Hewlett Packard Enterprise

UNIX95 のプログラミングと HP-UX のバイナリ互換性

本書では、以前のリリースの HP-UX との互換性を維持しながら UNIX 95 のブランドを獲得するために、 HP-UX リリース 10.10 をどう変更したかについて説明します。 C 言語のプログラマやシェルスクリプトのプログラマのために、簡単なコード例と動作環境を提供します。 UNIX 95 のブランドは、「統一 UNIX」を実現するための「Spec1170」イニシアチブの成果として、各製品が統一 UNIX 仕様に準じていることを示しています。

本書は、英文資料 "Programming for UNIX95 and HP-UX Binary Compatibility"の翻訳です。

《 2000 年時点》

はじめに

HP-UX リリース 10.10 は、統一 UNIX 仕様の UNIX 95 プロファイル (以降、UNIX 95 と呼びます) に準拠する UNIX システムとして、他社に先駆けて 96 年春に出荷開始されました。 このリリースを使用して開発したアプリケーションは、UNIX 95 に準拠する他のシステムに移植できます。
UNIX 95 では、一組のアプリケーションプログラミングインタフェース (API) を定義します。 これらの APInv に合わせてシステムインタフェースを使用するアプリケーションは、UNIX 95 に準拠する任意のシステムに移植できます。
このリリースと以前のリリースの HP-UX で開発したアプリケーションは、ユーザが UNIX 95 環境を有効にしていない限り、以前のリリースの場合と同じように動作します。このリリースで開発された UNIX 95 準拠のアプリケーションは、HP-UX 10.10 以降のリリースでのみ動作します。

―目次―

UNIX95 環境に合わせたシステムの設定

UNIX95 の環境設定をサポートするようリリース 10.10 の HP-UX システムを更新するには、次のコードパッチを組み込みます。

パッチ	コンポーネント
PHCO_6587	df(1)
PHCO_6633	ex(1)、vi(1)
PHCO_6705	od(1)
PHCO_6712	stty(1)
PHCO_6772	getconf(3)
PHCO_6809	libc
PHCO_7499	read(1)
PHCO_7035	ps(1)
PHKL_6765	POSIX_UPE
PHNE_6688	mailx(1)
PHNE_6726	STREAMS(s800)
PHNE_6727	STREAMS(s700)

これらのパッチを組み込めば、ユーザはセッションを設定できるため、コマンドとライブラリが UNIX95 プロファイルに準拠します。

UNIX95 セッションを設定するには、ユーザは環境変数 UNIX95 を設定し、さらに環境変数 PATH を設定して /bin エントリと /usr/bin エントリの前に /usr/bin/xpg4 エントリを書き込む必要があります。 たとえば、ユーザのプロファイルに次のエントリを書き込めば、UNIX95 の環境が有効になります。

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

セッションに UNIX95 プロファイルが設定されたら、ユーザは、厳密に準拠するコマンドとライブラリだけに限定してシステムを使用してください。 コマンドとライブラリは、UNIX95 プロファイルに記述された API だけを使用する場合、または元から厳密に準拠している他のコマンドやライブラリを使用する場合に、厳密に準拠するものと言えます。 UNIX95 プロファイルに先行するコマンドと関数は、本書では Classic HP-UX のコマンドと関数として示します。 セッションが UNIX95 に設定されている場合、UNIX95 プロファイルの範囲に含まれない API の使用はサポートされていません。（*注）

UNIX95 プロファイルの目的は、異なるベンダーによって開発されたシステム間で移植性を実現することにあります。 UNIX95API と Classic API の混在が望ましくない理由は、次に示すように 2 つあります。

1. アプリケーションが統一 UNIX 仕様の範囲に含まれない API を使用する場合、その関数が UNIX95 に準拠する他のプラットフォームでも使用できるという保証はありません。
2. いくつかの Classic API は、同じ名前の API の UNIX95 定義と衝突したり、他の UNIX95API と衝突することがあります。 時間の経過と共に、アプリケーションは準拠するように変更され、他のプラットフォームに移植されます。 その場合、アプリケーションはこれらの API を使用できます。 過渡期においては、アプリケーションは UNIX95 環境に移行されます。 以上のような理由から、厳密な準拠という制約は、不合理なものとはいえません。

*注)　XPG4 の UNIX95 プロファイルは Classic HP-UX とは互換性のない関数を提供し、ヒューレットパッカードは顧客のソフトウェア投資を保護することを確約しているため、このリリースではデフォルトとして Classic の動作を行ないます。 Classic の多くのライブラリとアプリケーションは、まだ厳密に準拠するように変更されていません。 このリリースの HP-UX の目的は、アプリケーション開発者がこれを使用して、自分のアプリケーションとライブラリを厳密に準拠させるよう変更することにあります。 将来のリリースでは、UNIX95 をデフォルト設定とする可能性があります。

コマンドによるプログラミング 1

以前のバージョンの HP-UX との互換性

Classic HP-UX と互換性のない UNIX95 動作の導入を可能にするために、 互換性のない動作を行なうコマンドを変更して、 環境変数 UNIX95 が設定されている場合は UNIX95 の動作を行なうようにしました。 Classic HP-UX の動作と UNIX95 の動作の両方をサポートすることが困難なコマンドについては、 別個の bin ディレクトリに格納しました。 つまり、 UNIX95 に準拠するユーザセッションの設定を行なうには、 ユーザは検索パスの環境変数を /usr/bin/xpg4 で始めることが必要です。

アプリケーション開発者は、 ユーザがセッションを Classic HP-UX 環境に合わせて設定した場合でも、 UNIX95 環境に合わせて設定した場合でも、 アプリケーションが正しく機能するようにアプリケーションを変更することが必要です。 UNIX95 環境がアプリケーションに与える影響を開発者が評価するための資料として、 UNIX95 環境を有効にした状態で主要なコマンド群の動作内容を次の 4 つの表に示します。 これらの表は、 Classic モードのコマンドで現在動作するアプリケーションに対する予想効果によって分類されています。 これらの変更の詳細については、 これらのコマンドのマニュアルページを参照してください。

次の表では、スクリプトの中断、プログラムの異常終了、または不適切な動作を招くことによりアプリケーションに影響を与える可能性のある、UNIX95 によるコマンドの変化を一覧にしています。

コマンド	Classic の動作から見たUNIX95 の変化
delta	出力フォーマットが異なる。
df	出力フォーマットが異なる。
ex	コマンド処理、正規表現の処理、置換処理が異なる。新しいエラー条件を追加。-t の意味が異なる。
get	出力フォーマットが異なる。
nl	図形文字だけが存在する場合に限って、テキスト行に番号が付けられる。一度に処理されるファイルは 1 つだけ。
od	'-s'、'-Ad'、'-Ao'、'-Ax'、'-An'、'-N count' オプションによる出力が異なる。
pr	時刻表示行のフォーマットが変更された。pr -F コマンドによる改ページ文字の数が異なる。-sc と -columns のオプションによる出力が異なる。"-ad -Frt -e -h" my_header " -i -l 54 -n -o 2 -s -w 72" の各オプションを一緒に使用した場合の出力が異なる。オプション引数がない場合の終了コードが異なる。
ps	時刻表示行のフォーマットが変更された。列見出しが異なる。-a -d -g -s によるプロセス選択は、セッションまたはプロセス・グループによって行われる。-f と -l によって表示される uid/user 列には、実ユーザではなく実効ユーザ名が使われる。-u users を用いると、実 UID ではなく実効 UID に基づいて選択される。フィールド幅/区切りが変更された。
sh	XPG4 で指定されていない組み込みコマンドでエラー条件が発生しても、サブシェル処理はサブシェルを終了しない。
sort	-o オプションにより終了値が変更された。オプションが個別に指定された場合とオプションがグループ化されて指定された場合の動作が異なる。オプション相互の順序が異なると、出力も異なる。キー記述の中で修飾子 I が多雨替われた場合、sort を呼び出すと、これらの修飾子は対応するオプションとは同じ方法で解釈されず、修飾子は (すべてのキー記述にではなく) オプション指定が無効にされるような特定のキーにだけ適用される。"/"、".."、"." で始まるファイルと、"f1/f2/.../fn" タイプであらわされたファイルでは動作が異なる。
wc	構文が変更された。-c は、-l あるいは -w とは一緒に使えない。出力フォーマットが変更された。
who	-H と -T の出力が大幅に変更された。-s は、-T、-a、-d と一緒に使えない。

コマンドによるプログラミング 2

次の表では、誤った結果を生み出すことによってアプリケーションに影響を与える可能性のあるコマンドの変化を一覧にしています。

コマンド	Classic の動作から見たUNIX95 の変化
cal	エラーメッセージは、stdout ではなく stderr に出力される。ハードコードを使っていた日/月については、ロケール内の省略形を使用する。1 行に 3 ヶ月ではなく、2 ヶ月を連続出力する。
ctags	実行できなかった場合、EXIT は非ゼロを戻す。-t 付き (typedef 用のタグを作成) をデフォルト動作とする。
cu	-d では、最高レベルのデバッグを行なう。-D は、古い動作を使用するための方法である。
dd	conv-unblock 用の最後 (部分) の変換バッファの最後に '\n' を追加する。入力バッファが変換バッファの整数倍ではない場合、すべての入力バッファの最後までをひとつの変換 "ブロック" として扱う。
du	-r (読み取りできないディレクトリに関するメッセージを印刷) をデフォルト動作とする。-s と -a は一緒に使えない。
m4	組み込みマクロ ifdef の動作が変更された。オプション・デリミタに -- が追加された。組み込みマクロ decr、divert、incr、m4exit、substr、undivert、eval に数値以外の引数が渡されると、エラー・フラグが立てられる。組み込みマクロ 'ifdef' の最初の引数が未定義またはゼロであり、しかも 3 番目の引数が存在しない場合、そのマクロ値には null が設定される。'm4exit' の前のテキストは印刷される。m4wrap が複数回呼び出された場合、ファイル終端文字を読み込んだ後で、引数が同じ順序で処理される。m4 は、オプションとその引数の間に空白を付けると無効になる。'm4' に複数のファイルが与えられた場合、たとえ最初のファイルに関するアクションの実行が行われなくても、次のファイルにはその影響が及ばない。マクロ 'dumpdef' の出力は、stderr ではなく stdout に出力される。
make	オプションデリミタに -- が追加された。-k オプションを使えば、ターゲット作成ができなかった場合でも、終了ステータスはゼロとなる。-q オプションの場合、ターゲットが古い場合は 1 を戻す。オプション -S はオプション -k を無効にする。環境変数 MAKEFLAGS が使用される。オペランドはコマンド行で指定された順に処理される。コマンド行マクロが MAKEFLAGS 変数に追加される。ターゲットが最新のものであれば、"target up to date" メッセージが stdout に書き込まれる。以前は、何のメッセージも書き込まれなかった。終了コードが変更された。makefile で定義された MAKEFLAGS は、同じ名前の環境マクロを無効にする。.SILENT フラグは、指定のターゲットにだけ影響を与える。$? マクロの場合のファイルリストが変更された。.c、.f、および .sh の各接尾辞ルールにおける出力フォーマットが変更された。

wc	結果はもはや指定のオプションの順に表示されるのではなく、順番は固定された。'\v'、'\f'、および '\r' も語に対するデリミタとして除外される。

コマンドによるプログラミング 3

次の表では、エラーの戻り値が変更されたり、その他の副次的な変更があるために、アプリケーションに影響を与える可能性のあるコマンドの変化を一覧にしています。

コマンド	Classic の動作から見たUNIX95 の変化
asa	入力行から取り出された最初の文字が空白である場合、asa は残りの入力行を変更なしに出力する。入力行から取り出された最初の文字が '0' である場合、asa は復帰改行文字に続けて残りの入力行を出力する。入力行から取り出された最初の文字が '1' である場合、asa は次のページへ進むための 1 つ以上の文字に続けて残りの入力行を出力する。入力行から取り出された最初の文字が '+' である場合、asa は前の行の復帰改行文字を、印刷を 1 カラムだけ戻す機能のある処理系依存の 1 つ以上の文字で置き換え、その後に残りの入力行を続ける。 入力から取り出された最初の文字が '+' である場合、asa は残りの入力行を変更なしに出力する。ファイルオペランドが指定されていない場合、asa は標準入力を使用する。文字 -- は、オペランドを受け付けるコマンドに対してオプションの最後を区切るために指定される。コマンドは、- で始まる -- の後に続く引数を、(オプションではなく) オペランドとみなす。オペランドはコマンド行の順に処理される。入力ファイルがテキストファイルとして定義されている場合、連続した 1 組の入力行から少なくとも LINE_MAX バイトを蓄積することができる。 ユーティリティの実行中にエラーが発生しなかった場合、エラーメッセージは標準エラーに書き込まれず、ユーティリティからの終了ステータスはゼロとなる。ユーティリティがオペランドによって指定された外部オブジェクト (ファイル、ディレクトリ、ユーザ、プロセスなど) に対して要求されたアクションを実行できない場合、そのユーティリティは標準エラーに診断メッセージを発行して、その後のオペランドの処理を継続する。ユーティリティの最後の終了ステータスは非ゼロである。
bc	きわめて大きな数字は 1 行あたり 70 文字 ('\' と '\n' も含まれる) に分割される。bc がたとえ -l オプションを付けて呼び出されても、scale (<zero>) は 0 である。
cancel	ユーティリティの実行中にエラーが発生した場合、診断メッセージが標準エラーに書き込まれ、終了ステータスは非ゼロとなる。終了値は 2 である。
cd	新しい作業ディレクトリの絶対パス名が標準出力に書き込まれる。

compress	新しい作業ディレクトリの絶対パス名が標準出力に書き込まれる。
date	伝統的な動作では過去の日付を設定する場合、このコマンドは確認を要求する。しかし、XPG4 バージョンにおいては、このコマンドの stdin が NULL でありユーザに入力要求を行なうことができないため、この動作は変更される。
expr	比較を実行するとき、2 つの正規表現が文字列であって一致しなかった場合、標準出力にはヌル文字列が書き込まれる。復帰改行文字は書き込まれない。expr では、括弧の入れ子がサポートされる。部分式は任意の深さまで入れ子にできる。
grep	stdin を入力とする -l オプション付きの grep は、stdout に "(standard input)\n" メッセージを出力する。-l -m の場合も同様。正規表現に対する grep は正しい結果を出力し、-e、-f、-i、-x、および -v オプション付きの成功した grep の場合は exit 0 を戻す。ヌルパターンではすべての入力行を選択する。-e、-f、-v オプションの場合と同様、オプション -E 付きのヌルパターンでは、すべての入力行を選択する。正規表現 "\" は機能する。grep -F と fgrep の結果は一致する。この 2 つに関しては、両方に類似の有効パラメータを適用した場合にも出力は一致する。-q オプションがあり、最初にアクセスできない入力ファイルがある場合、grep はゼロステータスで終了する。
localedef	処理系が POSIX2_C_BIND オプションをサポートする場合、localedef は system() と open() の両関数を使う。正しくない入力で localedef を呼び出すと、エラーコード 3 が戻される。文字 -- は、オペランドを受け付けるコマンドに対してオプションの最後を区切るために指定できる。コマンドは、- で始まる -- の後に続く引数を、(オプションではなく) オペランドとみなす。-i オプションを付けると、戻り値 0 になる。-f オプションが指定されない場合、stdin から読み込む。実行できた場合は 0 を戻す。シンボリック定数 POSIX2_LOCALEDEF が定義される。
lpstat	以前はオプションの終わり (--) をサポートしなかった。
renice	エラー処理を改善する。
sed	エラーは stderr にリダイレクトされる。ラベル長が 8 文字の場合に機能する。sed -f script_file を実行した場合、1 行に 1 つずつの編集コマンドから成る script_file を受け付ける。sed コマンドの '/A/p' コマンドでは正しい結果が得られる。スクリプト '/BRE/p' では、終了ステータス 0 で正しい結果が得られる。編集コマンド D が現在のパターン・スペース全体を削除しない場合、次のサイクルの編集コマンドは残りのパターンスペースに適用される。編集コマンド H は、<復帰改行>に続けてパターン・スペースの内容をホールド・スペースに追加する。r <パス名> コマンドは、正しい動作の場合に終了ステータス 0 で戻る。w <パス名> コマンドは、正しい動作の場合に終了ステータス 0 で戻る。編集コマンド x は、パターン・スペースの内容とホールド・スペースを切り換える。
strings	実行できない場合の終了値は非ゼロ。
tabs	新しいエラー条件を検出。

time	実行可能ファイルを検出できない。呼び出せない。等のエラー条件での終了値が変更された。
type	発生したエラーは stderr に出力される。エラーが発生すると、終了ステータスが非ゼロとなる。
uncompress	オペランドの処理中にエラーが発生すると、最後の終了値は非ゼロとなる。
unexpand	タブ文字に先行して空白文字を出力しない。
uucp	転送されるファイル名は、? や * などのメタ文字を使用して指定できる。
uulog	終了コードの変更。エラーメッセージは、(stdout ではなく) stderr に出力される。
uupick	終了コードの変更。エラーメッセージは、(stdout ではなく) stderr に出力される。
uustat	終了コードの変更。エラーメッセージは、(stdout ではなく) stderr に出力される。
uuto	終了コードの変更。オプションデリミタに "--" が追加された。エラーメッセージは、(stdout ではなく) stderr に出力される。
uux	終了コードの変更。オプションデリミタに "--" が追加された。オプションの -j と -n は、まとめることができる。コマンドが実行できなかった場合に書き込まれる出力が変更された。オプションの順序が変更されても機能する。パス名の展開が実行される。エイリアスの置き換えは、処理されるシェルパイプラインで実行される。絶対パス名で機能する。~name が前にあるパス名でも機能する。~が使用された場合、PUBDIR 値に展開される。ファイルパスが指定されない場合、カレントディレクトリからファイルを取り出す。uux は stdin からの入力を受け付ける。uux リクエストの内部で非ローカルファイル名が一意でなければならない場合でも、uux は機能する。
what	無効なオプションに対するエラー処理を改善する。
xargs	実行できなかった場合、終了値が非ゼロとなる。
zcat	オペランドの処理中にエラーが発生すると、最後の終了値は非ゼロとなる。

動作環境

システムコマンドの Classic HP-UX 動作と UNIX95 動作の両方を許容するよう、アプリケーションが変更されるまでは、コマンドを適切に機能させることのできる簡単な動作環境によってアプリケーションを変更できます。UNIX95 コマンドを呼び出す前に UNIX95 環境をクリアすると、コマンドはその Classic 動作で実行されます。

C 言語によるプログラミング

コマンドプログラミングと同様、UNIX95 の開発環境が有効になっている場合、コード開発エンジニアは、API の使用を UNIX95 の定義に厳密に準拠 (3 ページを参照) するように制限しなければなりません。UNIX95 の開発環境で開発されたアプリケーションが、アプリケーションによって提供されるライブラリを使用する場合、そのライブラリは厳密に準拠するものでなければなりません。互換性のない Classic の関数が UNIX95 の関数で動作するように設計されたプログラムにリンクされることのないよう、厳密な準拠というこの過渡的な特性を遵守することが必要です。Classic 環境でコンパイルされたか UNIX95 環境でコンパイルされたかによって、関数の意味が異なるだけではなく、一部のデータ構造も異なる場合があります。

Classic と UNIX95 の開発環境が混在する構成はサポートされていません。UNIX95 の開発環境を部分的に有効にする構成もサポートされていません。いずれの場合も、アプリケーションが異常動作したり、異常終了する可能性があります。

Classic の開発環境でコンパイルされたリロケータブルモジュールや共有ライブラリを UNIX95 の開発環境でコンパイルされたモジュールと混ぜ合わせた場合、環境によってデータ構造が異なることがあるため、予期しえない動作を招く可能性があります。たとえば、シグナルコンテキスト構造は開発環境によって異なります。UNIX95 環境とシグナルコンテキストを使用するアプリケーションでは、コンテキスト構造を使用する関数をすべて UNIX95 環境でコンパイルする必要があります。

UNIX95 環境に準拠した開発環境の設定

開発エンジニアは、環境変数 UNIX95 と PATH を設定するだけではなく、マクロ定義 _XOPEN_SOURCE_EXTENDED をアプリケーションのソースコードに設定するか、C コンパイラに対するコンパイル時オプションとして設定することが必要です。開発エンジニアが環境変数 PATH を設定しなかった場合、リンカは、「コンテキスト」 関数への参照に対して未定義外部記号を通知します。

開発環境が UNIX 95 環境に合わせて正しく設定されると、アプリケーションがリンクされたときに、作成されたリロケータブルモジュールはシステムライブラリ内の関数と整合します。アプリケーションがリンクされると、それは UNIX 95 のアプリケーションとして識別されます。システムは、UNIX 95 として識別されるアプリケーションを、マークなし (デフォルトの Classic モード) のアプリケーションとは異なる別の扱いを行ないます。これらのアプリケーションは、たとえ UNIX 95 環境を無効にして実行された場合でも、UNIX 95 アプリケーションとして動作します。

プロセスシグナル

UNIX95 API への準拠のために、HP-UX には 2 つの新しいシグナル (SIGXCPU と SIGXFSIZE) が追加されました。前者のシグナルは、アプリケーションが CPU 秒数の限度を超過したときに、システムによって生成されます。後者のシグナルは、アプリケーションによるファイル書き込みによって、ファイルがブロック数の限度を越えて大きくなるときに、システムによって生成されます。

デフォルトでは、CPU 秒数とファイルサイズに対する限度には、それぞれ無限大が設定されます。ユーザがこれらの限度に有限の値を設定しない限り、どのアプリケーションにも影響を与えません。ユーザが CPU 秒数やファイルサイズの限度を有限の値に変更し、しかもアプリケーションがいずれかの限度を超過した場合、Classic アプリケーションでは、システムはプロセスにいず

れのシグナルも出力しません。しかし、たとえ Classic アプリケーションであっても、ユーザまたはアプリケーションがプロセスにいずれかのシグナルを出力するよう、システムに明示的に指示した場合、システムはそのシグナルを出力します。

アプリケーションは、ユーザが CPU 時間やファイルサイズを制限したいシステム上で実行できるものと想定する必要があります。そのため、シグナルをキャッチし、広範囲のシステム上で動作することを期待されるすべてのアプリケーション (Classic のアプリケーションを含む) は、これらのシグナルに対応するよう変更してください。この変更が行なわれない状態で、キャッチする必要のあるシグナルをシステムが出力した場合、アプリケーションのデータが破壊されることがあります。

SIGXFSIZ シグナル

この新しいシグナルがシステムに追加されています。アプリケーションに対して設定された限度を越えてファイルが拡張されるような書き込みをアプリケーションが試みるたびに、カーネルがこのシグナルを UNIX 95 アプリケーションに出力します。UNIX 95 アプリケーションに対するデフォルトアクションは、プロセスをキルしてコアファイルを残すことです。アプリケーションが Classic アプリケーションである場合、ファイルサイズがアプリケーションの限度を超過した場合でも、カーネルはシグナルを送信しません。しかし、アプリケーションが Classic アプリケーションにこのシグナルを明示的に送信した場合、シグナルが出力され、デフォルトアクションが実行されます。

SIGXCPU シグナル

この新しいシグナルがシステムに追加されています。アプリケーションに対して設定された CPU 時間の限度をアプリケーションが超過するたびに、カーネルがこのシグナルを UNIX95 アプリケーションに出力します。UNIX95 アプリケーションに対するデフォルトアクションは、プロセスをキルしてコアファイルを残すことです。アプリケーションが Classic アプリケーションである場合、アプリケーションの限度を超過した場合でも、カーネルはシグナルを送信しません。しかし、アプリケーションが Classic アプリケーションにこのシグナルを明示的に送信した場合、シグナルが出力され、デフォルトアクションが実行されます。

sigcontext 構造

sigcontext データ構造は、クリーンなネームスペースをもつように変更されました。アプリケーションは、UNIX95 API に記述されていない、この構造のエレメントにはアクセスしてはいけません。

シグナルパラダイムの混合

関数 bsdproc、signal、sigvector、sigblock、sigsetmask、sigspace によって提供される Classic シグナルパラダイムと UNIX 95 シグナルパラダイムを一緒に使用すると、アプリケーションが異常動作することがあります。UNIX 95 の関数 bsd_signal は、該当する場合には signal の代わりに使用できます。

nftw/sigpause/setpgrp 関数

nftw 関数

関数 nftw は、UNIX95 に準拠するよう変更されました。この変更は、nftw からアプリケーションの指定した関数に渡される 4 番目の引数で行なわれています。このパラメータは、Classic 環境では値呼びのパラメータであり、UNIX 95 環境では参照呼び (ポインタ) となります。

Classic 版の nftw を使用するアプリケーションは、UNIX95 環境でコンパイルされたリロケータブルモジュールと混在させないでください。

sigpause 関数

関数 sigpause は、UNIX95 に準拠するように変更されました。Classic 版ではシグナルマスクを引数として受け取ります。UNIX95 版ではシグナル番号を引数として受け取ります。

UNIX オペレーティングシステムが初めて導入されたとき、提供された固有のシグナルは 32 未満でした。年の経過とともに、この処理系は、各ビットで 1 つのシグナルを表わしながら、ビットマスクを介してシグナル情報をアプリケーションからシステムに渡す API を提供するようになってきました。long integer 宣言は少なくとも 32 ビット幅であることが保証されていたため、このビットマスクを実現するために使用されました。

もはや、カーネルがサポートするシグナル数はシグナルマスクによって表現できるシグナル数を超過しているため、この追加シグナルを有効にする関数を対象にして Classic 版の sigpause (または 32 ビットのシグナルマスクを使用するその他の関数) を使用しても適切に機能しません。

setpgrp 関数

setpgrp 関数は UNIX95 API に準拠するように変更されました。Classic 版の setpgrp では、セッション ID を変更しません。UNIX95 版の setpgrp では、プロセスグループリーダーがセッションリーダーを兼務しない場合、セッション ID に PID を設定します。

HP-UX の setsid のマニュアルページには、setpgrp は下位互換性だけを目的として提供されていることが数年前から明記されています。setpgrp の代わりに setsid 関数を使用するよう、すべてのアプリケーションを変更してください。setpgrp が好ましいインタフェースとはいえない理由については、関数 setpgrp がたどってきた変化の歴史を、本書付録の「Setpgrp」 で参照してください。

ライブラリファイルの混在

Classic の動作に依存するライブラリファイルは、UNIX95 の動作に依存するライブラリと混在することができません。ライブラリが setpgrp3 または nftw2 に変更された (10.0 のリリースで推奨) 場合、これらは setpgrp や nftw を使用する UNIX95 アプリケーションと一緒に使用できます。

アプリケーションが依存するソースコードの場合、サードパーティのライブラリから提供される関数が使用されているため、ソースコードを変更することができない場合もあります。この状況に対応するには、以下の解決策を利用します。これらの動作環境は厳密な準拠という原理に違反しますが、UNIX95 API に完全に準拠するようアプリケーションを変更できるまでは、うまく機能します。

- **setpgrp -**
  Classic 動作に依存するアプリケーションのソースコードが入手できず、しかもアプリケーションの他の部分は関数 setpgrp の UNIX 95 動作に依存しない場合、次の例に示すコードを含むアプリケーションをリンクするときに、アプリケーション開発者はリロケータブルモジュールを組み込むことができます。 #include <signal.h> int setpgrp(pid, pgrp) int pid; int pgrp; { return setpgrp3(pid, pgrp); } この関数を使用すれば、開発者は、変更されていないリロケータブルモジュールを Classic の setpgrp 関数にリンクできます。
- **nftw -**
  Classic 動作に依存するアプリケーションのソースコードが入手できず、しかもアプリケーションの他の部分は関数 nftw の UNIX 95 動作に依存しない場合、次の例に示すコードを含むアプリケーションをリンクするときに、アプリケーション開発者はリロケータブルモジュールを組み込むことができます。 #include <ftw.h> int nftw(filename, fn, flag, depth) char *filename; int (*fn)(); int flag; int depth; { return nftw2(filename, fn, flag, depth); } この関数を使用すれば、開発者は、変更されていないリロケータブルモジュールを一見 Classic らしい nftw 関数にリンクできます。

付録

ヒューレットパッカードは、X/Open Portability Guide (XPG)、System V Interface Definition (SVID)、Application Environment Specification (AES)など、重要な仕様を提供することによって、顧客のプログラミングニーズに対応してきました。しかし、これらの仕様のうちいくつかを組み合わせると、機能の異なるプログラミングインタフェースが同じ名前をもつという状況が生まれます。

歴史

1994 年における業界の努力の結果、X/Open は、X/Open Portability Guide リリース 4 (XPG4)を増強して、すべての UNIX システムプラットフォームで共通となる API の数をほぼ倍増しました。関連するできごととして、Novell Corporation が UNIX の商標に対する権利を X/Open に譲渡しました。このため、UNIX の商標とブランドを(独自の処理系のオペレーティングシステムではなく)統一されたオープン仕様に関連付けることによって、UNIX システムの市場を強化しました。設計上、HP-UX 10.0 はすでにこれらの API の多くを内蔵しており、統一 UNIX としての XPG4v2 仕様に近い将来準拠することができます。

XPG4 API セットの拡張を達成するために、システムベンダーのチームが、他の仕様で記述されていた既存 API のセットを統一しました。この中には、Novell Corporation からの SVID、および Open Software Foundation からの AES が含まれます。

その他の API は、1 つ以上の UNIX システムプラットフォーム上で動作する一般的なアプリケーションで使用されて実証されているように、業界に受け入れられている事実に基づいて組み込まれました。

互換性の課題

名前は同じでも微妙に異なる関数を提供するシステム上でこれらの API が内蔵する矛盾を最小限に抑えるため、あらゆる努力が払われました。不完全な世界ではよくあることですが、これによって必然的に、仕様の妥協によっては解決できないような名前の矛盾が生まれました。この状態になったとき、結果として得られる仕様に対して、いくつかの処理系を一時的に非準拠の状態におくという理解のもとに、矛盾する仕様セットから選択を行なうという決定がなされました。この選択は独断的に思えるかもしれませんが、矛盾する API のいずれかを選択することによって生じる影響については、相当な考慮が払われました。一般的なアプリケーションにおける業界での使用によって保証された、できるだけ多くの API が盛り込まれるように、あらゆる努力が払われました。

API のセットが選択された後、ヒューレットパッカード社の技術スタッフは、新しいバージョンの XPG4 に準拠する HP-UX 処理系を設計しました。インストールドベースに対する当社の方針に基づき、この新しいバージョンの HP-UX が既存の顧客やアプリケーション開発者に与える影響を最小限に抑えることに目標が設定されました。XPG4 バージョン 2 に記述されている約 1,170 件の API のうち、HP-UX ではリリース 10.0 に先行して、25 件を除くすべての API を実装しました。実装されなかった API のうち、4 件を除いて残りすべての API は、現在 HP-UX を構成する API との矛盾を生じさせることなく、リリース 10.0 で実装できます。

矛盾する 4 つの API を識別してから、開発スタッフは既存の顧客ベースに対する適切な移行計画を作成しました。この移行計画では、XPG4v2 で定義されている共通インタフェースを実現するリリース 10.10 に向けて前進すると同時に、顧客やアプリケーション開発者がソフトウェア投資を保持するために使用できる、いくつかの選択肢を提供します。

関数の変化

Setpgrp -
この関数の意味の変化がいかに微妙なものであるかをアプリケーション開発エンジニアが理解するには、次の抜粋が役立ちます。

- SVID-1('85)
  Setpgrp は、呼び出しプロセスのプロセスグループ ID に呼び出しプロセスのプロセス ID を設定し、新しいプロセスグループ ID を戻します。
- SVID-2('86)
  関数 setpgrp は、呼び出しプロセスのプロセスグループ ID に呼び出しプロセスのプロセス ID を設定し、新しいプロセスグループ ID を戻します。
- XPG/2('87)
  Setpgrp は、呼び出しプロセスのプロセスグループ ID に呼び出しプロセスのプロセス ID を設定し、新しいプロセスグループ ID を戻します。
  プロセスがすでにプロセスグループリーダーである場合を除いて、setpgrp はターミナルグループからプロセスを分離します(もしあれば)。

- SVID-3('89)
  呼び出しプロセスがまだセッションリーダーではない場合、関数 setpgrp は、呼び出しプロセスのプロセスグループ ID とセッション ID に呼び出しプロセスのプロセス ID を設定し、呼び出しプロセスの制御ターミナルを解放します。
- XPG/3('89)
  取り消し(setsid で代替)
- HP-UX
  呼び出しプロセスがプロセスグループリーダーではない場合、setsid や setprgp は新しいセッションを作成します。呼び出しプロセスは、この新しいセッションのセッションリーダーとなり、新しいプロセスグループのプロセスグループリーダーとなり、制御ターミナルをもちません。呼び出しプロセスのプロセスグループ ID は、呼び出しプロセスのプロセス ID に等しく設定されます。この呼び出しプロセスは、新しいプロセスグループにおける唯一のプロセスであり、新しいセッションにおける唯一のプロセスです。
  setprgp は、呼び出しプロセスのプロセスグループ ID の値を戻します。
- XPG/4v2('94)
  呼び出しプロセスがまだセッションリーダーではない場合、setpgrp は、呼び出しプロセスのプロセスグループ ID に呼び出しプロセスのプロセス ID を設定します。setpgrp が新しいセッションを作成した場合、新しいセッションには制御ターミナルがありません。
  呼び出しプロセスがセッションリーダーである場合、setpgrp 関数には何の効果もありません。

まとめ

ヒューレットパッカードは、オペレーティングシステムの標準化に鋭意努力しています。これらの標準の立案に際して他の組織と協力することにより、HP-UX 上ですでに動作しているアプリケーションに重大な影響を与えることなく標準化を実現することができます。HP-UX 上で動作しているアプリケーションに影響を与えることなくこれらの変更を行なうことができない場合、将来のリリースに向けて簡単に移行が行なえるよう、本書のような White Paper を発行いたします。

HP-UX
www.hpe.com/jp/hpux

Hewlett Packard Enterprise