AIX เวอร์ชัน 7.2

การเขียนโปรแกรม Coherent Accelerator Processor Interface (CAPI)

AIX เวอร์ชัน 7.2

การเขียนโปรแกรม Coherent Accelerator Processor Interface (CAPI)

IBM

หมายเหตุ

ก่อนที่คุณจะใช้ข้อมูลนี้และผลิตภัณฑ์ที่สนับสนุน โปรดอ่านข้อมูลใน “คำประกาศ” ในหน้า 35

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

สารบัญ

เกี่ยวกับเอกสารนี้

คุณสามารถใช้ Coherent Accelerator Processor Interface (CAPI) เพื่ออนุญาตให้ตัวเร่งความเร็วที่ใช้ Field Programmable Gate Array (FPGA) เข้าถึงหน่วยความจำของแอ็พพลิเคชัน (พื้นที่ผู้ใช้) ได้โดยตรง

Highlighting

The following highlighting conventions are used in this document:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Bold highlighting also identifies graphical objects, such as buttons, labels, and icons that the you select.
Italics	Identifies parameters for actual names or values that you supply.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or text that you must type.

Case sensitivity in AIX

Everything in the AIX® operating system is case sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command `is not found`. Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

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

การเขียนโปรแกรม CAPI

คุณสามารถใช้ Coherent Accelerator Processor Interface (CAPI) เพื่ออนุญาตให้ตัวเร่งความเร็วที่ใช้ Field Programmable Gate Array (FPGA) สามารถเข้าถึงหน่วยความจำของแอ็พพลิเคชัน (พื้นที่ผู้ใช้) ได้โดยตรง

ตัวเร่งความเร็วที่ใช้ FPGA แบบดั้งเดิมดำเนินการถ่ายโอนเข้าถึงหน่วยความจำโดยตรง (DMA) ในสแต็ก Peripheral Component Interconnect (PCI) เพื่อย้ายข้อมูลระหว่างตัวเร่งความเร็วและแอ็พพลิเคชัน CAPI จัดเตรียมเฟรมเวิร์กที่มีวัตถุประสงค์ทั่วไปที่มีตัวเร่งความเร็วที่ใช้ CAPI ที่สามารถถ่ายโอนข้อมูลกลับไปกลับมา จากหน่วยความจำของแอ็พพลิเคชันโดยไม่ต้องใช้ DMA

อะแด็ปเตอร์ CAPI Flash

Coherent Accelerator Processor Interface (CAPI) จัดเตรียมพาธที่มีแบนด์วิดท์สูงและเวลาแฝงต่ำ ระหว่างอุปกรณ์ภายนอก คอร์ POWER8® และสถาปัตยกรรมหน่วยความจำเปิด ของระบบ อะแด็ปเตอร์ CAPI ถูกวางไว้ในสล็อต PCI Express (PCIe) x16 และใช้อะแด็ปเตอร์ PCIe Gen3 เป็นกลไก transport มาตรฐาน

อุปกรณ์ที่มีความสามารถ CAPI สามารถแทนที่ แอ็พพลิเคชันโปรแกรมที่สามารถรันโปรแกรมที่รันบนคอร์ POWER8 หรือจัดเตรียมการประยุกต์ใช้การเร่งความเร็วที่กำหนดเอง อะแด็ปเตอร์ CAPI Flash จะขจัดความซับซ้อนของระบบย่อย I/O เพื่อให้ตัวเร่งความเร็วสามารถทำงานเป็นส่วนหนึ่งของแอ็พพลิเคชัน ซึ่งช่วยลดพาธของโค้ด เนื่องจากแอ็พพลิเคชันสามารถโต้ตอบกับตัวเร่งความเร็วได้โดยตรงโดยไม่ใช้ เคอร์เนลของระบบปฏิบัติการ

ไลบรารีบล็อก CAPI Flash

ไลบรารีบล็อกสำหรับอะแด็ปเตอร์ Coherent Accelerator Processor Interface (CAPI) Flash จัดเตรียมอินเตอร์เฟสพื้นที่ผู้ใช้ให้กับดิสก์ CAPI Flash ที่ระดับบล็อกหรือเซ็กเตอร์ โดยข้ามเคอร์เนลสำหรับการอ่านและเขียนคำร้องขอ I/O ไลบรารีบล็อกสำหรับอะแด็ปเตอร์ CAPI Flash จะสร้างอินเตอร์เฟสสำหรับแอ็พพลิเคชัน เพื่อให้แอ็พพลิเคชันไม่ต้องเข้าถึงรายละเอียดอะแด็ปเตอร์ CAPI Flash ระดับต่ำ

ในระบบปฏิบัติการ AIX ไลบรารีบล็อกสำหรับอะแด็ปเตอร์ CAPI Flash คือ `libcflsh_block.a` บนแพล็ตฟอร์ม Linux ไลบรารีนี้ คือ `libcflsh_block.so`

cblk_init API

วัตถุประสงค์

กำหนดค่าเริ่มต้นของไลบรารีบล็อกสำหรับอะแด็ปเตอร์ Coherent Accelerator Processor Interface (CAPI) Flash

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

int rc = cblk_int(void *arg, int flags)
```

รายละเอียด

cblk_init API จะกำหนดค่าเริ่มต้นให้กับไลบรารีบล็อกสำหรับอะแด็ปเตอร์ CAPI Flash คุณต้องเรียกใช้ **cblk_init** API ก่อนที่คุณจะใช้ API อื่นในไลบรารีบลอกสำหรับอะแด็ปเตอร์ CAPI Flash

พารามิเตอร์

arg
พารามิเตอร์นี้ไม่ได้ใช้ในปัจจุบัน โดยจะตั้งค่าเป็น NULL

แฟล็ก
ระบุแฟล็กสำหรับการกำหนดค่าเริ่มต้น ค่าดีฟอลต์เป็น 0

ค่าที่ส่งคืน

0 API เสร็จสมบูรณ์

ค่าที่ไม่ใช่ศูนย์
เกิดข้อผิดพลาด

cblk_term API

วัตถุประสงค์

ล้างรีซอร์สสำหรับไลบรารีบล็อก Coherent Accelerator Processor Interface (CAPI) Flash เมื่อไลบรารีไม่ได้ใช้แล้ว

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_term(void *arg, int flags)
```

รายละเอียด

cblk_term API จะยกเลิกไลบรารีบล็อกสำหรับอะแด็ปเตอร์ CAPI Flash เมื่อไม่ได้ใช้แล้ว

พารามิเตอร์

arg
พารามิเตอร์นี้ไม่ได้ใช้ในปัจจุบัน (ตั้งค่าเป็น NULL)

แฟล็ก
ระบุแฟล็กสำหรับการกำหนดค่าเริ่มต้น ค่าดีฟอลต์เป็น 0

ค่าที่ส่งคืน

0 API เสร็จสมบูรณ์

ค่าที่ไม่ใช่ศูนย์
เกิดข้อผิดพลาด

cblk_open API

วัตถุประสงค์

เปิดคอลเล็กชันของบล็อกที่ต่อเนื่องจากที่เรียกว่า *chunk* บนอุปกรณ์ Coherent Accelerator Processor Interface (CAPI) Flash ที่สามารถดำเนินการ I/O (อ่านและ เขียน) ให้เสร็จสมบูรณ์ chunk สามารถพิจารณาเป็น logical unit number (LUN) ที่จัดเตรียมการเข้าถึงเซ็กเตอร์ 0 – *n*–1 โดยที่ *n* เป็นขนาดของ chunk หน่วยเป็นเซ็กเตอร์ หากระบุ LUN เสมือน ดังนั้น chunk จะเป็นชุดย่อยของเซ็กเตอร์บนฟิสิคัล LUN

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

chunk_id_t chunk_id = cblk_open(const char *path, int max_num_requests, int mode,
uint64_t ext_arg, int flags)
```

รายละเอียด

cblk_open API จะสร้าง chunk บน CAPI Flash LUN chunk นี้ใช้สำหรับคำร้องขอ I/O (cblk_read หรือ cblk_write) ค่า `chunk_id` ที่ส่งคืนถูกกำหนดให้กับพาธเฉพาะจากอะแด็ปเตอร์เฉพาะให้กับ กระบวนการที่เรียกใช้ ผู้ใช้ของเลเยอร์บล็อกไม่สามารถเห็นฟิสิคัลเซ็กเตอร์พื้นฐานที่ใช้โดย chunk ได้โดยตรง เมื่อมีการตั้งค่าแฟล็ก `CBLK_OPN_VIRT_LUN`

เมื่อการเรียกใช้ cblk_open API เสร็จสมบูรณ์ chunk ID ที่แทน อินสแตนซ์ chunk ที่สร้างขึ้นจะถูกส่งคืนให้กับกระบวนการที่เรียกใช้ที่ใช้สำหรับการเรียกใช้ API ในอนาคต

พารามิเตอร์

`path`
: พารามิเตอร์นี้ระบุชื่อไฟล์พิเศษสำหรับดิสก์ CAPI ตัวอย่างเช่น `/dev/hdisk1` (AIX) และ `/dev/sg0` (Linux)

`max_num_requests`
: พารามิเตอร์นี้ระบุจำนวนสูงสุดของคำสั่งที่สามารถเข้าคิวกับอะแด็ปเตอร์ สำหรับ chunk เฉพาะในเวลาที่ระบุ หากค่านี้เป็น 0 เลเยอร์บล็อกจะเลือกขนาดดีฟอลต์ หาก ค่าที่ระบุมีขนาดใหญ่มาก คำร้องขอ cblk_open จะล้มเหลวโดยมีค่าข้อผิดพลาด `ENOMEM`

โหมด
: พารามิเตอร์นี้ระบุโหมดการเข้าถึง (`O_RDONLY`, `O_WRONLY`, หรือ `O_RDWR`)

`ext_arg`
: พารามิเตอร์นี้ไม่ได้ใช้ในปัจจุบัน

แฟล็ก
: พารามิเตอร์นี้เป็นคอลเล็กชันของแฟล็กบิตต่อไปนี้:

  `CBLK_OPN_VIRT_LUN`
  : แฟล็กนี้ระบุว่า LUN เสมือนถูกเตรียมบนฟิสิคัล LUN หากไม่ได้ระบุแฟล็กนี้ จะจัดเตรียมการเข้าถึงฟิสิคัล LUN ที่สมบูรณ์โดยตรง แฟล็กนี้ใช้ได้สำหรับ หน่วยเก็บข้อมูลชั่วคราวเท่านั้น เมื่อ cblk_close API ถูกเรียกใช้ เซ็กเตอร์ข้อมูลทั้งหมดสำหรับ chunk นี้จะถูกรีลีสเพื่อให้การดำเนินการอื่นสามารถใช้ได้

CBLK_OPN_NO_INTRP_THREADS
: แฟล็กนี้ระบุว่าไลบรารีบล็อก cflash ไม้เริ่มต้นเธรดเบื้องหลังใดๆ สำหรับการประมวลผลและการแตกข้อมูลเกี่ยวกับการเสร็จสมบูรณ์แบบอะซิงโครนัสของคำร้องขอ I/O จาก อะแด็ปเตอร์ CAPI กระบวนการที่ใช้ไลบรารีนี้ต้องเรียกใช้ไลบรารี cblk_aresult หรือไลบรารี cblk_listio เพื่อโพลการเสร็จสมบูรณ์ ของการดำเนินการ I/O

CBLK_OPN_SCRUB_DATA
: แฟล็กนี้ใช้ได้เมื่อมีการระบุแฟล็ก CBLK_OPN_VIRT_LUN เท่านั้น แฟล็กนี้ ระบุว่าข้อมูลบน LUN เสมือนต้องถูกล้างออกก่อนที่ LUN สามารถนำกลับมาใช้ใหม่ได้โดย การดำเนินการอื่น แฟล็กนี้ไม่สนับสนุนสำหรับระบบปฏิบัติการ AIX ในปัจจุบัน

CBLK_OPN_MPIO_FO
: แฟล็กนี้ใช้ได้สำหรับระบบปฏิบัติการ AIX เท่านั้น แฟล็กนี้ระบุว่าไลบรารีบล็อก cflash ใน Multipath I/O (MPIO) failover พาธหนึ่งใช้สำหรับคำร้องขอ I/O ทั้งหมด ยกเว้นพบข้อผิดพลาดเฉพาะพาธ หาก เกิดข้อผิดพลาดพาธเหล่านี้ พาธสำรองจะถูกใช้ หากพร้อมใช้งาน เมื่อต้องการระบุพาธสำหรับดิสก์ CAPI Flash ให้รันคำสั่ง **lspath -l hdiskN** แฟล็กนี้ใช้ไม่ได้หากมีการระบุแฟล็ก CBLK_OPN_VIRT_LUN, CBLK_OPN_RESERVE หรือ CBLK_OPN_FORCED_RESERVE

CBLK_OPN_RESERVE
: แฟล็กนี้ใช้ได้สำหรับระบบปฏิบัติการ AIX เท่านั้น แฟล็กนี้ระบุว่าไลบรารีบล็อก cflash ใช้แอ็ตทริบิวต์นโยบายการจองที่เชื่อมโยงกับ ดิสก์ที่เริ่มต้นการจองดิสก์ คุณไม่สามารถใช้แฟล็กนี้กับแฟล็ก CBLK_OPN_MPIO_FO

CBLK_OPN_FORCED_RESERVE
: แฟล็กนี้ใช้ได้สำหรับระบบปฏิบัติการ AIX เท่านั้น ลักษณะการทำงานของแฟล็กนี้เหมือนกับแฟล็ก CBLK_OPEN_RESERVE ยกเว้น เมื่ออุปกรณ์ถูกเปิดเป็นครั้งแรก โดยจะทำให้การจองดิสก์ที่ยังไม่แก้ไขเสียหาย คุณไม่สามารถใช้แฟล็กนี้กับแฟล็ก CBLK_OPN_MPIO_FO

ค่าที่ส่งคืน

NULL_CHUNK_ID
: เกิดข้อผิดพลาด

cblk_close API

วัตถุประสงค์

ปิดคอลเล็กชันของบล็อกที่ต่อเนื่องจากที่เรียกว่า chunk บนอุปกรณ์หน่วยความจำ Coherent Accelerator Processor Interface (CAPI) Flash ที่สามารถดำเนินการ I/O (อ่านหรือ เขียน) ให้เสร็จสมบูรณ์

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_close(chunk_id_t chunk_id, int flags))
```

รายละเอียด

cblk_close API จะรีลีสบล็อกที่เชื่อมโยงกับ chunk เพื่อให้ การดำเนินการอื่นนำไปใช้ได้ใหม่ ก่อนที่การดำเนินการอื่นจะสามารถใช้บล้อก บล็อกข้อมูล ต้องถูกล้างออกเพื่อลบข้อมูลผู้ใช้ใดๆ หากแฟล็ก CBLK_OPN_SCRUB_DATA ถูกตั้งค่าใน cblk_open API ที่สอดคล้องที่ส่งคืนค่า chunk_id นี้

พารามิเตอร์

`chunk_id`

Handle สำหรับ chunk ที่จะถูกปิดและรีลีสสำหรับนำกลับมาใช้ใหม่

แฟล็ก

คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

`0` API เสร็จสมบูรณ์

ค่าที่ไม่ใช่ศูนย์

เกิดข้อผิดพลาด

cblk_get_lun_size API

วัตถุประสงค์

คืนค่าขนาด (จำนวนของบล็อก) ของ logical unit number (LUN) แบบฟิสิคัลที่ chunk เฉพาะเชื่อมโยงด้วย

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_get_lun_size(chunk_id_t chunk_id, size_t *size, int flags))
```

รายละเอียด

cblk_get_lun_size API จะส่งคืนจำนวนของบล็อกของฟิสิคัล LUN ที่เชื่อมโยงกับ chunk นี้ เมื่อต้องการใช้เซอร์วิส cblk_get_lun_size คุณต้องมี cblk_open API ที่สมบูรณ์เพื่อรับค่า `chunk_id` ที่ถูกต้อง

พารามิเตอร์

`chunk_id`

Handle สำหรับ chunk ที่ต้องส่งคืนขนาดฟิสิคัล LUN

`size`

ระบุจำนวนทั้งหมดของล็อกขนาด 4K สำหรับฟิสิคัล LUN ที่เชื่อมโยงกับ chunk เฉพาะ

แฟล็ก

คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

`0` API เสร็จสมบูรณ์

`>0` เกิดข้อผิดพลาด

cblk_get_size API

วัตถุประสงค์

คืนค่าขนาด (จำนวนของบล็อก) ที่กำหนดให้กับ chunk ID ที่ระบุ ซึ่งเป็น logical unit number (LUN) เสมือน นั่นคือ มีการระบุแฟล็ก CBLK_OPN_VIRT_LUN สำหรับการเรียกใช้ cblk_open ที่ส่งคืน chunk ID นี้ เซอร์วิสนี้ใช้ไม่ได้สำหรับ LUN ที่ไม่ได้ตั้งค่าแฟล็ก CBLK_OPN_VIRT_LUN เมื่อ chunk ถูกเปิดโดย การใช้ cblk_open API

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_get_size(chunk_id_t chunk_id, size_t *size, int flags))
```

รายละเอียด

เซอร์วิส cblk_get_size ส่งคืนจำนวนของบล็อกที่จัดสรรให้กับ chunk ที่ระบุ เมื่อต้องการใช้เซอร์วิส cblk_get_size คุณต้องมี cblk_open API ที่สมบูรณ์เพื่อรับค่า chunk_id ที่ถูกต้อง

พารามิเตอร์

chunk_id
: Handle สำหรับ chunk ที่ขนาด LUN ต้องถูกเปลี่ยนแปลง

size
: ระบุจำนวนทั้งหมดของล็อกขนาด 4K สำหรับ LUN ที่เชื่อมโยงกับ chunk ที่ระบุ

แฟล็ก
: คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

0 API เสร็จสมบูรณ์

>0 เกิดข้อผิดพลาด

cblk_set_size API

วัตถุประสงค์

£กำหนดขนาด (จำนวนของบล็อก) ให้กับ chunk ID เฉพาะที่เป็น logical unit number (LUN) เสมือน นั่นคือ มีการระบุแฟล็ก CBLK_OPN_VIRT_LUN สำหรับการเรียกใช้ cblk_open ที่ส่งคืน chunk ID นี้ หากบล็อกถูกกำหนดให้กับ chunk ID นี้อยู่แล้ว คุณสามารถเพิ่มหรือลดขนาดได้โดยการระบุขนาดที่ใหญ่ขึ้น หรือเล็กลง เซอร์วิสนี้ใช้ไม่ได้สำหรับ LUN ที่ไม่ได้ตั้งค่าแฟล็ก CBLK_OPN_VIRT_LUN เมื่อ chunk ถูกเปิดโดย การใช้ cblk_open API

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_set_size(chunk_id_t chunk_id, size_t size, int flags))
```

รายละเอียด

เมื่อคุณใช้ LUN เสมือน เซอร์วิส cblk_set_size จะจัดสรรจำนวนของบล็อก ให้กับ chunk ที่ระบุ cblk_set_size API ต้องถูกเรียกใช้ก่อนการเรียกใช้ cblk_read หรือ cblk_write chunk นี้ เมื่อต้องการใช้เซอร์วิส cblk_set_size และเมื่อต้องการรับค่า `chunk_id` ที่ถูกต้อง การเรียกใช้ cblk_open ต้องเสร็จสมบูรณ์

หากบล็อกถูกกำหนดให้กับ chunk นี้เมื่อเริ่มต้น ซึ่งไม่ถูกนำมาใช้ใหม่หลังจาก cblk_set_size API จัดสรรบล็อกใหม่ให้กับ chunk เดียวกัน และหากแฟล็ก `CBLK_SCRUB_DATA_FLG` ถูกตั้งค่าในพารามิเตอร์ **flags** บล็อกเดิมจะถูกล้างออกก่อนที่จะสามารถนำกลับมาใช้ใหม่โดยการดำเนินการ cblk_set_size อื่น

หลังจาก cblk_set_size API เสร็จสมบูรณ์ chunk สามารถมี ขนาด logical block address (LBA) ในช่วง 0 – 1 ที่สามารถอ่านหรือเขียน

พารามิเตอร์

`chunk_id`
Handle สำหรับ chunk ที่ต้องตั้งค่าขนาด LUN

`size`
ระบุจำนวนของล็อกขนาด 4K สำหรับ LUN ที่เชื่อมโยงกับ chunk ที่ระบุ

แฟล็ก
คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

`0` API เสร็จสมบูรณ์

`>0` เกิดข้อผิดพลาด

cblk_get_stats API

วัตถุประสงค์

ส่งคืนสถิติสำหรับ chunk ID ที่ระบุ

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
typedef struct chunk_stats_s {
uint64_t max_transfer_size;    /* Maximum transfer size in */
                               /* blocks of this chunk. */
uint64_t num_reads;            /* Total number of reads issued */
                               /* via cblk_read interface */
uint64_t num_writes;           /* Total number of writes issued */
                               /* via cblk_write interface */
uint64_t num_areads;           /* Total number of async reads */
                               /* issued via cblk_aread interface */
uint64_t num_awrites;          /* Total number of async writes */
                               /* issued via cblk_awrite interface*/
uint32_t num_act_reads;        /* Current number of reads active */
                               /* via cblk_read interface */
```

```
uint32_t num_act_writes;       /* Current number of writes active */
                               /* via cblk_write interface */
uint32_t num_act_areads;       /* Current number of async reads */
                               /* active via cblk_aread interface */
uint32_t num_act_awrites;      /* Current number of async writes */
                               /* active via cblk_awrite interface*/
uint32_t max_num_act_writes;   /* High water mark on the maximum */
                               /* number of writes active at once */
uint32_t max_num_act_reads;    /* High water mark on the maximum */
                               /* number of reads active at once */
uint32_t max_num_act_awrites;  /* High water mark on the maximum */
                               /* number of asyync writes active */
                               /* at once. */
uint32_t max_num_act_areads;   /* High water mark on the maximum */
                               /* number of asyync reads active */
                               /* at once. */
uint64_t num_blocks_read;      /* Total number of blocks read */
uint64_t num_blocks_written;   /* Total number of blocks written */
uint64_t num_errors;           /* Total number of all error */
                               /* responses seen */
uint64_t num_aresult_no_cmplt; /* Number of times cblk_aresult */
                               /* returned with no command */
                               /* completion */
uint64_t num_retries;          /* Total number of all commmand */
                               /* retries. */
uint64_t num_timeouts;         /* Total number of all commmand */
                               /* time-outs. */
uint64_t num_fail_timeouts;    /* Total number of all commmand */
                               /* time-outs that led to a command */
                               /* failure. */
uint64_t num_no_cmds_free;     /* Total number of times we didm't */
                               /* have free command available */
uint64_t num_no_cmd_room ;     /* Total number of times we didm't */
                               /* have room to issue a command to */
                               /* the AFU. */
uint64_t num_no_cmds_free_fail; /* Total number of times we didn't */
                               /* have free command available and */
                               /* failed a request because of this*/
uint64_t num_fc_errors;        /* Total number of all FC */
                               /* error responses seen */
uint64_t num_port0_linkdowns;  /* Total number of all link downs */
                               /* seen on port 0. */
uint64_t num_port1_linkdowns;  /* Total number of all link downs */
                               /* seen on port 1. */
uint64_t num_port0_no_logins;  /* Total number of all no logins */
                               /* seen on port 0. */
uint64_t num_port1_no_logins;  /* Total number of all no logins */
                               /* seen on port 1. */
uint64_t num_port0_fc_errors;  /* Total number of all general FC */
                               /* errors seen on port 0. */
uint64_t num_port1_fc_errors;  /* Total number of all general FC */
                               /* errors seen on port 1. */
uint64_t num_cc_errors;        /* Total number of all check */
                               /* condition responses seen */
uint64_t num_afu_errors;       /* Total number of all AFU error */
```

```
                              /* responses seen */
uint64_t num_capi_false_reads; /* Total number of all times */
                              /* poll indicated a read was ready */
                              /* but there was nothing to read. */
uint64_t num_capi_adap_resets; /* Total number of all adapter */
                              /* reset errors. */
uint64_t num_capi_afu_errors;  /* Total number of all */
                              /* CAPI error responses seen */
uint64_t num_capi_afu_intrpts; /* Total number of all */
                              /* CAPI AFU interrupts for command */
                              /* responses seen. */
uint64_t num_capi_unexp_afu_intrpts; /* Total number of all of */
                              /* unexpected AFU interrupts */
uint64_t num_active_threads;   /* Current number of threads */
                              /* running. */
uint64_t max_num_act_threads;  /* Maximum number of threads */
                              /* running simultaneously. */
uint64_t num_cache_hits;       /* Total number of cache hits */
                              /* seen on all reads */
} chunk_stats_t;
int rc = cblk_get_stats(chunk_id_t chunk_id, chunk_stats_t *stats, int flags))
```

รายละเอียด

เซอร์วิส cblk_get_stats ส่งคืนสถิติสำหรับ chunk ID ที่ระบุ

พารามิเตอร์

chunk_id
Handle สำหรับ chunk ที่สถิติต้องถูกกำหนด

stats
ระบุแอดเดรสของโครงสร้าง chunk_stats_t

แฟล็ก
คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

0 API เสร็จสมบูรณ์

>0 เกิดข้อผิดพลาด

cblk_read API

วัตถุประสงค์

อ่านบล็อกขนาด 4K จาก chunk ที่ logical block address (LBA) ที่ระบุลงใน บัฟเฟอร์ที่ระบุ เมื่อคุณใช้ logical unit numbers (LUNs) เสมือน LBA นี้จะไม่เหมือนกับ LBA ของ LUN เนื่องจาก chunk ไม่เริ่มต้นที่ LBA ของ LUN, 0 ทุกครั้ง

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_read(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int flags));
```

รายละเอียด

เซอร์วิส cblk_read จะอ่านข้อมูลจาก chunk และวางข้อมูลดังกล่าวลงใน บัฟเฟอร์ที่จัดเตรียมไว้ การเรียกใช้นี้จะถูกบล็อกจนกว่าการดำเนินการอ่านจะเสร็จสมบูรณ์โดยสำเร็จ หรือมีข้อผิดพลาด หมายความว่า การเรียกใช้นี้จะไม่กลับคืนจนกว่าการดำเนินการอ่านจะเสร็จสมบูรณ์ ในกรณีของ LUN เสมือน คุณต้องเรียกใช้ cblk_set_size API ก่อนการเรียกใช้ cblk_read, cblk_write, cblk_aread หรือ cblk_awrite ไปยัง chunk ที่ระบุ

พารามิเตอร์

`chunk_id`
Handle สำหรับ chunk ที่จะถูกอ่าน

`buf`
ระบุบัฟเฟอร์ที่เก็บข้อมูลที่ถูกอ่านจาก chunk ค่าพารามิเตอร์นี้ต้องตรงกับ ขอบเขต 16 ไบต์

`lba`
ระบุ LBA (ออฟเซ็ตขนาด 4K) ภายใน chunk

`nblocks`
ระบุขนาดของการถ่ายโอนในเซ็กเตอร์ขนาด 4K สำหรับฟิสิคัล LUN ขีดจำกัดด้านบน คือ 16 MB สำหรับ LUN เสมือน ขีดจำกัดด้านบน คือ 4K

`แฟล็ก`
คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

`">` ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

`0` ระบุว่าไม่มีข้อมูลที่ถูกอ่าน

n `> 0`
ระบุว่าการดำเนินการอ่านสำเร็จ โดยที่ *n* เป็นจำนวนของการอ่าน บล็อก

cblk_write API

วัตถุประสงค์

เขียนบล็อกขนาด 4K ไปยัง chunk ที่ logical block address (LBA) ที่ระบุโดยใช้ข้อมูลจาก บัฟเฟอร์ที่ระบุ เมื่อคุณใช้ logical unit numbers (LUNs) เสมือน LBA นี้จะไม่เหมือนกับ LBA ของ LUN เนื่องจาก chunk ไม่เริ่มต้นที่ LBA 0

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_write(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int flags));
```

รายละเอียด

cblk_write API จะเขียนข้อมูลจาก chunk และวางข้อมูลดังกล่าวลงใน บัฟเฟอร์ที่ระบุ การเรียกใช้ cblk_write จะถูกบล็อกจนกว่าการดำเนินการเขียน จะเสร็จสมบูรณ์โดยสำเร็จหรือมีข้อผิดพลาด หมายความว่า การเรียกใช้นี้จะไม่กลับคืนจนกว่าการ

ดำเนินการเขียนจะเสร็จสมบูรณ์ในกรณีของ LUN เสมือน คุณต้องเรียกใช้ cblk_set_size API ก่อนที่คุณจะเรียกใช้ cblk_write API ไปยัง chunk ที่ระบุ

พารามิเตอร์

`chunk_id`
Handle สำหรับ chunk ที่จะถูกเขียน

`buf`
ระบุบัฟเฟอร์ที่ข้อมูลถูกเขียนจากลงใน chunk ค่าพารามิเตอร์นี้ต้องตรงกับ ขอบเขต 16 ไบต์

`lba`
ระบุ LBA (ออฟเซ็ตขนาด 4K) ภายใน chunk

`nblocks`
ระบุขนาดของการถ่ายโอนในเซ็กเตอร์ขนาด 4K สำหรับฟิสิคัล LUN ขีดจำกัดด้านบน คือ 16 MB สำหรับ LUN เสมือน ขีดจำกัดด้านบน คือ 4K

แฟล็ก
คอลเล็กชันของแฟล็กบิต

ค่าที่ส่งคืน

″> ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

0 ระบุว่าไม่มีข้อมูลที่ถูกเขียน

n > 0
ระบุว่าการดำเนินการเขียนเสร็จสมบูรณ์โดยที่ *n* เป็นจำนวนของบล็อก ที่ถูกเขียน

cblk_aread API

วัตถุประสงค์

อ่านบล็อกขนาด 4K จาก chunk ที่ logical block address (LBA) ที่ระบุลงใน บัฟเฟอร์ที่ระบุ เมื่อคุณใช้ logical unit numbers (LUNs) เสมือน LBA นี้จะไม่เหมือนกับ LBA ของ LUN เนื่องจาก chunk ไม่เริ่มต้นที่ LBA 0

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX


typedef enum {
    CBLK_ARW_STATUS_PENDING = 0,     /* Command has not completed */
    CBLK_ARW_STATUS_SUCCESS = 1      /* Command completed successfully */
    CBLK_ARW_STATUS_INVALID = 2      /* Caller's request is invalid */
    CBLK_ARW_STATUS_FAIL = 3         /* Command completed with an error */
} cblk_status_type_t;

typedef struct cblk_arw_status_s {
    cblk_status_type_t status;       /* Status of the command */
                                     /* See errno field for additional */
                                     /* details about the failure */
```

```
    size_t blocks_transferred;       /* Number of block transferred by */
                                     /* this reqeuest. */
    int errno;                       /* Errno when status indicates */
                                     /* CBLK_ARW_STAT_FAIL */
} cblk_arw_status_t;

int rc = cblk_aread(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int
*tag, cblk_arw_status_t *status, int flags));
```

รายละเอียด

เซอร์วิส cblk_aread จะอ่านข้อมูลจาก chunk และวางข้อมูลดังกล่าวลงใน บัฟเฟอร์ที่จัดเตรียมไว้ การเรียกใช้นี้จะไม่ถูกบล็อกจนกว่าการดำเนินการอ่านจะเสร็จสมบูรณ์ ซึ่งหมายความว่า การเรียกใช้นี้จะกลับคืนทันทีหลังจากส่งคำร้องขอแล้ว ก่อนที่การดำเนินการอ่านอาจเสร็จสมบูรณ์ การเรียกใช้ cblk_aresult ต่อมาอาจถูกเรียกใช้เพื่อโพลการเสร็จสมบูรณ์ ในกรณีของ LUN เสมือน คุณต้องเรียกใช้ cblk_set_size API ก่อนที่คุณจะเรียกใช้ cblk_aread API

พารามิเตอร์

chunk_id

Handle สำหรับ chunk ที่จะถูกอ่าน

buf

ระบุบัฟเฟอร์ที่เก็บข้อมูลที่ถูกอ่านจาก chunk ค่าพารามิเตอร์นี้ต้องตรงกับ ขอบเขต 16 ไบต์

lba

ระบุ LBA (ออฟเซ็ตขนาด 4K) ภายใน chunk

nblocks

ระบุขนาดของการถ่ายโอนในเซ็กเตอร์ขนาด 4K สำหรับฟิสิคัล LUN ขีดจำกัดด้านบน คือ 16 MB สำหรับ LUN เสมือน ขีดจำกัดด้านบน คือ 4K

แท็ก

ระบุตัวบ่งชี้ที่ส่งคืนเพื่อให้คุณสามารถระบุแต่ละคำสั่ง ที่ใช้

สถานะ

ระบุแอดเดรสที่จัดเตรียมโดยกระบวนการเรียกใช้ ซึ่งถูกอัพเดตโดยไลบรารี capiblock เมื่อ cblk_aread API เสร็จสมบูรณ์ แอ็พพลิเคชันสามารถใช้กระบวนการโพลสำหรับอาร์กิวเมนต์ **status** แทนการใช้เซอร์วิส cblk_aresult

อะแด็ปเตอร์ CAPI ไม่สามารถอัพเดตฟิลด์นี้ได้โดยตรง เธรดซอฟต์แวร์จำเป็นสำหรับการอัพเดตพารามิเตอร์ status ฟิลด์นี้ ไม่ถูกใช้ หากมีการระบุแฟล็ก CBLK_OPN_NO_INTRP_THREADS สำหรับ cblk_open API ที่ส่งคืนค่า chunk_id นี้

แฟล็ก

คอลเล็กชันของแฟล็กบิตมีดังต่อไปนี้:

CBLK_ARW_WAIT_CMD_FLAGS

บล็อกเซอร์วิส cblk_aread จนกว่าคำสั่ง free จะพร้อมใช้งานเพื่อส่ง คำร้องขอ ไม่เช่นนั้น เซอร์วิสนี้สามารถส่งคืนค่าของ -1 ที่มีค่าข้อผิดพลาดเป็น EWOULDBLOCK (หากไม่มี คำสั่ง free ที่พร้อมใช้งานในปัจจุบัน)

CBLK_ARW_USER_TAG_FLAGS

ระบุว่ากระบวนการเรียกใช้กำลังระบุแท็กที่ผู้ใช้กำหนดสำหรับคำร้องขอนี้ ดังนั้น ผู้เรียกใช้ต้องใช้แท็กนี้กับ cblk_aresult API และตั้งค่าแฟล็ก CBLK_ARESULT_USER_TAG

CBLK_ARW_USER_STATUS_FLAG

ระบุว่ากระบวนการเรียกใช้ตั้งค่าพารามิเตอร์ status ที่จะอัพเดตเมื่อ คำสั่งเสร็จสมบูรณ์

ค่าที่ส่งคืน

"> ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

0 ระบุว่า API นี้สำเร็จ

n > 0

ระบุว่าการดำเนินการอ่านเสร็จสมบูรณ์ (อาจมาจากแคช) โดยที่ *n* เป็นจำนวนของการอ่านบล็อก

cblk_awrite API

วัตถุประสงค์

เขียนบล็อกขนาด 4K ไปยัง chunk ที่ logical block address (LBA) ที่ระบุโดยใช้ข้อมูลจาก บัฟเฟอร์ที่ระบุ เมื่อคุณใช้ logical unit numbers (LUNs) เสมือน LBA นี้จะไม่เหมือนกับ LBA ของ LUN เนื่องจาก chunk ไม่เริ่มต้นที่ LBA 0

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

typedef enum {
       CBLK_ARW_STAT_NOT_ISSUED = 0  /* Command has not been issued */
       CBLK_ARW_STAT_PENDING = 1     /* Command has not completed */
       CBLK_ARW_STAT_SUCCESS = 2     /* Command completed successfully */
       CBLK_ARW_STAT_FAIL = 3        /* Command completed with error */
} cblk_status_type_t;

typedef struct cblk_arw_status_s {
       cblk_status_type_t status;    /* Status of command */
                                     /* See errno field for additional */
                                     /* details about the failure */
       size_t blocks_transferred;    /* Number of block transferred by */
                                     /* this reqeuest. */
       int errno;                    /* Errno when status indicates */
                                     /* CBLK_ARW_STAT_FAIL */
} cblk_arw_status_t;

int rc = cblk_awrite(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int
*tag, cblk_arw_status_t *status, int flags));
```

รายละเอียด

cblk_awrite API จะเขียนข้อมูลจาก chunk และวางข้อมูลดังกล่าวลงใน บัฟเฟอร์ที่ระบุ การเรียกใช้นี้จะไม่ถูกบล็อกจนกว่าการดำเนินการเขียนจะเสร็จสมบูรณ์ ซึ่งหมายความว่า การเรียกใช้นี้จะกลับคืนทันทีหลังจากส่งคำร้องขอแล้ว ก่อนที่การดำเนินการเขียน อาจเสร็จสมบูรณ์ การเรียกใช้ cblk_aresult ต่อมาอาจถูกเรียกใช้เพื่อโพลการเสร็จสมบูรณ์ ในกรณีของ LUN เสมือน คุณต้องเรียกใช้ cblk_set_size API ก่อนที่คุณจะเรียกใช้ cblk_awrite API

พารามิเตอร์

`chunk_id`
Handle สำหรับ chunk ที่จะถูกเขียน

`buf`
ระบุบัฟเฟอร์ที่ข้อมูลถูกเขียนจากลงใน chunk ค่าพารามิเตอร์นี้ต้องตรงกับ ขอบเขต 16 ไบต์

`lba`
ระบุ LBA (ออฟเซ็ตขนาด 4K) ภายใน chunk

`nblocks`
ระบุขนาดของการถ่ายโอนในเซ็กเตอร์ขนาด 4K สำหรับฟิสิคัล LUN ขีดจำกัดด้านบน คือ 16 MB สำหรับ LUN เสมือน ขีดจำกัดด้านบน คือ 4K

แท็ก
ระบุตัวบ่งชี้ที่ส่งคืนเพื่อให้คุณสามารถระบุแต่ละคำสั่ง ที่ใช้

สถานะ
ระบุแอดเดรสที่จัดเตรียมโดยกระบวนการเรียกใช้ซึ่งไลบรารี `capiblock` จะถูกอัพเดตเมื่อ cblk_aread API เสร็จสมบูรณ์ cblk_aread API สามารถใช้โดยแอ็พพลิเคชันแนการใช้เซอร์วิส cblk_aresult

อะแด็ปเตอร์ CAPI ไม่สามารถอัพเดตฟิลด์นี้ได้โดยตรง โดยต้องการเธรดซอฟต์แวร์เพื่ออัพเดตขอบเขตสถานะ ฟิลด์นี้ไม่ถูกใช้ หากมีการระบุแฟล็ก `CBLK_OPN_NO_INTRP_THREADS` สำหรับ cblk_open API ที่ส่งคืนค่า `chunk_id` นี้

แฟล็ก
คอลเล็กชันของแฟล็กบิตมีดังต่อไปนี้:

`CBLK_ARW_WAIT_CMD_FLAGS`
บล็อกเซอร์วิส cblk_aread เพื่อรอให้คำสั่ง free ส่ง คำร้องขอ ไม่เช่นนั้น เซอร์วิสนี้สามารถส่งคืนค่าของ -1 ที่มีค่าข้อผิดพลาดเป็น EWOULDBLOCK (หากไม่มี คำสั่ง free ที่พร้อมใช้งานในปัจจุบัน)

`CBLK_ARW_USER_TAG_FLAGS`
ระบุว่ากระบวนการเรียกใช้กำลังระบุแท็กที่ผู้ใช้กำหนดสำหรับคำร้องขอนี้ ดังนั้น กระบวนการเรียกใช้ต้องใช้แท็กนี้กับ cblk_aresult API และตั้งค่าแฟล็ก `CBLK_ARESULT_USER_TAG`

`CBLK_ARW_USER_STATUS_FLAG`
ระบุว่ากระบวนการเรียกใช้ตั้งค่าพารามิเตอร์ `status` ที่จะอัพเดตเมื่อ คำสั่งเสร็จสมบูรณ์

ค่าที่ส่งคืน

`">` ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

`0` ระบุว่า API เสร็จสมบูรณ์

n > 0
ระบุว่าการดำเนินการอ่านเสร็จสมบูรณ์โดยที่ *n* เป็นจำนวนของบล็อก ที่ถูกเขียน

cblk_aresult API

วัตถุประสงค์

ส่งคืนข้อมูลสถานะและความเสร็จสมบูรณ์สำหรับคำร้องขออะซิงโครนัส

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
rc = cblk_aresult(chunk_id_t chunk_id, int *tag, uint64_t *status, int flags);
```

รายละเอียด

cblk_aresult API จะส่งคืนสถานะของคำร้องขอที่ค้างอยู่ที่ส่งโดย ใช้ cblk_aread หรือ cblk_awrite API หากคำร้องขอที่ค้างอยู่เหล่านี้ เสร็จสมบูรณ์ API จะส่งคืนข้อมูลการเสร็จสมบูรณ์

พารามิเตอร์

`chunk_id`

Handle สำหรับ chunk ที่จะถูกเขียน

แท็ก

ตัวชี้เพื่อแท็กกระบวนการการเรียกใช้ที่กำลังรอให้การเรียกใช้เสร็จสมบูรณ์ หากตั้งค่าแฟล็ก `CBLK_ARESULT_NEXT_TAG` ฟิลด์นี้จะส่งคืนแท็กสำหรับการเสร็จสมบูรณ์ ของคำร้องขออะซิงโครนัสถัดไป

สถานะ

ตัวชี้ไปยังสถานะ สถานะถูกส่งคืนเมื่อคำร้องขอเสร็จสมบูรณ์

แฟล็ก

ระบุแฟล็กต่อไปนี้กับ cblk_aresult API:

`CBLK_ARESULT_BLOCKING`

ระบุแฟล็กนี้หากคุณต้องการให้ cblk_aresult API ถูกบล็อกจนกว่า คำสั่งจะเสร็จสมบูรณ์ (จัดเตรียมคำสั่งแอ็คทีฟที่มีอยู่) หากมีการระบุแฟล็ก `CBLK_ARESULT_NEXT_TAG` การเรียกใช้นี้จะส่งคืนการเสร็จสมบูรณ์ของคำร้องขอ I/O แบบอะซิงโครนัส

`CBLK_ARESULT_USER_TAG`

ใช้แฟล็กนี้เพื่อตรวจสอบสถานะของคำร้องขออะซิงโครนัสที่สั่งโดยใช้ แท็กที่ผู้ใช้ระบุ

ค่าที่ส่งคืน

`"> ` ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

`0` ระบุว่า API นี้เสร็จสมบูรณ์

n > 0

ระบุว่าคำร้องขอเสร็จสมบูรณ์โดยที่ *n* เป็นจำนวนของบล็อก ที่อ่านและเขียน

cblk_clone_after_fork API

วัตถุประสงค์

กำหนดกระบวนการชายน์เพื่อเข้าถึง logical unit number (LUN) เสมือนเดียวกันกับ กระบวนการพาเรนต์ เซอร์วิสนี้ใช้ได้สำหรับแพลตฟอร์ม Linux เท่านั้น

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
rc = cblk_clone_after_fork(chunk_id_t chunk_id, int mode, int flags);
```

รายละเอียด

เซอร์วิส cblk_clone_after_fork กำหนดกระบวนการชายน์เพื่อเข้าถึงข้อมูล จากกระบวนการพาเรนต์ กระบวนการชายน์ต้องดำเนินการการดำเนินการนี้ทันทีหลังจากการเรียกใช้ระบบ **fork( )** โดยใช้ chunk ID ของพาเรนต์เพื่อเข้าถึงหน่วยเก็บข้อมูลดังกล่าว หากกระบวนการชายน์ ไม่ได้ดำเนินการนี้ กระบวนการชายน์จะไม่มีสิทธิ์ใดๆ เพื่อเข้าถึง chunk ID ของพาเรนต์ เซอร์วิสนี้ใช้ไม่ได้สำหรับฟิสิคัล LUN

หมายเหตุ: เซอร์วิสนี้ใช้ได้สำหรับแพลตฟอร์ม Linux เท่านั้น

พารามิเตอร์

`chunk_id`

Handle สำหรับ chunk ที่ใช้โดยกระบวนการพาเรนต์ หากการเรียกใช้นี้กลับคืนสำเร็จ chunk ID นี้ยังสามารถใช้โดยกระบวนการชายน์ได้

โหมด

ระบุโหมดการเข้าถึงสำหรับกระบวนการชายน์ (`O_RDONLY`, `O_WRONLY` หรือ `O_RDWR`)

หมายเหตุ: กระบวนการชายน์ไม่สามารถมีสิทธิ์มากกว่า กระบวนการพาเรนต์ กระบวนการที่มีต่อมาสามารถมีสิทธิ์น้อยกว่าได้

แฟล็ก

พารามิเตอร์นี้เป็ยแฟล็กบิตที่ระบุโดยกระบวนการการเรียกใช้

ค่าที่ส่งคืน

`0` ระบุว่าคำร้องขอเสร็จสมบูรณ์

`">` ระบุข้อผิดพลาด หมายเลขข้อผิดพลาดถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

cblk_listio API

วัตถุประสงค์

ส่งหลายคำร้องขอ I/O ไปยังดิสก์ Coherent Accelerator Processor Interface (CAPI) Flash โดยใช้การร้องขอเดียวและรอให้หลายคำร้องขอ I/O เสร็จสมบูรณ์จากดิสก์ CAPI Flash

ไวยากรณ์

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

typedef struct cblk_io {
      uchar version;                /* Version of the structure */
#define CBLK_IO_VERSION_0 "I"       /* Initial version 0 */
      int flags;                    /* Flags for request */
#define CBLK_IO_USER_TAG 0x0001     /* Caller is specifying a user defined */
                                    /* tag. */
#define CBLK_IO_USER_STATUS 0x0002  /* Caller is specifying a status location */
                                    /* to be updated */
#define CBLK_IO_PRIORITY_REQ 0x0004 /* This is (high) priority request that */
                                    /* must be expediated vs non-priority */
                                    /* requests */
      uchar request_type;           /* Type of request */
#define CBLK_IO_TYPE_READ 0x01      /* Read data request */
#define CBLK_IO_TYPE_WRITE 0x02     /* Write data request */
      void *buf;                    /* Data buffer for the request */
      offset_t lba;                 /* Starting Logical block address for */
                                    /* the request. */
      size_t nblocks;               /* Size of request based on number of */
                                    /* blocks. */
      int tag;                      /* Tag for the request. */
      cblk_arw_status_t stat;       /* Status of the request */
} cblk_io_t

int rc = cblk_listio(chunk_id_t chunk_id,cblk_io_t *issue_io_list[],int
issue_io_items,cblk_io_t *pending_io_list[],int pending_io_items, cblk_io_t
*wait_io_list[], int wait_items, cblk_io_t *completion_io_list[],int
*completion_items, uint64_t timeout, int flags));
```

รายละเอียด

เซอร์วิส cblk_listio จัดเตรียมอินเตอร์เฟสเพื่อส่งหลายคำร้องขอ I/O โดยการเรียกใช้เดียว และโพลความเสร็จสมบูรณ์ของหลายคำร้องขอ I/O โดยใช้การเรียกใช้เดียว แต่ละ คำร้องขอถูกระบุโดยชนิด cblk_io_t ซึ่งรวมถึงบัฟเฟอร์ข้อมูล, logical block address (LBA) เริ่มต้น และขนาดการถ่ายโอนเป็นบล็อกขนาด 4K

เซอร์วิสนี้สามารถอัพเดตคำร้องขอ I/O ที่เชื่อมโยงกับชนิด cblk_io_t (นั่นคือ สถานะการอัพเดต แท็ก และแฟล็ก โดยอิงตามการกำหนด คำร้องขอ I/O)

เซอร์วิสไม่สามารถใช้เพื่อตรวจสอบการเสร็จสมบูรณ์ของคำร้องขอ I/O ที่ส่งผ่าน cblk_aread หรือ cblk_awrite API

พารามิเตอร์

chunk_id

Handle สำหรับ chunk ที่เชื่อมโยงกับคำร้องขอ I/O

issue_io_list

พารามิเตอร์นี้ระบุอาร์เรย์ของคำร้องขอ I/O เพื่อส่งไปยังดิสก์ CAPI Flash แต่ละอิลิเมนต์อาร์เรย์ ที่เป็นชนิด cblk_io_t จะระบุแต่ละคำร้องขอ I/O ที่มีบัฟเฟอร์ข้อมูล, LBA เริ่มต้น และขนาดการถ่ายโอนเป็นบล็อกขนาด 4K อิลิเมนต์อาร์เรย์เหล่านี้สามารถอัพเดตได้โดย API นี้เพื่อระบุสถานะการเสร็จสมบูรณ์และแท็ก ฟิลด์สถานะของแต่ละอิลิเมนต์อาร์เรย์

cblk_io_t ถูกกำหนดค่าเริ่มต้นโดย API นี้ หากพารามิเตอร์ issue_io_list เป็น ดังนั้น API นี้สามารถใช้เพื่อรอให้คำร้องขออื่นที่ส่งโดยการเรียกใช้ cblk_listio ก่อนหน้านี้เสร็จสมบูรณ์โดยการตั้งค่าพารามิเตอร์ pending_io_list

issue_io_items
ระบุจำนวนของอิลิเมนต์อาร์เรย์ในอาร์เรย์ issue_io_list

pending_io_list
ระบุอาร์เรย์ของคำร้องขอ I/O ที่ส่งผ่านคำร้องขอ cblk_listio ก่อนหน้านี้ คุณสามารถใช้พารามิเตอร์ pending_io_list เพื่อโพลการเสร็จสมบูรณ์ของคำร้องขอ I/O โดยไม่ต้องรอให้คำร้องขอทั้งหมดเสร็จสมบูรณ์ (นั้นคือ การตั้งค่าพารามิเตอร์ completion_io_list)

pending_io_items
ระบุจำนวนของอิลิเมนต์อาร์เรย์ในอาร์เรย์ pending_io_list

wait_io_list
ระบุอาร์เรย์ของคำร้องขอ I/O ที่เซอร์วิส cblk_listio ถูกบล็อก บนกว่าคำร้องขอ I/O จะเสร็จสมบูรณ์ คำร้องขอ I/O เหล่านี้ต้องระบุในพารามิเตอร์ issue_io_list หรือพารามิเตอร์ pending_io_list ด้วย หากคำร้องขอ I/O ในอาร์เรย์ issue_io_list ไม่สามารถส่งได้เนื่องจาก ค่าติดตั้งไม่ถูกต้องโดยกระบวนการที่เรียกใช้ หรือไม่มีรีซอร์ส อิลิเมนต์ของคำร้องขอ I/O ดังกล่าวใน io_list จะถูกอัพเดตเพื่อระบุความล้มเหลวนี้ (สถานะถูกตั้งค่าเป็น CBLK_ARW_STAT_NOT_ISSUED) และ cblk_listio API จะไม่รอให้คำร้องขอ I/O ดังกล่าวเสร็จสมบูรณ์ ดังนั้น คำร้องขอ I/O ทั้งหมดในอาร์เรย์ wait_io_list ที่ เสร็จสมบูรณ์จะมีสถานะเป็น CBLK_ARW_STAT_SUCCESS หรือ CBLK_ARW_STAT_FAIL สถานะไม่ถูกอัพเดต สำหรับคำร้องขอ I/O ที่ไม่สมบูรณ์

wait_items
ระบุจำนวนของอิลิเมนต์อาร์เรย์ในอาร์เรย์ wait_io_list

completion_io_list
พารามิเตอร์นี้ถูกตั้งค่าโดยกระบวนการที่เรียกใช้ให้กับอาร์เรย์ของคำร้องขอ I/O ที่กำหนดค่าเริ่มต้นแล้ว (ทำให้เป็นศูนย์) และพารามิเตอร์ completion_items เพื่อตั้งค่าจำนวนของอิลิเมนต์อาร์เรย์ใน อาร์เรย์ เมื่อ cblk_listio API กลับคืน อาร์เรย์จะมีคำร้องขอ I/O ที่ระบุใน พารามิเตอร์ issue_io_list และ pending_io_list ที่เสร็จสมบูรณ์โดยอุปกรณ์ CAPI แต่ไม่ถูกระบุในพารามิเตอร์ wait_io_list หากคำร้องขอ I/O ใน อาร์เรย์ io_list ไม่สามารถส่งได้เนื่องจากค่าติดตั้งไม่ถูกต้องโดยกระบวนการที่เรียกใช้ หรือไม่มีรีซอร์ส อิลิเมนต์ของคำร้องขอ I/O ดังกล่าวจะไม่ถูกคัดลอกไปยังพารามิเตอร์ completion_io_list และสถานะในอาร์เรย์ io_list จะถูกอัพเดตเพื่อระบุความล้มเหลวนี้ (สถานะถูกตั้งค่าเป็น CBLK_ARW_STAT_NOT_ISSUED) ดังนั้น คำร้องขอ I/O ทั้งหมดที่ส่งคืนในรายการนี้จะมีสถานะเป็น CBLK_ARW_STAT_SUCCESS หรือ CBLK_ARW_STAT_FAIL

completion_items
พารามิเตอร์นี้ถูกตั้งค่าโดยกระบวนการที่เรียกใช้เพื่อระบุจำนวนของอิลิเมนต์อาร์เรย์ที่ API นี้ถูกวางในพารามิเตอร์ completion_io_list เมื่อ API นี้กลับคืน ค่าของพารามิเตอร์นี้จะถูกอัพเดตเป็นจำนวนของคำร้องขอ I/O ที่ถูกวางในพารามิเตอร์ completion_io_list

timeout
ระบุค่าการหมดเวลาหน่วยเป็นไมโครวินาทีที่รอคำร้องขอ I/O ทั้งหมดในพารามิเตอร์ wait_io_list พารามิเตอร์นี้ใช้ได้ หากพารามิเตอร์ wait_io_list ไม่ใช่ null เท่านั้น หากคำร้องขอ I/O ใดๆ ในพารามิเตอร์ wait_io_list ไม่เสร็จสมบูรณ์ภายในค่าการหมดเวลา API นี้ จะคืนค่า -1 และตั้งค่าหมายเลขข้อผิดพลาดเป็นค่า ETIMEDOUT (เมื่อเกิดข้อผิดพลาดนี้ บางคำสั่งอาจเสร็จสมบูรณ์ในพารามิเตอร์ wait_io_list) ดังนั้น กระบวนการที่เรียกใช้ ต้องตรวจสอบแต่ละคำร้องขอในพารามิเตอร์ wait_io_list เพื่อตรวจสอบคำร้องขอ ที่เสร็จสมบูรณ์ กระบวนการที่เรียกใช้ต้องลบไอเท็มที่

เสร็จสมบูรณ์แล้วออกจากพารามิเตอร์ pending_io_list ก่อนเรียกใช้ API นี้ในครั้งต่อไป ค่าหมดเวลา ที่เป็น 0 ระบุว่า API นี้ถูกบล็อกจนกว่าคำร้องขอในพารามิเตอร์ wait_io_list จะเสร็จสมบูรณ์

แฟล็ก

ระบุแฟล็กบิตต่อไปนี้:

CBLK_LISTIO_WAIT_ISSUE_CMD

บล็อก cblk_listio API จนกว่าคำสั่ง free จะพร้อมใช้งานเพื่อส่งคำร้องขอทั้งหมด แม้ว่าจะเกินค่าหมดเวลาแล้ว และแฟล็ก CBLK_LISTIO_WAIT_CMD_FLAG ถูกตั้งค่า ไม่เช่นนั้น เซอร์วิสนี้สามาถคืนค่า -1 ที่มีค่าข้อผิดพลาดที่เป็น EWOULDBLOCK หากคำสั่ง free ไม่พร้อมใช้งานในปัจจุบัน (สำหรับสถานการณ์นี้ บางคำสั่งอาจเข้าคิว สำหรับรายการที่ส่งได้สำเร็จ กระบวนการที่เรียกใช้ต้องตรวจสอบแต่ละคำร้องขอ I/O ในพารามิเตอร์ issue_io_list เพื่อตรวจสอบคำร้องขอที่ล้มเหลว)

ค่าที่ส่งคืน

"> ข้อผิดพลาดและหมายเลขข้อผิดพลาดจะถูกตั้งค่าสำหรับรายละเอียดเพิ่มเติม

0 API นี้เสร็จสมบูรณ์โดยไม่มีข้อผิดพลาด

ไลบรารีคีย์-ค่า CAPI Flash

ไลบรารีคีย์-ค่าจัดเตรียมอินเตอร์เฟสสำหรับอุปกรณ์ Coherent Accelerator Processor Interface (CAPI) Flash เพื่อเก็บ ดึงข้อมูล และจัดการอาร์เรย์ ไลบรารีคีย์-ค่าจะแม็พ semantic ของคีย์-ค่ากับไลบรารีบล็อก CAPI Flash

ในระบบปฏิบัติการ AIX ไลบรารีคีย์-ค่า คือ libarkdb.a บนแพล็ตฟอร์ม Linux ไลบรารีนี้ คือ libarkdb.so

ark_create API

วัตถุประสงค์

สร้างอินสแตนซ์ที่เก็บคีย์-ค่า

ไวยากรณ์

```
int ark_create(path, ark, flags)
char * file;
ARK ** handle;
uint64_t flags;
```

รายละเอียด

ark_create API จะสร้างอินสแตนซ์ที่เก็บคีย์-ค่าบนระบบโฮสต์

พารามิเตอร์ path สามารถใช้เพื่อระบุไฟล์พิเศษ (เช่น ไฟล์ /dev/sdx สำหรับแพล็ตฟอร์ม the Linux หรือไฟล์ /dev/hdiskx สำหรับระบบปฏิบัติการ AIX) ที่แทน logical unit number (LUN) แบบฟิสิคัลที่สร้างบนหน่วยเก็บข้อมูลแฟลช หากพารามิเตอร์ path ไม่ใช่ไฟล์พิเศษ API จะถือว่าไฟล์ต้องถูกใช้ สำหรับที่เก็บคีย์-ค่า หากไม่มีไฟล์ ไฟล์จะถูกสร้างขึ้น หากพารามิเตอร์ path เป็น NULL จะใช้หน่วยความจำสำหรับที่เก็บคีย์-ค่า

พารามิเตอร์ flags ระบุคุณสมบัติของที่เก็บคีย์-ค่า หากคุณ ต้องการระบุไฟล์พิเศษสำหรับฟิสิคัล LUN คุณสามารถระบุว่าต้องการใช้ที่เก็บคีย์-ค่าที่มีอยู่ ในฟิสิคัล LUN หรือต้องการสร้างที่เก็บคีย์-ค่าใน LUN เสมือน โดยดีฟอลต์ ทั้งฟิสิคัล LUN จะถูกใช้สำหรับที่เก็บคีย์-ค่า หากต้องการใช้ LUN เสมือน แฟล็กบิต ARK_KV_VIRTUAL_LUN ต้องถูกตั้งค่าในพารามิเตอร์ flags

ที่เก็บคีย์-ค่าที่กำหนดค่าเพื่อใช้ทั้งฟิสิคัล LUN สามารถคงอยู่ได้ คุณสามารถ ปิดการทำงานอินสแตนซ์ที่เก็บคีย์-ค่าได้โดยใช้ persistence (นั้นคือ การบันทึกสถานะปัจจุบันเป็นข้อมูล) ของที่เก็บคีย์-ค่า และจากนั้น คุณสามารถเปิดฟิสิคัล LUN เดิมและโหลดอินสแตนซ์ที่เก็บคีย์-ค่า ในสถานะเดียวกันกับที่เป็นเมื่อถูกปิด เมื่อต้องการกำหนดค่าอินสแตนซ์ที่เก็บคีย์-ค่า ให้คงอยู่เมื่ออินสแตนซ์ที่เก็บคีย์-ค่าปิดการทำงาน (ark_delete) ให้ตั้งค่าบิต ARK_KV_PERSIST_STORE ในพารามิเตอร์ flags โดยดีฟอลต์ อินสแตนซ์ที่เก็บคีย์-ค่าจะไม่ถูกกำหนดค่า เพื่อให้คงอยู่ เมื่อต้องการโหลดอินสแตนซ์ที่เก็บคีย์-ค่าที่เก็บไว้บนฟิสิคัล LUN ให้ตั้งค่า บิต ARK_KV_PERSIST_LOAD ในพารามิเตอร์ flags โดยดีฟอลต์ อินสแตนซ์ที่คงอยู่ (หากมี) จะไม่ถูกโหลดและจะถูกเขียนทับโดยข้อมูลที่คงอยู่ ใหม่

เฉพาะที่เก็บคีย์-ค่าที่สามารถทำให้คงอยู่ได้เท่านั้นที่สามารถเก็บไว้บนฟิสิคัล LUN

เมื่อเสร็จสมบูรณ์แล้ว พารามิเตอร์ handle จะแทนอินสแตนซ์ที่เก็บคีย์-ค่าที่สร้างขึ้นใหม่ที่ใช้สำหรับการเรียกใช้ API ในอนาคต

พารามิเตอร์

path
: ระบุอะแด็ปเตอร์ CAPI ไฟล์ หรือหน่วยความจำ สำหรับที่เก็บคีย์-ค่า

ark
: ระบุ handle ที่แทนที่เก็บคีย์-ค่า

แฟล็ก
: คอลเล็กชันของแฟล็กบิตเพื่อกำหนดคุณสมบัติของที่เก็บคีย์-ค่า:

  ARK_KV_VIRTUAL_LUN
  : ระบุที่เก็บคีย์-ค่าเพื่อใช้ LUN เสมือนที่สร้างจากฟิสิคัล LUN ที่แทนโดย ไฟล์พิเศษ

  ARK_KV_PERSIST_STORE
  : กำหนดค่าอินสแตนซ์ที่เก็บคีย์-ค่าที่ต้องการให้คงอยู่เมื่อปิดการทำงานอินสแตนซ์ ที่เก็บคีย์-ค่า คุณสามารถปิดการทำงาน หรือลบอินสแตนซ์ที่เก็บคีย์-ค่าได้โดยใช้ ark_delete API.

  ARK_KV_PERSIST_LOAD
  : โหลดคอนฟิกูเรชันที่เก็บไว้ หากข้อมูลที่คงอยู่มีอยู่บนฟิสิคัล LUN

ค่าที่ส่งคืน

0
: หมายถึงเสร็จสมบูรณ์ พารามิเตอร์ handle ชี้ไปยังอินสแตนซ์ที่เก็บคีย์-ค่า ที่สร้างขึ้นใหม่

EINVAL
: ค่าไม่ถูกต้องสำหรับหนึ่งในพารามิเตอร์ต่อไปนี้

ENOSPC
: หน่วยความจำหรือหน่วยเก็บข้อมูลแฟลชไม่เพียงพอ

ENOTREADY
ระบบไม่พร้อมใช้งานสำหรับการกำหนดคอนฟิกที่เก็บคีย์-ค่า

ark_delete API

วัตถุประสงค์

ลบอินสแตนซ์ที่เก็บคีย์-ค่า

ไวยากรณ์

```
int ark_delete(ark)
ARK *ark;
```

รายละเอียด

ark_delete API จะลบอินสแตนซ์ที่เก็บคีย์-ค่าที่ระบุโดย พารามิเตอร์ ark บนระบบโฮสต์ เมื่อเสร็จสมบูรณ์ หน่วยความจำและรีซอร์สหน่วยเก็บข้อมูลทั้งหมดที่เกี่ยวข้อง จะถูกรีลีส และ หากอินสแตนซ์ ARK ถูกกำหนดค่าให้คงอยู่ คอนฟิกูเรชันจะคงอยู่เพื่อให้อินสแตนซ์สามารถโหลดได้ในภายหลัง

พารามิเตอร์

ark
handle ที่แทนอินสแตนซ์ที่เก็บคีย์-ค่า

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_delete API จะล้างและลบ รีซอร์สทั้งหมดที่เชื่อมโยงกับอินสแตนซ์ที่เก็บคีย์-ค่า และคืนค่า 0 หากไม่สำเร็จ ark_delete API จะคืนค่าหนึ่งในโค้ดระบุความผิดพลาดที่ไม่ใช่ศูนย์ต่อไปนี้:

0 API เสร็จสมบูรณ์ รีซอร์สทั้งหมดที่เชื่อมโยงกับอินสแตนซ์ที่เก็บคีย์-ค่า จะถูกลบออก

EINVAL
handle ที่เก็บคีย์-ค่าไม่ถูกต้อง

ค่าที่ไม่ใช่ศูนย์
เกิดข้อผิดพลาดและ API ไม่เสร็จสมบูรณ์

ark_set, ark_set_async_cb API

วัตถุประสงค์

เขียนคู่ของคีย์-ค่า

ไวยากรณ์

```
int ark_set(ark, klen, key, vlen, val, res)
int ark_set_async_cb(ark, klen, key, vlen, val, callback, dt)

ARK * ark;
uint64_t klen;
void * key;
```

```
uint64_t vlen;
void * val;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

รายละเอียด

ark_set API จะเก็บคีย์และค่าลงในที่เก็บสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า ที่แทนโดยพารามิเตอร์ **ark** ark_set_async_cb API ทำงานในโหมดอะซิงโครนัส ที่ API จะกลับไปยังกระบวนการที่เรียกใช้ทันที และการดำเนินการ จะถูกกำหนดเวลาให้ทำงาน หลังจากการดำเนินการ ฟังก์ชัน `callback` จะถูกเรียกใช้เพื่อแจ้งกระบวนการที่เรียกใช้เกี่ยวกับการเสร็จสมบูรณ์ของการดำเนินการ

สำหรับอินสแตนซ์ที่เก็บคีย์-ค่า หากมีคีย์อยู่ ค่าที่เก็บไว้จะถูกแทนที่ด้วยค่า `val`

เมื่อเสร็จสมบูรณ์ คู่ของคีย์-ค่าจะถูกเขียนในที่เก็บ และจำนวนของไบต์ที่เขียนไปยัง ที่เก็บคีย์-ค่าจะถูกส่งคืนไปยังกระบวนการที่เรียกใช้ผ่านพารามิเตอร์ **res**

พารามิเตอร์

ark
ระบุ handle ที่แทนการเชื่อมต่อสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า

คีย์
ระบุคีย์สำหรับคู่ของคีย์-ค่า

klen
ระบุความยาวของคีย์หน่วยเป็นไบต์

val
ระบุค่าสำหรับคู่ของคีย์-ค่า

vlen
ระบุความยาวของค่าหน่วยเป็นไบต์

res
ระบุจำนวนของไบต์ที่ถูกเขียนไปยังที่เก็บคีย์-ค่าเมื่อการดำเนินการ I/O เสร็จสมบูรณ์

callback
ระบุฟังก์ชันที่จะเรียกใช้เมื่อการดำเนินการ I/O เสร็จสมบูรณ์

dt ระบุค่า 64 บิตที่ต้องการแท็กการเรียกใช้ API อะซิงโครนัส

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_set และ ark_set_async_cb API จะเขียนคีย์-ค่าในที่เก็บที่เชื่อมโยงกับอินสแตนซ์ที่เก็บคีย์-ค่าและคืนค่า จำนวนของไบต์ที่เขียน ค่าที่ส่งคืนของ ark_set API ระบุสถานะ ของการดำเนินการ ค่าที่ส่งคืนของ ark_set_async_cb API ระบุว่าการดำเนินการอะซิงโครนัสได้รับการยอมรับ หรือถูกปฏิเสธ สถานะจะถูกเก็บไว้ในพารามิเตอร์ **errcode** เมื่อฟังก์ชัน `callback` รัน หาก API ไม่สำเร็จ ark_set และ ark_set_async_cb API จะส่งคืนโค้ดระบุความผิดพลาดที่ไม่ใช่ศูนย์ต่อไปนี้:

EINVAL
พารามิเตอร์ไม่ถูกต้อง

ENOSPC
พื้นที่ที่เหลืออยู่ในที่เก็บคีย์-ค่าไม่เพียงพอ

ark_get, ark_get_async_cb API

วัตถุประสงค์

ดึงค่าสำหรับคีย์เฉพาะ

ไวยากรณ์

```
int ark_get(ark, klen, key, vbuflen, vbuf, voff, res)
int ark_get_async_cb(ark, klen, key, vbuflen, vbuf, voff, callback, dt)

ARK * ark;
uint64_t klen;
void * key;
uint64_t vbuflen;
void * vbuf;
uint64_t voff;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

รายละเอียด

ark_get และ ark_get_async_cb API จะเคียวรีที่เก็บคีย์-ค่า ที่เชื่อมโยงกับพารามิเตอร์ ark สำหรับพารามิเตอร์ key เฉพาะ หากพบคีย์ ค่าของคีย์จะถูกส่งคืนในพารามิเตอร์ vbuf โดยที่ค่าสูงสุดของไบต์ vbuflen ถูกเขียนในที่เก็บคีย์-ค่า เริ่มต้นที่พารามิเตอร์ออฟเซ็ต voff ในค่าของคีย์ ark_get_async_cb API ทำงานในโหมดอะซิงโครนัส ที่ API จะกลับไปยังกระบวนการที่เรียกใช้ทันที และการดำเนินการดึงข้อมูล จะถูกกำหนดเวลาให้ทำงาน หลังจากการดำเนินการเสร็จสมบูรณ์ ฟังก์ชัน callback จะถูกเรียกใช้เพื่อแจ้งกระบวนการที่เรียกใช้เกี่ยวกับการเสร็จสมบูรณ์ของการดำเนินการ

หาก API สำเร็จ ความยาวของค่าของคีย์จะถูกเก็บไว้ในพารามิเตอร์ res ของฟังก์ชัน callback

พารามิเตอร์

ark
ระบุ handle ที่แทนการเชื่อมต่อสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า

คีย์
ระบุคีย์สำหรับคู่ของคีย์-ค่า

klen
ระบุความยาวของคีย์หน่วยเป็นไบต์

vbuf
ระบุบัฟเฟอร์ที่ใช้เพื่อเก็บค่าของคีย์ สำหรับคู่ของคีย์-ค่า

vbuflen
ระบุความยาวของบัฟเฟอร์ vbuf

`voff`
ระบุค่าออฟเซ็ตในคีย์เพื่อเริ่มต้นการดำเนินการอ่าน

`res`
เก็บขนาดของคีย์หน่วยเป็นไบต์ หาก ark_get API เสร็จสมบูรณ์

`callback`
ระบุฟังก์ชัน `callback` ที่จะเรียกใช้เมื่อการดำเนินการ I/O เสร็จสมบูรณ์

`dt` ระบุค่า 64 บิตที่ต้องการแท็กการเรียกใช้ API อะซิงโครนัส

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_get และ ark_get_async_cb API จะคืนค่า 0 ค่าที่ส่งคืนของ ark_get API ระบุสถานะของการดำเนินการ ค่าที่ส่งคืนของ ark_get_async_cb API ระบุว่าการดำเนินการอะซิงโครนัสได้รับการยอมรับ หรือถูกปฏิเสธ สถานะของ API อะซิงโครนัสถูกเก็บไว้ในพารามิเตอร์ `errcode` ของฟังก์ชัน `callback` หากไม่สำเร็จ ark_get และ ark_get_async_cb API จะคืนค่าหนึ่งในโค้ดระบุความผิดพลาดที่ไม่ใช่ศูนย์ต่อไปนี้:

`EINVAL`
พารามิเตอร์ไม่ถูกต้อง

`ENOENT`
ไม่พบคีย์

`ENOSPC`
พื้นที่ในบัฟเฟอร์หน่วยความจำที่ต้องใช้เพื่อเก็บค่าของคีย์ไม่เพียงพอ

ark_del, ark_del_async_cb API

วัตถุประสงค์

ลบที่ที่เชื่อมโยงกับคีย์เฉพาะ

ไวยากรณ์

```
int ark_del(ark, klen, key, res)
int ark_del_async_cb(ark, klen, key, callback, dt)

ARK * ark
uint64_t klen;
void * key;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

รายละเอียด

ark_del และ ark_del_async_cb API จะเคียวรีที่เก็บคีย์-ค่า ที่เชื่อมโยงกับพารามิเตอร์ `handle` สำหรับพารามิเตอร์ `key` เฉพาะ หากพบคีย์ ark_del API จะลบค่า ออกจากที่เก็บคีย์-ค่า ark_del_async_cb API ทำงานในโหมดอะซิงโครนัส ที่ API จะกลับไปยังกระบวนการที่เรียกใช้ทันที และการดำเนินการลบ จะถูกกำหนดเวลาให้ทำงาน หลังจากการดำเนินการเสร็จสมบูรณ์ ฟังก์ชัน `callback` จะถูกเรียกใช้เพื่อแจ้งกระบวนการที่เรียกใช้เกี่ยวกับการเสร็จสมบูรณ์ของการดำเนินการ

หาก API สำเร็จ ความยาวของค่าของคีย์จะถูกส่งคืนไปยังกระบวนการที่เรียกใช้ในพารามิเตอร์ **res** ของฟังก์ชัน `callback`

พารามิเตอร์

ark
ระบุ handle ที่แทนการเชื่อมต่อสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า

คีย์
ระบุคีย์สำหรับคู่ของคีย์-ค่า

klen
ระบุความยาวของคีย์หน่วยเป็นไบต์

res
เก็บขนาดของคีย์หน่วยเป็นไบต์ หาก API นี้เสร็จสมบูรณ์

callback
ระบุฟังก์ชัน `callback` ที่จะเรียกใช้เมื่อการดำเนินการ I/O เสร็จสมบูรณ์

dt ระบุค่า 64 บิตที่ต้องการแท็กการเรียกใช้ API อะซิงโครนัส

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_del และ ark_del_async_cb API จะคืนค่า 0 ค่าที่ส่งคืนของ ark_del API ระบุสถานะของการดำเนินการ ค่าที่ส่งคืนของ ark_del_async_cb API ระบุว่าการดำเนินการอะซิงโครนัสได้รับการยอมรับ หรือถูกปฏิเสธ สถานะของ API อะซิงโครนัสถูกเก็บไว้ในพารามิเตอร์ **errcode** ของฟังก์ชัน `callback` หากไม่สำเร็จ ark_del และ ark_del_async_cb API จะคืนค่าหนึ่งในโค้ดระบุความผิดพลาดที่ไม่ใช่ศูนย์ต่อไปนี้:

EINVAL
พารามิเตอร์ไม่ถูกต้อง

ENOENT
ไม่พบคีย์

ark_exists, ark_exists_async_cb API

วัตถุประสงค์

เคียวรีที่เก็บคีย์-ค่าเพื่อตรวจสอบว่ามีคีย์เฉพาะหรือไม่

ไวยากรณ์

```
int ark_exist(ark, klen, key, res)
int ark_exist_async_cb(ark, klen, key, callback, dt)

ARK * ark
uint64_t klen;
void * key;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

รายละเอียด

ark_exists และ ark_exists_async_cb API จะเคียวรีที่เก็บคีย์-ค่า ที่เชื่อมโยงกับพารามิเตอร์ ark สำหรับพารามิเตอร์ key เฉพาะ หากพบคีย์ ark_exist API จะคืนค่าขนาดของค่าหน่วยเป็นไบต์ในพารามิเตอร์ res คีย์และค่าของคีย์ไม่ถูกเปลี่ยนแปลง ark_exists_async_cb API ทำงานในโหมดอะซิงโครนัส ที่ API จะกลับไปยังกระบวนการที่เรียกใช้ทันที และการดำเนินการเคียวรี จะถูกกำหนดเวลาให้ทำงาน หลังจากการดำเนินการเสร็จสมบูรณ์ ฟังก์ชัน callback จะถูกเรียกใช้เพื่อแจ้งกระบวนการที่เรียกใช้เกี่ยวกับการเสร็จสมบูรณ์ของการดำเนินการ

พารามิเตอร์

ark
: ระบุ handle ที่แทนการเชื่อมต่อสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า

คีย์
: ระบุคีย์สำหรับคู่ของคีย์-ค่า

klen
: ระบุความยาวของคีย์หน่วยเป็นไบต์

res
: เก็บขนาดของคีย์หน่วยเป็นไบต์ หาก API นี้เสร็จสมบูรณ์

callback
: ระบุฟังก์ชัน callback ที่จะเรียกใช้เมื่อการดำเนินการ I/O เสร็จสมบูรณ์

dt ระบุค่า 64 บิตที่ต้องการแท็กการเรียกใช้ API อะซิงโครนัส

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_exists และ ark_exists_async_cb API จะคืนค่า 0 ค่าที่ส่งคืนของ ark_exists API ระบุสถานะของการดำเนินการ ค่าที่ส่งคืนของ ark_exists_async_cb API ระบุว่าการดำเนินการอะซิงโครนัสได้รับการยอมรับ หรือถูกปฏิเสธ สถานะของ API อะซิงโครนัสถูกเก็บไว้ในพารามิเตอร์ errcode ของฟังก์ชัน callback หากไม่สำเร็จ ark_exists และ ark_exists_async_cb API จะคืนค่าหนึ่งในโค้ดระบุความผิดพลาดที่ไม่ใช่ศูนย์ ต่อไปนี้:

EINVAL
: พารามิเตอร์ไม่ถูกต้อง

ENOENT
: ไม่พบคีย์

ark_first API

วัตถุประสงค์

คืนค่าคีย์แรกที่พบในที่เก็บคีย์-ค่าและคืนค่า handle ที่ต้องการทำซ้ำ ผ่านที่เก็บคีย์-ค่า

ไวยากรณ์

```
ARI*ark_first(ark, kbuflen, klen, kbuf)
ARK * ark
uint64_t buflen;
int64_t *klen;
void * kbuf;
```

รายละเอียด

ark_first API จะคืนค่าคีย์แรกที่พบในที่เก็บคีย์-ค่าในบัฟเฟอร์ `kbuf` และขนาดของคีย์ในพารามิเตอร์ `klen` ขณะที่ขนาดของคีย์ (`klen`) น้อยกว่าขนาด `kbuf` (`kbuflen`)

หาก API นี้เสร็จสมบูรณ์ iterator handle จะกลับไปยังกระบวนการที่เรียกใช้ที่ต้องใช้ เพื่อดึงคีย์ถัดไปในที่เก็บคีย์-ค่าโดยการเรียกใช้ ark_next API

พารามิเตอร์

`ark`
ระบุ handle ที่แทนการเชื่อมต่อสำหรับอินสแตนซ์ที่เก็บคีย์-ค่า

`kbuflen`
ระบุความยาวของพารามิเตอร์ `kbuf`

`klen`
ระบุขนาดของคีย์ที่ส่งคืนในพารามิเตอร์ `kbuf`

`kbuf`
ระบุบัฟเฟอร์ที่เก็บคีย์

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_first API จะคืนค่า handle ที่ต้องใช้เพื่อ ทำซ้ำผ่านที่เก็บที่เก็บคีย์-ค่าในการเรียกใช้ครั้งถัดมาโดยใช้ ark_next API หากไม่สำเร็จ ark_first API จะคืนค่า NULL โดยตั้งค่าหมายเลขข้อผิดพลาดเป็นหนึ่งในค่าต่อไปนี้:

`EINVAL`
พารามิเตอร์ไม่ถูกต้อง

`ENOSPC`
พารามิเตอร์ `kbuf` มีพื้นที่ไม่เพียงพอสำหรับการเก็บคีย์

ark_next API

วัตถุประสงค์

คืนค่าคีย์ถัดไปที่พบในที่เก็บคีย์-ค่า

ไวยากรณ์

```
int ark_next(iter, kbuflen, klen, kbuf)
ARK * iter
uint64_t buflen;
int64_t *klen;
void *kbuf;
```

รายละเอียด

ark_next API คืนค่าคีย์ถัดไปที่พบในที่เก็บคีย์-ค่าโดยอิงตาม iterator handle, `iter`, ในบัฟเฟอร์ `kbuf` buffer และขนาดของคีย์ในพารามิเตอร์ `klen` ขณะที่ขนาดของคีย์ (`klen`) น้อยกว่าขนาด `kbuf` (`kbuflen`)

หากสำเร็จ handle จะกลับไปยังกระบวนการที่เรียกใช้ที่ต้องใช้เพื่อดึงคีย์ถัดไป ในที่เก็บคีย์-ค่าโดยการเรียกใช้ ark_next API หากถึงท้ายของ ที่เก็บคีย์-ค่า โค้ดระบุความผิดพลาด ENOENT จะถูกส่งคืน

หมายเหตุ: เนื่องจากลักษณะการทำงานแบบไดนามิกของที่เก็บ คีย์บางคีย์ที่ถูกเขียนอาจไม่ ถูกส่งคืน

พารามิเตอร์

`iter`
ระบุ iterator handle ที่เริ่มต้นการค้นหาในที่เก็บคีย์-ค่า

`kbuf`
ระบุบัฟเฟอร์ที่เก็บคีย์

`kbuflen`
ระบุความยาวของพารามิเตอร์ `kbuf`

`klen`
ระบุขนาดของคีย์ที่ส่งคืนในพารามิเตอร์ `kbuf`

ค่าที่ส่งคืน

เมื่อเสร็จสมบูรณ์ ark_next API จะคืนค่า handle ที่ต้องใช้เพื่อ ทำซ้ำผ่านที่เก็บที่เก็บคีย์-ค่าในการเรียกใช้ครั้งถัดมาโดยใช้ ark_next API หากไม่สำเร็จ ark_next API จะส่งคืนหนึ่งในค่าต่อไปนี้:

`EINVAL`
พารามิเตอร์ไม่ถูกต้อง

`ENOENT`
ถึงท้ายของที่เก็บแล้ว

ark_allocated API

วัตถุประสงค์

คืนค่าจำนวยไบต์ที่จัดสรรให้กับที่เก็บ

ไวยากรณ์

```
int ark_allocated(ark, size)
ARK * ark;
uint64_t *size;
```

รายละเอียด

ark_allocated API จะคืนค่าจำนวนไบต์ที่จัดสรรให้กับที่เก็บคีย์-ค่า ผ่านพารามิเตอร์ `size`

พารามิเตอร์

`ark`
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

`size`
เก็บขนาดของบล็อกที่จัดสรรให้กับที่เก็บคีย์-ค่าหน่วยเป็นไบต์

ค่าที่ส่งคืน

`0` หมายถึงเสร็จสมบูรณ์

`EINVAL`
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

ark_inuse API

วัตถุประสงค์

คืนค่าจำนวนไบต์ที่ใช้งานอยู่ในที่เก็บ คีย์-ค่า

ไวยากรณ์

```
int ark_inuse(ark, size)
ARK * ark;
uint64_t *size;
```

รายละเอียด

ark_inuse API จะคืนค่าจำนวนไบต์ที่ใช้งานอยู่ใน ที่เก็บคีย์-ค่าผ่านพารามิเตอร์ `size`

พารามิเตอร์

`ark`
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

`size`
เก็บขนาดของที่เก็บที่ใช้งานอยู่ หน่วยเป็นไบต์

ค่าที่ส่งคืน

`0` หมายถึงเสร็จสมบูรณ์

EINVAL
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

ark_actual API

วัตถุประสงค์

คืนค่าจำนวนไบต์ที่ใช้งานอยู่ในที่เก็บ คีย์-ค่า

ไวยากรณ์

```
int ark_actual(ark, size)
ARK * ark;
uint64 * size;
```

รายละเอียด

ark_actual API จะคืนค่าจำนวนไบต์ที่ใช้งานอยู่ใน ที่เก็บคีย์-ค่าผ่านพารามิเตอร์ `size` API นี้แตกต่างจาก ark_inuse API ที่ API นี้ใช้ขนาดจริงขิงแต่ละคีย์และค่าของคีย์ แทนที่จะใช้การจัดสรรบล็อกทั่วไปเพื่อเก็บค่าเหล่านี้

พารามิเตอร์

`ark`
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

`size`
เก็บขนาดของบล็อกที่ใช้งานอยู่ หน่วยเป็นไบต์

ค่าที่ส่งคืน

`0` หมายถึงเสร็จสมบูรณ์ พารามิเตอร์ `handle` ชี้ไปยังอินสแตนซ์ที่เก็บคีย์-ค่า ที่สร้างขึ้นใหม่

`EINVAL`
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

ark_fork, ark_fork_done API

วัตถุประสงค์

Fork ที่เก็บคีย์-ค่าสำหรับการเก็บแบบถาวร เซอร์วิสนี้ใช้ได้สำหรับแพลตฟอร์ม Linux เท่านั้น

ไวยากรณ์

```
int ark_fork(ark)
int ark_fork_done(ark)
ARK * handle;
```

รายละเอียด

ark_fork และ ark_fork_done API ถูกเรียกใช้โดยกระบวนการที่เก็บคีย์-ค่าพาเรนต์เพื่อเตรียมที่เก็บคีย์-ค่า สำหรับการ fork (แยกเป็นหลายกระบวนการ), fork กระบวนการชายน์ และเพื่อ ล้างสถานะการเรียกใช้หลังจากออกจากกระบวนการชายน์แล้ว ark_fork API จะ fork กระบวนการชายน์ และหลังจากเสร็จสมบูรณ์ API จะคืนค่า ID กระบวนการของกระบวนการชายน์ให้กับ

กระบวนการพาเรนต์ และคืนค่า 0 ให้กับกระบวนการชายน์ หลังจากกระบวนการพาเรนต์ตรวจพบว่าออกจากกระบวนการชายน์แล้ว ark_fork_done API จะถูกเรียกใช้เพื่อล้างค่าสถานะจากการเรียกใช้ ark_fork ทั้งหมด

หมายเหตุ: ark_fork API จะล้มเหลวหากมีคำสั่ง อะซิงโครนัสที่ค้างอยู่ เซอร์วิส ark_fork ใช้ได้สำหรับแพล็ตฟอร์ม Linux เท่านั้น

พารามิเตอร์

`ark`
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

ค่าที่ส่งคืน

`0` หมายถึงเสร็จสมบูรณ์

`EINVAL`
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

`EBUSY`
ระบุความล้มเหลวเนื่องจากการดำเนินการอะซิงโครนัสที่ค้างอยู่

`ENOMEM`
ระบุความล้มเหลวเนื่องจากพื้นที่ที่ใช้เพื่อโคลนที่เก็บไม่เพียงพอ

ark_random API

วัตถุประสงค์

คืนค่าคีย์แบบสุ่มจากที่เก็บคีย์-ค่า

ไวยากรณ์

```
int ark_random(ark, kbuflen, klen, kbuf)
ARK * ark;
uint64_t kbuflen
int64_t *klen;
void * kbuf;
```

รายละเอียด

ark_random API คืนค่าคีย์แบบสุ่มจากที่เก็บคีย์-ค่าโดยอิงตาม `ark` handle ในบัฟเฟอร์ `kbuf` และขนาดของ คีย์ในพารามิเตอร์ `klen` ขณะที่ขนาดของคีย์ (`klen`) น้อยกว่าขนาด `kbuf` (`kbuflen`)

พารามิเตอร์

`ark`
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

`kbuflen`
เก็บขนาดของที่เก็บคีย์-ค่าหน่วยเป็นไบต์

klen
ระบุขนาดของคีย์ที่ส่งคืนในพารามิเตอร์ kbuf

kbuf
ระบุบัฟเฟอร์ที่เก็บคีย์

ค่าที่ส่งคืน

0 หมายถึงเสร็จสมบูรณ์

EINVAL
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

ark_count API

วัตถุประสงค์

คืนค่าจำนวนของคีย์ที่พบในที่เก็บคีย์-ค่า

ไวยากรณ์

```
int ark_count(ark, count)
ARK * ark;
int * count;
```

รายละเอียด

ark_count API จะคืนค่าจำนวนทั้งหมดของคีย์ในที่เก็บคีย์-ค่า อิงตาม ark handle และเก็บผลลัพธ์ในพารามิเตอร์ count

พารามิเตอร์

ark
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

count
ระบุจำนวนของคีย์ที่พบในที่เก็บคีย์-ค่า

ค่าที่ส่งคืน

0 หมายถึงเสร็จสมบูรณ์

EINVAL
ระบุความล้มเหลวเนื่องจากพารามิเตอร์ไม่ถูกต้อง

ark_stats API

วัตถุประสงค์

คืนค่าจำนวนของการดำเนินการ I/O ของคีย์-ค่าและการดำเนินการ I/O ของบล็อก

ไวยากรณ์

```
#include <arkdb.h>

int ark_stats(ARK *ark, uint64_t *ops, uint64_t *ios);
```

รายละเอียด

ark_stats API จะคืนค่าจำนวนทั้งหมดของการดำเนินการ I/O ของคีย์-ค่าผ่าน พารามิเตอร์ ops และจำนวนทั้งหมดของการดำเนินการ I/O ของบล็อกผ่านพารามิเตอร์ ios

พารามิเตอร์

ark
ระบุ handle ที่แทนที่เก็บคีย์-ค่า

ops
ระบุจำนวนทั้งหมดของการดำเนินการ I/O ของคีย์-ค่า

ios
ระบุจำนวนทั้งหมดของการดำเนินการ I/O ของบล็อก

ค่าที่ส่งคืน

0 หมายถึงเสร็จสมบูรณ์

EINVAL
ระบุว่าพบข้อผิดพลาด

คำประกาศ

ข้อมูลนี้พัฒนาขึ้นสำหรับผลิตภัณฑ์และบริการที่มีในประเทศสหรัฐอเมริกาเท่านั้น

IBM อาจไม่นำเสนอผลิตภัณฑ์ เซอร์วิส หรือคุณลักษณะที่อธิบายในเอกสารนี้ในประเทศอื่น โปรดปรึกษาตัวแทน IBM ในท้องถิ่นของคุณสำหรับข้อมูลเกี่ยวกับผลิตภัณฑ์และการบริการที่มีอยู่ในพื้นที่ของคุณในปัจจุบัน การอ้างอิงใดๆ ถึงผลิตภัณฑ์ โปรแกรม หรือการบริการของ IBM ไม่ได้มีวัตถุประสงค์ที่จะระบุหรือตีความว่าสามารถใช้ได้เฉพาะผลิตภัณฑ์ โปรแกรม หรือการบริการของ IBM เพียงอย่างเดียวเท่านั้น ผลิตภัณฑ์ โปรแกรม หรือการบริการใดๆ ที่สามารถทำงานได้เท่าเทียมกัน และไม่ละเมิดสิทธิทรัพย์สินทางปัญญาของ IBM สามารถนำมาใช้แทนได้ อย่างไรก็ตาม เป็นความรับผิดชอบของผู้ใช้ ที่จะประเมินและตรวจสอบการดำเนินการของผลิตภัณฑ์ โปรแกรม หรือการบริการใดๆ ที่ไม่ใช่ของ IBM

IBM อาจมีสิทธิบัตร หรืออยู่ระหว่างดำเนินการขอสิทธิบัตร ที่ครอบคลุมถึงหัวข้อซึ่งอธิบายในเอกสารนี้ การตกแต่งเอกสารนี้ไม่ได้ให้สิทธิใช้งานใดๆ ในสิทธิบัตรเหล่านี้แก่คุณ คุณสามารถส่งการสอบถามเกี่ยวกับใบอนุญาตเป็นลายลักษณ์อักษร ไปที่:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

หากมีคำถามเกี่ยวกับข้อมูลไบต์คู่ (DBCS) โปรดติดต่อแผนกทรัพย์สินทางปัญญาของ IBM ในประเทศของคุณ หรือส่งคำถามเป็นลายลักษณ์อักษรไปที่ :

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 จัดเตรียมสิ่งพิมพ์นี้ "ตามสภาพที่เป็นอยู่" โดยไม่มีการรับประกันใดๆ ไม่ว่าจะโดยชัดแจ้งหรือโดยนัย ซึ่งรวมถึง แต่ไม่จำกัดถึงการรับประกันโดยนัยการไม่ละเมิดสิทธิ์ การจำหน่าย หรือความเหมาะสมสำหรับวัตถุประสงค์เฉพาะ ในบางรัฐไม่อนุญาตให้ ปฏิเสธการรับประกันทางตรงหรือทางอ้อมในธุรกรรมบางอย่าง ดังนั้น ข้อความนี้จึงอาจจะไม่ใช้กับคุณ

ข้อมูลนี้อาจมีความไม่ถูกต้องทางเทคนิคหรือความผิดพลาด ทางการพิมพ์ การเปลี่ยนแปลงข้อมูลในนี้จะมีเป็นระยะๆ ซึ่งจะสอดคล้องกับ การตีพิมพ์ในครั้งใหม่ IBM อาจปรับปรุงและ/หรือเปลี่ยนแปลงในผลิตภัณฑ์และ/หรือโปรแกรมที่อธิบายไว้ในสิ่งพิมพ์นี้ได้ตลอดเวลาโดยไม่ต้องแจ้งให้ทราบ

การอ้างอิงใดๆ ในข้อมูลนี้ถึงเว็บไซต์ที่ไม่ใช่ของ IBM มีการนำเสนอเพื่อความสะดวกเท่านั้น และไม่ได้เป็นการสนับสนุนเว็บไซต์ดังกล่าวในลักษณะใดๆ เนื้อหาที่อยู่ในเว็บไซต์เหล่านั้นไม่ได้เป็น ส่วนหนึ่งของเนื้อหาสำหรับผลิตภัณฑ์ของ IBM นี้ และการใช้เว็บไซต์ดังกล่าวถือเป็นความเสี่ยงของคุณเอง

IBM อาจใช้หรือแจกจ่ายข้อมูลใดๆ ที่คุณ ให้ในรูปแบบต่างๆ ซึ่ง IBM เชื่อว่ามีความเหมาะสมได้โดยไม่เกิดข้อผูกมัดใดๆ กับคุณ

ผู้รับใบอนุญาตของโปรแกรมนี้ที่ต้องการได้รับข้อมูลเกี่ยวกับโปรแกรมเพื่อเปิดใช้งาน: (i) การแลกเปลี่ยนข้อมูลระหว่างโปรแกรมที่สร้างขึ้นอย่างอิสระและโปรแกรมอื่นๆ (รวมถึงโปรแกรมนี้) และ (ii) การใช้ข้อมูลที่มีการแลกเปลี่ยนร่วมกัน ควรติดต่อ:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

ข้อมูลดังกล่าวอาจพร้อมใช้งานภายใต้ระยะเวลาและเงื่อนไขที่เหมาะสม โดยมีการชำระค่าธรรมเนียมในบางกรณี

โปรแกรมที่ได้รับอนุญาตซึ่งอธิบายไว้ในเอกสารนี้และเอกสารประกอบที่ได้รับอนุญาตทั้งหมดที่มีอยู่มีการนำเสนอโดย IBM ภายใต้ระยะเวลาของข้อตกลงกับลูกค้าของ IBM, ข้อตกลงเกี่ยวกับใบอนุญาตโปรแกรมระหว่างประเทศของ IBM หรือข้อตกลงที่เท่าเทียมกันใดๆ ระหว่างเรา

ข้อมูลประสิทธิภาพ และตัวอย่างลูกค้าที่ระบุมีการนำเสนอสำหรับวัตถุประสงค์การสาธิตเท่านั้น ผลลัพธ์ของประสิทธิภาพการทำงานจริงอาจขึ้นอยู่กับคอนฟิกูเรชันและเกณฑ์การทำงานที่ระบุเฉพาะ

ข้อมูลเกี่ยวกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM ได้มาจากผู้จัดจำหน่ายของผลิตภัณฑ์เหล่านั้น คำประกาศที่เผยแพร่หรือแหล่งข้อมูลที่เปิดเผยต่อ สาธารณะ IBM ไม่ได้ทดสอบผลิตภัณฑ์ดังกล่าว และไม่สามารถยืนยันความถูกต้องของประสิทธิภาพ ความเข้ากันได้ หรือการเรียกร้องอื่นใดที่เกี่ยวข้องกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM หากมีคำถามเกี่ยวกับความสามารถของผลิตภัณฑ์ที่ไม่ใช่ของ IBM ควรสอบถามกับ ผู้จัดจำหน่ายของผลิตภัณฑ์ดังกล่าว

ข้อความใดๆ ที่เกี่ยวข้องกับทิศทางในอนาคตและเจตจำนงค์ของ IBM อาจมีการเปลี่ยนแปลง หรือเพิกถอนได้โดยไม่ต้องแจ้งล่วงหน้า และนำเสนอเฉพาะเป้าหมาย และวัตถุประสงค์เท่านั้น

ราคาที่แสดงทั้งหมดของ IBM เป็นราคาขายปลีกที่แนะนำของ IBM ในปัจจุบัน และอาจเปลี่ยนแปลงได้โดยไม่ต้องแจ้งให้ทราบ ราคาของผู้แทนจำหน่ายอาจแตกต่างออกไป

ข้อมูลนี้ใช้สำหรับวัตถุประสงค์การวางแผนเท่านั้น ข้อมูลในเอกสารฉบับนี้อาจมีการเปลี่ยนแปลง ก่อนที่ผลิตภัณฑ์ที่กล่าวถึงจะมีจำหน่าย

ข้อมูลนี้ประกอบด้วยตัวอย่างข้อมูลและรายงานที่ใช้ในการดำเนินธุรกิจ ประจำวัน เพื่อแสดงให้เห็นอย่างสมบูรณ์ที่สุดเท่าที่จะเป็นไปได้ ตัวอย่างเหล่านี้จึงประกอบด้วย ชื่อของบุคคล บริษัท ตราสินค้า และผลิตภัณฑ์ ชื่อเหล่านี้ทั้งหมดเป็นชื่อสมมติ และความคล้ายคลึงใดๆ กับบุคคล หรือองค์กรธุรกิจที่มีอยู่จริง ถือเป็นเหตุบังเอิญ

ใบอนุญาตลิขสิทธิ์:

ข้อมูลนี้ประกอบด้วยโปรแกรมแอ็พพลิเคชันตัวอย่างในภาษาต้นฉบับ ซึ่งแสดงเทคนิคในการเขียนโปรแกรมบนแพล็ตฟอร์มปฏิบัติการที่หลากหลาย คุณสามารถคัดลอก ปรับเปลี่ยน และแจกจ่ายโปรแกรมตัวอย่างเหล่านี้ในรูปแบบต่างๆ ได้โดยไม่ต้องชำระเงินให้แก่ IBM เพื่อใช้สำหรับการพัฒนา การใช้งาน การตลาด หรือการแจกจ่ายโปรแกรมแอ็พพลิเคชันที่สอดคล้องกับ

อินเทอร์เฟซโปรแกรมแอ็พพลิเคชันของแพล็ตฟอร์มการดำเนินงานที่เขียนโปรแกรมตัวอย่าง ตัวอย่างเหล่านี้ยังไม่ได้ผ่านการทดสอบในทุกสภาพ ดังนั้น IBM จึงไม่สามารถรับประกันหรือแจ้งถึงความน่าเชื่อถือ การให้บริการได้ หรือฟังก์ชันของโปรแกรมเหล่านี้ได้ โปรแกรมตัวอย่างมีการนำเสนอ "ตาม สภาพ" โดยไม่มีการรับประกันประเภทใดๆ IBM ไม่ต้องรับผิดชอบต่อความเสียหายใดๆ ที่เกิดขึ้นจากการใช้โปรแกรมตัวอย่างของคุณ

แต่ละสำเนาหรือส่วนใดๆ ของโปรแกรมตัวอย่างเหล่านี้ หรืองานที่สืบเนื่องใดๆ ต้องมีคำประกาศ ลิขสิทธิ์ดังนี้:

© (ชื่อบริษัทของคุณ) (ปี)

ส่วนต่างๆ ของรหัสนี้ได้มาจากโปรแกรมตัวอย่างของ IBM Corp.

© ลิขสิทธิ์ IBM Corp. _ป้อนปี_

สิ่งที่ต้องพิจารณาเกี่ยวกับนโยบายความเป็นส่วนตัว

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.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ดัชนี

A

C

พิมพ์ในสหรัฐอเมริกา