Programming for UNIX 95 and HP-UX Binary Compatibility

This paper explains how HP-UX release 10.10 was modified to obtain the UNIX 95 brand while still providing compatibility with previous releases of HP-UX. Simple code examples and workarounds are provided for C language programmer and shell script programmers. The UNIX 95 brand was the result of the "Spec1170" initiative to provide a "unified UNIX".

1.0 Introduction

HP-UX release 10.10 is the first UNIX system available that conforms to the UNIX 95 Profile of the Single UNIX Specification (hereafter referred to as UNIX 95). This release enables the development of applications that will be portable to other systems that conform to UNIX 95.

UNIX95 defines a set of application programming interfaces (API's). Applications that limit their use of system interfaces to these API's will be portable to any UNIX95 conforming system.

Applications developed on this release and previous releases of HP-UX will continue to operate as they did on previous releases provided the user has not enabled the UNIX 95 environment. UNIX 95 conforming applications developed on this release will only run on the HP-UX 10.10 and later releases.

Further information regarding X/Open, UNIX 95 and the Single UNIX Specification see the X/Open Web page at http:/www.Xopen.org.

2.0 Configuring the system to conform to the UNIX 95 environment

Release 10.10 of the HP-UX System can be updated to support the UNIX 95 configuration by installing the following code patches.

TABLE 1.

Patch	**Component**
PHCO_6587	df(1)
PHCO_6633	ex(1), vi(1)
PHCO_6705	od(1)
PHCO_6712	stty(1)
PHCO_6772	getconf(3)
PHCO_6777	libc
PHCO_6820	read(1)
PHCO_7035	ps(1)
PHKL_6765	POSIX_UPE
PHNE_6688	mailx(1)
PHNE_6726	STREAMS (s800)
PHNE_6727	STREAMS (s700)

Installing these patches enables a user to configure a session so the commands and libraries will conform to the UNIX 95 profile.

To configure a UNIX 95 session, the user has to set the environment variable `UNIX95` and also set the `PATH` environment variable to contain the entry `/usr/bin/xpg4` before the `/bin` and `/usr/bin` entries. For example, the following entries in the users' `.profile` enable the UNIX95 environment:

```
UNIX95=
export UNIX95
PATH=/usr/bin/xpg4:/bin:/usr/bin
export PATH
```

Once a session has been configured to the UNIX 95 profile, users should restrict their usage of the system to commands and libraries that are strictly conforming. Commands and libraries are **strictly conforming** if they use only the API's described in the UNIX 95 profile or they use other commands or libraries that are themselves strictly conforming. Commands and functions that pre-date the UNIX 95 profile are denoted in this paper as **Classic** HP-UX commands and functions. Use of API's that are outside the scope of the UNIX 95 profile while the session is configured to the UNIX 95 is not supported[1].

The purpose of the UNIX 95 profile is to enable portability between systems implemented by different vendors. There are two reasons why mixing UNIX 95 API's with Classic API's is not wise: 1) If an application uses API's that are outside the scope of the Single UNIX Specification, there is no guarantee that the function will be available on other UNIX 95 conforming platforms and, 2) some Classic API's conflict with the UNIX 95 definition of API's of the same name or may conflict with other UNIX 95 API's. Over time, applications will be altered to be conforming and will be ported to the other platforms. When this happens, these API's may be used by an application. During a transition period, applications will be migrated to the UNIX 95 environment. For these reasons, the restriction of strict conformance is not considered unreasonable.

3.0 Programming with Commands

3.1 Compatibility with previous versions of HP-UX

To allow the introduction of UNIX 95 behavior that is incompatible with Classic HP-UX, the commands that exhibit incompatible behavior have been modified to exhibit UNIX95 behavior if the UNIX95 environment variable is set. Commands that are not easy to implement to support both the Classic HP-UX behavior and the UNIX 95 behavior are placed in a separate `bin` directory. This means, to complete the configuration of a user session for conformance to UNIX 95, the user also has to begin the search path environment variable with `/usr/bin/xpg4`.

Application developers should modify their applications to function properly regardless of whether the user has configured their session to be a Classic HP-UX environment or a UNIX 95 environment. To help developers assess how much the UNIX 95 environment might impact their application, the following four tables describe how key commands behave with the UNIX 95 environment enabled.These tables are grouped by the expected impact on applications that presently operate with the commands in the Classic mode. To obtain more detailed information regarding these changes, the developer should refer to the individual manual pages for these commands.

The following table lists command changes due to UNIX95 that may affect applications

1. Since the UNIX 95 profile of the XPG4 introduces functions that are not compatible with Classic HP-UX, and Hewlett-Packard is committed to protect the customers software investment, this release has Classic as the default behavior. Many Classic libraries and applications have not yet been modified to be strictly conforming. This release of HP-UX is intended to be used by application developers so they may alter their applications and libraries to be strictly conforming. A future release may make UNIX95 the default configuration.

by causing scripts to break, abnormal program termination or inappropriate behavior:

TABLE 2.

Command	UNIX95 change from Classic behavior
delta	Different output format.
df	Different output format.
ex	Different command processing, regular expression processing, substitute processing; new error conditions; `-t` semantics altered.
get	Different output format.
nl	Text line numbered if only graphic characters are present; only 1 file at a time is permitted
od	Output different with ‘`-s`’, ‘`-A d`’ ‘`-A o`’, ‘`-A x`’, ‘`-A n`’, ‘`-N` *`count`*’ option
pr	Time column formats changed; number of form feed characters different with pr `-F` command; output different with `-sc` and `-columns` options; output different if the following options are used together “`-ad -Frt -e -h`“*my_header*” `-i -l 54 -n -o 2 -s -w 72`”; exit code different when option arguments are missing
ps	Time column formats changed; column headings different; `-a -d -g -s` process selected by session nor process groups; uid/user column with `-f` and `-l` displays effective not real users; `-u` *users* selected based on effective UID not real UID; field widths/alignment changed
sh	Error conditions encountered in built-in commands not specified by XPG4 will no longer cause subshell processing to exit the subshell.
sort	Exit value has changed with `-o` option; behaves differently for the following two conditions: 1) when individual options are supplied 2) when options are grouped and supplied; The order of different options relative to one another produce different outputs; When the character *i* is specified in conjunction with a key description, then a call to sort does not interpret these modifier in the same manner as the corresponding option but only applies the modifier to the specific key, overriding any option specifications, rather than to all key descriptions; behaves differently for files starting with a “`/`”, “`..`”, ”`.`” and of the type “`f1/f2/.../fn`”
wc	Syntax altered; `-c` is now mutually exclusive to `-l` and `-w`; output format is altered.
who	`-H`, `-T` have major output changes; `-s` is now mutually exclusive with `-T, -a, -d`

The following table lists command changes that may affect applications by causing them to produce erroneous results:

TABLE 3.

Command	UNIX95 change from Classic behavior
cal	Error messages now go to stderr not stdout; hard-coded day/month now use abbreviated information in the locale; outputs two months in a row, not three
ctags	EXIT returns non-zero for failure; `-t` (create tags for `typedefs`) is now the default behavior
cu	`-d` now gives the MAX level of debugging; `-D` is a work-around to permit old behavior
dd	Appends '`\n`' at the end of the last (partial) conversion buffer for conv-unblock; conversion "blocks" at the end of every input buffer when the input buffer is not an even multiple of the conversion buffer.
du	`-r` (prints messages about unreadable directories) is now the default behavior; `-s` and `-a` are now mutually exclusive
m4	Behavior of the `ifdef` built-in macro changed; Added options delimiter `--`; Error flagged when a non-numeric argument is passed to the following built-in macros: `decr, divert, incr, m4exit, substr, undivert, eval`; When the first argument of built-in '`ifdef`' macro is not defined or is zero and there is no third argument, then the macro value set to null; The TEXT which is before '`m4exit`' is printed; When `m4wrap` is invoked more than once, then the arguments are processed in the same order after reading the end-of-file character; m4 passes when it is invoked with a 'space' between the option and its argument; when multiple files are given to '`m4`', even it fails to perform action on first file, it does not continue with next file; Output of the macro '`dumpdef`' goes to stdout instead of stderr.
make	Options delimiter `--` added; Exit status is zero when a target failed with `-k` option; With `-q` option for and out-of-date target returns 1; Option `-S` overrides option `-k`; Environment variable MAKEFLAGS are used; Operands processed in the order specified in the command line; Command-line macros are added to the MAKEFLAGS variable; When the target is up to date, "target up to date" message is written on stdout. Earlier, no message was written.; Exit code changed; MAKEFLAGS defined in the makefile overrides the environment macro of the same name; The .SILENT flag affects only specified targets; File listing modified in case of the `$?` macro; Output format changed in case of the `.c`, `.f` and the `.sh` suffix rules.
wc	Results no longer given per option specified now only one set of results provided in a fixed order; '`\v`', '`\f`'and '`\r`' are also excepted as delimiters to the word

The following table lists command changes that may affect applications because they have mutually exclusive options or other recoverable problems:

TABLE 4.

Command	**UNIX95 change from Classic behavior**
admin	Unreadable files silently ignored
cmp	Behaves differently for files starting with type “`F1/F2`”; exit values changed with not read only files i.e. whose read bit is not set.
cp	The `-f` and `-i` options are no longer mutually exclusive.
find	A period (`.`) in a filename is exactly matched by using a period as the first character or immediately following a slash character in the pattern.
lex	`-v` and `-n` options are mutually exclusive; `-v` and `-t` now output to stdout or stderr; if no input file specifications or `-` is used, stdin is used
locale	“`locale -k frac_digits int_frac_digits`” is changed to return -1.
ls	`-t` comparison is done by collating sequence if modified times are equal; first newline char is removed if first item is a directory.
mkdir	The mkdir with the `-m` option sets the mask as per the permissions provided. The remaining bits are retained as per the `umask`.
mkfifo	The mkfifo with the `-m` option sets the mask as per the permissions provided for the specified ‘*who*’. The other bits are retained as per the `umask`.
paste	Ignores unreadable files silently; returns with exit status 1 when one of the operands is invalid.
printf	External appearance is changed for unconverted strings.
prs	Unreadable files are silently ignored; `-c` and `-d` arguments are mandatory; if error occurs, processing continues but exits with non-zero status.
rmdel	Unreadable files are silently ignored.
sact	Unreadable files are silently ignored; When fopen returns zero for the file name containing the module, the error message “`No outstanding deltas for: %s`” will be printed on the stdout instead of stderr.
stty	When setting the speed of a terminal port, both the input and output speed are set to the same value.
unget/sact	Unreadable files are silently ignored; If error occurs, processing continues but exits with non-zero status

The following table lists command changes that may affect applications because they have error return values changed or other minor changes:

TABLE 5.

Command	UNIX95 change from Classic behavior
asa	When the first character removed from the input line is <space>, then asa outputs the rest of the input line without change; When the first character removed from the input line is '0', then asa outputs a newline character followed by the rest of the input line; When the first character removed from the input line is '1', then the asa outputs one or more characters that causes an advance to the next page followed by the rest of the input line; When the first character removed from the input line is '+', then the asa replaces the newline character of the previous line with one or more implementation-dependent characters that causes printing to return to column position 1 followed by the rest of the input line; When the first character removed from the input is '+', then the asa outputs the rest of the input line without change; When no file operands are specified, then the asa uses standard input; the characters `--` are specified to a command which accepts operands to delimit the end of options. Any arguments following the `--` which start with a `-` are considered by the command to be operands and not as options; processes operands in command line order; When an input file is defined as a text file, then at least `LINE_MAX` bytes can be accumulated from a set of continued input lines; When no error occurs during the execution of a utility, then no error messages are written to standard error and the exit status from the utility is zero; When a utility is unable to perform the requested action on an external object (file, directory, user, process etc.) specified by an operand, then the utility issues a diagnostic message to standard error and continues processing subsequent operands. The final exit status of the utility is non-zero.
bc	Very large numbers are split across lines with 70 characters, this includes '\' and '\n' characters; scale(<zero>) is 0 even though bc is invoked with `-l` option.
cancel	When an error occurs during execution of a utility, then the diagnostic message is written to standard error; Exit status is non-zero. The exit value is now 2.
cd	An absolute pathname of the new working directory are written to the standard output.
compress	When an error occurs while processing one of the operands, the final exit value is non-zero.
date	In the classic behavior, this command prompts for confirmation in case the date is to be set backward. This behavior will change in the XPG4 version because the stdin is NULL for this command and hence the user cannot be prompted.
expr	When performing a match, if the two regular expressions are strings and if there is no match, then null string is written to standard out. A newline character is not written; `expr` supports nested parenthesis. The sub-expression can be nested to any depth.

TABLE 5.

Command	**UNIX95 change from Classic behavior**
grep	`grep` with `-l` option stdin as input should give on stdout "`(standard input)\n`" message. Same for `-l -m`; `grep` on regular expression should print correct result and return exit 0 for successful grep with `-e, -f, -i, -x,` and `-v` options; Null pattern should select every input line. Same with `-e, -f, -v`, options; Null pattern with option `-E` should select every input line; RE "`\`" should work; grep `-F` and fgrep results should match; Also match with similar valid parameters applied to both; With `-q` option and first inaccessible input file, grep should exit with zero status
localedef	If the implementation supports the `POSIX2_C_BIND` option then localedef should be `system()`'d and `popen()`'d; invoking localedef with invalid input should return error code 3; The characters `--` can be specified to a command which accepts operands to delimit the end of options. Any arguments following the `--` which start with a `-` are considered by the command to be operands and not as options; `-i` option on should create return and exit 0; Should read from stdin if `-f` option not specified. Should return 0 if succeeds; symbolic constant `POSIX2_LOCALEDEF` is defined
lpstat	Previously did not support the end-of-options (`--`)
renice	Better error processing
sed	Errors should be redirected to stderr; should work when label length is eight characters; A call to `sed -f` *script_file* accepts a script_file consisting of editing commands, one per line; the sed command '`/A/p`' command produces correct results; script '`/BRE/p`' should print correct results with zero exit status; When the editing command `D` does not delete the whole of the current pattern space, then the next cycle of editing commands is applied to the remaining pattern space; The editing command `H` appends to the hold space a <*newline*> followed by the contents of the pattern space; `r` <*pathname*> command should return with zero exit status for correct operation; `w` <*pathname*> command should return with zero exit status for correct operation; The editing command `x` switches the contents of the pattern and hold spaces.
strings	Non zero exit value on failure.
tabs	Detects new error conditions.
time	Exit value changed for error conditions: cannot find executable, cannot invoke
type	Any errors occurred should be output to stderr; When an error occurs, the exit status should be non-zero.
uncom-press	When an error occurs while processing one of the operands, the final exit value is non-zero.
unexpand	There is no space character preceding the tab character in the output.
uucp	Filename to be transferred can be specified using metacharacters like `?, *..`
uulog	Change in the `/var/uucp/.Log` directory permission from the existing value of only "`r`" for all, to "`r+w`" for all.
uupick	Exit code changed; Error message is dumped in stderr instead of stdout.
uustat	Exit code changed; Error message is dumped in stderr instead of stdout.

TABLE 5.

Command	UNIX95 change from Classic behavior
uuto	Exit code changed; Options delimiter "`--`" added; Error message is dumped in stderr instead of stdout.
uux	Exit code changed; Options delimiter "`--`" added; Options `-j` and `-n` can be grouped; Output written changed if command fails; Works if order of options changed; Pathname expansion is carried out; Alias substitution is performed in shell pipeline processed; Works for absolute pathname; Works for pathname preceded by ~*name*; When ~ is used, expand to the `PUBDIR` value; Picks a file from the current directory if no file path specified; `uux` accepts input from stdin; `uux` works if non-local filenames must be unique within the `uux` request.
what	Better error processing for invalid options
xargs	Non zero exit value on failure.
zcat	When an error occurs while processing one of the operands, the final exit value is non-zero.

3.2 Workaround

Until an application has been modified to tolerate both the Classic HP-UX behavior or the UNIX 95 behavior of system commands, it can be altered with a simple workaround that will allow the command to continue to function properly. Clearing the `UNIX95` environment before invoking any UNIX 95 command will cause the command to execute with its Classic behavior.

4.0 Programming with the C language

As with command programming, the code development engineer must restrict API usage to a strictly conforming (see page 2) UNIX 95 definition if the UNIX 95 development environment is enabled. If an application, developed with the UNIX 95 development environment, uses libraries supplied by applications, those libraries must be strictly conforming. This transitive property of strict conformance must be adhered to so that incompatible Classic functions are not linked with a program that is designed to operate with UNIX 95 functions. Not only are function semantics different, but some data structures are different, depending upon whether they are compiled in the Classic or UNIX 95 environments.

Mixing the Classic and UNIX 95 development environments is not a supported configuration. Partially enabling the UNIX 95 development environment is not a supported configuration. Either of these two conditions may lead to unexplained application behavior or abnormal application termination.

Mixing relocatable modules or shared libraries compiled in the Classic development environment with modules compiled in the UNIX 95 development environment may lead to unexpected operation since data structures may be different depending upon the environment. For instance, the signal context structure differs depending upon the development environment. Applications that use the UNIX 95 environment and the signal context must insure all functions that use the context structure are compiled in the UNIX 95 environment.

4.1 Configuring the development environment to conform to the UNIX 95 environment

In addition to setting the `UNIX95` environment variable and the `PATH` environment variable, the development engineer has to set the macro definition `_XOPEN_SOURCE_EXTENDED` in the application source code or as a compile time option to the C compiler. If the development engineer fails to set the `PATH` environment variable, then the linker will report undefined external symbols for any references to the "context" functions.

Once the development environment has been properly configured to the UNIX 95 environment, the relocatable modules produced will be consistent with the functions in the system libraries when the application is linked. Once an application is linked, it is identified as a UNIX 95 application. The system will treat applications that are identified as UNIX 95 differently than those that are unmarked (the default, Classic mode). These applications will operate as UNIX 95 applications even if they are executed with the UNIX 95 environment disabled.

4.2 Process Signals

Two new signals have been added to HP-UX to conform to the UNIX 95 API. These are `SIGXCPU` and `SIGXFSIZE`. The former is generated by the system when an application exceeds the number of CPU seconds to which it has been limited. The later is generated by the system when an application attempts to write to a file that will cause it to grow larger than the number of blocks to which it has been limited.

By default, the limits for CPU seconds and file size are each set to infinity. Unless the user sets these limits to some value less than infinity, they will not affect any application. If the user lowers the limits for CPU seconds or file size, and an application exceeds either of these limits, the system will not deliver either of these signals to the process if it is a Classic application. However, if the user or an application explicitly directs the system to deliver either of these signals to a process, even if it is a Classic application, the system will deliver the signal.

It must be assumed that applications may be run on systems where the user wishes to limit CPU time or file size. For this reason, all applications (including Classic applications) that catch signals and are expected to run on a wide range of systems should be modified to handle these signals. If this change is not made, an application may experience data corruption if the system delivers a signal that it could have caught.

4.2.1 The SIGXFSIZ signal

This new signal has been added to the system and will be delivered by the kernel to a UNIX 95 application whenever the application attempts to write into a file that will cause the file to grow beyond the limit set for the application. The default action for a UNIX 95 application is to kill the process and leave a core file. If the application is a Classic application, the kernel will not send the signal as a result of the file size exceeding the application limit. However, if an application explicitly sends this signal to a Classic application, the signal will be delivered and the default action will be taken.

4.2.2 The SIGXCPU signal

This new signal has been added to the system and will be delivered by the kernel to a UNIX 95 application whenever the application exceeds the CPU time limit set for the application. The default action for a UNIX 95 application is to kill the process and leave a core file. If the application is a Classic application, the kernel will not send the signal as a result of exceeding the application limit. However, if an application explicitly sends this signal to a Classic application, the signal will be delivered and the default action will be taken.

4.2.3 The sigcontext structure

The sigcontext data structure was modified to have a clean namespace. Applications must not access elements of this structure that are not described in the UNIX 95 API.

4.2.4 Mixing signal paradigms

Using the Classic signal paradigm provided by the functions `bsdproc`, `signal`, `sigvector`, `sigblock`, `sigsetmask` or `sigspace` with the UNIX 95 signal paradigm may cause undetermined application behavior. The UNIX 95 function `bsd_signal` may be used in place of `signal` if appropriate.

4.3 The nftw function

The function `nftw` has been modified to conform to UNIX 95. This change occurs in the fourth argument passed by `nftw` to the application specified function. This parameter is a call-by-value parameter in the Classic environment and is a call-by-reference (pointer) in UNIX 95 environment.

Applications that use the Classic version of `nftw` should not be mixed with relocatable modules compiled in the UNIX 95 environment.

4.4 The sigpause function

The function `sigpause` has been modified to conform to UNIX 95. The Classic version takes a signal mask as an argument. UNIX 95 version takes a signal number as an argument.

When the UNIX operating system was first introduced it provided less than 32 unique signals. Over the intervening years, implementations provided API’s that passed the signal information from an application to the system via a bit-mask, with each bit representing one of the signals. Since the *long int*eger declaration was “guaranteed” to be at least 32-bits wide, it was used to implement this bit-mask.

Now that the number of signals supported by the kernel exceeds the number of signals that can be represented by the signal mask, using the Classic version of `sigpause` (or any other function that uses a 32-bit signal mask) will not function properly with functions that enable the additional signals.

4.5 The setpgrp function

The `setpgrp` function has been modified to conform to the UNIX 95 API. The Classic version of `setpgrp` does not change the session ID. The UNIX 95 version of `setpgrp` sets the session ID to PID when the process group leader is not also the session leader.

The `setsid` HP-UX manual page has stated for years that `setpgrp` is provided for backward compatibility only. All applications should be modified to use the `setsid` function instead of `setpgrp`. To understand why `setpgrp` is not a preferred interface, See “Setpgrp” on page 14 for a historical perspective of the changes the function `setpgrp` has undergone.

4.6 Mixing library files

Library files that depend upon the Classic behavior cannot be mixed with libraries that depend upon UNIX 95 behavior. If a library has been modified to `setpgrp3` or `nftw2` (as was recommended in the 10.0 release), they can be used with a UNIX 95 application that uses `setpgrp` and `nftw`.

It may not be possible to change the source code that an application depends upon because it uses functions provided by a third party library. To accommodate this situation, the following solutions will suffice. While these workarounds violate the principle of strict conformance, they will work until the application can be modified to completely conform to the UNIX 95 API’s.

4.6.1 setpgrp

If the source code for an application that depends upon the Classic behavior cannot be obtained, and no other part of the application depends on the UNIX 95 behavior of the function `setpgrp`, the application developer can include a relocatable module when linking the application that contains the code illustrated in the following example.

```
#include <signal.h>

int
setpgrp(pid, pgrp)
int pid;
int pgrp;
{
    return setpgrp3(pid, pgrp);
}
```

This function will allow the developer to link an unmodified relocatable module to the Classic `setpgrp` function.

4.6.2 nftw

If the source code for an application that depends upon the Classic behavior cannot be obtained, and no other part of the application depends on the UNIX 95 behavior of the function `nftw`, the application developer can include a relocatable module when linking the application that contains the code illustrated in the following example.

```
#include <ftw.h>

int
nftw(filename, fn, flag, depth)
char * filename;
int (*fn)();
int flag;
int depth;
{
    return nftw2(filename, fn, flag,
                     depth);
}
```

This function will allow the developer to link an unmodified relocatable module to the apparently Classic `nftw` function.

Appendix

Hewlett-Packard has met customer programming needs by implementing key specifications, such as the X/Open Portability Guide (XPG), the System V Interface Definition (SVID) and the Application Environment Specification (AES). However, the combination of some of these specifications creates a situation where programming interfaces that differ in functionality have the same name.

History

As a result of an industry effort in 1994, X/Open augmented the X/Open Portability Guide release 4 (XPG4), nearly doubling the number of Application Programming Interfaces (API) that will be common on all UNIX™ System platforms. In a related event, Novell Corporation transferred the rights to the UNIX trademark to X/Open. This transfer strengthened the UNIX System market by associating the UNIX trademark and brand with a unified, open specification rather than the proprietary implementation of an operating system. By design, HP-UX 10.0 already contains many of these API's, enabling near-term conformance to the XPG4v2 specification for the Single UNIX.

To accomplish the expansion of the XPG4 API set, a team of system vendors unified the set of existing API's that were described in other specifications. These included the

SVID from Novell Corporation and the AES from the Open Software Foundation. Additional API's were included due to their industry acceptance, as demonstrated by their use in popular applications running on one or more UNIX System platform.

Compatibility Challenge

Every effort was made to minimize the conflict these API's might have on systems that implement functions that have the same name but are subtly different. As might be expected in an imperfect world, this did inevitably lead to name collisions that could not be resolved by compromising the specification. When this situation occurred, a decision was made to select from the set of conflicting specifications with the understanding that this would put some implementations temporarily out of conformance with the resulting specification. Although the selection may seem arbitrary, considerable thought was given to the impact that would result from choosing between conflicting API's. Every effort was made to incorporate as many API's as were warranted by industry usage in popular applications.

After the set of API's was selected, Hewlett-Packard Company engineering staff designed an implementation of HP-UX that would conform to the new version of the XPG4. Due to HP's commitment to our installed base, a goal was set to minimize the impact this new version of HP-UX would have on existing customers and application developers. Of the approximately 1,170 API's described in XPG 4 version 2, HP-UX had implemented all except 25 API's prior to release 10.0. Of the API's that were not implemented, all except four of these API's could be implemented in release 10.0 without creating a conflict with the API's that presently constitute HP-UX.

Once the four conflicting API's had been identified, the development staff created an appropriate migration strategy for the existing customer base. This migration strategy provides several alternatives the customer or application developer can use to retain their investment in software while moving forward to release 10.10 which implements the common interface definition that is defined in XPG4v2.

Function Changes

Setpgrp

To understand how subtly the semantics of this function have changed, the following excerpts will help application development engineers understand the differences.

SVID-1 ('85)

> Setpgrp sets the process group ID of the calling process to the process ID of the calling process and returns the new process group ID.

SVID-2 ('86)

The function `setpgrp` sets the process-group-ID of the calling process to the process-ID of the calling process and returns the new process-group-ID.

XPG/2 ('87)

`Setpgrp` sets the process group ID of the calling process to the process ID of the calling process and returns the new process group ID.

Unless the process is already a process group leader, `setpgrp` disassociates the process from the terminal group, if any.

SVID-3 ('89)

If the calling process is not already a session leader, the function `setpgrp` sets the process group ID and session ID of the calling process to the process ID of the calling process, and releases the calling process's controlling terminal.

XPG/3 ('89)

WITHDRAWN (superseded by `setsid`).

HP-UX

If the calling process is not a process group leader, `setsid` or `setprgp` creates a new session. The calling process becomes the session leader of this new session, becomes the process group leader of a new process group, and has no controlling terminal. The process group ID of the calling process is set equal to the process ID of the calling process. The calling process is the only process in the new process group, and the only process in the new session.

`setprgp` returns the value of the process group ID of the calling process.

XPG/4v2 ('94)

If the calling process is not already a session leader, `setpgrp` sets the process group ID of the calling process to the process ID of the calling process. If `setpgrp` creates a new session, then the new session has no controlling terminal.

The `setpgrp` function has no effect when the calling process is a session leader.

Summary

Hewlett-Packard Company is committed to operating system standards. By working with other organizations during the formulation of these standards, they can be completed without a significant impact on applications already operating on HP-UX. When it is not possible for these changes to be made without affecting applications running on HP-UX, white papers similar to this one will be published to provide an easy migration path to future releases.