาย
สถานะ	ใช้โดยวิธีการอินพุตสำหรับการประมวลผลภายใน แอ็พพลิเคชันไม่ควรใช้ฟิลด์นี้ แต่ละโครงสร้าง **IMAuxInfo** มีความเป็นอิสระจากกัน วิธีที่ใช้ในการแสดงสมาชิก มีการกำหนดโดยผู้เรียกของวิธีการอินพุต โครงสร้าง **IMAuxInfo** ใช้โดย **IMAuxDraw** callback

หลักการที่เกี่ยวข้อง:

"การแก้ไขก่อนหน้าสำหรับ Kanji" ในหน้า 159

เมื่อใช้งานในโหมดการแปลง Romaji-เป็น-Kana คุณต้อง ทำสองขั้นตอนเพื่อจัดทำอักขระ Kanji อันดับแรก ผู้ใช้ป้อนอักขระฮิรากานะโดยการพิมพ์อักขระการออกเสียงแบบโรมันจิ ใน ขั้นตอนนี้ คุณจัดทำอักขระ Hiragana โดยการพิมพ์คีย์ตัวอักษร Romaji 1 ถึง 3 ซึ่งประกอบขึ้นเป็นเสียงของอักขระ Hiragana ขั้นตอนที่สอง แปลงอักขระ Hiragana เป็นอักขระ Kanji โดยการกดคีย์ Henkan อักขระ Kanji จำนวนมากมีการออกเสียงเหมือน กัน คีย์ Henkan จะแสดงตัวเลือก Kanji ที่ใกล้เคียงที่สุด การกดคีย์ Henkan ซ้ำๆ จะแสดงตัวเลือกเพิ่มเติมทั้งหมด

การเริ่มต้น callbacks

ต้องระบุ callbacks ทั้งหมดเมื่อคุณเรียกรูทีนย่อย **IMCreate** โครงสร้าง **IMCallback** มีที่อยู่สำหรับฟังก์ชัน callback แต่ละรายการ ผู้เรียกรูทีนย่อย **IMCreate** ต้องเริ่มต้นโครงสร้าง **IMCallback** ที่มีที่อยู่ต่างๆ

ฟังก์ชัน callback สามารถเรียกได้ก่อนที่รูทีนย่อย **IMCreate** จะส่งคืน การควบคุมไปยังผู้เรียก โดยปกติแล้ว **IMTextStart** callback มีการเรียกเพื่อระบุขนาดของบัฟเฟอร์การแก้ไขก่อนหน้า

วิธีการอินพุตแบบสองทิศทาง

วิธีการอินพุตแบบสองทิศทาง (BIM) คล้ายกับ single-byte input method ยกเว้นว่าจะถูกกำหนดให้ประมวลผล คีย์บอร์ดภาษาอาหรับ ฮีบรู และอูรดู BIM ยังเชื่อมโยงภาษาฮิบรูและ อารบิกกับภาษาลาตินด้วย ผู้ใช้สามารถใช้คีย์ Alt+Right Shift เพื่อ toggle ระหว่างชั้นภาษาอารบิก/ฮิบรู/เออร์ดูและชั้นภาษาลาติน ได้ การใช้คีย์เหล่านี้ได้มาจาก BIM

คุณลักษณะของ BIM มีดังนี้:

- สนับสนุนภาษาอารบิก ฮิบรู และลาติน
- สนับสนุนชุดโค้ด ISO8859-6, ISO8859-8, IBM-1046, IBM-856, และ Unicode UTF-8
- ดำเนินการเขียนการออกเสียง

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต สองทิศทาง (BIM)

คีย์แม็พที่ได้รับการสนับสนุนใน BIM มีดังนี้:

- ar_AA.ISO8859-6.imkeymap
- ar_AA@alt.ISO8859-6.imkeymap
- Ar_AA.IBM-1046.imkeymap
- Ar_AA@alt.IBM-1046.imkeymap
- iw_IL.ISO8859-8.imkeymap
- iw_IL@alt.ISO8859-8.imkeymap

- Iw_IL.IBM-856.imkeymap
- Iw_IL@alt.IBM-856.imkeymap

การตั้งค่าคีย์

ส่วนนี้อธิบายการตั้งค่าคีย์ที่ได้รับการสนับสนุนในวิธีการอินพุตสองทิศทาง (BIM)

การตั้งค่าคีย์ที่ได้รับการสนับสนุนใน BIM มีดังนี้:

การตั้งค่าคีย์	คำอธิบาย
scr-rev()	ย้อนกลับการจัดทิศทางบนจอภาพและตั้งค่าชั้นคีย์บอร์ด เป็นภาษาดีฟอลต์ของการจัดทิศทางใหม่
ltr-lang()	เปิดใช้งานชั้นคีย์บอร์ดภาษาลาติน
rtl-lang()	เปิดใช้งานชั้นคีย์บอร์ดภาษาอารบิก/ฮิบรู
col-mod()	เปิดใช้งานการปรับหัวข้อคอลัมน์ ซึ่งจะจัดการกับคำแต่ละคำ เป็นคอลัมน์ที่แยกต่างหากหนึ่งคอลัมน์
auto-push()	Toggles โหมด Autopush ซึ่งจะจัดการข้อความจากซ้ายไปขวา และข้อความจากขวาไปซ้ายผสมกัน เมื่อคุณเปิดใช้งานโหมด Autopush เซกเมนต์ที่ย้อนกลับ จะถูกเริ่มต้นโดยอัตโนมัติและยุติลงตาม อักขระที่ป้อนหรือชั้นภาษาที่เลือก ดังนั้น คุณจึงไม่ต้อง เรียกใช้ฟังก์ชัน Push ด้วยตนเอง
chg-push()	Toggles โหมด Push โหมดนี้ทำให้เคอร์เซอร์ยังคงอยู่ใน ตำแหน่งของเคอร์เซอร์และผลักอักขระที่พิมพ์ไปในทิศทางตรงกันข้าม กับทิศทางของฟิลด์
shp-in()	จัดรูปทรงอักขระอารบิกในรูปแบบแรกเริ่ม
shp-is()	จัดรูปทรงอักขระอารบิกในรูปแบบแยกต่างหาก
shp-p()	จัดรูปทรงอักขระอารบิกในรูปแบบ passthru
shp-asd()	จัดรูปทรงอักขระอารบิกในรูปแบบอัตโนมัติ
shp-m()	จัดรูปทรงอักขระอารบิกในรูปแบบกึ่งกลาง
shp-f()	จัดรูปทรงอักขระอารบิกในรูปแบบสุดท้าย

ตัวดัดแปลง

ส่วนนี้อธิบายตัวดัดแปลงที่ได้รับการสนับสนุนในวิธีการอินพุตสองทิศทาง (BIM)

ตัวดัดแปลงที่ได้รับการสนับสนุนใน BIM มีดังนี้:

โมดิฟายเออร์	คำอธิบาย
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

วิธีการอินพุตภาษา Cyrillic (CIM)

วิธีการอินพุตซิริลลิก (CIM) คล้ายกับ single-byte input method ยกเว้นว่าจะถูกกำหนดค่าเองสำหรับการประมวลผลคีย์บอร์ด ภาษาซิริลลิก

คุณลักษณะของ CIM มีดังนี้:

- สนับสนุนคำสั่ง Cyrillic และลาติน คุณสามารถสลับระหว่างสองคำสั่งได้โดยการกดปุ่ม Alt และปุ่ม Left หรือ Right Shift พร้อมกัน

  หมายเหตุ: ปุ่ม Alt-Graphic (Right Alt) สามารถใช้เพื่อ สร้างอักขระเพิ่มเติมภายในชั้นคีย์บอร์ดแต่ละชั้นได้
- สำหรับโลแคลรัสเซียนและบัลกาเรียน มีการสนับสนุนทั้งไดร์เวอร์คีย์บอร์ด 101-key และ 102-key
- สนับสนุนชุดโค้ด ISO8859-5

คีย์แม็พ

ส่วนนี้อธิบาย keymaps ที่สนับสนุนในวิธีการอินพุตภาษา ซิริลลิก (CIM)

คีย์แม็พที่ได้รับการสนับสนุนใน CIM มีดังนี้:

- bg_BG.ISO8859-5.imkeymap
- mk_MK.ISO8859-5.imkeymap
- sr_SP.ISO8859-5.imkeymap
- ru_RU.ISO8859-5.imkeymap
- be-BY.ISO8859-5.imkeymap
- uk-UA.ISO8859-5.imkeymap

Keysyms

CIM ใช้ keysyms ในกลุ่ม **XK_CYRILLIC**, **XK_LATIN1**, และ **XK_MISCELLANY**

Keysyms ที่สงวนไว้ต่อไปนี้ใช้เฉพาะกับวิธีการอินพุตของ ระบบนี้เท่านั้น:

Keysym	การแทนค่าเลขฐานสิบหก
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdiaeresis	0x180007ae

ตัวดัดแปลง

ส่วนนี้อธิบายตัวดัดแปลงที่ได้รับการสนับสนุนในวิธีการอินพุตภาษา Cyrillic (CIM)

ตัวดัดแปลงที่ได้รับการสนับสนุนใน CIM มีดังนี้:

โมดิฟายเออร์	การแทนค่าเลขฐานสิบหก
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

ตัวดัดแปลงภายในที่ได้รับการสนับสนุนใน CIM มีดังต่อไปนี้:

โมดิฟายเออร์	การแทนค่าเลขฐานสิบหก
Cyrillic Layer	0x20

วิธีอินพุตภาษากรีก (GIM)

วิธีการอินพุตภาษากรีก (GIM) คล้ายกับ single-byte input method (SIM) แต่จัดการทั้งชุดอักขระภาษาละติน และกรีก โดยการระบุสองเลเยอร์หรือสภาวะของการแม็พคีย์บอร์ด ซึ่งสอดคล้อง กับสองชุดอักขระ

โดยแรกเริ่ม คีย์บอร์ดมีคำสั่งอินพุตภาษาลาติน อย่างไรก็ตาม ถ้ากดปุ่ม left-shift ในขณะที่กดปุ่ม left-alt ค้างไว้ คีย์บอร์ดจะเปลี่ยนไปใช้คำสั่งอินพุตภาษากรีก คีย์บอร์ดอาจ กลับไปใช้คำสั่งภาษาลาตินได้โดยการกดปุ่ม right-shift ในขณะที่ กดปุ่ม left-alt ค้างไว้ นี่เป็นปุ่มเปลี่ยนการล็อค เนื่องจาก คำสั่งจะถูกล็อคเมื่อกดปุ่มดังกล่าว

ในขณะที่ใช้คำสั่งภาษากรีก วิธีการอินพุตจะรับรู้อักขระการออกเสียงต่อไปนี้ และอักขระตามมาที่ถูกต้องสำหรับการเขียนการออกเสียงดังแสดงในตารางต่อไปนี้:

ตารางที่ 11. อักขระการเขียนภาษากรีก

Keysym	อักขระการเขียนที่ถูกต้อง
dead_acute	ตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก: alpha, epsilon, eta, iota, omicron, upsilon, omega
dead_diaeresis	ตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก: iota, upsilon
dead_accentdiaeresis	ตัวพิมพ์เล็กอย่างเดียว: iota, upsilon

ในคำสั่งภาษาลาติน ไม่มีการเขียนการออกเสียง และปุ่มที่แสดง ในตารางข้างบนจะได้รับการจัดการเป็นอักขระกราฟิกธรรมดา

วิธีการอินพุตภาษากรีกและวิธีการอินพุตไบต์เดียวยังแตกต่างกันในการจัดการ กับลำดับการเขียนการออกเสียงที่ไม่ถูกต้องด้วย ในกรณีนั้น GIM จะส่งเสียงบีปและไม่ส่งคืนอักขระใดๆ SIM ไม่ส่งเสียงบีปและส่งคืนทั้ง อักขระการออกเสียงและอักขระกราฟิกที่เชื่อมโยงกับ ปุ่มที่ไม่ถูกต้อง

หมายเหตุ: ปุ่ม Alt-Graphic (right-alt) สามารถใช้เพื่อ สร้างอักขระเพิ่มเติมภายในคำสั่งคีย์บอร์ดแต่ละภาษา

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษากรีก (GIM)

คีย์แม็พที่ได้รับการสนับสนุนใน GIM มีดังนี้:

- el_GR.ISO8859-7.imkeymap

Keysyms

GIM ใช้ keysyms ในกลุ่ม **XK_LATIN1**, **XK_GREEK**, และ **XK_MISCELLANY**

Keysyms ที่สงวนไว้ต่อไปนี้ใช้เฉพาะกับวิธีการอินพุตของ ระบบนี้เท่านั้น

Keysym	การแทนค่าเลขฐานสิบหก
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdiaeresis	0x180007ae

ตัวดัดแปลง

ส่วนนี้อธิบายโมดิฟายเออร์ที่สนับสนุนในวิธีการอินพุต ภาษากรีก (GIM)

ตัวดัดแปลงที่ได้รับการสนับสนุนใน GIM มีดังนี้:

ตัวดัดแปลง	การแทนค่าเลขฐานสิบหก
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

ตัวดัดแปลงภายในที่ได้รับการสนับสนุนใน GIM มีดังนี้:

ตัวดัดแปลง	การแทนค่าเลขฐานสิบหก
ชั้นภาษากรีก	0x20

วิธีการอินพุตภาษาญี่ปุ่น (JIM)

ส่วนนี้อธิบายวิธีการอินพุตภาษาญี่ปุ่น (JIM)

วิธีการอินพุตภาษาญี่ปุ่น (JIM) มีคุณลักษณะต่อไปนี้:

- สนับสนุนการแปลงจากอักขระ Romaji เป็น Kana (RKC)
- สนับสนุนการแปลงจากอักขระ Kana เป็น Kanji (KKC)
- มีอินพุตอักขระ Hankaku (ครึ่งความกว้าง) และ Zenkaku (เต็มความ กว้าง)
- มีการค้นหาในพจนานุกรมระบบและผู้ใช้
- มีการลงทะเบียนรันไทม์ของคำในพจนานุกรมผู้ใช้
- ต้องใช้ฟังก์ชัน Callback เพื่อสนับสนุนสิ่งดังต่อไปนี้:
  - ภาพวาดสถานะและการแก้ไขก่อนหน้า
  - เมนูตัวเลือกทั้งหมด
  - อินพุตหมายเลข JIS Kutan และอินพุตหมายเลข IBM Kanji
- สนับสนุนชุดรหัส IBM-943, IBM-eucJP และ UTF-8 สำหรับการประมวลผลภายใน JIM ใช้ชุดรหัส IBM-943 อย่างไรก็ตาม JIM สนับสนุนชุดรหัสใดๆ เช่น IBM-eucJP ซึ่งสามารถแปลงจาก IBM-943
- ตั้งอยู่ในไฟล์ `/usr/lib/nls/loc/JP.im`วิธีการอินพุต แบบท้องถิ่นอื่นทั้งหมดมีการแสดงเป็นชื่อย่อในไฟล์นี้

Katakana และ Hiragana ต่างประกอบด้วยอักขระประมาณ 50 ตัว และทำให้เกิดเป็นชุดของอักขระการออกเสียงที่เรียกว่า Kana เสียงทั้งหมด ในภาษาญี่ปุ่นสามารถแสดงในอักขระ Kana ได้

Kanji คือชุดของอักขระทางภูมิศาสตร์ หลักการแบบง่ายสามารถแสดงโดยใช้ อักขระ Kanji เพียงตัวเดียว ในขณะที่ความหมายซึ่งซับซ้อนมากขึ้นสามารถแสดง โดยใช้สตริงของอักขระ Kanji อักขระ Kanji มีอยู่หลายพัน ตัว

ภาษาญี่ปุ่นยังสามารถใช้ตัวอักษร Roman ได้ด้วย ซึ่งเรียกว่า Romaji ตัวอักษร Roman มีอยู่ 26 อักขระ ตัวอักษรนี้ใช้กันมากที่สุดในสภาพแวดล้อมทางเทคนิค และอาชีพเพื่อแสดงถึงคำศัพท์เทคนิคซึ่ง ไม่มีอยู่ในภาษาญี่ปุ่น โดยปกติแล้ว ประโยคทั่วไปมีการใช้อักขระ Katakana, Hiragana, Kanji, Romaji, ตัวเลข และอักขระอื่นๆ ผสมกัน

การประมวลผลอักขระภาษาญี่ปุ่น

Japanese Industrial Standard (JIS) ระบุเกี่ยวกับอักขระ Kanji จำนวน 7000 ตัวที่ประมวลผลโดยระบบคอมพิวเตอร์ ผลิตภัณฑ์ภาษาญี่ปุ่น ที่จัดทำโดยผู้ผลิตรายนี้สนับสนุนอักขระมาตรฐานทั้งหมด ตลอดจนอักขระชนิดอื่นๆ

อินพุตของอักขระได้มาจากการแปลงต่อไปนี้:

- การแปลง Kana-เป็น-Kanji (KKC)
- การแปลง Romaji-เป็น-Kana (RKC)

ตารางที่ 12. คีย์ภาษาญี่ปุ่นพิเศษ

ฟังก์ชันคีย์	Key name	คำอธิบายฟังก์ชัน
คีย์การไม่แปลง KKC	muhenkan	ปล่อยอักขระ Kana ตามสภาพ
คีย์การแปลง KKC	henkan	แปลง Kana เป็น Kanji
KKC คีย์ตัวเลือกทั้งหมด	zenkouho	แสดงตัวแทน Kanji ทั้งหมดที่เป็นไปได้
คีย์ RKC Romaji Mode	romaji	สลับเปิดและปิด RKC
คีย์ Hiragana Shift	hiragana	กลายเป็นคำสั่งเปลี่ยน Hiragana
คีย์ Katakana Shift	katakana	กลายเป็นคำสั่งเปลี่ยน Katakana
คีย์ Romaji Shift	eisu	กลายเป็นคำสั่งเปลี่ยน Romaji

หมายเหตุ: คำสั่งเปลี่ยนมีการรักษาไว้จนกว่าคุณจะกดคีย์ shift อื่น คำสั่งแรกเริ่มคือ Romaji

เทคโนโลยีการแปลงคานะ-เป็น-คันจิ (KKC)

วิธีอินพุตภาษาญี่ปุ่น (JIM) ของเทคโนโลยี KKC อิง ตามข้อเท็จจริงที่ว่าอักขระคันจิทุกตัว หรือชุดของอักขระคันจิ มีการออกเสียงที่สามารถแสดงโดยอักขระคาตากานะ หรือ ฮิรากานะ

การอินพุตอักขระ Hiragana หรือ Katakana ง่ายกว่าการอินพุตอักขระ Kanji JIM จะวิเคราะห์ค่าการออกเสียงของอักขระ Hiragana และ Katakana เพื่อกำหนดอักขระ Kanji เทียบเท่าที่ดีที่สุด การวิเคราะห์การออกเสียงดังกล่าวขึ้นอยู่กับพจนานุกรมและตารางที่แสดงใน JIM

โหมดอินพุต

ส่วนนี้อธิบายโหมดต่างๆ ในวิธีการอินพุต ภาษาญี่ปุ่น

วิธีการอินพุตภาษาญี่ปุ่น (JIM) มีโหมดต่อไปนี้ที่ สามารถใช้เพื่อควบคุมการประมวลผลอินพุต:

- การแม็พคีย์บอร์ด
  ช่วยในการเรียกใช้ตัวอักษรและตัวเลขโหมด Katakana หรือ Hiragana
- ขนาดอักขระ
  อินพุตในโหมด Zenkaku (เต็มความกว้าง) หรือ Hankaku (ครึ่งความกว้าง)
- เปิด/ปิด RKC
  อินพุต Kana โดยตรงหรือเรียกใช้โหมดการเรียบเรียงการแก้ไขก่อนหน้า เพื่ออินพุต Kana พร้อมด้วยชุดของอักขระตัวอักษร ฟังก์ชัน การแก้ไขก่อนหน้าช่วยให้สามารถประมวลผลอักขระได้ก่อนที่อักขระนั้นจะถูก committed ในแอ็พพลิเคชัน

เมื่อโหมดการแม็พคีย์บอร์ดเป็นตัวอักษรและตัวเลข และโหมดขนาดอักขระ เป็น Hankaku วิธี JIM จะแม็พคีย์กับอักขระ Romaji ชุด โหมดแบบนี้เรียกว่าโหมด "ภาษาอังกฤษ" การแก้ไขก่อนหน้า ไม่เป็นที่ต้องการในโหมดภาษาอังกฤษและไม่สามารถเรียกใช้ได้ โดยไม่คำนึงถึง การตั้งค่าโหมด RKC ชุดโหมดแบบอื่นๆ อาจเริ่มต้นการแก้ไขก่อนหน้า และอักขระที่สร้างขึ้นในโหมดเหล่านี้ไม่ใช่ ASCII

คีย์ต่อไปนี้ใช้เพื่อทำการแปลงจาก Kana-เป็น-Kanji โดย JIM

Keysym	การแม็พคีย์บอร์ด
Katakana	เปลี่ยน Katakana
Eisu_toggle	เปลี่ยนตัวอักษรและตัวเลข
Hiragana	เปลี่ยน Hiragana

Keysym	ขนาดอักขระ
Zenkaku_Hankaku	สลับระหว่างเต็มความกว้างและครึ่งความกว้าง
Hankaku	ครึ่งความกว้าง
Zenkaku	เต็มความกว้าง

Keysym	RKC on/off
Alt-Hiragana	เปิด/ปิดการแปลงจาก Romaji-เป็น-Kana
Romaji	*ผลเหมือนกัน

* Keysyms เฉพาะของผู้ผลิต

คีย์ต่อไปนี้ยังสามารถใช้เมื่อ JIM เป็นการแก้ไขก่อนหน้าของสตริง Kanji

Keysym	การแก้ไขก่อนหน้าสำหรับ Kanji
Muhenkan	ไม่มีการแปลง - commit Kana
Henkan	การแปลง - เรียกใช้ตัวเลือกถัดไป
Kanji	เหมือนกับ Henkan
BunsetsuYomi	*ย้ายวลีกลับ
MaeKouko	*ย้ายไปยังตัวเลือกก่อนหน้านี้
LeftDouble	*ย้ายเคอร์เซอร์ไปทางซ้ายสองอักขระ
RightDouble	*ย้ายเคอร์เซอร์ไปทางขวาสองอักขระ
ErInput	*ทิ้งสตริงการแก้ไขก่อนหน้าปัจจุบัน

Keysym	Auxiliary pre-edit
Alt-Henkan	ตัวเลือกทั้งหมด
Touroku	การลงทะเบียนรันไทม์
ZenKouho	*ตัวเลือกทั้งหมด (ผลเหมือนกัน)
KanjiBangou	*อินพุตตัวเลข Kanji
HenkanMenu	*เปลี่ยนโหมดการแปลง

* Keysyms เฉพาะของผู้ผลิต

การแม็พคีย์บอร์ด

คำสั่งการแม็พคีย์บอร์ดที่ใช้ได้มีดังต่อไปนี้: Alphanumeric (Romaji), Katakana, และ Hiragana แต่ละคำสั่งสามารถเรียกใช้โดย keysym ที่ทำหน้าที่เป็นคีย์เปลี่ยนการล็อค Keysyms คือการเปลี่ยน Katakana, Eisu_toggle, และ Hiragana

เมื่อกด keysyms อย่างใดอย่างหนึ่ง การแม็พคีย์บอร์ดจะป้อน คำสั่งที่เชื่อมโยงกับคีย์ คำสั่งนี้มีการรักษาไว้จนกว่ามีการกด keysyms อื่นอย่างใดอย่างหนึ่ง คำสั่งเปลี่ยนแรกเริ่มคือ Eisu_toggle ซึ่งสามารถเปลี่ยนได้โดยการกำหนดเอง

เมื่อคุณเรียกใช้คำสั่ง Hiragana หรือ Katakana แต่ละคีย์จะถูกแม็พ กับอักขระการออกเสียงภายในชุดอักขระตามลำดับ ตัวอย่างเช่น ถ้าคุณกด *q* อักขระ Hiragana ที่ออกเสียงว่า "ta" จะถูกจัดทำขึ้น ในระหว่างคำสั่งเปลี่ยน Hiragana หรืออักขระ Katakana ที่ออกเสียงว่า "ta" จะถูกจัดทำขึ้นในระหว่างคำสั่งเปลี่ยน Katakana หรือ Romaji *q* จะถูก จัดทำขึ้นในระหว่างคำสั่งเปลี่ยน Eisu_toggle บนคีย์บอร์ด IBM ภาษาญี่ปุ่น ด้านบนของคีย์จะแสดงสัญลักษณ์ครบทั้งสามสัญลักษณ์

นอกจากนี้ เมื่อการแม็พคีย์บอร์ดอยู่ในคำสั่ง Hiragana วิธีการอินพุตจะถูก วางไว้โดยอัตโนมัติในโหมดการแก้ไขก่อนหน้าการเรียบเรียง ซึ่งอักขระ Hiragana แต่ละตัวสามารถถูกแปลงเป็นอักขระ Kanji ได้

บางคีย์มีการกำหนดอักขระ Hiragana หรือ Katakana สองตัว ตัว อย่างเช่น คีย์ 7 มีอักขระ Hiragana ตัวใหญ่และตัวเล็กซึ่งทั้งสองตัว มีการออกเสียงว่า "ya" อักขระเหล่านี้ไม่เทียบเท่ากับอักขระตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก ของอักขระอื่น เนื่องจาก Kanji, Hiragana, และ Katakana ไม่มี ตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก อักขระตัวเล็กใช้เพื่อ ระบุเสียงการออกเสียงพิเศษ อักขระเหล่านี้สามารถแยกแยะได้ โดยใช้คีย์ shift

ขนาดอักขระ

ชุดย่อยของชุดอักขระภาษาญี่ปุ่นมีการแสดงทั้งในขนาด เต็มความกว้างและครึ่งความกว้าง โดยปกติ อักขระทางภูมิศาสตร์ Kanji มีขนาดเต็มความกว้าง อักขระการออกเสียงและ ASCII มีการแสดงทั้งในขนาดเต็มความกว้าง และครึ่งความกว้าง ผู้ใช้ควบคุมขนาดอักขระโดย การกด Zenkaku_Henkaku keysym ซึ่งจะสลับระหว่างเต็มความกว้าง และครึ่งความกว้าง

การแปลง Romaji-เป็น-Kana (RKC)

สำหรับผู้ใช้ที่คุ้นเคยกับคีย์บอร์ดตัวอักษรและตัวเลข การพิมพ์เสียงที่ออก แทนอักขระ Hiragana หรือ Katakana เป็นสิ่งที่ง่ายกว่า JIM มีการแปลง Romaji-เป็น-Kana (RKC) ที่ช่วยให้ผู้ใช้สามารถพิมพ์ เสียงที่ออกของอักขระ Hiragana หรือ Katakana บนคีย์บอร์ดตัวอักษรและตัวเลขได้

การแก้ไขก่อนหน้าสำหรับ Kanji

เมื่อใช้งานในโหมดการแปลง Romaji-เป็น-Kana คุณต้อง ทำสองขั้นตอนเพื่อจัดทำอักขระ Kanji อันดับแรก ผู้ใช้ป้อนอักขระฮิรากานะโดยการพิมพ์อักขระการออกเสียงแบบโรมันจิ ใน ขั้นตอนนี้ คุณจัดทำอักขระ Hiragana โดยการพิมพ์คีย์ตัวอักษร Romaji 1 ถึง 3 ซึ่งประกอบขึ้นเป็นเสียงของอักขระ Hiragana ขั้นตอนที่สอง แปลงอักขระ Hiragana เป็นอักขระ Kanji โดยการกดคีย์ Henkan อักขระ Kanji จำนวนมากมีการออกเสียงเหมือน กัน คีย์ Henkan จะแสดงตัวเลือก Kanji ที่ใกล้เคียงที่สุด การกดคีย์ Henkan ซ้ำๆ จะแสดงตัวเลือกเพิ่มเติมทั้งหมด

ตัวอย่างเช่น เมื่อคุณป้อนอักขระ Kanji สำหรับเสียงที่ออกว่า "k-a-n-j-i" คุณต้องทำสองอย่างดังนี้:

1. ตั้งค่าการแม็พคีย์บอร์ดเป็นคำสั่ง Hiragana
2. เปิดใช้งานการแม็พ Romaji-กับ-Kana โดยการกดคีย์ Alt-Hiragana การดำเนินการนี้จะเรียกใช้คีย์บอร์ดตัวอักษรและตัวเลข

ขณะนี้ คุณสามารถกดคีย์ที่สะกดว่า "kanji" เมื่อเสียงที่ออก แต่ละเสียงเสร็จสิ้นแล้ว อักขระ Hiragana จะปรากฏขึ้น

อักขระ Hiragana แสดงขึ้นพร้อมกับภาพที่บ่งชี้ว่า JIM กำลังเรียบเรียงอยู่ในคำสั่งการแก้ไขก่อนหน้า อักขระมีการขีดเส้นใต้และแสดงขึ้นในภาพย้อนกลับ ฟังก์ชันผลป้อนกลับนี้เรียกว่า *callback*

เมื่อต้องการแปลงอักขระ Hiragana ภายในสตริงการแก้ไขก่อนหน้าเป็นอักขระ Kanji ให้กดคีย์ Henkan ตัวเลือกที่ใกล้เคียงมากที่สุด ซึ่งเชื่อมโยงกับเสียง Hiragana จะแสดงขึ้น การกดคีย์นี้ซ้ำๆ จะแสดงตัวเลือกอื่น

ในระหว่างกระบวนการเรียบเรียง สตริงการแก้ไขก่อนหน้าจะถูกแบ่งพาร์ทิชัน เป็นเซ็กเมนต์ซึ่งอาจพิจารณาว่าเป็นคำ Kanji ได้ หลังจากที่สตริงของอักขระ kana ถูกแปลงเป็นตัวเลือกแล้ว สตริงจะมีการจัดการเป็นเซ็กเมนต์ที่แปลงได้ ดังกล่าวเซ็กเมนต์หนึ่ง ในขณะที่สตริงการแก้ไขก่อนหน้าแสดงขึ้น JIM จะใช้คีย์เคอร์เซอร์และคีย์อื่นๆ เพื่อดำเนินการสตริง

เพื่อ commit สตริงการแก้ไขก่อนหน้าในโปรแกรม ผู้ใช้ต้องกดปุ่ม Enter ในกรณีนี้ โค้ดของปุ่ม Enter เองจะไม่ถูกส่งไปยังโปรแกรม แต่จะส่งเฉพาะสตริงเท่านั้น

Muhenkan keysym ยังสามารถใช้เพื่อปิดการแก้ไขก่อนหน้าและ commit อักขระ Hiragana หรือ Katakana ในโปรแกรมโดยตรงได้ด้วย

ตารางต่อไปนี้แสดงการแปลงคำสั่งเปลี่ยนและการโต้ตอบระหว่าง คีย์โหมด RKC กับคำสั่งเปลี่ยน

อักขระที่เข้ารหัส	จุดโค้ด	คำอธิบาย	จำนวน
000xxxxx	00–1F	ตัวควบคุม	32
00100000	20	พื้นที่ว่าง	1
0xxxxxxx	21–7E	7 บิต ASCII	94
01111111	7F	ลบ	1
10000000	80	ไม่ได้กำหนด	1
100xxxxx 01xxxxxx	[81–9F][40–7E]	ไบต์คู่	1953
100xxxxx 1xxxxxxx	[81–9F][80–FC]	ไบต์คู่	3844
10100000	A0	ไม่ได้กำหนด	1
1xxxxxxx	A1–DF	8 บิตไบต์เดียว	63
111xxxxx 01xxxxxx	[E0–FC][40–7E]	ไบต์คู่	1827
111xxxxx 1xxxxxxx	[E0–FC][80–FC]	ไบต์คู่	3596
11111101	FD	ไม่ได้กำหนด	1
11111110	FE	ไม่ได้กำหนด	1
11111111	FF	พร้อมกัน	1

JIM มีชนิดพื้นที่เสริมดังต่อไปนี้:

- เมนูตัวเลือกทั้งหมด
- กล่องโต้ตอบอินพุตตัวเลข Kanji
- เมนูโหมดการแปลง
- กล่องโต้ตอบการลงทะเบียนรันไทม์

การดำเนินการแปลงจาก Kana-เป็น-Kanji บนสตริงของอักขระ Hiragana หรือ Katakana อาจให้ตัวเลือก Kanji ได้ตั้งแต่หนึ่งถึงร้อยตัว ที่แย่ที่สุดคือคุณอาจต้องกดคีย์การแปลงมากกว่าร้อยครั้งเพื่อให้ได้ อักขระ Kanji ที่ถูกต้อง

ในกรณีเช่นนั้น การค้นหาอักขระที่ถูกต้องโดยการร้องขอเมนู ตัวเลือกทั้งหมดโดยใช้ ZenKouho หรือ Alt-Henkan keysym เป็นวิธีที่สะดวกกว่า เมนูนี้จะแสดงขึ้นถ้าเป้าหมายปัจจุบัน (คำ Kanji ที่เคอร์เซอร์ชี้อยู่ในพื้นที่การแก้ไขก่อนหน้า) มีตัวเลือกอื่นที่เชื่อมโยงด้วย หลายตัว เมนูมีตัวเลือกให้เลือก หลายตัว เมนูตัวเลือกทั้งหมดจะหายไปเมื่อกด keysym รีเซ็ต หรือกดปุ่ม Enter หรือเมื่อเลือกตัวเลือก

กล่องโต้ตอบอินพุตตัวเลข Kanji จะพร้อมต์ผู้ใช้ให้เลือกอักขระ Kanji โดยการป้อน 3 ถึง 5 ตัวเลข ตัวเลขแสดงถึงโค้ด ของอักขระ พจนานุกรมออนไลน์ช่วยให้ผู้ใช้สามารถค้นหา โค้ดได้ รูปแบบการจัดลำดับสำหรับพจนานุกรมประเภทนี้มีความแตกต่างกันไป ตัวอย่างเช่น พจนานุกรมหนึ่งอาจแสดงรายการโค้ดโดยเรียงตามเสียงที่ออก พจนานุกรมอื่นอาจจัดลำดับ รหัสโดยเรียงตามจำนวนของ strokes ที่ใช้ในการเขียนอักขระ KanjiBangou keysym จะเรียกใช้เมนูนี้ เมนูถูกยุติด้วย Reset หรือ Return keysym.

HenkanMenu keysym จะเรียกใช้เมนูโหมดการแปลง มีไอเท็มแสดงขึ้น ให้เลือกสี่ไอเท็ม ไอเท็มที่สำคัญที่สุดคือโหมดการแปลงคำ และโหมดการแปลงวลี เลือกโดยการเลือกหมายเลข และกด Return keysym เมนูนี้จะยุติลงเมื่อเลือกไอเท็มแล้ว หรือเมื่อกด Reset keysym อย่างใดอย่างหนึ่ง

ไดอะล็อกการรีจิสเตอร์รันไทม์พร้อมต์ให้ผู้ใช้ป้อนสตริง Kana และสตริงคันจิสำหรับการรีจิสเตอร์การแม็พของสตริง ในพจนานุกรมผู้ใช้ หลังจากลงทะเบียนอักขระทั้งสองชนิดแล้ว JIM สามารถ ใช้อักขระนั้นเป็นตัวเลือกการแปลงได้ เมนูถูกยุติด้วย Escape หรือ Reset keysym

การนำเสนอของเมนูขึ้นอยู่กับสภาพแวดล้อมอินเตอร์เฟส ซึ่ง JIM ดำเนินการอยู่ ตัวอย่างเช่น บางอินเตอร์เฟสสนับสนุนเมนูแบบเลื่อนที่ใช้คีย์ Page Down และ Page Up

หลักการที่เกี่ยวข้อง:

"การใช้ callbacks" ในหน้า 148

แอ็พพลิเคชันที่ใช้วิธีการอินพุตควรมีฟังก์ชัน callback เพื่อให้ตัวแก้ไขวิธีการอินพุต (IMED) สามารถสื่อสารกับ ผู้ใช้ได้ ชนิดของวิธีการอินพุตที่คุณใช้เป็นตัวกำหนดว่า callbacks เป็นสิ่งที่จำเป็นหรือไม่ ตัวอย่างเช่น วิธีการอินพุตไบต์เดียวไม่ต้องการ callbacks แต่วิธีการอินพุตภาษาญี่ปุ่นใช้ callbacks อย่างมากใน ฟังก์ชันการแก้ไขก่อนหน้า ฟังก์ชันการแก้ไขก่อนหน้าช่วยให้สามารถประมวลผลอักขระได้ก่อนที่ อักขระนั้นจะถูก committed ในแอ็พพลิเคชัน

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษาญี่ปุ่น (JIM)

คีย์แม็พที่ได้รับการสนับสนุนใน JIM มีดังต่อไปนี้:

- ja_JP.IBM-eucJP.imkeymap
- ja_JP.IBM-eucJP@alt.imkeymap
- Ja_JP.IBM-943.imkeymap
- Ja_JP.IBM-943@alt.imkeymap
- JA_JP.UTF-8.imkeymap
- JA_JP.UTF-8@alt.imkeymap

Keysyms

JIM ใช้ keysyms ในกลุ่ม **XK_KATAKANA, XK_LATIN1,** และ **XK_MISCELLANY**

Keysyms ที่สงวนไว้ต่อไปนี้ใช้เฉพาะกับวิธีการอินพุตของ ระบบนี้เท่านั้น:

Keysym	การแทนค่าเลขฐานสิบหก	คำอธิบาย
XK_BunsetsuYomi	0x1800ff05	ย้อนวลีกลับไปยัง Yomi
XK_MaeKouho	0x1800ff04	ตัวเลือกก่อนหน้านี้
XK_ZenKouho	0x1800ff01	ตัวเลือกทั้งหมด
XK_KanjiBangou	0x1800ff02	การป้อนตัวเลขคันจิ
XK_HenkanMenu	0x1800ff03	เปลี่ยนโหมดการแปลง
XK_LeftDouble	0x1800ff06	ย้ายเคอร์เซอร์ไปทางซ้ายสองตัวอักษร
XK_RightDouble	0x1800ff07	ย้ายเคอร์เซอร์ไปทางขวาสองตัวอักษร
XK_LeftPhrase	0x1800ff08	สำรองไว้ใช้ในอนาคต
XK_RightPhrase	0x1800ff09	สำรองไว้ใช้ในอนาคต
XK_ErInput	0x1800ff0a	ทิ้งสตริงการแก้ไขก่อนหน้าปัจจุบัน
XK_Resetreset	0x1800ff0b	รีเซ็ต

Keysym	คำอธิบาย
XK_Kanji	แปลง Hiragana เป็น Kanji
XK_Muhenkan	ยกเลิกการแปลง
XK_Romaji	วาง JIM ในโหมดอินพุต Romaji
XK_Hiragana	วาง JIM ในโหมดอินพุต Hiragana
XK_Katakana	วาง JIM ในโหมดอินพุต Katakana
XK_Zenkaku_Hankaku	สลับระหว่างโหมดอินพุตอักขระเต็มความกว้างและครึ่งความ กว้าง
XK_Touroku	ลงทะเบียนคำในพจนานุกรมผู้ใช้
XK_Eisu_toggle	วาง JIM ในโหมดอินพุตตัวอักษรและตัวเลข

ตัวดัดแปลง

ส่วนนี้อธิบายโมดิฟายเออร์ที่สนับสนุนบน Japanese input method (JIM)

ตัวดัดแปลงที่ได้รับการสนับสนุนใน JIM มีดังต่อไปนี้:

โมดิฟายเออร์	การแทนค่าเลขฐานสิบหก
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10

ตัวดัดแปลงภายในที่ได้รับการสนับสนุนใน JIM มีดังต่อไปนี้:

โมดิฟายเออร์	การแทนค่าเลขฐานสิบหก
Kana	0x20
Romaji	0x40

วิธีการอินพุตภาษาเกาหลี (KIM)

ส่วนนี้อธิบายวิธีการอินพุตภาษาเกาหลี (KIM)

ชุดโค้ด EUC ภาษาเกาหลีประกอบด้วยกลุ่มอักขระหลัก ดังต่อไปนี้:

- ASCII (ภาษาอังกฤษ)
- Hangul (อักขระภาษาเกาหลี)

ชุดโค้ด Hangul ประกอบด้วยอักขระ Hangul และ Hanja (ภาษาจีน) อักขระ Hangul หนึ่งตัวอาจประกอบด้วยพยัญชนะและสระหลายตัว อย่างไรก็ตาม คำ Hangul ส่วนใหญ่สามารถระบุใน Hanja ได้ อักขระ Hanja แต่ละตัว มีความหมายเป็นของตัวเอง ดังนั้นจึงมีลักษณะเฉพาะมากกว่า Hangul

ชุดโค้ดมาตรฐานภาษาเกาหลีปัจจุบัน, KSC5601, มีอักขระ Hangul, Hanja, และอักขระพิเศษจำนวน 8224 อักขระ เพื่อให้เป็นไปตามมาตรฐานภาษาเกาหลี Extended UNIX Code (EUC) ชุดอักขระนี้จึงถูกกำหนดให้กับ CS1 ของ EUC

อินพุตของอักขระได้มาจากแหล่งดังต่อไปนี้:

- ASCII

  โหมด ASCII ใช้สำหรับการป้อนอักขระภาษาอังกฤษ
- Hangul

  คีย์ XK_Hangul จะเรียกใช้โหมด Hangul ซึ่งต้องใช้ เพื่อป้อนอักขระ Hangul หลังจากเรียกใช้โหมด Hangul แล้ว KIM จะเรียบเรียงพยัญชนะและสระที่เข้ามาตามกฎการประกอบ Hangul อักขระ Hangul ประกอบด้วยพยัญชนะตามด้วย สระ พยัญชนะสุดท้ายเป็นอ็อพชัน ถ้าอักขระที่เข้ามาละเมิด กฎการสร้าง จะมีเสียงบีปเตือน

  มีอักขระพิเศษประมาณ 1500 อักขระในชุดโค้ดมาตรฐาน ต้องป้อนอักขระ เหล่านี้ด้วยฟังก์ชัน Code Input ของ KIM คีย์ Code Input จะเรียกใช้ฟังก์ชัน Code Input เมื่อมีการเรียกใช้ฟังก์ชัน Code Input จุดโค้ดสำหรับอักขระที่ต้องการจะสามารถป้อนได้ใน หน้าต่างเสริม Code Input
- Hanja

คีย์ XK_Hangul_Hanja จะเรียกใช้โหมด Hanja อักขระ Hanja สามารถถูกแปลงจากอักขระ Hangul ที่เหมาะสมเท่านั้น การแปลง Hangul-เป็น-Hanja (HHC) มีอยู่สองโหมดดังนี้: ตัวเลือกเดียว และหลายตัวเลือก ในบริบทนี้ ตัวเลือกคือการเลือกตัวเลือกอักขระที่เป็นไปได้

ในโหมดตัวเลือกเดียว ตัวเลือกจะแสดงขึ้น ทีละตัวบนบรรทัดคำสั่ง ในโหมดหลายตัวเลือก ตัวเลือกจะ แสดงขึ้นพร้อมกันได้มากถึงสิบตัวเลือกในหน้าต่างเสริม

เมื่อใช้โหมดการแปลง Hanja อักขระ Hangul สามารถถูกแปลงเป็น Hanja เมื่อกดคีย์การแปลง ในลักษณะคล้ายกัน คำ Hanja ใดๆ สามารถถูกแปลงเป็นคำ Hangul ที่เหมาะสมได้

Hanja ยังสามารถป้อนได้โดยใช้ฟังก์ชัน Code Input ในลักษณะเดียวกับที่ใช้สำหรับ การป้อน Hangul

เพื่อใช้สำหรับการแปลงเหล่านี้ จึงมีการแสดงคีย์พิเศษต่อไปนี้บน คีย์บอร์ดภาษาเกาหลี 106-คีย์

ตารางที่ 13. คีย์ภาษาเกาหลีพิเศษ

ฟังก์ชันคีย์	Keysym	คำอธิบายฟังก์ชัน
คีย์สลับ Hangul/ภาษาอังกฤษ	XK_Hangul	สลับระหว่างโหมด Hangul และภาษาอังกฤษ
คีย์สลับ Hanja	XK_Hangul_Hanja	สลับระหว่างการเปิดและปิดโหมด Hanja
คีย์ Code Input	XK_Hangul_ Codeinput	เรียกใช้ฟังก์ชัน Code Input ซึ่งช่วยให้สามารถป้อนอักขระ โดยใช้จุดโค้ดได้
คีย์ตัวเลือกทั้งหมด HHC	XK_Hangul_ MultipleCandidate	เรียกใช้โหมดหลายตัวเลือก
คีย์การแปลง HHC	XK_Hangul_ Conversion	เรียกใช้โหมดตัวเลือกเดียวและยังเลื่อนจอภาพตัวเลือก ไปข้างหน้า ทั้งในโหมดตัวเลือกเดียวและหลายตัวเลือก ด้วย
คีย์การไม่แปลง HHC	XK_Hangul_ NonConversion	เลื่อนจอภาพตัวเลือกย้อนหลัง

วิธีการอินพุตภาษาลัตเวียน (LVIM)

วิธีการอินพุตภาษาลัตเวีย (LVIM) คล้ายกับ single-byte input method (SIM) ยกเว้นว่าจะถูกกำหนดเองสำหรับการประมวลผล คีย์บอร์ดภาษาลัตเวีย

คุณลักษณะของ LVIM มีดังนี้:

- สนับสนุนกลุ่ม QWERTY และ Ergonomic เป็นกลุ่มหลักสองกลุ่ม มีกลุ่มเสริม เพิ่มเติมอีกสองกลุ่มซึ่งสามารถเข้าถึงได้โดยใช้ dead keys จากกลุ่มหลักทั้งสองกลุ่มดังนี้:
  - การกดคีย์ left-alt และคีย์ left-shift พร้อมกัน จะวาง คีย์บอร์ดไว้ในกลุ่ม Ergonomic
  - การกดคีย์ left-alt และคีย์ right-shift พร้อมกัน จะวาง คีย์บอร์ดไว้ในกลุ่ม QWERTY
- สนับสนุนชุดโค้ด IBM-921

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษาลัตเวียน (LVIM)

คีย์แม็พที่ได้รับการสนับสนุนใน LVIM มีดังต่อไปนี้:

- Lv_LV.IBM-921.imkeymap

วิธีการอินพุตภาษาลิทัวเนียน (LTIM)

วิธีการอินพุตภาษาลิทัวเนีย (LTIM) คล้ายกับ single-byte input method (SIM) ยกเว้นว่าจะถูกกำหนดเองสำหรับการประมวลผล คีย์บอร์ดภาษาลิทัวเนีย

คุณลักษณะของ LTIM มีดังนี้:

- สนับสนุนกลุ่มที่ลงโปรแกรมและภาษาลิทัวเนียนเป็นกลุ่มหลักสองกลุ่ม มีกลุ่มเสริมเพิ่มเติมอีกสองกลุ่มซึ่งสามารถเข้าถึงได้โดยใช้ dead keys จากกลุ่มหลักทั้งสองกลุ่มนี้
  - การกดคีย์ left-alt และคีย์ left-shift พร้อมกัน จะวาง คีย์บอร์ดไว้ในกลุ่มภาษาลิทัวเนียน
  - การกดคีย์ left-alt และคีย์ right-shift พร้อมกัน จะวาง คีย์บอร์ดไว้ในกลุ่มที่ลงโปรแกรม
- สนับสนุนชุดโค้ด IBM-921

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษาลิทัวเนียน (LTIM)

คีย์แม็พที่ได้รับการสนับสนุนใน LTIM มีดังต่อไปนี้:

- Lt_LT.IBM-921.imkeymap

วิธีการอินพุตภาษาไทย (THIM)

วิธีการอินพุตภาษาไทยคล้ายกับ single-byte input method (SIM) ยกเว้นว่าจะถูกกำหนดเองสำหรับการประมวลผลภาษาไทย

อธิบายโดยละเอียดคือ วิธีการนี้ได้รับการออกแบบมาเพื่อป้องกันการป้อนชุดของ อักขระภาษาไทย (พยัญชนะ สระข้างบน/ข้างล่าง วรรณยุกต์) ที่ ไม่ถูกต้องในภาษาไทย คุณลักษณะของ THIM มี ดังนี้:

- สนับสนุนกลุ่มภาษาลาตินและไทยเป็นสองกลุ่มหลักบน คีย์บอร์ด
  - การกดคีย์ left-alt และคีย์ left-shift จะวางคีย์บอร์ดไว้ใน กลุ่มภาษาไทย
  - การกดคีย์ left-alt และคีย์ right-shift จะวางคีย์บอร์ดไว้ใน กลุ่มภาษาลาติน
- สนับสนุนชุดโค้ด TIS-620

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษาไทย (THIM)

คีย์แม็พที่ได้รับการสนับสนุนใน THIM มีดังต่อไปนี้:

- th_TH.TIS-620.imkeymap

วิธีการอินพุตภาษาเวียตนาม (VNIM)

วิธีการอินพุตภาษาเวียดนาม (VNIM) คล้ายกับ single-byte input method (SIM) ยกเว้นว่าจะถูกกำหนดเองสำหรับการประมวลผล คีย์บอร์ดภาษาเวียดนาม

อธิบายโดยละเอียดคือ วิธีการนี้ได้รับการออกแบบมาเพื่อป้องกันการป้อนชุดของ อักขระภาษาเวียตนาม (เครื่องหมายวรรณยุกต์) ที่ไม่ถูกต้องใน ภาษาเวียตนาม อักขระเครื่องหมายวรรณยุกต์ภาษาเวียตนามสามารถป้อนได้ทันที ต่อจากสระภาษาเวียตนาม อย่างใดอย่างหนึ่ง (a, e, i, o, u, y, a-circumflex, e-circumflex, o-circumflex, a-breve, o-horn, หรือ u-horn)

VNIM สนับสนุนชั้นคีย์บอร์ดเดียว รวมถึงอักขระที่เรียบเรียงไว้แล้วและ เครื่องหมายวรรณยุกต์ภาษาเวียตนามบางรายการ

VNIM สนับสนุนชุดโค้ด IBM-1129

คีย์แม็พ

ส่วนนี้อธิบายคีย์แม็พที่ได้รับการสนับสนุนในวิธีการอินพุต ภาษาเวียตนาม (VNIM)

คีย์แม็พที่ได้รับการสนับสนุนใน VNIM มีดังต่อไปนี้:

- Vi_VN.IBM-1129.imkeymap

วิธีการอินพุตภาษาจีนแบบง่าย (ZIM-UCS)

ส่วนนี้อธิบายวิธีการอินพุตภาษาจีนแบบง่าย (ZIM-UCS)

ชุดโค้ด UCS-2 ประกอบด้วยกลุ่มอักขระเกือบทั้งหมด กลุ่ม อักขระที่มีอยู่สำหรับโลแคล ZH_CN มีดังต่อไปนี้:

- ASCII (ภาษาอังกฤษ)
- Glyphs
- อักขระภาษาจีน ภาษาญี่ปุ่น และภาษาเกาหลี (CJK) (อักขระมาตรฐาน)

ชุดอักขระ CJK ประกอบด้วย 20,992 ตำแหน่งอักขระ แต่มีการกำหนดให้กับ อักขระภาษาจีน 20,902 ตำแหน่งเท่านั้น

การออกเสียงของภาษาจีนแบบง่ายมีการแสดงโดยใช้สัญลักษณ์การออกเสียง ที่เรียกว่า Bopomofo สัญลักษณ์การออกเสียงมีอยู่ 25 รายการ อักขระ ภาษาจีนแบบง่ายมีการแสดงโดยสัญลักษณ์การออกเสียงหนึ่งถึงสาม

คุณลักษณะ ZIM-UCS มีลักษณะดังต่อไปนี้:

- วิธีการอินพุตที่ใช้ทั่วไปมีดังต่อไปนี้:

  Intelligent ABC
  วิธีการอินพุตตามการแสดงการออกเสียงของอักขระ ภาษาจีน

  วิธีการอินพุต Pin Yin
  วิธีการอินพุตตามการแสดงการออกเสียงของอักขระ ภาษาจีน อักขระภาษาจีนมีการแบ่งออกเป็นหนึ่งหรือหลายหน่วยเสียง ตามการออกเสียงของอักขระนั้น

  วิธีการอินพุต Wu Bi (Five Strike)
  วิธีการอินพุตตามการแสดงภาพของอักขระ ภาษาจีน ตามวิธีการอินพุตตามภาพ WuBi อักขระภาษาจีน มีการจัดประเภทเป็นสามระดับคือ stroke, ราก และอักขระเดี่ยว

  Zheng Ma
  วิธีการอินพุตตามการแสดงภาพของคำ ภาษาจีน

วิธีการอินพุต Biao Xing Ma

วิธีการอินพุตซึ่งมีการแบ่งอักขระภาษาจีนออกเป็นหลายส่วนประกอบ หรือ *ราก* เมื่อเข้ารหัสอักขระ รากเหล่านี้มีการแสดงด้วยตัวอักษรภาษาอังกฤษที่สอดคล้องกัน

เมธอด Code Input ภายใน

เมธอด Code Input ตามตารางโค้ดที่กำหนดไว้ใน GB18030 (ข้อมูลจำเพาะโค้ดภายในภาษาจีน) และ UCS-2 (ระบบ Unicode เวอร์ชัน 2)

- อินพุตอักขระครึ่งความกว้างและเต็มความกว้าง สนับสนุนอักขระ ASCII ทั้งในโหมดไบต์เดียวและมัลติไบต์
- หน้าต่างเสริมเพื่อสนับสนุนรายการตัวเลือกทั้งหมด ตัวอย่างเช่น Intelligent ABC จะสร้างรายการของอักขระที่เป็นไปได้ซึ่งมี สัญลักษณ์เสียงเหมือนกัน (*ราก*) ผู้ใช้เลือกอักขระที่ต้องการ โดยการกดคีย์การแปลง
- พื้นที่การวาดการแก้ไขก่อนหน้าเฉพาะจุด พื้นที่แสดงภาพย้อนกลับของ การป้อนรากซึ่งแสดงขึ้นชั่วคราวบนบรรทัดข้อความ อักขระที่สมบูรณ์จะถูกส่งไปยังตัวแก้ไขโดยการกดคีย์การแปลง

ไฟล์ UCS-ZIM อยู่ในไดเร็กทอรี **/usr/lib/nls/loc**

คีย์แม็พ UCS-ZIM อยู่ในไดเร็กทอรี **/usr/lib/nls/loc/ZH_CN.UTF-8.imkeymap**

การประมวลผลอักขระภาษาจีน (CJK)

UCS-ZIM มีการเรียกใช้โดยการกดคีย์วิธีการอินพุต คีย์ใดคีย์หนึ่ง สัญลักษณ์รากศัพท์หรือการออกเสียงแต่ละสัญลักษณ์จะถูกกำหนดให้กับคีย์หนึ่ง ผู้ใช้ อินพุตสัญลักษณ์รากศัพท์หรือการออกเสียงในพื้นที่การแก้ไขก่อนหน้าเฉพาะ จุด สำหรับวิธีการอินพุตโค้ดภายใน อักขระจะถูกสร้างขึ้นเมื่อ กดคีย์สุดท้าย วิธีการอินพุตอื่นสร้างรายการของตัวเลือก ที่แสดงขึ้นในหน้าต่าง ผู้ใช้เลือกอักขระที่ต้องการโดย การเลือกหมายเลขตัวเลือก อินพุตที่ไม่ถูกต้องจะทำให้เกิดเสียงบีปและ ข้อความแสดงข้อผิดพลาด Glyphs สามารถอินพุตได้โดยใช้วิธีการอินพุต ABC

Single-byte input method (SIM)

SIM เป็นมาตรฐานที่สนับสนุนโลแคลส่วนใหญ่ SIM คือฟังก์ชันการแม็พที่สนับสนุน การเรียบเรียงแบบง่ายที่กำหนดบนคีย์บอร์ดเวิร์คสเตชันซึ่งเชื่อมโยงกับ โลแคลไบต์เดียว SIM สนับสนุนคีย์บอร์ด ชุดโค้ด และภาษา ที่คำสั่ง **keycomp** สามารถอธิบายได้ คุณสามารถกำหนด SIM เองโดยใช้ imkeymaps สตริงที่เข้ารหัสซึ่งส่งคืนโดยวิธีการอินพุตขึ้นอยู่กับ imkeymap

โลแคลไบต์เดียวส่วนใหญ่แบ่งใช้หนึ่ง SIM คุณลักษณะของ SIM มี ดังนี้:

- สนับสนุนการแม็พคีย์บอร์ด 101-คีย์และ 102-คีย์
- สนับสนุนการเรียบเรียง Alt-Numpad

  เมื่อคุณกดคีย์ Alt วิธีการอินพุตจะเรียบเรียงอักขระโดยการใช้คีย์ตัวเลขที่กดสามคีย์ ถัดไป คีย์ตัวเลขสามคีย์แสดงถึงการเข้ารหัสฐานสิบ ของอักขระ ตัวอย่างเช่น การป้อนลำดับ XK_0, XK_9, XK_7 จะแม็พกับอักขระ *a* (097)
- สนับสนุนคำสั่ง Num-Lock สำหรับแผงแป้นตัวเลข
- สนับสนุนการเรียบเรียง diacritical

  คีย์ e-umlaut เป็นตัวอย่าง ของการเรียบเรียง diacritical เมื่อต้องการเรียบเรียง e-umlaut ผู้ใช้จะกดคีย์ diacritical ที่เหมาะสม (umlaut) ตามด้วยคีย์ตัวอักษร (e) ชุดเฉพาะของคีย์ diacritical ที่ใช้ขึ้นอยู่กับคำนิยามโลแคล และคีย์บอร์ด เมื่อมีช่องว่างตามหลังคีย์ diacritical จะมีการส่งคืนอักขระ diacritical ที่คีย์แสดงถึง ถ้าอักขระนั้นมีอยู่ในชุดโค้ด ของโลแคล
- ไม่ต้องใช้ฟังก์ชัน callback
- ตั้งอยู่ในไฟล์ **/usr/lib/nls/loc/sbcs.im** วิธีการอินพุต แบบท้องถิ่นอื่นส่วนใหญ่มีการแสดงเป็นชื่อย่อในไฟล์นี้

คีย์แม็พ

ส่วนนี้อธิบาย keymaps ที่สนับสนุนโดย single-byte input method (SIM)

คีย์แม็พที่ใช้ใน SIM มีดังต่อไปนี้:

- cs_CZ.ISO8859-2.imkeymap
- da_DK.ISO8859-1.imkeymap
- de_CH.ISO8859-1.imkeymap
- de_DE.ISO8859-1.imkeymap
- en_GB.ISO8859-1.imkeymap
- en_GB.ISO8859-1@alt.imkeymap
- en_US.ISO8859-1.imkeymap
- es_ES.ISO8859-1.imkeymap
- Et_EE.IBM-922 - imkeymap
- pl_PL.ISO8859-2@alt.imkeymap
- sq_AL.ISO8859-1.imkeymap
- fi_FI.ISO8859-1.imkeymap
- fi_FI.ISO8859-1@alt.imkeymap
- fr_BE.ISO8859-1.imkeymap
- fr_CA.ISO8859-1.imkeymap
- fr_CH.ISO8859-1.imkeymap
- fr_FR.ISO8859-1.imkeymap
- fr_FR.ISO8859-1@alt.imkeymap
- hr_HR.ISO8859-2.imkeymap
- hu_HU.ISO8859-2.imkeymap
- is_IS.ISO8859-1.imkeymap
- it_IT.ISO8859-1.imkeymap
- it_IT.ISO8859-1@alt.imkeymap
- nl_BE.ISO8859-1.imkeymap
- nl_NL.ISO8859-1.imkeymap
- no_NO.ISO8859-1.imkeymap
- pl_PL.ISO8859-2.imkeymap
- pt_BR.ISO8859-1.imkeymap
- pt_PT.ISO8859-1.imkeymap
- ro_RO.ISO8859-2.imkeymap
- sh_SP.ISO8859-2.imkeymap
- sl_SI.ISO8859-2.imkeymap

- sk_SK.ISO8859-2.imkeymap
- sv_SE.ISO8859-1.imkeymap
- sv_SE.ISO8859-1@alt.imkeymap
- tr_TR.ISO8859-1.imkeymap

Keysyms ที่สงวนไว้

Keysyms ต่อไปนี้ใช้เฉพาะกับวิธีการอินพุตนี้และ มีการอธิบายไว้ในไฟล์ **/usr/include/X11/aix_keysym.h**

Keysym	ค่า Hex
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdieresis	0x180007ae

ตัวดัดแปลง

ส่วนนี้อธิบายโมดิฟายเออร์ที่สนับสนุนโดย single-byte input method (SIM)

ตัวดัดแปลงที่ใช้ใน SIM มีดังต่อไปนี้:

โมดิฟายเออร์	ค่า Hex
ShiftMask	0x01
LockMask	0x02
ControlMask	0x04
Mod1Mask (Left-Alt)	0x08
Mod2Mask (Right-Alt)	0x10
Mod5Mask (Num Lock)	0x80

วิธีการอินพุตภาษาจีนดั้งเดิม (TIM)

ส่วนนี้อธิบายวิธีอินพุตภาษาจีนดั้งเดิม (TIM)

ชุดโค้ดภาษาจีนดั้งเดิมประกอบด้วยกลุ่มอักขระ ดังต่อไปนี้:

- ASCII (ภาษาอังกฤษ)
- อักขระภาษาจีนดั้งเดิม

ชุดอักขระภาษาจีนดั้งเดิมมีมากกว่า 100,000 อักขระ แต่มีการใช้งานบ่อยประมาณ 5000 อักขระเท่านั้น แต่ละอักขระประกอบด้วยส่วนประกอบหนึ่งถึงห้าส่วนที่เรียกว่า *ราก*

การออกเสียงของภาษาจีนดั้งเดิมมีการแสดงโดยใช้สัญลักษณ์การออกเสียง ที่เรียกว่า Dsu-Yin หรือ Bo-Po-Mo-Fo สัญลักษณ์การออกเสียงมีอยู่ 37 รายการ และตัวบ่งชี้ทำนองเสียงสี่ตัว อักขระภาษาจีนมีการแสดง โดยใช้สัญลักษณ์การออกเสียงหนึ่งถึงสามรายการ อักขระสามารถมีสัญลักษณ์ทำนองเสียงได้หนึ่ง รายการ การละเว้นสัญลักษณ์ทำนองเสียงจะตีความเป็นการเน้นทำนองเสียง ที่ห้า

คุณลักษณะ TIM

ส่วนนี้อธิบายคุณลักษณะวิธีการอินพุตภาษาจีนดั้งเดิม (TIM)

คุณลักษณะ TIM มีลักษณะดังต่อไปนี้:

- วิธีการอินพุตที่ใช้มีดังต่อไปนี้:

  Tsang-Jye
  สนับสนุนรากที่จะสร้างอักขระ ใช้บ่อยที่สุด โดยเจ้าหน้าที่ป้อนข้อมูล

  Tsang-Jye แบบง่าย
  สนับสนุนอินพุต wildcard และราก และสามารถป้อนอักขระเพียงบางส่วน ได้ด้วย

  สัญลักษณ์การออกเสียง
  อินพุตอักขระตามการออกเสียงของอักขระนั้น

  โค้ดภายใน
  สร้างอักขระโดยใช้อินพุตจุดโค้ดฐานสิบหก EUC

  ค่าฐานสิบ
  สร้างอักขระโดยใช้ค่าฐานสิบ สามารถเรียกใช้จากโหมดอินพุต ใดก็ได้
- อินพุตอักขระครึ่งความกว้างและเต็มความกว้าง สนับสนุนอักขระ ASCII ทั้งในโหมดไบต์เดียวและหลายไบต์
- อินพุตอักขระที่ระบบกำหนดและที่ผู้ใช้กำหนดได้
- หน้าต่างเสริมเพื่อสนับสนุนรายการตัวเลือกทั้งหมด วิธีการอินพุต Tsang-Jye แบบง่ายและการออกเสียงสร้างรายการของตัวเลือกอักขระ ที่มีรากอินพุตหรือสัญลักษณ์เสียงเหมือนกัน ผู้ใช้เลือกอักขระโดยการกดหมายเลขที่สอดคล้องกัน
- พื้นที่การวาดการแก้ไขก่อนหน้าเฉพาะจุด พื้นที่แสดงภาพย้อนกลับของ การป้อนรากซึ่งแสดงขึ้นชั่วคราวบนบรรทัดข้อความ อักขระที่สมบูรณ์ จะถูกส่งไปยังตัวแก้ไขโดยการกดคีย์การแปลง

ไฟล์ TIM อยู่ในไดเร็กทอรี **/usr/lib/nls/loc/TW.im**

คีย์แม็พ TIM อยู่ในไดเร็กทอรี **/usr/lib/nls/loc/zh_TW.IBM-eucTW.imkeymap**

การประมวลผลอักขระภาษาจีนดั้งเดิม

TIM มีการเรียกใช้โดยการกดคีย์วิธีการอินพุตคีย์ใดคีย์หนึ่ง สัญลักษณ์รากศัพท์หรือการออกเสียงแต่ละสัญลักษณ์จะถูกกำหนดให้กับคีย์หนึ่ง ผู้ใช้ อินพุตสัญลักษณ์รากศัพท์หรือการออกเสียงในพื้นที่การแก้ไขก่อนหน้าเฉพาะจุด สำหรับอินพุต Tsang-Jye และโค้ดภายใน อักขระจะถูกสร้างขึ้นเมื่อ กดคีย์การแปลง อินพุต Tsang-Jye แบบง่ายและการออกเสียง สร้างรายการของตัวเลือกที่จะแสดงขึ้นในหน้าต่าง ผู้ใช้เลือกอักขระที่ต้องการโดย การเลือกหมายเลขตัวเลือก อินพุตที่ไม่ถูกต้องจะทำให้เกิดเสียงบีปและ ข้อความแสดงข้อผิดพลาด

คีย์พิเศษต่อไปนี้สำหรับวิธีการอินพุตภาษาจีนดั้งเดิม มีการกำหนดบนคีย์บอร์ดภาษาจีนดั้งเดิม 106-คีย์

ตารางที่ 14. คีย์ภาษาจีนดั้งเดิมพิเศษ

ฟังก์ชันคีย์	Keysym	คำอธิบายฟังก์ชัน
คีย์ Tsang-Jye Shift	XK_Chinese _Tsangjei	เรียกใช้ทั้งวิธีการอินพุต Tsang-Jye และ Tsang-Jye แบบง่าย
คีย์ Phonetic Shift	XK_Chinese _Phonetic	เรียกใช้วิธีการอินพุตการออกเสียง
คีย์สลับครึ่ง/เต็มความกว้าง	XK_Chinese _Full_Half	สลับระหว่างครึ่งความกว้างและเต็มความกว้าง
คีย์การแปลง	XK_Convert	แปลงสัญลักษณ์รากและการออกเสียงหรือสัญลักษณ์โค้ด EUC เป็น อักขระ แสดงรายการตัวเลือกในหน้าต่างเสริม ถ้า ต้องการ
คีย์ไม่แปลง	XK_Non _Convert	ตีความสัญลักษณ์การออกเสียงเป็นอักขระ
คีย์ภาษาอังกฤษ/ตัวเลข	XK_Alph_Num	เรียกใช้โหมด ASCII
คีย์ ALT-Tsang-Jye Shift	XK_Internal _Code	เรียกใช้วิธีการอินพุตโค้ดภายใน
ALT บวกแผงแป้นตัวเลข		เรียกใช้วิธีการอินพุตค่าตัวเลข

วิธีการอินพุตสากล

วิธีอินพุตสากลถูกใช้ในโลแคล Unicode/UTF-8 เพื่อระบุการสนับสนุนวิธีอินพุตหลายภาษาทั้งหมด

คุณลักษณะของวิธีอินพุตสากลเป็นดังนี้:

- สนับสนุนการสลับวิธีอินพุต
  - การกดคีย์ `Ctrl` และ `Alt` ซ้ายและตัวอักษร `i` พร้อมกัน จะแสดงเมนู ที่แสดงรายการของวิธีการอินพุตอื่นที่มีอยู่ การเลือกวิธีการอินพุต จากรายการจะรีแม็พคีย์บอร์ดและโหลดวิธีการอินพุตที่กำหนด ซึ่งทำให้สามารถป้อนอักขระโดยใช้วิธีการอินพุตที่โหลดได้
- สนับสนุนอินพุตอักขระแบบชี้และคลิก
  - การกดคีย์ `Ctrl` และ `Alt` ซ้ายและตัวอักษร `l` พร้อมกัน จะแสดงเมนู ที่แสดงรายการหมวดหมู่ต่างๆ ของอักขระซึ่งมีอยู่ในมาตรฐาน Unicode การเลือกรายการอักขระจะแสดงเมตริกซ์ของอักขระที่มีอยู่ จากรายการ จากนั้น การคลิกบนอักขระที่กำหนดจะส่ง อักขระนั้นผ่านทางวิธีการอินพุตไปยังแอ็พพลิเคชัน
  - การกดคีย์ `Ctrl` และ `Alt` ซ้ายและตัวอักษร `c` จะทำให้กลับไปยังแอ็พพลิเคชัน หรือถ้า อยู่ในแอ็พพลิเคชันอยู่แล้ว จะทำให้กลับไปยังรายการอักขระที่ใช้ล่าสุดสำหรับ การป้อนอักขระแบบชี้และคลิก
- สนับสนุนชุดโค้ด UTF-8

คีย์แม็พ

นี่เป็นคีย์แม็พเฉพาะวิธีการอินพุตสากล

- XX_XX.UTF-8.imkeymap

Keysyms ที่สงวนไว้

ส่วนนี้อธิบาย keysyms ที่สงวนไว้โดย วิธีการอินพุต

Keysyms ที่แสดงรายการต่อไปนี้ถูกสงวนไว้สำหรับการใช้โดยวิธีการอินพุต:

Keysyms	การแทนค่าเลขฐานสิบหก
XK_dead_acute	0x180000b4
XK_dead_grave	0x18000060
XK_dead_circumflex	0x1800005e
XK_dead_diaeresis	0x180000a8
XK_dead_tilde	0x1800007e
XK_dead_caron	0x180001b7
XK_dead_breve	0x180001a2
XK_dead_doubleacute	0x180001bd
XK_dead_degree	0x180000b0
XK_dead_abovedot	0x180001ff
XK_dead_macron	0x180000af
XK_dead_cedilla	0x180000b8
XK_dead_ogonek	0x180001b2
XK_dead_accentdieresis	0x180007ae
XK_BunsetsuYomi	0x1800ff05
XK_MaeKouho	0x1800ff04
XK_ZenKouho	0x1800ff01
XK_KanjiBangou	0x1800ff02
XK_HenkanMenu	0x1800ff03
XK_LeftDouble	0x1800ff06
XK_RightDouble	0x1800ff07
XK_LeftPhrase	0x1800ff08
XK_RightPhrase	0x1800ff09
XK_ErInput	0x1800ff0a
XK_Reset	0x1800ff0b

หลักการที่เกี่ยวข้อง:

“วิธีอินพุต” ในหน้า 141
สำหรับแอ็พพลิเคชันที่จะรันในสภาวะแวดล้อมระหว่างประเทศที่ globalization เป็นพื้นฐาน จำเป็นต้องใช้วิธีการอินพุต วิธีการอินพุตคืออินเตอร์เฟสการเขียนโปรแกรมแอ็พพลิเคชัน (API) ที่ช่วยให้คุณสามารถจัดทำภาษา คีย์บอร์ด หรือชุดโค้ดเฉพาะที่ เป็นอิสระจากแอ็พพลิเคชัน

keysyms ที่เรียกใช้สำหรับภาษาจีนดั้งเดิม

ส่วนนี้แสดงรายการ Keysyms ที่สงวนไว้สำหรับภาษาจีน ดั้งเดิม

Keysyms	การแทนค่าเลขฐานสิบหก
XK_Full_Size	0xff42
XK_Phonetic	0xff48
XK_Alph_Num	0xaff50
XK_Non_Convert	0xaff52
XK_Convert	0xaff51
XK_Tsang_Jye	0xff47
XK_Internal_Code	0xff4a

ค่า keysyms ที่สงวนไว้สำหรับภาษาจีนประยุกต์ (ZIM และ ZIM-UCS)

ส่วนนี้แสดงรายการ keysyms ที่สงวนไว้สำหรับภาษา จีนประยุกต์ (ZIM และ ZIM-UCS)

Keysym	การแทนค่าเลขฐานสิบหก
XK_Alph_Num	0xaff47
XK_Non_Convert	0xaff59
XK_Row_Column	0xaff48
XK_PinYin	0xaff49
XK_English_Chinese	0xaff50
XK_ABC	0xaff51
XK_Fivestroke	0xaff62
XK_User-defined	0xaff56
XK_Legend	0xaff55
XK_ABC_Set_Option	0xaff60
XK_Half_full	0xaff53

ฟังก์ชันข้อความ

คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ การจัดระเบียบนี้ช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้ เพื่อช่วยในภารกิจนี้ ฟังก์ชันข้อความจึงนำเสนอคำสั่งและรูทีนย่อย ต่างๆ

ไฟล์ต้นฉบับข้อความที่มีข้อความแอ็พพลิเคชันมีการสร้างขึ้นโดยโปรแกรมเมอร์และถูกแปลงเป็นแค็ตตาล็อกข้อความ แอ็พพลิเคชันใช้แค็ตตาล็อกเหล่านี้เพื่อดึงข้อมูลและแสดงข้อความ ตามความต้องการ การแปลไฟล์ต้นฉบับข้อความเป็นภาษาอื่นแล้วแปลงไฟล์นั้น เป็นแค็ตตาล็อกข้อความไม่จำเป็นต้องเปลี่ยนและรีคอมไพล์โปรแกรม

หลักการที่เกี่ยวข้อง:

"การแยกข้อความออกจากโปรแกรม" ในหน้า 2
คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ ซึ่งจะช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความที่ถูกแปลพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้

"AIXwindowsรายการตรวจสอบ" ในหน้า 212
ไอเท็มรายการตรวจสอบที่เหลือเป็นไอเท็มเฉพาะสำหรับระบบ AIXwindows

"รายการตรวจสอบการดำเนินงานโปรแกรม" ในหน้า 207

การสร้างไฟล์ต้นฉบับข้อความ

ฟังก์ชันข้อความมีคำสั่งและรูทีนย่อยสำหรับ การดึงข้อมูลและแสดงข้อความโปรแกรมที่อยู่ในแค็ตตาล็อกข้อความ ภายนอกโปรแกรมเมอร์จะสร้างไฟล์ต้นฉบับข้อความที่มีข้อความแอ็พพลิเคชัน และแปลงไฟล์นั้นเป็นแค็ตตาล็อกข้อความโดยใช้คำสั่ง **gencat**

เมื่อต้องการสร้างไฟล์ต้นฉบับข้อความ ให้เปิดไฟล์โดยใช้ตัวแก้ไข ข้อความ ป้อนหมายเลขจำเพาะของข้อความหรือตัวระบุที่เป็นสัญลักษณ์ จากนั้น ป้อนข้อความดังแสดงในตัวอย่างต่อไปนี้:

```
1 message-text   $       (ข้อความนี้มีการกำหนดหมายเลข)
2 message-text   $       (ข้อความนี้มีการกำหนดหมายเลข)
OUTMSG message-text  $   (ข้อความนี้มีตัวระบุที่เป็นสัญลักษณ์ \
                          เรียกว่า OUTMSG)
4 message-text   $       (ข้อความนี้มีการกำหนดหมายเลข)
```

ข้อควรพิจารณาการใช้งาน

ส่วนนี้อธิบายข้อควรพิจารณาการใช้งาน

ให้พิจารณาข้อมูลดังต่อไปนี้:

- ต้องมีอักขระว่างเปล่าหนึ่งอักขระอยู่ระหว่างหมายเลข ID ข้อความ (หรือ ตัวระบุ) และข้อความข่าวสาร
- ตัวระบุที่เป็นสัญลักษณ์ต้องขึ้นต้นด้วยอักขระตัวอักษร และสามารถมีได้เฉพาะตัวอักษร ตัวเลขฐานสิบ และ ขีดเส้นใต้
- อักขระแรกของตัวระบุที่เป็นสัญลักษณ์ไม่สามารถเป็นตัวเลขได้
- ความยาวสูงสุดของตัวระบุที่เป็นสัญลักษณ์คือ 64 ไบต์
- หมายเลข ID ข้อความต้องมีการกำหนดในลำดับจากน้อยไปมาก ภายในชุดข้อความเดียว แต่หมายเลขไม่จำเป็นต้องต่อเนื่องกัน 0 (เลขศูนย์) ไม่ใช่ หมายเลข ID ข้อความที่ถูกต้อง
- หมายเลข ID ข้อความต้องมีการกำหนดเหมือนกับว่ามีการกำหนดหมายเลขให้กับ ตัวระบุที่เป็นสัญลักษณ์ซึ่งแทรกอยู่ด้วย ตัวอย่างเช่น ถ้าคุณกำหนดหมายเลข บรรทัดดังเช่นในตัวอย่างก่อนหน้านี้, `1`, `2`, `OUTMSG`, และ `3`, โปรแกรมจะมีข้อผิดพลาด เนื่องจากคำสั่ง **mkcatdefs** ยังกำหนดหมายเลขให้กับตัวระบุที่เป็นสัญลักษณ์ด้วย และจะกำหนด หมายเลข 3 ให้กับตัวระบุที่เป็นสัญลักษณ์ `OUTMSG`

หมายเหตุ: ตัวบ่งชี้ เชิงสัญลักษณ์เป็นค่าเฉพาะสำหรับโปรแกรมอำนวยความสะดวกข้อความ การใช้ได้หลายระบบ ของไฟล์ต้นฉบับข้อความอาจได้รับผลกระทบจากการใช้ตัวระบุที่เป็นสัญลักษณ์

การเพิ่มข้อคิดเห็นลงในไฟล์ต้นฉบับข้อความ

คุณสามารถรวมข้อคิดเห็นไว้ที่ใดก็ได้ในไฟล์ต้นฉบับข้อความ ยกเว้นเฉพาะภายในข้อความข่าวสาร ให้เว้นอย่างน้อยหนึ่งพื้นที่ว่างหรือหนึ่งแท็บ (ว่างเปล่า) หลังจาก $ (เครื่องหมายดอลลาร์)

ข้อมูลต่อไปนี้เป็นตัวอย่างของข้อคิดเห็น:

```
$ นี่เป็นข้อคิดเห็น
```

ข้อคิดเห็นไม่ปรากฏขึ้นในแค็ตตาล็อกข้อความที่สร้างขึ้นจาก ไฟล์ต้นฉบับข้อความ

ข้อคิดเห็นสามารถช่วยผู้พัฒนาในกระบวนการของการรักษาไฟล์ต้นฉบับ ข้อความ ช่วยผู้แปลในกระบวนการของการแปล และช่วยผู้เขียน ในกระบวนการของการแก้ไขและการจัดทำเอกสารเกี่ยวกับข้อความ ใช้ข้อคิดเห็นเพื่อ ระบุว่าตัวแปรต่างๆ เช่น **%s, %c,** และ **%d,** แสดงถึงอะไร ตัวอย่างเช่น สร้างหมายเหตุ ที่ระบุว่าตัวแปรอ้างอิงถึงผู้ใช้ ไฟล์ ไดเร็กทอรี หรือแฟล็ก นอกจากนี้ ยังสามารถใช้ข้อคิดเห็นเพื่อระบุข้อความที่ล้าสมัยได้อีกด้วย

เพื่อให้เกิดความชัดเจน คุณควรวางบรรทัดข้อคิดเห็นไว้ข้างใต้ต่อจากข้อความ ซึ่งข้อคิดเห็นนั้นอ้างอิง แทนที่จะวางไว้ที่ด้านล่างของแค็ตตาล็อก ข้อความ ข้อคิดเห็นโดยรวมสำหรับทั้งชุดสามารถวางไว้ข้างใต้ต่อจาก คำสั่ง **$set**

ข้อความที่ต่อเนื่องบนบรรทัดถัดไป

ข้อความทั้งหมดที่ตามหลังพื้นที่ว่างต่อจากหมายเลขข้อความจะถูกรวมไว้ เป็นข้อความข่าวสาร จนสิ้นสุดบรรทัด ใช้อักขระ escape\(backslash) เพื่อทำให้ข้อความต่อเนื่องบนบรรทัด ต่อมา

เครื่องหมาย\(backslash) ต้องเป็นอักขระตัวสุดท้ายบนบรรทัดดังเช่นใน ตัวอย่างต่อไปนี้:

```
5 นี่เป็นข้อความที่เชื่อมโยงกับ \
ข้อความหมายเลข 5
```

บรรทัดทางกายภาพสองบรรทัดเหล่านี้กำหนดข้อความบรรทัดเดียว:

```
นี่เป็นข้อความที่เชื่อมโยงกับข้อความหมายเลข 5
```

หมายเหตุ: การใช้อักขระพื้นที่ว่างมากกว่าหนึ่งตัวต่อจากหมายเลขข้อความ หรือตัวระบุที่เป็นสัญลักษณ์เป็นลักษณะเฉพาะของฟังก์ชันข้อความ การใช้ได้หลายระบบ ของไฟล์ต้นฉบับข้อความอาจได้รับผลกระทบจากการใช้มากกว่าหนึ่ง พื้นที่ว่าง

การรวมอักขระพิเศษในข้อความข่าวสาร

สามารถใช้เครื่องหมาย\(backslash) เพื่อแทรกอักขระพิเศษ เข้าในข้อความข่าวสาร

อักขระพิเศษเหล่านี้มีดังนี้:

อักขระพิเศษ	คำอธิบาย
\n	แทรกอักขระบรรทัดใหม่
\t	แทรกอักขระแท็บแนวนอน
\v	แทรกอักขระแท็บแนวตั้ง
\b	แทรกอักขระ backspace
\r	แทรกอักขระปัดแคร่
\f	แทรกอักขระป้อนกระดาษ
\\	แทรกอักขระ\(backslash)
ddd	แทรกอักขระไบต์เดียวที่เชื่อมโยงกับค่าฐานแปด ซึ่งแสดงโดยตัวเลขฐานแปดที่ถูกต้อง *ddd* **หมายเหตุ:** สามารถ ระบุตัวเลขฐานแปดได้หนึ่ง สอง หรือสามตัวเลข อย่างไรก็ตาม คุณต้องรวม เลขศูนย์นำถ้าอักขระที่ตามหลังตัวเลขฐานแปดเป็น ตัวเลขฐานแปดที่ถูกต้องด้วย ตัวอย่างเช่น ค่าฐานแปดสำหรับ $ (เครื่องหมายดอลลาร์) คือ 44 เมื่อต้องการแสดง $5.00 ให้ใช้\0445.00 ไม่ใช่\445.00 หรือ 5 จะถูก แจงเป็นส่วนหนึ่งของค่าฐานแปด
\x*dd*	แทรกอักขระไบต์เดียวที่เชื่อมโยงกับค่าฐานสิบหก ซึ่งแสดงโดยตัวเลขฐานสิบหกที่ถูกต้อง *dd* สองรายการ คุณต้องใส่เลขศูนย์นำเพื่อหลีกเลี่ยงข้อผิดพลาดการแจง (ให้ดูหมายเหตุ เกี่ยวกับ*ddd*)
\x*dddd*	แทรกอักขระไบต์คู่ที่เชื่อมโยงกับค่าฐานสิบหก ซึ่งแสดงโดยตัวเลขฐานสิบหกที่ถูกต้อง *dddd* สี่รายการ คุณต้องใส่เลขศูนย์นำเพื่อหลีกเลี่ยงข้อผิดพลาดการแจง (ให้ดูหมายเหตุ เกี่ยวกับ*ddd*)

การกำหนดอักขระเพื่อคั่นข้อความข่าวสาร

คุณสามารถใช้คำสั่ง $quote ในไฟล์ต้นฉบับข้อความ เพื่อกำหนดอักขระสำหรับคั่นข้อความข่าวสารได้ อักขระ นี้ควรจะเป็นอักขระ ASCII

รูปแบบคือ:

```
$quote [อักขระ] [ข้อคิดเห็น]
```

ใช้อักขระที่ระบุในตำแหน่งก่อนหน้าและตามหลังข้อความข่าวสาร ในตัวอย่างต่อไปนี้ คำสั่ง **$quote** จะตั้งค่าอักขระคำพูด เป็น _ (ขีดเส้นใต้) จากนั้นปิดใช้งานอักขระนั้นก่อนหน้าข้อความล่าสุด ซึ่งมีเครื่องหมายคำพูด:

```
$quote _    ใช้ขีดเส้นใต้เพื่อคั่นข้อความข่าวสาร
$set MSFAC          ฟังก์ชันข้อความ - ตัวระบุที่เป็นสัญลักษณ์
SYM_FORM    _ตัวระบุที่เป็นสัญลักษณ์อาจประกอบด้วยตัวอักษรและตัวเลข อักขระ \
หรือ \_ (อักขระขีดเส้นใต้)\n_
SYM_LEN     _ตัวระบุที่เป็นสัญลักษณ์สามารถยาวได้สูงสุดถึง 65 \
อักขระ \n_
5               _คุณสามารถผสมตัวระบุที่เป็นสัญลักษณ์และตัวเลขได้ \n_
$quote
MSG_H      โปรดจำไว้ว่าต้องรวมไฟล์ _msg_h_ ไว้ในโปรแกรมของคุณ\n
```

คำสั่ง **$quote** ล่าสุดในตัวอย่างก่อนหน้านี้จะปิดใช้งาน อักขระขีดเส้นใต้

ในตัวอย่างต่อไปนี้ คำสั่ง **$quote** จะกำหนด " (เครื่องหมายคำพูดคู่) เป็นอักขระคำพูด อักขระคำพูด ต้องเป็นอักขระที่ไม่ว่างเปล่าตัวแรกต่อจากหมายเลขข้อความ ข้อความใดๆ ที่ต่อจากอักขระคำพูดถัดไปจะถูกละเว้นไป

```
$quote "    ใช้อักขระคำพูดคู่เพื่อคั่นข้อความข่าวสาร
$set 10             ฟังก์ชันข้อความ - ข้อความคำสั่งคำพูด
1   "ใช้คำสั่ง $quote เพื่อกำหนดอักขระ \
\n สำหรับคั่นข้อความข่าวสาร"
```

```
2   "คุณสามารถรวมอักขระ \"คำพูด\" ในข้อความ \n \
โดยการวาง \\ ไว้ข้างหน้าข้อความ"
3   คุณสามารถรวมอักขระ "คำพูด" ในข้อความ \n \
โดยให้มีอักขระอื่นเป็นอักขระที่ไม่ว่างเปล่า \
\n ตัวแรกหลังจากหมายเลข ID ข้อความ
$quote
4   คุณสามารถปิดใช้งานกลไกอักขระคำพูดโดย \n \
ใช้คำสั่ง $quote โดยไม่มีอักขระ \n\
หลังจากนั้น
```

ตัวอย่างก่อนหน้านี้สาธิตวิธีการรวมอักขระคำพูดใน ข้อความข่าวสารสองวิธี:

- ใส่\(backslash) ข้างหน้าอักขระคำพูด
- ใช้อักขระอื่นบางตัวเป็นอักขระที่ไม่ว่างเปล่าตัวแรกต่อจาก หมายเลขข้อความ การทำเช่นนี้จะปิดใช้งานอักขระคำพูด เฉพาะสำหรับข้อความนั้น เท่านั้น

ตัวอย่างก่อนหน้านี้ยังแสดงดังต่อไปนี้:

- เครื่องหมาย\(backslash) ยังคงเป็นสิ่งจำเป็นสำหรับการแบ่งข้อความคำพูดบน บรรทัด
- เมื่อต้องการแสดง\(backslash) ในข้อความ ให้ใส่\(backslash) อีกเครื่องหมายหนึ่งไว้ข้างหน้า
- คุณสามารถจัดรูปแบบข้อความด้วยอักขระบรรทัดใหม่ได้โดยใช้ **\n**
- การใช้คำสั่ง **$quote** โดยไม่มีอาร์กิวเมนต์อักขระจะปิดใช้งาน กลไกอักขระคำพูด

การกำหนดหมายเลขชุดข้อความและหมายเลข ID ข้อความ

ชุดข้อความทั้งหมดต้องมีหมายเลขชุดหรือตัวระบุที่เป็นสัญลักษณ์

ใช้คำสั่ง **$set** ในไฟล์ต้นฉบับเพื่อกำหนดหมายเลขหรือตัวระบุให้กับ กลุ่มของข้อความ:

$set *n* [*ข้อคิดเห็น*]

หมายเลขชุดข้อความมีการระบุโดยค่าของ *n*, ตัวเลขระหว่าง 1 และ NL_SETMAX แทนที่จะใช้ตัวเลข คุณ สามารถใช้ตัวระบุที่เป็นสัญลักษณ์ได้ ข้อความทั้งหมดหลังจากคำสั่ง **$set** จะถูกกำหนดให้กับหมายเลขชุดนั้น จนกว่าจะมีคำสั่ง **$set** ถัดไป หมายเลขชุดดีฟอลต์คือ 1 หมายเลขชุดต้องมีการกำหนดในลำดับจากน้อยไปมาก แต่หมายเลขไม่จำเป็นต้องต่อเนื่องกัน ระบบจะสร้างชุดที่ว่างสำหรับหมายเลข ที่ข้ามไป อย่างไรก็ตาม ลำดับหมายเลขที่มีช่องว่างมากอาจทำให้ประสิทธิผลและ ประสิทธิภาพการทำงานลดลงได้ ยิ่งไปกว่านั้น การใช้มากกว่าหนึ่งหมายเลขชุดในหนึ่งแค็ตตาล็อก ไม่ได้ช่วยให้ประสิทธิภาพดีขึ้น

คุณยังสามารถรวมข้อคิดเห็นไว้ในคำสั่ง **$set** ได้ด้วย ดังนี้:

```
$set 10  ข้อความแสดงข้อผิดพลาดการสื่อสาร

$set OUTMSGS  ข้อความแสดงข้อผิดพลาดเอาต์พุต
```

ชุดข้อความ AIX จำนวนมากมีตัวบ่งชี้ที่เป็นสัญลักษณ์ในรูปแบบ **MS**_*PROG* โดยที่ **MS** แทนชุดข้อความ และ *PROG* เป็นชื่อของโปรแกรมหรือยูทิลิตีที่เกี่ยวข้องกับชุดข้อความ ตัวอย่างเช่น:

```
$set MS_WC  ชุดข้อความสำหรับยูทิลิตี้ wc

$set MS_XLC1  ชุดข้อความ 1 สำหรับคอมไพเลอร์ C For AIX

$set MS_XLC2  ชุดข้อความ 2 สำหรับคอมไพเลอร์ C For AIX
```

การลบข้อความออกจากแค็ตตาล็อก

ส่วนนี้อธิบายวิธีการลบข้อความออกจากแค็ตตาล็อก โดยใช้คำสั่ง **$delset**

คำสั่ง $delset จะลบข้อความทั้งหมดซึ่งอยู่ในชุดที่ระบุ ออกจากแค็ตตาล็อกที่มีอยู่:

```
$delset n [ comment
```

ชุดข้อความมีการระบุโดย *n* ต้องวางคำสั่ง **$delset** ในลำดับ หมายเลขชุดที่เหมาะสมกับคำสั่ง **$set** ในไฟล์ต้นฉบับเดียวกัน คุณยังสามารถรวมข้อคิดเห็นในคำสั่ง **$delset** ได้ด้วย

ความยาวของข้อความ

ส่วนนี้อธิบายความยาวสูงสุดในการแสดงผลของ ข้อความ

คำสั่ง **$len** กำหนดความยาวสูงสุดในการแสดงผลของ ข้อความ:

```
$len [n [ ข้อคิดเห็น ] ]
```

ถ้าไม่ได้ระบุ *n* หรือถ้าไม่ได้รวมคำสั่ง **$len** การแสดงข้อความจะถูกกำหนดเป็นค่า **NL_TEXTMAX** ความยาวในการแสดงข้อความคือจำนวนไบต์สูงสุดที่ใช้ได้ สำหรับข้อความ การระบุคำสั่ง **$len** ในเวลาต่อมา จะบันทึกทับการระบุก่อนหน้านี้ ค่าของ n ต้องไม่ เกินกว่าค่า **NL_TEXTMAX**

เนื้อหาของข้อความข่าวสาร

เมื่อใดก็ตามที่เป็นไปได้ ควรแจ้งให้ผู้ใช้ทราบอย่างชัดเจนว่าเกิดอะไรขึ้น และผู้ใช้สามารถแก้ไขสถานการณ์นั้นได้อย่างไร

ตัวอย่างต่อไปนี้แสดงให้เห็นว่าข้อมูลสาเหตุและการแก้ไขสามารถช่วย ปรับปรุงข้อความได้อย่างไร:

```
ข้อความดั้งเดิม: อาร์กิวเมนต์ไม่ถูกต้อง
ข้อความที่แก้ไข: ระบุปีเป็นค่าระหว่าง 1 และ 9999
```

ข้อความ อาร์กิวเมนต์ไม่ถูกต้อง ไม่ได้ช่วยผู้ใช้มากนัก ในขณะที่ข้อความ ห้ามระบุมากกว่า 2 ไฟล์บน บรรทัดคำสั่ง บอกผู้ใช้อย่างชัดเจนว่าพวกเขาต้องทำอะไรเพื่อให้ คำสั่งทำงานได้ ในลักษณะคล้ายกัน ข้อความ บรรทัดยาวเกินไป ไม่ได้ ให้ข้อมูลวิธีแก้ไขแก่ผู้ใช้ ในขณะที่ข้อความ บรรทัดต้องยาว ไม่เกิน 20 อักขระ ช่วยให้ข้อมูลที่ขาดหายไป

ตัวอย่างของไฟล์ต้นฉบับข้อความ

หัวข้อนี้จะแสดงตัวอย่างของไฟล์ต้นฉบับข้อความ

ตัวอย่างไฟล์ต้นฉบับข้อความต่อไปนี้ใช้หมายเลขสำหรับหมายเลข ID ข้อความและสำหรับหมายเลขชุดข้อความ:

```
$ นี่เป็นตัวอย่างไฟล์ต้นฉบับข้อความ
$ กำหนดอักขระคำพูด
$quote "
$set 1 นี่เป็นชุดที่ 1 ของข้อความ
1 "ไฟล์ที่ระบุไม่มีสิทธิการอ่าน\n"
2 "ไฟล์ %1$s และไฟล์ %2$s เหมือนกัน\n"
3 "Hello world!\n"
$กำหนดอักขระคำพูด
$quote ’
$set 2 นี่เป็นชุดที่ 2 ของข้อความ
1       ’fieldef: ไม่สามารถเปิด %1$s \n’
2       ’Hello world\n’
```

ตัวอย่างไฟล์ต้นฉบับข้อความต่อไปนี้ใช้ตัวระบุที่เป็นสัญลักษณ์สำหรับหมายเลข ID ข้อความและสำหรับหมายเลขชุดข้อความ:

```
$ นี่เป็นตัวอย่างไฟล์ต้นฉบับข้อความ
$ กำหนดอักขระคำพูด
$quote "
$set MS_SET1 นี่เป็นชุดที่ 1 ของข้อความ
MSG_1   "ไฟล์ที่ระบุไม่มีสิทธิการอ่าน\n"
MSG_2   "ไฟล์ %1$s และไฟล์ %2$s เหมือนกัน\n"
MSG_3   "Hello world\n"
$กำหนดอักขระคำพูด
$quote
$set 2 นี่เป็นชุดที่ 2 ของข้อความ
$EMSG_1 'fieldef: ไม่สามารถเปิด %1$s\n'
$EMSG_2 'Hello world!\n'
```

ตัวอย่างต่อไปนี้แสดงให้เห็นว่าตัวระบุที่เป็นสัญลักษณ์ทำให้สามารถเข้าใจ ข้อกำหนดจำเพาะของข้อความได้มากขึ้นได้อย่างไร:

```
catgets(cd, 1, 1, "ข้อความดีฟอลต์")

catgets(cd, MS_SET1, MSG_1, "ข้อความดีฟอลต์")
```

การสร้างแค็ตตาล็อกข้อความ

ฟังก์ชันข้อความมีคำสั่งและรูทีนย่อยสำหรับ การดึงข้อมูลและแสดงข้อความโปรแกรมที่อยู่ในแค็ตตาล็อกข้อความ ภายนอก โปรแกรมเมอร์จะสร้างไฟล์ต้นฉบับข้อความที่มีข้อความแอ็พพลิเคชัน และแปลงไฟล์นั้นเป็นแค็ตตาล็อกข้อความ การแปลไฟล์ต้นฉบับข้อความ เป็นภาษาอื่น แล้วแปลงไฟล์นั้นเป็น แค็ตตาล็อกข้อความไม่จำเป็นต้องเปลี่ยนหรือรีคอมไพล์โปรแกรม

เมื่อต้องการสร้างแค็ตตาล็อกข้อความ ให้ประมวลผลไฟล์ต้นฉบับข้อความที่เสร็จสมบูรณ์แล้ว ของคุณโดยใช้คำสั่ง **gencat** ของฟังก์ชันข้อความ คำสั่งนี้ สามารถใช้ในวิธีดังต่อไปนี้:

- ใช้คำสั่ง **gencat** เพื่อประมวลผลไฟล์ต้นฉบับข้อความ ที่มีหมายเลขชุด หมายเลข ID ข้อความ และข้อความข่าวสาร ไฟล์ต้นฉบับ ข้อความซึ่งมีตัวระบุที่เป็นสัญลักษณ์ไม่สามารถประมวลผลด้วยคำสั่ง gencat โดยตรง ตัวอย่างต่อไปนี้ใช้ข้อมูลในไฟล์ต้นฉบับข้อความ **x.msg** เพื่อสร้างไฟล์แค็ตตาล็อก:

  ```
  gencat x.cat x.msg
  ```

- ใช้คำสั่ง **mkcatdefs** เพื่อประมวลผลไฟล์ต้นฉบับข้อความซึ่งมีตัวระบุที่เป็นสัญลักษณ์เตรียมไว้ก่อน จากนั้น piped ไฟล์ที่ได้กับคำสั่ง **gencat** คำสั่ง **mkcatdefs** จะจัดทำ ไฟล์ *SymbolName*_**msg.h** ที่มีข้อความ คำนิยาม ข้อความเหล่านี้เข้าสมการตัวระบุที่เป็นสัญลักษณ์กับหมายเลขชุด และหมายเลข ID ข้อความซึ่งกำหนดโดยคำสั่ง **mkcatdefs** ไฟล์ *SymbolName*_**msg.h** ควรถูกรวมไว้ในโปรแกรม ซึ่งใช้ตัวระบุที่เป็นสัญลักษณ์เหล่านี้ คำสั่ง **mkcatdefs** เป็นคำสั่งเฉพาะของ AIX ตัวอย่างต่อไปนี้ใช้ข้อมูลในไฟล์ต้นฉบับข้อความ **x.msg** เพื่อ สร้างไฟล์ส่วนหัว **x_msg.h**:

  ```
  mkcatdefs x x.msg
  ```

- ใช้คำสั่ง **runcat** เพื่อ ประมวลผลไฟล์ต้นฉบับซึ่งมีตัวระบุที่เป็นสัญลักษณ์โดยอัตโนมัติ คำสั่ง **runcat** จะเรียกใช้คำสั่ง **mkcatdefs** และ pipes เอาต์พุตกับคำสั่ง **gencat** คำสั่ง **runcat** เป็นคำสั่งเฉพาะของ AIX ตัวอย่างต่อไปนี้ใช้ข้อมูลใน ไฟล์ต้นฉบับข้อความ `x.msg` เพื่อสร้างไฟล์ส่วนหัว **x_msg.h** และไฟล์แค็ตตาล็อก **X.cat**:

  ```
  runcat x x.msg
  ```

ตัวอย่างก่อนหน้านี้เทียบเท่ากับตัวอย่างต่อไปนี้:

```
mkcatdefs x x.msg | gencat x.cat
```

ถ้ามีแค็ตตาล็อกข้อความที่มีชื่อซึ่งระบุโดยพารามิเตอร์ *CatalogFile* อยู่ คำสั่ง **gencat** จะดัดแปลงแค็ตตาล็อกตาม ข้อความในไฟล์ต้นฉบับข้อความ ถ้าแค็ตตาล็อกขอความ ไม่มีอยู่ คำสั่ง **gencat** จะสร้างไฟล์แค็ตตาล็อกที่มีชื่อ ซึ่งระบุโดยพารามิเตอร์ *CatalogFile*

คุณสามารถระบุหมายเลขของไฟล์ต้นฉบับข้อความได้ ไฟล์หลายไฟล์ จะมีการประมวลผลในลำดับที่คุณระบุ แต่ละไฟล์ต้นฉบับที่ต่อเนื่องมา จะดัดแปลงแค็ตตาล็อก ถ้าคุณไม่ได้ระบุไฟล์ต้นฉบับ คำสั่ง **gencat** จะยอมรับข้อมูลต้นฉบับข้อความจากอินพุตมาตรฐาน

คำนามหรือวลีคำนาม (เช่น: ใบรับรองไคลเอ็นต์)

หัวข้อนี้อธิบายการกำหนดขนาดแค็ตตาล็อกและแมโครที่ใช้ เพื่อกำหนดขนาด

สามารถปรับภาพของแค็ตตาล็อกข้อความเป็นขนาดใดก็ได้ จำนวนสูงสุด ของชุดในแค็ตตาล็อก ข้อความในแค็ตตาล็อก และไบต์ในข้อความ มีการกำหนดไว้ในไฟล์ **/usr/include/limits.h** โดยแมโคร ต่อไปนี้:

แมโคร	คำอธิบาย
NL_SETMAX	ระบุจำนวนสูงสุดของหมายเลขชุดที่สามารถระบุได้โดยคำสั่ง **$set** ถ้ามีการใช้งานเกินขีดจำกัด **NL_SETMAX** คำสั่ง **gencat** จะออกข้อความแสดงข้อผิดพลาดและไม่สร้าง หรืออัพเดตแค็ตตาล็อกข้อความ
NL_MSGMAX	ระบุจำนวนสูงสุดของหมายเลข ID ข้อความที่ระบบอนุญาต ให้ใช้ได้ ถ้ามีการใช้งานเกินขีดจำกัด **NL_MSGMAX** คำสั่ง **gencat** จะออกข้อความแสดงข้อผิดพลาดและไม่สร้าง หรืออัพเดตแค็ตตาล็อกข้อความ
NL_TEXTMAX	ระบุจำนวนไบต์สูงสุดที่ข้อความสามารถมีได้ ถ้ามีการใช้งานเกินขีดจำกัด **NL_TEXTMAX** คำสั่ง **gencat** จะออกข้อความแสดงข้อผิดพลาดและไม่สร้าง หรืออัพเดตแค็ตตาล็อกข้อความ

ตัวอย่าง

ส่วนนี้อธิบายอ็อพชันวิธีการสร้างแค็ตตาล็อกข้อความ จากไฟล์ต้นฉบับ

มักจะเริ่มต้นด้วยคำนิยามที่ตอบคำถาม "สิ่งนี้คือ อะไร?"

- ตัวอย่างนี้แสดงวิธีการสร้างแค็ตตาล็อกข้อความ จากไฟล์ต้นฉบับที่มีหมายเลขจำเพาะข้อความ ข้อมูลต่อไปนี้เป็น ข้อความของไฟล์ต้นฉบับข้อความ **hello.msg**:

  ```
  $ file: hello.msg
  $set 1    prompts
  1 Please, enter your name.
  2 Hello, %s \n
  $ end of file: hello.msg
  ```

  เมื่อต้องการสร้างแค็ตตาล็อกข้อความ **hello.cat** จากไฟล์ต้นฉบับ **hello.msg** ให้พิมพ์ดังนี้:

  ```
  gencat hello.cat hello.msg
  ```

- ตัวอย่างนี้แสดงวิธีการสร้างแค็ตตาล็อกข้อความ จากไฟล์ต้นฉบับที่มีการอ้างอิงสัญลักษณ์ ข้อมูลต่อไปนี้เป็นข้อความของไฟล์ต้นฉบับข้อความ **hello.msg** ที่มีการอ้างอิงสัญลักษณ์ในชุดข้อความและ ข้อความ:

  ```
  $ file: hello.msg
  $quote "
  $set PROMPTS
  PLEASE  "Please, enter your name."
  HELLO   "Hello, %s \n"
  $ end of file: hello.msg
  ```

  เมื่อต้องการประมวลผลไฟล์ต้นฉบับข้อความ **hello.msg** และ **msgerrs** ให้พิมพ์ดังนี้:

```
runcat hello hello.msg
runcat msgerrs msgerrs.msg /usr/lib/nls/msg/$LANG/msgerrs.cat
```

คำสั่ง **runcat** จะเรียกใช้คำสั่ง **mkcatdefs** และ **gencat** การเรียก ครั้งแรกไปยังคำสั่ง **runcat** จะใช้ไฟล์ต้นฉบับ **hello.msg** และใช้พารามิเตอร์ที่สอง, `hello`, เพื่อจัดทำแค็ตตาล็อกข้อความ **hello.cat** และไฟล์คำนิยาม **hello_msg.h**

ไฟล์คำนิยาม **hello_msg.h** มีชื่อที่เป็นสัญลักษณ์ สำหรับแค็ตตาล็อกข้อความและ IDs สัญลักษณ์สำหรับข้อความและชุดต่างๆ ชื่อที่เป็นสัญลักษณ์สำหรับแค็ตตาล็อกข้อความ **hello.cat** คือ `MF_HELLO` ชื่อนี้มีการจัดทำโดยอัตโนมัติโดยคำสั่ง **mkcatdefs**

การเรียก ที่สองไปยังคำสั่ง `runcat` จะใช้ไฟล์ต้นฉบับ msgerrs.msg และใช้พารามิเตอร์แรก, `msgerrs`, เพื่อจัดทำไฟล์คำนิยาม **msgerrs_msg.h**

เนื่องจากมี พารามิเตอร์ที่สาม, `/usr/lib/nls/msg/$LANG/msgerrs.cat`, คำสั่ง **runcat** จะใช้พารามิเตอร์นี้เป็น ชื่อไฟล์แค็ตตาล็อก พารามิเตอร์นี้เป็นชื่อพาธสมบูรณ์ที่ระบุอย่างชัดเจนว่า คำสั่ง **runcat** ต้องวางไฟล์ไว้ที่ใด ชื่อที่เป็นสัญลักษณ์สำหรับแค็ตตาล็อก `msgerrs.cat` คือ `MF_MSGERRS`

การแสดงข้อความภายนอกแอ็พพลิเคชันโปรแกรม

คำสั่งต่อไปนี้ช่วยให้คุณสามารถแสดงข้อความภายนอก แอ็พพลิเคชันโปรแกรม คำสั่งเหล่านี้เป็นคำสั่งเฉพาะของ AIX

คำสั่ง	คำอธิบาย
dspcat	แสดงข้อความที่มีอยู่ในแค็ตตาล็อกข้อความ ที่ระบุ ตัวอย่างต่อไปนี้แสดงข้อความที่มีอยู่ในไฟล์ต้นฉบับข้อความ **x.cat** : `dspcat x.cat`
dspmsg	แสดงข้อความเดียวจากแค็ตตาล็อกข้อความ ตัวอย่างต่อไปนี้ แสดงข้อความที่มีอยู่ในไฟล์ต้นฉบับข้อความ **x.cat** ซึ่งมีหมายเลข ID เป็น 1 และหมายเลขชุด เป็น 2: `dspmsg x.cat -s 2 1` คุณ สามารถใช้คำสั่ง **dspmsg** ใน shell scripts เมื่อต้องได้รับ ข้อความจากแค็ตตาล็อกข้อความ

การแสดงข้อความโดยใช้แอ็พพลิเคชันโปรแกรม

หัวข้อนี้จะอธิบายไอเท็มที่ต้องรวมไว้ในการเขียนโปรแกรมแอ็พพลิเคชัน ของคุณ

เมื่อเขียนโปรแกรมด้วยฟังก์ชันข้อความ คุณต้องรวมไอเท็มต่อไปนี้ ไว้ในแอ็พพลิเคชันโปรแกรมของคุณ:

- ไฟล์คำนิยาม *CatalogFile*_**msg.h** ที่สร้างขึ้นโดยคำสั่ง **mkcatdefs** หรือ **runcat** ถ้าคุณใช้ตัวระบุที่เป็นสัญลักษณ์ ในไฟล์ต้นฉบับข้อความ หรือไฟล์ **limits.h** และ **nl_types.h** ถ้าคุณไม่ได้ใช้ตัวระบุที่เป็นสัญลักษณ์
- การเรียกเพื่อเริ่มต้นสภาพแวดล้อมโลแคล
- การเรียกเพื่อเปิดแค็ตตาล็อก
- การเรียกเพื่ออ่านข้อความ
- การเรียกเพื่อแสดงข้อความ
- การเรียกเพื่อปิดแค็ตตาล็อก

รูทีนย่อยต่อไปนี้นำเสนอการบริการที่จำเป็นสำหรับการแสดง ข้อความโปรแกรมด้วยฟังก์ชันข้อความ:

รูทีนย่อย	คำอธิบาย
setlocale	ตั้งค่าโลแคล ระบุตัวแปรสภาพแวดล้อม **LC_ALL** ในการเรียกไปยังรูทีนย่อย **setlocale** สำหรับ ภาษาแค็ตตาล็อกข้อความที่ต้องการ
catopen	เปิดแค็ตตาล็อกข้อความที่ระบุและส่งคืน คำอธิบายแค็ตตาล็อก ซึ่งคุณใช้เพื่อดึงข้อมูลข้อความจาก แค็ตตาล็อก
catgets	ดึงข้อมูลข้อความจากแค็ตตาล็อกหลังจากการเรียกไปยัง รูทีนย่อย **catopen** เสร็จเรียบร้อยแล้ว
printf	แปลง จัดรูปแบบ และบันทึกลงในสตรีม stdout (เอาต์พุตมาตรฐาน)
catclose	ปิดแค็ตตาล็อกข้อความที่ระบุ

โปรแกรม C ต่อไปนี้, `hello`, สาธิตการเปิดแค็ตตาล็อก `hello.cat` โดยใช้รูทีนย่อย **catopen** การดึงข้อมูลข้อความ จาก แค็ตตาล็อกโดยใช้รูทีนย่อย **catgets** การแสดงข้อความ โดยใช้รูทีนย่อย **printf** และการปิดแค็ตตาล็อกโดยใช้รูทีนย่อย **catclose** .

```
/* program: hello */
#include <nl_types.h>
#include <locale.h>
nl_catd catd;
main()
{
/*  เริ่มต้นโลแคล  */
setlocale (LC_ALL, "");
/* เปิดแค็ตตาล็อก */
catd=catopen("hello.cat",NL_CAT_LOCALE);
printf(catgets(catd,1,1,"Hello World!"));
catclose(catd);                      /* ปิดแค็ตตาล็อก */
exit(0);
}
```

ในตัวอย่างก่อนหน้านี้ รูทีนย่อย **catopen** จะอ้างอิง แค็ตตาล็อกข้อความ **hello.cat** โดยใช้ชื่อไฟล์เท่านั้น ด้วยเหตุนี้ คุณต้อง ตรวจสอบให้แน่ใจว่าตัวแปรสภาพแวดล้อม **NLSPATH** มีการตั้งค่าอย่างถูกต้อง ถ้าแค็ตตาล็อกข้อความถูกเปิดโดยรูทีนย่อย **catopen** เสร็จเรียบร้อย แล้ว รูทีนย่อย **catgets** จะส่งคืนตัวชี้ไปยังข้อความที่ระบุ ในแค็ตตาล็อก `hello.cat` ถ้าไม่พบ แค็ตตาล็อกข้อความหรือข้อความไม่มีอยู่ในแค็ตตาล็อก รูทีนย่อย **catgets** จะส่งคืน `Hello World!` สตริง ดีฟอลต์

การทำความเข้าใจกับตัวแปรสภาพแวดล้อม NLSPATH

ตัวแปรสภาพแวดล้อม **NLSPATH** ระบุไดเร็กทอรี ในการค้นหาแค็ตตาล็อกข้อความ

รูทีนย่อย **catopen** จะค้นหาไดเร็กทอรีเหล่านี้ในลำดับที่ระบุเมื่อเรียกเพื่อระบุตำแหน่งและเปิดแค็ตตาล็อกข้อความ ถ้าไม่พบ แค็ตตาล็อกข้อความ รูทีนที่ดึงข้อมูลข้อความ จะส่งคืนข้อความดีฟอลต์ที่โปรแกรมจัดส่ง ให้ดูไฟล์ **/etc/environment** สำหรับ พาธดีฟอลต์ **NLSPATH**

การดึงข้อมูลข้อความดีฟอลต์ที่โปรแกรมจัดส่ง

รูทีนการดึงข้อมูลข้อความทั้งหมดจะส่งคืนข้อความดีฟอลต์ที่โปรแกรมจัดส่ง ถ้าไม่สามารถดึงข้อมูลข้อความที่ต้องการเนื่อง จาก เหตุผลใดๆ โดยทั่วไป ข้อความดีฟอลต์ที่โปรแกรมจัดส่งเป็นข้อความย่อ บรรทัดเดียวซึ่งไม่มีหมายเลขข้อความอยู่ในข้อ ความ ผู้ใช้ ที่ต้องการข้อความดีฟอลต์เหล่านี้สามารถตั้งค่าหมวดหมู่ **LC_MESSAGES** เป็นโลแคล C หรือยกเลิกการตั้งค่า ตัวแปรสภาพแวดล้อม **NLSPATH** เมื่อไม่มีการตั้งค่าตัวแปรสภาพแวดล้อม **LC_ALL**, **LC_MESSAGES**, หรือ **LANG** หมวด หมู่ **LC_MESSAGES** จะมีค่าดีฟอลต์เป็นโลแคล C

การตั้งค่าลำดับชั้นภาษา

ผู้ใช้หลายภาษาสามารถระบุลำดับชั้นภาษาสำหรับข้อความได้

เมื่อต้องการตั้งค่าลำดับชั้นภาษาสำหรับค่าดีฟอลต์ระบบหรือสำหรับ ผู้ใช้แต่ละราย ให้ดู "การเปลี่ยนสภาพแวดล้อมภาษา" ในหน้า 6 หรือใช้ SMIT เมื่อต้องการใช้ SMIT ให้ตั้งค่าลำดับชั้นภาษา พิมพ์พาธด่วน SMIT `smit mlang` ที่บรรทัดคำสั่ง

เลือก **เปลี่ยน / แสดงลำดับชั้นภาษา**

หรือ

ที่บรรทัดคำสั่ง ให้พิมพ์:

```
smit
```

เลือก **สภาพแวดล้อมระบบ**

เลือก **จัดการ สภาวะแวดล้อมภาษา**

เลือก **เปลี่ยน / แสดงลำดับชั้นภาษา**

ตัวอย่างของการดึงข้อมูลข้อความจากแค็ตตาล็อก

ตัวอย่างนี้มีสามส่วนคือ: ไฟล์ต้นฉบับข้อความ คำสั่งที่ใช้ในการสร้างไฟล์แค็ตตาล็อกข้อความ และตัวอย่างโปรแกรมที่ใช้แค็ตตาล็อกข้อความ

1. ตัวอย่างต่อไปนี้แสดงไฟล์ต้นฉบับข้อความ **example.msg** :

```
$quote "
$ ทุกแค็ตตาล็อกข้อความควรมีหมายเลขชุดเริ่มต้น
$set MS_SET1
MSG1 "Hello world\n"
MSG2 "Good Morning\n"
ERRMSG1 "ตัวอย่าง: 1000.220 สิทธิการอ่านถูกปฏิเสธสำหรับไฟล์
%s.\n"
$set MS_SET2
MSG3 "Howdy\n"
```

2. คำสั่งต่อไปนี้ใช้ไฟล์ต้นฉบับข้อความ **example.msg** เพื่อสร้าง ไฟล์ส่วนหัว **example.h** และไฟล์แค็ตตาล็อก **example.cat** ในไดเร็กทอรีปัจจุบัน:

```
runcat example example.msg
```

3. ตัวอย่างโปรแกรมต่อไปนี้ใช้ไฟล์ส่วนหัว **example.h** และ เข้าถึงไฟล์แค็ตตาล็อก **example.cat**:

```
#include <locale.h>
#include <nl_types.h>
#include "example_msg.h" /*contains definitions for symbolic
                           identifiers*/
main()
{
        nl_catd catd;
        int error;

        (void)setlocale(LC_ALL, "");
```

```
        catd = catopen(MF_EXAMPLE, NL_CAT_LOCALE);
        /*
        ** ได้รับหมายเลขข้อความ 1 จากชุดแรก
        */
        printf( catgets(catd,MS_SET1,MSG1,"Hello world\n") );

        /*
        ** ได้รับหมายเลขข้อความ 1 จากชุดที่สอง
        */
        printf( catgets(catd, MS_SET2, MSG3,"Howdy\n") );
        /*
        ** แสดงข้อความแสดงข้อผิดพลาด
        */
        printf( catgets(catd, MS_SET1, ERRMSG1,"example: 100.220
                สิทธิถูกปฏิเสธสำหรับการอ่านไฟล์ %s.\n") ,
                filename);
        catclose(catd);
}
```

การเขียนข้อความ

ส่วนนี้แสดงเทคนิคที่ช่วยทำให้ข้อความของคุณ มีความหมายและกระชับ

เทคนิคต่อไปนี้ช่วยทำให้ข้อความของคุณมีความหมายและกระชับ:

- แผนงานสำหรับการทำข้อความทั้งหมดให้เป็นโกลบอล รวมถึงข้อความ ที่แสดงบนพาเนล
- จัดให้มีพื้นที่ว่างที่เพียงพอสำหรับข้อความแปลซึ่งจะแสดงขึ้น ข้อความแปลมักใช้คอลัมน์การแสดงผลมากกว่าข้อความข่าวสาร ต้นฉบับ โดยทั่วไป ควรให้มีพื้นที่ว่างมากขึ้นประมาณ 20% ถึง 30% สำหรับข้อความแปล แต่ในบางกรณี คุณอาจต้องใช้พื้นที่ว่างมากขึ้นถึง 100% สำหรับข้อความแปล
- ใช้แค็ตตาล็อกข้อความเพื่อส่งออกข้อความผู้ใช้และข้อความแสดงข้อผิดพลาด แอ็พพลิเคชัน X สามารถใช้ไฟล์รีซอร์สเพื่อส่งออกข้อความสำหรับ แต่ละโลแคล
- แสดงข้อความดีฟอลต์
- ทำให้แต่ละข้อความในไฟล์ต้นฉบับข้อความเป็นเอนทิตี้สมบูรณ์ การสร้างข้อความโดยการนำส่วนประกอบมาต่อกันทำให้การแปลเป็นเรื่องยาก
- ใช้คำสั่ง **$len** ในไฟล์ต้นฉบับข้อความเพื่อควบคุม ความยาวในการแสดงผลสูงสุดของข้อความข่าวสาร (คำสั่ง **$len** เป็นลักษณะเฉพาะ ฟังก์ชันข้อความ)
- ใช้ตัวระบุที่เป็นสัญลักษณ์เพื่อระบุหมายเลขชุดและหมายเลข ข้อความ โปรแกรมควรอ้างอิงหมายเลขชุดและหมายเลขข้อความโดยใช้ ตัวระบุที่เป็นสัญลักษณ์ ไม่ใช่ใช้หมายเลขจริง (การใช้ ตัวระบุที่เป็นสัญลักษณ์เป็นลักษณะเฉพาะฟังก์ชันข้อความ)
- สนับสนุนการจัดลำดับใหม่ของประโยคย่อยโดยวิธีการกำหนดหมายเลขตัวแปร **%s** ซึ่งช่วยให้นักแปลสามารถจัดลำดับประโยคย่อยใหม่ได้ ถ้าต้องการ ตัวอย่างเช่น ถ้าโปรแกรมต้องการแสดงข้อความภาษาอังกฤษ: ไฟล์ **%s** มีการอ้างอิงใน **%s** โปรแกรม อาจจัดส่งสองสตริงดังนี้:

  ```
  printf(message_pointer, name1, name2)
  ```

  ข้อความ ภาษาอังกฤษกำหนดหมายเลขตัวแปร **%s** ดังนี้:

  ```
  ไฟล์ %1$s มีการอ้างอิงใน %2$s\n
  ```

ข้อความ แปลของข้อความนี้อาจเป็น:

```
%2$s มีการอ้างอิงถึงไฟล์ %1$s\n
```

- อย่าใช้ **sys_errlist[errno]** เพื่อให้ได้รับข้อความแสดงข้อผิดพลาด สิ่งนี้ทำลายการส่งออกข้อความ **sys_errlist[]** เป็น อาร์เรย์ของข้อความแสดงข้อผิดพลาดซึ่งแสดงในภาษาอังกฤษเท่านั้น ใช้ **strerror**(*errno*) เนื่องจากฟังก์ชันนี้ได้รับข้อความจากแค็ตตาล็อก
- อย่าใช้ **sys_siglist[signo]** เพื่อให้ได้รับข้อความแสดงข้อผิดพลาด สิ่งนี้ทำลายการส่งออกข้อความ **sys_siglist[]** เป็น อาร์เรย์ของข้อความแสดงข้อผิดพลาดซึ่งแสดงในภาษาอังกฤษเท่านั้น ใช้ **psignal( )**, เนื่องจากฟังก์ชันนี้ได้รับข้อความจาก แค็ตตาล็อก
- ใช้ฟังก์ชันข้อคิดเห็นเกี่ยวกับข้อความเพื่อช่วยในการรักษาและ การแปลข้อความ
- โดยทั่วไป สร้างไฟล์ต้นฉบับข้อความแยกต่างหากและแค็ตตาล็อก สำหรับข้อความที่ใช้กับแต่ละคำสั่งหรือยูทิลิตี้

การอธิบายไวยากรณ์คำสั่งในข้อความ

ส่วนนี้จะแสดงไวยากรณ์คำสั่งในคำสั่งการใช้งาน

- แสดงไวยากรณ์คำสั่งในคำสั่งการใช้งาน ตัวอย่างเช่น คำสั่งการใช้งานที่เป็นไปได้สำหรับคำสั่ง **rm** คือ:

  ```
  Usage: rm [-firRe] [--] File ...
  ```

- ใช้ตัวพิมพ์ใหญ่สำหรับอักษรตัวแรกของคำ เช่น File, Directory, String, และ Number ในข้อความคำสั่งการใช้งาน
- อย่าย่อพารามิเตอร์บนบรรทัดคำสั่ง ตัวอย่างเช่น `Num` ที่สะกด เป็น `Number` สามารถแปลได้ ง่ายขึ้น
- ใช้เฉพาะอักขระคั่นต่อไปนี้ในข้อความคำสั่งการใช้งาน:

ตัวคั่น	คำอธิบาย
`[]`	ใส่พารามิเตอร์ที่เป็นอ็อพชัน
`{}`	ใส่หลายพารามิเตอร์ซึ่งต้องใช้ตัวใดตัวหนึ่ง
`|`	คั่นพารามิเตอร์ที่ไม่สามารถเลือกทั้งสองตัวได้ ตัวอย่างเช่น **[a|b]** บ่งชี้ว่าคุณสามารถเลือก `a`, `b`, หรือไม่เลือกทั้ง `a` และ `b`; และ `{a|b}` บ่งชี้ว่าคุณต้องเลือก `a` หรือ `b`
`...`	ตามหลังพารามิเตอร์ที่ไม่สามารถทำซ้ำบนบรรทัดคำสั่ง หมายเหตุว่ามีพื้นที่ว่างก่อนหน้าจุดละ
`-`	บ่งชี้อินพุตมาตรฐาน

- อย่าใช้อักขระคั่นใดๆ สำหรับพารามิเตอร์บังคับซึ่งเป็นตัวเลือกเพียงตัวเดียว เท่านั้น ตัวอย่างเช่น:

  ```
  banner String
  ```

- ใส่อักขระพื้นที่ว่างระหว่างแฟล็กซึ่งต้องแบ่งแยกบน บรรทัดคำสั่ง ตัวอย่างเช่น:

  ```
  unget [-n] [-rSID] [-s] {File|-}
  ```

- อย่าแบ่งแยกแฟล็กที่สามารถใช้ร่วมกันได้บนบรรทัด คำสั่ง ตัวอย่างเช่น:

  ```
  wc [-cwl] {File ...|-}
  ```

- ใส่แฟล็กในลำดับตามตัวอักษรเมื่อลำดับของแฟล็กบน บรรทัดคำสั่งไม่ทำให้เกิดความแตกต่าง ใส่แฟล็กตัวพิมพ์เล็กก่อนแฟล็กตัวพิมพ์ใหญ่ ตัวอย่างเช่น:

  ```
  get -aAijlmM
  ```

- ใช้ดุลยพินิจของคุณในการกำหนดว่าควรจะจบบรรทัดในข้อความคำสั่งการใช้งาน ที่ใด ตัวอย่างต่อไปนี้แสดงความยาวของข้อความคำสั่งการใช้งาน:

  ```
  Usage: get [-e|-k] [-c Cutoff] [-i List] [-r SID] [-w String]
   [-x List] [-b] [-gmnpst] ...
  ```

  พิมพ์ข้อมูลการใช้งาน บนบรรทัดที่สองต่อไป ถ้าจำเป็น ตัวอย่างเช่น:

```
Usage: get [-e|-k] [-c Cutoff] [-i List] [-r SID] [-w String]
           [-x List] [-b] [-gmnpst] [-l[p]] File ...
```

สไตล์การเขียนข้อความ

การเขียนที่ชัดเจนช่วยในการแปลข้อความ แนวทางเกี่ยวกับสไตล์การเขียนข้อความต่อไปนี้รวมถึง คำศัพท์เฉพาะ การวรรคตอน อารมณ์ เสียง กาล การใช้ตัวพิมพ์ใหญ่ รูปแบบ และคำถามเกี่ยวกับการใช้งานอื่นๆ

- เขียนข้อความที่สั้นกระชับ ข้อความควรจะยาวหนึ่งประโยค
- ใช้รูปแบบประโยคสมบูรณ์
- เพิ่มคำกำกับนาม (a, an, the) เมื่อจำเป็นเพื่อไม่ให้คลุมเครือ
- ใช้ตัวพิมพ์ใหญ่ในคำแรกของประโยค และใช้เครื่องหมายจุดที่ตอนท้ายของประโยค
- ใช้กาลปัจจุบัน อย่าใช้กาลอนาคตในข้อความ ตัวอย่างเช่น ให้ใช้ประโยค:

  คำสั่ง `cal` แสดงปฏิทิน

  แทน:

  คำสั่ง `cal` จะแสดงปฏิทิน
- อย่าใช้บุรุษที่หนึ่ง (ฉันหรือเรา) ในข้อความ
- หลีกเลี่ยงการใช้บุรุษที่สอง (คุณ) ยกเว้นในวิธีใช้และข้อความโต้ตอบ
- ใช้รูปแบบแอ็คทีฟ ตัวอย่างต่อไปนี้แสดงวิธีการเปลี่ยนข้อความที่เขียนในรูปแบบพาสซีฟเป็นข้อความรูปแบบแอ็คทีฟ

  พาสซีฟ: เดือนและปีต้องถูกป้อนเป็นตัวเลข
  แอ็คทีฟ: ป้อนเดือนและปีเป็นตัวเลข
- ใช้รูปแบบคำสั่ง (วลีคำสั่ง) และกริยาแอ็คทีฟ เช่น ระบุ ใช้ ตรวจสอบ เลือก และรอ
- ระบุข้อความในเชิงบวก ตัวอย่างต่อไปนี้แสดงข้อความเชิงลบที่ทำให้เป็นเชิงบวกมากขึ้น

  เชิงลบ: ห้ามใช้อ็อพชัน `f` มากกว่าหนึ่งครั้ง
  เชิงบวก: ใช้แฟล็ก `-f` เพียงครั้งเดียวเท่านั้น
- ใช้คำเฉพาะในประเภททางไวยากรณ์ที่แสดงในพจนานุกรมเท่านั้น ถ้าคำมีการแสดงเป็นคำนามอย่างเดียว อย่าให้คำนั้นเป็นกริยา ตัวอย่างเช่น อย่า *solution* ปัญหา หรือ *architect* ระบบ
- อย่าใช้คำเสริมหน้าหรือคำเติมท้าย นักแปลอาจไม่ทราบว่าคำที่ขึ้นต้นด้วย re-, un-, in- หรือ non- หมายความว่าอะไร และการแปลข้อความที่ใช้ส่วนนำหน้าและส่วนต่อท้ายอาจไม่ได้มีความหมายตรงตามที่คุณตั้งใจไว้ ข้อยกเว้นของกฎนี้คือเมื่อคำเสริมหน้าเป็นส่วนประกอบรวมของคำที่ใช้กันโดยทั่วไป ตัวอย่างเช่น คำว่า `previous` และ `premature` สามารถใช้ได้ แต่คำว่า `nonexistent` ใช้ไม่ได้
- อย่าใช้วงเล็บเพื่อแสดงเอกพจน์หรือพหูพจน์ เช่น `error(s)` ซึ่งไม่สามารถแปลได้ ถ้าคุณต้องแสดงเอกพจน์และพหูพจน์ ให้เขียน *error or errors* คุณยังสามารถปรับปรุงโค้ดเพื่อให้ข้อความที่แตกต่างกัน ขึ้นอยู่กับว่าต้องการคำเอกพจน์หรือพหูพจน์
- อย่าใช้การย่อ
- อย่าใช้เครื่องหมายคำพูด ทั้งเครื่องหมายคำพูดขีดเดียวและขีดคู่ ตัวอย่างเช่น อย่าใช้เครื่องหมายคำพูดล้อมตัวแปร เช่น **%s**, **%c**, และ **%d** หรือล้อมคำสั่ง ผู้ใช้อาจตีความเครื่องหมายคำพูดแบบตรงตัว
- อย่าใส่เครื่องหมาย hyphen คำที่ตอนท้ายของบรรทัด
- อย่าใช้แนวทางการไฮไลต์มาตรฐานในข้อความ และอย่าแทนอักษรขึ้นต้นหรือตัวพิมพ์ใหญ่ทั้งหมดเพื่อการไฮไลต์อื่น (การไฮไลต์มาตรฐานรวมแนวทางดังกล่าวเป็นตัวหนาสำหรับคำสั่ง รูทีนย่อย และไฟล์; เป็นตัวเอนสำหรับตัวแปรและพารามิเตอร์; ฟอนต์ typewriter หรือ courier สำหรับตัวอย่างและข้อความที่แสดงผล)

- อย่าใช้โครงสร้าง และ/หรือ โครงสร้างนี้ไม่มีอยู่ในภาษาอื่น โดยปกติ ควรพูด หรือ เพื่อบ่งชี้ว่าไม่จำเป็นต้องทำทั้งสองอย่าง
- ใช้นาฬิกา 24 ชั่วโมง อย่าใช้ a.m. หรือ p.m. เพื่อระบุเวลา ตัวอย่างเช่น ให้เขียน 1:00 p.m. เป็น 1300
- หลีกเลี่ยงชื่อย่อ ใช้เฉพาะชื่อย่อที่ผู้อ่านของคุณรู้จักเป็นอย่างดีแทนการสะกดคำนั้น เมื่อต้องการทำให้ชื่อย่อเป็นพหูพจน์ ให้เพิ่มตัวพิมพ์เล็ก s โดยไม่มีเครื่องหมาย apostrophe ตรวจสอบว่าชื่อย่อไม่ใช่เครื่องหมายการค้าก่อนการใช้ชื่อย่อนั้น
- อย่าสร้างข้อความจากประโยคย่อย ใช้แฟล็กหรือสื่ออื่นภายในโปรแกรมเพื่อส่งผ่านข้อมูล เพื่อให้สามารถใช้ข้อความที่สมบูรณ์ในเวลาที่เหมาะสม
- อย่าใช้ข้อความ hard-coded เป็นตัวแปรสำหรับสตริง %s ในข้อความ
- จบบรรทัดสุดท้ายของข้อความด้วย **n** (การบ่งชี้บรรทัดใหม่) ซึ่งใช้กับข้อความบรรทัดเดียวด้วย
- ขึ้นต้นบรรทัดที่สองและบรรทัดที่เหลืออยู่ของข้อความด้วย **t** (การบ่งชี้แท็บ)
- จบบรรทัดอื่นทั้งหมดด้วย **n**\ (การบ่งชี้บรรทัดใหม่)
- บังคับใช้บรรทัดใหม่บนตำแหน่งคำที่ต้องการ เพื่อให้สตริงข้อความที่ยอมรับได้ แสดงขึ้น รูทีนย่อย **printf** ซึ่งมักใช้เพื่อแสดงข้อความข่าวสาร เพิกเฉยตำแหน่งคำและตัดข้อความในเวลาที่จำเป็น ในบางครั้งยังมีการใช้เพื่อแบ่งคำในตำแหน่งกึ่งกลางคำด้วย
- ถ้าด้วยเหตุผลบางอย่าง ข้อความควรจะจบด้วยอักขระบรรทัดใหม่ ให้ผู้เขียนแสดงข้อคิดเห็นเกี่ยวกับผลกระทบนั้น
- วางชื่อของคำสั่งที่เรียกข้อความไว้ข้างหน้าแต่ละข้อความ ตามด้วยเครื่องหมาย colon ตัวอย่างต่อไปนี้เป็นข้อความที่มีชื่อคำสั่ง:

  ```
  OPIE "my_example: การเปิดไฟล์"
  ```
- อย่าบอกผู้ใช้ว่า ลองอีกครั้งในภายหลัง ยกเว้นว่าระบบโอเวอร์โหลด ความต้องการให้พยายามอีกครั้งควรจะแสดงแยกจากข้อความอย่างชัดเจน
- ใช้คำว่า "พารามิเตอร์" เพื่ออธิบายข้อความบนบรรทัดคำสั่ง คำว่า "ค่า" เพื่อบ่งชี้ข้อมูลตัวเลข และคำว่า "สตริงคำสั่ง" เพื่ออธิบายคำสั่งพร้อมกับพารามิเตอร์
- อย่าใช้คอมมา เพื่อคั่นตำแหน่งหลักพันในค่า ตัวอย่างเช่น ใช้1000 แทน 1,000
- ถ้าข้อความต้องมีการคั่นด้วย * (ดอกจัน) ให้ใช้ดอกจันสองเครื่องหมายที่ตอนต้นของข้อความ และดอกจันอีกสองเครื่องหมายที่ตอนท้ายของข้อความ ตัวอย่างเช่น:

  ```
  ** ทั้งหมด **
  ```
- ใช้คำว่า "log in" และ "log off" เป็นกริยา ตัวอย่างเช่น:

  ```
  ใช้คำว่า "log in" และ "log off" เป็นกริยา ตัวอย่างเช่น:
  ```
- ใช้คำว่า "user name," "group name," และ "login" เป็นคำนาม ตัวอย่างเช่น:

  ```
  ผู้ใช้คือ sam
  ชื่อกลุ่มคือทีมงาน
  ไดเร็กทอรีที่ล็อกอินคือ /u/sam
  ```
- อย่าใช้คำว่า "superuser" โปรดสังเกตว่า ผู้ใช้รูทอาจไม่มีสิทธิ์ทั้งหมด
- ใช้ข้อความมาตรฐานที่เกิดขึ้นบ่อยต่อไปนี้เมื่อสามารถใช้ได้:

ข้อความมาตรฐานที่ต้องการ	ข้อความที่ไม่ควรใช้
ไม่พบหรือไม่สามารถเปิดไฟล์ได้	ไม่สามารถเปิดชื่อไฟล์
ไม่พบหรือไม่สามารถเข้าถึงไฟล์ได้	ไม่สามารถเข้าถึงได้
ไวยากรณ์ของพารามิเตอร์ไม่ถูกต้อง	ข้อผิดพลาดไวยากรณ์

การจัดการข้อมูลเฉพาะวัฒนธรรม

การจัดการข้อมูลเฉพาะวัฒนธรรมอาจเป็นส่วนประกอบหนึ่งของโปรแกรม และโปรแกรมดังกล่าวอาจแสดงข้อมูลที่แตกต่างกันสำหรับโลแคลที่แตกต่างกัน นอกจากนี้ โปรแกรมอาจใช้ขั้นตอนวิธีการที่แตกต่างกันในการประมวลผลข้อมูล อักขระโดยขึ้นอยู่กับภาษาและวัฒนธรรม

ตัวอย่างเช่น การรับรู้การเริ่มต้นและการสิ้นสุดของคำและ วิธีการใช้ยัติภังค์ของคำระหว่างสองบรรทัดมีความแตกต่างกันไปโดยขึ้นอยู่กับ โลแคล โปรแกรมที่จัดการกับฟังก์ชันดังกล่าวต้องเข้าถึง ตารางหรือขั้นตอนวิธีการเหล่านี้ ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน ณ รันไทม์ คุณสามารถจัดการโปรแกรมดังกล่าวในวิธีดังต่อไปนี้:

- คอมไพล์ขั้นตอนวิธีการและตารางทั้งหมด จากนั้นโหลดพร้อมกับ โปรแกรม

  วิธีการนี้ทำให้การเพิ่มหรือการดัดแปลงขั้นตอนวิธีการและตาราง กลายเป็นเรื่องยาก เมื่อใดก็ตามที่มีการเพิ่มขั้นตอนวิธีการหรือตารางใหม่ ต้องเชื่อมโยง ทั้งโปรแกรมใหม่
- เก็บรักษาขั้นตอนวิธีการและตารางเฉพาะโลแคลไว้ในไฟล์ และ โหลดข้อมูลเหล่านั้น ณ รันไทม์ ทั้งนี้ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน

  วิธีการ นี้ทำให้การดัดแปลงและการเพิ่มขั้นตอนวิธีการและตารางง่ายขึ้น อย่างไรก็ตาม ไม่มีการกำหนดวิธีมาตรฐานในการโหลดขั้นตอนวิธีการ ใน AIX คุณสามารถทำเช่นนั้นได้โดยใช้รูทีนย่อย **load** แต่โปรแกรมที่ใช้รูทีนย่อย **load** อาจไม่สามารถใช้กับระบบอื่นได้

ตารางเฉพาะวัฒนธรรม

ถ้าข้อมูลเฉพาะวัฒนธรรมสามารถประมวลผลได้โดยการเข้าถึง ตารางตามการตั้งค่าโลแคลปัจจุบัน แสดงว่าการดำเนินการเดียวกันนี้สามารถทำได้ โดยใช้รูทีนย่อยไฟล์ I/O มาตรฐาน (**fopen, fread, open, read,** และอื่นๆ) ต้องมีการจัดเตรียมตารางดังกล่าวไว้ในไดเร็กทอรีที่กำหนดไว้ใน **/usr/lpp/***ชื่อ* โดยที่ *ชื่อ* คือชื่อของ แอ็พพลิเคชันเฉพาะภายใต้ชื่อโลแคลที่เหมาะสม

คำเสริมหน้าพาธมาตรฐาน
: **/usr/lpp/***ชื่อ* (ชื่อพาธเฉพาะ AIX)

ไดเร็กทอรีเฉพาะวัฒนธรรม
: ได้ข้อมูลโลแคลปัจจุบันสำหรับหมวดหมู่ที่เหมาะสมซึ่ง อธิบายตาราง นำไปต่อกับคำเสริมหน้าข้างบน

การเข้าถึง
: ใช้รูทีนย่อยการเข้าถึงไฟล์มาตรฐาน (**fopen, fread,** และอื่นๆ) ตามความเหมาะสม

ขั้นตอนวิธีการเฉพาะวัฒนธรรม

ขั้นตอนวิธีการเฉพาะวัฒนธรรมอยู่ในไดเร็กทอรี **/usr/lpp/***ชื่อ***/%L** ที่นี่ **%L** แสดงถึงการตั้งค่าโลแคลปัจจุบัน สำหรับหมวดหมู่ที่เหมาะสม

ใช้รูทีนย่อย **load** เพื่อเข้าถึงขั้นตอนวิธีการเฉพาะโปรแกรม จากโมดูลอ็อบเจ็กต์

คำเสริมหน้าพาธมาตรฐาน

/usr/lpp/*ชื่อ*

ไดเร็กทอรีเฉพาะวัฒนธรรม

ได้ข้อมูลโลแคลปัจจุบันสำหรับหมวดหมู่ที่เหมาะสม นำไปต่อ กับคำเสริมหน้าข้างบน

วิธีการ นำชื่อวิธีการไปต่อกับข้อมูลข้างบน

ตัวอย่างของการโหลดโมดูลเฉพาะวัฒนธรรมสำหรับข้อความอารบิกสำหรับแอ็พพลิเคชัน

ส่วนนี้อธิบายการโหลดโมดูลเฉพาะวัฒนธรรมสำหรับข้อความอารบิก สำหรับแอ็พพลิเคชัน

ไฟล์ส่วนหัว

ส่วนนี้อธิบายโครงสร้างหนึ่งในไฟล์รวม **method.h**

ไฟล์รวม **methods.h** มีโครงสร้างหนึ่งดังนี้:

```
struct Methods {
        int     version;
        char    *(*hyphen)();
        char    *(*wordbegin)();
        char    *(*wordend)();
} ;
```

โปรแกรมหลัก

ในตัวอย่างนี้ชื่อโปรแกรมคือ **textpr**

โปรแกรมหลักกำหนดโมดูลที่จะโหลดและเรียกใช้โมดูลนั้น หมายเหตุว่า textpr.h มีไฟล์ที่จะใช้เพื่อระบุ ชื่อพาธของอ็อบเจ็กต์ที่โหลด ในวิธีนี้ชื่อพาธซึ่งเป็นข้อมูลเฉพาะระบบ สามารถเปลี่ยนได้ง่าย

```
#include    <stdio.h&gt;
#include    <errno.h&gt;
#include    "methods.h"
#include    "textpr.h"    /* contains the pathname where
                             the load object can be found */

extern     int   errno;

main()
{
     char libpath[PATH_MAX];&rbl;&rbl;&rbl;/* stores the full pathname of the
                                load object */
      char *prefix_path=PREFIX_PATH;        /* from textpr.h */
      char *method=METHOD;                  /* from textpr.h */
      int (*func)();
      char *path;
      /* Methods */
      int   ver;
      char  *p;
      struct Methods *md;
```

```
      setlocale(LC_ALL, "");

      path = setlocale(LC_CTYPE, 0);    /* obtain the locale
                                         for LC_CTYPE category */
      /* Construct the full pathname for the */
      /* object to be loaded                 */
      strcpy(libpath, prefix_path);
      strcat(libpath, path);
      strcat(libpath, "/");
      strcat(libpath, method);

      func = load(conv, 1, libpath);        /* load the object */
      if(func==NULL){
            strerror(errno);
            exit(1);
      }
      /* invoke the loaded module ");
      md =(struct Methods *) func();       /* Obtain the methods
                                              structure */
      ver = md-&gt;version;
      /* Invoke the methods as needed */
      p = (md-&gt;hyphen)();
      p = (md-&gt;wordbegin)();
      p = (md-&gt;wordend)();
}
```

วิธีการ

ส่วนนี้มีขั้นตอนวิธีการเฉพาะวัฒนธรรม

ในตัวอย่างนี้ แสดงวิธีการอารบิก โปรแกรม `method.c` **มีดังนี้:**

```
#include "methods.h"

char *Arabic_hyphen(char *);
char *Arabic_wordbegin(char *);
char *Arabic_wordend(char *);

static struct Methods ArabicMethods= {
        1,
        Arabic_hyphen,
        Arabic_wordbegin,
        Arabic_wordend
} ;

struct Methods *start_methods()
{
        /* เริ่มต้นวิธีการ */
        return ( &ArabicMethods);
}

char *Arabic_hyphen(char *string)
{
        /* hyphen อารบิก */
        return( string );
```

```
}
char *Arabic_wordbegin(char *string)
{
        /*คำขึ้นต้นอารบิก */);
        return( string );
}
char *Arabic_wordend(char *string)
{
        /* คำลงท้ายอารบิก */;
        return( string);
}
```

ไฟล์รวม

ไฟล์รวม **textpr** มีชื่อพาธของโมดูลที่จะโหลด

```
#define PREFIX_PATH "/usr/lpp/textpr"
                    /* นี่เป็นชื่อพาธเฉพาะ AIX */
```

ภาพรวมของโครงร่าง (การจัดรูปแบบข้อความและอักขระสองทิศทาง)

ข้อความสองทิศทาง (BIDI) เกิดขึ้นเมื่อข้อความที่มีการจัดวางทิศทาง แตกต่างกันปรากฏขึ้นพร้อมกัน ตัวอย่างเช่น ข้อความภาษาอังกฤษมีการอ่านจาก ซ้ายไปขวา ข้อความภาษาอารบิกและฮิบรูมีการอ่านจากขวา ไปซ้าย ถ้าทั้งข้อความภาษาอังกฤษและข้อความภาษาฮิบรูปรากฏขึ้นบนบรรทัดเดียวกัน ข้อความจะเป็นแบบสองทิศทาง

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับการจัดรูปทรงข้อความและอักขระสองทิศทาง รวมถึงรายการของสิ่งพิมพ์ที่มีอยู่ ให้ดูที่อยู่เว็บดัง ต่อไปนี้:

http://www.opengroup.org

บันทึกข้อความสองทิศทางตามแนวทางต่อไปนี้:

- คำภาษาอารบิกและฮิบรูมีการบันทึกจากขวาไปซ้าย (สตริงอักขระ หนึ่งตัวถือว่าเป็นหนึ่งคำในการจัดลำดับในสภาพแวดล้อม ตัวอักษรและตัวเลข)
- การอ้างอิงหมายเลขและภาษาอังกฤษมีการบันทึกจากซ้ายไปขวา
- ตัวเลขและเครื่องหมายวรรคตอนมีการบันทึกจากซ้ายไปขวา

สคริปต์สองทิศทางมีการอ่านจากขวาไปซ้ายและจากบนลง ล่าง

ถ้ามีข้อความที่ฝังอยู่ในบรรทัดหนึ่ง ข้อความจะมีการบันทึก จากซ้ายไปขวาและฝังอยู่ในข้อความสองทิศทาง อย่างไรก็ตาม ถ้าข้อความที่ฝังแบ่งอยู่ระหว่างสองบรรทัดขึ้นไป ต้องรักษาลำดับที่ถูกต้องในส่วน จากซ้ายไปขวาเพื่อให้สามารถอ่านจากบนลงล่างได้

ตัวอย่างเช่น ข้อความจากขวาไปซ้ายซึ่งฝังอยู่ในข้อความจากซ้ายไปขวา ซึ่งมีอยู่ในหนึ่งบรรทัดจะมีการบันทึกดังนี้:

```
THERE IS txet lanoitceridib deddebme IN THIS SENTENCE
```

ข้อความจากขวาไปซ้ายซึ่งฝังอยู่ในข้อความจากซ้ายไปขวา ซึ่งแบ่งอยู่ระหว่างสองบรรทัดจะมีการบันทึกดังนี้:

```
THERE IS senil owt neewteb tilps si taht txet lanoitceridib deddebme IN THIS SENTENCE
```

ข้อความทั้งสองมีการรักษาให้สามารถอ่านได้แม้ว่าจะมีการแบ่งข้อความ ที่ฝัง

หลักการที่เกี่ยวข้อง:

"ความเป็นสองทิศทางและการจัดรูปทรงอักขระ" ในหน้า 12
โปรแกรมที่เป็นโกลบอลอาจต้องจัดการข้อความแบบ สองทิศทาง และการจัดรูปทรงอักขระ

สตรีมข้อมูล

ส่วนนี้อธิบายสตรีมข้อมูลสำหรับสภาพแวดล้อมข้อความแบบ สองทิศทาง

สภาพแวดล้อมข้อความแบบสองทิศทางใช้สตรีมข้อมูลต่อไปนี้:

สตรีมข้อมูล	คำอธิบาย

สตรีมข้อมูล ภาพ	ระบบ จัดระเบียบอักขระในลำดับซึ่งมีการแสดงบน จอภาพ
	
	ถ้าสตรีมข้อมูลที่มองเห็นได้มีการแสดงจากซ้ายไปขวา อักขระแรกของสตรีมข้อมูลจะอยู่บนด้านซ้ายของ viewport (จอภาพ หน้าต่าง บรรทัด ฟิลด์ และอื่นๆ) ถ้าสตรีมข้อมูลเดียวกัน มีการแสดงจากขวาไปซ้าย อักขระแรกของสตรีมข้อมูล จะอยู่บนด้านขวา
	
	ถ้าภาษาของการจัดวางการบันทึก ตรงกันข้ามถูกฝังอยู่ในสตรีมข้อมูลที่มองเห็นได้ ลำดับของแต่ละข้อความจะมีการ เก็บรักษาไว้เมื่อการจัดวาง viewport ถูกกลับทิศทาง ตัวอย่างเช่น (ข้อความตัวพิมพ์เล็กแสดงถึงข้อความแบบสองทิศทาง) ถ้าคำสั่ง keystroke เป็น:
	
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	ผลคือ สตรีมข้อมูลที่มองเห็นได้เป็น:
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	
	การแสดงของ สตรีมข้อมูลที่มองเห็นได้บน viewport จากซ้ายไปขวาจะถูกปรับชิดซ้าย ดังนี้:
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	`-------> <---------- ----------->`
	
	ลูกศร บ่งชี้ทิศทางการอ่าน
	
	ถ้าคุณเปลี่ยนการจัดวางของ viewport เป็นจากขวาไปซ้าย สตรีมข้อมูลที่มองเห็นได้จะถูกกลับทิศทาง ปรับชิดขวา และไม่สามารถอ่านได้ดังนี้:
	
	`.ECNETNES SIHT NI bidirectional text SI EREHT`
	`<---------- ----------> <-------`
	
	ดังนั้น ถ้าข้อความภาษาอังกฤษถูกฝังอยู่ในข้อความภาษาอารบิกหรือฮิบรู ข้อความทั้งหมดจะ อยู่ในลำดับการอ่านที่ถูกต้องเฉพาะบน viewport จากซ้ายไปขวาเท่านั้น ลักษณะนี้ เป็นจริงสำหรับภาษาอารบิกหรือฮิบรูที่ฝังอยู่ในภาษาอังกฤษด้วย การกลับทิศทางการจัดวางของ viewport ทำให้ข้อความทั้งสองไม่สามารถอ่านได้

สตรีมข้อมูล	คำอธิบาย

สตรีมข้อมูลโลจิคัล	ระบบ จัดระเบียบอักขระในลำดับที่สามารถอ่านได้ ฟังก์ชันการจัดการแสดงแบบสองทิศทาง จะจัดเรียงสตริงข้อความในลำดับที่สามารถอ่านได้
	
	ถ้าสตรีมข้อมูลเชิงตรรกะ มีการแสดงบน viewport จากซ้ายไปขวา อักขระแรกของสตรีมข้อมูล จะแสดงขึ้นบนด้านซ้าย ถ้าสตรีม ข้อมูลเดียวกันมีการแสดงบน viewport จากขวาไปซ้าย อักขระแรก ของสตรีมข้อมูลจะแสดงขึ้นบนด้านขวา และ ยังคงแสดงในลำดับที่อ่านได้
	
	ถ้าภาษาของการจัดวางการบันทึก ตรงกันข้ามถูกฝังอยู่ในสตรีมข้อมูลเชิงตรรกะ การจัดวางของแต่ละข้อความจะมีการ เก็บรักษาไว้โดยฟังก์ชันการจัดการแสดงแบบสองทิศทาง ตัวอย่างเช่น ถ้าคำสั่ง keystroke เป็น:
	
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	ผลคือ สตรีมข้อมูลเชิงตรรกะจะเหมือนกัน ตัวอย่างเช่น:
	
	`THERE IS bidirectional text IN THIS SENTENCE.`
	
	การแสดงของ สตรีมข้อมูลเชิงตรรกะนี้บน viewport จากซ้ายไปขวา (ปรับชิดซ้าย) เป็นดังนี้:
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	`-------> <---------- ----------`
	
	การแสดงของ สตรีมข้อมูลเชิงตรรกะบน viewport จากขวาไปซ้าย (ปรับชิดขวา) เป็นดังนี้:
	
	`IN THIS SENTENCE. txet lanoitceridib THERE IS`
	`----------> <---------- ------->`
	
	สตรีม ข้อมูลเชิงตรรกะสามารถอ่านได้ในการจัดวาง viewport ทั้งสองทิศทาง

การย้ายเคอร์เซอร์

ส่วนนี้อธิบายการย้ายเคอร์เซอร์บนหน้าจอเมื่อ มีข้อความสองทิศทาง

การย้ายเคอร์เซอร์บนจอภาพที่มีข้อความแบบสองทิศทางเป็น ดังนี้:

การย้ายเคอร์เซอร์	คำอธิบาย

แบบภาพ	เคอร์เซอร์ย้ายจากตำแหน่งปัจจุบันไปทางด้านซ้ายหรือขวา ไปยังอักขระตัวถัดไป หรือเลื่อนขึ้นหรือลงไปยังแถวถัดไป ตัวอย่างเช่น ถ้าเคอร์เซอร์อยู่ที่ตำแหน่งสุดท้ายของส่วนแรกที่มีทิศทางจากซ้ายไปขวา ของประโยคแบบผสม:
	
	`THERE IS_txet lanoitceridib IN THIS SENTENCE.`
	
	ผลคือ การย้ายเคอร์เซอร์ไปทางขวาซึ่งมองเห็นได้จะทำให้เคอร์เซอร์ย้ายไปหนึ่งอักขระ ทางด้านขวาดังนี้:
	
	`THERE IS txet lanoitceridib IN THIS SENTENCE.`
	
	เคอร์เซอร์ ย้ายโดยไม่คำนึงถึงเนื้อหาของข้อความ

การย้ายเคอร์เซอร์	คำอธิบาย

โลจิคัล	เคอร์เซอร์ย้ายจากตำแหน่งปัจจุบันไปยังอักขระตัวถัดไปหรือ อักขระก่อนหน้านี้ในสตรีมข้อมูล อักขระอาจอยู่ติดกับ ตำแหน่งของเคอร์เซอร์ หรืออยู่ที่ตำแหน่งใดก็ได้ในบรรทัดเดียวกัน หรืออยู่ บนบรรทัดอื่นบนจอภาพ การย้ายเคอร์เซอร์เชิงตรรกะต้องการการสแกน สตรีมข้อมูลเพื่อค้นหาอักขระเชิงตรรกะตัวถัดไป ตัวอย่างเช่น ถ้าเคอร์เซอร์อยู่ที่ตำแหน่งสุดท้ายของส่วนแรกที่มีทิศทางจากซ้ายไปขวา ของประโยคแบบผสม:
	`THERE IS_txet lanoitceridib IN THIS SENTENCE.`
	ผลคือ การย้ายเคอร์เซอร์เชิงตรรกะไปยังอักขระถัดไปส่งผลให้เกิดการสแกนสตรีมข้อมูล เพื่อค้นหาอักขระเชิงตรรกะตัวถัดไป เคอร์เซอร์ จะย้ายไปยังส่วนเชิงตรรกะถัดไปของประโยคดังนี้:
	`THERE IS txet lanoitceridib_IN THIS SENTENCE.`
	เคอร์เซอร์ จะย้ายตามเนื้อหา

การจัดรูปทรงอักขระ

การจัดรูปทรงอักขระเกิดขึ้นเมื่อรูปทรงของอักขระ ขึ้นอยู่กับตำแหน่งของอักขระในบรรทัดข้อความ ในบางภาษา เช่น อารบิก อักขระมีรูปทรงที่แตกต่างกันขึ้นอยู่กับ ตำแหน่งของอักขระนั้นในสตริงและอักขระรอบข้าง

ลักษณะต่อไปนี้กำหนดการจัดรูปทรงอักขระในสคริปต์ อารบิก:

- ภาษาเขียนไม่มีตัวนำที่เทียบเท่า
- อักขระมีรูปทรงที่แตกต่างกันขึ้นอยู่กับตำแหน่งของอักขระนั้น ในสตริงและอักขระรอบข้าง
- ภาษาเขียนเป็นตัวพิมพ์แบบเขียน อักขระส่วนใหญ่ของคำเชื่อมต่อกัน ดังเช่นในลายมือภาษาอังกฤษ
- อักขระที่เชื่อมกันทำให้เกิดอักขระที่ไม่มีพื้นที่ว่าง นอกจากนี้ อักขระอาจมีสระหรือเครื่องหมายการออกเสียงที่บันทึกไว้ข้างบนหรือข้างใต้ อักขระนั้น
- อักขระอาจมีความยาวแตกต่างกัน ส่งผลให้เกิดเอาต์พุตที่มีรูปทรงเข้ารหัสสอง รูปทรง

วิธีการจัดรูปทรงอักขระ:

ดำเนินการจัดรูปทรงอักขระแยกต่างหากจากคอมโพเนนต์ระบบ อื่นๆ อย่างไรก็ตาม คอมโพเนนต์โปรแกรมอื่นๆ ควรจะสามารถเข้าถึง การจัดรูปทรงอักขระได้ในฐานะเป็นยูทิลิตี้

ระบบอาจใช้การจัดรูปทรงอักขระในวิธีดังต่อไปนี้:

- เมื่อผู้ใช้ป้อนข้อมูลเข้าในคอมพิวเตอร์ ระบบจะใช้การจัดรูปทรงอักขระ เพื่อจัดรูปทรงอักขระ ระบบจัดเก็บอักขระเหล่านี้ในรูปแบบที่จัดรูปทรงไว้
  วิธีการนี้หลีกเลี่ยงความจำเป็นในการใช้การจัดรูปทรงอักขระ ในทุกครั้งที่อักขระเหล่านี้แสดงขึ้น วิธีการนี้ มีประโยชน์สำหรับข้อมูลคงที่ เช่น เมนูและวิธีใช้ วิธีการนี้ต้องการ การประมวลผลล่วงหน้าสำหรับการเรียงลำดับ การค้นหา หรือการจัดทำดัชนีของอักขระอย่างถูกต้อง
  อักขระ อาจต้องการการจัดรูปทรงใหม่หลังจากการประมวลผลสำหรับการนำเสนอที่เหมาะสม
- เมื่อผู้ใช้ป้อนข้อมูลเข้าในคอมพิวเตอร์ ระบบจะจัดเก็บอักขระ ในรูปแบบที่ยังไม่ได้จัดรูปทรง
  วิธีการนี้อนุญาตให้เรียงลำดับ ค้นหา หรือจัดทำดัชนีอักขระได้ อย่างไรก็ตาม ระบบต้องใช้ การจัดรูปทรงอักขระในทุกครั้งที่อักขระแสดงขึ้น

รูปทรงพื้นฐานคือรูปทรงที่แยกต่างหากซึ่งไม่มีการสร้างขึ้นโดยการจัดรูปทรง อักขระ ใช้รูปทรงพื้นฐานในระหว่างการแก้ไข การค้นหาสตริงอักขระ หรือการดำเนินงานข้อความอื่นๆ ใช้การจัดรูปทรงเฉพาะถ้าข้อความแสดงขึ้น หรือพิมพ์ ถ้าอักขระมีการจัดเก็บอยู่ในรูปทรงที่จัดรูปแบบ ระบบต้อง ยกเลิกการจัดรูปแบบอักขระนั้นก่อนการเรียงลำดับ จัดเรียง ค้นหา หรือจัดทำดัชนี รูปทรงอักขระที่ไม่ใช่รูปทรงที่กำหนดตามตำแหน่งของอักขระนั้นในสตริง เป็นสิ่งที่จำเป็นสำหรับแอ็พพลิเคชันซึ่งจัดการอักขระเฉพาะ และสำหรับการสื่อสารที่มีสภาพแวดล้อมการเข้ารหัสแตกต่างกัน

การจัดรูปทรงอักขระบริบท:

โดยทั่วไปแล้ว การจัดรูปทรงอักขระบริบทคือการเลือก รูปทรงที่ต้องการของอักขระในฟอนต์ที่กำหนดให้ โดยขึ้นอยู่กับตำแหน่งของอักขระนั้นในคำและอักขระรอบข้าง

รูปทรงที่ใช้ได้มีดังต่อไปนี้:

การจัดรูปทรงอักขระ	คำอธิบาย
แยกต่างหาก	อักขระที่ไม่เชื่อมต่อกับอักขระก่อนหน้านี้และไม่เชื่อมต่อกับ อักขระที่ตามมา
สุดท้าย	อักขระที่เชื่อมต่อกับอักขระก่อนหน้านี้แต่ไม่เชื่อมต่อกับ อักขระที่ตามมา
เริ่มต้น	อักขระที่เชื่อมต่อกับอักขระที่ตามมาแต่ไม่เชื่อมต่อกับ อักขระก่อนหน้านี้
กึ่งกลาง	อักขระที่เชื่อมต่อกับทั้งอักขระก่อนหน้านี้และอักขระ ที่ตามมา

อักขระยังอาจมีลักษณะดังต่อไปนี้:

- การเชื่อมต่อกับอักขระก่อนหน้านี้
- การเชื่อมต่อกับอักขระที่ตามมา
- อนุญาตให้การเชื่อมต่อของอักขระรอบข้างส่งผ่านอักขระนั้นไปได้

ชื่อย่อ หมายเลขส่วนประกอบ และอักขระกราฟิกไม่ต้องการการจัดรูปทรง อักขระบริบท เพื่อป้อนอักขระเหล่านี้อย่างถูกต้อง ให้ปิด การจัดรูปทรงอักขระบริบทและใช้อินเตอร์เฟสคีย์บอร์ดเฉพาะ ในการเลือกรูปทรงที่ต้องการอย่างแม่นยำ ติดป้ายอักขระเหล่านี้โดยใช้ ฟิลด์ บรรทัด หรืออักขระควบคุมเพื่อให้เข้าใจความหมายได้ง่ายในภายหลัง

ภาษาและโลแคลที่ได้รับการสนับสนุน

นอกจากโหมดไคลเอ็นต์แบบเต็มจะสนับสนุนหลายโลแคล ปัจจุบัน AIX มี “การสนับสนุนด้านเซิร์ฟเวอร์” สำหรับโลแคล Unicode ใหม่หลายโลแคล

การสนับสนุนโลแคลด้านเซิร์ฟเวอร์จัดเตรียมคุณลักษณะต่อไปนี้:

- ที่เก็บข้อมูลและการดำเนินการของสตริง Unicode
- ตัวช่วยการแปลง (iconv)
- ฟังก์ชันการจัดรูปแบบและการวิเคราะห์สำหรับตัวเลข อัตราแลกเปลี่ยน วันที่ และข้อความ
- ตังวช่วยข้อความเพื่อสนับสนุนการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้
- อัลกอริทึมการตรวจทานความเป็นไปตาม การทำให้เป็นปกติ และขอบเขตของข้อความ
- ความพร้อมใช้งานพร้อมกันของหลายโลแคลในแอ็พพลิเคชันเดียวกัน
- โลแคลที่ไม่ขึ้นอยู่กับชุดโค้ด เช่น อักขระ Unicode ทั้งหมดสามารถใช้ได้ในโลแคลใดๆ

คุณลักษณะต่อไปนี้ไม่มีให้สำหรับการสนับสนุนโลแคลด้านเซิร์ฟเวอร์ และไคลเอ็นต์ต้องเป็นผู้จัดเตรียม:

- ส่วนติดต่อผู้ใช้แบบกราฟิก
- อินพุตของผู้ใช้ (เช่น การแม็พคีย์บอร์ด และวิธีการอินพุต)
- เอาต์พุตขั้นสุดท้าย (ตัวอย่างเช่น หน้าจอ ฟอนต์ การจัดรูปทรงอักขระ และการพิมพ์)

ในการสนับสนุนการทำให้เป็นโกลบอล ซอร์สไฟล์ที่มี นิยามโลแคลอาจแตกต่างกัน และไม่สอดคล้องกัน เนื่องจากมีมาตรฐานต่างกัน นิยามโลแคลหลักสองนิยามถูกใช้เพื่อสร้าง อ็อบเจ็กต์โลแคล: นิยาม IBM และ Unicode Common Locale Data Repository (CLDR)

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX

ภาษา	เขตแดน หรือ เขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาแอฟริกา	แอฟริกาใต้	UTF-8	af_ZA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาแอลเบเนีย	อัลบาเนีย	UTF-8	SQ_AL	SQ_AL.UTF-8	ไม่ใช่	ใช่	452	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาแอลเบเนีย	อัลบาเนีย	UTF-8	sq_AL.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาแอมฮาริค	เอธิโอเปีย	UTF-8	am_ET.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาอารบิก	อัลจีเรีย	UTF-8	AR_DZ	AR_DZ.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	อัลจีเรีย	UTF-8	ar_DZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	บาห์เรน	UTF-8	AR_BH	AR_BH.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	บาห์เรน	UTF-8	ar_BH.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	ดีฟอลต์	UTF-8	AR_AA	AR_AA.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	อียิปต์	UTF-8	AR_EG	AR_EG.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	อียิปต์	UTF-8	ar_EG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	อิรัก	UTF-8	ar_IQ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาอารบิก	จอร์แดน	UTF-8	AR_JO	AR_JO.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	จอร์แดน	UTF-8	ar_JO.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	คูเวต	UTF-8	AR_KW	AR_KW.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	คูเวต	UTF-8	ar_KW.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	เลบานอน	UTF-8	AR_LB	AR_LB.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	เลบานอน	UTF-8	ar_LB.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	ลิเบีย	UTF-8	ar_LY.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาอารบิก	มอริเตเนีย	UTF-8	ar_MR.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอารบิก	โมร็อกโก	UTF-8	AR_MA	AR_MA.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	โมร็อกโก	UTF-8	ar_MA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	โอมาน	UTF-8	AR_OM	AR_OM.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	โอมาน	UTF-8	ar_OM.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	กาตาร์	UTF-8	AR_QA	AR_QA.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	กาตาร์	UTF-8	ar_QA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	ซาอุดีอาระเบีย	UTF-8	AR_SA	AR_SA.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ซาอุดีอาระเบีย	UTF-8	ar_SA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	ซีเรีย	UTF-8	AR_SY	AR_SY.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ซีเรีย	UTF-8	ar_SY.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	ตูนีเซีย	UTF-8	AR_TN	AR_TN.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ตูนีเซีย	UTF-8	ar_TN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดนหรือเขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาอารบิก	สหรัฐอาหรับเอมิเรตส์	UTF-8	AR_AE	AR_AE.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	สหรัฐอาหรับเอมิเรตส์	UTF-8	ar_AE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอารบิก	เยเมน	UTF-8	AR_YE	AR_YE.UTF-8	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	เยเมน	UTF-8	ar_YE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอาร์เมเนีย	อาร์มาเนีย	UTF-8	hy_AM.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาอัสสัม	อินเดีย	UTF-8	AS_IN	AS_IN.UTF-8	ไม่ใช่	ใช่	485	ใช่	5.3
ภาษาอัสสัม	อินเดีย	UTF-8	as_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอาเซอร์ไบจาน (ลาติน)	อเซอร์ไบจัน	UTF-8	AZ_AZ	AZ_AZ.UTF-8	ไม่ใช่	ใช่	490	ใช่	6.1
ภาษาอาเซอร์ไบจาน (ลาติน)	อาเซอร์ไบจัน	UTF-8	az_Latn_AZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาบาสก์	สเปน	UTF-8	eu_ES.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเบลารุส	เบลารุส	UTF-8	BE_BY	BE_BY.UTF-8	ไม่ใช่	ใช่	463	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเบลารุส	เบลารุส	UTF-8	be_BY.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเบงกาลี	บังกลาเทศ	UTF-8	bn_BD.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเบงกาลี	อินเดีย	UTF-8	BN_IN	BN_IN.UTF-8	ไม่ใช่	ใช่	480	ใช่	5.3
ภาษาเบงกาลี	อินเดีย	UTF-8	bn_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาบอสเนีย	บอสเนีย	UTF-8	bs_Latn_BA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.2.1.0
ภาษาบัลแกเรีย	บัลกาเรีย	UTF-8	BG_BG	BG_BG.UTF-8	ไม่ใช่	ใช่	442	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาบัลแกเรีย	บัลกาเรีย	UTF-8	bg_BG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.20
ภาษาเมียนมา	เมียนมา	UTF-8	my_MM.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.2.1.0
ภาษาคาตาลัน	สเปน	UTF-8	CA_ES	CA_ES.UTF-8	ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาคาตาลัน	สเปน	UTF-8	ca_ES.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาจีน (ฮั่น)	จีน	UTF-8	ZH_CN	ZH_CN.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	จีน	UTF-8	zh_Hans_CN.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาจีน (ฮั่น)	เขตปกครองพิเศษฮ่องกงของจีน	UTF-8	ZH_HK	ZH_HK.UTF-8	ใช่	ใช่	467	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	สิงคโปร์	UTF-8	ZH_SG	ZH_SG.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	สิงคโปร์	UTF-8	zh_Hans_SG.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาจีน (ฮั่น)	เขตปกครองพิเศษฮ่องกงของจีน	UTF-8	zh_Hant_HK.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาจีน (ฮั่น)	มาเก๊า	UTF-8	zh_Hant_MO.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.2.1.0
ภาษาจีน (ฮั่น)	ไต้หวัน	UTF-8	ZH_TW	ZH_TW.UTF-8	ใช่	ใช่	467	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	ไต้หวัน	UTF-8	zh_Hant_TW.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาโครเอเชีย	โครเอเชีย	UTF-8	HR_HR	HR_HR.UTF-8	ไม่ใช่	ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโครเอเชีย	โครเอเชีย	UTF-8	hr_HR.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเช็ก	สาธารณรัฐเช็ก	UTF-8	CS_CZ	CS_CZ.UTF-8	ใช่	ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเช็ก	สาธารณรัฐเช็ก	UTF-8	cs_CZ.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเดนมาร์ก	เดนมาร์ก	UTF-8	DA_DK	DA_DK.UTF-8	ไม่ใช่	ใช่	159	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเดนมาร์ก	เดนมาร์ก	UTF-8	da_DK.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาดัตช์	เบลเยียม	UTF-8	NL_BE	NL_BE.UTF-8	ไม่ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดน หรือ เขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาดัตช์	เบลเยียม	UTF-8	nl_BE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาดัตช์	เนเธอร์แลนด์	UTF-8	NL_NL	NL_NL.UTF-8	ไม่ใช่	ใช่	143	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เนเธอร์แลนด์	UTF-8	nl_NL.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	ออสเตรเลีย	UTF-8	EN_AU	EN_AU.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ออสเตรเลีย	UTF-8	en_AU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	เบลเยียม	UTF-8	EN_BE	EN_BE.UTF-8	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	เบลเยียม	UTF-8	en_BE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	แคนาดา	UTF-8	EN_CA	EN_CA.UTF-8	ใช่	ใช่	445	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	แคนาดา	UTF-8	en_CA.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	แคเมอรูน	UTF-8	en_CM.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	กานา	UTF-8	en_GH.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	เขตปกครองพิเศษฮ่องกงของจีน	UTF-8	EN_HK	EN_HK.UTF-8	ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	เขตปกครองพิเศษฮ่องกงของจีน	UTF-8	en_HK.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	เคนยา	UTF-8	en_KE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	อินเดีย	UTF-8	EN_IN	EN_IN.UTF-8	ใช่	ใช่	468	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	อินเดีย	UTF-8	en_IN.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	ไอร์แลนด์	UTF-8	EN_IE	EN_IE.UTF-8	ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ไอร์แลนด์	UTF-8	en_IE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	มอริเชียส	UTF-8	en_MU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	นิวซีแลนด์	UTF-8	EN_NZ	EN_NZ.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	นิวซีแลนด์	UTF-8	en_NZ.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	ไนจีเรีย	UTF-8	en_NG.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	ฟิลิปปินส์	UTF-8	EN_PH	EN_PH.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ฟิลิปปินส์	UTF-8	en_PH.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	สิงคโปร์	UTF-8	EN_SG	EN_SG.UTF-8	ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สิงคโปร์	UTF-8	en_SG.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	แอฟริกาใต้	UTF-8	EN_ZA	EN_ZA.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	แอฟริกาใต้	UTF-8	en_ZA.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	ทานซาเนีย	UTF-8	en_TZ.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอังกฤษ	สหราชอาณาจักร	UTF-8	EN_GB	EN_GB.UTF-8	ใช่	ใช่	166	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหราชอาณาจักร	UTF-8	en_GB.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	สหรัฐอเมริกา	UTF-8	EN_US	EN_US.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหรัฐอเมริกา	UTF-8	en_US.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอังกฤษ	แซมเบีย	UTF-8	en_ZM.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาเอสโตเนีย	เอสโตเนีย	UTF-8	ET_EE	ET_EE.UTF-8	ไม่ใช่	ใช่	454	ไม่ใช่	5.3
ภาษาเอสโตเนีย	เอสโตเนีย	UTF-8	et_EE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฟิลิปปินส์ (ลาติน)	ฟิลิปปินส์	UTF-8	fil_PH.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาฟินแลนด์	ฟินแลนด์	UTF-8	FI_FI	FI_FI.UTF-8	ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เซตแคน หรือ เซตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของโคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาฟินแลนด์	ฟินแลนด์	UTF-8	fi_FI.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	อัลจีเรีย	UTF-8	fr_DZ.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	เบลเยียม	UTF-8	FR_BE	FR_BE.UTF-8	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	เบลเยียม	UTF-8	fr_BE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	แคนาดา	UTF-8	FR_CA	FR_CA.UTF-8	ใช่	ใช่	58	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	แคนาดา	UTF-8	fr_CA.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	แคเมอรูน	UTF-8	fr_CM.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	สาธารณรัฐประชาธิปไตยคองโก	UTF-8	fr_CD.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	ฝรั่งเศส	UTF-8	FR_FR	FR_FR.UTF-8	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ฝรั่งเศส	UTF-8	fr_FR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	ลักเซมเบิร์ก	UTF-8	FR_LU	FR_LU.UTF-8	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ลักเซมเบิร์ก	UTF-8	fr_LU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	ไอโวรีโคสต์	UTF-8	fr_CI.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	มอริเตเนีย	UTF-8	fr_MR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	มอริเชียส	UTF-8	fr_MU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	โมร็อกโก	UTF-8	fr_MA.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาฝรั่งเศส	เซเนกัล	UTF-8	fr_SN.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.3.0
ภาษาฝรั่งเศส	สวิตเซอร์แลนด์	UTF-8	FR_CH	FR_CH.UTF-8	ใช่	ใช่	150F	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	สวิตเซอร์แลนด์	UTF-8	fr_CH.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฝรั่งเศส	ตูนีเซีย	UTF-8	fr_TN.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษากาลิเซีย	สเปน	UTF-8	gl_ES.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษากานดา	ยูกานดา	UTF-8	lg_UG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาจอร์เจีย	จอร์เจีย	UTF-8	ka_GE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเยอรมัน	ออสเตรีย	UTF-8	DE_AT	DE_AT.UTF-8	ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	ออสเตรีย	UTF-8	de_AT.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเยอรมัน	เยอรมนี	UTF-8	DE_DE	DE_DE.UTF-8	ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	เยอรมนี	UTF-8	de_DE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเยอรมัน	ลักเซมเบิร์ก	UTF-8	DE_LU	DE_LU.UTF-8	ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	ลักเซมเบิร์ก	UTF-8	de_LU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเยอรมัน	สวิตเซอร์แลนด์	UTF-8	DE_CH	DE_CH.UTF-8	ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	สวิตเซอร์แลนด์	UTF-8	de_CH.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษากรีก	กรีซ	UTF-8	EL_GR	EL_GR.UTF-8	ไม่ใช่	ใช่	319	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษากรีก	กรีซ	UTF-8	el_GR.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาคุชราต	อินเดีย	UTF-8	GU_IN	GU_IN.UTF-8	ไม่ใช่	ใช่	477	ไม่ใช่	5.3
ภาษาคุชราต	อินเดีย	UTF-8	gu_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเฮาซา	ไนจีเรีย	UTF-8	ha_Latn_NG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาฮิบรู	อิสราเอล	UTF-8	HE_IL	HE_IL.UTF-8	ไม่ใช่	ใช่	115	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฮิบรู	อิสราเอล	UTF-8	he_IL.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาฮินดี	อินเดีย	UTF-8	HI_IN	HI_IN.UTF-8	ไม่ใช่	ใช่	468	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฮินดี	อินเดีย	UTF-8	hi_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดน หรือ เขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาฮังการี	ฮังการี	UTF-8	HU_HU	HU_HU.UTF-8	ใช่	ใช่	208	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฮังการี	ฮังการี	UTF-8	hu_HU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาลาว	ลาว	UTF-8	lo_LA.UTF-8	ไม่ใช่	ไม่ใช่		103P		7.2.1.0
ภาษาไอซ์แลนด์	ไอซ์แลนด์	UTF-8	IS_IS	IS_IS.UTF-8	ไม่ใช่	ใช่	197	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาไอซ์แลนด์	ไอซ์แลนด์	UTF-8	is_IS.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอิกโบ	ไนจีเรีย	UTF-8	ig_NG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาอินโดนีเซีย	อินโดนีเซีย	UTF-8	ID_ID	ID_ID.UTF-8	ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอินโดนีเซีย	อินโดนีเซีย	UTF-8	id_ID.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอิตาลี	อิตาลี	UTF-8	IT_IT	IT_IT.UTF-8	ใช่	ใช่	142	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	อิตาลี	UTF-8	it_IT.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอิตาลี	สวิตเซอร์แลนด์	UTF-8	IT_CH	IT_CH.UTF-8	ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	สวิตเซอร์แลนด์	UTF-8	it_CH.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาญี่ปุ่น	ญี่ปุ่น	UTF-8	JA_JP	JA_JP.UTF-8	ใช่	ใช่	194	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาญี่ปุ่น	ญี่ปุ่น	UTF-8	ja_JP.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษากันนาดา	อินเดีย	UTF-8	KN_IN	KN_IN.UTF-8	ไม่ใช่	ใช่	483	ใช่	5.3
ภาษากันนาดา	อินเดีย	UTF-8	kn_IN.UTF-8	8	ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาคาซัค	คาซัคสถาน	UTF-8	KK_KZ	KK_KZ.UTF-8	ไม่ใช่	ใช่	476	ไม่ใช่	5.3
ภาษาคาซัค	คาซัคสถาน	UTF-8	kk_Cyrl_KZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาคินยาร์วันดา	รวันดา	UTF-8	rw_RW.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเขมร	กัมพูชา	UTF-8	km_KH.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษากงกณี	อินเดีย	UTF-8	kok_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเกาหลี	เกาหลีใต้	UTF-8	KO_KR	KO_KR.UTF-8	ใช่	ใช่	413	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเกาหลี	เกาหลีใต้	UTF-8	ko_KR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาลัตเวีย	ลัตเวีย	UTF-8	LV_LV	LV_LV.UTF-8	ไม่ใช่	ใช่	455	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาลัตเวีย	ลัตเวีย	UTF-8	lv_LV.UTF-8		ไม่ใช่	ใช่	103P	ใช่	7.1.2.0
ภาษาลิทัวเนีย	ลิทัวเนีย	UTF-8	LT_LT	LT_LT.UTF-8	ไม่ใช่	ใช่	456	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาลิทัวเนีย	ลิทัวเนีย	UTF-8	lt_LT.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามาเซโดเนีย	มาเซโดเนีย	UTF-8	MK_MK	MK_MK.UTF-8	ไม่ใช่	ใช่	449	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษามาเซโดเนีย	มาเซโดเนีย	UTF-8	mk_MK.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามาเลย์	มาเลเซีย	UTF-8	MS_MY	MS_MY.UTF-8	ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษามาเลย์	มาเลเซีย	UTF-8	ms_MY.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามาลายาลัม	อินเดีย	UTF-8	ML_IN	ML_IN.UTF-8	ไม่ใช่	ใช่	479	ไม่ใช่	5.3
ภาษามาลายาลัม	อินเดีย	UTF-8	ml_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามอลตา	มอลตา	UTF-8	MT_MT	MT_MT.UTF-8	ไม่ใช่	ใช่	491	ไม่ใช่	6.1
ภาษามอลตา	มอลตา	UTF-8	mt_MT.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามราฐี	อินเดีย	UTF-8	MR_IN	MR_IN.UTF-8	ไม่ใช่	ใช่	468	ไม่ใช่	5.3
ภาษามราฐี	อินเดีย	UTF-8	mr_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษามองโกเลียน (ซีริลลิก)	มองโกเลีย	UTF-8	mn_Cyrl_MN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาเนปาล	อินเดีย	UTF-8	ne_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเนปาล	เนปาล	UTF-8	ne_NP.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เซตแดน หรือ เซตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษานอร์เวย์บุ๊กมอล	นอร์เวย์	UTF-8	nb_NO.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษานอร์เวย์นีอนอสก์	นอร์เวย์	UTF-8	nn_NO.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาโอริยา	อินเดีย	UTF-8	OR_IN	OR_IN.UTF-8	ไม่ใช่	ใช่	482	ใช่	5.3
ภาษาโอริยา	อินเดีย	UTF-8	or_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.4.1.0
ภาษาโอโรโม	เอธิโอเปีย	UTF-8	om_ET.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาโปแลนด์	โปแลนด์	UTF-8	PL_PL	PL_PL.UTF-8	ใช่	ใช่	214	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปแลนด์	โปแลนด์	UTF-8	pl_PL.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาโปรตุเกส	แองโกลา	UTF-8	pt_AO.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาโปรตุเกส	บราซิล	UTF-8	PT_BR	PT_BR.UTF-8	ใช่	ใช่	275	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	บราซิล	UTF-8	pt_BR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาโปรตุเกส	มาเก๊า	UTF-8	pt_MO.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.2.1.0
ภาษาโปรตุเกส	โมแซมบิก	UTF-8	pt_MZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาโปรตุเกส	โปรตุเกส	UTF-8	PT_PT	PT_PT.UTF-8	ไม่ใช่	ใช่	163	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	โปรตุเกส	UTF-8	pt_PT.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาปัญจาบ	อินเดีย	UTF-8	PA_IN	PA_IN.UTF-8	ไม่ใช่	ใช่	484	ใช่	5.3
ภาษาปัญจาบ	อินเดีย	UTF-8	pa_Guru_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาโรมาเนีย	โรมาเนีย	UTF-8	RO_RO	RO_RO.UTF-8	ไม่ใช่	ใช่	446	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโรมาเนีย	โรมาเนีย	UTF-8	ro_RO.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษารัสเซีย	รัสเซีย	UTF-8	RU_RU	RU_RU.UTF-8	ใช่	ใช่	441	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษารัสเซีย	รัสเซีย	UTF-8	ru_RU.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเซอร์เบีย (ลาติน)	มอนเตเนโกร	UTF-8	sr_Latn_ME.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.2.1.0
ภาษาเซอร์เบีย (ซีริลลิก)	เซอร์เบีย	UTF-8	sr_Cyrl_RS.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเซอร์เบีย (ซีริลลิก)	เซอร์เบีย	UTF-8	SR_SP	SR_SP.UTF-8	ไม่ใช่	ไม่ใช่	450	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเซอร์เบีย (ลาติน)	เซอร์เบีย	UTF-8	sr_Latn_RS.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาเซอร์เบีย (ลาติน)	เซอร์เบีย	UTF-8	SH_SP	SH_SP.UTF-8	ไม่ใช่	ไม่ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสิงหล	ศรีลังกา	UTF-8	si_LK.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาสโลวัก	สโลวาเกีย	UTF-8	SK_SK	SK_SK.UTF-8	ใช่	ใช่	245	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสโลวัก	สโลวาเกีย	UTF-8	sk_SK.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสโลวีเนีย	สโลวีเนีย	UTF-8	SL_SI	SL_SI.UTF-8	ไม่ใช่	ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสโลวีเนีย	สโลวีเนีย	UTF-8	sl_SI.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	อาร์เจนตินา	UTF-8	ES_AR	ES_AR.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	อาร์เจนตินา	UTF-8	es_AR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	โบลิเวีย	UTF-8	ES_BO	ES_BO.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	โบลิเวีย	UTF-8	es_BO.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	ชิลี	UTF-8	ES_CL	ES_CL.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ชิลี	UTF-8	es_CL.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	โคลัมเบีย	UTF-8	ES_CO	ES_CO.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดน หรือ เขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาสเปน	โคลัมเบีย	UTF-8	es_CO.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	คอสตาริกา	UTF-8	ES_CR	ES_CR.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	คอสตาริกา	UTF-8	es_CR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	สาธารณรัฐโดมินิกัน	UTF-8	ES_DO	ES_DO.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สาธารณรัฐโดมินิกัน	UTF-8	es_DO.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เอกวาดอร์	UTF-8	ES_EC	ES_EC.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เอกวาดอร์	UTF-8	es_EC.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เอลซัลวาดอร์	UTF-8	ES_SV	ES_SV.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เอลซัลวาดอร์	UTF-8	es_SV.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	กัวเตมาลา	UTF-8	ES_GT	ES_GT.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	กัวเตมาลา	UTF-8	es_GT.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	ฮอนดูรัส	UTF-8	ES_HN	ES_HN.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ฮอนดูรัส	UTF-8	es_HN.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เม็กซิโก	UTF-8	ES_MX	ES_MX.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เม็กซิโก	UTF-8	es_MX.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	นิการากัว	UTF-8	ES_NI	ES_NI.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	นิการากัว	UTF-8	es_NI.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	ปานามา	UTF-8	ES_PA	ES_PA.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ปานามา	UTF-8	es_PA.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	ปารากวัย	UTF-8	ES_PY	ES_PY.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ปารากวัย	UTF-8	es_PY.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เปรู	UTF-8	ES_PE	ES_PE.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เปรู	UTF-8	es_PE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เปอร์โตริโก	UTF-8	ES_PR	ES_PR.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เปอร์โตริโก	UTF-8	es_PR.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	สเปน	UTF-8	ES_ES	ES_ES.UTF-8	ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สเปน	UTF-8	es_ES.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	สหรัฐฯ	UTF-8	ES_US	ES_US.UTF-8	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สหรัฐฯ	UTF-8	es_US.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	อุรุกวัย	UTF-8	ES_UY	ES_UY.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	อุรุกวัย	UTF-8	es_UY.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสเปน	เวเนซุเอลา	UTF-8	ES_VE	ES_VE.UTF-8	ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เวเนซุเอลา	UTF-8	es_VE.UTF-8		ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาสวาฮีลี	เคนยา	UTF-8	sw_KE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาสวาฮีลี	ทานซาเนีย	UTF-8	sw_TZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาสวีเดน	สวีเดน	UTF-8	SV_SE	SV_SE.UTF-8	ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสวีเดน	สวีเดน	UTF-8	sv_SE.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาทมิฬ	อินเดีย	UTF-8	TA_IN	TA_IN.UTF-8	ไม่ใช่	ใช่	474	ไม่ใช่	5.3
ภาษาทมิฬ	อินเดีย	UTF-8	ta_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเตลูกู	อินเดีย	UTF-8	TE_IN	TE_IN.UTF-8	ไม่ใช่	ใช่	473	ไม่ใช่	5.3

ตารางที่ 15. ภาษาที่เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เซตแคน หรือ เซตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาเตลูกู	อินเดีย	UTF-8	te_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาไทย	ไทย	UTF-8	TH_TH	TH_TH.UTF-8	ไม่ใช่	ใช่	191	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาไทย	ไทย	UTF-8	th_TH.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาตุรกี	ตุรกี	UTF-8	TR_TR	TR_TR.UTF-8	ไม่ใช่	ใช่	179	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาตุรกี	ตุรกี	UTF-8	tr_TR.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษายูเครน	ยูเครน	UTF-8	UK_UA	UK_UA.UTF-8	ไม่ใช่	ใช่	465	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษายูเครน	ยูเครน	UTF-8	uk_UA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอูรดู	อินเดีย	UTF-8	UR_IN	UR_IN.UTF-8	ไม่ใช่	ใช่	492	ใช่	6.1
ภาษาอูรดู	อินเดีย	UTF-8	ur_IN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอูรดู	ปากีสถาน	UTF-8	UR_PK	UR_PK.UTF-8	ไม่ใช่	ใช่	492	ใช่	6.1
ภาษาอูรดู	ปากีสถาน	UTF-8	ur_PK.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาอุซเบ	อุซเบกิสถาน	UTF-8	uz_Cyrl_UZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาอุซเบ	อุซเบกิสถาน	UTF-8	uz_Latn_UZ.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเวียตนาม	เวียตนาม	UTF-8	VI_VN	VI_VN.UTF-8	ไม่ใช่	ใช่	461	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเวียตนาม	เวียตนาม	UTF-8	vi_VN.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.2.0
ภาษาเวลส์	สหราชอาณาจักร	UTF-8	CY_GB	CY_GB.UTF-8	ไม่ใช่	ใช่	166W	ใช่	6.1
ภาษาเวลส์	สหราชอาณาจักร	UTF-8	cy_GB.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.4.0
ภาษาโยรูบา (ละติน)	ไนจีเรีย	UTF-8	yo_NG.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0
ภาษาซูลู	แอฟริกาใต้	UTF-8	zu_ZA.UTF-8		ไม่ใช่	ไม่ใช่	103P	ใช่	7.1.1.0

ตารางที่ 16. ภาษาที่ไม่ได้เข้ารหัส Unicode และโลแคลใน AIX

ภาษา	เซตแคน หรือ เซตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาแอลเบเนีย	แอลเบเนีย	ISO8859-15	sq_AL.8859-15		ไม่ใช่	ใช่	452	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาแอลเบเนีย	แอลเบเนีย	ISO8859-1	sq_AL	sq_AL.ISO8859-1	ไม่ใช่	ใช่	452	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	อัลจีเรีย	ISO8859-6	ar_DZ	ar_DZ.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	บาห์เรน	ISO8859-6	ar_BH	ar_BH.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ดีฟอลต์	ISO8859-6	ar_AA	ar_AA.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ดีฟอลต์	IBM-1046	Ar_AA	Ar_AA.IBM-1046	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	อียิปต์	ISO8859-6	ar_EG	ar_EG.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	จอร์แดน	ISO8859-6	ar_JO	ar_JO.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	คูเวต	ISO8859-6	ar_KW	ar_KW.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	เลบานอน	ISO8859-6	ar_LB	ar_LB.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	โมร็อกโก	ISO8859-6	ar_MA	ar_MA.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	โอมาน	ISO8859-6	ar_OM	ar_OM.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	กาตาร์	ISO8859-6	ar_QA	ar_QA.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ซาอุดีอาระเบีย	ISO8859-6	ar_SA	ar_SA.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ซีเรีย	ISO8859-6	ar_SY	ar_SY.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	ตูนีเซีย	ISO8859-6	ar_TN	ar_TN.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอารบิก	สหรัฐอาหรับเอมิเรตส์	ISO8859-6	ar_AE	ar_AE.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 16. ภาษาที่ไม่ได้เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดนหรือเขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ดLFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาอารบิก	เยเมน	ISO8859-6	ar_YE	ar_YE.ISO8859-6	ไม่ใช่	ใช่	253	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเบลารุส	เบลารุส	ISO8859-5	be_BY	be_BY.ISO8859-5	ไม่ใช่	ใช่	463	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาบัลกาเรีย	บัลกาเรีย	ISO8859-5	bg_BG	bg_BG.ISO8859-5	ไม่ใช่	ใช่	442	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาคาตาลัน	สเปน	ISO8859-15	ca_ES.8859-15		ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาคาตาลัน	สเปน	ISO8859-1	ca_ES	ca_ES.ISO8859-1	ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาคาตาลัน	สเปน	IBM-1252	ca_ES.IBM-1252		ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	จีน	IBM-eucCN	zh_CN	zh_CN.IBM-eucCN	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	จีน	GB18030	zh_CN	Zh_CN.GB18030	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	เขตปกครองพิเศษฮ่องกงของจีน	BIG5-HKSCS	Zh_HK	Zh_HK.BIG5-HKSCS	ใช่	ใช่	467	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	ไต้หวัน	IBM-eucTW	zh_TW	zh_TW.IBM-eucTW	ใช่	ใช่	467	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาจีน (ฮั่น)	ไต้หวัน	big-5	Zh_TW	Zh_TW.big-5	ใช่	ใช่	467	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโครเอเชีย	โครเอเชีย	ISO8859-2	hr_HR	hr_HR.ISO8859-2	ไม่ใช่	ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเช็ก	สาธารณรัฐเช็ก	ISO8859-2	cs_CZ	cs_CZ.ISO8859-2	ใช่	ใช่	243	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเดนมาร์ก	เดนมาร์ก	ISO8859-15	da_DK.8859-15		ไม่ใช่	ใช่	159	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเดนมาร์ก	เดนมาร์ก	ISO8859-1	da_DK	da_DK.ISO8859-1	ไม่ใช่	ใช่	159	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เบลเยียม	ISO8859-15	nl_BE.8859-15		ไม่ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เบลเยียม	ISO8859-1	nl_BE	nl_BE.ISO8859-1	ไม่ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เบลเยียม	IBM-1252	nl_BE.IBM-1252		ไม่ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เนเธอร์แลนด์	ISO8859-15	nl_NL.8859-15		ไม่ใช่	ใช่	143	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เนเธอร์แลนด์	ISO8859-1	nl_NL	nl_NL.ISO8859-1	ไม่ใช่	ใช่	143	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาดัตช์	เนเธอร์แลนด์	IBM-1252	nl_NL.IBM-1252		ไม่ใช่	ใช่	143	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ออสเตรเลีย	ISO8859-15	en_AU.8859-15		ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	เบลเยียม	ISO8859-15	en_BE.8859-15		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	แคนาดา	ISO8859-15	en_CA.8859-15		ใช่	ใช่	445	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	เขตปกครองพิเศษฮ่องกงของจีน	ISO8859-15	en_HK.8859-15		ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	อินเดีย	ISO8859-15	en_IN.8859-15		ใช่	ใช่	468	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ไอร์แลนด์	ISO8859-15	en_IE.8859-15		ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	นิวซีแลนด์	ISO8859-15	en_NZ.8859-15		ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	ฟิลิปปินส์	ISO8859-15	en_PH.8859-15		ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สิงคโปร์	ISO8859-15	en_SG.8859-15		ใช่	ใช่	168	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	แอฟริกาใต้	ISO8859-15	en_ZA.8859-15		ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหราชอาณาจักร	ISO8859-15	en_GB.8859-15		ใช่	ใช่	166	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหราชอาณาจักร	ISO8859-1	en_GB	en_GB.ISO8859-1	ใช่	ใช่	166	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหราชอาณาจักร	IBM-1252	en_GB.IBM-1252		ใช่	ใช่	166	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหรัฐอเมริกา	ISO8859-15	en_US.8859-15		ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอังกฤษ	สหรัฐอเมริกา	ISO8859-1	en_US	en_US.ISO8859-1	ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเอสโตเนีย	เอสโตเนีย	ISO8859-4	et_EE	et_EE.ISO8859-4	ไม่ใช่	ใช่	454	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเอสโตเนีย	เอสโตเนีย	IBM-922	Et_EE	Et_EE.IBM-922	ไม่ใช่	ใช่	454	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฟินแลนด์	ฟินแลนด์	ISO8859-15	fi_FI.8859-15		ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 16. ภาษาที่ไม่ได้เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดนหรือเขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของโลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาฟินแลนด์	ฟินแลนด์	ISO8859-1	fi_FI	fi_FI.ISO8859-1	ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฟินแลนด์	ฟินแลนด์	IBM-1252	fi_FI.IBM-1252		ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	เบลเยียม	ISO8859-15	fr_BE.8859-15		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	เบลเยียม	ISO8859-1	fr_BE	fr_BE.ISO8859-1	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	เบลเยียม	IBM-1252	fr_BE.IBM-1252		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	แคนาดา	ISO8859-15	fr_CA.8859-15		ใช่	ใช่	58	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	แคนาดา	ISO8859-1	fr_CA	fr_CA.ISO8859-1	ใช่	ใช่	58	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ฝรั่งเศส	ISO8859-15	fr_FR.8859-15		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ฝรั่งเศส	ISO8859-1	fr_FR	fr_FR.ISO8859-1	ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ฝรั่งเศส	IBM-1252	fr_FR.IBM-1252		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	ลักเซมเบิร์ก	ISO8859-15	fr_LU.8859-15		ใช่	ใช่	120	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	สวิตเซอร์แลนด์	ISO8859-15	fr_CH.8859-15		ใช่	ใช่	150F	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฝรั่งเศส	สวิตเซอร์แลนด์	ISO8859-1	fr_CH	fr_CH.ISO8859-1	ใช่	ใช่	150F	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	ออสเตรีย	ISO8859-15	de_AT.8859-15		ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	เยอรมนี	ISO8859-15	de_DE.8859-15		ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	เยอรมนี	ISO8859-1	de_DE	de_DE.ISO8859-1	ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	เยอรมนี	IBM-1252	de_DE.IBM-1252		ใช่	ใช่	129	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	ลักเซมเบิร์ก	ISO8859-15	de_LU.8859-15		ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	สวิตเซอร์แลนด์	ISO8859-15	de_CH.8859-15		ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเยอรมัน	สวิตเซอร์แลนด์	ISO8859-1	de_CH	de_CH.ISO8859-1	ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษากรีก	กรีซ	ISO8859-7	el_GR	el_GR.ISO8859-7	ไม่ใช่	ใช่	319	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาฮังการี	ฮังการี	ISO8859-2	hu_HU	hu_HU.ISO8859-2	ใช่	ใช่	208	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาไอซ์แลนด์	ไอซ์แลนด์	ISO8859-15	is_IS.8859-15		ไม่ใช่	ใช่	197	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาไอซ์แลนด์	ไอซ์แลนด์	ISO8859-1	is_IS	is_IS.ISO8859-1	ไม่ใช่	ใช่	197	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอินโดนีเซีย	อินโดนีเซีย	ISO8859-15	id_ID.8859-15		ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	อิตาลี	ISO8859-15	it_IT.8859-15		ใช่	ใช่	142	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	อิตาลี	ISO8859-1	it_IT	it_IT.ISO8859-1	ใช่	ใช่	142	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	อิตาลี	IBM-1252	it_IT.IBM-1252		ใช่	ใช่	142	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาอิตาลี	สวิตเซอร์แลนด์	ISO8859-15	it_CH.8859-15		ใช่	ใช่	150G	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาญี่ปุ่น	ญี่ปุ่น	IBM-eucJP	ja_JP	ja_JP.IBM-eucJP	ใช่	ใช่	194	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาญี่ปุ่น	ญี่ปุ่น	IBM-943	Ja_JP	Ja_JP.IBM-943	ใช่	ใช่	194	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเกาหลี	เกาหลีใต้	IBM-eucKR	ko_KR	ko_KR.IBM-eucKR	ใช่	ใช่	413	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาลัตเวีย	ลัตเวีย	ISO8859-4	lv_LV	lv_LV.ISO8859-4	ไม่ใช่	ใช่	455	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาลิทัวเนีย	ลิทัวเนีย	ISO8859-4	lt_LT	lt_LT.ISO8859-4	ไม่ใช่	ใช่	456	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาลิทัวเนีย	ลิทัวเนีย	IBM-921	Lt_LT	Lt_LT.IBM-921	ไม่ใช่	ใช่	456	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษามาเซโดเนีย	มาเซโดเนีย	ISO8859-5	mk_MK	mk_MK.ISO8859-5	ไม่ใช่	ใช่	449	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษามาเลย์	มาเลเซีย	ISO8859-15	ms_MY.8859-15		ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปแลนด์	โปแลนด์	ISO8859-2	pl_PL	pl_PL.ISO8859-2	ใช่	ใช่	214	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	บราซิล	ISO8859-15	pt_BR.8859-15		ใช่	ใช่	275	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	บราซิล	ISO8859-1	pt_BR	pt_BR.ISO8859-1	ใช่	ใช่	275	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	โปรตุเกส	ISO8859-15	pt_PT.8859-15		ไม่ใช่	ใช่	163	ไม่ใช่	5.2 หรือก่อนหน้า

ตารางที่ 16. ภาษาที่ไม่ได้เข้ารหัส Unicode และโลแคลใน AIX (ต่อ)

ภาษา	เขตแดนหรือเขตภูมิภาค	ชุดโค้ด	ชื่อโลแคลดีฟอลต์	นามแฝง	แปลงหรือไม่?	มีการจัดเตรียมการสนับสนุนของไคลเอ็นต์หรือไม่?	ID ของคีย์บอร์ด LFT ดีฟอลต์	โลแคล CLDR?	นำมาใช้ในรีลีส AIX
ภาษาโปรตุเกส	โปรตุเกส	ISO8859-1	pt_PT	pt_PT.ISO8859-1	ไม่ใช่	ใช่	163	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโปรตุเกส	โปรตุเกส	IBM-1252	pt_PT.IBM-1252		ไม่ใช่	ใช่	163	ไม่ใช่	5.2 หรือก่อนหน้า
POSIX	POSIX	ISO8859-1	C	C.ISO8859-1	ไม่ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาโรมาเนีย	โรมาเนีย	ISO8859-2	ro_RO	ro_RO.ISO8859-2	ไม่ใช่	ใช่	446	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษารัสเซีย	รัสเซีย	ISO8859-5	ru_RU	ru_RU.ISO8859-5	ใช่	ใช่	441	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสโลวัก	สโลวาเกีย	ISO8859-2	sk_SK	sk_SK.ISO8859-2	ใช่	ใช่	245	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสโลวีเนีย	สโลวีเนีย	ISO8859-2	sl_SI	sl_SI.ISO8859-2	ไม่ใช่	ใช่	234	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	อาร์เจนตินา	ISO8859-15	es_AR.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	โบลิเวีย	ISO8859-15	es_BO.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ชิลี	ISO8859-15	es_CL.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	โคลัมเบีย	ISO8859-15	es_CO.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	คอสตาริกา	ISO8859-15	es_CR.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สาธารณรัฐโดมินิกัน	ISO8859-15	es_DO.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เอกวาดอร์	ISO8859-15	es_EC.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เอลซัลวาดอร์	ISO8859-15	es_SV.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	กัวเตมาลา	ISO8859-15	es_GT.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ฮอนดูรัส	ISO8859-15	es_HN.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เม็กซิโก	ISO8859-15	es_MX.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	นิการากัว	ISO8859-15	es_NI.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ปานามา	ISO8859-15	es_PA.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	ปารากวัย	ISO8859-15	es_PY.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เปรู	ISO8859-15	es_PE.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เปอร์โตริโก	ISO8859-15	es_PR.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สเปน	ISO8859-15	es_ES.8859-15		ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สเปน	ISO8859-1	es_ES	es_ES.ISO8859-1	ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สเปน	IBM-1252	es_ES.IBM-1252		ใช่	ใช่	173	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	สหรัฐอเมริกา	ISO8859-15	es_US.8859-15		ใช่	ใช่	103P	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	อุรุกวัย	ISO8859-15	es_UY.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสเปน	เวเนซุเอลา	ISO8859-15	es_VE.8859-15		ใช่	ใช่	171	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสวีเดน	สวีเดน	ISO8859-15	sv_SE.8859-15		ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาสวีเดน	สวีเดน	ISO8859-1	sv_SE	sv_SE.ISO8859-1	ไม่ใช่	ใช่	153	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาไทย	ไทย	TIS-620	th_TH	th_TH.TIS-620	ไม่ใช่	ใช่	191	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาตุรกี	ตุรกี	ISO8859-9	tr_TR	tr_TR.ISO8859-9	ไม่ใช่	ใช่	179	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษายูเครน	ยูเครน	IBM-1124	Uk_UA	Uk_UA.IBM-1124	ไม่ใช่	ใช่	465	ไม่ใช่	5.2 หรือก่อนหน้า
ภาษาเวียตนาม	เวียตนาม	IBM-1129	Vi_VN	Vi_VN.IBM-1129	ไม่ใช่	ใช่	461	ไม่ใช่	5.2 หรือก่อนหน้า

การอ้างอิง Globalization

ส่วนนี้ให้ข้อมูลเกี่ยวกับรายการตรวจสอบ globalization และรูทีนย่อยการสนับสนุน multicultural

รายการสำหรับตรวจสอบ Globalization

รายการสำหรับตรวจสอบ globalization ให้วิธีในการวิเคราะห์โปรแกรมว่ามีการพึ่งพาการแปลและการสนับสนุน multicultural หรือไม่ โดยใช้รายการตรวจสอบนี้ ทำให้สามารถกำหนดได้ว่าต้องพิจารณาฟังก์ชัน globalization ใด ถ้ามี ซึ่งเป็นประโยชน์สำหรับทั้งการเขียนโปรแกรมและการทดสอบ ถ้าคุณระบุชุดของไอเท็ม globalization ที่โปรแกรมพึ่งพา คุณสามารถกำหนดกลยุทธการทดสอบได้ กลยุทธ์นี้แสดงแนวทางทั่วไปในการทดสอบโปรแกรมทั้งหมด

มีการระบุการพิจารณา globalization ที่สำคัญทั้งหมด อย่างไรก็ตาม รายการนี้ไม่ได้ครอบคลุมการพิจารณาทั้งหมด อาจมีคำถามเกี่ยวกับ globalization อื่นที่ไม่มีอยู่ในรายการ

รายการตรวจสอบการดำเนินงานโปรแกรม

1. โปรแกรมแสดงข้อความที่แปลได้ให้แก่ผู้ใช้โดยทางตรง หรือทางอ้อม? ตัวอย่างของข้อความทางอ้อมคือข้อความ ที่จัดเก็บไว้ในไลบรารี
   ถ้าใช่:
   - ข้อความเหล่านี้มีการส่งออกไปจากโปรแกรมโดยใช้ รูทีนย่อยฟังก์ชันข้อความหรือไม่?
   - คุณนำเสนอไฟล์ต้นฉบับข้อความสำหรับข้อความดังกล่าวทั้งหมดหรือไม่?
   - โปรแกรมจะรันอยู่ใต้โลแคลใด?
     - ถ้าโปรแกรมรันอยู่ในโลแคลที่กำหนดโดยตัวแปรสภาพแวดล้อมโลแคล คุณเรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้ หรือไม่?

       ```
       setlocale(LC_ALL, "")
       ```

       หมายเหตุ: หมวดหมู่โลแคลในลำดับตามลำดับชั้นที่กำหนดไว้แล้วมีดังนี้: **LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC,** และ **LC_TIME**
     - ถ้าโปรแกรมรันโลแคล "C" ยกเว้นการแสดงข้อความ ในโลแคลที่ระบุโดยตัวแปรสภาวะแวดล้อมโลแคล คุณได้เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้หรือไม่?

       ```
       setlocale(LC_MESSAGES, "")
       ```
     - หลังจากการเรียกใช้รูทีนย่อย **setlocale** คุณเรียกใช้รูทีนย่อย **catopen** ในลักษณะต่อไปนี้หรือไม่?

       ```
       catopen(catalog_name, NL_CAT_LOCALE)
       ```
     - คุณเรียกใช้รูทีนย่อย **catopen** พร้อมกับชื่อแค็ตตาล็อกที่ถูกต้อง หรือไม่?
     - ให้ดู "ฟังก์ชันข้อความ" ในหน้า 173 สำหรับข้อมูลเพิ่มเติม เกี่ยวกับข้อความที่แปลได้
2. โปรแกรมเปรียบเทียบสตริงข้อความหรือไม่?
   ถ้าใช่:
   - เปรียบเทียบสตริงเพื่อตรวจสอบความเท่าเทียมอย่างเดียวหรือไม่?
     ถ้าใช่:
     - ใช้รูทีนย่อย **strcmp** หรือ **strncmp**
     - ห้ามใช้รูทีนย่อย **strcoll** หรือ **strxfrm**
   - เปรียบเทียบสตริงเพื่อดูว่าสตริงใดเรียงลำดับอยู่ก่อนหน้าสตริงอื่น ตามที่กำหนด ในโลแคลปัจจุบันหรือไม่?
     ถ้าใช่:
     - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:

```
setlocale(LC_ALL, "")
```

- ใช้รูทีนย่อย **strcoll, strxfrm, wcscoll,** หรือ **wcsxfrm**
- ห้ามใช้รูทีนย่อย **strxfrm** หรือ **strncmp**

3. โปรแกรมแจงชื่อพาธของไฟล์หรือไม่?
   ถ้าใช่:
   - ถ้าค้นหา / (slash) ให้ใช้รูทีนย่อย **strchr**
   - ถ้าค้นหาอักขระ ระวังว่าชื่อไฟล์อาจมีอักขระ หลายไบต์ ในกรณีเช่นนั้น ให้เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้ แล้วใช้รูทีนย่อยการค้นหาที่เหมาะสม:

     ```
     setlocale(LC_ALL, "")
     ```

4. โปรแกรมใช้ชื่อระบบ เช่น ชื่อโหนด ชื่อผู้ใช้ ชื่อเครื่องพิมพ์ และชื่อคิวหรือไม่?
   ถ้าใช่:
   - ชื่อระบบอาจมีอักขระหลายไบต์
   - เมื่อต้องการระบุอักขระหลายไบต์ อันดับแรก เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้ แล้วใช้รูทีนย่อยที่เหมาะสมในไลบรารี

     ```
     setlocale(LC_ALL, "")
     ```

5. โปรแกรมใช้คุณสมบัติคลาสอักขระ เช่น ตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก และตัวอักษรหรือไม่?
   ถ้าใช่:
   - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:

     ```
     setlocale(LC_ALL, "")
     ```

   - อย่ากำหนดสมมุติฐานเกี่ยวกับคุณสมบัติอักขระ ให้ใช้ รูทีนย่อยระบบเพื่อกำหนดคุณสมบัติอักขระเสมอ
   - อักขระมีการจำกัดเฉพาะชุดโค้ดไบต์เดียวหรือไม่?
     ถ้าใช่:
     - ใช้รูทีนย่อย **ctype** อย่างใดอย่างหนึ่งดังนี้: **isalnum, isalpha, iscntrl, isdigit, isgraph, isprint, isspace,** หรือ **isxdigit**

       ถ้าไม่ อักขระอาจเป็นอักขระหลายไบต์ได้:
     - ใช้รูทีนย่อย **iswalnum, iswalpha, iswcntrl, iswdigit, iswgraph, iswlower, iswprint, iswpunct, iswspace, iswupper,** หรือ **iswxdigit**

6. โปรแกรมแปลงตัวพิมพ์ (ใหญ่หรือเล็ก) ของอักขระหรือไม่?
   ถ้าใช่:
   - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:

     ```
     setlocale(LC_ALL, "")
     ```

   - อักขระมีการจำกัดเฉพาะชุดโค้ดไบต์เดียวหรือไม่?
     ถ้าใช่:
     - ใช้รูทีนย่อย **conv** เหล่านี้: **_tolower, _toupper, tolower,** หรือ **toupper**

     ถ้าไม่ อักขระอาจเป็นอักขระหลายไบต์ได้:
     - ใช้รูทีนย่อย **towlower** หรือ **towupper**

7. โปรแกรมเก็บประวัติการเคลื่อนที่เคอร์เซอร์บนเทอร์มินัล tty หรือไม่?
   ถ้าใช่:
   - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
     ```
     setlocale(LC_ALL, "")
     ```
   - คุณอาจต้องการกำหนดความกว้างคอลัมน์ในการแสดงผลของอักขระ ใช้รูทีนย่อย **wcwidth** หรือ **wcswidth**
8. โปรแกรมทำ I/O อักขระหรือไม่?
   ถ้าใช่:
   - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
     ```
     setlocale(LC_ALL, "")
     ```
   - อักขระมีการจำกัดเฉพาะชุดโค้ดไบต์เดียวหรือไม่?
     ถ้าใช่:
     - ใช้ตระกูลรูทีนย่อยต่อไปนี้:
       - **fgetc, getc, getchar, getw**
       - **fgets, gets**
       - **fputc, putc, putchar, putw**
       - **printf, scanf**

     ถ้าไม่:
     - ใช้ตระกูลรูทีนย่อยต่อไปนี้:
       - **fgetwc, getwc, getwchar**
       - **fgetws, getws**
       - **fputwc, putwc, putwchar**
9. โปรแกรมดำเนินการผ่านอาร์เรย์ของอักขระหรือไม่?
   ถ้าใช่:
   - อาร์เรย์มีการจำกัดเฉพาะอักขระไบต์เดียวหรือไม่?
     ถ้าใช่:
     - ไม่ต้องการ `setlocale(LC_ALL, "")`
     - ถ้า p เป็นตัวชี้ไปยังอาร์เรย์นี้ของอักขระไบต์เดียว ให้ดำเนินการ ผ่านอาร์เรย์นี้โดยใช้ p++

   ถ้าไม่:
   - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
     ```
     setlocale(LC_ALL, "")
     ```
   - ใช้รูทีนย่อย **mblen** หรือ **wcslen**
10. โปรแกรมจำเป็นต้องทราบจำนวนไบต์สูงสุดที่ใช้ในการเข้าโค้ดอักขระภายในชุดโค้ดหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```

- ใช้แมโคร **MB_CUR_MAX**

11. โปรแกรมจัดรูปแบบวันที่หรือเวลาด้วยปริมาณตัวเลขหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - ใช้รูทีนย่อย **nl_langinfo** หรือ **localeconv** เพื่อให้ได้รับข้อมูลเฉพาะ โลแคล
    - ใช้รูทีนย่อย **strftime** หรือ **strptime**
12. โปรแกรมจัดรูปแบบปริมาณตัวเลขหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย
      ```
      setlocale
      ```
      ใน ลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - ใช้รูทีนย่อย **nl_langinfo** หรือ **localeconv** เพื่อให้ได้รับข้อมูลเฉพาะ โลแคล
    - ใช้คู่ของรูทีนย่อยต่อไปนี้ ตามความต้องการ: **printf, scanf**
13. โปรแกรมจัดรูปแบบปริมาณการเงินหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - ใช้รูทีนย่อย **nl_langinfo** หรือ **localeconv** เพื่อให้ได้รับข้อมูลเฉพาะ โลแคล
    - ใช้รูทีนย่อย **strfmon** เพื่อจัดรูปแบบจำนวนเงิน
14. โปรแกรมค้นหาสตริงหรือระบุตำแหน่งอักขระหรือไม่?
    ถ้าใช่:
    - คุณค้นหาอักขระไบต์เดียวในข้อความไบต์เดียวหรือไม่?
      - ไม่ต้องการ `setlocale(LC_ALL, "")`
      - ใช้รูทีนย่อยสตริง **libc** มาตรฐาน เช่น รูทีนย่อย **strchr**
    - คุณค้นหาอักขระในช่วง 0x00-0x3F (ช่วงจุดโค้ด เฉพาะ) หรือไม่?
    - ไม่ต้องการ `setlocale(LC_ALL, "")`
    - ใช้รูทีนย่อยสตริง **libc** มาตรฐาน เช่น รูทีนย่อย **strchr, strcspn, strpbrk, strrchr, strspn, strstr, strtok,** และ **memchr**
    - คุณค้นหาอักขระในช่วง 0x00-0xFF หรือไม่?
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - มีอยู่สองวิธีการดังนี้:
      ใช้รูทีนย่อย **mblen** เพื่อข้ามอักขระหลายไบต์ จากนั้น ในอักขระไบต์เดียวที่พบ ให้ตรวจสอบความเท่าเทียมกัน ให้ดู ไอเท็มรายการตรวจสอบ 2

หรือ

แปลงอักขระที่ค้นหา และสตริงที่ค้นหาเป็นรูปแบบอักขระ wide แล้วใช้รูทีนย่อยการค้นหาอักขระ wide

15. โปรแกรมทำการจับคู่รูปแบบนิพจน์ธรรมดาหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - ใช้รูทีนย่อย **regcomp, regexec** หรือ **regerror**
16. โปรแกรมขอให้ผู้ใช้ตอบรับ/ตอบปฏิเสธหรือไม่?
    ถ้าใช่:
    - เรียกใช้รูทีนย่อย **setlocale** ในลักษณะต่อไปนี้:
      ```
      setlocale(LC_ALL, "")
      ```
    - วางพร้อมต์ในแค็ตตาล็อกข้อความ ใช้รูทีนย่อย **catopen** และ **catgets** เพื่อดึงข้อมูลแค็ตตาล็อก และแสดงพร้อมต์
    - ใช้รูทีนย่อย **rpmatch** เพื่อจับคู่การตอบกลับของผู้ใช้
17. โปรแกรมใช้อักขระภาพวาดกล่องพิเศษหรือไม่?
    ถ้าใช่:
    - อย่าใช้อักขระภาพวาดกล่องเฉพาะชุดโค้ด
    - ให้ใช้อักขระภาพวาดกล่องและแอ็ตทริบิวต์ที่ระบุในไฟล์ **terminfo** แทน
18. โปรแกรมทำการประมวลผลเฉพาะวัฒนธรรมหรือเฉพาะโลแคล ที่ไม่ได้ระบุที่นี่หรือไม่?
    ถ้าใช่:
    - ส่งโมดูลเฉพาะวัฒนธรรมไปภายนอก อย่ากำหนดโมดูลเฉพาะวัฒนธรรม เป็นส่วนหนึ่งของโปรแกรมที่ดำเนินการได้
    - โหลดโมดูลที่รันไทม์โดยใช้รูทีนย่อยที่นำเสนอโดยระบบ เช่น รูทีนย่อย **load**
    - ถ้าระบบไม่มีรูทีนย่อยดังกล่าว ให้เชื่อมโยงรูทีนย่อยเฉพาะที่ แต่นำเสนอในลักษณะโมดูล

หลักการที่เกี่ยวข้อง:

"การตั้งค่าโลแคล" ในหน้า 16

โปรแกรมที่ทำให้เป็นโกลบอลทุกโปรแกรมต้องตั้งค่าโลแคลปัจจุบันโดยใช้ รูทีนย่อย **setlocale** รูทีนย่อยนี้ช่วยให้กระบวนการสามารถเปลี่ยนหรือเคียวรี่โลแคลปัจจุบันโดยการเข้าถึงฐานข้อมูลโลแคล

"ฟังก์ชันข้อความ" ในหน้า 173

คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ การจัดระเบียบนี้ช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้ เพื่อช่วยในภารกิจนี้ ฟังก์ชันข้อความจึงนำเสนอคำสั่งและรูทีนย่อย ต่างๆ

"รูทีนย่อยการจัดประเภทอักขระ wide" ในหน้า 31

รูทีนย่อยการจัดประเภทอักขระ wide ส่วนใหญ่ คล้ายกับรูทีนย่อยการจัดประเภทอักขระดั้งเดิม ยกเว้นว่ารูทีนย่อยการจัดประเภทอักขระ wide ดำเนินงานอาร์กิวเมนต์ชนิดข้อมูล **wchar_t** ที่ส่งผ่านเป็นอาร์กิวเมนต์ชนิดข้อมูล **wint_t**

ข้อมูลที่เกี่ยวข้อง:

หมวดหมู่ LC_COLLATE

รูทีนย่อย setlocale

รูทีนย่อย catopen
รูทีนย่อย strcmp
รูทีนย่อย strxfrm
รูทีนย่อย wcscoll
รูทีนย่อย wcsxfrml
รูทีนย่อย strlenl
รูทีนย่อย ctype
รูทีนย่อย iswalnum
รายชื่อรูทีนย่อย (A ถึง P)
รายชื่อรูทีนย่อย (Q ถึง Z)

AIXwindowsรายการตรวจสอบ

ไอเท็มรายการตรวจสอบที่เหลือเป็นไอเท็มเฉพาะสำหรับระบบ AIXwindows

1. โปรแกรมของคุณใช้ข้อมูลจำเพาะเกี่ยวกับชุดฟอนต์เพื่อให้เป็นอิสระ จากชุดโค้ดในแอ็พพลิเคชัน X หรือไม่?
2. ไคลเอ็นต์ของคุณใช้ป้ายชื่อ ปุ่ม หรือ widgets เฉพาะเอาต์พุตอื่นๆ เพื่อแสดงข้อความที่สามารถแปลได้หรือไม่?
   ถ้าใช่:
   - ให้เรียกใช้รูทีนย่อย ***XtSetLanguageProc** ในลักษณะดังต่อไปนี้:

     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```
   - ข้อความสามารถวางไว้ในแค็ตตาล็อกข้อความหรือไฟล์รีซอร์ส ที่โลคัลไลซ์ อย่างใดอย่างหนึ่ง ให้ดูไอเท็มรายการตรวจสอบ 1 หรือ 20 ตามลำดับ
   - เมื่อต้องการทำให้ widgets เป็นอิสระจากชุดโค้ด ให้ระบุฟอนต์ที่ใช้ ชุดฟอนต์
3. ไคลเอ็นต์ของคุณใช้ไฟล์รีซอร์ส X เพื่อกำหนดข้อความของป้ายชื่อ ปุ่ม หรือ widgets ข้อความหรือไม่?
   ถ้าใช่:
   - วางรีซอร์สทั้งหมดที่ต้องการการแปลไว้ในที่แห่งเดียว
   - พิจารณาการใช้แค็ตตาล็อกข้อความสำหรับสตริงข้อความ
   - อย่าใช้ชื่อสีที่แปล เนื่องจากชื่อสีถูกจำกัด เป็นชื่อที่เข้ารหัสอยู่ เฉพาะชื่อที่ใช้ได้หลายระบบเท่านั้นจะมีการเข้ารหัสใน ชุดอักขระ ที่ใช้ได้หลายระบบ
   - วางไฟล์รีซอร์สเฉพาะภาษาไว้ใน **/usr/lib/X11/%L/app-defaults/%N** โดยที่ **%L** คือชื่อของโลแคล เช่น fr_FR, และ **%N** คือ ชื่อของไคลเอ็นต์
4. อินพุตคีย์บอร์ดมีการโลคัลไลซ์โดยภาษาหรือไม่?
   ถ้าใช่:
   - ให้เรียกใช้รูทีนย่อย ***XtSetLanguageProc** ในลักษณะดังต่อไปนี้:

     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```
   - ใช้ **XmText** หรือ **XmTextField** widgets สำหรับอินพุตข้อความ ทั้งหมด

     อาร์กิวเมนต์บางตัวของ **XmText** widgets มีการกำหนด ในรูปแบบของความยาวอักขระแทนความยาวไบต์ ตำแหน่งเคอร์เซอร์ จะอยู่ในตำแหน่งอักขระ ไม่ใช่อยู่ในตำแหน่งไบต์
   - คุณกำลังใช้ **XmDrawingArea** widget เพื่อทำอินพุตที่โลคัลไลซ์ หรือไม่?

– ใช้รูทีนย่อยวิธีการอินพุตเพื่อทำอินพุตซึ่งกำลังประมวลผลอยู่ในภาษา ที่แตกต่างกัน

5. ไคลเอ็นต์ของคุณนำเสนอรายการหรือป้ายชื่อที่ประกอบด้วยข้อความซึ่งโลคัลไลซ์ จากไฟล์ผู้ใช้แทนที่จะโลคัลไลซ์จากไฟล์รีซอร์ส X หรือไม่?
   ถ้าใช่:
   - ให้เรียกใช้รูทีนย่อย ***XtSetLanguageProc*** ในลักษณะดังต่อไปนี้:
     ```
     XtSetLanguageProc(NULL, NULL, NULL);
     ```
   - ใช้รูทีนย่อย **XmStringCreateSimple** เพื่อสร้างชนิดข้อมูล **XmString** สำหรับข้อความที่โลคัลไลซ์ แม้ว่าสามารถใช้รูทีนย่อย **XmStringCreate** ได้ แต่แนะนำให้ใช้ **XmSTRING_DEFAULT_CHARSET**
   - เมื่อต้องการทำให้ widgets เป็นอิสระจากชุดโค้ด ให้ระบุฟอนต์โดยใช้ชุดฟอนต์ รีซอร์สฟอนต์ (ตัวอย่างเช่น ***fontList:** แทน) ในไฟล์ app-defaults ควรใช้รูปแบบตัวพิมพ์ใหญ่และคลาส แทนที่จะใช้รูปแบบตัวพิมพ์เล็ก (ตัวอย่างเช่น ***FontList:** แทน) ซึ่งช่วยให้ผู้จัดการเดสก์ท็อปสไตล์สามารถเลือกฟอนต์ของแอ็พพลิเคชัน ได้อย่างถูกต้อง
6. โปรแกรมของคุณดำเนินงานนำเสนอใดๆ (การวาด การพิมพ์ การจัดรูปแบบ หรือการแก้ไข Xlib) บนข้อความแบบสองทิศทางหรือไม่?
   ถ้าใช่:
   - ให้ใช้ **XmText** หรือ **XmTextField** ในไลบรารี Xm (Motif) Widgets เหล่านี้มีการเปิดใช้งานสำหรับข้อความแบบสองทิศทาง
   - ถ้าไม่สามารถใช้ไลบรารี Xm ได้ ให้ใช้รูทีนย่อยโครงร่าง เพื่อทำการจัดลำดับใหม่และการจัดรูปทรงบนข้อความ
   - จัดเก็บและเผยแพร่ข้อความในรูปแบบที่ชัดแจ้ง (โลจิคัล) ยูทิลีตีบางอย่าง (ตัวอย่างเช่น aixterm) สนับสนุนรูปแบบภาพของข้อความแบบสองทิศทาง แต่รูทีนย่อย multicultural ส่วนใหญ่ไม่สามารถประมวลผลรูปแบบภาพของข้อความแบบสองทิศทางได้

ถ้าคำตอบของไอเท็มด้านบนคือ ไม่ แสดงว่าโปรแกรมอาจไม่มีการพึ่งพา multicultural ในกรณีนี้ คุณอาจไม่ต้องการรูทีนย่อย การตั้งค่าโลแคล **setlocale** และรูทีนอุปกรณ์แค็ตตาล็อก **catopen** และ **catgets**

หลักการที่เกี่ยวข้อง:

"ฟังก์ชันข้อความ" ในหน้า 173

คุณจำเป็นต้องแยกข้อความออกจากโปรแกรมโดยการแสดงข้อความในรูปแบบของแค็ตตาล็อกข้อความที่โปรแกรมสามารถเข้าถึงเมื่อรันไทม์ การจัดระเบียบนี้ช่วยอำนวยความสะดวกในการแปลข้อความเป็นภาษาต่างๆ และทำให้ข้อความพร้อมสำหรับโปรแกรมตามโลแคลของผู้ใช้ เพื่อช่วยในภารกิจนี้ ฟังก์ชันข้อความจึงนำเสนอคำสั่งและรูทีนย่อย ต่างๆ

"วิธีอินพุต" ในหน้า 141

สำหรับแอ็พพลิเคชันที่จะรันในสภาวะแวดล้อมระหว่างประเทศที่ globalization เป็นพื้นฐาน จำเป็นต้องใช้วิธีการอินพุต วิธีการอินพุตคืออินเตอร์เฟสการเขียนโปรแกรมแอ็พพลิเคชัน (API) ที่ช่วยให้คุณสามารถจัดทำภาษา คีย์บอร์ด หรือชุดโค้ดเฉพาะที่ เป็นอิสระจากแอ็พพลิเคชัน

ข้อมูลที่เกี่ยวข้อง:

รูทีนย่อย IMAuxDraw callback

การสนับสนุนโครงร่าง (สองทิศทาง) ในไลบรารี Xm (Motif)

รูทีนย่อย setlocale

รูทีนย่อย catopen

รูทีนย่อย catgets

รายการของรูทีนการสนับสนุน multicultural

รูทีนย่อยการสนับสนุน multicultural ใช้สำหรับการจัดการข้อมูลเฉพาะโลแคล ดำเนินการกับอักขระแบบกว้าง และอักขระหลายไบต์ และการใช้นิพจน์ธรรมดา

หลักการที่เกี่ยวข้อง:

"รายการรูทีนย่อยของโลแคล"
ส่วนนี้จะแสดงรายการรูทีนย่อยที่นำเสนอเพื่อให้ได้รับ และประมวลผลข้อมูลเฉพาะโลแคล

"ชุดโค้ดสำหรับการสนับสนุน multicultural" ในหน้า 15
ส่วนนี้แนะนำโปรแกรมเมอร์ในการใช้รูทีนย่อยเมื่อ พัฒนาโปรแกรมที่ทำให้เป็นโกลบอลแบบเคลื่อนย้ายได้ ใช้ฟังก์ชัน Open Group, ISO/ANSI C, และ POSIX มาตรฐานเพื่อให้สามารถใช้ได้หลายระบบสูงสุด

รายการรูทีนย่อยของโลแคล

ส่วนนี้จะแสดงรายการรูทีนย่อยที่นำเสนอเพื่อให้ได้รับ และประมวลผลข้อมูลเฉพาะโลแคล

รูทีนย่อยที่นำเสนอเพื่อให้ได้รับและประมวลผลข้อมูล เฉพาะโลแคลมีดังต่อไปนี้:

รูทีนย่อย	คำอธิบาย
localeconv	ดึงข้อมูลระเบียบอิงโลแคลของโปรแกรมโลแคล
nl_langinfo	ส่งคืนข้อมูลเกี่ยวกับภาษาหรือพื้นที่ทางวัฒนธรรมในโปรแกรม โลแคล
rpmatch	กำหนดว่าการตอบกลับเป็นการตอบรับหรือตอบปฏิเสธ ในโลแคลปัจจุบัน
setlocale	เปลี่ยนหรือเคียวรี่โลแคลปัจจุบันของโปรแกรม

หลักการที่เกี่ยวข้อง:

"รายการของรูทีนการสนับสนุน multicultural"
รูทีนย่อยการสนับสนุน multicultural ใช้สำหรับการจัดการข้อมูลเฉพาะโลแคล ดำเนินการกับอักขระแบบกว้าง และอักขระหลายไบต์ และการใช้นิพจน์ธรรมดา

รายการของรูทีนย่อยการจัดรูปแบบเวลาและเงิน

รูทีนย่อย	คำอธิบาย
strfmon	จัดรูปแบบสตริงการเงินตามโลแคลปัจจุบัน
strftime	จัดรูปแบบเวลาและวันที่ตามโลแคลปัจจุบัน
strptime	แปลงสตริงอักขระเป็นค่าเวลาตาม โลแคลปัจจุบัน
wcsftime	แปลงเวลาและวันที่เป็นสตริงอักขระ wide ตาม โลแคลปัจจุบัน

รายการของรูทีนย่อยอักขระหลายไบต์

รูทีนย่อย	คำอธิบาย
mblen	กำหนดความยาวของอักขระหลายไบต์
mbstowcs	แปลงสตริงอักขระหลายไบต์เป็นสตริงอักขระ wide
mbtowc	แปลงอักขระหลายไบต์เป็นอักขระ wide

รายการของรูทีนย่อยอักขระ wide

รูทีนย่อยที่ประมวลผลอักขระในรูปแบบโค้ดกระบวนการมีดังต่อไปนี้:

รูทีนย่อย	คำอธิบาย
fgetwc	เรียกใช้อักขระ wide หรือคำจากอินพุตสตรีม
fgetws	เรียกใช้สตริงอักขระ wide จากสตรีม
fputwc	บันทึกอักขระ wide หรือคำลงในสตรีม
fputws	บันทึกสตริงอักขระ wide ลงในสตรีม
getwc	เรียกใช้อักขระ wide หรือคำจากอินพุตสตรีม
getwchar	เรียกใช้อักขระ wide หรือคำจากอินพุตสตรีม
getws	เรียกใช้สตริงอักขระ wide จากสตรีม
iswalnum	กำหนดว่าอักขระ wide เป็นตัวอักษรและตัวเลขหรือไม่
iswalpha	กำหนดว่าอักขระ wide เป็นตัวอักษรหรือไม่
iswcntrl	กำหนดว่าอักขระ wide เป็นอักขระควบคุมหรือไม่
iswctype	กำหนดคุณสมบัติของอักขระ wide
iswdigit	กำหนดว่าอักขระ wide เป็นตัวเลขหรือไม่
iswgraph	กำหนดว่าอักขระ wide (ไม่รวม "อักขระพื้นที่ว่าง") เป็นอักขระการพิมพ์หรือไม่
iswlower	กำหนดว่าอักขระ wide เป็นตัวพิมพ์เล็กหรือไม่
iswprint	กำหนดว่าอักขระ wide (รวมถึง "อักขระพื้นที่ว่าง") เป็นอักขระการพิมพ์หรือไม่
iswpunct	กำหนดว่าอักขระ wide เป็นอักขระวรรคตอนหรือไม่
iswspace	กำหนดว่าอักขระ wide เป็นพื้นที่ว่างเปล่าหรือไม่
iswupper	กำหนดว่าอักขระ wide เป็นตัวพิมพ์ใหญ่หรือไม่
iswxdigit	กำหนดว่าอักขระ wide เป็นตัวเลขฐานสิบหกหรือไม่
putwc	บันทึกอักขระ wide หรือคำลงในสตรีม
putwchar	บันทึกอักขระ wide หรือคำลงในสตรีม
putws	บันทึกสตริงอักขระ wide ลงในสตรีม
strcoll	เปรียบเทียบสองสตริงโดยใช้น้ำหนักการจัดเรียงในโลแคลปัจจุบัน
strxfrm	แปลงสตริงเป็นค่าการจัดเรียงโลแคล
towlower	แปลงอักขระ wide ตัวพิมพ์ใหญ่เป็นอักขระ wide ตัวพิมพ์เล็ก
towupper	แปลงอักขระ wide ตัวพิมพ์เล็กเป็นอักขระ wide ตัวพิมพ์ใหญ่
ungetwc	ออกใช้อักขระ wide บนสตรีม

รูทีนย่อย	คำอธิบาย
wcsid	ส่งคืน charsetID ของอักขระ wide
wcscat	เชื่อมต่อสตริงอักขระ wide
wcschr	ค้นหาอักขระ wide
wcscmp	เปรียบเทียบสตริงอักขระ wide
wcscoll	เปรียบเทียบน้ำหนักการจัดเรียงของสตริงอักขระ wide
wcscpy	คัดลอกสตริงอักขระ wide
wcscspn	ค้นหาสตริงอักขระ wide
wcslen	กำหนดจำนวนของอักขระในสตริงอักขระ wide
wcsncat	เชื่อมต่อหมายเลขที่ระบุของอักขระ wide
wcsncmp	เปรียบเทียบหมายเลขที่ระบุของอักขระ wide
wcsncpy	คัดลอกหมายเลขที่ระบุของอักขระ wide
wcspbrk	ระบุตำแหน่งการเกิดครั้งแรกของอักขระ wide ในสตริง อักขระ wide
wcsrchr	ระบุตำแหน่งการเกิดครั้งล่าสุดของอักขระ wide ในสตริง อักขระ wide
wcsspn	ส่งคืนจำนวนของอักขระ wide ในเซกเมนต์แรกเริ่ม ของสตริง
wcstod	แปลงสตริงอักขระ wide เป็นค่าจุด floating ความแม่นยำดับเบิล
wcstok	แบ่งสตริงอักขระ wide เป็นลำดับของสตริงอักขระ wide ที่แบ่งแยก
wcstol	แปลงสตริงอักขระ wide เป็นค่าจำนวนเต็มแบบยาว
wcstombs	แปลงลำดับของอักขระ wide เป็นลำดับของอักขระ หลายไบต์
wcstoul	แปลงสตริงอักขระ wide เป็นค่าจำนวนเต็มแบบยาว ที่ไม่มีเครื่องหมาย
wcswcs	ระบุตำแหน่งการเกิดครั้งแรกของลำดับอักขระ wide ในสตริง อักขระ wide
wcswidth	กำหนดความกว้างในการแสดงผลของสตริงอักขระ wide
wcsxfrm	แปลงสตริงอักขระ wide เป็นค่าที่แสดงถึงน้ำหนักการจัดเรียง อักขระ
wctomb	แปลงอักขระ wide เป็นอักขระหลายไบต์
wctype	เรียกใช้การจัดการชื่อคุณสมบัติที่ถูกต้องตามที่กำหนดไว้ใน โลแคลปัจจุบัน
wcwidth	กำหนดความกว้างในการแสดงผลของอักขระ wide

รายการของรูทีนย่อยไลบรารีโครงร่าง

รูทีนย่อยต่อไปนี้ของไลบรารีโครงร่าง (**libi18n.a**) แปลงสภาพข้อความสองทิศทางและขึ้นกับบริบทเป็นรูปแบบอื่น:

รูทีนย่อย	คำอธิบาย
layout_object_create	เริ่มต้นบริบทโครงร่าง
layout_object_free	ทำให้โครงสร้าง **LayoutObject** ว่าง
layout_object_editshape	แก้ไขรูปทรงของข้อความบริบท
layout_object_getvalue	เคียวรีค่าโครงร่างปัจจุบันของโครงสร้าง **LayoutObject**
layout_object_setvalue	ตั้งค่าโครงร่างของโครงสร้าง **LayoutObject**
layout_object_shapeboxchars	จัดรูปทรงอักขระกล่อง
layout_object_transform	แปลงข้อความตามค่าโครงร่างปัจจุบันของ โครงสร้าง **LayoutObject**

รายการของรูทีนย่อยฟังก์ชันข้อความ

โปรแกรมอำนวยความสะดวกข้อความประกอบด้วยมาตรฐานที่กำหนดรูทีนย่อย และ คำสั่ง และส่วนขยายที่เพิ่มมูลค่าผู้ผลิต เพื่อสนับสนุนแค็ตตาล็อกข้อความ ที่ทำให้เป็นภายนอก แอ็พพลิเคชันใช้แค็ตตาล็อกเหล่านี้เพื่อดึงข้อมูล และแสดงข้อความตามความต้องการ รูทีนย่อยโปรแกรมอำนวยความสะดวกข้อความต่อไปนี้ รับข้อความสำหรับแอ็พพลิเคชัน:

รูทีนย่อย	คำอธิบาย
catopen	เปิดแค็ตตาล็อก
catgets	เรียกใช้ข้อความจากแค็ตตาล็อก
catclose	ปิดแค็ตตาล็อก
strerror	แม็พหมายเลขข้อผิดพลาดกับสตริงข้อความแสดงข้อผิดพลาดที่เหมาะสม สำหรับโลแคลปัจจุบัน

รายการของรูทีนย่อยตัวแปลง

ในสภาพแวดล้อมสากล มักจำเป็นต้องแปลง ข้อมูลจากชุดโค้ดหนึ่งเป็นชุดโค้ดอื่น รูทีนย่อยตัวแปลง ที่ได้รับการสนับสนุนสำหรับวัตถุประสงค์นี้มีดังต่อไปนี้:

รูทีนย่อย	คำอธิบาย
iconv_open	ทำการเริ่มต้นที่จำเป็นเพื่อแปลงอักขระจากชุดโค้ดที่ระบุโดย พารามิเตอร์ **FromCode** เป็นชุดโค้ดที่ระบุโดย พารามิเตอร์ **ToCode**
iconv	เรียกใช้ฟังก์ชันตัวแปลงที่ใช้คำอธิบายซึ่งได้รับ จากรูทีนย่อย **iconv_open**
iconv_close	ปิดคำอธิบายการแปลงที่ระบุโดยตัวแปร **cd** และทำให้ใช้ได้อีกครั้งหนึ่ง
ccsidtocs	ส่งคืนชุดโค้ด

รายการของรูทีนย่อยวิธีการอินพุต

วิธีการอินพุตคือชุดของรูทีนย่อยที่แปล key strokes เป็นสตริงอักขระในชุดโค้ดที่ระบุโดยโลแคล รูทีนย่อย วิธีการอินพุตมีตรรกะสำหรับการประมวลผลอินพุตเฉพาะโลแคล และตัวควบคุมบนคีย์บอร์ด (ตัวอย่างเช่น Ctrl, Alt, Shift, Lock, และ Alt-Graphic) รูทีนย่อยที่ได้รับการสนับสนุนในวิธีการอินพุตนี้มีดังต่อไปนี้:

รูทีนย่อย	คำอธิบาย
IMAIXMapping	แปลคู่ของพารามิเตอร์ *KeySymbol* และ *State* เป็นสตริงและส่งคืนตัวชี้ไปยังสตริงนั้น
IMAuxCreate	บอกให้แอ็พพลิเคชันโปรแกรมสร้างพื้นที่เสริม
IMAuxDestroy	แจ้ง callback ให้ทำลายความรู้เกี่ยวกับพื้นที่ เสริม
IMAuxDraw	บอกให้แอ็พพลิเคชันโปรแกรมวาดพื้นที่เสริม
IMAuxHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนพื้นที่เสริม
IMBeep	บอกให้แอ็พพลิเคชันโปรแกรมส่งเสียงบีป
IMClose	ปิดวิธีการอินพุต
IMCreate	สร้างอินสแตนซ์ของวิธีการอินพุตเฉพาะขึ้นหนึ่งรายการ

รูทีนย่อย	คำอธิบาย
IMDestroy	ทำลายอินสแตนซ์ของวิธีการอินพุต
IMFilter	ตรวจสอบว่าเหตุการณ์คีย์บอร์ดมีการใช้โดยวิธีการอินพุต สำหรับการประมวลผลภายในหรือไม่
IMFreeKeymap	ทำให้รีซอร์สที่จัดสรรโดยรูทีนย่อย **IMInitialzieKeymap** ว่าง
IMIndicatorDraw	บอกให้แอ็พพลิเคชันโปรแกรมวาดตัวบ่งชี้
IMIndicatorHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนตัวบ่งชี้
IMInitialize	เริ่มต้นวิธีการอินพุตสำหรับภาษาเฉพาะ
IMInitializeKeymap	เริ่มต้นวิธีการอินพุตสำหรับภาษาเฉพาะ
IMIoctl	ทำการควบคุมหรือเคียวรี่หลายอย่างเกี่ยวกับ วิธีการอินพุต
IMLookupString	แม็พคู่สัญลักษณ์คีย์บอร์ด/คำสั่งกับสตริงที่ผู้ใช้ กำหนด
IMProcessAuxiliary	แจ้งวิธีการอินพุตของอินพุตสำหรับพื้นที่เสริม
IMQueryLanguage	ตรวจสอบว่าภาษาที่ระบุได้รับการสนับสนุนหรือไม่
IMSimpleMapping	แปลคู่ของพารามิเตอร์ *KeySymbol* และ *State* เป็นสตริงและส่งคืนตัวชี้ไปยังสตริงนั้น
IMTextCursor	ตั้งค่าตำแหน่งเคอร์เซอร์ที่แสดงผลใหม่
IMTextDraw	ขอให้แอ็พพลิเคชันโปรแกรมวาดสตริงถัดไป
IMTextHide	บอกให้แอ็พพลิเคชันโปรแกรมซ่อนพื้นที่ข้อความ
IMTextStart	แจ้งแอ็พพลิเคชันโปรแกรมให้ทราบถึงความยาวของพื้นที่ว่าง การแก้ไขก่อนหน้า

รายการของรูทีนย่อยนิพจน์ธรรมดา

รูทีนย่อยการจัดการนิพจน์ธรรมดามีดังต่อไปนี้:

รูทีนย่อย	คำอธิบาย
regcomp	คอมไพล์นิพจน์ธรรมดาสำหรับการเปรียบเทียบโดยรูทีนย่อย **regexec**
regerror	ส่งคืนข้อความแสดงข้อผิดพลาดที่เหมาะสมสำหรับโลแคลปัจจุบัน ซึ่งสอดคล้องกับโค้ดระบุความผิดพลาดที่ส่งคืนโดยรูทีนย่อย **regcomp** หรือ **regexec**
regexec	เปรียบเทียบสตริงกับนิพจน์ธรรมดาที่คอมไพล์จาก การเรียกไปยังรูทีนย่อย **regcomp** ครั้งก่อนหน้านี้
regfree	ทำให้หน่วยความจำที่จัดสรรโดยการเรียกไปยังรูทีนย่อย **regcomp** ครั้งก่อนหน้านี้ว่าง

หลักการที่เกี่ยวข้อง:

"ชุดโค้ดสำหรับการสนับสนุน multicultural" ในหน้า 15

ส่วนนี้แนะนำโปรแกรมเมอร์ในการใช้รูทีนย่อยเมื่อ พัฒนาโปรแกรมที่ทำให้เป็นโกลบอลแบบเคลื่อนย้ายได้ ใช้ฟังก์ชัน Open Group, ISO/ANSI C, และ POSIX มาตรฐานเพื่อให้สามารถใช้ได้หลายระบบสูงสุด

แม็พอักขระ

ส่วนนี้มีการแสดงข้อความของแม็พ อักขระ

ส่วนนี้มีการแสดงข้อความของแม็พอักขระ ซึ่งอธิบายใน "ชุดโค้ดสำหนับการสนับสนุน multicultural" ในหน้า 55

ชุดโค้ด ISO

ส่วนนี้อธิบายชุดโค้ด ISO

ISO8859-1

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-1

ตารางที่ 17. ชุดโค้ด ISO8859-1

ชื่อสัญลักษณ์	ค่า Hex
No break space	A0
เครื่องหมายตกใจกลับด้าน	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายสกุลเงิน	A4
เครื่องหมายเงินเยน	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
เครื่องหมายลิขสิทธิ์	A9
ตัวบ่งชี้ลำดับที่ไม่เน้น	AA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Macron	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
Acute accent	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
เครื่องหมาย Cedilla	B8
ยกกำลังหนึ่ง	B9

ตารางที่ 17. ชุดโค้ด ISO8859-1 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวบ่งชี้ลำดับที่เน้น	BA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านขวา	BB
Vulgar fraction one quarter	BC
Vulgar fraction one half	BD
Vulgar fraction three quarters	BE
เครื่องหมายคำถามกลับด้าน	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี grave	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มี tilde	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
อักษรละติน E ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลกซ์	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี grave	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
อักษรละติน I ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	CF
ตัวอักษรละตินตัวพิมพ์เล็ก eth	D0
ตัวอักษรละตินตัวพิมพ์เล็ก n ที่มี tilde	D1
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี grave	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7

ตารางที่ 17. ชุดโค้ด ISO8859-1 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี grave	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Y ที่มี acute	DD
ตัวอักษรละตินตัวพิมพ์เล็ก thorn	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี grave	E0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี tilde	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี circumflex	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี grave	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี diaeresis	EF
ละตินตัวอักษร eth ตัวพิมพ์เล็ก	F0
อักษร n ตัวเล็กภาษาละตินที่มีเครื่องหมายทิลเดอ	F1
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี grave	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5

ตารางที่ 17. ชุดโค้ด ISO8859-1 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี grave	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี acute	FD
ละตินตัวอักษร thorn ตัวพิมพ์เล็ก	FE
ละตินตัวอักษร y ตัวพิมพ์เล็กที่มี diaeresis	FF

ISO8859-2

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-2

ตารางที่ 18. ชุดโค้ด ISO8859-2

ชื่อสัญลักษณ์	ค่า Hex
No break space	A0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี ogonek	A1
Bleve	A2
ตัวอักษรตัวพิมพ์ใหญ่ L ที่มี stroke	A3
เครื่องหมายสกุลเงิน	A4
ตัวอักษรละตินตัวพิมพ์ใหญ่ L ที่มี caron	A5
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี acute	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี caron	A9
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี cedilla	AA
ตัวอักษรละตินตัวพิมพ์ใหญ่ T ที่มี caron	AB
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี acute	AC
Soft hyphen	AD
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี caron	AE
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มีจุด ด้านบน	AF

ตารางที่ 18. ชุดโค้ด ISO8859-2 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
สัญลักษณ์องศา	B0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี ogenek	B1
Ogenek	B2
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี stroke	B3
Acute accent	B4
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี caron	B5
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี acute	B6
Caron	B7
เครื่องหมาย Cedilla	B8
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี caron	B9
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี cedilla	BA
ตัวอักษรละตินตัวพิมพ์เล็ก T ที่มี caron	BB
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี acute	BC
Double acute accent	BD
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี caron	BE
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มีจุดด้านบน	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ R ที่มี acute	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี breve	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ตัวอักษรละตินตัวพิมพ์ใหญ่ L ที่มี acute	C5
ตัวอักษรละตินตัวพิมพ์ใหญ่ C ที่มี acute	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ C ที่มี caron	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี ogonek	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี caron	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD

ตารางที่ 18. ชุดโค้ด ISO8859-2 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
ตัวอักษรละตินตัวพิมพ์ใหญ่ D ที่มี caron	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ D ที่มี stroke	D0
ตัวอักษรละตินตัวพิมพ์ใหญ่ N ที่มี acute	D1
ตัวอักษรละตินตัวพิมพ์ใหญ่ N ที่มี caron	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี double acute	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ตัวอักษรละตินตัวพิมพ์ใหญ่ R ที่มี caron	D8
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มีวงแหวน ด้านบน	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี double acute	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Y ที่มี acute	DD
ตัวอักษรละตินตัวพิมพ์ใหญ่ T ที่มี cedilla	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี acute	E0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี breve	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี acute	E5
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี acute	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี caron	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี ogonek	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB

ตารางที่ 18. ชุดโค้ด ISO8859-2 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี caron	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก D ที่มี caron	EF
ตัวอักษรละตินตัวพิมพ์เล็ก D ที่มี stroke	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี acute	F1
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี caron	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี double acute	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี caron	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มีวงแหวน ด้านบน	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี double acute	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี acute	FD
ตัวอักษรละตินตัวพิมพ์เล็ก T ที่มี cedilla	FE
จุดด้านบน	FF

ISO8859-4

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-4

ตารางที่ 19. ชุดโค้ด ISO8859-4

ชื่อสัญลักษณ์	ค่า Hex
No break space	A0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี ogonek	A1
ตัวอักษรละตินตัวพิมพ์เล็ก kra	A2
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี cedilla	A3
เครื่องหมายสกุลเงิน	A4
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี tilde	A5

ตารางที่ 19. ชุดโค้ด ISO8859-4 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี cedilla	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี caron	A9
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี macron	AA
ตัวอักษรละตินตัวพิมพ์เล็ก G ที่มี cedilla	AB
ตัวอักษรละตินตัวพิมพ์ใหญ่ T ที่มี stroke	AC
Soft hyphen	AD
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี caron	AE
Macron	AF
สัญลักษณ์องศา	B0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี ogonek	B1
Ogonek	B2
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี cedilla	B3
Acute accent	B4
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	B5
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี cedilla	B6
Caron	B7
เครื่องหมาย Cedilla	B8
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี caron	B9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี macron	BA
ตัวอักษรละตินตัวพิมพ์เล็ก G ที่มี cedilla	BB
ตัวอักษรละตินตัวพิมพ์เล็ก T ที่มี stroke	BC
ตัวอักษรละตินตัวพิมพ์เล็ก eng	BD
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี caron	BE
ตัวอักษรละตินตัวพิมพ์เล็ก eng	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี macron	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มี tilde	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4

ตารางที่ 19. ชุดโค้ด ISO8859-4 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก ae	C6
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี ogonek	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ C ที่มี caron	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี ogonek	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มีจุด ด้านบน	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี macron	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ D ที่มี stroke	D0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี cedilla	D1
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี macron	D2
ตัวอักษรละตินตัวพิมพ์เล็ก K ที่มี cedilla	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี ogonek	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี tilde	DD
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี macron	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี macron	E0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี tilde	E3

ตารางที่ 19. ชุดโค้ด ISO8859-4 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก ae	E6
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี ogonek	E7
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี caron	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี ogonek	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มีจุดด้านบน	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี macron	EF
ตัวอักษรละตินตัวพิมพ์เล็ก D ที่มี stroke	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี cedilla	F1
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี macron	F2
ตัวอักษรละตินตัวพิมพ์เล็ก K ที่มี cedilla	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี ogonek	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี tilde	FD
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี macron	FE
จุดด้านบน	FF

ISO8859-5

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-5

ตารางที่ 20. ชุดโค้ด ISO8859-5

ชื่อสัญลักษณ์	**ค่า Hex**
No break space	A0
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก io	A1
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก dje	A2
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก gje	A3
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ukrainian ie	A4
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก dze	A5
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก byelorussian-ukrainian I	A6
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก yi	A7
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก je	A8
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก lje	A9
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก nje	AA
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก tshe	AB
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก kje	AC
Soft hyphen	AD
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก short U	AE
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก dzhe	AF
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก A	B0
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก be	B1
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ve	B2
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ghe	B3
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก de	B4
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ie	B5
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก zhe	B6
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ze	B7
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก I	B8
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก short I	B9
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ka	BA
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก el	BB
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก em	BC
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก en	BD
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก O	BE

ตารางที่ 20. ชุดโค้ด ISO8859-5 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก pe	BF
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก er	C0
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก es	C1
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก te	C2
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก U	C3
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ef	C4
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ha	C5
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก tse	C6
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก che	C7
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก sha	C8
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก shcha	C9
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก hard sign	CA
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก yeru	CB
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก soft sign	CC
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก E	CD
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก tu	CE
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ya	CF
ตัวอักษรตัวเล็กซิริลลิก A	D0
ตัวอักษรตัวเล็กซิริลลิก be	D1
ตัวอักษรตัวเล็กซิริลลิก ve	D2
ตัวอักษรตัวเล็กซิริลลิก ghe	D3
ตัวอักษรตัวเล็กซิริลลิก de	D4
ตัวอักษรตัวเล็กซิริลลิก ie	D5
ตัวอักษรตัวเล็กซิริลลิก zhe	D6
ตัวอักษรตัวเล็กซิริลลิก ze	D7
ตัวอักษรตัวเล็กซิริลลิก I	D8
ตัวอักษรตัวเล็กซิริลลิก short I	D9
ตัวอักษรตัวเล็กซิริลลิก ka	DA
ตัวอักษรตัวเล็กซิริลลิก el	DB
ตัวอักษรตัวเล็กซิริลลิก em	DC
ตัวอักษรตัวเล็กซิริลลิก en	DD

ตารางที่ 20. ชุดโค้ด ISO8859-5 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
ตัวอักษรตัวเล็กซิริลลิก O	DE
ตัวอักษรตัวเล็กซิริลลิก pe	DF
ตัวอักษรตัวเล็กซิริลลิก er	E0
ตัวอักษรตัวเล็กซิริลลิก es	E1
ตัวอักษรตัวเล็กซิริลลิก te	E2
ตัวอักษรตัวเล็กซิริลลิก U	E3
ตัวอักษรตัวเล็กซิริลลิก ef	E4
ตัวอักษรตัวเล็กซิริลลิก ha	E5
ตัวอักษรตัวเล็กซิริลลิก tse	E6
ตัวอักษรตัวเล็กซิริลลิก che	E7
ตัวอักษรตัวเล็กซิริลลิก sha	E8
ตัวอักษรตัวเล็กซิริลลิก shcha	E9
ตัวอักษรตัวเล็กซิริลลิก hard sign	EA
ตัวอักษรตัวเล็กซิริลลิก yeru	EB
ตัวอักษรตัวเล็กซิริลลิก soft sign	EC
ตัวอักษรตัวเล็กซิริลลิก E	ED
ตัวอักษรตัวเล็กซิริลลิก yu	EE
ตัวอักษรตัวเล็กซิริลลิก ta	EF
Numero sign	F0
ตัวอักษรตัวเล็กซิริลลิก io	F1
ตัวอักษรตัวเล็กซิริลลิก dje	F2
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก gje	F3
ตัวอักษรตัวเล็กซิริลลิก ukrainian ie	F4
ตัวอักษรตัวเล็กซิริลลิก dze	F5
ตัวอักษรตัวเล็กซิริลลิก byelorussian-ukrainian	F6
ตัวอักษรตัวเล็กซิริลลิก yi	F7
ตัวอักษรตัวเล็กซิริลลิก je	F8
ตัวอักษรตัวเล็กซิริลลิก lje	F9
ตัวอักษรตัวเล็กซิริลลิก nje	FA
ตัวอักษรตัวเล็กซิริลลิก tshe	FB
ตัวอักษรตัวเล็กซิริลลิก kje	FC

ตารางที่ 20. ชุดโค้ด ISO8859-5 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
เครื่องหมายการเลือก	FD
ตัวอักษรตัวเล็กซิริลลิก short U	FE
ตัวอักษรตัวเล็กซิริลลิก dzhe	FF

ISO8859-6

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-6

ตารางที่ 21. ชุดโค้ด ISO8859-6

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
เครื่องหมายสกุลเงิน	A4
คอมมาอารบิก	AC
Soft hyphen	AD
เซมิโคลอนอารบิก	BB
เครื่องหมายคำถามอารบิก	BF
ตัวอักษรอารบิก hamza	C1
ตัวอักษรอารบิก alef ที่มี madda ข้างบน	C2
ตัวอักษรอารบิก alef ที่มี hamza ข้างบน	C3
ตัวอักษรอารบิก waw ที่มี hamza ข้างบน	C4
ตัวอักษรอารบิก alef ที่มี hamza ข้างใต้	C5
ตัวอักษรอารบิก yeh ที่มี hamza ข้างบน	C6
ตัวอักษรอารบิก alef	C7
ตัวอักษรอารบิก beh	C8
ตัวอักษรอารบิก teh marbuta	C9
ตัวอักษรอารบิก teh	CA
ตัวอักษรอารบิก theh	CB
ตัวอักษรอารบิก jeem	CC
ตัวอักษรอารบิก hah	CD
ตัวอักษรอารบิก khah	CE
ตัวอักษรอารบิก dal	CF
ตัวอักษรอารบิก thal	D0
ตัวอักษรอารบิก reh	D1

ตารางที่ 21. ชุดโค้ด ISO8859-6 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรอารบิก zain	D2
ตัวอักษรอารบิก seen	D3
ตัวอักษรอารบิก sheen	D4
ตัวอักษรอารบิก sad	D5
ตัวอักษรอารบิก dad	D6
ตัวอักษรอารบิก tah	D7
ตัวอักษรอารบิก zah	D8
ตัวอักษรอารบิก ain	D9
ตัวอักษรอารบิก ghain	DA
ตัวอักษรอารบิก tatweel	E0
ตัวอักษรอารบิก feh	E1
ตัวอักษรอารบิก qaf	E2
ตัวอักษรอารบิก kaf	E3
ตัวอักษรอารบิก lam	E4
ตัวอักษรอารบิก meem	E5
ตัวอักษรอารบิก noon	E6
ตัวอักษรอารบิก heh	E7
ตัวอักษรอารบิก waw	E8
ตัวอักษรอารบิก alef maksura	E9
ตัวอักษรอารบิก yeh	EA
ตัวอักษรอารบิก fathatan	EB
ตัวอักษรอารบิก dammatan	EC
ตัวอักษรอารบิก kasratan	ED
ตัวอักษรอารบิก fatha	EE
ตัวอักษรอารบิก damma	EF
ตัวอักษรอารบิก kasra	F0
ตัวอักษรอารบิก shadda	F1
ตัวอักษรอารบิก sukun	F2

ISO8859-7

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-7

ตารางที่ 22. ISO8859-7

ชื่อสัญลักษณ์	ค่า Hex
No break space	A0
เครื่องหมายคำพูดขีดเดียวด้านซ้าย	A1
เครื่องหมายอัญประกาศเดี่ยวด้านขวา	A2
เครื่องหมายปอนด์	A3
เครื่องหมายเงินยูโร	A4
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
เครื่องหมายลิขสิทธิ์	A9
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
แถบแนวนอน	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
Greek tonos	B4
Greek dialytika tonos	B5
Greek capital letter alpha with tonos	B6
จุดกลาง	B7
Greek capital letter epsilon with tonos	B8
Greek capital letter eta with tonos	B9
Greek capital letter iota with tonos	BA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านขวา	BB
Greek capital letter omicron with tonos	BC
Vulgar fraction one half	BD
Greek capital letter upsilon with tonos	BE
Greek capital letter omega with tonos	BF
Greek small letter iota with dialytika and tonos	C0

ตารางที่ 22. ISO8859-7 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
ตัวอักษรกรีกตัวพิมพ์ใหญ่ alpha	C1
ตัวอักษรกรีกตัวพิมพ์ใหญ่ beta	C2
ตัวอักษรกรีกตัวพิมพ์ใหญ่ gamma	C3
ตัวอักษรกรีกตัวพิมพ์ใหญ่ delta	C4
ตัวอักษรกรีกตัวพิมพ์ใหญ่ epsilon	C5
ตัวอักษรกรีกตัวพิมพ์ใหญ่ zeta	C6
ตัวอักษรกรีกตัวพิมพ์ใหญ่ eta	C7
ตัวอักษรกรีกตัวพิมพ์ใหญ่ theta	C8
ตัวอักษรกรีกตัวพิมพ์ใหญ่ iota	C9
ตัวอักษรกรีกตัวพิมพ์ใหญ่ kappa	CA
ตัวอักษรกรีกตัวพิมพ์ใหญ่ lambda	CB
ตัวอักษรกรีกตัวพิมพ์ใหญ่ mu	CC
ตัวอักษรกรีกตัวพิมพ์ใหญ่ nu	CD
ตัวอักษรกรีกตัวพิมพ์ใหญ่ xi	CE
ตัวอักษรกรีกตัวพิมพ์ใหญ่ omicron	CF
ตัวอักษรกรีกตัวพิมพ์ใหญ่ pi	D0
ตัวอักษรกรีกตัวพิมพ์ใหญ่ rho	D1
ตัวอักษรกรีกตัวพิมพ์ใหญ่ sigma	D3
ตัวอักษรกรีกตัวพิมพ์ใหญ่ tau	D4
ตัวอักษรกรีกตัวพิมพ์ใหญ่ upsilon	D5
ตัวอักษรกรีกตัวพิมพ์ใหญ่ phi	D6
ตัวอักษรกรีกตัวพิมพ์ใหญ่ chi	D7
ตัวอักษรกรีกตัวพิมพ์ใหญ่ psi	D8
ตัวอักษรกรีกตัวพิมพ์ใหญ่ omega	D9
ตัวอักษรกรีกตัวพิมพ์ใหญ่ iota ที่มี dialytika	DA
ตัวอักษรกรีกตัวพิมพ์ใหญ่ upsilon ที่มี dialytika	DB
ตัวอักษรกรีกตัวพิมพ์เล็ก alpha ที่มี tonos	DC
ตัวอักษรกรีกตัวพิมพ์เล็ก epsilon ที่มี tonos	DD
ตัวอักษรกรีกตัวพิมพ์เล็ก eta ที่มี tonos	DE
ตัวอักษรกรีกตัวพิมพ์เล็ก iota ที่มี tonos	DF
ตัวอักษรกรีกตัวพิมพ์เล็ก upsilon ที่มี dialytika และ tonos	E0

ตารางที่ 22. ISO8859-7 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรกรีกตัวพิมพ์เล็ก alpha	E1
ตัวอักษรกรีกตัวพิมพ์เล็ก beta	E2
ตัวอักษรกรีกตัวพิมพ์เล็ก gamma	E3
ตัวอักษรกรีกตัวพิมพ์เล็ก delta	E4
ตัวอักษรกรีกตัวพิมพ์เล็ก epsilon	E5
ตัวอักษรกรีกตัวพิมพ์เล็ก zeta	E6
ตัวอักษรกรีกตัวพิมพ์เล็ก eta	E7
ตัวอักษรกรีกตัวพิมพ์เล็ก theta	E8
ตัวอักษรกรีกตัวพิมพ์เล็ก iota	E9
ตัวอักษรกรีกตัวพิมพ์เล็ก kappa	EA
ตัวอักษรกรีกตัวพิมพ์เล็ก lambda	EB
ตัวอักษรกรีกตัวพิมพ์เล็ก mu	EC
ตัวอักษรกรีกตัวพิมพ์เล็ก nu	ED
ตัวอักษรกรีกตัวพิมพ์เล็ก xi	EE
ตัวอักษรกรีกตัวพิมพ์เล็ก omicron	EF
ตัวอักษรกรีกตัวพิมพ์เล็ก pi	F0
ตัวอักษรกรีกตัวพิมพ์เล็ก rho	F1
ตัวอักษรกรีกตัวพิมพ์เล็ก final sigma	F2
ตัวอักษรกรีกตัวพิมพ์เล็ก sigma	F3
ตัวอักษรกรีกตัวพิมพ์เล็ก tau	F4
ตัวอักษรกรีกตัวพิมพ์เล็ก upsilon	F5
ตัวอักษรกรีกตัวพิมพ์เล็ก phi	F6
ตัวอักษรกรีกตัวพิมพ์เล็ก chi	F7
ตัวอักษรกรีกตัวพิมพ์เล็ก psi	F8
ตัวอักษรกรีกตัวพิมพ์เล็ก omega	F9
ตัวอักษรกรีกตัวพิมพ์เล็ก iota ที่มี dialytika	FA
ตัวอักษรกรีกตัวพิมพ์เล็ก upsilon ที่มี dialytika	FB
ตัวอักษรกรีกตัวพิมพ์เล็ก omicron ที่มี tonos	FC
ตัวอักษรกรีกตัวพิมพ์เล็ก upsilon ที่มี tonos	FD
ตัวอักษรกรีกตัวพิมพ์เล็ก omega ที่มี tonos	FE

ISO8859-8

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-8

ตารางที่ 23. ISO8859-8

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายสกุลเงิน	A4
เครื่องหมายเงินเยน	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
เครื่องหมายลิขสิทธิ์	A9
สัญลักษณ์เครื่องหมายคูณ	AA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Overline	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
Acute accent	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
เครื่องหมาย Cedilla	B8
ยกกำลังหนึ่ง	B9
เครื่องหมายหาร	BA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านขวา	BB
Vulgar fraction one quarter	BC

ตารางที่ 23. ISO8859-8 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Vulgar fraction one half	BD
Vulgar fraction three quarters	BE
เส้นคู่ล่าง	DF
ตัวอักษรฮีบรู alef	EO
ตัวอักษรฮีบรู bet	E1
ตัวอักษรฮีบรู gimel	E2
ตัวอักษรฮีบรู dalet	E3
ตัวอักษรฮีบรู he	E4
ตัวอักษรฮีบรู vav	E5
ตัวอักษรฮีบรู zayin	E6
ตัวอักษรฮีบรู het	E7
ตัวอักษรฮีบรู tet	E8
ตัวอักษรฮีบรู yod	E9
ตัวอักษรฮีบรู final kaf	EA
ตัวอักษรฮีบรู kaf	EB
ตัวอักษรฮีบรู lamed	EC
ตัวอักษรฮีบรู final mem	ED
ตัวอักษรฮีบรู mem	EE
ตัวอักษรฮีบรู final nun	EF
ตัวอักษรฮีบรู nun	F0
ตัวอักษรฮีบรู samekh	F1
ตัวอักษรฮีบรู ayin	F2
ตัวอักษรฮีบรู final pe	F3
ตัวอักษรฮีบรู pe	F4
ตัวอักษรฮีบรู final tsadi	F5
ตัวอักษรฮีบรู tsadi	F6
ตัวอักษรฮีบรู qof	F7
ตัวอักษรฮีบรู resh	F8
ตัวอักษรฮีบรู shin	F9
ตัวอักษรฮีบรู tav	FA

ISO8859-9

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-9

ตารางที่ 24. ชุดโค้ด ISO8859-9

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
เครื่องหมายตกใจกลับด้าน	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายสกุลเงิน	A4
เครื่องหมายเงินเยน	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
เครื่องหมายลิขสิทธิ์	A9
ตัวบ่งชี้ลำดับที่ไม่เน้น	AA
Left-pointing double quotation mark	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Macron	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
Acute accent	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
เครื่องหมาย Cedilla	B8
ยกกำลังหนึ่ง	B9
ตัวบ่งชี้ลำดับที่เน้น	BA
เครื่องหมายคำพูดมุมคู่ขวา	BB

ตารางที่ 24. ชุดโค้ด ISO8859-9 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Vulgar fraction one quarter	BC
Vulgar fraction one half	BD
Vulgar fraction three quarters	BE
Inverted question mark	BF
Latin capital letter A with grave	C0
Latin capital letter A with acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มี tilde	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
อักษรละติน E ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลกซ์	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี grave	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
อักษรละติน I ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ G ที่มี breve	D0
ละตินตัวอักษร N ตัวพิมพ์ใหญ่ที่มี tilde	D1
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี grave	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี grave	D9

ตารางที่ 24. ชุดโค้ด ISO8859-9 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มีจุด ด้านบน	DD
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี cedilla	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี grave	E0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี tilde	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี circumflex	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี grave	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี diaeresis	EF
ตัวอักษรละตินตัวพิมพ์เล็ก G ที่มี breve	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี tilde	F1
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี grave	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8

ตารางที่ 24. ชุดโค้ด ISO8859-9 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี grave	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	FD
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี cedilla	FE
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี diaeresis	FF

ISO8859-15

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ ISO8859-15

ตารางที่ 25. ISO8859-15

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
เครื่องหมายตกใจกลับด้าน	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายเงินยูโร	A4
เครื่องหมายเงินเยน	A5
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี caron	A6
สัญลักษณ์ส่วน	A7
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี caron	A8
เครื่องหมายลิขสิทธิ์	A9
ตัวบ่งชี้ลำดับที่ไม่เน้น	AA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Macron	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2

ตารางที่ 25. ISO8859-15 (ต่อ)

ชื่อสัญลักษณ์	**ค่า Hex**
สามตัวยก	B3
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี caron	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี caron	B8
ยกกำลังหนึ่ง	B9
ตัวบ่งชี้ลำดับที่เน้น	BA
Right-pointing double angle quotation marks	BB
ตัวอักษรละตินตัวพิมพ์ใหญ่ oe	BC
อักษร oe ตัวเล็กติดกันภาษาละติน	BD
ตัวอักษรละตินตัวพิมพ์ใหญ่ Y ที่มี diaeresis	BE
เครื่องหมายคำถามกลับด้าน	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี grave	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มี tilde	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
ละตินตัวอักษร W ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี grave	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
อักษรละติน I ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	CF
ตัวอักษรละตินตัวพิมพ์เล็ก eth	D0

ตารางที่ 25. ISO8859-15 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ละตินตัวอักษร N ตัวพิมพ์ใหญ่ที่มี tilde	D1
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี grave	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี grave	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Y ที่มี acute	DD
ตัวอักษรละตินตัวพิมพ์เล็ก thorn	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี grave	EO
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี tilde	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี circumflex	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี grave	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี diaeresis	EF

ตารางที่ 25. ISO8859-15 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ละตินตัวอักษร eth ตัวพิมพ์เล็ก	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี tilde	F1
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี grave	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี grave	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี acute	FD
ละตินตัวอักษร thorn ตัวพิมพ์เล็ก	FE
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี diaeresis	FF

ชุดโค้ด IBM PC

ส่วนนี้จะอธิบายชุดโค้ด IBM PC

IBM-856

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-856

ตารางที่ 26. ชุดโค้ด IBM-856

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรฮีบรู alef	80
ตัวอักษรฮีบรู bet	81
ตัวอักษรฮีบรู gimel	82
ตัวอักษรฮีบรู dalet	83
ตัวอักษรฮีบรู he	84
ตัวอักษรฮีบรู vav	85
ตัวอักษรฮีบรู zayin	86
ตัวอักษรฮีบรู het	87

ตารางที่ 26. ชุดโค้ด IBM-856 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรฮีบรู tet	88
ตัวอักษรฮีบรู yod	89
ตัวอักษรฮีบรู final kaf	8A
ตัวอักษรฮีบรู kaf	8B
ตัวอักษรฮีบรู lamed	8C
ตัวอักษรฮีบรู final mem	8D
ตัวอักษรฮีบรู mem	8E
ตัวอักษรฮีบรู final nun	8F
ตัวอักษรฮีบรู nun	90
ตัวอักษรฮีบรู samekh	91
ตัวอักษรฮีบรู ayin	92
ตัวอักษรฮีบรู final pe	93
ตัวอักษรฮีบรู pe	94
ตัวอักษรฮีบรู final tsadi	95
ตัวอักษรฮีบรู tsadi	96
ตัวอักษรฮีบรู qof	97
ตัวอักษรฮีบรู resh	98
ตัวอักษรฮีบรู shin	99
ตัวอักษรฮีบรู tav	9A
เครื่องหมายปอนด์	9C
สัญลักษณ์เครื่องหมายคูณ	9E
เครื่องหมายที่ลงทะเบียน	A9
เครื่องหมายไม่	AA
Vulgar fraction one half	AB
Vulgar fraction one quarter	AC
เครื่องหมายคำพูดมุมคู่ ซ้าย	AE
เครื่องหมายคำพูดมุมคู่ ขวา	AF
เฉดจาก	B0
เฉดปานกลาง	B1
เฉดเข้ม	B2
Box drawings light vertical	B3

ตารางที่ 26. ชุดโค้ด IBM-856 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Box drawings light vertical and left	B4
เครื่องหมายลิขสิทธิ์	B8
Box drawings double vertival and left	B9
Box drawings double vertical	BA
Box drawings double down and left	BB
Box drawings double up and left	BC
สัญลักษณ์สกุลเงินเซนต์	BD
เครื่องหมายเงินเยน	BE
Box drawings light down and left	BF
Box drawings light up and right	C0
Box drawings light up and horizontal	C1
Box drawings light down and horizontal	C2
Box drawings light vertical and right	C3
Box drawings light horizontal	C4
Box drawings light vertical and horizontal	C5
Box drawings double up and right	C8
Box drawings double down and right	C9
Box drawings double up and horizontal	CA
Box drawings double down and horizontal	CB
Box drawings double vertical and right	CC
Box drawings double horizontal	CD
Box drawings double vertical and horizontal	CE
เครื่องหมายสกุลเงิน	CF
Box drawings light up and left	D9
Box drawings light down and right	DA
บล็อกเต็ม	DB
บล็อกครึ่งล่าง	DC
แท่งหัก	DD
บล็อกครึ่งบน	DF
เครื่องหมายไมโคร	E6
Overline	EE

ตารางที่ 26. ชุดโค้ด IBM-856 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Acute accent	EF
Soft hyphen	F0
เครื่องหมายบวก-ลบ	F1
เส้นคู่ล่าง	F2
Vulgar fraction three quarters	F3
เครื่องหมายย่อหน้า	F4
สัญลักษณ์ส่วน	F5
เครื่องหมายหาร	F6
เครื่องหมาย Cedilla	F7
สัญลักษณ์องศา	F8
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	F9
จุดกลาง	FA
ยกกำลังหนึ่ง	FB
สามตัวยก	FC
ยกกำลังสอง	FD
สี่เหลี่ยมสีดำ	FE
No-break space	FF

IBM-921

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-921

ตารางที่ 27. ชุดโค้ด IBM-921

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
เครื่องหมายคำพูดขีดคู่ด้านขวา	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายเงินยูโร	A4
เครื่องหมายคำพูดขีดคู่ low-9	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	A8

ตารางที่ 27. ชุดโค้ด IBM-921 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
เครื่องหมายลิขสิทธิ์	A9
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี cedilla	AA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
ตัวอักษรละตินตัวพิมพ์เล็ก AE	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
เครื่องหมายคำพูดขีดคู่ด้านซ้าย	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	B8
ยกกำลังหนึ่ง	B9
ตัวอักษรละตินตัวพิมพ์เล็ก R ที่มี cedilla	BA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านขวา	BB
Vulgar fraction one quarter	BC
Vulgar fraction one half	BD
Vulgar fraction three quarters	BE
ตัวอักษรละตินตัวพิมพ์เล็ก AE	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี ogonek	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี ogonek	C1
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี macron	C2
ตัวอักษรละตินตัวพิมพ์ใหญ่ C ที่มี acute	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี ogonek	C6

ตารางที่ 27. ชุดโค้ด IBM-921 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี macron	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ C ที่มี caron	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี acute	CA
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มีจุด ด้านบน	CB
ตัวอักษรละตินตัวพิมพ์เล็ก G ที่มี cedilla	CC
ตัวอักษรละตินตัวพิมพ์เล็ก K ที่มี cedilla	CD
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี macron	CE
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี cedilla	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี caron	D0
ตัวอักษรละตินตัวพิมพ์ใหญ่ N ที่มี acute	D1
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี cedilla	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี macron	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี ogonek	D8
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี stroke	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี acute	DA
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี macron	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มีจุด ด้านบน	DD
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี caron	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี ogonek	E0
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี ogonek	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี macron	E2
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี acute	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5

ตารางที่ 27. ชุดโค้ด IBM-921 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี ogonek	E6
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี macron	E7
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี caron	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี acute	EA
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มีจุดด้านบน	EB
ตัวอักษรละตินตัวพิมพ์เล็ก G ที่มี cedilla	EC
ตัวอักษรละตินตัวพิมพ์เล็ก K ที่มี cedilla	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี macron	EE
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี cedilla	EF
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี caron	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี acute	F1
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี cedilla	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี macron	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี ogonek	F8
ตัวอักษรละตินตัวพิมพ์เล็ก L ที่มี stroke	F9
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี macron	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มีจุดด้านบน	FD
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี caron	FE
เครื่องหมายอัญประกาศเดี่ยวด้านขวา	FF

IBM-922

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-922

ตารางที่ 28. ชุดโค้ด IBM-922

ชื่อสัญลักษณ์	ค่า Hex
No break space	A0
เครื่องหมายตกใจกลับด้าน	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายเงินยูโร	A4
เครื่องหมายเงินเยน	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
สัญลักษณ์ diaeresis (สัญลักษณ์ที่ใส่เหนือสระตัวที่สอง เพื่อแสดงว่าออกเสียงแยกจากสระตัวแรก)	A8
เครื่องหมายลิขสิทธิ์	A9
ตัวบ่งชี้ลำดับที่ไม่เน้น	AA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Macron	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
Acute accent	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
เครื่องหมาย Cedilla	B8
ยกกำลังหนึ่ง	B9
ตัวบ่งชี้ลำดับที่เน้น	BA
เครื่องหมายคำพูดมุมคู่ชี้ไปด้านขวา	BB
Vulgar fraction one quarter	BC
Vulgar fraction one half	BD

ตารางที่ 28. ชุดโค้ด IBM-922 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Vulgar fraction three quarters	BE
เครื่องหมายคำถามกลับด้าน	BF
Latin capital letter A with grave	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มี tilde	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
อักษรละติน E ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลกซ์	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี grave	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
อักษรละติน I ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ S ที่มี caron	D0
ละตินตัวอักษร N ตัวพิมพ์ใหญ่ที่มี tilde	D1
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี grave	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายทิลเดอ	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี grave	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB

ตารางที่ 28. ชุดโค้ด IBM-922 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ Y ที่มี acute	DD
ตัวอักษรละตินตัวพิมพ์ใหญ่ Z ที่มี caron	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี grave	E0
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี tilde	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี circumflex	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี grave	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี diaeresis	EF
ตัวอักษรละตินตัวพิมพ์เล็ก S ที่มี caron	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี tilde	F1
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี grave	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี tilde	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี grave	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA

ตารางที่ 28. ชุดโค้ด IBM-922 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี acute	FD
ตัวอักษรละตินตัวพิมพ์เล็ก Z ที่มี caron	FE
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี diaeresis	FF

IBM-1046

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-1046

ตารางที่ 29. IBM-1046

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรอาหรับ alef ที่มี hamza ด้านล่าง ฟอร์มสุดท้าย	80
สัญลักษณ์เครื่องหมายคูณ	81
เครื่องหมายหาร	82
ตัวอักษร seen ส่วนแรกของ ฟอร์มสุดท้าย	83
ตัวอักษรอาหรับ sheen สำหรับแรก ของฟอร์มสุดท้าย	84
ตัวอักษรอาหรับ sad สำหรับแรก ของฟอร์มสุดท้าย	85
ตัวอักษรอาหรับ dadfirst ส่วนของ ฟอร์มสุดท้าย	86
Arabic tatweel with fathatan above	87
บล็อกเต็ม	89
Box drawings light vertical	8A
Box drawings light horizontal	8B
Box drawings light down and left	8C
Box drawings light down and right	8D
Box drawings light up and right	8E
Box drawings light up and left	8F
Arabic damma medial form	90
Arabic kasra medial form	91
Arabic shadda medial form	92
Arabic sukun medial form	93
Arabic fatha medial form	94
Arabic letter yeh with hamza above final form	95

ตารางที่ 29. IBM-1046 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Arabic letter alef maksura final form	96
Arabic letter yeh initial form	97
Arabic letter yeh final form	98
Arabic letter ghain final form	99
Arabic letter ghain initial form	9A
Arabic letter ghain medial form	9B
Arabic ligature lam with alef with madda above final form	9C
Arabic ligature lam with alef with hamza above final form	9D
Arabic ligature lam with alef with hamza below final form	9E
Arabic ligature lam with alef final form	9f
No-break space	A0
Arabic letter alef with madda above after lam	A1
Arabic letter alef with hamza above after lam	A2
Arabic letter alef with hamza below after lam	A3
เครื่องหมายสกุลเงิน	A4
Arabic letter alef after lam	A5
Arabic letter yeh with hamza above initial form	A6
Arabic letter beh with initial form	A7
Arabic letter teh with initial form	A8
Arabic letter theh with initial form	A9
Arabic letter jeem with initial form	AA
Arabic letter hah with initial form	AB
คอมมาอารบิก	AC
Soft hyphen	AD
Arabic letter khan initial form	AE
Arabic letter seen initial form	AF
Arabic-indic digit zero	B0
Arabic-indic digit one	B1
Arabic-indic digit two	B2
Arabic-indic digit three	B3
Arabic-indic digit four	B4

ตารางที่ 29. IBM-1046 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Arabic-indic digit five	B5
Arabic-indic digit six	B6
Arabic-indic digit seven	B7
Arabic-indic digit eight	B8
Arabic-indic digit nine	B9
Arabic letter sheen initial form	BA
เซมิโคลอนอารบิก	BB
Arabic letter sad initial form	BC
Arabic letter dad initial form	BD
Arabic letter ain initial form	BE
เครื่องหมายคำถามอารบิก	BF
Arabic letter ain initial form	C0
ตัวอักษรอารบิก hamza	C1
ตัวอักษรอารบิก alef ที่มี madda ข้างบน	C2
ตัวอักษรอารบิก alef ที่มี hamza ข้างบน	C3
ตัวอักษรอารบิก waw ที่มี hamza ข้างบน	C4
ตัวอักษรอารบิก alef ที่มี hamza ข้างใต้	C5
ตัวอักษรอารบิก yeh ที่มี hamza ข้างบน	C6
ตัวอักษรอารบิก alef	C7
ตัวอักษรอารบิก beh	C8
ตัวอักษรอารบิก teh marbuta	C9
ตัวอักษรอารบิก teh	CA
ตัวอักษรอารบิก theh	CB
ตัวอักษรอารบิก jeem	CC
ตัวอักษรอารบิก hah	CD
ตัวอักษรอารบิก khah	CE
ตัวอักษรอารบิก dal	CF
ตัวอักษรอารบิก thal	D0
ตัวอักษรอารบิก reh	D1
ตัวอักษรอารบิก zain	D2
ตัวอักษรอารบิก seen	D3

ตารางที่ 29. IBM-1046 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรอารบิก sheen	D4
ตัวอักษรอารบิก sad	D5
ตัวอักษรอารบิก dad	D6
ตัวอักษรอารบิก tah	D7
ตัวอักษรอารบิก zah	D8
ตัวอักษรอารบิก ain	D9
ตัวอักษรอารบิก ghain	DA
Arabic letter ain medial form	DB
Arabic letter alef with madda above final form	DC
Arabic letter alef with hamza above final form	DD
Arabic letter alef with final form	DE
Arabic letter feh initial form	DF
Arabic tatweel	E0
ตัวอักษรอารบิก feh	E1
ตัวอักษรอารบิก qaf	E2
ตัวอักษรอารบิก kaf	E3
ตัวอักษรอารบิก lam	E4
ตัวอักษรอารบิก meem	E5
ตัวอักษรอารบิก noon	E6
ตัวอักษรอารบิก heh	E7
ตัวอักษรอารบิก waw	E8
ตัวอักษรอารบิก alef maksura	E9
ตัวอักษรอารบิก yeh	EA
Arabic fathatan	EB
Arabic dammatan	EC
Arabic kasratan	ED
Arabic fatha	EE
Arabic damma	EF
Arabic kasra	F0
Arabic shadda	F1
Arabic sukun	F2

ตารางที่ 29. IBM-1046 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Arabic letter qar initial form	F3
Arabic letter kaf initial form	F4
Arabic letter lam initial form	F5
Arabic kasseh	F6
Arabic ligature lam with alef with madda above isolated form	F7
Arabic ligature lam with alef with hamza above isolated form	F8
Arabic ligature lam with alef with madda below isolated form	F9
Arabic ligature lam with alef isolated form	FA
Arabic letter meem initial form	FB
Arabic letter noon initial form	FC
Arabic letter heh initial form	FD
Arabic letter heh final form	FE
เครื่องหมายเงินยูโร	FF

IBM-1124

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-1124

ตารางที่ 30. ชุดโค้ด IBM-1124

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก io	A1
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก dje	A2
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ghe ที่มี upturn	A3
Cyrillic capital letter ukrainian ie	A4
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก dze	A5
Cyrillic capital letter byelorussian-ukranian i	A6
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก yi	A7
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก je	A8
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก lje	A9
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก nje	AA
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก tshe	AB
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก kje	AC

ตารางที่ 30. ชุดโค้ด IBM-1124 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Soft hyphen	AD
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก short U	AE
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก dzhe	AF
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก A	B0
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก be	B1
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ve	B2
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ghe	B3
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก de	B4
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ie	B5
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก zhe	B6
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ze	B7
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก I	B8
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก short I	B9
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ka	BA
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก el	BB
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก em	BC
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก en	BD
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก O	BE
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก pe	BF
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก er	C0
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก es	C1
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก te	C2
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก U	C3
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ef	C4
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก ha	C5
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก tse	C6
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก che	C7
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก sha	C8
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก shcha	C9
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก hard sign	CA
ตัวอักษรตัวพิมพ์ใหญ่ซิริลลิก yeru	CB

ตารางที่ 30. ชุดโค้ด IBM-1124 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก soft sign	CC
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก E	CD
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก yu	CE
ตัวอักษรตัวพิมพ์ใหญ่ซีริลลิก ya	CF
ตัวอักษรตัวเล็กซีริลลิก A	D0
ตัวอักษรตัวเล็กซีริลลิก be	D1
ตัวอักษรตัวเล็กซีริลลิก ve	D2
ตัวอักษรตัวเล็กซีริลลิก ghe	D3
ตัวอักษรตัวเล็กซีริลลิก de	D4
ตัวอักษรตัวเล็กซีริลลิก ie	D5
ตัวอักษรตัวเล็กซีริลลิก zhe	D6
ตัวอักษรตัวเล็กซีริลลิก ze	D7
ตัวอักษรตัวเล็กซีริลลิก I	D8
ตัวอักษรตัวเล็กซีริลลิก short I	D9
ตัวอักษรตัวเล็กซีริลลิก ka	DA
ตัวอักษรตัวเล็กซีริลลิก el	DB
ตัวอักษรตัวเล็กซีริลลิก em	DC
ตัวอักษรตัวเล็กซีริลลิก en	DD
ตัวอักษรตัวเล็กซีริลลิก O	DE
ตัวอักษรตัวเล็กซีริลลิก pe	DF
ตัวอักษรตัวเล็กซีริลลิก er	E0
ตัวอักษรตัวเล็กซีริลลิก es	E1
ตัวอักษรตัวเล็กซีริลลิก te	E2
ตัวอักษรตัวเล็กซีริลลิก u	E3
ตัวอักษรตัวเล็กซีริลลิก ef	E4
ตัวอักษรตัวเล็กซีริลลิก ha	E5
ตัวอักษรตัวเล็กซีริลลิก tse	E6
ตัวอักษรตัวเล็กซีริลลิก che	E7
ตัวอักษรตัวเล็กซีริลลิก sha	E8
ตัวอักษรตัวเล็กซีริลลิก shcha	E9
ตัวอักษรตัวเล็กซีริลลิก hard sign	EA

ตารางที่ 30. ชุดโค้ด IBM-1124 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรตัวเล็กซิริลลิก yeru	EB
ตัวอักษรตัวเล็กซิริลลิก soft sign	EC
ตัวอักษรตัวเล็กซิริลลิก E	ED
ตัวอักษรตัวเล็กซิริลลิก yu	EE
ตัวอักษรตัวเล็กซิริลลิก ya	EF
Numero sign	F0
ตัวอักษรตัวเล็กซิริลลิก io	F1
ตัวอักษรตัวเล็กซิริลลิก dje	F2
ตัวอักษรตัวเล็กซิริลลิก ghe ที่มี upturn	F3
ตัวอักษรตัวเล็กซิริลลิก ukrainian ie	F4
ตัวอักษรตัวเล็กซิริลลิก dze	F5
ตัวอักษรตัวเล็กซิริลลิก byelorussian-ukrainian	F6
ตัวอักษรตัวเล็กซิริลลิก yi	F7
ตัวอักษรตัวเล็กซิริลลิก je	F8
ตัวอักษรตัวเล็กซิริลลิก lje	F9
ตัวอักษรตัวเล็กซิริลลิก nje	FA
ตัวอักษรตัวเล็กซิริลลิก tshe	FB
ตัวอักษรตัวเล็กซิริลลิก kje	FC
สัญลักษณ์ส่วน	FD
ตัวอักษรตัวเล็กซิริลลิก short u	FE
ตัวอักษรตัวเล็กซิริลลิก dzhe	FF

IBM-1129

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ IBM-1129

ตารางที่ 31. ชุดโค้ด IBM-1129

ชื่อสัญลักษณ์	ค่า Hex
No-break space	A0
เครื่องหมายตกใจกลับด้าน	A1
สัญลักษณ์สกุลเงินเซนต์	A2
เครื่องหมายปอนด์	A3
เครื่องหมายเงินยูโร	A4

ตารางที่ 31. ชุดโค้ด IBM-1129 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
เครื่องหมายเงินเยน	A5
แท่งหัก	A6
สัญลักษณ์ส่วน	A7
Latin small ligature OE	A8
เครื่องหมายลิขสิทธิ์	A9
ตัวบ่งชี้ลำดับที่ไม่เน้น	AA
เครื่องหมายคำพูดมุมคู่ซ้าย	AB
เครื่องหมายไม่	AC
Soft hyphen	AD
เครื่องหมายที่ลงทะเบียน	AE
Macron	AF
สัญลักษณ์องศา	B0
เครื่องหมายบวก-ลบ	B1
ยกกำลังสอง	B2
สามตัวยก	B3
ตัวพิมพ์ใหญ่ละติน Y ที่มี diaeresis	B4
เครื่องหมายไมโคร	B5
เครื่องหมายย่อหน้า	B6
จุดกลาง	B7
ละติน ligature OE ตัวพิมพ์ใหญ่	B8
ยกกำลังหนึ่ง	B9
ตัวบ่งชี้ลำดับที่เน้น	BA
เครื่องหมายคำพูดมุมคู่ขวา	BB
Vulgar fraction one quarter	BC
Vulgar fraction one half	BD
Vulgar fraction three quarters	BE
เครื่องหมายคำถามกลับด้าน	BF
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี grave	C0
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี acute	C1
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	C2

ตารางที่ 31. ชุดโค้ด IBM-1129 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์ใหญ่ A ที่มี breve	C3
ตัวอักษร A ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายแสดงการออกเสียงแยกจากสระตัวแรก	C4
ละตินตัวอักษร A ตัวพิมพ์ใหญ่ที่มีวงแหวนอยู่ข้างบน	C5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	C6
ละตินตัวอักษร C ตัวพิมพ์ใหญ่ที่มี cedilla	C7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	C8
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี acute	C9
อักษรละติน E ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลกซ์	CA
ละตินตัวอักษร E ตัวพิมพ์ใหญ่ที่มี diaeresis	CB
Combining grave accent	CC
ตัวอักษรละตินตัวพิมพ์ใหญ่ I ที่มี acute	CD
ละตินตัวอักษร I ตัวพิมพ์ใหญ่ที่มีเครื่องหมายกำกับเสียงรูปหมวก	CE
อักษรละติน I ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	CF
ตัวอักษรละตินตัวพิมพ์ใหญ่ D ที่มี stroke	D0
ละตินตัวอักษร N ตัวพิมพ์ใหญ่ที่มี tilde	D1
Combining hook above	D2
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี acute	D3
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายเซอร์คัมเฟลก	D4
ตัวอักษรละตินตัวพิมพ์ใหญ่ O ที่มี horn	D5
อักษรละติน O ตัวใหญ่ที่มีเครื่องหมายไดเอเรซิส	D6
สัญลักษณ์เครื่องหมายคูณ	D7
ละตินตัวอักษร O ตัวพิมพ์ใหญ่ที่มี stroke	D8
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี grave	D9
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี acute	DA
ตัวอักษร U ตัวพิมพ์ใหญ่แบบละตินที่มีเครื่องหมายกำกับเสียงรูปหมวก	DB
ละตินตัวอักษร U ตัวพิมพ์ใหญ่ที่มี diaeresis	DC
ตัวอักษรละตินตัวพิมพ์ใหญ่ U ที่มี horn	DD
Combining tilde	DE
ตัวอักษรละตินตัวพิมพ์เล็ก sharp S	DF
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี grave	E0

ตารางที่ 31. ชุดโค้ด IBM-1129 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี acute	E1
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี circumflex	E2
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี breve	E3
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มี diaeresis	E4
ตัวอักษรละตินตัวพิมพ์เล็ก A ที่มีวงแหวน ด้านบน	E5
ตัวอักษรละตินตัวพิมพ์เล็ก AE	E6
ตัวอักษรละตินตัวพิมพ์เล็ก C ที่มี cedilla	E7
ตัวอักษรละตินตัวพิมพ์ใหญ่ E ที่มี grave	E8
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี acute	E9
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี circumflex	EA
ตัวอักษรละตินตัวพิมพ์เล็ก E ที่มี diaeresis	EB
Combining acute accent	EC
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี acute	ED
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี circumflex	EE
ตัวอักษรละตินตัวพิมพ์เล็ก I ที่มี diaeresis	EF
ตัวอักษรละตินตัวพิมพ์เล็ก D ที่มี stroke	F0
ตัวอักษรละตินตัวพิมพ์เล็ก N ที่มี tilde	F1
การรวมจุดด้านล่าง	F2
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี acute	F3
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี circumflex	F4
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี horn	F5
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี diaeresis	F6
เครื่องหมายหาร	F7
ตัวอักษรละตินตัวพิมพ์เล็ก O ที่มี stroke	F8
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี grave	F9
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี acute	FA
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี circumflex	FB
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี diaeresis	FC
ตัวอักษรละตินตัวพิมพ์เล็ก U ที่มี horn	FD
เครื่องหมาย Dong	FE
ตัวอักษรละตินตัวพิมพ์เล็ก Y ที่มี diaeresis	FF

TIS-620

ข้อมูลต่อไปนี้อธิบายชุดโค้ดสำหรับ TIS-620

ตารางที่ 32. ชุดโค้ด TIS-620

ชื่อสัญลักษณ์	ค่า Hex
Thai character ko kai	A1
Thai character kho khai	A2
Thai character kho khuat	A3
Thai character kho khwai	A4
Thai character kho khon	A5
Thai character kho rakhang	A6
Thai character ngo ngu	A7
Thai character cho chan	A8
Thai character cho ching	A9
Thai character cho chang	AA
Thai character so so	AB
Thai character cho choe	AC
Thai character yo ying	AD
Thai character do chada	AE
Thai character to patak	AF
Thai character tho than	B0
Thai character tho nangmontho	B1
Thai character tho phuthao	B2
Thai character no nen	B3
Thai character do dek	B4
Thai character to tao	B5
Thai character tho thung	B6
Thai character tho thahan	B7
Thai character tho thong	B8
Thai character no nu	B9
Thai character bo baimai	BA
Thai character po pla	BB
Thai character pho phung	BC
Thai character fo fa	BD

ตารางที่ 32. ชุดโค้ด TIS-620 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Thai character pho phan	BE
Thai character fo fan	BF
Thai character pho samphao	C0
Thai character mo ma	C1
Thai character yo yak	C2
Thai character ro rua	C3
Thai character ru	C4
Thai character lo ling	C5
Thai character lu	C6
Thai character wo waen	C7
Thai character so sala	C8
Thai character so rusi	C9
Thai character so sua	CA
Thai character ho hip	CB
Thai character lo chula	CC
Thai character o ang	CD
Thai character ho nokhuk	CE
Thai character paiyannoi	CF
Thai character sara a	D0
Thai character mai han-akat	D1
Thai character sara aa	D2
Thai character sara am	D3
Thai character sara i	D4
Thai character sara ii	D5
Thai character sara ue	D6
Thai character sara uee	D7
Thai character sara u	D8
Thai character uu	D9
Thai character phinthu	DA
Thai currency symbol baht	DF
Thai character sara e	E0

ตารางที่ 32. ชุดโค้ด TIS-620 (ต่อ)

ชื่อสัญลักษณ์	ค่า Hex
Thai character sara ae	E1
Thai character sara O	E2
Thai character sara ai maimuan	E3
Thai character sara ai maimalai	E4
Thai character lakkhangyao	E5
Thai character maiyamok	E6
Thai character maitaikhu	E7
Thai character mai ek	E8
Thai character mai tho	E9
Thai character mai tri	EA
Thai character mai chattawa	EB
Thai character thanthakhat	EC
Thai character nikhahit	ED
Thai character yamakkan	EE
Thai character fongman	EF
Thai digit zero	F0
Thai digit one	F1
Thai digit two	F2
Thai digit three	F3
Thai digit four	F4
Thai digit five	F5
Thai digit six	F6
Thai digit seven	F7
Thai digit eight	F8
Thai digit nine	F9
Thai character angkhankhu	FA
Thai character khomut	FB

ตัวอย่างโปรแกรมการสนับสนุน Multicultural

ส่วนนี้มีส่วนของโปรแกรมตัวอย่าง `my_example.c` ซึ่งแสดงการทำให้เป็นโกลบอลผ่านโปรแกรมมิงที่ไม่ขึ้นกับชุดโค้ด

หลักการที่เกี่ยวข้อง:

“ความเป็นอิสระของชุดโค้ด” ในหน้า 13
ระบบต้องใช้ข้อมูลบางอย่างเกี่ยวกับชุดโค้ดเพื่อ สื่อสารกับสภาพแวดล้อมภายนอก ข้อมูลนี้ถูกซ่อนไว้โดยรูทีนย่อยไลบรารีที่เป็นอิสระจากชุดโค้ด (ไลบรารี globalization) รูทีนย่อย เหล่านี้ส่งผ่านข้อมูลไปยังฟังก์ชันที่พึ่งพาชุดโค้ด เนื่องจากรูทีนย่อย multicultural จัดการข้อมูลชุดโค้ดที่จำเป็น ดังนั้นคุณจึงไม่ต้องมีความรู้เกี่ยวกับชุดโค้ดอย่างชัดแจ้งเมื่อคุณเขียนโปรแกรมที่ประมวลผลอักขระ เทคนิคการเขียนโปรแกรมนี้เรียกว่า *ความ เป็นอิสระของชุดโค้ด*

ไฟล์ต้นฉบับข้อความสำหรับ my_example

ตัวอย่างไฟล์ต้นฉบับขอความสำหรับยูทิลีตี my_example มีให้ที่นี้ หมายเหตุว่าเรากำหนดเพียงหนึ่งชุดและสามข้อความ ในแค็ตตาล็อกนี้สำหรับวัตถุประสงค์ในการสาธิตเท่านั้น แค็ตตาล็อกปกติ มีข้อความดังกล่าวหลายข้อความ

ต่อไปนี้เป็นไฟล์ต้นฉบับข้อความสำหรับ my_example, my_example.msg

```
$quote "
$set MS_MY_EXAMPLE
CANTOPEN        "my_example: cannot open %s\n"
BYTECNT         "number of bytes: %d\n"
CHARCNT         "number of characters: %d
```

การสร้างไฟล์ส่วนหัวข้อความสำหรับ my_example

เมื่อต้องการสร้างแค็ตตาล็อกรันไทม์ ให้ใช้คำสั่ง **runcat** ดังนี้:

```
runcat my_example my_example.msg
```

ซึ่งจะสร้างไฟล์ส่วนหัว **my_example_msg.h** ดังแสดงในส่วนต่อไปนี้ โปรดสังเกตว่า mnemonic ของชุดคือ MS_MY_EXAMPLE และ mnemonics ของข้อคความคือ CANTOPEN, BYTECNT , และ CHARCNT ตัวช่วยจำเหล่านี้ใช้ในโปรแกรมต่างๆ ในภาคผนวกนี้

```
/*
** ไฟล์ส่วนหัว: my_example_msg.h เป็นดังนี้:
*/

#ifndef _H_MY_EXAMPLE_MSG
#define _H_MY_EXAMPLE_MSG
#include <limits.h>
#include <nl_types.h>
#define MF_MY_EXAMPLE "my_example.cat"

/* ต่อไปนี้ถูกสร้างขึ้นจาก my_example.msg */


/* คำนิยามสำหรับชุด MS_MY_EXAMPLE */
#define MS_MY_EXAMPLE 1

#define CANTOPEN 1
#define BYTECNT  2
#define CHARCNT  3

#endif
```

เวอร์ชันอิสระของชุดโค้ด Single-source, single-path

คำว่า *single-source single-path* หมายถึงหนึ่ง พาธในหนึ่งแอ็พพลิเคชันที่จะใช้เพื่อประมวลผลทั้งชุดโค้ดไบต์เดียว และหลายไบต์ เมธอด single source single path นี้ลด **ifdef** ทั้งหมดสำหรับการทำให้เป็นโกลบอล อักขระทั้งหมดมีการจัดการ ในวิธีเดียวกัน ไม่ว่าอักขระนั้นเป็นสมาชิกของชุดโค้ดไบต์เดียวหรือ หลายไบต์

Single-source single-path ควรใช้ แต่อาจลดประสิทธิภาพการทำงาน ดังนั้น จึงไม่แนะนำวิธีนี้สำหรับโปรแกรมทั้งหมด อาจมีบางโปรแกรม ที่ไม่ได้รับการผลจากการลดประสิทธิภาพการทำงาน เมื่อโปรแกรมถูกทำให้เป็นโกลบอล แบบสมบูรณ์ ในกรณีเหล่านั้น ให้ใช้เมธอด single-source single-path

เวอร์ชันที่ทำให้เป็นโกลบอลแบบสมบูรณ์ต่อไปนี้ของยูทิลีตี `my_example` สนับสนุนชุดโค้ดทั้งหมดผ่าน single source single path โปรแกรมมิง ที่ไม่ขึ้นกับชุดโค้ด:

```
/*
 * COMPONENT_NAME:
 *
 * FUNCTIONS: my_example
 *
 * โค้ดต่อไปนี้แสดงวิธีการนับจำนวนไบต์และ
 * จำนวนของอักขระในไฟล์ข้อความ
 *
 * ตัวอย่างนี้ใช้เพื่อการสาธิตเท่านั้น  ยังคงสามารถ
 * ปรับปรุงประสิทธิภาพได้
 *
 */

#include        <stdio.h>
#include        <ctype.h>
#include        <locale.h>
#include        <stdlib.h>
#include        "my_example_msg.h"

#define MSGSTR(Num,Str) catgets(catd,MS_MY_EXAMPLE,Num,Str)

/*
 * NAME: my_example
 *
 * FUNCTION: นับจำนวนของอักขระในไฟล์
 *
 */

main(argc,argv)
int argc;
char **argv;
{
    int bytesread,  /* จำนวนไบต์ที่อ่าน */
        bytesprocessed;
    int     leftover;

   int     i;
    int     mbcnt;           /* จำนวนไบต์ในหนึ่งอักขระ */
    int     f;               /* คำอธิบายไฟล์ */
    int mb_cur_max;
```

```
int    bytect;             /* ชื่อที่เปลี่ยนจาก charct... */
int    charct;             /* สำหรับการนับอักขระจริง */
char   *curp, *cure;     /* ตัวชี้ปัจจุบันและตัวชี้สุดท้ายเข้าใน
                            ** บัฟเฟอร์ */
char         buf[BUFSIZ+1];

nl_catd      catd;

wchar_t    wc;

/* ได้รับโลแคลปัจจุบัน */
(void) setlocale(LC_ALL,"");

/* หลังจากการตั้งค่าโลแคล ให้เปิดแค็ตตาล็อกข้อความ */
catd = catopen(MF_MY_EXAMPLE,NL_CAT_LOCALE);

/* แจงอาร์กิวเมนต์ถ้ามี */

/*
** ได้รับจำนวนไบต์สูงสุดในหนึ่งอักขระใน
** โลแคลปัจจุบัน
*/
mb_cur_max = MB_CUR_MAX;
i = 1;

/* เปิดไฟล์ที่ระบุและออกใช้ข้อความแสดงข้อผิดพลาด ถ้ามี */
f = open(argv[i],0);
if(f<0){
    fprintf(stderr,MSGSTR(CANTOPEN,              /*MSG*/
        "my_example: cannot open %s\n"), argv[i]);      /*MSG*/
        exit(2);
}

/* เริ่มต้นตัวแปรสำหรับการนับ */
bytect = 0;
charct = 0;

/* เริ่มต้นการนับไบต์และอักขระ  */

leftover = 0;

for(;;) {
    bytesread = read(f,buf+leftover, BUFSIZ-leftover);
    /* ออกใช้ข้อความแสดงข้อผิดพลาดใดๆ ที่นี่ ถ้าต้องการ */
    if(bytesread <= 0)
         break;

    buf[leftover+bytesread] = '\0';
            /* ป้องกันการอ่านบางส่วน */
    bytect += bytesread;
    curp=buf;
    cure = buf + bytesread+leftover;
    leftover=0; /* ไม่เหลือแล้ว */

    for(; curp<cure ;){
```

```
            /* แปลงเป็นอักขระ wide */
            mbcnt= mbtowc(&wc, curp, mb_cur_max);
            if(mbcnt <= 0){
                mbcnt = 1;
            }else if (cure - curp >=mb_cur_max){
                wc = *curp;
                mbcnt =1;
            }else{
                /* ต้องการข้อมูลเพิ่มเติม */
                leftover= cure - curp;
                strcpy(buf, curp, leftover);
                break;
            }
            curp +=mbcnt;
            charct++;
        }
    }

        /* พิมพ์จำนวนของอักขระและไบต์ */
    fprintf(stderr,MSGSTR(BYTECNT, "number of bytes:%d\n"),
            bytect);
    fprintf(stderr,MSGSTR(CHARCNT, "number of characters:%d\n"),
            charct);
    close(f);
    exit(0);
}
```

เวอร์ชันพาธคู่ ต้นทางเดียวที่ออปติไมซ์สำหรับชุดโค้ด ไบต์เดียว

คำว่า *พาธคู่ ต้นทางเดียว* หมายถึงสองพาธ ในแอ็พพลิเคชันเดียวโดยหนึ่งในพาธนั้นถูกเลือกตอนรันไทม์ ขึ้นอยู่กับการตั้งค่าโลแคลปัจจุบัน ซึ่งระบุว่า ชุดโค้ดที่ใช้งานอยู่เป็นไบต์เดียว หรือหลายไบต์

ถ้าโปรแกรมสามารถคงประสิทธิภาพการทำงาน และไม่เพิ่มขนาดไฟล์ เรียกทำงานมากนัก เมธอดพาธคู่ ต้นทางเดียวจะเป็นตัวเลือก ที่เหมาะสม คุณควรประเมินการเพิ่มขึ้นในขนาดไฟล์ที่ดำเนินการได้ สำหรับแต่ละคำสั่งหรือแต่ละยูทิลิตี้

ในเมธอดพาธคู่ ต้นทางเดียว แมโคร **MB_CUR_MAX** ระบุจำนวนไบต์สูงสุดในอักขระหลายไบต์ใน โลแคลปัจจุบัน วิธีการนี้เพื่อกำหนดที่รันไทม์ว่า พาธการประมวลผลที่จะเลือกเป็นพาธไบต์เดียวหรือ หลายไบต์ ใช้ boolean flag เพื่อบ่งชี้พาธที่จะเลือก ตัวอย่างเช่น:

```
int mbcodeset ;
/* หลังจากทำ setlocale(LC_ALL,"") ให้กำหนดพาธ
** ที่จะเลือก
*/
if(MB_CUR_MAX == 1)
        mbcodeset = 0;
else    mbcodeset = 1;
```

ในวิธีนี้ ชุดโค้ดปัจจุบันจะถูกตรวจสอบเพื่อดูว่าเป็นชุดโค้ดหลายไบต์หรือไม่ และถ้าเป็นเช่นนั้น จะมีการตั้งค่าแฟล็ก `mbcodeset` อย่างเหมาะสม การทดสอบแฟล็กนี้มีผลกระทบต่อประสิทธิภาพน้อยกว่าการทดสอบแมโคร **MB_CUR_MAX** หลายครั้ง

```
if(mbcodeset){
        /* ชุดโค้ดหลายไบต์ (ยังสนับสนุนชุดโค้ด
        ** ไบต์เดียวด้วย )
        */
        /* ใช้ฟังก์ชันการประมวลผลอักขระหลายไบต์หรือ
        อักขระ wide */
}else{
        /* ชุดโค้ดไบต์เดียว */
        /* ประมวลผลตามนั้น */
}
```

แนวทางก่อนหน้าเป็นแนวทางที่เหมาะสมถ้าการทำให้เป็นโกลบอลมีผล กับโมดูลส่วนน้อย การทดสอบที่มากเกินไปสำหรับการนำเสนอพาธคู่ อาจทำให้ประสิทธิภาพการทำงานด้อยลงได้ ควรทดสอบในระดับที่ไม่เกินกว่า ความถี่ในการทดสอบสำหรับกรณีนี้

เวอร์ชันต่อไปนี้ของยูทิลีตี my_example จะสร้างอ็อบเจ็กต์ขึ้นหนึ่งอ็อบเจ็กต์ที่รันไทม์ พาธที่เหมาะสมจะถูกเลือกตามชุดโค้ด เพื่อให้ได้ประสิทธิภาพสูงสุดสำหรับชุดโค้ดนั้น หมายเหตุว่า เราแยกแยะความแตกต่างระหว่างชุดโค้ดไบต์เดียวและหลายไบต์เท่านั้น

```
/*
 * COMPONENT_NAME:
 *
 * FUNCTIONS: my_example
 *
 * โค้ดต่อไปนี้แสดงวิธีการนับจำนวนไบต์และ
 * จำนวนของอักขระในไฟล์ข้อความ
 *
 * ตัวอย่างนี้ใช้เพื่อการสาธิตเท่านั้น ยังคงสามารถ
 * ปรับปรุงประสิทธิภาพได้
 *
 */

#include        <stdio.h>
#include        <ctype.h>
#include        <locale.h>
#include        <stdlib.h>
#include        "my_example_msg.h"

#define MSGSTR(Num,Str) catgets(catd,MS_MY_EXAMPLE,Num,Str)

/*
 * NAME: my_example
 *
 * FUNCTION: นับจำนวนของอักขระในไฟล์
 *
 */

main(argc,argv)
int argc;
char **argv;
{
    int bytesread,  /* จำนวนไบต์ที่อ่าน */
        bytesprocessed;
    int   leftover;
```

```
int   i;
int   mbcnt;   /* จำนวนไบต์ในหนึ่งอักขระ */
int   f;       /* คำอธิบายไฟล์ */
int   mb_cur_max;
int    bytect;             /* ชื่อที่เปลี่ยนจาก charct... */
int    charct;             /* สำหรับการนับอักขระจริง */
char   *curp, *cure; /* ตัวชี้ปัจจุบันและตัวชี้สุดท้ายเข้าในบัฟเฟอร์ */
char        buf[BUFSIZ+1];

    nl_catd          catd;

    wchar_t    wc;

    /* แฟล็กเพื่อบ่งชี้ว่าชุดโค้ดปัจจุบันเป็น
    ** ชุดโค้ดหลายไบต์
    */
    int     multibytecodeset;

    /* ได้รับโลแคลปัจจุบัน */
    (void) setlocale(LC_ALL,"");

/* หลังจากการตั้งค่าโลแคล ให้เปิดแค็ตตาล็อกข้อความ */
catd = catopen(MF_MY_EXAMPLE,NL_CAT_LOCALE);

/* แจงอาร์กิวเมนต์ถ้ามี */

/*
** ได้รับจำนวนไบต์สูงสุดในหนึ่งอักขระใน
** โลแคลปัจจุบัน
*/
mb_cur_max = MB_CUR_MAX;

if(mb_cur_max >1)
    multibytecodeset = 1;
else
    multibytecodeset = 0;

i = 1;

/* เปิดไฟล์ที่ระบุและออกใช้ข้อความแสดงข้อผิดพลาด ถ้ามี */
f = open(argv[i],0);
if(f<0){
    fprintf(stderr,MSGSTR(CANTOPEN,              /*MSG*/
        "my_example: cannot open %s\n"), argv[i]);      /*MSG*/
        exit(2);
}

/* เริ่มต้นตัวแปรสำหรับการนับ */
bytect = 0;
charct = 0;

/* เริ่มต้นการนับไบต์และอักขระ  */

leftover = 0;
```

```
if(multibytecodeset){
    /* Full globalization */
    /* จัดการชุดโค้ดหลายไบต์ที่สนับสนุน */
    for(;;) {
        bytesread = read(f,buf+leftover,
                BUFSIZ-leftover);
        /* ออกใช้ข้อความแสดงข้อผิดพลาดใดๆ ที่นี่ ถ้าต้องการ */
        if(bytesread <= 0)
            break;

        buf[leftover+bytesread] = '\0';
                /* ป้องกันการอ่านบางส่วน */
        bytect += bytesread;
        curp=buf;

        cure = buf + bytesread+leftover;
        leftover=0; /* ไม่เหลือแล้ว */

        for(; curp<cure ;){
            /* แปลงเป็นอักขระ wide */
            mbcnt= mbtowc(&wc, curp, mb_cur_max);
            if(mbcnt <= 0){
                mbcnt = 1;
            }else if (cure - curp >=mb_cur_max){
                wc = *curp;
                mbcnt =1;

            }else{
                /* ต้องการข้อมูลเพิ่มเติม */
                leftover= cure - curp;
                strcpy(buf, curp, leftover);
                break;
            }
            curp +=mbcnt;
            charct++;
        }
    }
}else {

    /* โค้ดเฉพาะชุดโค้ดไบต์เดียวที่
    ** หลีกเลี่ยงการแปลงเป็น widechars ดังนั้นจึงให้ประสิทธิภาพ
    ** ที่ดีที่สุดสำหรับชุดโค้ดไบต์เดียว
    */

    for(;;) {
        bytesread = read(f,buf, BUFSIZ);
        /* ออกใช้ข้อความแสดงข้อผิดพลาดใดๆ ที่นี่ ถ้าต้องการ */
        if(bytesread <= 0)
                break;

            bytect += bytesread;
            charct += bytesread;
    }
```

```
    }

        /* พิมพ์จำนวนของอักขระและไบต์ */
    fprintf(stderr,MSGSTR(BYTECNT, "number of bytes:%d\n"),
            bytect);
    fprintf(stderr,MSGSTR(CHARCNT, "number of characters:%d\n"),
            charct);
    close(f);
    exit(0);
}
```

การใช้ libcur package

ส่วนนี้อธิบายการเปลี่ยนแปลงที่จำเป็นต้องทำสำหรับ โปรแกรมที่ใช้ libcur package (รวมถึง AT&T's libcurses package)

โปรแกรมที่ใช้ libcur package (รวมถึง AT&T's libcurses package) จำเป็นต้องทำการเปลี่ยนแปลงดังต่อไปนี้:

1. ลบสมมุติฐานที่ว่าจำนวนไบต์ที่ต้องใช้เพื่อแสดงถึง อักขระในชุดโค้ดยังแสดงถึงความกว้างคอลัมน์แสดงผล ของอักขระด้วย ใช้รูทีนย่อย **wcwidth** เพื่อกำหนดจำนวนของ คอลัมน์แสดงผลที่ต้องการโดยชุดอักขระ wide ของอักขระ
2. **NLSCHAR** มีการกำหนดใหม่เป็น **wchar_t**
3. `win->_y[y][x]` มีการเข้ารหัส **wchar_t**
4. โปรแกรมไม่ควรสมมุติการเข้ารหัสเฉพาะใดๆ บน **wchar_t**
5. โปรแกรมควรใช้รูทีนย่อย **addstr, waddstr, mvaddstr,** และ **mvwaddstr** แทนตระกูลของรูทีนย่อย **addch** สตริงอาร์กิวเมนต์ ทั้งหมดอยู่ในรูปแบบหลายไบต์
6. รูทีนย่อย **addch** และ **waddch** ยอมรับการเข้ารหัส **wchar_t** ของอักขระ โปรแกรมที่ใช้รูทีนย่อยเหล่านี้ ควรตรวจสอบให้แน่ใจว่ามีการใช้ **wchar_t** ในการเรียก ฟังก์ชันเหล่านี้ (x,y) เพิ่มขึ้นตามจำนวนของคอลัมน์ที่ครอบครองโดย **wchar_t** ซึ่งส่งผ่านไปยังรูทีนย่อยเหล่านี้
7. รูทีนย่อย **delch, wdelch, mvdelch,** และ **mvwdelch** สนับสนุนการลบและ backspace บนอักขระหลายไบต์ โดยขึ้นอยู่กับตำแหน่งปัจจุบันของ (x,y) ถ้าตำแหน่งคอลัมน์ (x,y) ปัจจุบัน ชี้ไปยังคอลัมน์แรกหรือคอลัมน์ที่สองของอักขระแบบสองคอลัมน์ รูทีนย่อย **delch** จะลบทั้งสองคอลัมน์และปรับส่วนที่เหลือของบรรทัด ตามจำนวนของคอลัมน์ที่ลบออก
8. รูทีนย่อย **insch, winsch, mvinsch,** และ **mvwinsch** สามารถใช้เพื่อแทรกการเข้ารหัส wchar_t ของอักขระไว้ที่ ตำแหน่ง (x,y) ปัจจุบันได้ บรรทัดจะถูกปรับตามจำนวนของคอลัมน์ ที่ต้องการโดย wchar_t
9. Libcur package มีการดัดแปลงเพื่อสนับสนุนอักขระภาพวาดกล่อง ตามที่กำหนดไว้ในฐานข้อมูล **terminfo** และไม่สมมุติอักขระกราฟิก ในชุดโค้ด IBM-850 Libcur package สนับสนุนภาพวาดของ อักขระกล่องหลักและกล่องอื่นตามที่กำหนดไว้ในรายการ **box_chars_1** และ **box_chars_2** ในฐานข้อมูล terminfo เมื่อต้องการใช้ความสามารถนี้ ควรดัดแปลงโปรแกรม ในลักษณะต่อไปนี้:

   ภาพวาดอักขระกล่องหลัก:

   ```
   wcolorout(win, Bxa);
   cbox(win);
   wcolorend(win);

   or,
   wcolorout(win, Bxa);
   ```

```
        drawbox(win, y,x, height, width);
        wcolorend(win);
```

ภาพวาดอักขระกล่องอื่น:

```
        wcolorout(win, Bya)
        cboxalt(win);
        wcolorend(win);

        or,

        wcolorout(win, Bya);
        drawbox(win, y, x, height, width);
        wcolorend(win);
```

Bxa และ Bya หมายถึง แอ็ตทริบิวต์หลักและแอ็ตทริบิวต์อื่นที่กำหนดไว้ในฐานข้อมูล **terminfo**

Macros ที่เพิ่มในไฟล์ **cur01.h** มีดังต่อไปนี้:

```
cboxalt(win)

drawboxalt(win, y,x, height, width)
```

10. โปรแกรมที่ต้องการการสนับสนุนอินพุตของอักขระหลายไบต์ไม่ควรตั้งค่า **_extended** เป็น TRUE โดยการเรียกไปยัง **extended(TRUE)** เมื่อแฟล็ก **_extended** เป็นจริง รูทีนย่อย **wgetch** จะส่งคืนการเข้ารหัส **wchar_t** ของอักขระ ด้วยอักขระหลายไบต์ การเข้ารหัสนี้ของ **wchar_t** อาจขัดแย้งกับค่าที่กำหนดไว้แล้ว สำหรับลำดับ escape หรือคีย์ฟังก์ชัน หลีกเลี่ยงความขัดแย้งนี้ เมื่อใช้ชุดโค้ดหลายไบต์โดยการตั้งค่า extended เป็น off (**extended(FALSE)**) ก่อนการอินพุต (ค่าดีฟอลต์คือ TRUE)

    โปรแกรมที่อินพุตอักขระหลายไบต์ ควรทำดังต่อไปนี้:

    อินพุตรูทีน:

    ตัวอย่าง:

```
        int c, count;
        char buf[];

        extended(FALSE); /* ได้รับทีละหนึ่งไบต์ */
        count =0;
        while(1){
            c = wgetch();  /* ได้รับทีละหนึ่งไบต์ */
            buf[count++] = c;
            if(count <=MB_CUR_MAX)
                if(mblen(buf, count) != -1)
                    break; /* พบอักขระ* /
            else
                /*ข้อผิดพลาด ไม่พบอักขระ */
                /* จัดการตัวพิมพ์นี้อย่างเหมาะสม */
                break;
        }
/* buf มีอินพุตลำดับหลายไบต์ */
/* ขณะนี้ จัดการคีย์ PF หรือลำดับ escape ใดๆ ที่นี่ */
```

11. รูทีนย่อย **inch, winch, mvinch,** และ **mvwinch** ส่งคืน **wchar_t** ที่ตำแหน่ง (x,y) ปัจจุบัน หมายเหตุว่า ในกรณีของอักขระความกว้างคอลัมน์คู่ ถ้าจุด (x,y) อยู่ที่คอลัมน์แรก จะมีการส่งคืนโค้ด **wchar_t** ของอักขระความกว้าง คอลัมน์คู่ ถ้าจุด (x,y) อยู่ที่คอลัมน์ที่สอง จะมีการส่งคืน WEOF

คำประกาศ

ข้อมูลนี้พัฒนาขึ้นสำหรับผลิตภัณฑ์ และบริการที่มีในประเทศสหรัฐอเมริกา

IBM อาจไม่นำเสนอผลิตภัณฑ์ เซอร์วิส หรือคุณลักษณะที่อธิบายในเอกสารนี้ในประเทศอื่น โปรดปรึกษาตัวแทน IBM ในท้องถิ่นของคุณสำหรับข้อมูลเกี่ยวกับผลิตภัณฑ์ และเซอร์วิส ที่มีอยู่ในพื้นที่ของคุณในปัจจุบัน การอ้างอิงใดๆ ถึงผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสของ IBM ไม่ได้มีวัตถุประสงค์ที่จะระบุหรือตีความว่า สามารถใช้ได้เฉพาะผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสของ IBM เพียงอย่างเดียว เท่านั้น ผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสใดๆ ที่สามารถทำงานได้เท่าเทียมกัน และไม่ละเมิดสิทธิทรัพย์สินทางปัญญาของ IBM อาจนำมาใช้ แทนได้ อย่างไรก็ตาม ถือเป็นความรับผิดชอบของผู้ใช้ที่จะประเมินและตรวจสอบการดำเนินการของ ผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสใดๆ ที่ไม่ใช่ของ IBM

IBM อาจมีสิทธิบัตร หรืออยู่ระหว่างดำเนินการขอ สิทธิบัตรที่ครอบคลุมถึงหัวข้อซึ่งอธิบายในเอกสารนี้ การนำเสนอเอกสารนี้ไม่ได้เป็นการให้ไลเซนส์ใดๆ ในสิทธิบัตรเหล่านี้แก่คุณ คุณสามารถส่งการสอบถามเกี่ยวกับไลเซนส์ เป็นลายลักษณ์อักษรไปยัง:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

หากมีคำถามเกี่ยวกับข้อมูลชุดอักขระไบต์คู่ (DBCS) โปรดติดต่อแผนกทรัพย์สินทางปัญญาของ IBM ในประเทศของคุณ หรือส่งคำถาม เป็นลายลักษณ์อักษร ไปยัง:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION จัดเตรียมเอกสาร "ตามสภาพที่เป็น" โดยไม่มีการรับประกันใดๆ ทั้งโดยชัดแจ้งหรือโดยนัย ซึ่งรวมถึง แต่ไม่จำกัดถึงการรับประกันโดยนัยที่ไม่ละเมิดความสามารถในการจัดจำหน่าย หรือตามความเหมาะสมสำหรับวัตถุประสงค์อย่างใดอย่างหนึ่ง ในบางรัฐไม่อนุญาตให้ มีการจำกัดความรับผิดชอบในการรับประกันโดยชัดแจ้งหรือโดยนัยในการทำธุรกรรมบางอย่าง ดังนั้นข้อความข้างต้นนี้อาจใช้ไม่ได้กับคุณ

ข้อมูลนี้อาจมีความไม่ถูกต้องด้านเทคนิคหรือข้อผิดพลาดจากการพิมพ์ มีการเปลี่ยนแปลง ข้อมูลในเอกสารนี้เป็นระยะ และการเปลี่ยนแปลงเหล่านี้จะรวมอยู่ในเอดิชันใหม่ของ สิ่งพิมพ์ IBM อาจปรับปรุง และ/หรือเปลี่ยนแปลงในผลิตภัณฑ์ และ/หรือโปรแกรมที่อธิบายในสิ่งพิมพ์นี้ได้ตลอดเวลา โดยไม่ต้องแจ้งให้ทราบ

การอ้างอิงใดๆ ในข้อมูลนี้ถึงเว็บไซต์ไม่ใช่ของ IBM มีการจัดเตรียมเพื่อความสะดวกเท่านั้น และ ไม่ได้เป็นการรับรองเว็บไซต์เหล่านั้นในลักษณะใดๆ เอกสารประกอบที่เว็บไซต์เหล่านั้นไม่ได้เป็น ส่วนหนึ่งของเอกสารประกอบสำหรับผลิตภัณฑ์ IBM นี้ และการใช้เว็บไซต์เหล่านั้นถือเป็นความเสี่ยงของคุณเอง

IBM อาจใช้หรือแจกจ่ายข้อมูลที่คุณจัดหาให้ในลักษณะใดก็ตามที่พิจารณาแล้วว่าเหมาะสม โดยไม่ทำให้เกิดข้อผูกมัดต่อคุณแต่อย่างใด

ผู้รับไลเซนส์ของโปรแกรมนี้ที่ต้องการข้อมูลเกี่ยวกับโปรแกรมสำหรับวัตถุประสงค์ในการเปิดใช้งาน: (i) การแลกเปลี่ยนข้อมูลระหว่างโปรแกรมที่สร้างขึ้นอย่างอิสระกับโปรแกรมอื่น (รวมถึง โปรแกรมนี้) และ (ii) การใช้ข้อมูลซึ่งแลกเปลี่ยนร่วมกัน ควร ติดต่อ:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

ข้อมูลดังกล่าวอาจพร้อมใช้งาน ภายใต้ข้อตกลงและเงื่อนไขที่เหมาะสม รวมถึง การชำระค่าธรรมเนียมในบางกรณี

โปรแกรมที่มีไลเซนส์ซึ่งอธิบายในเอกสารนี้ และเอกสารประกอบที่มีไลเซนส์ทั้งหมดสำหรับโปรแกรม นั้น มีการจัดเตรียมโดย IBM ภายใต้ข้อตกลงของข้อตกลงกับลูกค้าของ IBM, ข้อตกลงไลเซนส์โปรแกรมระหว่างประเทศของ IBM หรือข้อตกลงที่เท่าเทียมกันใดๆ ระหว่างเรา

ข้อมูลประสิทธิภาพ และตัวอย่างลูกค้าที่ระบุมีการนำเสนอสำหรับวัตถุประสงค์การสาธิตเท่านั้น ผลลัพธ์ของประสิทธิภาพการทำงานจริงอาจขึ้นอยู่กับคอนฟิกูเรชันและเกณฑ์การทำงานที่ ระบุเฉพาะ

ข้อมูลเกี่ยวกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM ได้รับมาจากซัพพลายเออร์ของผลิตภัณฑ์เหล่านั้น ประกาศที่เผยแพร่ หรือแหล่งข้อมูลที่เปิดเผยต่อสาธารณะ IBM ไม่ได้ทดสอบผลิตภัณฑ์ดังกล่าว และไม่สามารถยืนยันความถูกต้องของ ประสิทธิภาพ ความเข้ากันได้ หรือการเรียกร้องอื่นใดที่เกี่ยวข้องกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM คำถามเกี่ยวกับ ความสามารถของผลิตภัณฑ์ที่ไม่ใช่ของ IBM ควรส่งไปยังซัพพลายเออร์ของผลิตภัณฑ์เหล่านั้น

ข้อความใดๆ ที่เกี่ยวข้องกับทิศทางในอนาคตและเจตจำนงค์ของ IBM อาจมีการเปลี่ยนแปลง หรือเพิกถอนได้โดยไม่ต้องแจ้งล่วงหน้า และ นำเสนอเฉพาะเป้าหมาย และวัตถุประสงค์เท่านั้น

ราคาของ IBM ทั้งหมดที่แสดงเป็นราคาขายปลีกที่แนะนำของ IBM ซึ่งเป็นราคาปัจจุบัน และอาจเปลี่ยนแปลงได้โดยไม่ต้องแจ้งให้ทราบ ราคาของผู้แทนจำหน่ายอาจแตกต่างไป

ข้อมูลนี้ใช้สำหรับวัตถุประสงค์ของการวางแผนเท่านั้น ข้อมูลในเอกสารนี้อาจมีการเปลี่ยนแปลง ก่อนผลิตภัณฑ์ที่อธิบายจะวางจำหน่าย

ข้อมูลนี้มีตัวอย่างของข้อมูลและรายงานที่ใช้ในการดำเนินการทางธุรกิจรายวัน เพื่อ สาธิตข้อมูลให้สมบูรณ์ที่สุดเท่าที่จะเป็นไปได้ ตัวอย่างจึงมีชื่อของแต่ละบุคคล บริษัท ยี่ห้อ และผลิตภัณฑ์ ชื่อทั้งหมดเหล่านี้เป็นชื่อสมมติ และมีความคล้ายคลึงใดๆ กับบุคคล หรือองค์กรทางธุรกิจใดๆ ถือเป็นความบังเอิญทั้งสิ้น

ไลเซนส์ลิขสิทธิ์:

ข้อมูลนี้มีตัวอย่างแอ็พพลิเคชันโปรแกรมในภาษาต้นฉบับ ซึ่งแสดงถึง เทคนิคด้านโปรแกรมในหลากหลายแพล็ตฟอร์ม คุณอาจคัดลอก ปรับเปลี่ยน และแจกจ่าย โปรแกรมตัวอย่างเหล่านี้ในรูปแบบใดๆ โดยไม่ต้องชำระเงินให้แก่ IBM สำหรับวัตถุประสงค์ในการพัฒนา การใช้ การตลาด หรือการแจกจ่ายโปรแกรมแอ็พพลิเคชัน ที่สอดคล้องกับอินเตอร์เฟสการเขียน

โปรแกรมแอ็พพลิเคชันสำหรับแพล็ตฟอร์มปฏิบัติการ ซึ่งเขียน โปรแกรมตัวอย่าง ตัวอย่างเหล่านี้ยังไม่ได้ผ่านการทดสอบในทุกสภาพ ดังนั้น IBM จึงไม่สามารถรับประกัน หรือบอกเป็นนัยถึง ความน่าเชื่อถือ ความสามารถบริการได้ หรือฟังก์ชันของโปรแกรมเหล่านี้ โปรแกรมตัวอย่างมีการนำเสนอ "ตาม สภาพ" โดยไม่มีการรับประกันประเภทใดๆ IBM ไม่รับผิดชอบ ต่อความเสียหายใดๆ ที่เกิดขึ้นเนื่องจากการใช้โปรแกรมตัวอย่างของคุณ

แต่ละสำเนาหรือส่วนใดๆ ของโปรแกรมตัวอย่างเหล่านี้ หรืองานที่สืบเนื่องใดๆ ต้องมีคำประกาศ ลิขสิทธิ์ดังนี้:

© (ชื่อบริษัทของคุณ) (ปี)

ส่วนของโค้ดนี้ได้มาจากโปรแกรมตัวอย่างของ IBM Corp.

© Copyright IBM Corp. (C) ลิขสิทธิ์ IBM Corp. _ป้อน ปี_

ข้อควรพิจารณาเกี่ยวกับนโยบายความเป็นส่วนตัว

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as the customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

เครื่องหมายการค้า

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Microsoft และ Windows คือเครื่องหมายการค้าของ Microsoft Corporation ในสหรัฐอเมริกา ประเทศอื่นๆ หรือทั้งสอง

UNIX is a registered trademark of The Open Group in the United States and other countries.

ดัชนี

อักขระพิเศษ

A

B

C

E

F

G

I

K

L

M

N

S

T

W

ก

ข

ค

ช

ต

ท

น

พ

ฟ

ภ

ม

ร

ล

ว

ส

ห

อ

พิมพ์ในสหรัฐอเมริกา