System i
バージョン 7.2

プログラミング
Qshell

System i
バージョン 7.2

プログラミング
Qshell

ご注意

本書および本書で紹介する製品をご使用になる前に、 221 ページの『特記事項』に記載されている情報をお読みください。

本製品およびオプションに付属の電源コードは、他の電気機器で使用しないでください。

本書にはライセンス内部コードについての参照が含まれている場合があります。ライセンス内部コードは機械コードであり、IBM 機械コードのご使用条件に基づいて使用権を許諾するものです。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： System i
Version 7.2
Programming
Qshell

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2014.4

目次

Qshell

Qshell は POSIX と X/Open 標準を基にしたコマンド環境です。

以下の 2 つの部分で構成されます。

- シェル・インタープリター (すなわち **qsh**)。これは、入力ソースからコマンドを読み取って、各コマンドを解釈してから、オペレーティング・システムのサービスを使ってコマンドを実行します。
- ユーティリティー (すなわちコマンド)。これは、別の機能を提供する外部プログラムですが、きわめて単純化することも非常に複雑化することもできます。

シェル・インタープリターとユーティリティーをまとめると、強力な標準ベースのスクリプト記述環境が成立します。 i5/OS™ に備わっている新規のプログラミング・モデルを使用して Qshell は、以下を行うための拡張可能なコマンド環境を実現します。

- 統合ファイル・システムでサポートされている任意のファイル・システムのファイルを管理する。
- 対話式セッションに対してスレッド・セーフ入出力処理を実行するスレッド化プログラムを実行する。
- クロスプラットフォーム・コマンド言語を使って変更なしにそのまま他のシステムで実行できるシェル・スクリプトを書く。
- Qshell に備わった機能を拡張するための自分独自のユーティリティーを作成する。

このピックでは、新規ユーザーと経験を積んだユーザーの両方に、 Qshell コマンドの使用および Qshell スクリプトの作成に必要な情報を提供しています。

注: この資料では、「ジョブ」と「プロセス」という用語は同じ意味で使われています。「ジョブ」は i5/OS で使われている用語であり、「プロセス」は POSIX に属する用語です。

Qshell の PDF ファイル

この情報の PDF ファイルを表示または印刷できます。

この資料に関する PDF 版を表示またはダウンロードするには、Qshell を選択します。

以下の関連トピック PDF を表示またはダウンロードできます。

- IBM® Developer Kit for Java™
- IBM Toolbox for Java

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ご使用のブラウザーで PDF リンクを右クリックする。
2. PDF をローカルに保存するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「**保存**」をクリックする。

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、システムに Adobe Reader をインストールしている必要があります。無料のコピーを、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html)  からダウンロードできます。

チュートリアル

Qshell コマンド言語およびユーティリティーの使用について学習するには、このリンクを選択します。初めてシェルおよびシェル・コマンドを使用する場合には、ここから始めてください。

このトピックは、Qshell コマンド言語およびユーティリティーの使用を開始する上で助けとなるチュートリアルです。

Qshell コマンド言語の機能

コマンド、入出力リダイレクト、パス名展開、パラメーター展開、およびコマンド置換について学習します。

シェル・インタープリターは、対話式セッションまたはシェル・スクリプトの記述のどちらかの目的で使用することができます。シェル・スクリプトは、シェル・コマンドの入ったテキスト・ファイルに過ぎません。 Qshell コマンド言語は、対話式に使用するときでも、スクリプトを記述するときでも同一です。対話式コマンド行から実行するいずれのコマンドでも、シェル・スクリプトに入れて同じように実行することができます。 Qshell コマンド言語はインタープリター型なので、シェル・スクリプトも実行されるたびに読み取られて、解釈されます。

コマンド

単純コマンドとは、実行したいユーティリティーの名前のことです。コマンドの完全修飾パス名を指定した場合 ("/usr/bin/ls" など)、**qsh** はそのコマンドを実行します。コマンドの相対パス名を指定した場合 ("ls" など)、**qsh** は PATH 変数により指定されたディレクトリーを検索して、そのコマンドを探します。PATH 変数はコロンで区切られたディレクトリーのリストであり、**qsh** にコマンドの検索場所を指し示します。 PATH 変数が次のように設定されているとします。

```
/usr/bin:.:/QOpenSys/usr/bin
```

このとき、**qsh** は最初に "/usr/bin" ディレクトリーでコマンドを探し、それから現行作業ディレクトリー、そして最後に "/QOpenSys/usr/bin" ディレクトリーを探します。PATH 変数の先頭または末尾がコロンの場合、隣接する 2 個のコロンが含まれている場合、またはディレクトリーとしてドット (.) を指定する場合、qsh は現行作業ディレクトリーを検索します。

デフォルトでは、**qsh** は、そのコマンドが完了するまで待ってから次のコマンドを実行します。コマンドが完了すると、その結果を示す終了状況が設定されます。終了状況がゼロの場合、コマンドは正常に終了したことを意味します。終了状況がゼロより大きい場合は、コマンドは不成功だったことを意味します。通常、コマンドに障害が起こると、終了状況は 1 になります。しかし、コマンドが検出されても実行できなかった場合、**qsh** は終了状況を 126 に設定し、コマンドが検出されなかった場合は終了状況を 127 に設定します。

複合コマンドには、if-then-else 条件文、 [[...]] 条件文、case 条件文、 select 条件文、while ループ、until ループ、for ループ、および関数が含まれます。これらのコマンドは、高水準プログラム言語に期待される機能を提供するとともに、複雑なシェル・スクリプトを記述することを可能にします。

パイプラインを使用すると、複数のコマンドを連結して、あるコマンドの出力を次のコマンドの入力データとすることができます。たとえば、次のようなパイプラインがあるとします。

```
ls | grep ^apple
```

この場合、ls ユーティリティーからの出力は、 grep ユーティリティーの入力データとなります。 **ls** ユーティリティーは、ディレクトリーの内容をリストし、 **grep** ユーティリティーは、パターンの一致を検索します。上述のパイプラインの最終出力は、現行ディレクトリー内にある、名前が "apple" で始まるファイルのリストとなります。

3 つ以上のコマンドをパイプラインで連結することも可能です。これは **qsh** の非常に強力な機能であり、これにより複雑なタスクを実行するために、幾つものコマンドを結合させることができます。

パイプラインに似た、別のタイプのリストもあります。 "and" リストは、リストの最初のコマンドの終了状況がゼロ以外の場合、停止します。 "or" リストは、リストの最初のコマンドの終了状況がゼロの場合、停止します。

非同期リストは、コマンドをバックグラウンドで実行します。たとえば、次のようなコマンドがあるとします。

```
mypgm &
```

この場合、mypgm を開始し、その後 mypgm が完了する前に、別のコマンドを実行することができます。実行に長時間を要するコマンドがある場合、非同期リストを使用すれば、コマンドを開始した後、そのコマンドが完了するまで待たないで済みます。

入出力リダイレクト

入出力リダイレクトを使用すると、コマンドの入力先と出力先を変更することができます。 Qshell コマンドでは、入出力は記述子に対して行われます。記述子は、統合ファイル・システム内のオブジェクトか、TCP/IP ソケットのいずれかに対してオープンすることができます。入力データは記述子 0 (標準入力) から入力され、通常の出力は記述子 1 (標準出力) から出力され、さらにエラー出力は記述子 2 (標準エラー) から出力されます。

入力先は標準入力をリダイレクトすることにより変更することができます。たとえば次のようなコマンドがあるとします。

```
grep orange <fruits.list
```

この場合、**grep** ユーティリティーは、標準入力から読み取りを行うとき、fruits.list というファイルの内容を受け取ります。

出力先は標準出力をリダイレクトすることにより変更することができます。たとえば次のようなコマンドがあるとします。

```
grep apple fruits.list >apple.list
```

この場合、**grep** ユーティリティーが標準出力に結果を書き込むとき、結果は apple.list というファイルに書き込まれます。

標準出力と標準エラーを同じファイルに送ることもできます。たとえば次のようなコマンドがあるとします。

```
grep apple fruits.list >apple.list 2>&1
```

この場合、標準出力 (記述子 1) は、ファイル apple.list に書き込まれ、標準エラー (記述子 2) は記述子 1 と同じ場所にリダイレクトされます。

たいていの場合、リダイレクトは標準入力、標準出力、および標準エラーを制御するためだけに使用されますが、リダイレクトを使用して記述子を 0 から 9 まで制御することもできます。

パス名展開

パス名展開は、 パターンに一致するすべてのファイルのパターンを置換します。シェル・パターンは次の記号を使用します。

- * はあらゆる文字ストリングに一致します。たとえば次のようなコマンドがあるとします。

  ```
  ls *.java
  ```

  この場合、**qsh** は *.java を、現行作業ディレクトリー内の、ファイル名が .java で終わるすべてのファイルに展開します。
- ? を使用すると、単一のあらゆる文字に一致します。たとえば次のようなコマンドがあるとします。

  ```
  ls *.?
  ```

  この場合、**qsh** は *.? を、拡張子が単一文字のすべてのファイルに展開します。
- [ ] は文字クラスのために使用します。 **qsh** は、文字クラスを使って、文字のセットまたは範囲の突き合わせを行います。たとえば次のようなコマンドがあるとします。

  ```
  ls *.[ch]
  ```

  この場合、**qsh** は *.[ch] を、現行作業ディレクトリー内の .c または .h のいずれかで終わるすべてのファイルに展開します。文字の範囲を指定することもできます。たとえば次のようなコマンドがあるとします。

  ```
  ls *.jav[a-c]
  ```

  この場合、**qsh** は *.jav[a-c] を、.java、.javb、または .javc で終わるすべてのファイルに展開します。

パラメーター展開

パラメーター展開は、変数の値を置換します。最も簡単な次のような形式のものがあるとします。

```
$myvar
```

この場合、**qsh** は、myvar という変数の値を置換します。

デフォルトまたは代替値を使用するための修飾子、または変数が設定されていないかヌルである場合にエラーを示すための修飾子があります。たとえば次のようなパラメーター展開があるとします。

```
${counter:=0}
```

この場合、変数が設定されていないかヌルであるなら、**qsh** はその変数 counter のデフォルト値をゼロに設定します。変数 counter がすでに設定されていた場合には、その値は変更されず、現行値が置換されます。

大小のプレフィックス・パターンまたはサフィックス・パターンを除去するための修飾子もあります。パターンはパス名展開のために使用されるパターンと同一です。 4 つのパターン修飾子があります。

- % 修飾子は最小サフィックス・パターンを除去するという意味です。
- %% 修飾子は最大サフィックス・パターンを除去するという意味です。
- # 修飾子は最小プレフィックス・パターンを除去するという意味です。

- ## 修飾子は最大プレフィックス・パターンを除去するという意味です。

たとえば、変数 pathname が "/fruits/apples/grannysmith" に設定されている場合に、次のようなパラメーター展開があるとします。

```
${pathname%/*}
```

この場合、**qsh** は "/*" と一致する最小右方パターンを除去して、"/fruits/apples" に置き換えます。

コマンド置換

コマンド置換を使用すると、コマンド名をコマンドの出力で置き換えることができます。たとえば次のようなコマンド置換があるとします。

```
$(grep apple fruit.list)
```

この場合、**qsh** は **grep** コマンドの出力を置換します。このようにすると、スクリプト内でさらに処理を行うためにコマンド出力を簡単に取り込むことができます。

逆引用符 (`) を使用する旧式のコマンド置換はサポートされていますが、その引用規則があいまいなため、使用しないようにしてください。

Qshell ユーティリティー機能

Qshell には、さまざまな機能を実行する、100 を上回るユーティリティーが用意されています。

Qshell には、さまざまな機能を実行する、100 を上回るユーティリティーが用意されています。ユーティリティーの種類は次の 2 つです。

- 組み込みユーティリティー。qsh が検索を行わずに直接実行できるユーティリティーです。シェル・インタープリターと同じプロセスで実行されます。
- 正規のユーティリティー。別個のプログラム・オブジェクトであり、qsh はこれを検索しなければなりません。シェル・インタープリターが開始する、新規のプロセスで実行されます。

Qshell ユーティリティーのフォーマット設定は次のとおりです。大括弧はオプションで指定されるものを表します。

```
utility [ options ] [ parameters ]
```

あるユーティリティーでは、負符号 (-) を前に付けた単一文字の options を使用します。たとえば、いくつかのユーティリティーは、ディレクトリー・ツリー上で再帰的作業をするときに -r オプションを使用します。複数のオプションを指定できます。オプションはすべてパラメーターよりも先に指定しなければなりません。パラメーターが負符号 (-) で始まる場合、オプションの末尾を表すために -- オプションを使用できます。たとえば次のようなコマンド行があるとします。

```
utility -r -- -1
```

この場合、-- がオプションの終了を示すため、この -1 はパラメーターとして扱われます。

統合ファイル・システムのナビゲート

統合ファイル・システムをナビゲートするときには、必ず現行作業ディレクトリーがあります。先頭にスラッシュ (/) を付けずにファイルまたはディレクトリーが指定された場合、それは現行作業ディレクトリーにあるものと見なされます。

現行作業ディレクトリーは、cd ユーティリティーを使用して変更できます。たとえば、現行作業ディレクトリーを /usr/bin に変更する場合には、次のコマンドを使用します。

```
cd /usr/bin
```

現行作業ディレクトリーを表示するときには、 pwd ユーティリティーまたは pwdx ユーティリティーを使用できます。 **pwdx** ユーティリティーは、シンボリック・リンクを解決し、絶対現行作業ディレクトリーを表示します。

ディレクトリーの内容をリスト表示するときは、ls ユーティリティーを使用します。パラメーターを付けずに **ls** を実行すると、現行作業ディレクトリーの内容がリスト表示されます。また、パラメーターとして複数のディレクトリーを指定することもできます。 -l (小文字のエル) オプションを使用すると、**ls** は、ディレクトリー内の各オブジェクトに関する詳細情報をリスト表示します。それには、オブジェクトの許可、オブジェクトの所有者およびグループ、オブジェクトのサイズ、およびオブジェクトが最後に使用された日付が含まれます。

ファイルおよびディレクトリーを処理する

新規のディレクトリーを作成するには、mkdir ユーティリティーを使用します。 -p オプションを指定すると、**mkdir** はパスに含まれるすべてのディレクトリーを作成します。たとえば、新しく "/fruits" と "/fruits/pears" のディレクトリーを作成する場合は、次のコマンドを使用します。

```
mkdir -p /fruits/pears
```

cp ユーティリティーを使用すると、ファイルをコピーすることができます。たとえば、ファイル "/fruits/apples/delicious" を、ファイル "/fruits/apples/grannysmith" にコピーする場合は、次のコマンドを使用します。

```
cp /fruits/apples/delicious /fruits/apples/grannysmith
```

リネームまたはオブジェクトの移動は、mv ユーティリティーを使用して行うことができます。たとえば、現行ディレクトリーにあるファイル orange を、"/fruits" ディレクトリーにあるファイル "tangerine" に移動する場合は、次のコマンドを使用します。

```
mv orange /fruits/tangerine
```

オブジェクトを削除する場合は rm ユーティリティーを使用し、ディレクトリーを削除する場合は rmdir ユーティリティーを使用します。 -r オプションを指定すると、**rm** はディレクトリー・ツリー内のすべてのオブジェクトを再帰的に削除します。このようにすると、多数のオブジェクトを 1 つのコマンドで簡単に削除することができます。たとえば、"/fruits" というディレクトリー・ツリー内のすべてのファイルおよびディレクトリーを削除する場合には、次のコマンドを使用します。

```
rm -r /fruits
```

すべてを組み合わせてスクリプトを作成する

シェル・スクリプトの記述方法を示す例を表示します。

以下の例は、シェル・インタープリターおよびシェル・ユーティリティー機能の実例を紹介する単純なシェル・スクリプトです。このスクリプトは入力パラメーターが 1 つあります。このパラメーターはディレクトリー名です。スクリプトはその後、拡張子が .java のすべてのファイルを入力ディレクトリーから現行ディレクトリーにコピーし、コピーしたファイルの数を数えます。

```
1 # Get a list of files
2 filelist=$(ls ${1}/*.java)
3 count=0
4 # Process each file
```

```
 5 for file in $filelist ; do
 6   # Strip directory name
 7   target=${file##*/}
 8   # Copy file to current directory
 9   cp $file $target
10   count=$((count+=1))
11   # Print message
12   print Copied $file to $target
13 done
14 print Copied $count files
```

1、4、6、8、11 の各行の # 文字は、コメントを表しています。 # 文字の後にあるいずれの文字も qsh は解釈しません。

2 行目で、変数 filelist が ls コマンドからの出力に設定されます。 ${1} は最初の入力パラメーターに展開され、*.java は拡張子が .java のすべてのファイルに展開されます。

3 行目では、変数 count がゼロに設定されます。

5 行目にあるのは for ループです。ループの反復ごとに変数 file が変数 filelist 内の次の要素に設定されます。各エレメントはフィールド区切り文字により区切られます。デフォルトのフィールド区切り文字は、タブ、スペース、および改行です。セミコロン文字はコマンド区切り文字であり、これを使用すると、1 行に複数のコマンドを入れることができます。

7 行目で、変数 target が完全修飾パス名からのファイル名に設定されます。 ${file##*/} パラメーター展開は、左端から最後のスラッシュ文字まで、パターンに最長一致する文字を除去します。

9 行目で、cp ユーティリティーにより、ファイルが指定されたディレクトリーから現行作業ディレクトリーにコピーされます。

10 行目では、変数 count が 1 加算されます。

12 行目では、print ユーティリティーを使用して、コピーされたファイルを示すメッセージが出力されます。

13 行目では、done が for ループの終了を示します。

14 行目では、コピーされたファイルの合計数を示すメッセージが出力されます。

ディレクトリー /project/src に .java 拡張子の付いた 2 つのファイルが含まれているとします。ここで、次のコマンドを使用してこのスクリプトを呼び出します。

```
javacopy /project/src
```

その場合、スクリプトからの出力は次のようになります。

```
Copied /project/src/foo.java to foo.java
Copied /project/src/bar.java to bar.java
Copied 2 files
```

コマンド言語

シェル・スクリプトを書き込んでいるか、経験を積んだシェル・ユーザーである場合には、この詳細な参照情報から始めると良いでしょう。

qsh は次のタスクを行うプログラムです。

- ファイルまたは端末から入力を読み取る。

- 入力をトークンに分割する。
- 入力を解析して、単純コマンドおよび複合コマンドにする。
- 各コマンドについて各種の展開を行う。
- 入力と出力のリダイレクトを行う。
- コマンドを実行する。
- 必要に応じて、コマンドが完了するまで待つ。

qsh は、フロー制御構成体、変数、および関数を持つコマンド言語を実装しています。このインタープリター式言語は、対話式使用と非対話式 (シェル・スクリプト) のどちらでも使用できます。したがって、対話式コマンド行でコマンドを入力することも、ファイルに格納したコマンドを、**qsh** に直接実行させることもできます。

コマンドについて詳しくは、AIX® Information Center を参照してください。

関連タスク:

53 ページの『qsh - Qshell コマンド言語インタープリター』

引用符

引用符は、特定の文字が **qsh** に対して持つ意味を取り消すのに使用します。

以下の文字を使用することができます。

- エスケープ文字 (円記号 (¥)) は、その次の文字 (<newline> を除く) の特別な意味を取り消します。円記号のあとに <newline> が続いている場合は、 **qsh** はそれを行の継続と解釈します。たとえば、 **¥$** は、ドル記号が持つ特別な意味を取り消します。
- リテラル (単一) 引用符 (**'...'**) は、単一引用符を除くすべての文字の特別な意味を取り消します。
- グループ化 (二重) 引用符 (**"..."**) は、ドル記号 (**$**)、逆引用符 (**`**)、および円記号 (**¥**) を除くすべての文字の特別な意味を取り消します。円記号は、その後に続く文字がドル記号 (**$**)、逆引用符 (**`**)、二重引用符 (**"**)、円記号 (**¥**)、または <newline> である場合に限り、エスケープ文字としての特別な意味を保持します。

パラメーター

パラメーターは、データを格納するために使用されます。

特定のパラメーターの値を知りたいときは、そのパラメーターの名前の前にドル記号 (**$**) を付け、名前を中括弧 (**{ }**) で囲みます。名前が 1 桁の数字、特殊パラメーター、または単一の ID である場合は、中括弧を使っても使わなくても構いません。

定位置パラメーター

定位置パラメーターは、1 から始まる 10 進数です。最初は、**qsh** は、定位置パラメーターをシェル・スクリプトの名前の後に続くコマンド行引数に設定します。定位置パラメーターは、シェル関数が呼び出されると一時的に置き換えられます。また、定位置パラメーターは、set および shift ユーティリティーを使って再割り当てすることができます。

特殊パラメーター

特殊パラメーターは、次に示す特殊文字の 1 つで表します。

*** (定位置パラメーター)**

(アスタリスク) 1 から順に、定位置パラメーターに展開されます。引用符で囲まれたストリングの中で展開される場合は、展開後のストリングは単一のフィールドになり、各パラメーターの値は、**IFS** 変数の最初の文字 (**IFS** が未設定の場合は<スペース>) で区切られます。

@ (定位置パラメーター)

(アットマーク) 1 から順に、定位置パラメーターに展開されます。引用符の中で展開される場合は、各定位置パラメーターはそれぞれ単独の引数として展開されます。定位置パラメーターがない場合は、@ が引用符で囲まれていても、 @ の展開では引数は生成されません。

 # (定位置パラメーターの数)

(番号記号) 定位置パラメーターの数を示す 10 進数に展開されます。この値は、最初は、**qsh** が呼び出されたときの引数の数に設定されます。この値は、set、shift、またはdot ユーティリティーを使うか、または関数を呼び出すことによって変更できます。

? (終了状況)

(疑問符) 最後に実行されたコマンドの戻り状況を示す 10 進数に展開されます。値 0 は正常終了を示します。 0 以外の値はエラーを示します。シグナル番号で終わるコマンドの戻り状況は、128 にシグナル番号を加えた値です。

- (オプション・フラグ)

(負符号) **qsh** が呼び出されたときに、set によって指定されているか、または **qsh** で暗黙指定されている現行のオプション・フラグ (1 文字のオプション名を連結して 1 つのストリングにしたもの) に展開されます。

$ (現行シェルのプロセス ID)

(ドル記号) 現行シェルのプロセス ID を示す 10 進数に展開されます。サブシェルが別のプロセスの中で実行されている場合でも、そのサブシェルは現行シェルと同じ **$** の値を維持します。

! (バックグラウンド・プロセス ID)

(感嘆符) 現行シェルから最後に実行されたバックグラウンド・コマンドのプロセス ID を示す 10 進数に展開されます。パイプラインの場合は、これはそのパイプライン内の最後のコマンドのプロセス ID です。

0 (シェル・スクリプトの名前)

(ゼロ) シェルまたはシェル・スクリプトの名前に展開されます。

関連概念:

15 ページの『パラメーター展開』
qsh がどのようにパラメーターを展開するかについての情報を表示するには、このリンクを選択します。

変数

qsh は、開始されると、定義済みの環境変数に基づいてシェル変数を初期設定します。変数は、データを格納するために使用されます。 .

次の方法のいずれかを使って、既存の変数の値を変更すること、または新しい変数を作成することができます。

- name=value を使って変数を割り当てる。
- read または getopts ユーティリティーを呼び出す。
- **for** ループまたは **select** 条件構成の中で、*name* パラメーターを使用する。
- ${name=value} のパラメーター展開を使用する。
- declare または typeset ユーティリティーを呼び出す。

変数名には、英字、数字、下線 (_) を使用できます。 変数名の最初の 1 文字には数字は使用できません。

qsh が設定する変数

_ (一時変数)
この変数は、**qsh** によって直前の単純コマンドの最後の引数に設定されます。

EGID (実効 1 次グループ ID)
この変数は、**qsh** の開始時に、**qsh** によって実効 1 次グループ ID に設定されます。この変数は読み取り専用です。

EUID (実効ユーザー ID)
この変数は、**qsh** の開始時に、**qsh** によって実効 ユーザー ID に設定されます。この変数は読み取り専用です。

GID (1 次グループ ID)
この変数は、**qsh** の開始時に、**qsh** によって 1 次グループ ID に設定されます。この変数は読み取り専用です。

HOSTID (ホスト IP の ID)
この変数は、**qsh** によってホスト・システムの IP アドレスに設定されます。

HOSTNAME (ホストの名前)
この変数は、**qsh** によってホスト・システム名に設定されます。

HOSTTYPE (ホスト・タイプ)
この変数は、**qsh** によってホスト・システムのタイプを表すストリングに設定されます。この変数の値は、"powerpc" に設定されます。

JOBNAME (修飾ジョブ名)
この変数は、**qsh** によって現行ジョブの修飾ジョブ名に設定されます。 修飾ジョブ名は、 CL コマンドによってジョブを識別するために使用されます。

LAST_JOBNAME (最後のジョブの修飾ジョブ名)
この変数は、**qsh** によって最後のジョブの修飾ジョブ名に設定されます。修飾ジョブ名は、 CL コマンドによってジョブを識別するために使用されます。

LINENO (行番号)
この変数は、各コマンドの実行前に、**qsh** によってスクリプトまたは関数内の現在行の番号 (10 進数) に設定されます。

MACHTYPE (マシン・タイプ)
この変数は、**qsh** によって、マシン・タイプを表すストリングに設定されます。この変数の値は "powerpc-ibm-os400" に設定されます。

OLDPWD (直前の作業ディレクトリー)
この変数は、現行作業ディレクトリーが変更されたあとで、 cd によって前の作業ディレクトリーに設定されます。

OPTARG (オプション引数)
この変数は、引数を必要とするオプションが見つかったときに、getopts によって設定されます。

OPTIND (オプション索引)
この変数は、getopts によって、次のオプション用として調べる必要がある引数の索引に設定されます。 **qsh**、スクリプト、または関数が呼び出された時点で、この変数は 1 に設定されます。

OSTYPE (オペレーティング・システムのタイプ)
この変数は、**qsh** によって、オペレーティング・システムのタイプを表すストリングに設定されます。この変数の値は、"os400" に設定されます。

PPID (親プロセス ID)
この変数は、**qsh** によって、現行シェルを呼び出したプロセスの 10 進プロセス ID に設定されます。 1 つのサブシェルの中では、そのサブシェルが別のプロセス内で実行されている場合でも、この変数の値は変更されません。

PWD (作業ディレクトリー)
この変数は、変更後、cd によって現行作業ディレクトリーに設定されます。

QSH_VERSION (現行バージョン)
この変数は、**qsh** によって、現行バージョンを表すストリングに設定されます。このストリングの形式は VxRyMz です。x はバージョン番号、y はリリース番号、z はモディフィケーション・レベル番号です。この変数は読み取り専用です。

RANDOM (乱数発生ルーチン)
この変数は、参照されるたびに **qsh** によって 1 - 32767 の範囲内の整数の乱数に設定されます。この変数を設定すれば、乱数発生ルーチンにシードを与えることができます。

REPLY (応答変数)
この変数は、引数を指定しなかった場合に読み取られる文字に設定されます (read による設定)。また select 複合コマンドを使用すると、標準入力から読み取られる入力行の内容に設定されます。

TERMINAL_TYPE (端末のタイプ)
この変数は、**qsh** によって、標準ファイル記述子に接続される端末のタイプに設定されます。 この変数の値は、5250 ディスプレイに接続されるときは "5250"、リモート・クライアントに接続されるときは "REMOTE"、そしてパイプに接続されるときは "PIPELINE" に設定されます。

UID (ユーザー ID)
この変数は、**qsh** の開始時に、**qsh** によってユーザー ID に設定されます。この変数は読み取り専用です。

qsh が使用する変数

CDPATH (cd のための検索パス)
cd 用として指定したディレクトリーがスラッシュ (/) で始まっていない場合は、**qsh** は、**CDPATH** 内に表示されているディレクトリーを順番に検索して、指定されたディレクトリーを見つけます。 この変数の値は、コロンで区切ったディレクトリーのリストです。 現行作業ディレクトリーは、 最初のコロンの前、2 つのコロンの間、または最後のコロンのあとのピリオド (.) またはヌル・ディレクトリーで指定されます。 デフォルト値はありません。

ENV (環境ファイル)
qsh は、呼び出された時点で、この変数に対してパラメーター展開、コマンド置換、および算術展開を行って、現行の環境内で実行するシェル・スクリプトのパス名を生成します。この変数は、一般に、別名を設定するため、関数を定義するため、またはオプションを設定するために使用されます。 デフォルト値はありません。

HOME (ホーム・ディレクトリー)
この変数の値は、ホーム・ディレクトリーのパス名です。この値は、ティルド展開に使用され、また cd 用のデフォルトの引数として使用されます。デフォルトでは、この値はユーザー・プロファイル内に指定されている値に設定されています。

IFS (内部フィールド区切り記号)
この値は、フィールドの分割のため、および read で行をフィールドに分割するために使用される文字のリストとして扱われるストリングです。 値の最初の文字は、特殊パラメーター * を展開するときに、引数を区切るために使用されます。 デフォルト値は "<space><tab><newline>" です。

LANG (言語ロケール)
この変数は、**LC_** で始まる変数を使って明示的に設定されていないカテゴリー用に使用されるロケール・カテゴリーを定義します。デフォルト値はありません。

LC_ALL (ロケール設定)
この変数は、**LC_** で始まる変数の値を一時変更します。 デフォルト値はありません。

LC_COLLATE (ロケール照合)
この変数は、文字間の照合関係を定義します。 デフォルト値はありません。

LC_CTYPE (ロケール文字クラス)
この変数は、文字のタイプ (大文字、小文字、スペース、数字、句読記号など) を定義します。 デフォルト値はありません。

LC_MESSAGES (ロケール・メッセージ形式設定)
この変数は、アプリケーションからの肯定応答および否定応答の形式と値を定義します。 デフォルト値はありません。

LC_MONETARY (ロケール通貨形式設定)
この変数は、通貨名、記号、およびその他の詳細事項を定義します。 デフォルト値はありません。

LC_NUMERIC (ロケール数値形式設定)
定様式入出力関数およびストリング変換関数で使用する小数点文字を定義します。 デフォルト値はありません。

LC_TIME (ロケール時刻形式設定)
この変数は、使用するカレンダー、時間帯、曜日など、日付と時刻の規則を定義します。 デフォルト値はありません。

LC_TOD (ロケール時間帯)
この変数は、時間帯の名前、時間帯の時差、および、サマー・タイムの開始日と終了日を定義します。 デフォルト値はありません。

NLSPATH (メッセージ・カタログの検索パス)
メッセージ・カタログをオープンするときに、システムは、そのカタログが見つかるまで、表示されているディレクトリーを指定された順序で検索します。 この変数の値は、コロンで区切ったディレクトリーのリストです。 デフォルト値はありません。

PATH (コマンドの検索パス)
指定したコマンドがスラッシュ (/) で始まっていない場合は、 **qsh** は、実行するコマンドが見つかるまで、表示されているディレクトリーを指定された順序で検索します。 この変数の値は、コロンで区切ったディレクトリーのリストです。 現行作業ディレクトリーは、 最初のコロンの前、2 つのコロンの間、または最後のコロンのあとのピリオド (.) またはヌル・ディレクトリーで指定されます。 デフォルト値は "/usr/bin:.:/QOpenSys/usr/bin" です。

PS1 (1 次プロンプト・ストリング)
対話式オプションが設定されている場合、**qsh** は、この変数についてパラメーター展開、コマンド置換、および算術展開を行い、**qsh** がコマンドを読み取る準備ができた時点でこの変数を stderr に表示します。デフォルト値は "$" です。

PS2 (2 次プロンプト・ストリング)
コマンドが完了する前に <newline> を入力すると、 **qsh** は stderr にこの変数の値を表示します。デフォルト値は ">" です。

PS3 (select コマンド・プロンプト)
select 複合コマンドを実行する場合、**qsh** は、この変数についてパラメーター展開、コマンド置換、および算術展開を行い、 **select** によって表示される選択項目の 1 つをユーザーが選択するように促すため、この変数を stderr に表示します。デフォルト値は "#?" です。.

PS4 (デバッグ・プロンプト・ストリング)
実行トレース・オプションが設定され、対話式オプションが設定されると、**qsh** は、この変数についてパラメーター展開、コマンド置換、および算術展開を行い、実行トレース内の各行の前に、この変数を stderr 表示します。デフォルト値は "+" です。

QIBM_CCSID (変換用 CCSID)
この変数を数値に設定した場合は、**qsh**、および種々のユーティリティーは、ファイルを作成したり、ジョブの CCSID からデータを変換したりするときに、その値を使用します。デフォルト値は "0" です。これは、デフォルト・ジョブの CCSID を表します。値 "65535" は、変換を行わないことを意味します。

QIBM_CHILD_JOB_SNDINQMSG (子プロセスの開始時点での照会メッセージの送信)
この変数が正の数値に設定されているときは、子プロセスの修飾ジョブ名の付いた照会メッセージが、親プロセスに送られます。 子プロセスは、コマンド発行者がメッセージに応答するまで保留にされます。 この変数を設定することによって、プログラムの実行前にブレークポイントを設定して、子プロセス内で実行されているプログラムをデバッグすることができます。 この変数の値は、デバッグする子孫プロセスのレベルです。 この値が 1 に設定されると子プロセスは保留にされ、2 に設定されると子プロセスと孫プロセスが保留にされます (以下同様です)。デフォルト値はありません。

QIBM_MULTI_THREADED (マルチスレッド機能付きプロセスの開始)
この変数は、**qsh** が開始するプロセスが複数のスレッドを作成できるかどうかを決定します。 この変数の値が "Y" であれば、**qsh** が開始するすべての子プロセスがスレッドを開始できます。デフォルト値は "N" です。

QSH_REDIRECTION_TEXTDATA (ファイルのリダイレクト用のテキストとしてのデータの処理)
この変数は、リダイレクトに指定されたファイルから読み取るデータ、またはそのファイルに書き込むデータを、テキスト・データとして取り扱うか 2 進データとして取り扱うかを決定します。この変数の値が "Y" の場合は、**qsh** は、ファイルから読み取るデータまたはファイルに書き込むデータをテキスト・データとして取り扱います。この変数の値が "Y" でない場合は、**qsh** は、ファイルから読み取るデータまたはファイルに書き込むデータを、2 進データとして取り扱います。デフォルト値は "Y" です。

QSH_USE_PRESTART_JOBS (使用可能な場合の事前開始ジョブの使用)
この変数は、事前開始ジョブが使用可能な場合に、 **qsh** が開始するプロセスで事前開始ジョブを使用するかどうかを決定します。 この変数の値が "Y" であれば、現行サブシステム内で事前開始ジョブが使用可能である場合に、**qsh** は事前開始ジョブを使用します。この変数の値が "Y" でないか、または事前開始ジョブが使用可能でない場合は、 **qsh** が開始するプロセスはバッチ即時ジョブです。デフォルト値は "Y" です。

SHELL (シェルのパス名)
最初の行に "#!" が指定されていないスクリプト・ファイルを実行するとき、**qsh** は、この変数の値をシェル・インタープリターのパス名として使用して、スクリプトを実行します。デフォルト値はありません。

TRACEFILE (トレース・ファイルのパス名)
トレース・オプションの設定時に **qsh** は、トレース情報を保管するファイルのパス名としてこの変数の値を使用します。デフォルト値は "$HOME/qsh_trace" です。

TRACEOPT (トレース・ファイルのオプション)
トレース・オプションの設定時に **qsh** は、トレース・ファイルをハンドルする方法を決定するためにこの変数の値を使用します。この変数の値が "UNLINK" の場合は、**qsh** は、ルート・シェルでトレース・ファイルをオープンする前にトレース・ファイルをリンク解除します。この変数の値が "KEEP" の場合は、**qsh** は、現行のトレース・ファイルを保持します。デフォルト値は "UNLINK" です。

その他の変数

QIBM_CMP_FILE_SIZE
この変数は、パフォーマンスを向上するために **cmp** が内部バッファーに読み取る最大ファイル・サイズ (バイト単位) を制御します。最大サイズよりも大きいファイルの場合、**cmp** は一度に 1 バイトずつファイルを読み取ります。

QIBM_OD_OUTPUT_FORMAT (od の出力フォーマット)
この変数は od ユーティリティーの出力フォーマットを制御します。この変数の値が "OLD" の場合は、 **od** は前のリリースから古いフォーマットを使用します。古いフォーマットは現行の業界標準とは互換性がないので、できるだけ使わないようにしてください。デフォルト値はありません。

QIBM_QSH_CMD_ESCAPE_MSG (QSH CL コマンドからのエスケープ・メッセージの送信)
この変数は、CMD パラメーターが指定されている場合に、どのように QSH CL コマンドからメッセージを送信するかを制御します。値が "Y" の場合、QSH0006 および QSH0007 メッセージが常にエスケープ・メッセージとして送信され、終了状況がゼロより大きいときは、QSH0005 メッセージもエスケープ・メッセージとして送信されます。デフォルト値はありません。

QIBM_QSH_CMD_OUTPUT (QSH CL コマンドの出力を制御)
この変数は、CMD パラメーターが指定されている場合に、QSH CL コマンドからの出力を制御します。この値が "STDOUT" の場合、出力は C ランタイム端末セッションに表示されます。この値が "NONE" の場合、出力は廃棄されます。この値が "FILE" の場合、出力は指定されたファイルに書き込まれます。この値が "FILEAPPEND" の場合、出力は指定されたファイルに追加されます。デフォルト値は "STDOUT" です。

QIBM_QSH_INTERACTIVE_CMD (初期対話式コマンド)
この変数がコマンド・ストリングに設定されている場合、対話式セッションの開始時に **qsh** はそのコマンドを実行します。 **qsh** にコマンドを実行させるには、QSH CL コマンドを呼び出す前にその値を設定する必要があります。デフォルト値はありません。

QIBM_QSH_INTERACTIVE_TYPE (対話式セッションのタイプ)
この変数は、QSH CL コマンドによって開始される対話式セッションのタイプを設定します。この値が "NOLOGIN" の場合、対話式セッションはログイン・セッションではありません。それ以外の値の場合、対話式セッションはログイン・セッションです。デフォルト値はありません。

QIBM_SYSTEM_ALWMLTTHD (システムのマルチスレッド・ジョブの許可)
この変数は、マルチスレッド対応ジョブで system ユーティリティーが動作する方法を制御します。その変数の値が "Y" で、ジョブにスレッドが 1 つしかない場合、 **system** はジョブの中で CL コマンドを実行します。それ以外の場合、**system** は CL コマンドを実行するために新規のジョブを開始します。デフォルト値はありません。

QIBM_SYSTEM_USE_ILE_RC
system ユーティリティーが終了状況を設定する方法を制御するのにこの環境変数を設定します。変

数の値が "Y" である場合、**system** は終了状況を、CL コマンドによって呼び出されたプログラムの ILE 戻りコードに設定しますが、プログラムが戻りコードを設定していない場合には、ゼロに設定します。デフォルト値はありません。

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

ワード展開

ティルド展開、パラメーター展開、コマンド置換、算術展開、フィールド分割、パス名展開、および引用削除を含むワード展開についての情報を表示します。

ティルド展開

qsh がどのようにティルド文字を展開するかについての情報を表示するには、このリンクを選択します。

ワードの先頭にある引用符なしのティルド文字 (**~**) は、以下の規則に従って展開されます。

- **~** は、**HOME** 変数の値 (現行ユーザーのホーム・ディレクトリー) に展開される。
- **~user** は、指定されたユーザーのホーム・ディレクトリーに展開される。スラッシュ (**/**) またはワードの終わりまでのすべての文字は、ユーザー名と見なされる。
- **~+** は、**PWD** (作業ディレクトリー) 変数の値に展開される。
- **~-** は、**OLDPWD** (直前の作業ディレクトリー) 変数の値が設定されていれば、その値に展開される。

例

1. 現行ディレクトリーをユーザーのホーム・ディレクトリーに変更する場合:

   ```
   cd ~
   ```

2. 現行ディレクトリーをユーザー smith のホーム・ディレクトリーにある bin ディレクトリーに変更する場合:

   ```
   cd ~smith/bin
   ```

パラメーター展開

qsh がどのようにパラメーターを展開するかについての情報を表示するには、このリンクを選択します。

パラメーター展開のフォーマットは次のとおりです。

${*expression*}

expression は、対応する右中括弧 (**}**) までのすべての文字です。対応する右中括弧を判別する際には、円記号でエスケープされた右中括弧または引用符で囲まれたストリング内の右中括弧、および、組み込みの算術展開、コマンド置換、および変数展開内の右中括弧は除外されます。

パラメーター展開の最も単純な形式は次のとおりです。

${*parameter*}

parameter の値がある場合は、それが置き換えられます。パラメーターの名前または記号を中括弧で囲むことができます。ただし、複数桁の定位置パラメーターの場合、または *parameter* のあとに名前の一部と解釈されるおそれのある文字が続いている場合は、必ず中括弧を使用しなければなりません。二重引用符の中でのパラメーター展開は、次のように行われます。

1. 展開の結果についてのパス名展開は行われません。
2. 特殊パラメーター **@** を除き、展開の結果についてのフィールド分割は行われません。

パラメーター展開は、次のいずれかのフォーマットを使って変更できます。

${*parameter:-word***}**
: デフォルト値を使用します。 *parameter* が設定されていない、またはヌルである場合は、*word* の展開で置換されます。その他の場合は、*parameter* で置換されます。

${*parameter:=word***}**
: デフォルト値を代入します。 *parameter* が設定されていない、またはヌルである場合は、*word* の展開結果が *parameter* に代入されます。どの場合も、*parameter* の最終値で置換されます。この形式で代入できるのは、変数だけです (定位置パラメーターや特殊パラメーターへの代入はできません)。

${*parameter:?word***]}**
: ヌルまたは設定されていない場合にエラーを戻します。 *parameter* が設定されていない、またはヌルの場合は、*word* の展開 (word を省略した場合は、設定されていないことを示すメッセージ) が標準エラーに書き込まれ、非対話式シェルは 0 以外の終了状況で終了します。その他の場合は、*parameter* で置換されます。

${*parameter:+word***}**
: 代替値を使用します。 *parameter* が設定されていない、またはヌルの場合は、ヌルで置換されます。その他の場合は、*word* の展開で置換されます。

上記 4 つのパラメーター展開のフォーマットの中でコロンを使用すると、未設定またはヌルの parameter についてのテストが行われます。コロンを省略した場合は、未設定の parameter だけについてのテストが行われます。

${#*parameter***}**
: ストリングの長さ。 *parameter* が @ または * である場合は、定位置パラメーターの数で置換されます。その他の場合は、*parameter* の値の長さで置換されます。

${*parameter***%***word***}**
: 最小のサフィックス・パターンを削除します。 *word* はパターンに展開されます。結果は、パターンに一致するサフィックスの最小部分を削除したあとの *parameter* です。

${*parameter***%%***word***}**
: 最大のサフィックス・パターンを削除します。 *word* はパターンに展開されます。結果は、パターンに一致するサフィックスの最大部分を削除したあとの *parameter* です。

${*parameter***#***word***}**
: 最小のプレフィックス・パターンを削除します。 *word* はパターンに展開されます。結果は、パターンに一致するプレフィックスの最小部分を削除したあとの *parameter* です。

${*parameter***##***word***}**
: 最大のプレフィックス・パターンを削除します。 *word* はパターンに展開されます。結果は、パターンに一致するプレフィックスの最大部分を削除したあとの *parameter* です。

${*parameter:offset***}**

${*parameter:offset:length***}**
: オフセットで始まるサブストリング。この展開の値は、*offset* で指定されるバイトで始まる *length* バイトのサブストリングです。 *length* が指定されていないか、*length* の値が原因で *parameter* の長さを超えて展開されてしまう場合、サブストリングは、*parameter* の最後のバイトで終了します。 *offset* と *length* は両方とも算術式 であり、ゼロ以上の値に評価されなければなりません。*parameter* の最初のバイトは、オフセットがゼロと定義されます。

${*parameter/pattern/string***}**

${*parameter***/***pattern***/***string***}**

パターンをストリングに置換します。この展開の値は、*pattern* を最長一致法で *string* に置き換えた、*parameter* の値です。最初の形式では、*pattern* の最初の一致部分だけが置き換えられます。 2 番目の形式では、*pattern* のすべての一致部分が置き換えられます。 *pattern* が # で始まる場合、*parameter* の先頭部分が一致している必要があります。 *pattern* が % で始まる場合、*parameter* の末尾が一致している必要があります。

例

1. 変数 QSH_VERSION を展開する場合:

   ```
   echo ${QSH_VERSION}
   ```

2. 変数ファイル名を展開し、デフォルト値を使用する場合:

   ```
   echo ${filename:-/tmp/default.txt}
   ```

3. 変数インデックスを展開し、デフォルト値を割り当てる場合:

   ```
   echo ${index:=0}
   ```

4. 変数ファイル名を展開し、未設定の場合にエラーを示す場合:

   ```
   echo ${filename:?Variable is not set}
   ```

5. ストリングの長さを使用して、変数 DIRLIST を展開する場合:

   ```
   DIRLIST=/usr/bin:/home/mike
   echo ${#DIRLIST}
   ```

6. 最小サフィックス除去パターンを使用して、変数 DIRLIST を展開する場合:

   ```
   DIRLIST=/usr/bin:/home/mike
   echo ${DIRLIST%/*}
   ```

7. 最大サフィックス除去パターンを使用して、変数 DIRLIST を展開する場合:

   ```
   DIRLIST=/usr/bin:/home/mike
   echo ${DIRLIST%%:*}
   ```

8. 最小プレフィックス除去パターンを使用して、変数 DIRLIST を展開する場合:

   ```
   DIRLIST=/usr/bin:/home/mike
   echo ${DIRLIST#/usr}
   ```

9. 最大プレフィックス除去パターンを使用して、変数 DIRLIST を展開する場合:

   ```
   DIRLIST=/usr/bin:/home/mike
   echo ${DIRLIST##*/}
   ```

10. オフセットから始まるサブストリングを使用して、変数 DIRLIST を展開する場合:

    ```
    DIRLIST=/usr/bin:/home/mike
    echo ${DIRLIST:5:3}
    ```

11. パターンの置換ストリングを使用して変数 DIRLIST を展開する場合:

    ```
    DIRLIST=/usr/bin:/home/mike
    echo ${DIRLIST/m?ke/joel}
    ```

関連概念:

8 ページの『パラメーター』
パラメーターは、データを格納するために使用されます。

コマンド置換

qsh がどのようにコマンド置換を展開するかについての情報を表示するには、このリンクを選択します。

コマンド置換を使用すると、コマンド名自体をコマンドの出力で置き換えることができます。コマンド置換が行われるのは、次のようにコマンドが括弧で囲まれている場合です。

$(*command*)

また、次のように単一引用符 (`) で囲まれている場合も、コマンド置換が行われます。

`*command*`

単一引用符方式は互換性確保のために提供されているものなので、できるだけ使わないようにしてください。

シェルは、コマンド置換を展開するときに、サブシェル環境の中で *command* を実行し、コマンド置換を *command* の標準出力で置き換えます。コマンド置換の末尾に <newline> があれば除去します。 出力の終わりより前にある埋め込み <newline> は除去されません。ただし、**IFS** 変数の値、および有効な引用方式によっては、フィールド分割時に <space> に変換されることがあります。

例

1. ls コマンドの出力に変数リストを設定する場合:

```
list=$(ls)
```

算術展開

qsh がどのように算術式を展開するかについての情報を表示するには、このリンクを選択します。

算術展開は、算術式を評価し、その式の値で置換するためのメカニズムを提供します。算術展開のフォーマットは次のとおりです。

$((*expression*))

expression は、二重引用符で囲まれている場合と同様に取り扱われます。ただし、*expression* の中の二重引用符は特別の取り扱いを受けません。シェルは、パラメーター展開、コマンド置換、および引用符除去のために、*expression* の中のすべてのトークンを展開します。 **qsh** は、結果を算術式として扱い、その式の値で置換します。

算術式

算術式は次のような状況下で指定することができます。

- 算術展開の中で。
- let ユーティリティーの各引数として。
- shift ユーティリティーの引数として。
- printf ユーティリティーの算術形式のオペランドとして。
- test ユーティリティーの算術比較演算子へのオペランドとして。
- ulimit ユーティリティーの引数として。
- 「オフセットで始まるサブストリング」パラメーター展開の中で。

qsh では、float オプションの設定に基づいて、整数演算または浮動小数点演算が実行されます。 float オプションがオンに設定されているときは、**qsh** で浮動小数点演算が実行されます。

整数の形式は [*base*#]*number* です。

- *base* は、算術の基数を指定する 2 - 36 の範囲内の 10 進整数です。デフォルトの基数は 10 です。
- *number* は負ではない数値です。 10 より大きい基数の場合は、9 より大きい数はアルファベットの文字で表されます。たとえば、基数 16 を使用する場合は、10 進数の 10 は A で表されます。

浮動小数点数の形式は、[+|-] *number*[.*number*] [*exponent*] です。

- *number* は負ではない 10 進数です。
- *exponent* は、E または e の後ろに + または - と負ではない 10 進数を付けたものです。

算術式では、次の ANSI C 言語演算子が次の優先順位で使用されます。

(式) 括弧を使用すると、優先順位の規則を変更することができます。

単項演算子

+*expression* 単項 +

-*expression* 単項 -

~*expression* ビット単位の否定

!*expression* 論理否定

乗除演算子

expression * *expression* 乗算

expression / *expression* 除算

expression % *expression* 剰余

加減演算子

expression + *expression* 加算

expression - *expression* 減算

ビット単位移動演算子

expression << *expression* 第 1 の式を第 2 の式に指定するビット数だけ左に移動

expression >> *expression* 第 1 の式を第 2 の式に指定するビット数だけ右に移動

関係演算子

expression < *expression* より小さい

expression <= *expression* より小さいか等しい

expression > *expression* より大きい

expression >= *expression* より大きいか等しい

ビット単位 AND 演算子

expression & *expression* ビット単位 AND で、どちらの式でも 1 である各ビット位置には 1 が入り、その他のすべてのビット位置には 0 が入ります。

ビット単位排他 OR 演算子

expression ^ *expression* ビット単位排他 OR で、いずれか一方の式のみが 1 である各ビットには 1 が入り、その他のすべてのビット位置には 0 が入ります。

ビット単位 OR 演算子

expression | *expression* ビット単位 OR で、いずれか一方の式が 1 である各ビット位置には 1 が入り、その他のすべてのビットには 0 が入ります。

論理 AND 演算子

expression && *expression* 論理 AND で、両方の式が真である場合に結果が真になります。

論理 OR 演算子

expression || *expression* 論理 OR で、いずれか一方の式が真である場合に結果が真になります。

条件演算子

expression ? *expression* : *expression* 第 1 の式が真である場合に、第 2 の式が評価される条件演算子。第 1 の式が真でない場合は、第 3 の式が評価されます。

代入演算子

expression = *expression* 単純代入

expression *= *expression* 代入および乗算

expression /= *expression* 代入および除算

expression %= *expression* 代入および剰余

expression += *expression* 代入および加算

expression -= *expression* 代入および減算

expression <<= *expression* 代入および左移動

expression >>= *expression* 代入および右移動

expression &= *expression* 代入およびビット単位 AND

expression ^= *expression* 代入およびビット単位排他 OR

expression |= *expression* 代入およびビット単位 OR

注:	浮動小数点演算を使用するときは、剰余、左移動、右移動、ビット単位 AND、ビット単位排他 OR、ビット単位 OR の各演算子は使用できません。

例

1. 2 つの 10 進数を加算する場合:

   ```
   echo $((2+2))
   ```

2. 2 つの 16 進数を加算する場合:

   ```
   echo $((16#A + 16#20))
   ```

3. 変数指標を 1 だけ増分する場合:

   ```
   let index+=1
   ```

4. 複合式を評価する場合:

   ```
   echo $((5+9-2*3/2))
   ```

5. 2 つの浮動小数点数を加算する場合:

   ```
   set -F
   echo $((5.75+9.157))
   set +F
   ```

フィールド分割

qsh がどのようにフィールドをワードに分割し、パターンを使用してパス名を展開し、引用符を除去するかについての情報を表示するには、このリンクを選択します。

パラメーター展開、コマンド置換、および算術展開のあとで、 **qsh** は、二重引用符に囲まれていない展開と置換の結果をフィールド分割のためにスキャンします。その結果複数のフィールドが発生することがあります。

qsh は、**IFS** 変数の各文字をそれぞれ区切り文字と見なし、これらの区切り文字を使用して、パラメーター展開およびコマンド置換の結果をフィールドに分割します。 **IFS** 変数の値がヌルの場合は、フィールド分割は行われません。

パス名展開

noglob オプションが設定されていない場合は、フィールド分割の完了後にパス名展開が行われます。各ワードは、スラッシュで区切った一連のパターンと見なされます。展開のプロセスでは、各パターンを指定のパターンに一致するストリングで置き換えれば、名前を作成できるすべての既存ファイルの名前で、ワードが置き換えられます。この場合、以下のような 2 つの制約事項があります。

1. スラッシュの入ったストリングに対してパターンを突き合わせることができません。
2. パターンの最初の文字がピリオドでない限り、ピリオドで始まるストリングに対してそのパターンを突き合わせることはできません。

引用符除去

引用文字、円記号 **(¥)**、単一引用符 **(')**、および二重引用符 **(")** は、引用符で囲まれている場合以外は除去されます。

パターン

qsh がどのようにパターンを展開するかについての情報を表示するには、このリンクを選択します。

パターンは、通常文字 (互いに突き合わせることができる文字) とメタ文字で構成されます。 メタ文字には次のものがあります。

!、*****、**?**、および **[**

これらの文字は、引用符で囲まれているときはそれぞれの特別な意味を失います。コマンド置換または変数置換が行われるときに、ドル記号 **($)** または単一引用符 **(')** が二重引用符で囲まれていない場合は、その変数の値またはコマンドの出力がスキャンされ、これらの文字があれば、メタ文字として扱われます。

アスタリスク **(*)** はどのようなストリングにも一致します。

疑問符 **(?)** はどのような単一の文字にも一致します。

左大括弧 **([)** は文字クラスの始めを示します。 文字クラスの終わりは右大括弧 **(])** で指示されます。 右大括弧がない場合は、左大括弧は、文字クラスの始めを示すのではなく、文字 **[** として扱われます。 文字クラスは、大括弧の間のどの文字にも一致します。 負符号 **(-)** を使用すると、一定範囲の文字を指定することができます。文字クラスの最初の文字として感嘆符 **(!)** を使用すると、文字クラスは、大括弧の間の文字以外の文字と一致します。

注:	EBCDIC 文字は連続していないため、文字の範囲を指定した場合、他のシステムとは異なる結果が生じることがあります。

文字クラスに右大括弧を使用するときは、リストの最初の文字 (**!** を使用する場合はその次の文字) として指定してください。 文字クラスに負符号を使用するときは、リストの最初または最後の文字として指定してください。

リダイレクト

リダイレクトは、コマンドが読み込む入力元、またはコマンドの実行結果を送る出力先を変更するために使用します。 リダイレクトでは、一般に、ファイル (実際にはその既存の参照) がオープン、クローズ、または複製されます。

リダイレクトに使用するフォーマットは次のとおりです。

[*n*] *redir-op file*

redir-op は、下記に示すリダイレクト演算子の 1 つで、*n* はファイル記述子を指す数値 (指定は任意) です。 リダイレクトには次のものがあります。

[*n* **]<** *file*
: 標準入力 (または *n*) を *file* からの入力に変更します。

[*n1* **]<&***n2*
: ファイル記述子 *n2* の内容に基づいて標準入力 (または *n1*) の複製を作成します。

[*n* **]<&-**
: 標準入力 (または *n*) をクローズします。

[*n* **]>** *file*
: 標準出力 (または *n*) を *file* への出力に変更します。

[*n* **]>|** *file*
: 標準出力 (または *n*) を *file* への出力に変更しますが、noclobber オプションは無効にします。

[*n* **]>>** *file*
: 標準出力 (または *n*) を *file* に追加します。

[*n1* **]>&***n2*
: *n2* の内容に基づいて標準出力 (または *n1*) の複製を作成します。

[*n* **]>&-**
: 標準出力 (または *n*) をクローズします。

/QSYS.LIB/QTEMP.LIB ディレクトリーは、ジョブが終了するときに削除され、各コマンドごとに新しいジョブが開始されて終了するため、リダイレクトには使用しないのが最善です。

here ドキュメント

here ドキュメントのフォーマットは次のとおりです。

[n]<<[-] *delimiter*
: *here-doc-text ...*

delimiter

delimiter より前の連続した行のすべてのテキストが保管され、標準入力 (または、指定されていればファイル記述子 *n*) においてコマンドで使えるようになります。最初の行に指定する *delimiter* を引用符で囲んだ場合は、*here-doc-text* はストリングとして取り扱われます。引用符がなければ、テキストには、パラメーター展開、コマンド置換、および算術展開が適用されます。 演算子に **<<** でなく **<<-** を使用した場合は、*here-doc-text* 内の先行タブは取り除かれます。

単純コマンド

単純コマンドは、一連のオプショナルの変数割り当ておよびリダイレクトのあとにコマンド名を付けたものです。

qsh が単純コマンドを認識すると、次の処理が行われます。

1. name=value の形式の先行語が取り除かれ、単純コマンドの環境に割り当てられます。各リダイレクト演算子およびその引数が、ステップ 3 での処理のために保管されます。
2. 残りのワードがワード展開の説明に従って展開されますが、これらの残りのワードのうちの最初のワードがコマンド名と見なされます。それ以降のワードは、コマンドの引数と見なされます。コマンド名が見つからない場合は、ステップ 1 で認識された name=value の変数代入で現行シェルの処理が決まります。
3. リダイレクトの説明どおりにリダイレクトが行われます。

パス検索

単純コマンドにスラッシュが入っていない場合、 **qsh** は次の順番で検索してコマンドを見つけ出します。

1. 該当の名前を持つ特殊組み込みユーティリティーを検索する。
2. 該当の名前を持つシェル関数を検索する。
3. 該当の名前を持つ正規組み込みユーティリティーを検索する。
4. **PATH** 変数の中の各ディレクトリーを順に調べて正規ユーティリティーを検索する。

コマンド名にスラッシュ (/) が入っている場合は、上記の検索は行われず、正規ユーティリティーとして実行されます。

組み込みユーティリティーは、新しいプロセスを開始することなく、シェルで内部的に実行されます。特殊組み込みユーティリティーは、次のような点で正規組み込みユーティリティーと異なります。

1. 特殊組み込みユーティリティー内に構文エラーがあると、非対話式シェルが終了します。
2. 特殊組み込みユーティリティーで指定されている変数代入は、ユーティリティーの終了後も有効です。

特殊組み込みユーティリティーには次のものがあります。break、 colon、continue、declare、dot、 eval、exec、exit、 export、local、readonly、 return、set、shift、 source、trap、typeset、および unset。

シェル関数が実行されるときに、シェルの定位置パラメーターはすべてそのシェル関数の引数に設定されます (ただし、特殊パラメーター **0** だけは変更されません)。明示的にコマンド環境で登録された変数 (関数名の前に変数代入の指定を介して) は、関数のローカル変数となり、指定の値に設定されます。定位置パラメーターは、シェル関数が完了すると元の値に戻ります。

正規ユーティリティーが実行されると、**qsh** は新しいプロセスを開始し、引数と環境をプログラムに渡します。プログラムがシェル・スクリプトである場合は、**qsh** はサブシェル内でそのプログラムを解釈します。この場合、**qsh** が再初期設定されるので、結果は、シェル・スクリプトを処理するために新しいシェルが呼び出された場合と同じになります。

コマンドの終了状況

各コマンドの終了状況は、他のシェル・コマンドの動作に影響を与えることがあります。規則ではコマンドの終了状況は、正常 (成功) の場合は 0 であり、失敗、エラー、偽を示す場合は 0 以外の値です。各コマンドが戻す終了コードとそれぞれの意味は、各コマンドに関するドキュメンテーションを参照してください。終了状況は次のいずれかの値です。

- 0 成功。
- 1 - 125 失敗。
- 126 **qsh** がコマンドを検出したが、そのコマンドが実行可能でない。
- 127 **qsh** がコマンドを検出できなかった。
- 128 以上 コマンドがシグナルによって終了された。値は、128 にシグナル番号を加えた値です。

パイプライン

パイプラインは、パイプライン制御演算子 (|) で区切った 1 つまたは複数のコマンドです。 最後のコマンドを除く各コマンドの標準出力は、その次のコマンドの標準入力と結合されます。

パイプラインのフォーマットは次のとおりです。

[!] *command1* [| *command2* ...]

command1 の標準出力は、*command2* の標準入力に結合されます。 パイプラインは、コマンドに組み込まれたリダイレクト演算子で指定されたリダイレクトを実行する前に、そのコマンドの標準入力、標準出力、またはその両方を割り当てます。パイプラインの終了状況は、最後の *command* の終了状況です。

パイプラインがバックグラウンド内にない場合は (以下を参照)、**qsh** はすべてのコマンドが終了するまで待ちます。

パイプラインの前に予約語である **!** がない場合は、終了状況は、パイプラインで指定されている最後のコマンドの終了状況です。パイプラインの前に ! がある場合は、終了状況は、最後のコマンドの終了状況の NOT です。つまり、最後のコマンドの戻り値が 0 の場合、終了状況は 1、最後のコマンドの戻り値が 0 より大きい場合、終了状況は 0 です。

パイプラインによる標準入力または標準出力またはその両方の割り当ては、リダイレクトより前に行われるので、リダイレクトによってその割り当てを変更することができます。 たとえば次のようにします。

command1 2>&1 | command2

この場合、*command1* の標準出力および標準エラーが、 *command2* の標準入力に渡されます。

リスト

リストは、アンパーサンド (**&**) またはセミコロン (**;**) で区切った一連のコマンドです。オプションとして、末尾に <newline>、アンパーサンド、またはセミコロンを付けることができます。

AND-OR リストは、**&&** または || で区切った一連のコマンドです。 どちらの演算子も優先順位は同じです。

非同期リスト

コマンドが制御演算子アンパーサンド (**&**) で終わっている場合は、**qsh** はそのコマンドを非同期に実行します。 つまり、**qsh** は、そのコマンドが終了するまで待たずに次のコマンドを実行します。バックグラウンドでコマンドを実行する場合のフォーマットは次のとおりです。

command1 **&** [*command2* **&** ...]

対話式オプションが設定されていない場合は、非同期コマンドの標準入力は /dev/qsh-stdin-null に設定されます。非同期リストの終了状況は、最後の *command* の終了状況です。

順次リスト

セミコロン (**;**) で区切られたコマンドは順次に実行されます。順次リストのフォーマットは次のとおりです。

command1 [**;** *command2* ...]

リスト内のコマンドは、リストに指定されたとおりの順序で実行されます。順序リストの終了状況は、最後の *command* の終了状況です。

AND リスト

AND リストのフォーマットは次のとおりです。

command1 [**&&** *command2* ...]

AND リストの場合、**qsh** はまず *command1* を実行し、*command1* の戻り状況が 0 であれば、*command2* を実行します。そして、いずれかのコマンドの戻り状況が 0 以外の値になるか、または実行すべきコマンドがなくなるまで、この処理を繰り返します。 AND リストの戻り状況は、最後に実行された *command* の戻り状況です。

OR リスト

OR リストのフォーマットは次のとおりです。

command1 [|| *command2* ...]

OR リストの場合、**qsh** はまず *command1* を実行し、 *command1* の終了状況が 0 以外であれば、*command2* を実行します。そして、いずれかのコマンドの終了状況が 0 になるか、または実行すべきコマンドがなくなるまで、この処理を繰り返します。 OR リストの終了状況は、最後に実行された *command* の終了状況です。

複合コマンド

複合コマンドは、他のコマンドの制御フローを提供します。複合コマンドは予約語で始まり、それに対応する予約語で終わります。

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

コマンドのグループ化

コマンドのグループ化についての情報を表示するには、このリンクを選択します。

次のどちらかを使って、コマンドをグループ化することができます。

(*list* **)**

または

{ *list;* **}**

(*list*) の場合は、**qsh** はサブシェル環境内で *list* を実行します。

例

1. 2 つのコマンドをサブシェルにグループ化する場合:

```
( ls | grep apple )
```

if コマンド

if-then-else-fi コマンドについての情報を表示するには、このリンクを選択します。

if コマンドの構文は次のとおりです。

if *list1*

then *list2*

[elif *list3*

then *list4* **]** ...

[else *list5* **]**

fi

最初に **qsh** は *list1* を実行します。その戻り状況が 0 なら、**qsh** は *list2* を実行します。 0 以外の場合は *list3* が実行されます。そして、この結果の戻り状況が 0 なら、**qsh** は *list4* を実行します。 0 でない場合は、**qsh** は *list5* を実行します。

例

1. if-then-fi コマンドの場合:

```
x=4
y=9
if test $x -lt $y
then
  echo $x is less than $y
fi
```

2. if-then-else-fi コマンドの場合:

```
x=10
y=9
if test $x -lt $y
then
  echo echo $x is less than $y
else
  echo echo $x is greater than or equal to $y
fi
```

3. if-then-elif-else-fi コマンドの場合:

```
x=4
y=4
if test $x -lt $y
then
  echo echo $x is less than $y
elif test $x -eq $y
then
  echo $x is equal to $y
else
  echo $x is greater than or equal to $y
fi
```

条件コマンド

条件コマンドについての情報を表示するには、このリンクを選択します。

[[...]] コマンドの構文は次のとおりです。

[[*expression* **]]**

このコマンドは、条件式 *expression* の評価結果により、0 または 1 の状況を戻します。条件式のフォーマットは、test ユーティリティーで評価された式と同じです。 *expression* が評価される前に、**qsh** は、ティルド展開、パラメーター展開、算術展開、コマンド置換、および引用削除を行います。

例

1. コマンド置換を使用する条件コマンドの場合:

```
if [[ $(grep -c apple fruits.txt) -eq 0 ]]
then
  echo There are no apples in fruit.txt
fi
```

case コマンド

case-esac コマンドについての情報を表示するには、このリンクを選択します。

case コマンドの構文は次のとおりです。

case *word* **in**

pattern1) *list1* ;;

pattern2 | *pattern3*) *list2* ;;

...

esac

qsh は、各 *pattern* を順に展開して、*word* を展開したものと一致するかどうかを調べます。一致するものがあれば、**qsh** は対応する *list* を実行します。一致するものが 1 つ見つかると、それ以後は pattern は展開されません。 pattern についての詳細は、パターンを参照してください。

例

1. コマンド行オプションを処理するための case コマンドの場合:

```
while getopts ap:t: c ; do
  case $c in
    a) aflag=1;;
    p) pflag=1
       path=$OPTARG;;
    t) time=$OPTARG;;
    *) print -u2 "Invalid option"
       exit 1;;
  esac
done
```

select コマンド

select-do-done コマンドについての情報を表示するには、このリンクを選択します。

select コマンドの構文は次のとおりです。

select *name* **[in** *word ...* **]**

do *list*

done

words は展開され、項目リストを生成します。 *word* を指定しない場合、定位置パラメーターが展開されます。展開された word のセットは、それぞれ番号が前に付けられた上で、標準エラーに書き込まれます。その後、PS3 プロンプトが表示され、標準入力から 1 行読み取ります。その行が、表示されている word の 1 つに対応する数値を含んでいた場合、**qsh** は *name* の値を、その数値に対応する word に設定します。その行が空白だった場合、**qsh** は再びリストを表示します。 REPLY 変数が入力行の内容に設定されます。

qsh は、break、return、または exit の各コマンドが実行されるまで、*list* にあるコマンドを実行します。標準入力から EOF を読み取った場合にも、**select** は完了します。

例

1. リストから選択するための select コマンドの場合:

```
PS3="Please select a number "
list="alpha beta gamma delta epsilon"
select value in $list ; do
  echo Value for selection $REPLY is $value
  break
done
```

while コマンド

while-do-done コマンドについての情報を表示するには、このリンクを選択します。

while コマンドの構文は次のとおりです。

while *list1*

do *list2*

done

list1 の戻り状況が 0 である限り、**qsh** は 2 つのリストを繰り返し実行します。 *list1* の戻り状況が 0 以外の値になると、コマンドは完了します。

例

1. 条件が満たされるまで繰り返す while コマンドの場合:

```
max=100
index=0
while [[ $index -lt $max ]] ; do
  echo Index is $index
  let index+=1
done
```

until コマンド

until-do-done コマンドについての情報を表示するには、このリンクを選択します。

until コマンドの構文は次のとおりです。

until *list1*

do *list2*

done

list1 の戻り状況が 0 以外の値である限り、**qsh** は 2 つのリストを繰り返し実行します。 *list1* の戻り状況が 0 になると、コマンドは完了します。

例

1. 条件が満たされるまで繰り返す until コマンドの場合:

```
max=100
index=0
until [[ $index -eq $max ]] ; do
  echo Index is $index
  let index+=1
done
```

for コマンド

for-do-done コマンドについての情報を表示するには、このリンクを選択します。

for コマンドの構文は次のとおりです。

for *variable* **in** *word ...*

do *list*

done

words が展開され、*variable* を各 *word* に順に設定して、*list* が繰り返し実行されます。**do** および **done** は中括弧 (**{ }**) で置換できます。

例

1. オブジェクトのリストを処理する for コマンドの場合:

```
list=$(ls *.class)
for object in $list
do
  system "DSPJVAPGM $object"
done
```

関数

関数についての情報を表示するには、このリンクを選択します。

関数定義の構文は次のとおりです。

[function] *name* **( )** *command*

関数定義は、*name* という名前の関数をインストールし、戻り状況 0 を戻すステートメントです。 *command* は、通常は中括弧 (**{ }**) で囲まれたリストです。

name が単純コマンドとして指定されている場合は、 **qsh** は *command* を実行します。 単純コマンドに指定された引数は、関数の実行中は一時的に定位置パラメーターになります。 特殊パラメーター **0** は変更されません。 **local** を使用することによって、関数内部でローカル変数を定義できます。 **return** を使用することによって、関数を終了させ、その関数呼び出しの後にある次のコマンドの実行を再開させることができます。

例

次に示すコードは、PING CL コマンドとの **qsh** インターフェースを提供する関数の例です。

```
ping()
{
  # Initialize variables and make them local to this function
  local nbrpkt='' waittime='' intnetadr='' msgmode='' pktlen='' ipttl='' host=''
  local c

  # Process the options
  while getopts c:i:I:qs:T:v c
  do  case $c in
      c)  nbrpkt="NBRPKT($OPTARG)";;
      i)  waittime="WAITTIME($OPTARG)";;
      I)  intnetadr="INTNETADR('$OPTARG')"
          host="*INTNETADR";;
      q)  msgmode='MSGMODE(*QUIET)';;
      s)  pktlen="PKTLEN($OPTARG)";;
      T)  ipttl="IPTTL($OPTARG)";;
      v)  msgmode='MSGMODE(*VERBOSE)';;
     \?)  print -u2 "Usage: ping [-c count] [-i seconds] [-I ipaddr] [-q]" \
            "[-s size] [-T ttl] [-v] hostname"
          return 1;;
      esac
  done

  # Run the command
  shift $OPTIND-1
  system ping ${host:-$1} $intnetadr $nbrpkt $waittime $msgmode $pktlen $ipttl
}
```

Qshell の使用

QSH CL コマンドを使用する方法、Qshell 環境を構成する方法、およびユーティリティーを開発する方法を見つけるには、このリンクを選択します。

Qshell 対話式セッションの使用

QSH 開始 (STRQSH) コマンドは、QSH とも呼ばれます。これは、Qshell 対話式セッションの開始、または Qshell コマンドの実行のいずれをも行える CL (制御言語) コマンドです。

パラメーターを指定しないで対話式ジョブを実行すると、STRQSH は対話式 Qshell セッションを開始します。ジョブ内で Qshell セッションがまだ活動状態になっていない場合、次のイベントが発生します。

1. 新しい Qshell セッションが開始され、端末ウィンドウが表示される。
2. ファイル /etc/profile が存在していれば、**qsh** はそのファイルからコマンドを実行する。
3. ユーザーのホーム・ディレクトリーの中にファイル .profile が存在していれば、**qsh** はそのファイルからコマンドを実行する。
4. **ENV** 変数の展開によって指定されたファイルが存在していれば、**qsh** はそのファイルからコマンドを実行する。

対話式ジョブで Qshell セッションがすでに活動状態になっていれば、その既存セッションに再接続されます。

端末ウィンドウで Qshell コマンドを入力し、そのコマンドからの出力を見ることができます。端末ウィンドウには次の 2 つの部分があります。

- コマンドを入力するための入力行

- 入力したコマンドのエコー、およびコマンドによって生成された出力を表示する出力域

以下の機能キーを使用できます。

機能キー	説明
F3 (終了)	端末ウィンドウをクローズし、Qshell セッションを終了します。
F5 (最新表示)	出力域を最新表示します。
F6 (印刷)	出力域をスプール・ファイルに出力します。
F7 (次ページ)	出力域を 1 ページ分ロールアップします。コマンド行上に数値があると、出力域はその行数分ロールアップします。
F8 (前ページ)	出力域を 1 ページ分ロールダウンします。コマンド行上に数値があると、出力域はその行数分ロールダウンします。
F9 (検索)	以前使用したコマンドを検索します。このキーを複数回押すことによって、以前のいずれのコマンドも検索できます。 たとえば、最後から 2 番目に入力したコマンドを検索するには、このキーを 2 回押してください。特定のコマンドにカーソルを置いてこのキーを押すことによって、そのコマンドを選択して再実行することもできます。 2 バイト CCSID を使って対話式ジョブを実行しているときは、このキーは使用できません。
F11 (行の折り返しトグル)	出力域の行の折り返しモードと切り捨てモードを切り替えます。行の折り返しモードでは、端末ウィンドウの幅より長い行は折り返されて次行に表示されます。 切り捨てモードでは、端末ウィンドウ幅を超えた行部分は表示されません。
F12 (切断)	Qshell セッションから切断します。このキーは、端末ウィンドウをクローズするだけで、Qshell セッションを終了させるわけではありません。切断した Qshell セッションは、STRQSH を再び実行すれば再表示できます。
F13 (消去)	出力域を消去します。
F14 (コマンド行の長さの調整)	コマンド行の長さを 4 行に調整します。コマンド行上に数値があると、コマンド行の長さはその行数に調整されます。
F17 (上部)	出力域の上部を表示します。
F18 (下部)	出力域の下部を表示します。
F19 (左)	出力域を左へシフトします。コマンド行上に数値があると、出力域はその列数分シフトされます。
F20 (右)	出力域を右へシフトします。コマンド行上に数値があると、出力域はその列数分シフトされます。
F21 (コマンド入力)	CL コマンドを入力できるコマンド入力ウィンドウを表示します。
SysReq 2	すべての子プロセスに SIGINT シグナルを送信して、現行のコマンドの実行を中断します。

CL から Qshell コマンドを実行

Qshell 開始コマンドを使用して CL コマンド環境から Qshell コマンドを実行できます。

QSH 開始 (STRQSH) コマンドは、QSH とも呼ばれます。これは、Qshell 対話式セッションの開始、または Qshell コマンドの実行のいずれをも行える CL (制御言語) コマンドです。

STRQSH が CMD パラメーター内で呼び出されると、指定された Qshell コマンドが STRQSH によって実行されます。 CMD パラメーターに指定できる値は次のとおりです。

***NONE**
コマンドを指定せずに、対話式セッションが開始されます。 CMD(*NONE) を指定した場合、STRQSH をバッチ・ジョブで実行しても、STRQSH は何の処理もしません。

コマンド
実行する Qshell コマンド。コマンドの長さは最大 5000 バイトです。 ブランクまたはその他の特殊文字を使用する場合は、コマンドをアポストロフィで囲む必要があります。 アポストロフィ自体をコマンド内部で使う場合は、2 個のアポストロフィを使う必要があります。

コマンドの実行時には STRQSH は **qsh** を実行し、指定された Qshell コマンドを実行し、C 実行時端末セッションに対するコマンドで生成されたすべての出力を表示してから、 **qsh** を終了します。 **qsh** は、コマンドの実行のために始動されると、プロファイル・ファイルを実行しないことに注意してください。

QIBM_QSH_CMD_OUTPUT 環境変数を設定することで、出力の仕方を制御できます。環境変数には次のような値を指定することができます。

STDOUT
C 実行時端末セッションに出力を表示します。これはデフォルト値です。

NONE 生成されたすべて出力を廃棄します。

FILE=pathname
パス名で指定されたファイルに出力を保管します。出力がそのファイルに書き込まれる前に、そのファイルは切り捨てられます。

FILEAPPEND=pathname
パス名で指定されたファイルに出力を保管します。出力はファイルの末尾に追加されます。

コマンドが終了すると STRQSH は以下のメッセージのいずれかを送信します。

- QSH0005 コマンドを実行したプロセスが正常に完了した場合。プロセスの終了状況がそのメッセージに入っています。
- QSH0006 コマンドを実行したプロセスがシグナルで完了した場合。シグナル番号がそのメッセージに入っています。
- QSH0007 コマンドを実行したプロセスが例外で完了した場合。

デフォルトでは、メッセージは完了メッセージとして送信されます。環境変数 QIBM_QSH_CMD_ESCAPE_MSG を設定すれば、メッセージをエスケープ・メッセージとして送信することができます。環境変数の値が Y の場合、QSH0006 および QSH0007 メッセージは常にエスケープ・メッセージとして送信され、また、終了状況がゼロより大であれば QSH0005 メッセージもエスケープ・メッセージとして送信されます。

関連タスク:

30 ページの『Qshell 対話式セッションの使用』
QSH 開始 (STRQSH) コマンドは、QSH とも呼ばれます。 これは、Qshell 対話式セッションの開始、 ま

たは Qshell コマンドの実行のいずれをも行える CL (制御言語) コマンドです。

PASE から Qshell コマンドを実行

PASE 環境から Qshell コマンドを実行できます。

i5/OS PASE には qsh コマンドが備わっており、これによって、対話式セッションまたはコマンドのいずれかを実行する qsh を呼び出すことができます。これを使用して i5/OS PASE シェルからあらゆる Qshell コマンドを実行できます。

関連情報:

i5/OS PASE

環境のカスタマイズ

Qshell 環境をカスタマイズするには、これらの 3 つのプロファイル・ファイルを使用します。各プロファイル・ファイルは、任意の Qshell コマンドを含むことができるシェル・スクリプトです。

サポートされる環境変数の完全なリストについては、変数のトピックを参照してください。

グローバル・プロファイル・ファイル

ファイル /etc/profile が存在する場合、**qsh** はログイン時に現行環境でこのファイルを実行します。 通常は、管理者がこのファイルを保守して、すべてのユーザーに対応するシステム全体のデフォルト値を設定します。 共通権限を読み取りおよび実行に設定することで、このファイルを保護する必要があります。

すべてのユーザーに対応するシステム全体の PATH 変数を定義する /etc/profile ファイルの例を次に示します。

```
# Sample /etc/profile file
export PATH=/usr/bin:.:/QOpenSys/usr/bin
```

プロファイル・ファイル

ファイル .profile がユーザーのホーム・ディレクトリーに存在している場合、**qsh** はログイン時に現行環境でこのファイルを実行します。このファイルは、ログイン環境をカスタマイズするために使用されます。

ユーザーの環境ファイルを定義し、PATH 変数をカスタマイズしてユーザーのホーム・ディレクトリーの下のサブディレクトリーを組み込む .profile ファイルの例を次に示します。

```
# Sample .profile file
export ENV=$HOME/.qshrc
export PATH=$PATH:$HOME/bin
```

環境ファイル

ENV 変数の展開によって指定されたファイルが存在している場合、**qsh** は対話式シェルを開始するときに、現行環境でこのファイルを実行します。 環境ファイルは、通常、別名を設定するため、関数を定義するため、または対話式シェル・セッション用のオプションを設定するために使用されます。

環境ファイルの例を次に示します。

```
# Sample environment file
PS1='$PWD'
```

注: qsh を開始する際には、ジョブ・レベルおよびシステム・レベルの環境変数も **qsh** で定義されます。例えば、以下の CL コマンドを使用してシステム全体の PATH 変数を確立できます。

```
ADDENVVAR ENVVAR(PATH) VALUE('/usr/bin:.:/QOpenSys/usr/bin') LEVEL(*SYS)
```

各国語サポート (NLS) に関する考慮事項

qsh の開始時には、コマンド処理の内部テーブルはジョブの CCSID に基づいて初期化されます。ファイルの読み込み時に、**qsh** はファイルの CCSID からジョブの CCSID にファイルを動的に変換します。

すべてを正常に実行するには、環境を下表に示すように構成する必要があります。

ロケールには、日付、時刻、数字、通貨の値の文字やフォーマットのソートと分類の方法を含め、言語と国または地域別の情報が収められています。ロケールを設定するには、LANG 環境変数をロケール・オブジェクトへのパス名に設定します。たとえば、US English のロケールを設定するには LANG 変数を次のように設定します。

```
LANG=/QSYS.LIB/EN_US.LOCALE
```

qsh を開始する前に LANG 環境変数を設定するのが最善です。ロケールが、以下の表にあるようにジョブの CCSID および言語 ID に対して有効でないと、ユーティリティーによっては正常に作動しない場合があります。

以下のような状況では、問題が生じる場合もあります。

- 対話式セッションで、ジョブの CCSID が表示装置 CCSID と異なる場合、**qsh** は特定の特殊文字を認識しません。
- スクリプト・ファイルの CCSID とジョブの CCSID との間の変換がサポートされていない場合は、ファイルを開くことができません。

サポートされている CCSID

サポートされている CCSID を次表に示します。項目は CCSID 番号順に並んでいます。この表にない CCSID の場合は、**qsh** はメッセージ 001-0072 を送信し、CCSID 37 で開始されたかのように実行されます。

サポートされている CCSID		
CCSID	**コード・ページ**	**説明**
00037	00037	米国、カナダ
00256	00256	国際 #1
00273	00273	ドイツ、オーストリア
00277	00277	デンマーク、ノルウェー
00278	00278	フィンランド、スウェーデン
00280	00280	イタリア
00284	00284	スペイン、ラテンアメリカ
00285	00285	英国
00297	00297	フランス
00424	00424	イスラエル (ヘブライ語)
00425	00425	アラビア語
00500	00500	ベルギー、カナダ、スイス
00833	00833	韓国語 (拡張単一バイト)
00836	00836	中国語 (簡体字、拡張単一バイト)
00838	00838	タイ (拡張)

サポートされている CCSID		
CCSID	**コード・ページ**	**説明**
00870	00870	Latin-2 (多国語)
00871	00871	アイスランド
00875	00875	ギリシャ
00880	00880	キリル文字 (多国語)
00905	00905	トルコ (拡張)
00918	00918	パキスタン
00933	00833, 00834	韓国語 (拡張混合)
00935	00836, 00837	中国語 (簡体字、拡張混合)
00937	00037, 00835	中国語 (繁体字、拡張混合)
00939	01027, 00300	日本 (英語拡張混合)
01025	01025	キリル文字 (多国語)
01026	01026	トルコ
01027	01027	日本 (Latin 拡張単一バイト)
01097	01097	ペルシア語
01112	01112	バルト語 (多国語)
01122	01122	エストニア語
01123	01123	ウクライナ (キリル文字)
01130	01130	ベトナム
01132	01132	ラオ語
01137	01137	デーバナーガリー
01140	01140	米国、カナダ (ユーロ)
01141	01141	ドイツ、オーストリア (ユーロ)
01142	01142	デンマーク、ノルウェー (ユーロ)
01143	01143	フィンランド、スウェーデン (ユーロ)
01144	01144	イタリア (ユーロ)
01145	01145	スペイン、ラテンアメリカ (ユーロ)
01146	01146	英国 (ユーロ)
01147	01147	フランス (ユーロ)
01148	01148	ベルギー、カナダ、スイス (ユーロ)
01149	01149	アイスランド (ユーロ)
01153	01153	Latin-2 (多国語ユーロ)
01154	01154	キリル文字 (多国語ユーロ)
01155	01155	トルコ (ユーロ)
01156	01156	バルト語 (多国語ユーロ)
01157	01157	エストニア (ユーロ)
01158	01158	キリル文字 (ウクライナ・ユーロ)
01160	01160	タイ (拡張ユーロ)
01164	01164	ベトナム (ユーロ)

サポートされている CCSID		
CCSID	**コード・ページ**	**説明**
01388	00836, 00837	中国語 (簡体字、ホスト・データ混合)
01399	01399, 00300	日本 (英語拡張混合ユーロ)
05035	01027, 00300	日本 (英語拡張混合)
05123	01399	日本 (英語拡張単一バイト・ユーロ)
09030	00838	タイ (拡張単一バイト)
13124	00836	中国語 (簡体字、ホスト・データ単一バイト)
28709	00037	中国語 (繁体字、拡張)

サポートされる言語

サポートされている言語を次表に示します。項目は言語順に並んでいます。言語フィールドには、CHGJOB CL コマンドの LANGID パラメーターに使用する値が括弧の中に示されています。国または地域フィールドには、CHGJOB CL コマンドの CNTRYID パラメーターに使用する値が括弧の中に示されています。

言語、国または地域、CCSID、ロケールの有効な組み合わせは、この表に記載されているものに限らないことに注意してください。たとえば、スペイン語は複数の国または地域で使われていますが、その項目は 1 つしかありません。

Qshell を実行する際には、LANGID、CNTRYID、CCSID ジョブ属性をこの表に記載されている値に設定し、LANG 環境変数を記載されたロケールに設定する必要があります。

サポートされる言語			
言語	**国または地域別 ID**	**CCSID**	**ロケール**
アルバニア語 (SQI)	アルバニア (AL)	00500	/QSYS.LIB/SQ_AL.LOCALE
アラビア語 (ARA)	アラビア語圏 (AA)	00425	/QSYS.LIB/AR_AA.LOCALE
ベルギー・オランダ語 (NLB)	ベルギー (BE)	00500	/QSYS.LIB/NL_BE.LOCALE
ベルギー・オランダ語ユーロ (NLB)	ベルギー (BE)	01148	/QSYS.LIB/NL_BE_E.LOCALE
ベルギー・フランス語 (FRB)	ベルギー (BE)	00500	/QSYS.LIB/FR_BE.LOCALE
ベルギー・フランス語ユーロ (FRB)	ベルギー (BE)	01148	/QSYS.LIB/FR_BE_E.LOCALE
ベルギー英語 (ENB)	ベルギー (BE)	00500	/QSYS.LIB/EN_BE.LOCALE
ブラジル・ポルトガル語 (PTB)	ブラジル (BR)	00037	/QSYS.LIB/PT_BR.LOCALE
ブルガリア語 (BGR)	ブルガリア (BG)	00037	/QSYS.LIB/BG_BG.LOCALE
カナダ・フランス語 (FRC)	カナダ (CA)	00500	/QSYS.LIB/FR_CA.LOCALE
クロアチア語 (HRV)	クロアチア (HR)	00870	/QSYS.LIB/HR_HR.LOCALE
チェコ語 (CSY)	チェコ共和国 (CZ)	00870	/QSYS.LIB/CS_CZ.LOCALE
デンマーク語 (DAN)	デンマーク (DK)	00277	/QSYS.LIB/DA_DK.LOCALE

サポートされる言語			
言語	**国または地域別 ID**	**CCSID**	**ロケール**
オランダ語 (NLD)	オランダ (NL)	00037	/QSYS.LIB/NL_NL.LOCALE
オランダ語ユーロ (NLD)	オランダ (NL)	01140	/QSYS.LIB/NL_NL_E.LOCALE
英大文字 (ENP)	米国 (US)	00037	/QSYS.LIB/EN_UPPER.LOCALE
エストニア語 (EST)	エストニア (EE)	01122	/QSYS.LIB/ET_EE.LOCALE
フィンランド語 (FIN)	フィンランド (FI)	00278	/QSYS.LIB/FI_FI.LOCALE
フィンランド語ユーロ (FIN)	フィンランド (FI)	01143	/QSYS.LIB/FI_FI_E.LOCALE
フランス語 (FRA)	フランス (FR)	00297	/QSYS.LIB/FR_FR.LOCALE
フランス語ユーロ (FRA)	フランス (FR)	01147	/QSYS.LIB/FR_FR_E.LOCALE
ドイツ語 (DEU)	ドイツ (DE)	00273	/QSYS.LIB/DE_DE.LOCALE
ドイツ語ユーロ (DEU)	ドイツ (DE)	01141	/QSYS.LIB/DE_DE_E.LOCALE
ギリシャ語 (ELL)	ギリシャ (GR)	00875	/QSYS.LIB/EL_GR.LOCALE
ヘブライ語 (HEB)	イスラエル (IL)	00424	/QSYS.LIB/IW_IL.LOCALE
ハンガリー語 (HUN)	ハンガリー (HU)	00870	/QSYS.LIB/HU_HU.LOCALE
アイスランド語 (ISL)	アイスランド (IS)	00871	/QSYS.LIB/IS_IS.LOCALE
イタリア語 (ITA)	イタリア (IT)	00280	/QSYS.LIB/IT_IT.LOCALE
イタリア語ユーロ (ITA)	イタリア (IT)	01144	/QSYS.LIB/IT_IT_E.LOCALE
日本語カタカナ (JPN)	日本 (JP)	05035	/QSYS.LIB/JA_5035.LOCALE
日本語 (JPN)	日本 (JP)	13488	/QSYS.LIB/JA_13488.LOCALE
韓国語 (KOR)	韓国 (KR)	00933	/QSYS.LIB/KO_KR.LOCALE
ラトビア語 (LVA)	ラトビア (LV)	01112	/QSYS.LIB/LV_LV.LOCALE
リトアニア語 (LTU)	リトアニア (LT)	01112	/QSYS.LIB/LT_LT.LOCALE
マケドニア語 (MKD)	マケドニア (MK)	01025	/QSYS.LIB/MK_MK.LOCALE
ノルウェー語 (NOR)	ノルウェー (NO)	00277	/QSYS.LIB/NO_NO.LOCALE
ポーランド語 (PLK)	ポーランド (PL)	00870	/QSYS.LIB/PL_PL.LOCALE
ポルトガル語 (PTG)	ポルトガル (PT)	00037	/QSYS.LIB/PT_PT.LOCALE
ポルトガル語ユーロ (PTG)	ポルトガル (PT)	01140	/QSYS.LIB/PT_PT_E.LOCALE
ルーマニア語 (ROM)	ルーマニア (RO)	00870	/QSYS.LIB/RO_RO.LOCALE
ロシア語 (RUS)	ロシア (RU)	01025	/QSYS.LIB/RU_RU.LOCALE
セルビア語キリル文字 (SRB)	セルビア (SQ)	01025	/QSYS.LIB/SR_SP.LOCALE
セルビア語ローマ字 (SRL)	セルビア (SQ)	00870	/QSYS.LIB/SH_SP.LOCALE
中国語 (簡体字) (CHS)	中国 (CN)	00935	/QSYS.LIB/ZH_CN.LOCALE

サポートされる言語			
言語	**国または地域別 ID**	**CCSID**	**ロケール**
スロバキア語 (SKY)	スロバキア (SK)	00870	/QSYS.LIB/SK_SK.LOCALE
スロベニア語 (SLO)	スロベニア (SI)	00870	/QSYS.LIB/SL_SI.LOCALE
スペイン語 (ESP)	スペイン (ES)	00284	/QSYS.LIB/ES_ES.LOCALE
スペイン語ユーロ (ESP)	スペイン (ES)	01145	/QSYS.LIB/ ES_ES_E.LOCALE
スウェーデン語 (SVE)	スウェーデン (SE)	00278	/QSYS.LIB/SV_SE.LOCALE
スイス・フランス語 (FRS)	スイス (CH)	00500	/QSYS.LIB/FR_CH.LOCALE
スイス・ドイツ語 (DES)	スイス (CH)	00500	/QSYS.LIB/DE_CH.LOCALE
タイ語 (THA)	タイ (TH)	00838	/QSYS.LIB/TH_TH.LOCALE
トルコ語 (TRK)	トルコ (TR)	00905	/QSYS.LIB/TR_TR.LOCALE
ウクライナ語 (UKR)	ウクライナ (UA)	01025	/QSYS.LIB/UK_UA.LOCALE
英国英語 (ENG)	英国 (GB)	00285	/QSYS.LIB/EN_GB.LOCALE
米国英語 (ENU)	米国 (US)	00037	/QSYS.LIB/EN_US.LOCALE

関連情報:

「iSeries Globalization」の「IBM Code Pages」

パフォーマンスに関する考慮事項

システムで可能な最高のパフォーマンスが得られるよう Qshell を構成します。

以下のヒントは、**qsh** を使用するときに、パフォーマンスを向上させるのに役立ちます。

- **PS1** 変数の値の中ではコマンド置換を使用できません。これを使用すると、<enter> キーを押すたびに新しいプロセスが開始されることになります。
- cat の代わりに入力リダイレクトを使用してください。たとえば、次のようなコマンドがあるとします。

  ```
  cat myfile | grep Hello
  ```

  これは次のコマンドで置き換えることができます。

  ```
  grep Hello < myfile
  ```
- できるだけ組み込みユーティリティーを使用してください。組み込みユーティリティーは現行プロセスの中で実行されるからです。
- **SHELL** 変数を未設定のままにします。スクリプト・ファイルの最初の行に #! が置かれていない場合は、**qsh** の起動時にそのスクリプトが実行されます。

独自のユーティリティーの開発

任意の言語を使って独自のユーティリティーを開発することができます。ただし、ILE/C、ILE/C++、および Java を使用する時に最高の実行時サポートが得られます。

ILE/C または ILE/C++ プログラムの作成時に、ユーティリティー・プログラム内のすべてのモジュールを作成するときは、統合ファイル・システム入出力を使用してください。

ユーティリティーは、入力を標準入力または記述子 0 から読み取り、出力を標準出力または記述子 1 に書き込み、エラーを標準エラーまたは記述子 2 に書き込みます。

入出力に ILE/C または ILE/C++ 標準ファイルを使用するユーティリティー・プログラムの場合、 **qsh** コマンド行または QCMD コマンド行からユーティリティーを実行できます。記述子 0、1、および 2 から直接読み書きするユーティリティーの場合は、Qshell コマンド行からでなければそのユーティリティーを実行できません。

Qshell インタープリターでのファイルの編集

EDTF CL コマンドを使用すると、任意のファイル・システムにあるファイルを編集できます。これは、ストリーム・ファイルまたはデータベース・ファイルを編集するための原始ステートメント入力ユーティリティー (SEU) によく似たエディターです。

EDTF CL コマンドを使用すると、任意のファイル・システムにあるファイルを編集できます。これは、ストリーム・ファイルまたはデータベース・ファイルを編集するための原始ステートメント入力ユーティリティー (SEU) によく似たエディターです。 DSPF CL コマンドを使って、ストリーム・ファイルまたはデータベース・ファイルを表示することもできます。

これに代わるもう 1 つの方法として、System i® ナビゲーター を使用してサーバーに接続し、クライアント上でエディターを実行してファイルを編集することもできます。そのファイルを ASCII で保管し、しかも Qshell で使用することができます。

シェル・スクリプトは、シェル・コマンドの入ったテキスト・ファイルに過ぎません。シェル・スクリプトの保管には、正しいファイル・システムを使うことが大切です。シェル・スクリプトはストリーム・データであり、「ルート」ファイル・システムに保管しなければなりません。シェル・スクリプトを QSYS.LIB ファイル・システムのソース物理ファイル内に保管することはできますが、そうすると、シェル・スクリプトに要するストレージが大きくなり、実行が遅くなります。

その他のインタープリターとの相違点

qsh には他の標準シェル・インタープリターと互換性がありますが、いくつか相違点があります。

- **<>** リダイレクト演算子はサポートされていません。
- コマンド・ヒストリー・リスト、**HISTSIZE** 変数と **HISTFILE** 変数、または **fc** (または **hist**) 組み込みユーティリティーはサポートされていません。その代わりとして、QSH CL コマンドは、コマンド検索をサポートしています。
- コマンド行編集および **EDITOR** 変数はサポートされていません。
- **MAIL** 変数、**MAILCHECK** 変数、および **MAILPATH** 変数はサポートされていません。
- ジョブ制御はサポートされていません。 i5/OS には、フォアグラウンド・プロセス・グループまたはバックグラウンド・プロセス・グループという概念はありません。これは、同じ端末から同時に複数のジョブが読み取りを実行している可能性があるということです。 **qsh** は、次のものをサポートしていません。
  - **fg** または **bg** 組み込みユーティリティー。
  - *Suspend (中断)* キー (通常は <ctrl>z) を使用して SIGTSTP シグナルをフォアグラウンド・プロセス・グループに送信する機能。
  - *Stop (停止)* キー (通常は <ctrl>s) を使用して SIGSTOP シグナルをフォアグラウンド・プロセス・グループに送信する機能。
  - *Restart (再始動)* キー (通常は <ctrl>q) を使用して SIGCONT シグナルをフォアグラウンド・プロセス・グループに送信する機能。

– *Interrupt (割り込み)* キー (通常は <ctrl>c) を使用して SIGINT シグナルをフォアグラウンド・プロセス・グループに送信する機能。 その代わりとして、対話式シェル・セッションから SysReq 2 を使って、SIGINT シグナルをシェル・インタープリター・プロセスおよび現在実行中の任意の子プロセスに送信することができます。

- *End-of-file (ファイル終わり)* キー (通常は <ctrl>d) はサポートされていません。ただし、here ドキュメントを使って、コマンド行から入力されたテキストをユーティリティーの標準入力にリダイレクトすることができます。
- プログラムを呼び出すときに、コマンドに渡せるパラメーターの最大数には限度があります。プログラムが V5R3 より前のリリースで構築された場合、パラメーターの限度は 255 です。プログラムが V5R3 またはそれ以降のリリースで構築された場合、パラメーターの限度は 65535 です。
- 大文字小文字を区別しないファイル・システムでパス名拡張を使用するときは、パターン内では大文字を使用する必要があります。 たとえば、QSHELL ライブラリー内のプログラム・オブジェクトをすべて表示するには、次のコマンドを使ってください。

```
ls /qsys.lib/qshell.lib/*.PGM.
```

ユーティリティー

以下のすべてのユーティリティーのアルファベット順リストを使用して、目的のユーティリティーに直接進んでください。

全ユーティリティーのリスト

A B C D E F G H I J K L M N O P Q R S T U W X Z

A	
ajar	代替 Java アーカイブ・ツール
alias	別名を定義または表示する
appletviewer	web ブラウザーを使わずにアプレットを実行する
attr	ファイルの属性を取得または設定する
B	
basename	パス名のディレクトリー以外の部分を戻す
break	for、while、または until ループを終了する
builtin	シェル組み込みユーティリティーを実行する
C	
cat	ファイルを連結して出力する
catsplf	スプール・ファイルを連結して出力する
cd	作業ディレクトリーを変更する
chgrp	ファイル・グループの許可を変更する
chmod	ファイル・モード (許可) を変更する
chown	ファイルの所有権を変更する
clrtmp	/tmp ディレクトリーをクリアする
cmp	2 つのファイルを比較する
colon (:)	ヌル・ユーティリティー
command	単純コマンドを実行する

compress	データを圧縮する
continue	for、while、または until ループを継続する
cp	ファイルをコピーする
cut	ファイルの各行から選択したフィールドを切り取る
D	
dataq	i5/OS データ待ち行列からメッセージを送受信する
datarea	i5/OS データ域の読み書きをする
date	日付と時刻を書き込む
db2profc	DB2® SQLJ プロファイル・カスタマイザー
db2profp	SQLJ プロファイルの DB2 カスタマイズ済みバージョンを印刷する
declare	変数を宣言し、属性設定をする
dirname	パス名のディレクトリー部分を戻す
dot (.)	現行環境でコマンドを実行する
dspmsg	メッセージ・カタログのメッセージを表示する
E	
echo	引数を標準出力に書き込む
egrep	ファイル内の拡張正規表現パターンを検索する
env	コマンド呼び出し用の環境を設定する
eval	引数を連結してコマンドを作成する
exec	コマンドを実行し、記述子をオープン、クローズ、またはコピーする
exit	シェルを終了する
export	変数に対してエクスポート属性を設定する
expr	引数を式として評価する
extcheck	Java アーカイブ競合を検出する
F	
false	偽の値を戻す
fgrep	ファイル内の固定ストリング・パターンを検索する
file	ファイル・タイプを判別する
find	ファイルを検索する
G	
gencat	フォーマットされたメッセージ・カタログを生成する
getconf	構成値を取得する
getjobid	ジョブ情報を表示する
getopts	ユーティリティー・オプションを解析する
grep	パターンに対応するファイルを検索する

H	
hash	ユーティリティー・ロケーションを記憶または報告する
head	ファイルの先頭部分をコピーする
help	組み込みユーティリティーの情報を表示する
hostname	現在のホスト・システムの名前を表示する
I	
iconv	ある CCSID から別の CCSID に文字を変換する
id	ユーザー ID を戻す
ipcrm	プロセス間通信 ID を削除する
ipcs	プロセス間通信状況を報告する
J	
jar	Java ファイルをアーカイブする
jarsigner	Java アーカイブ署名と検証
java	Java インタープリターを実行する
javac	Java プログラムをコンパイルする
javadoc	Java 文書を生成する
javah	C ヘッダーまたはスタブ・ファイルを生成する
javakey	Java セキュリティー・キーおよび証明書を管理する
javap	コンパイル済みの Java プログラムを逆アセンブルする
jobs	現行セッションのジョブの状況を表示する
K	
kdestroy	Kerberos 信任状キャッシュを破棄する
keytab	Kerberos キー・テーブル・ファイルを管理する
keytool	キーと証明書の管理ツール
kill	プロセスを終了するかまたはシグナルを送る
kinit	Kerberos 発券許可証を取得または更新する
klist	Kerberos 信任状キャッシュまたはキー・テーブル・ファイルの内容を表示する
ksetup	Kerberos レルム用の LDAP ディレクトリーにある Kerberos サービス・エントリーを管理する
L	
ldapadd	LDAP 項目の追加ツール
ldapchangepwd	LDAP パスワードの変更ツール
ldapdelete	LDAP 項目の削除ツール
ldapdiff	LDAP 複製の同期の比較ツール
ldapexop	LDAP 操作の拡張ツール
ldapmodify	LDAP 項目の変更ツール
ldapmodrdn	LDAP 相対識別名 (RDN®) の変更ツール

ldapsearch	LDAP サーバーの検索ツール
let	算術式を評価する
liblist	ライブラリー・リストを管理する
ln	ファイルをリンクする
local	関数内でローカル変数を割り当てる
locale	ロケール特定情報を入手する
logger	メッセージをログに記録する
logname	ユーザーのログイン名を戻す
ls	ディレクトリーの内容をリスト表示する
M	
mkdir	ディレクトリーを作成する
mkfifo	FIFO 特殊ファイルを作成する
mv	ファイルを移動する
N	
native2ascii	ネイティブ文字を ASCII に変換する
nohup	ユーティリティーをハングアップしないように実行する
O	
od	様々なフォーマットのファイルをダンプする
P	
pax	ポータブル・アーカイブを交換する
policytool	ポリシー・ファイルの作成および管理ツール
pr	ファイルを出力する
print	出力を書き出す
printenv	環境変数の値を表示する
printf	形式化された出力を書き出す
profconv	SQLJ 直列化プロファイル・インスタンスを Java クラスに変換する
profdb	SQLJ プロファイル監査プログラム・インストーラー
profp	SQLJ プロファイルを印刷する
ps	プロセス状況を表示する
pwd	作業ディレクトリー名を戻す
pwdx	展開された作業ディレクトリーを戻す
Q	
qsh	Qshell コマンド言語インタープリター
R	
read	標準入力から行を読み取る

readonly	変数に対して読み取り専用属性を設定する
return	関数から戻る
rexec	リモート・コマンドを実行する
rexx	REXX プロシージャーを実行する
Rfile	レコード・ファイルの読み書きをする
rm	ディレクトリー項目を削除する
rmdir	ディレクトリーを削除する
rmic	Java RMI スタブをコンパイルする
rmid	Java RMI 活動化システム
rmiregistry	リモート・オブジェクト・レジストリーを開始する
S	
sed	ストリーム・エディター
serialver	シリアル・バージョンを戻す
set	オプションおよび定位置パラメーターを設定または設定解除する
setccsid	ファイルの CCSID 属性を設定する
sh	Qshell コマンド言語インタープリター
shift	定位置パラメーターをシフトする
sleep	呼び出しを一定期間中断する
sort	テキスト・ファイルをソート、マージ、またはシーケンス検査する
source	現行環境でコマンドを実行する
split	ファイルを分割する
sqlj	Structured Query Language for Java (SQLJ) 変換プログラム
system	CL コマンドを実行する
sysval	システム値またはネットワーク属性を検索する
T	
tail	ファイルの末尾部分をコピーする
tar	ファイル・アーカイバー
tee	標準入力を複製する
test	式を評価する
tnameserv	ネーム・サービス
touch	ファイルのアクセス時刻および変更時刻を変更する
tr	文字を変換する
trap	シグナルをトラップする
true	真の値を戻す
type	コマンドのタイプを検索する
typeset	変数を宣言し、属性設定をする

U	
ulimit	リソース限界を設定または表示する
umask	ファイル・モード作成マスクを入手または設定する
unalias	別名定義を削除する
uname	システム名を戻す
uncompress	圧縮データを圧縮解除する
uniq	ファイル内の繰り返し行を報告または抽出する
unset	変数および関数の、値および属性を設定解除する
W	
wait	プロセスの完了を待つ
wc	ワード、行、およびバイト/文字をカウントする
whence	コマンドの解釈の仕方を判断する
X	
xargs	引数リストを作成してユーティリティーを起動する
Z	
zcat	データを拡張および連結する

別名定義用のユーティリティー

別名定義用のユーティリティーを表示します。

alias - 別名を定義または表示する

alias ユーティリティーは、指定された *value* を持つ別名 *name* を定義します。 *name* だけを指定すると、**qsh** は別名の名前および値を表示します。

構文

alias [-p] [*name* [=*value*] ...]

説明

引数を指定しなかった場合は、**qsh** はすべての別名とその値のリストを表示します。

qsh は次のデフォルトの別名を定義します。

- float='declare -E'
- functions='declare -f'
- integer='declare -i'

オプション

-p 出力の各行の前に "alias" という語を付けます。したがって、各行をそのまま再入力に使えます。

オペランド

name には、現行の環境での別名を指定します。 *value* も同時に指定した場合は、別名の値が更新されます。

終了状況

- 0 正常終了。
- >0 不成功の場合。値は、別名でない *names* の数を示します。

例

1. ディレクトリーの内容を表示する別名を定義する場合:

   ```
   alias ll='ls -l'
   ```

2. 別名 11 の値を表示する場合:

   ```
   alias ll
   ```

3. 現在定義されているすべての別名の値を表示する場合:

   ```
   alias - 別名を定義または表示する
   ```

関連タスク:

『unalias - 別名定義を削除する』
unalias を使うと、定義済みの別名のリストから *names* を削除できます。

unalias - 別名定義を削除する

unalias を使うと、定義済みの別名のリストから *names* を削除できます。

構文

unalias *name* ...

unalias -a

説明

オプション

-a すべての別名を削除します。

オペランド

各 *name* は定義済みの別名です。

終了状況

- 0 正常終了。
- >0 不成功の場合。値は、別名でない *names* の数を示します。

例

別名 ll を削除する場合: **unalias** ll

関連タスク:

45 ページの『alias - 別名を定義または表示する』
alias ユーティリティーは、指定された *value* を持つ別名 *name* を定義します。 *name* だけを指定すると、**qsh** は別名の名前および値を表示します。

コマンド実行用のユーティリティー

コマンド実行用のユーティリティーを表示します。

builtin - シェル組み込みユーティリティーを実行する

構文

builtin [*utility* [*argument* ...]]

説明

builtin ユーティリティーは、指定された *arguments* を使用して、シェルに組み込まれている *utility* を実行します。 **builtin** を使用すると、組み込みユーティリティーを、同じ名前のシェル関数から実行することができます。

オペランド

utility は、シェル組み込みユーティリティーの名前です。ユーティリティーのタイプを判別するために、command、type、または whence を使用することができます。

終了状況

- *utility* の終了状況
- 1 *utility* が組み込みユーティリティーではない場合。

関連概念:

25 ページの『複合コマンド』
複合コマンドは、他のコマンドの制御フローを提供します。 複合コマンドは予約語で始まり、それに対応する予約語で終わります。

関連タスク:

『command - 単純コマンドを実行する』

59 ページの『type - コマンドのタイプを検索する』

60 ページの『whence - コマンドの解釈の仕方を判断する』

52 ページの『help - 組み込みユーティリティーの情報を表示する』

command - 単純コマンドを実行する

構文

command [**-p**] *command_name* [*argument* ...]

command [**-vV**] *command_name*

説明

command を使うと、指定された *arguments* を使用して *command_name* を実行し、検索順序から関数を除外することができます。 *command_name* が特殊組み込みユーティリティーの場合は、通常の組み込みユーティリティーとして取り扱われます。 それ以外の場合は、**command** の実行結果は **command** を指定しなかった場合と同じです。

command -v は **whence** と同じであり、**command -V** は **whence -v** と同じです。

オプション

-p **PATH** 変数のデフォルト値を使用してコマンド検索を実行します。そうすれば、すべての標準ユーティリティーを確実に見つけ出すことができます。

-v 現行環境で **qsh** が *command_name* を呼び出すために使用するパス名またはコマンドを示すストリングを表示します。

-V 現行環境で **qsh** が *command_name* をどのように解釈するかを示すストリングを表示します。

オペランド

command_name は、現行環境の中にあるユーティリティーです。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. 特殊組み込みユーティリティー export を通常の組み込みユーティリティーとして実行する場合: **command** export ALPHA
2. ls ユーティリティーを呼び出すために使用されるパス名を表示する場合: **command** -v ls
3. 予約語 for がどのように解釈されるかを表示する場合: **command** -V for

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

47 ページの『builtin - シェル組み込みユーティリティーを実行する』
『ドット (.) - 現行環境でコマンドを実行する』
50 ページの『eval - 引数を連結してコマンドを作成する』
60 ページの『whence - コマンドの解釈の仕方を判断する』
52 ページの『help - 組み込みユーティリティーの情報を表示する』
52 ページの『nohup - ユーティリティーをハングアップしないように実行する』
59 ページの『type - コマンドのタイプを検索する』
57 ページの『source - 現行環境でコマンドを実行する』

ドット (.) - 現行環境でコマンドを実行する

構文

. *name* [*argument* ...]

説明

ドットを使うと、スクリプトまたは関数を現行環境で実行できます。

オプション

なし。

オペランド

name が関数を参照している場合は、**qsh** はその関数を現行環境で実行します。 それ以外の場合は、**qsh** は、**PATH** 変数によって指定されている検索パスを使って *name* を検索します。 *name* が見つかると、**qsh** はそのファイルの内容を読み取って、そこに指定されたコマンドを現行環境で実行します。

arguments を指定した場合は、*name* の実行中に定位置パラメーターがそれらの引数で置換されます。 引数を指定しなかった場合は、定位置パラメーターは変更されません。

終了状況

name に指定された最後のコマンドの終了状況。

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

47 ページの『command - 単純コマンドを実行する』

50 ページの『eval - 引数を連結してコマンドを作成する』

60 ページの『whence - コマンドの解釈の仕方を判断する』

env - コマンド呼び出し用の環境を設定する

構文

env [-i | -] [*name=value* ...] [*utility* [*argument* ...]]

説明

env ユーティリティーは、現行の環境を取得し、その環境を引数に従って変更し、その後指定された *utility* を呼び出します。 *utility* には、すべての *arguments* が渡されます。 *utility* が指定されない場合、1 行ごとに 1 つの *name=value* の形式で結果の環境が標準出力に書き出されます。

オプション

- コマンドで指定された通りの環境で *utility* を呼び出します。継承された環境は完全に無視されます。

-i '-' と同じです。

オペランド

name=value
これはランタイム環境を変更し、*utility* が呼び出される前に、継承された環境に挿入されます。

utility 呼び出されるコマンドまたはユーティリティーの名前。

argument
呼び出されるコマンドまたはユーティリティーに渡すストリング。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

52 ページの『nohup - ユーティリティーをハングアップしないように実行する』

166 ページの『printenv - 環境変数の値を表示する』

eval - 引数を連結してコマンドを作成する

構文

eval [*argument* ...]

説明

eval を使うと、個々の引数の間を <space> で区切って *arguments* をすべて連結し、コマンドを作成することができます。 **qsh** は、作成されたコマンドを読み取って実行します。

オプション

なし。

オペランド

各 *argument* は 2 回展開されます。1 回めはコマンドを作成するため、2 回めは作成されたコマンドを実行するときです。

終了状況

作成されたコマンドの戻り状況。

関連概念:

『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

47 ページの『command - 単純コマンドを実行する』
48 ページの『ドット (.) - 現行環境でコマンドを実行する』
57 ページの『source - 現行環境でコマンドを実行する』
60 ページの『whence - コマンドの解釈の仕方を判断する』
60 ページの『xargs - 引数リストを作成してユーティリティーを起動する』

exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする

構文

exec [**-c**] [*command* [*argument* ...]]

説明

exec ユーティリティーは、新しいプロセスを作成せずに、**qsh** を *command* で置換します。指定された *arguments* は、*command* に対する引数です。リダイレクトはすべて現行環境に影響を及ぼします。

command を指定しなかった場合、リダイレクトは現行環境内部で処理されます。リダイレクトによってオープンされたファイル記述子のうち、2 より大きい記述子は、**qsh** が別のプログラムを呼び出すときに継承されません。

オプション

-c 一連の環境変数を空にして *command* を実行します。

オペランド

各 *argument* は、*command* の定位置パラメーターの順序に従って割り当てられます。

終了状況

command を指定しなかった場合は 0。それ以外の場合は、**qsh** に戻りません。

例

1. 記述子 5 で読み取り用にファイルをオープンする場合:

```
exec 5<$HOME/input
```

2. 記述子 5 をクローズする場合:

```
exec 5<&-
```

関連概念:

54 ページの『rexec - リモート・コマンドを実行する』

関連タスク:

47 ページの『command - 単純コマンドを実行する』
48 ページの『ドット (.) - 現行環境でコマンドを実行する』
50 ページの『eval - 引数を連結してコマンドを作成する』
52 ページの『nohup - ユーティリティーをハングアップしないように実行する』
140 ページの『print - 出力を書き出す』
142 ページの『read - 標準入力から行を読み取る』
57 ページの『source - 現行環境でコマンドを実行する』

exit - シェルを終了する

構文

exit [*n*]

説明

exit を使うと、シェルを終了して、**qsh** を呼び出したプログラムに戻ることができます。

オプション

なし。

オペランド

n の値は、0 - 255 の整数です。

終了状況

n を指定した場合は、*n*。 n を指定しなかった場合は、直前のコマンドの終了状況。

関連タスク:

174 ページの『return - 関数から戻る』
53 ページの『qsh - Qshell コマンド言語インタープリター』

help - 組み込みユーティリティーの情報を表示する

構文

help [*utility* ...]

説明

help ユーティリティーは、指定された組み込み *utility* の使用法メッセージを表示します。引数が指定されていない場合、**help** はすべての組み込みユーティリティーのリストを表示します。

オペランド

utility は、シェル組み込みユーティリティーの名前です。

終了状況

- 0 正常終了。
- >0 *utility* が組み込みユーティリティーではない場合。

関連タスク:

47 ページの『builtin - シェル組み込みユーティリティーを実行する』
47 ページの『command - 単純コマンドを実行する』
59 ページの『type - コマンドのタイプを検索する』
60 ページの『whence - コマンドの解釈の仕方を判断する』

nohup - ユーティリティーをハングアップしないように実行する

構文

nohup [**-C** *ccsid*] *utility* [*argument* ...]

説明

nohup ユーティリティーは、指定された *arguments* を使用して、指定された *utility* を実行します。 *utility* が呼び出されるとき、SIGHUP シグナルを無視するように設定されます。 **nohup** を使用すると、Qshell セッションの終了後であっても、*utility* を実行できます。

標準出力が端末装置の場合、*utility* からその標準出力に書き出されるすべての出力は、現行ディレクトリーの nohup.out ファイルに追加されます。追加するためにそのファイルを作成できない、またはオープンできない場合、すべての出力は $HOME/nohup.out ファイルに追加されます。いずれのファイルも作成できない、またはオープンできない場合、*utility* は実行されません。デフォルトでは、nohup.out ファイルの読み書きを許可されているのは所有者だけです。

標準エラーが端末装置の場合、*utility* からその標準エラーに書き出されるすべての出力は、標準出力と同じ記述子にリダイレクトされます。

オプション

-C *ccsid*
: nohup.out ファイルは指定された *ccsid* で作成されます。そのファイルに書き込まれるすべてのデータは、ジョブの CCSID から、指定された *ccsid* へと変換されます。このオプションは、QIBM_CCSID 環境変数の値をオーバーライドします。

オペランド

utility は、現行環境内の正規のユーティリティーの名前です。

環境変数

nohup は、次の環境変数の影響を受けます。

QIBM_CCSID
この環境変数の値は、nohup.out ファイルを作成するときに使用される CCSID です。そのファイルに書き込まれるすべてのデータは、ジョブの CCSID から、指定された CCSID へ変換されます。

終了状況

- 126 *utility* は検出されたが実行できなかった
- 127 *utility* が検出されなかった、または **nohup** にエラーがあった
- その他の場合は、*utility* の終了状況

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

47 ページの『command - 単純コマンドを実行する』

49 ページの『env - コマンド呼び出し用の環境を設定する』

qsh - Qshell コマンド言語インタープリター

構文

qsh [-abCefFijlmntuvx] [-o *option*] *command_file arg ...*

qsh -c [-abCefFijlmntuvx] [-o *option*] *command_string*

qsh -s [-abCefFijlmntuvx] [-o *option*] *arg ...*

説明

qsh ユーティリティーは Qshell コマンド言語インタープリターです。最初の構文形式の **qsh** は、指定された *command_file* を読み取ってから、そのファイルに入っているコマンドを実行します。 2 番目の構文形式の **qsh** は、指定された *command_string* を実行してから終了します。 3 番目の構文形式の **qsh** は、標準入力からコマンドを読み取ります。

オプション

a、**b**、**C**、**e**、**f**、**F**、**j**、**l**、**m**、**n**、**-o** *option*、**t**、**u**、**v**、 および **x** オプションについては、set - オプションおよび定位置パラメーターを設定または設定解除するの項に説明されています。

-c *command_string* に指定されているコマンドを実行してから終了します。

-i シェルは対話式です。オペランドがない場合に -c オプションが指定されないと、デフォルトで -s オプションが設定されます。

-s 標準入力からコマンドが読み取られます。オペランドがない場合に **-c** オプションが指定されないと、デフォルトで **-s** オプションが設定されます。

オペランド

command_file は、Qshell コマンドの入った正規ファイルのパス名です。そのパス名にスラッシュ (/) 文字が入っていないと、 **qsh** は **PATH** 変数を使って *command_file* を検索します。特殊パラメーター 0 は、*command_file* の値に設定されます。各 *arg* は、定位置パラメーターです。

command_string は、複合コマンドを含む任意の Qshell コマンドです。

終了状況

- 0 正常終了。
- 1 不成功。
- 2 スクリプト内でエラー発生。
- 3 ルート・シェル内で予期しない例外が存在した。
- 4 ルート・シェルの例外ハンドラー内で予期しない例外が存在した。
- 5 子シェル内で予期しない例外が存在した。
- 6 子シェルの例外ハンドラー内で予期しない例外が存在した。
- 7 記述子 0 が利用不能。
- 8 記述子 1 が利用不能。
- 9 記述子 2 が利用不能。
- 10 メッセージ・カタログのオープン時にエラー。
- 11 - 125 不成功。
- 126 コマンドは検出されたが呼び出せなかった。
- 127 コマンドを検出できなかった。
- >128 コマンドがシグナルによって終了。値は、128 にシグナル番号を加えた値です。

関連概念:

7 ページの『コマンド言語』
シェル・スクリプトを書き込んでいるか、経験を積んだシェル・ユーザーである場合には、この詳細な参照情報から始めると良いでしょう。

関連タスク:

51 ページの『exit - シェルを終了する』

168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』

rexec - リモート・コマンドを実行する

構文

rexec [-C *ccsid* **] [-p** *password***] [-u** *user***] [-i]** *host command*

説明

rexec ユーティリティーは、*host* で指定されたリモート・システム上で、指定された *command* を実行します。そのコマンドを処理するためには、そのリモート・システムが rexec サーバーを実行している必要があります。デフォルトで **rexec** は、リモート・システムの有効なユーザー名とパスワードを入力するプロンプトを出します。ユーザー名とパスワードがリモート・システムに送信されるとき、それらは暗号化されません。

リモート・システムの *command* で生成された標準出力および標準エラーは、ローカル・システムの標準出力および標準エラーに書き込まれます。ローカル・システムの標準入力から読み込まれたデータはいずれも、**-i** オプションが指定されていない場合、そのリモート・システム上で実行している *command* の標準入力に送信されます。

デフォルトでは、リモート・システムとの間でやり取りされる送信データは、CCSID 819 でエンコードされます。データのエンコード時に使用される CCSID は、**-C** オプションあるいは QIBM_CCSID 変数のいずれかを使用して指定することができます。もし CCSID 値が 65535 であれば、データの変換は行われません。

オプション

-C *ccsid*
: リモート・システムとの間でやり取りされる送信データを、指定された *ccsid* にエンコードします。このオプションは、QIBM_CCSID 環境変数の値をオーバーライドします。

-i
: ローカル・システム上の標準入力を無視します。

-p *password*
: *host* のユーザーのパスワード。

-u *user*
: *host* の有効なユーザー名。

オペランド

host は、コマンドを実行しているリモート・システムの名前です。 *command* は、リモート・システム上で実行中の rexec サーバーにより解釈されるコマンド・ストリングです。

環境変数

rexec は、次の環境変数を使用します。

QIBM_CCSID
: この変数の値は、リモート・システムとの間でやり取りされる送信データをエンコードする時に使用する CCSID です。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

rexx - REXX プロシージャーを実行する

構文

rexx [**-c** *cmdenv*] [**-t** *type*] *path* [*arg* ...]

説明

rexx ユーティリティーは、*path* で指定された REXX プロシージャーを、指定された *arguments* を使って実行します。

REXX インタープリターは、標準入力から REXX コマンドを読み取ることができません。 REXX インタープリターは、QSYS.LIB ファイル・システムにあるデータベース・ファイルのメンバーに保管されている REXX プロシージャーしか実行できません。 REXX インタープリターの対話式デバッグ機能は、**rexx** ユーティリティーではサポートされていません。

/QSYS.LIB/QSHELL.LIB/QZSHSHRX.PGM プログラムは、REXX プロシージャー用の Qshell コマンド環境を実装しています。 Qshell コマンド環境では、REXX 戻りコードおよび条件を以下のように設定しています。

- シェル・コマンドが戻り状況ゼロで正常終了したときには、REXX 戻りコードはゼロに設定され、条件は発生しません。
- シェル・コマンドがゼロ以外の戻り状況で通常終了した場合、REXX 戻りコードはその戻り状況の値に設定され、ERROR 条件が発生します。
- シェル・コマンドがシグナルによって終了した場合、REXX 戻りコードはシグナル番号 + 128 に設定され、FAILURE 条件が発生します。
- シェル・コマンドが例外によって終了した場合、REXX 戻りコードは wait() からの例外番号に設定され、FAILURE 条件が発生します。

オプション

-c *cmdenv*
REXX プロシージャー用のコマンドを処理するコマンド環境プログラムを設定します。オプションが指定されていない場合、デフォルト値は *command* です。 *cmdenv* は、次のいずれかの値です。
- *command* (i5/OS CL コマンド環境の場合)
- *cpicomm* (通信コマンド環境用共通プログラミング・インターフェースの場合)
- *execsql* (構造化照会言語 (SQL) コマンド環境の場合)
- *qsh* (Qshell コマンド環境の場合)
- *path* (コマンド環境プログラムへのパスを指定する場合。パスは、QSYS.LIB ファイル・システムのプログラムを指定する必要があります。)

-t *type* REXX プロシージャーのトレースを制御します。オプションが指定されていない場合、デフォルト値は*normal* です。 *type* は、次のいずれかの値です。
- *all* (処理の前にすべての文節をトレースする場合)
- *commands* (処理およびエラー戻りコードを表示する前にホスト・コマンドをトレースする場合)
- *error* (処理の結果としてエラー戻りコードが戻ってきた後にホスト・コマンドをトレースする場合)
- *failure* (処理が失敗して戻りコードが戻ってきた後にホスト・コマンドをトレースする場合)
- *intermediates* (処理の前にすべての文節をトレースすると共に、式の評価中に中間結果を表示する場合)
- *labels* (処理中にラベルをトレースする場合)
- *normal* (処理が失敗した後にホスト・コマンドをトレースする場合)
- *off* (すべてのトレースをオフにする場合)
- *results* (処理の前にすべての文節をトレースする場合)

オペランド

path は、REXX プロシージャーのパス名です。 i5/OS では、REXX プロシージャーは QSYS.LIB ファイル・システムにしか格納できません。

終了状況

- 0 正常終了。
- 1 REXX プロシージャーの実行におけるエラー発生。
- >1 不成功の場合。

関連タスク:

『system - CL コマンドを実行する』

関連情報:

REXX マニュアル

source - 現行環境でコマンドを実行する

構文

source *name* [*argument* ...]

説明

sourceを使うと、スクリプトまたは関数を現行環境で実行できます。これは dot ユーティリティーと同じ働きをします。

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

47 ページの『command - 単純コマンドを実行する』

50 ページの『eval - 引数を連結してコマンドを作成する』

system - CL コマンドを実行する

構文

system [-iKknpqsv] *CLcommand* [*arg* ...]

説明

system ユーティリティーは、CL コマンドを実行します。 *CLcommand* で生成されたスプール・ファイル出力はすべて標準出力に書き込まれます。デフォルトでは、そのスプール・ファイルは書き込み後に削除され、 **system** を実行しているジョブのジョブ・ログも削除されます。

CLcommand で生成されたメッセージはすべて標準エラーに書き込まれます。デフォルトでは、*CLcommand* で生成されたメッセージは、すべて次のフォーマットで書き出されます。

```
MsgId: テキスト
```

"MsgId" は 7 文字の i5/OS メッセージ ID (たとえば CPF0001) であり、"テキスト" はメッセージのテキストのことです。 "MsgId" プレフィックスを含まないためには、**-n** オプションを使用します。

デフォルトでは、**system** はジョブ内で実行されているスレッド数を検査します。複数のスレッドが実行されていると、別のジョブを開始し、その別のジョブで *CLcommand* を実行します。**system** が現行のジョブで常に *CLcommand* を実行するよう強制するには、 **-i** オプションを使用します。

オプション

-i 現行のジョブで常に *CLcommand* を実行し、終了状況を *CLcommand* によって呼び出されたプログ

ラムの ILE 戻りコードに設定します。マルチスレッド対応ジョブ内で、またはジョブ内で実行している複数のスレッドがあるときに、いくつかの CL コマンドは実行しないことに注意してください。

-K *CLcommand* で生成されたすべてのスプール・ファイルと、 **system** を実行しているジョブのジョブ・ログを保存します。このオプションが指定されていない場合、そのスプール・ファイルは書き込み後に削除され、ジョブ・ログも削除されます。

-k *CLcommand* で生成されたすべてのスプール・ファイルを保存します。このオプションが指定されていない場合、そのスプール・ファイルは書き込み後に削除されます。

-n メッセージを標準エラーに書き込むときに、メッセージ ID を組み込みません。メッセージのメッセージ・テキストだけが標準エラーに書き込まれます。 **-q** オプションも指定されている場合は、このオプションは無視されます。

-p *CLcommand* によってプログラムのメッセージ待ち行列に送信されたメッセージのみを、標準エラーに書き込みます。 **-q** オプションも指定されている場合は、このオプションは無視されます。

-q *CLcommand* で生成されたメッセージを標準エラーに書き出しません。

-s *CLcommand* で生成されたスプール・ファイルを標準出力に書き出しません。

-v 完全な形のコマンド・ストリングを、実行前に標準出力に書き込みます。

PASE システム・ユーティリティーと互換性がある場合、 **-b**、**-e**、**-E**、**-I**、または **-O** オプションが指定されていれば、**system** はエラーを戻しませんが、オプションは無視されることに注意してください。

オペランド

各 *arg* は *CLcommand* コマンドに渡すパラメーターです。 **qsh** が特殊文字を展開しないようにするためには、*CLcommand* と *args* を引用符で囲まなければなりません。 CL と **qsh** は共に、特定の同じ特殊文字 (たとえばアスタリスク (*) 文字) を使用します。

環境変数

system ユーティリティーは、次の環境変数から影響を受けます。

QIBM_SYSTEM_ALWMLTTHD
マルチスレッド・ジョブで **system** ユーティリティーが動作する方法を制御するのにこの環境変数を設定します。その変数の値が "N" である場合、ジョブに実行しているスレッドが 1 つしかない場合でも、現行のジョブがマルチスレッド対応であるときに、**system** は CL コマンドを実行する新規ジョブを開始します。デフォルト値はありません。

QIBM_SYSTEM_USE_ILE_RC
system ユーティリティーが終了状況を設定する方法を制御するのにこの環境変数を設定します。変数の値が "Y" である場合、**system** は終了状況を、 *CLcommand* によって呼び出されたプログラムの ILE 戻りコードに設定しますが、プログラムが戻りコードを設定していない場合には、ゼロに設定します。デフォルト値はありません。**-i** オプションを指定した場合は、環境変数は無視されます。

終了状況

- 0 *CLcommand* が成功の場合。
- >0 *CLcommand* が不成功か、 *CLcommand* によって呼び出されたプログラムによって設定された場合。

-i オプションが指定されているか、環境変数 QIBM_SYSTEM_USE_ILE_RC=Y が設定されている場合、**system** は終了状況を、 *CLcommand* によって呼び出された ILE 戻りコードに設定しますが、プログラムが戻りコードに設定されていない場合には、ゼロに設定します。

例

1. すべての活動ジョブをリスト表示する場合:

   ```
   system wrkactjob
   ```

2. テスト・ライブラリーを作成する場合:

   ```
   system "CRTLIB LIB(TESTDATA) TYPE(*TEST)"
   ```

3. ライブラリーを削除し、メッセージを書き込まない場合:

   ```
   system -q "DLTLIB LIB(TESTDATA)"
   ```

関連タスク:

55 ページの『rexx - REXX プロシージャーを実行する』

関連情報:

CL コマンド検索プログラム

i5/OS PASE から i5/OS コマンドを実行

type - コマンドのタイプを検索する

構文

type [-apt] *name ...*

説明

type ユーティリティーは、指定されたそれぞれの *name* のタイプを表示します。 *name* は、別名、関数、特殊シェル組み込み、シェル組み込み、予約語、またはファイルのいずれかになります。

オプション

-a *name* のすべての使用を表示します。

-p *name* が予約語、組み込みユーティリティー、別名、関数かどうかをチェックしません。

-t *name* のタイプの説明を 1 ワードで表示します。

オペランド

各 *name* は、現行の環境の中のユーティリティーです。

終了状況

- 0 すべての *name* が見つかった場合。
- >0 不成功の場合。

関連タスク:

47 ページの『builtin - シェル組み込みユーティリティーを実行する』

47 ページの『command - 単純コマンドを実行する』

52 ページの『help - 組み込みユーティリティーの情報を表示する』

60 ページの『whence - コマンドの解釈の仕方を判断する』

whence - コマンドの解釈の仕方を判断する

構文

whence [-afpv] *name ...*

説明

whence ユーティリティーは、指定されたそれぞれの *name* がどのように解釈されるかを表示します。name は、別名、関数、特殊シェル組み込み、シェル組み込み、予約語、またはファイルのいずれかになります。

whence は **command -v** と同じで、**whence -v** は **command -V** と同じです。

オプション

-a *name* のすべての使用を表示します。

-f *name* が関数かどうかをチェックしません。

-p *name* が予約語、組み込みユーティリティー、別名、関数かどうかをチェックしません。

-v *name* のタイプを表示します。

オペランド

各 *name* は、現行の環境の中のユーティリティーです。

終了状況

- 0 すべての *name* が見つかった場合。
- >0 不成功の場合。

例

予約語 for のタイプを検索する場合、

```
whence -v for
```

関連タスク:

47 ページの『builtin - シェル組み込みユーティリティーを実行する』
47 ページの『command - 単純コマンドを実行する』
48 ページの『ドット (.) - 現行環境でコマンドを実行する』
52 ページの『help - 組み込みユーティリティーの情報を表示する』
59 ページの『type - コマンドのタイプを検索する』
50 ページの『eval - 引数を連結してコマンドを作成する』

xargs - 引数リストを作成してユーティリティーを起動する

構文

xargs [-t] [-e[*eofstring***]] [-E** *eofstring***] [-l[***number* **]] [-L** *number***] [-n** *number* **[-x]] [-s** *size***]** [*utility* [*arguments* ...]]

説明

xargs ユーティリティーは、スペース、タブ、改行、およびファイル終わりで区切られた *arguments* を標準入力から読み取ってから、それらを引数として使って、指定された *utility* を実行します。

呼び出しのたびに、*utility* およびコマンド行で指定されたすべての *arguments* が、標準入力から読み取られた特定数の *arguments* を後につけて *utility* に渡されます。標準入力を読みつくすまでこの *utility* は繰り返し実行されます。

単一引用符 (') または二重引用符 (") あるいは円記号 (¥) を使って、スペース、タブ、および改行を引数に組み込むことができます。単一引用符は、一致する単一引用符までの単一引用符以外のすべての文字 (改行を除く) をエスケープします。二重引用符は、一致する二重引用符までの二重引用符以外のすべての文字 (改行を除く) をエスケープします。改行を含むどの文字でも、円記号 (¥) でエスケープすることができます。

utility を指定しないと、デフォルトで **echo** が使われます。

utility が標準入力から読み取りを行うと、未定義の動作が行われることがあります。

コマンド行のアセンブルが不可能であると、**xargs** ユーティリティーはただちに終了し (それ以上入力を処理しないで)、 *utility* は呼び出せなくなり、*utility* の呼び出しはシグナルで終了するか、または *utility* の呼び出しは 255 の値で終了します。

オプション

-E *eofstring*
: 論理ファイル終わりストリングを指定します。 **xargs** は、ファイル終わりストリングまたは論理ファイル終わりストリングのいずれかが現れるまで標準入力を読み取ります。

-e[*eofstring***]**
: このオプションは **-E** オプションと同等のものです。 *eofstring* が指定されていない場合、デフォルト値は _ (単一下線) です。

-L *number*
: 標準入力から読み取られる引数を *number* 行ずつまとめて (空の行は除く)、*utility* を実行します。*utility* が最後に呼び出されるとき、残っている行が *number* を下回っているなら、最後の呼び出しに指定される引数の行数は少なくなります。行は、最初に改行文字が現れた箇所で終わるものと見なされます。ただし、行の最後の文字がブランク文字の場合は除きます。末尾ブランク文字は、この行と、その次の空でない行が連結することを示す記号です。 **-L** オプションと **-n** オプションは、互いに排他的関係にあります。後に指定した方のオプションが有効になります。

-l[*number* **]**
: このオプションは **-L** オプションと同等のものです。 *number* を指定しなかった場合のデフォルト値は 1 です。

-n *number*
: *utility* の呼び出しごとに標準入力から読み取られる *arguments* の最大数を設定します。累積されたバイト数 (**-s** を参照) が、指定サイズを超えるか、または *utility* の最後の呼び出し用に残っている引数の *number* よりも少ない場合、 *utility* の呼び出しでは、標準入力の引数の *number* よりも少ない数の引数が使われます。 i5/OS からプログラムに渡すことのできる引数の最大数は 255 です。*number* のデフォルト値は 250 です。 **-n** オプションと **-L** オプションは、互いに排他的関係にあります。後に指定した方のオプションが有効になります。

-s *size*
: *utility* に渡されるコマンド行の長さの最大バイト数を設定します。 *utility* に渡されるユーティリティー名と引数の長さの合計 (NULL 終了文字を含む) は、 *size* 以下になります。 *size* のデフォルト値は 16 252 928 バイトです。

-t トレース・モードをオンにします。実行されるコマンドは、その実行の直前に標準エラーに書き込まれます。

-x *number* 引数の入ったコマンドが指定 (またはデフォルト) のコマンド行の長さに入りきらない場合に、**xargs** を強制的にただちに終了します。

終了状況

- 0 呼び出されたすべての *utility* が、0 の戻り状況を戻した場合、
- 1 - 125 呼び出された *utility* のうち、最低 1 つがゼロ以外の戻り状況を戻した場合、あるいはエラーがあった場合。
- 126 *utility* は検出されたものの、呼び出せなかった場合。
- 127 *utility* を検出できなかった場合。
- >128 *utility* がシグナルによって終了した場合。値は、128 にシグナル番号を加えた値です。

関連タスク:

140 ページの『echo - 引数を標準出力に書き込む』

50 ページの『eval - 引数を連結してコマンドを作成する』

101 ページの『find - ファイルを検索する』

データの管理用のユーティリティー

データの管理用のユーティリティーを表示します。

cmp - 2 つのファイルを比較する

構文

cmp [-l | -s] [-t] *file1 file2* [*skip1* [*skip2*]]

説明

cmp を使って 2 つのファイルを比較することができます。デフォルトでは、バイナリー・ファイルは 1 バイトずつ比較されます。相違が見つからなければ、出力は書き出されません。オプション・フラグが指定されていないと **cmp** は、最初に見つけた相違のバイトと行番号をメッセージに書き出してから、エラーで終了します。バイトと行は、1 から始まって番号が付けられます。

オプション

-l (英字の小文字のエル) バイト番号を 10 進数で書き出し、相違があるごとにその相違バイトを書き出します。

-s ファイルの相違に関する出力を書き出さない無音モードです。終了状況のみが設定されます。

-t ファイルはテキスト・モードでオープンされてジョブの CCSID に変換されてから、 1 バイトずつ比較されます。

オペランド

file1 および *file2* オペランドは、1 バイトずつ比較される 2 つのファイルです。オプションの *skip1* および *skip2* は、各ファイルの先頭から比較の開始地点までの間にあってそれぞれスキップされるバイト数です。

環境変数

cmp は、次の環境変数の影響を受けます。

QIBM_CMP_FILE_SIZE

パフォーマンスを向上するために **cmp** が内部バッファーに読み取る最大ファイル・サイズ (バイト単位) を制御します。最大サイズよりも大きいファイルの場合、**cmp** は一度に 1 バイトずつファイルを読み取ります。

終了状況

- 0 ファイルは同一です。
- 1 ファイルに相違があります。
- >1 エラー発生。

例

2 つのファイルの厳密な相違個所を見つけ出す場合: 参照ファイルまたは正しいファイルを先に置いてから、変更後または新規のファイルを置くのが望ましいやり方です。

```
cmp myApplet.java.old myApplet.java.new
```

関連タスク:

67 ページの『sed - ストリーム・エディター』

73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』

75 ページの『split - ファイルを分割する』

78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』

cut - ファイルの各行から選択したフィールドを切り取る

構文

cut -b *list* [*file* ...]

cut -c *list* [*file* ...]

cut -f *list* **[-d** *string***] [-s]** [*file* ...]

説明

cut ユーティリティーは、指定された *list* に従って各 *file* (またはデフォルトの標準入力) から各行部分を選択し、それを標準出力に書き込みます。 *list* では、列位置を軸にするか、または特殊文字で区切られたフィールドを軸にして項目を指定することができます。列番号は 1 から始まります。

list は、順次増大する一連の番号または番号範囲 (またはこの両方) をコンマまたは空白文字で区切ったものです。番号の範囲は、番号、ダッシュ (-)、および 2 番目の番号で構成され、 1 番目から 2 番目までの間の番号 (これらを含む) の各フィールドまたは列が選択されます。番号または番号範囲の前にダッシュを付ければ、 1 から最初の番号までのすべてのフィールドまたは列を選択することができます。番号または番号範囲の後にダッシュを付ければ、最初の番号から行の末尾までのすべてのフィールドまたは列を選択することができます。番号と番号範囲は、反復および重複することができ、任意の順序にすることができます。入力行内に存在しないフィールドまたは列を選択してもエラーになりません。

オプション

-b *list* *list* はバイト位置を指定します。

-c *list* *list* は文字位置を指定します。

-d *string*
string の先頭文字を、タブ文字ではなくフィールド区切り文字として使います。

-f *list* *list* は、入力内で 1 つのタブ文字で区切られたフィールドを指定します。出力フィールドは、1 つのタブ文字で区切られます。

-s フィールドの区切り文字のない行を抑止します。指定がなければ、区切り文字のない行は無変更のまま渡されます。

終了状況

- 0 正常終了。
- 1 エラー発生。

関連タスク:

『grep - パターンに対応するファイルを検索する』

76 ページの『tr - 文字を変換する』

79 ページの『wc - ワード、行、およびバイト/文字をカウントする』

egrep - ファイル内の拡張正規表現パターンを検索する

構文

egrep [-c|-l|-q] [-ihnsvwxy] [-e *pattern_list*] **[-f** *pattern_file*] [*pattern*] [*file ...*]

説明

egrep ユーティリティーは、**-E** オプションを指定して **grep** ユーティリティーを実行するのと同等です。

関連タスク:

『fgrep - ファイル内の固定ストリング・パターンを検索する』

『grep - パターンに対応するファイルを検索する』

fgrep - ファイル内の固定ストリング・パターンを検索する

構文

fgrep [-c|-l|-q] [-ihnsvwxy] [-e *pattern_list*] **[-f** *pattern_file*] [*pattern*] [*file ...*]

説明

fgrep ユーティリティーは、**-F** オプションを指定して **grep** ユーティリティーを実行するのと同等です。

関連タスク:

『egrep - ファイル内の拡張正規表現パターンを検索する』

grep - パターンに対応するファイルを検索する

構文

grep [-E|-F] [-c|-l|-q] [-R [-H | -L | -P]] [-ihnsvwxy] [-e *pattern_list*] **[-f** *pattern_file*] [*pattern*] [*file ...*]

説明

grep ユーティリティーは、1 つ以上の *pattern* に一致する行を選択して、指定された入力 *file* を検索します。パターンのタイプは指定したオプションによって制御されます。デフォルト設定では、パターン内の正

規表現 (RE) が 1 つでも入力行に一致していれば、パターンは入力行に一致するものと見なされます (後続の改行は無視されます)。ヌル RE はすべての行に一致します。各入力行は、一致するパターンが少なくとも 1 つあれば、標準出力に書き込まれます。

-E オプションと **-F** オプションの両方が指定されている場合は、最後に指定したオプションが使用されます。

オプション

-E 拡張正規表現 (ERE) を使用します。

-F 正規表現は認識されません。

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。

-R *file* にディレクトリーが指定されている場合、その時点で連結されている全サブツリー中の各ファイルが **grep** によって検索されます。

-c 選択された行の数のみが標準出力に書き込まれます。

-e *pattern_list* には、1 つ以上の検索パターンを指定します。各パターンは改行文字で区切ります。

-f *pattern_file* には、検索パターンが入っているファイルを指定します。 各パターンは改行文字で区切ります。

-h ファイル名ヘッダーを出力しません。

-i 比較するときに大文字と小文字を区別しません。つまり、大文字と小文字は同一の文字と見なされます。

-l 選択された行の入ったファイルの名前のみが、標準出力に書き込まれます。検索されたファイル 1 つにつき 1 回、パス名がリスト表示されます。標準入力が検索される場合は、パス名 "-" が書き込まれます。

-n 各出力行の前には、ファイル内でのその行の相対行番号が付きます。各ファイルは、行番号 1 から始まります。行番号カウンターは、ファイルを 1 つ処理するたびにリセットされます。 **-c**、**-l**、または **-s** オプションが指定されている場合は、このオプションは無視されます。

-q メッセージが出力されていないときは静止モードになります。戻り状況のみが戻されます。

-s 存在しないファイルまたは読み取ることができないファイルの場合には、通常なら書き込まれるエラー・メッセージを抑制します。 その他のメッセージは抑制されません。

-v 指定したパターンに一致しない行が選択されます。

-w 正規表現全体を 1 つのワードとして ([[:<:]] や [[:>:]] で囲まれている場合と同じように) 検索します。

-x 行に含まれているのがパターン**だけ**の場合、行に一致するものと見なされます。このオプションは **-w** オプションより優先されます。両方が指定されている場合は、**-w** オプションは無視されます。

-y 大文字小文字が区別されません (**-i** と同じ)。

オペランド

各 *file* はテキスト・ファイルへのパスを指定します。 *file* オペランドを指定しなかった場合は、標準入力が使用されます。

終了状況

- 1 つまたは複数の行が選択された場合は、0。
- 行が選択されなかった場合は、1。
- >1 エラー発生。

拡張正規表現 (ERE)

grep では、次の文字は以下のように解釈されます。

$ 行末から突き合わせをそろえる。

^ 行の先頭から突き合わせをそろえる。 (注: この文字は 5250 端末セッションでは正しく動作しない場合があります。)

| 別のパターンの追加 (下の例を参照してください)。

? パターンの 1 回以下の連続反復に一致。

+ パターンの 1 回以上の連続反復に一致。

***** パターンの 0 回以上の連続反復に一致。

. 任意の単一の文字に一致します。

[ ] 大括弧で囲まれた任意の 1 文字または文字の範囲に一致。

grep にとって意味を持つエスケープ特殊文字、つまり {$,.,^,[,],|,?,+,*,(,)} の組み合わせ。

例

1. ワード patricia のすべての出現をファイルの中で検索する場合 :
   ```
   grep patricia myfile
   ```
2. 行の先頭にある、パターン「.Pp」すべてを検索する場合。単一引用符で囲めば、シェルでなく、確実に **grep** によって式全体が評価されます。脱字記号 (^) は行の先頭からの意味 :
   ```
   grep '^.Pp' myfile
   ```
3. ファイル calendar の中で、19、20、または 25 を検索する場合 :
   ```
   grep -E '19|20|25' calendar
   ```
4. a から Z までの範囲内の文字に一致する文字を含む行の合計行数を調べる場合 :
   ```
   grep -c '[a-z]' reference/alphabet.text
   ```
5. ドル記号 ($) を備えた行をすべて表示する場合。**grep** がドル記号文字を拡張正規表現として解釈しないように、ドル記号をエスケープさせる必要があります。この例では、一致する行と共に、行番号も表示されます :
   ```
   grep -n '$' valid.file
   ```

関連概念:

63 ページの『cut - ファイルの各行から選択したフィールドを切り取る』

関連タスク:

64 ページの『egrep - ファイル内の拡張正規表現パターンを検索する』

76 ページの『tr - 文字を変換する』

79 ページの『wc - ワード、行、およびバイト/文字をカウントする』

iconv - ある CCSID から別の CCSID に文字を変換する

構文

iconv -f *fromCCSID* **-t** *toCCSID* **[** *file ...* **]**

説明

iconv ユーティリティーは、標準入力または指定した *file* のいずれかから読み取った文字のエンコード方式を、ある CCSID から別の CCSID に変換し、その後、結果を標準出力に書き込みます。入力データは、*fromCCSID* パラメーターで指定される CCSID を使用しているものと見なされます。 *file* が指定されていない場合、**iconv** ユーティリティーは標準入力から読み取りを行います。

fromCCSID パラメーターと *toCCSID* パラメーターには、変換のサポートされた、i5/OS で定義された CCSID 値を指定する必要があります。

オプション

-f *fromCCSID*
入力データは *fromCCSID* でエンコードされています。

-t *toCCSID*
出力データは *toCCSID* でエンコードされます。

オペランド

file オペランドには、正規ファイルへのパス名を指定します。

終了状況

- 0 正常終了。
- 1 その変換がサポートされていない、または *file* にエラー発生。
- 2 変換中にエラー発生。

関連タスク:

191 ページの『locale - ロケール特定情報を入手する』
76 ページの『tr - 文字を変換する』
131 ページの『setccsid - ファイルの CCSID 属性を設定する』
『sed - ストリーム・エディター』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
75 ページの『split - ファイルを分割する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』

sed - ストリーム・エディター

構文

sed [-an] [-C *ccsid* **]** *command file ...*

sed [-an] [-C *ccsid* **] [-e** *command***] [-f** *command_file***]** *file ...*

説明

sed ユーティリティーは、指定された *files* を読み取って (ただし、ファイルが指定されていなければ標準入力を読み取って) から、 *commands* のリストで指定されているとおりに入力を変更します。次にその入力は標準出力に書き込まれます。

sed の最初の引数として *command* を指定することができます。 **-e** または **-f** オプションを使うと、複数のコマンドを指定することができます。すべてのコマンドは、その起点がどこであっても、指定された順に入力に対して適用されます。

オプション

-a
デフォルトでは、**w** 関数用のパラメーターとしてリストされるファイルが作成され (または切り捨てられ) てから、処理が開始されます。 **-a** オプションを使うと **sed** は、それに関連した **w** 関数の入ったコマンドが入力行に適用されるまで各ファイルのオープンを遅らせます。

-C *ccsid*
sed によって作成されるファイルはどれも、 *ccsid* によって指定される CCSID で作成されます。このオプションは、QIBM_CCSID 環境変数の値をオーバーライドします。

-e *command*
command 引数で指定された編集コマンドをコマンド・リストに付加します。

-f *command_file*
ファイル *command_file* 内で見つかった編集コマンドをコマンド・リストに付加します。編集コマンドは各々別々の行にリストされていなければなりません。

-n
デフォルトでは、すべてのコマンドが入力行に対して適用された後、それらの入力行は 1 つずつ標準出力にエコーされます。 **-n** オプションは、このような動作を抑止します。

オペランド

sed コマンドの形式は次のとおりです。

```
[address[,address]]function[arguments]
```

コマンドの最初の

address

と

function

部分の前に空白文字を挿入してもかまいません。

通常 **sed** はサイクルとして、終了の改行文字以外の入力行を「パターン・スペース」にコピーし (ただし、 **D** の後に何も残っていない場合に限ります)、そのパターン・スペースを選択したアドレスを指定してすべてのコマンドを適用し、改行を付加したうえでそのパターン・スペースを標準出力にコピーし、そしてそのパターン・スペースを削除します。

一部の関数は、以後の検索用にパターン・スペースの一部または全部を保管するための「保留スペース」を使います。

補足説明

sed アドレス

アドレスは必要ありませんが、指定する場合は以下のとおりでなければなりません。

- 複数の入力ファイルの入力行を累積してカウントした数。
- 入力の最終行をアドレス指定するドル ($) 文字。
- 正規表現で構成され、前後に区切り文字を付けられたコンテキスト・アドレス。

アドレスの指定のないコマンド行は、すべてのパターンを選択します。

1 つのアドレスを指定されたコマンド行は、そのアドレスに一致するすべてのパターン・スペースを選択します。

2 つのアドレスを指定されたコマンド行は、最初のアドレスに一致する最初のパターン・スペースから、 2 番目のアドレスに一致する次のパターン・スペースまでの範囲を選択します。 2 番目のアドレスが最初に選択された行番号より小さいか等しいと、その行だけが選択されます。 **sed** は、選択した範囲の後に続く最初の行から開始して、最初のアドレスをもう一度探索します。

感嘆符 (!) 関数を使用すれば、選択されていないパターン・スペースに対して編集コマンドを適用することができます。

sed 正規表現

sed 正規表現は、基本の正規表現です。さらに **sed** には、それ以外に次のような 2 つの基本の正規表現があります。

- コンテキスト・アドレスでは円記号 (¥) または改行文字以外の文字はすべて、正規表現の区切り文字として使用することができます。また、区切り文字の前に円記号 (¥) を置くと、その文字はリテラルとして扱われます。たとえば、コンテキスト・アドレス *¥xabc¥xdefx* では、正規表現区切り文字は *x* ですが、2 番目の *x* は自身を表すので、正規表現は *abcxdef* になります。
- エスケープ・シーケンス ¥n は、パターン・スペースに組み込まれた改行文字に一致します。ただし、アドレスまたは置換コマンド内でリテラルの改行文字を使用することはできません。

sed 正規表現の特殊機構の 1 つに、最後に使われた正規表現をデフォルトにできる機能があります。正規表現が空の場合、つまり区切り文字だけを指定すると、最後に検出された正規表現がその代わりに使われます。最後の正規表現とは、コンパイル時ではなくランタイム時に、アドレスまたは置換コマンドの一部として最後に使われた正規表現と定義されます。たとえば次のようなコマンドがあるとします。

```
/abc/s//XXX/
```

上記の場合、XXX がパターン abc に置き換えられます。

sed 関数

以下のコマンド・リストでは、各コマンドごとに指定可能なアドレスの最大数は、それぞれゼロ、1、または 2 つのアドレスを表す [*0addr*]、[*1addr*]、または [*2addr*] で示されます。

引数 *text* は 1 つ以上の行で構成されます。テキストに改行を組み込むためには、その前に円記号 (¥) を付けます。 *text* 内のその他の円記号は削除され、その後の文字がリテラルと見なされます。

r 関数と **w** 関数は、オプションの *file* パラメーターをとりますが、これらは、空白文字で関数文字から区切らなければなりません。 **sed** に引数として指定した各ファイルは、他のどの入力処理の開始よりも前に作成され (またはその内容が切り捨てられ) ます。

b、**r**、**s**、**t**、**w**、**y**、**!**、および **&** のすべての関数は、さらに別の引数を受け入れます。以下の構文は、どの引数を空白文字で関数文字から区切らなければならないかを示しています。

この関数のうちの 2 つは *function-list* をとります。それは次のような、改行で区切られた **sed** 関数リストです。

```
{ function
  function
  ...
  function
}
```

{ の前および後には空白文字を付けることができます。この関数の前には空白文字を付けることができます。終了の **}** の前には改行またはオプションの空白文字を付けなければなりません。

[*2addr***]** *function-list*
: パターン・スペースが選択されている場合のみ *function-list* を実行します。

[*1addr***]a¥** *text*
: 入力行の読み取りが試みられるたびに、その直前に *text* を標準出力に書き込みます。そのために、**N** 関数を実行するか、または新規のサイクルを開始します。

[*2addr***]b[***label***]**
: 指定の *label* を使って **&** 関数に分岐します。 *label* を指定しないと、スクリプトの末尾に分岐します。

[*2addr***]c¥** *text*
: パターン・スペースを削除します。 0 または 1 個のアドレスを使うか、または 2 アドレス範囲の終わりに、*text* が標準出力に書き込まれます。

[*2addr***]d**
: パターン・スペースを削除してから、次のサイクルを開始します。

[*2addr***]D**
: 最初の改行文字を介してパターン・スペースの初期セグメントを削除してから、次のサイクルを開始します。

[*2addr***]g**
: パターン・スペースの内容を、保留スペース内の内容に置き換えます。

[*2addr***]G**
: 保留スペース内の内容を後に付けた改行文字をパターン・スペースに付け加えます。

[*2addr***]h**
: 保留スペースの内容を、パターン・スペース内の内容に置き換えます。

[*2addr***]H**
: パターン・スペース内の内容を後に付けた改行文字を保留スペースに付け加えます。

[*1addr***]i¥** *text*
: *text* を標準出力に書き込みます。

[*2addr***]l**
: (英字の小文字の「エル」。) 視覚的にあいまいな形式でパターン・スペースを標準出力に書き込みます。その形式は次のとおりです。

  - 円記号 (¥)
  - アラート (¥a)
  - 書式送り (¥f)
  - 改行 (¥n)
  - 復帰 (¥r)

- タブ (¥t)
- 垂直タブ (¥v)

印刷不能文字は、文字内の各バイトごとに 3 桁の 8 進数 (前に円記号が付けられます) で書き込まれます (再重要バイトを最初に)。長い行は折り返されますが、折り返し地点は、円記号とその後に続く改行の表示によって示されます。各行の末尾には、ドル記号 ($) が付けられます。

[*2addr***]n**

デフォルト出力が抑止されていなければ、パターン・スペースを標準出力に書き込んでから、そのパターン・スペースを次の入力行に置き換えます。

[*2addr***]N**

付加マテリアルとオリジナルの内容を区切るのに組み込み改行文字を使って、次の入力行をパターン・スペースに付加します。現在行番号が変わることに注意してください。

[*2addr***]p**

パターン・スペースを標準出力に書き込みます。

[*2addr***]P**

パターン・スペースの最初の改行文字までを標準出力に書き込みます。

[*1addr***]q**

スクリプトの末尾に分岐してから、新規のサイクルを開始しないで終了します。

[*1addr***]r** *file*

入力行の読み取りが次に試みられる直前に、*file* の内容を標準出力にコピーします。何らかの理由で *file* が読み取り不能の場合、通告なしに無視され、エラー条件は設定されません。

[*2addr***]s/***regular_expression***/***replacement***/** *flags*

replacement ストリングを、パターン・スペース内の *regular_expression* の最初のインスタンスと置き換えます。 *regular_expression* と *replacement* を区切るのに、スラッシュの代わりに、円記号または改行以外のすべての文字を使うことができます。 *regular_expression* と *replacement* 内では、正規表現区切り文字の前に円記号を付ければ、その区切り文字そのものをリテラルとして使うことができます。

replacement 内に & 記号が置かれていると、正規表現に一致するストリングに置き換えられます。この場合のコンテキストで & の意味を抑止するには、その前に円記号を付けます。ストリング ¥# (# は数字です) は、対応する逆参照式に一致するテキストに置き換えられます。

改行文字を行に代入すれば、その行を分割することができます。置き換えストリングに改行を指定するには、その前に円記号 (¥) を付けます。

置き換え関数内の *flags* の値は、以下のうちの 0 個以上です。

0 ... 9 パターン・スペース内で N 番目に出現した正規表現でのみ置換を行います。

g 最初のものだけでなく、正規表現に一致する非オーバーラップ・マッチングのすべてで置換を行います。

p 置換が行われた場合、パターン・スペースを標準出力に書き込みます。置き換えストリングが置き換え前のものと同一であっても、やはり置き換えと見なされます。

w *file* 置換が行われた場合、パターン・スペースを *file* に付加します。置き換えストリングが置き換え前のものと同一であっても、やはり置き換えと見なされます。

[*2addr***]t [***label***]**

入力行の最新の読み取りまたは **t** 関数の実行以後に置換が行われた場合、 *label* の付いた **:** 関数に分岐します。 *label* を指定しないと、スクリプトの末尾に分岐します。

[*2addr***]w** *file*
パターン・スペースを *file* に付加します。

[*2addr***]x**
パターン・スペースと保留スペースの内容をスワップします。

[*2addr***]y/***string1***/***string2***/**
パターン・スペースの *string1* 内に文字が現れるごとにそれらをすべて、 *string2* の対応する文字に置き換えます。ストリングを区切るスラッシュの代わりに、円記号または改行以外のすべての文字を使うことができます。 *string1* と *string2* では、改行以外の任意の文字が後に続く円記号はリテラル文字になり、 **/n** は改行文字に置き換えられます。

[*2addr***]!***function*

[*2addr***]!***function-list*
function または *function-list* を、アドレスで選択されて**いない**行にだけ適用します。

[*0addr***]:***label*
この関数は何もしません。**b** コマンドと **t** コマンドの分岐先になる可能性のある *label* が付けられています。

[*1addr***]=**
後に改行文字を付けて、行番号を標準出力に書き込みます。

[*0addr***]**
空行は無視されます。

[*0addr***]#**
とその行の残りは無視されます (注記と解釈されます)。ただし、ファイル内の最初の 2 文字が **#n** の場合にデフォルト出力が抑止されることを唯一の例外とします。これは、コマンド行に -n オプションを指定するのと同じことです。

環境変数

sed は、次の環境変数の影響を受けます。

QIBM_CCSID
sed によって作成されるファイルはどれも、環境変数の値によって指定される CCSID で作成されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

62 ページの『cmp - 2 つのファイルを比較する』
67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
191 ページの『locale - ロケール特定情報を入手する』
76 ページの『tr - 文字を変換する』
131 ページの『setccsid - ファイルの CCSID 属性を設定する』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
75 ページの『split - ファイルを分割する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』

sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する

構文

sort **[-cmubdfinr]** **[-t** *char***]** **[-T** *char***]** **[-k** *keydef* **...]** **[-o** *output***]** [*file*] ...

説明

sort ユーティリティーはテキスト・ファイルを行別にソートします。比較は、各入力行から抽出された 1 つ以上のソート・キーを基にして、辞書編集方式で行われます。デフォルトでは、キーが指定されない場合には、**sort** は各入力行を 1 つのフィールドと見なします。

オプション

-c 1 つの入力ファイルがソートされているかどうかを調べます。ファイルがソートされていない場合 **sort** は、該当するエラー・メッセージを作成し、コード 1 で終了します。そうでなければ **sort** は、0 を戻します。このオプションは出力を生成しません。

-m マージのみ。入力ファイルは事前ソートされているものと見なされます。

-o *output*
output 引数は、標準出力の代わりに使われる出力ファイルの名前です。このファイルは、入力ファイルのうちの 1 つと同じであってもかまいません。

-u 等しいキーをもつ一連の行のうちの 1 つを除くすべての行を抑止するための独特な処理法。 **-c** オプションと一緒に使用された場合、重複キーをもつ行がないかどうかを検査します。

以下のオプションは、デフォルトの順序付け規則をオーバーライドします。キー・フィールドの指定に従わない順序付けオプションが出現すると、要求されたフィールドの順序付け規則がすべてのソート・キーにグローバルで適用されます。順序付けオプションは、特定のキーに付加された場合、そのキーのすべてのグローバルな順序付けオプションをオーバーライドします。

-d 比較を行うのに、ブランク・スペースと英数字だけが使われます。

-f 比較の際、同等の英大文字をもつすべての英小文字を同一視します。

-i 印刷不能文字をすべて無視します。

-n オプションのブランク・スペース、オプションの負符号、および 0 個以上の数字 (小数点を含む) から成る初期数値ストリングが、数値別にソートされます。

-r 比較の意義を反転します。

フィールド区切り記号の処理法は、以下のオプションを使って変更することができます。

-b 制限付きのソート・キーの先頭と末尾を判別するときに、先行ブランク・スペースを無視します。最初の **-k** オプションの前に **-b** オプションを指定すると、後者はすべての **-k** オプションにグローバル適用されます。あるいは、**-k** オプションの各フィールド引数に **-b** オプションを単独で付加することができます (以下を参照)。キー・フィールドが指定されていないと、**-b** オプションには効力はないことに注意してください。

-t *char* *char* 引数をフィールドの区切り文字として使用します。キー・オフセットを指定するときは、最初の *char* はフィールドの一部と見なしません。 *char* は出現するごとに有効です (たとえば char-char は空のフィールドを区切ります)。 **-t** を指定しないと、デフォルトのフィールド区切り記号としてブランク・スペース文字が使われます。

-T *char*

char 引数をレコードの区切り文字として使用します。このオプションは、慎重に使用する必要があります。 **-T** オプションを英数字の *char* と一緒に使うと、たいていは望ましくない結果を生じます。デフォルトの行区切り文字は改行です。

-k *keydef*

ソートのために使用するキー・フィールドを選択します。 *keydef* のフォーマットは次のとおりです。

field_start[*type*][,*field_end*[*type*]]

ここで、*field_start* はキー・フィールドの開始位置であり、*field_end* はキー・フィールドの終了位置です (*field_end* はオプション)。 *field_end* が指定されていない場合、終了位置は行の末尾です。 *type* は、b、d、f、i、n、r の文字のいずれかです。 *type* は、それに対応するオプションと同じ働きをしますが、指定されたキー・フィールドに対してのみ働きます。 **-k** オプションが指定されていないと、デフォルトのソート・キーが使用されます。最大 9 個の **-k** オプションを指定することができます。

オペランド

ソート、マージ、または検査したいファイルのパス名。 *file* オペランドを指定しないと、標準入力が使用されます。

補足説明

フィールドとは、最小限の回数だけ出現する文字とその後に続くフィールド区切り文字または改行文字と定義されます。デフォルトでは、一つながりのブランク・スペースの最初のブランク・スペースがフィールド区切り文字として働きます。一つながりのブランク・スペースでは、それぞれのブランク・スペースが次のフィールドの一部と見なされます。たとえば、行の先頭にあるどのブランク・スペースも、最初のフィールドの一部と見なされます。

フィールドは、 **-k** *field_start*[*type*][,*field_end*[*type*]] オプションにより指定されます。

オプションの引数の *field_start* の部分は次の形式です。

field_number[.*first_character*]

フィールドおよびフィールド内の文字には、1 から順番に番号が振られます。*field_number* および *first_character* は正の 10 進整数で、ソート・キーの一部として使用される先頭文字を指定します。 *first_character* が指定されていない場合、それはフィールドの先頭文字を意味します。

オプションの引数の *field_end* の部分は次の形式です。

field_number[.*last_character*]

field_number は正の 10 進整数で、*last_character* は負でない 10 進整数です。 *last_character* が指定されていないかゼロの場合、それはフィールドの最後の文字を意味します。

-b オプションまたは b タイプ修飾子が有効な場合、フィールドの文字は最初の非ブランク文字からカウントされます。

終了状況

- 0 通常処理。

- 1 -c オプションの場合に、順序付けされていない (または固有性がない)。
- 2 エラー発生。

関連タスク:

62 ページの『cmp - 2 つのファイルを比較する』
67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
191 ページの『locale - ロケール特定情報を入手する』
76 ページの『tr - 文字を変換する』
131 ページの『setccsid - ファイルの CCSID 属性を設定する』
67 ページの『sed - ストリーム・エディター』
『split - ファイルを分割する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』

split - ファイルを分割する

構文

split [-b *byte_count***[k|m]] [-l** *line_count***]** [*file* [*prefix*]]

説明

split ユーティリティーは、指定された *file* (ファイルが指定されていない場合は標準入力) を読み取り、それをそれぞれ 1000 行ずつのファイルに分割します。

オプション

-b 長さが *byte_count* バイトのファイルが作成されます。数値の後ろに **k** が付いている場合、ファイルは *byte_count* キロバイトの小ファイルに分割されます。 数値の後ろに **m** が付いている場合は、ファイルは *byte_count* メガバイトの小ファイルに分割されます。

-l 長さが *line_count* 行のファイルが作成されます。

オペランド

追加の引数を指定した場合、最初の引数は、分割する入力ファイルの名前として使用されます。第 2 の追加引数は、分割後の各ファイルの名前に付加するプレフィックスとして使用されます。 この場合、*file* を分割してできた各ファイルの名前は、このプレフィックスの後ろに、aa - zz の範囲内のサフィックスをアルファベット順に付けたものとなります。 *prefix* 引数を指定しなかった場合、デフォルトのプレフィックスは x です。可能な出力ファイル名の最大数は、676 です。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. jdk_v11.jar ファイルを 1.44MB のサイズのファイルに分割し、出力ファイルのプレフィックスとして jdk_v11 を使用する場合:

   ```
   split -b1440k jdk_v11.jar jdk_v11.
   ```

2. myapp.java ファイルをそれぞれ 100 行のファイルに分割する場合 :

   ```
   split -l 100 myapp.java
   ```

関連タスク:

62 ページの『cmp - 2 つのファイルを比較する』
67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
191 ページの『locale - ロケール特定情報を入手する』
『tr - 文字を変換する』
131 ページの『setccsid - ファイルの CCSID 属性を設定する』
67 ページの『sed - ストリーム・エディター』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』

tr - 文字を変換する

構文

tr [-cs] *string1 string2*

tr [-c] -d *string1*

tr [-c] -s *string1*

tr [-c] -ds *string1 string2*

説明

tr ユーティリティーは、選択された文字を置き換え、または削除した上で、標準入力を標準出力にコピーします。

最初の構文形式では、*string1* 内の文字が *string2* 内の文字に変換されます。ただし *string1* 内の最初の文字は、*string2* 内の最初の文字に変換され、以後同様に変換されます。 *string1* のほうが *string2* より長い場合、*string2* 内で最後に検出された文字が複製され、これが *string1* の文字がなくなるまで続けられます。

2 番目の構文形式では *string1* 内の文字は入力から削除されます。

3 番目の構文形式では、以下の **-s** オプションの項に説明されているとおりに、 *string1* 内の文字は圧縮されます。

4 番目の構文形式では、*string1* 内の文字は入力から削除され、 *string2* 内の文字は、以下の **-s** オプションの項に説明されているとおりに圧縮されます。

一連の文字を設定するのに、*string1* と *string2* で以下のような規則を使用することができます。以下の規則のいずれにも解説されていない文字は、その文字自身を指します。

nnn 円記号 (¥) とその後に続く 1、2、または 3 桁の 8 進数字は、そのエンコード値を備えた文字を表します。

char 8 進文字列の後に文字として 1 桁を続けるには、最大 3 桁すべてを使って 8 進数のゼロをその 8 進文字列の左側に埋め込みます。円記号 (¥) の後に特定の特殊文字を続けると、特殊値にマップされます。その特殊文字とその値は次のとおりです。

- a - アラート文字
- b - バックスペース文字
- f - 書式送り

- n - 改行
- r - 復帰
- t - タブ
- v - 垂直タブ
- 円記号 (¥) の後に他の任意の文字を続けると、その文字にマップされます。

c-c 2 つの範囲終了地点間の文字範囲を表します。

[:class:]

定義された文字クラスに属するすべての文字を表します。そのクラス名は次のとおりです。

- alnum - 英数字
- alpha - 英字
- cntrl - 制御文字
- digit - 数字
- graph - 図形文字
- lower - 小文字の英字
- print - 印刷可能文字
- punct - 句読文字
- space - スペース文字
- upper - 英大文字
- xdigit - 16 進文字

注 :	英字の大文字と小文字のクラス以外のクラス内の文字は、不定の順序になります。英字の大文字と小文字のクラスでは文字は昇順で入力されます。

オプション

-c *string1* 内の一連の文字を完成します。つまり、 a と b 以外のすべての文字を **-c ab** に組み込みます。

-d 入力から文字を削除します。

-s 入力内の最後のオペランド (*string1* または *string2*) に一覧で示された複数回出現する文字を 1 つの文字インスタンスにまとめます。これが行われるのは、すべての削除と変換が完了した後です。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. file1 内に 1 行に 1 ワードずつのワード・リストを作成する場合 (ただし、1 つのワードは最大数のストリングであると見なされます):

   ```
   tr -cs '[:alpha:]' 'n' < file1
   ```

2. file1 の内容を英大文字に変換する場合:

   ```
   tr '[:lower:]' '[:upper:]' < file1
   tr 'a-z' 'A-Z' < file1
   ```

3. 印刷不能文字を file1 から除去する場合:

```
tr -cd '[:print:]' < file1
```

関連概念:

63 ページの『cut - ファイルの各行から選択したフィールドを切り取る』

関連タスク:

64 ページの『grep - パターンに対応するファイルを検索する』
67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
67 ページの『sed - ストリーム・エディター』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
75 ページの『split - ファイルを分割する』
『uniq - ファイル内の繰り返し行を報告または抽出する』
79 ページの『wc - ワード、行、およびバイト/文字をカウントする』
191 ページの『locale - ロケール特定情報を入手する』

uniq - ファイル内の繰り返し行を報告または抽出する

構文

uniq [-c | -du] [-f *fields*] **[-s** *chars*] [*input_file* [*output_file*]]

説明

uniq ユーティリティーは、標準入力を読み取って隣接する行同士を比較し、個々の固有入力行を標準出力にコピーします。 同じ入力行が隣接して複数ある場合は、2 番目以降の行は書き込まれません。 入力内に反復行があっても、隣接していないと検出されないので、最初にファイルをソートしておくことが必要な場合があります。

オプション

-c 各出力行の前に、入力の中でのその行の出現回数を示すカウント数と、その数の後ろにスペースが1つ入ります。

-d 入力内にある反復していない行は、書き込まれないようにします。

-f *fields*
比較を行うときに、各入力行の最初の *fields* フィールドが無視されます。フィールドはブランク以外の文字のストリングであり、隣接するフィールドどうしはブランクで互いに区切られます。フィールド番号の基数は 1 です。つまり、最初のフィールドはフィールド 1 になります。

-s *chars*
比較を行うときに、各入力行の最初の *chars* 文字が無視されます。 **-f** オプションも同時に指定した場合は、最初の *fields* フィールドの最初の *chars* 文字は無視されます。文字番号の基数は 1 です。つまり、最初の文字は文字 1 になります。

-u 入力内にある反復している行は、書き込まれないようにします。

オペランド

コマンド行で追加の引数を指定した場合は、その中の最初の引数は、入力ファイルの名前として使用され、2 番目の引数は出力ファイルの名前として使用されます。

終了状況

- 0 正常終了。

- >0 エラー発生。

例

以下の例で使用するファイルの内容は次のとおりです。

```
There are 5 apples
There are 9 oranges
There are 9 oranges
There are 2 pears
```

1. ファイル fruit の中の固有の行を表示する場合 :

   ```
   uniq fruit

   There are 5 apples
   There are 9 oranges
   There are 2 pears
   ```

2. ファイル fruit の中の反復している行を表示する場合 :

   ```
   uniq -d fruit

   There are 9 oranges
   ```

3. ファイル fruit の中の行の反復している回数のリストを表示する場合 :

   ```
   uniq -c fruit

   1 There are 5 apples
   2 There are 9 oranges
   1 There are 2 pears
   ```

関連タスク:

62 ページの『cmp - 2 つのファイルを比較する』

67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』

191 ページの『locale - ロケール特定情報を入手する』

76 ページの『tr - 文字を変換する』

131 ページの『setccsid - ファイルの CCSID 属性を設定する』

67 ページの『sed - ストリーム・エディター』

73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』

75 ページの『split - ファイルを分割する』

wc - ワード、行、およびバイト/文字をカウントする

構文

wc [-c | -m] [-lw] [*file* ...]

説明

wc ユーティリティーは、各入力 *file* (デフォルトでは標準入力) に入っている行、ワード、およびバイトの数を標準出力に表示します。行とは、改行文字で区切られたストリングと定義されます。ワードとは、空白文字で区切られたストリングと定義されます。複数の入力ファイルを指定すると、すべてのファイルの累積カウントの行が、最後のファイルの出力後に別個の行上に表示されます。

オプション

c 各入力ファイル内のバイト数を標準出力に書き出します。

l 各入力ファイル内の行数を標準出力に書き出します。

m 各入力ファイル内の文字数を標準出力に書き出します。

w 各入力ファイル内のワード数を標準出力に書き出します。

オペランド

オプションが指定されると **wc** は、そのオプションから要求された情報だけを報告します。デフォルトのアクションは、すべてのフラグを指定するのと同等です。

files を指定しないと、標準出力が使われ、ファイル名は表示されません。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連概念:

63 ページの『cut - ファイルの各行から選択したフィールドを切り取る』

関連タスク:

64 ページの『grep - パターンに対応するファイルを検索する』

76 ページの『tr - 文字を変換する』

DB2 Universal Database 用ユーティリティー

DB2 Universal Database™ 用ユーティリティーを表示するには、このリンクを選択します。

関連情報:

db2profc - DB2 SQLJ プロファイル・カスタマイザー

db2profp - SQLJ プロファイルの DB2 カスタマイズ済みバージョンを印刷する

profconv - SQLJ 直列化プロファイル・インスタンスを Java クラスに変換する

profdb - SQLJ プロファイル監査プログラム・インストーラー

profp - SQLJ プロファイルを印刷する

sqlj - Structured Query Language for Java (SQLJ) 変換プログラム

Qshell db2 ユーティリティー

db2 ユーティリティーは SQL CLI (コール・レベル・インターフェース) を使用しており、このユーティリティーを使用して SQL ステートメントを直接実行したり、対話式に実行したり、ファイルから実行したりできます。

SQL を対話式に処理するかファイルから処理する際には、db2 ユーティリティーは行の末尾の円記号文字を継続文字として扱います。円記号や改行文字は除去され、残りのテキストが SQL ステートメントとして使用されます。

構文

db2 [*General Options*] [*Delimiter Options*] [*Connection Options*] [*SQL Source Options*]

一般オプション

-v SQL ステートメントを標準出力にエコー表示します。

-S 出力中のスペースと埋め込みを抑止します。テキスト・データを含む LOB 列を表示する場合に便利です。

区切り文字オプション

以下の 1 つのみ指定できます。

-T, *character*
指定された文字が終了文字として使用されます。

-t セミコロンをステートメント終了文字として使用します。

-d 感嘆符 (!) を終了文字として使用します。

接続オプション

-r *rdbname*
指定されたリモート・データベース (WRKRDBDIRE で名前指定しなければならない) に接続します。指定されていない場合は、ローカル・データベースに対する接続になります。

-u *username*
リモート・データベースへの接続に関するユーザー・プロファイル名。-r オプションと併用する場合のみ使用できます。

-p *username*
リモート・データベース接続で使用するパスワード。

SQL ソース・オプション

SQL Statement
SQL ステートメントのテキスト。ステートメントにスペースかシェル文字が含まれている場合は、Qshell コマンド行で正しく引用符で囲まれているか確認してください。

-f *filename default_lib*
指定されたファイルから SQL ステートメントを読み取って実行します。default_lib パラメーターはオプションです。指定されている場合は、このパラメーターがすべてのステートメントのデフォルト・ライブラリー/スキーマとして使用されます。

-i 対話式に SQL ステートメントを入力します。対話式 SQL セッションを終了するには、quit または exit を入力します。

特殊文字およびコマンドのサポート

- 先頭が 2 つのダッシュ (--) の行はコメント
- 先頭が感嘆符の行は Qshell コマンド
- 先頭が 'at' 記号 (@) の行は CL コマンド
- -r オプションを指定しないと、connect コマンドは無視され、ユーティリティーはローカル接続を使用する
- echo コマンドは、db2 ユーティリティーに組み込まれており、テキストをエコー表示するコマンド
- exit または quit コマンドは db2 SQL セッションを終了する
- help および ? コマンドは基本ヘルプをリストする
- terminate コマンドは無視される

例

```
db2 select constraint_name from qsys2.syscst
db2 -t -f mysqlfile.txt
```

mysqlfile.txt の内容は以下のとおりです。

```
select constraint_name from qsys2.syscst;
create table qgpl.testtable (c1 integer);
```

Perl ユーティリティー

Perl ユーティリティーを使用すると、システム上で Perl スクリプトを実行できます。 Perl ユーティリティーはフリーウェアとして使用できます。

このユーティリティーのダウンロードと使用法について詳しくは、System i Web サイトの、 DB2 for i5/OS: Qshell, Perl, and DB2 for i5/OS のトピックを参照してください。

ファイルおよびディレクトリー処理用のユーティリティー

ファイルおよびディレクトリー処理用のユーティリティーを表示するには、このリンクを選択します。

attr - ファイルの属性を取得または設定する

構文

attr **[** **-hp** **]** *file* **[** *attribute* **[** =*value* **]** ... **]**

説明

attr ユーティリティーは、*file* で指定されたオブジェクトの属性を取得または設定します。 *attribute* を指定しないと、**attr** は、再入力可能なフォーマットで、オブジェクトのすべての属性を標準出力に表示します。 *attribute* を指定すると、**attr** は属性の値を標準出力に表示します。 *attribute* と *value* を指定すると、**attr** は属性を *value* の値に設定します。すべての属性を表示できるものの、設定できるのはその一部に過ぎないという点に注意してください。

日付および時刻属性の場合、デフォルトで表示される値は asctime() 関数でフォーマット設定されます。日付および時刻を別のフォーマットで表示するには、LC_TIME 環境変数を、希望するフォーマットを定義するロケールのパスに設定します。そのロケールの LC_TIME セクションにある d_t_fmt キーワードが定義するフォーマットで、日付および時刻が表示されるようになります。ロケールのソース表示およびロケールの作成について詳しくは、ロケール・プログラミングの例を参照してください。例: export -s LC_TIME=/QSYS.LIB/EN_US.LOCALE

オプション

-h シンボリック・リンクが指すオブジェクトではなく、シンボリック・リンクの属性を表示または設定します。

-p *attribute* を再入力可能なフォーマットで表示します。

オペランド

file オペランドには、オブジェクトへのパス名を指定します。 *attribute* オペランドは、次の値を取ることができます。

ACCESS_TIME
オブジェクトが最後にアクセスされた日付および時刻。この属性は表示のみが可能です。

ALLOC_SIZE
オブジェクトのために割り振られたバイト数 (32 ビット数として表示)。この属性は表示のみが可能です。

ALLOC_SIZE_64
オブジェクトのために割り振られたバイト数 (64 ビット数として表示)。この属性は表示のみが可能です。

ALWCKPWRT
save-while-active (アクティブ中保存) チェックポイント処理中に、ストリーム・ファイルを読み取り機能および書き込み機能と共有できるかをどうか示す標識。この属性は表示または設定が可能です。

ALWSAV
オブジェクトを保管できるかどうかを示す標識。この属性は表示または設定が可能です。

ASP オブジェクトが格納されている補助記憶域プール。この属性は表示のみが可能です。

AUDIT オブジェクトと関連した監査値。この属性は表示のみが可能です。

AUTH_GROUP
オブジェクトの 1 次グループのユーザー・プロファイル名。この属性は表示のみが可能です。

AUTH_LIST_NAME
オブジェクトを保護するために使用される権限リストの名前。この属性は表示のみが可能です。

AUTH_OWNER
オブジェクトの所有者のユーザー・プロファイル名。この属性は表示のみが可能です。

AUTH_USERS
オブジェクトを使用する許可があるユーザー・プロファイルのリスト。この属性は表示のみが可能です。

CCSID オブジェクトのコード化文字セット ID (CCSID)。この属性は表示または設定が可能です。

CHANGE_TIME
オブジェクトのデータまたは属性の最終変更日時。この属性は表示のみが可能です。

CHECKED_OUT
オブジェクトがチェックアウトされるかどうかを示す標識。この属性は表示のみが可能です。

CHECKED_OUT_USER
オブジェクトをチェックアウトするユーザー・プロファイル。この属性は表示のみが可能です。

CHECKED_OUT_TIME
オブジェクトがチェックアウトされた日付および時刻。この属性は表示のみが可能です。

CODEPAGE
オブジェクトのコード化文字セット ID (CCSID) から派生するコード・ページ。この属性は表示または設定が可能です。

CREATE_TIME
オブジェクトが作成された日付および時刻。この属性は表示のみが可能です。

CRTOBJAUD
ディレクトリーに関連した作成オブジェクト監査値。監査値は、ディレクトリーで作成されるすべてのオブジェクトに指定されます。この属性は表示または設定が可能です。

CRTOBJSCAN
出口プログラムがいずれかの統合ファイル・システム・スキャン関連の出口点に登録されるときに、ディレクトリーで作成されるオブジェクトがスキャンされるかどうかを示す標識。この属性は表示または設定が可能です。

DATA_SIZE
オブジェクト内のデータのサイズのバイト数 (32 ビット数で表示)。この属性は表示のみが可能です。

DATA_SIZE_64
オブジェクト内のデータのサイズのバイト数 (64 ビット数で表示)。この属性は表示のみが可能です。

DIR_FORMAT
ディレクトリー・オブジェクトのフォーマットの標識。この属性は表示のみが可能です。

DISK_STG_OPT
オブジェクトのためにシステムによって補助記憶域がどのように割り振られるかを示す標識。この属性は表示または設定が可能です。

EXTENDED_ATTR_SIZE
オブジェクトの拡張属性用に使用されるバイト数。この属性は表示のみが可能です。

FILE_FORMAT
ストリーム・ファイルのフォーマット。この属性は表示のみが可能です。

FILE_ID
オブジェクトが、"root" (/)、QOpenSys、またはユーザー定義のファイル・システムのうち、どれに格納されているかを示すオブジェクトのファイル ID。この属性は表示のみが可能です。

JOURNAL_APPLY_CHANGES
トランザクションを完了するためにジャーナル処理済み変更の適用 (APYJRNCHG) コマンドを必要とする部分的なトランザクションで、オブジェクトが復元されたかどうかを示す標識。この属性は表示のみが可能です。

JOURNAL_ID
ジャーナルに関連するコマンドと API で使用可能なジャーナル ID。この属性は表示のみが可能です。

JOURNAL_LIBRARY
オブジェクトがジャーナル処理されている場合、現在使用されているジャーナルを含むライブラリー。オブジェクトがジャーナル処理されていない場合、最後に使用されたジャーナルを含むライブラリー。この属性は表示のみが可能です。

JOURNAL_NAME
オブジェクトがジャーナル処理されている場合、現在使用されているジャーナルの名前。オブジェクトがジャーナル処理されていない場合、最後に使用されたジャーナルの名前。この属性は表示のみが可能です。

JOURNAL_OPTIONS
現行のジャーナル処理オプション。この属性は表示のみが可能です。

JOURNAL_RCVR_ASP
ジャーナル・レシーバーを含むライブラリー用の ASP の名前。この属性は表示のみが可能です。

JOURNAL_RCVR_LIBRARY
ジャーナル・レシーバーを含むライブラリーの名前。この属性は表示のみが可能です。

JOURNAL_RCVR_NAME
ジャーナル処理済み変更の適用 (APYJRNCHG) が成功するために必要な最も古いジャーナル・レシーバーの名前。この属性は表示のみが可能です。

JOURNAL_ROLLBACK_ENDED
トランザクションのロールバック要求が完了する前に、オブジェクトがロールバックを終了したかどうかを示す標識。この属性は表示のみが可能です。

JOURNAL_START_TIME
オブジェクトのジャーナル処理が最後に開始された日付と時刻。この属性は表示のみが可能です。

JOURNAL_STATUS
オブジェクトが現在ジャーナル処理されているかどうかを示す標識。この属性は表示のみが可能です。

LOCAL_REMOTE
オブジェクトがローカル・システムにあるか、それともリモート・システムにあるかを示す標識。この属性は表示のみが可能です。

MAIN_STG_OPT
オブジェクトのためにシステムによって主記憶域がどのように割り振られ使用されるかを示す標識。この属性は表示または設定が可能です。

MODIFY_TIME
オブジェクトのデータの最終変更日時。この属性は表示のみが可能です。

MULT_SIGS
オブジェクトに複数の i5/OS ディジタル署名があるかどうかを示す標識。この属性は表示のみが可能です。

OBJTYPE
オブジェクトのタイプを記すテキスト・ストリング。この属性は表示のみが可能です。

PC_ARCHIVE
前回ファイルを調べた後、オブジェクトに変更があったかを示す標識。この属性は表示または設定が可能です。

PC_HIDDEN
オブジェクトが隠されているかどうかを示す標識。この属性は表示または設定が可能です。

PC_READ_ONLY
オブジェクトが読み取り専用であるかどうかを示す標識。この属性は表示または設定が可能です。

PC_SYSTEM
オブジェクトがシステム・オブジェクトであるかどうかを示す標識。この属性は表示または設定が可能です。

RSTDRNMUNL
ディレクトリー内のオブジェクトに対して名前変更およびリンク解除が制限されているかどうかを示す標識。オブジェクトをこの属性が設定されているディレクトリー内にリンクすることができますが、適切な権限がない場合、そこから名前変更およびリンク解除することはできません。この属性は表示または設定が可能です。

SCAN
出口プログラムがいずれかの統合ファイル・システム・スキャン関連の出口点に登録されるときに、オブジェクトがスキャンされるかどうかを示す標識。この属性は表示または設定が可能です。

SCAN_BINARY
オブジェクトが、以前にスキャンされたときに、バイナリー・モードでスキャンされているかどうかを示す標識。この属性は表示のみが可能です。

SCAN_CCSID1
オブジェクトがテキスト・モードでスキャンされている場合に、以前にスキャンされたときに使用された最初の CCSID。この属性は表示のみが可能です。

SCAN_CCSID2
オブジェクトがテキスト・モードでスキャンされている場合に、以前にスキャンされたときに使用された 2 番目の CCSID。この属性は表示のみが可能です。

SCAN_SIGS_DIFF
オブジェクト用のスキャンの署名が全体的なスキャンの署名と異なるかどうかを示す標識。この属性は表示のみが可能です。

SCAN_STATUS
オブジェクトのスキャン状況。この属性は表示のみが可能です。

SGID 実行時に有効グループ ID が設定されているかどうかを示す標識。この属性は表示または設定が可能です。

SIGNED
オブジェクトに i5/OS ディジタル署名があるかどうかを示す標識。この属性は表示のみが可能です。

STG_FREE
データがオフラインで移動するかを示す標識。この属性は表示のみが可能です。

SUID 実行時に有効ユーザー ID が設定されているかどうかを示す標識。この属性は表示または設定が可能です。

SYSTEM_ARCHIVE
オブジェクトに変更点があって、保管が必要かどうかを示す標識。この属性は表示または設定が可能です。

SYSTEM_USE
オブジェクトがシステムに特殊な方法で使用されるかどうかを示す標識。この属性はストリーム・ファイルのみで有効です。この属性は表示のみが可能です。

SYS_SIGNED
i5/OS ディジタル署名がシステムが信頼するソースからのものかどうかを示す標識。この属性は表示のみが可能です。

UDFS_DEFAULT_FORMAT
ユーザー定義のファイル・システムで作成されたストリーム・ファイルのデフォルトのファイル・フォーマット。この属性は表示のみが可能です。

USAGE_DAYS_USED
オブジェクトが使用された日数。この属性は表示のみが可能です。

USAGE_LAST_USED_TIME
オブジェクトが最後に使用された日付および時刻。この属性は表示のみが可能です。

USAGE_RESET_TIME
オブジェクトの使用日数カウントがゼロにリセットされた日付および時刻。この属性は表示のみが可能です。

環境変数

attr は、次の環境変数の影響を受けます。

LANG LC_ で始まる変数を使って明示的に設定されていないロケール・カテゴリー用のデフォルト値を提供します。

LC_TIME
日付および時刻属性の出力フォーマットを定義します。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. ファイルのすべての属性を表示します。

   ```
   attr script.sh
   ```

2. ファイルの OBJTYPE および PC_READ_ONLY 属性を表示します。

   ```
   attr script.sh OBJTYPE PC_READ_ONLY
   ```

3. 再入力可能なフォーマットでファイルの DATA_SIZE_64 属性を表示します。

   ```
   attr -p script.sh DATA_SIZE_64
   ```

4. ファイルに PC_HIDDEN 属性を設定します。

   ```
   attr script.sh PC_HIDDEN=1
   ```

関連タスク:

131 ページの『setccsid - ファイルの CCSID 属性を設定する』

135 ページの『touch - ファイルのアクセス時刻および変更時刻を変更する』

関連情報:

Qp0lGetAttr() - Get attributes

Qp0lSetAttr() - Set attributes

basename - パス名のディレクトリー以外の部分を戻す

構文

basename *string* [*suffix*]

説明

basename を使うと、*string* 内の最後のスラッシュで終わるプレフィックス、および *suffix* (指定した場合) を削除することができます。 結果のファイル名は標準出力に書き込まれます。 *string* の処理には、次の規則が使用されます。

- *string* 全体がスラッシュ文字から成る場合は、単一のスラッシュ文字が標準出力に書き込まれ、処理が終了します。
- *string* 内に後書きスラッシュ文字がある場合、そのスラッシュは削除されます。
- *string* 内に残っているスラッシュ文字がある場合は、最後のスラッシュ文字 (最後のスラッシュを含む) までの *string* のプレフィックスが削除されます。
- *suffix* が指定されていて、*string* 内の残りの文字には一致せず、*string* 内の残りの文字のサフィックスに一致する場合は、そのサフィックスが削除されます。 その他の場合は、*string* は変更されません。 *string* 内で *suffix* が検出されないのは、エラーではありません。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. シェル変数 FOO を "trail" に設定する場合 :

```
FOO=$(basename /usr/bin/trail)
```

2. パス "/usr/bin/this_test" の最後の部分、"test" サフィックスを削除して戻す場合 :

```
basename /usr/bin/this_test test
```

関連タスク:

100 ページの『dirname - パス名のディレクトリー部分を戻す』

cat - ファイルを連結して出力する

構文

cat [-bcensStuv] [-] [*file* ...]

説明

cat ユーティリティーは、指定ファイル を順次に読み取って、標準出力に書き込みます。 file オペランドは、コマンド行に指定された順序に従って処理されます。 単一のダッシュは標準入力を表します。

デフォルトでは、*file* がテキスト・データとして **cat** で読み取られるため、データはファイルの CCSID から変換されます。 **-c** オプションが指定されると、**cat** はファイルをバイナリー・データとして読み取ります。

出力のリダイレクトに使用されるシェル言語のメカニズムが原因で、コマンド **cat** `file1 file2 > file2` を指定すると file2 の元のデータは破棄されることに注意してください。さらに、プロセスが無限ループに入ります。

オプション

-b 空白行を除く出力行に番号を付けます。

-c データは読み取ったままの状態で、変換しません。

-e 出力行に番号を付け、また各行の終わりにドル記号 ($) を表示します。

-n 出力行に、1 から始まる番号を付けます。

-s 複数の空の隣接行を狭めて、出力をシングル・スペースにします。

-S 複数の空の隣接行を狭めて、出力をシングル・スペースにします。

-t 非印刷文字を表示し、**-v** オプションのように見えるようにします。またタブ文字も表示します。

-u 出力がバッファーに入らないようにします。

-v 非印刷文字を表示し、見えるようにします。制御文字は "^X" (制御文字) のように出力されます。削除文字は "^?" のように出力されます。 . 非表示文字は、"M-x" (メタ文字) のように出力されます。大部分のロケールでは、すべての文字が表示できることに注意してください。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイルの内容を表示する場合 : "myfile"

   ```
   cat myfile
   ```

2. 3 つの異なるファイルの内容を同時に表示し、その内容を 1 つの新しいファイルに保存する場合 :

   ```
   cat file1 file2 file3 > all.files
   ```

関連タスク:

107 ページの『head - ファイルの先頭部分をコピーする』
132 ページの『tail - ファイルの末尾部分を表示する』
138 ページの『zcat - データを拡張および連結する』
『catsplf - スプール・ファイルを連結して出力する』
115 ページの『od - 様々なフォーマットのファイルをダンプする』
124 ページの『pr - ファイルを出力する』

catsplf - スプール・ファイルを連結して出力する

構文

catsplf -j *qualified-job* **[-aen]** *splfname splfnum*

catsplf -p *pid* **[-aen]** *splfname splfnum*

説明

catsplf ユーティリティーは指定されたスプール・ファイルを読み取り、標準出力に書き込みます。

最初の構文形式では、**catsplf** は *qualified-job* によって指定されたジョブに関連したスプール・ファイルを検索します。

2 番目の構文形式では、**catsplf** は *pid* によって指定されたジョブに関連したスプール・ファイルを検索します。

オプション

-a 指定されたジョブに関連するすべてのスプール・ファイルを出力します。

-e 出力行に 1 から始まる番号を付け、各行の終わりにドル記号 ($) を表示します。

-j *qualified-job*
qualified-job が *number/user/name* 形式のストリングである場合に、 *qualified-job* によって識別されるジョブに関連したスプール・ファイルを検索します。 *number* は 6 桁の 10 進数であり、*user* はジョブが開始されたユーザー・プロファイルであり、*name* はジョブの名前です。

-n 出力行に、1 から始まる番号を付けます。

-p *pid* *pid* がジョブの 10 進プロセス ID である場合に、 *pid* によって識別されるジョブに関連したスプール・ファイルを検索します。

オペランド

splfname オペランドはスプール・ファイルの名前を指定し、 *splfnum* オペランドはスプール・ファイルの番号を指定します。両方のオペランドが、スプール・ファイルを一意的に識別するのに必要です。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. QPRINT と名付けられ、修飾ジョブ名を使用するジョブの番号 1 のスプール・ファイルを出力します。

   ```
   catsplf -j 386687/SHELLTST/QZSHCHLD QPRINT 1
   ```

2. QPRINT と名付けられ、pid を使用するジョブの番号 1 のスプール・ファイルを出力します。

   ```
   catsplf -p 942 QPRINT 1
   ```

3. ジョブのすべてのスプール・ファイルを出力します。

   ```
   catsplf -a -j 386687/SHELLTST/QZSHCHLD
   ```

関連タスク:

88 ページの『cat - ファイルを連結して出力する』

128 ページの『Rfile - レコード・ファイルの読み書きをする』

138 ページの『zcat - データを拡張および連結する』

cd - 作業ディレクトリーを変更する

構文

cd [*directory*]

説明

cd を使うと、作業ディレクトリーを変更できます。 **qsh** は、**PWD** 変数を新規作業ディレクトリーに設定し、**OLDPWD** 変数を前の作業ディレクトリーに設定します。

オプション

なし。

オペランド

directory には、次のものを指定できます。

- (負符号)
: **qsh** は、作業ディレクトリーを前のディレクトリーに変更して、新しい作業ディレクトリー名を表示します。

/name **または** *../name*
: **qsh** は、作業ディレクトリーを指定された *name* に変更します。

name **(最初の文字が / でも ../ でもない名前)**
: **CDPATH** 変数を設定してあると、**qsh** は **CDPATH** の中の各ディレクトリーを *name* の前に付けて、ディレクトリー名を作成します。 **qsh** は、コマンド発行者に許可された最初のディレクトリーに変わります。 **qsh** は、新しい作業ディレクトリー名を表示します。

  CDPATH 変数を設定していない場合は、 **qsh** は、作業ディレクトリーを指定された *name* に変更します。

何も指定しない。
: **qsh** は、作業ディレクトリーを **HOME** 変数の値に変更します。

指定する *directory* に対する許可をコマンド発行者が持っていることが必要です。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

154 ページの『hash - ユーティリティー・ロケーションを記憶または報告する』
127 ページの『pwd - 作業ディレクトリー名を戻す』
127 ページの『pwdx - 展開された作業ディレクトリーを出力する』

chgrp - ファイル・グループの所有権を変更する

構文

chgrp [-R [-H | -L | -P]] [-h] *group file ...*

説明

chgrp を使うと、*file* のグループを *group* で指定されているグループ ID またはプロファイルに設定できます。

グループ ID を変更するには、次のいずれかの権限が必要です。

- 現行ユーザーには *ALLOBJ 特殊権限がある。
- 現行ユーザーが *file* の所有者であり、また次のいずれかである。
  - ジョブの基本グループが *group*。
  - ジョブの補足グループのうちの 1 つが *group*。

さらに、現行ユーザーには *group* で指定されたグループ・プロファイルに対する *USE 権限が必要です。

デフォルトでは **chgrp** はシンボリック・リンクをたどって、シンボリック・リンクが指すファイルのグループを変更します。

-R オプションを指定しない限り、 **-H**、**-L**、および **-P** の各オプションは無視されます。また、これらのオプションは互いに上書きしあうので、コマンドのアクションは最後に指定されたオプションによって決まります。

ファイルのグループを、ファイルの所有者と同じにすることはできません。

オプション

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。

-R *file* がディレクトリーである場合、そこに連結されている全サブツリー中の各ファイルのグループが **chgrp** によって繰り返し変更されます。

-h シンボリック・リンクが指すファイルではなく、シンボリック・リンクの所有者とグループを変更します。

オペランド

group オペランドには、グループ ID 番号またはグループ・プロファイル名のいずれかを指定します。 *file* オペランドには、オブジェクトへのパス名を指定します。

終了状況

- 0 正常終了し、要求されたすべての変更が行われた。
- >0 エラー発生。

例

1. ファイル "newgui.java" で、グループをグループ・プロファイル "abbey" に変更する。

   ```
   chgrp abbey newgui.java
   ```

2. サブディレクトリー "personal.dir" およびその下のすべてのファイルとサブディレクトリーで、グループをグループ・プロファイル "managers" に変更する。

   ```
   chgrp -R managers personal.dir
   ```

3. ファイル "memo.txt" で、グループをグループ ID "442" に変更する。

   ```
   chgrp 442 memo.txt
   ```

関連タスク:

『chmod - ファイル・モードを変更する』

95 ページの『chown - ファイルの所有権を変更する』

109 ページの『ls - ディレクトリーの内容をリスト表示する』

chmod - ファイル・モードを変更する

構文

chmod [-R [-H | -L | -P]] [-h] *mode file ...*

説明

chmod ユーティリティーは、*file* のファイル・モード・ビットを *mode* オペランドに指定された値に変更します。

ファイルのモードを変更するには、次のいずれかの権限が必要です。

- 現行ユーザーには *ALLOBJ 特殊権限がある。
- 現行ユーザーがそのファイルの所有者である。

デフォルトでは、**chmod** はシンボリック・リンクをたどっていって、シンボリック・リンクが指すファイルのモードを変更します。シンボリック・リンクにはモードがありません。したがって、シンボリック・リンク上で **chmod** を使用すると常に正常終了しますが実際には無効です。

-R オプションを指定しない限り、 **-H**、**-L**、および **-P** の各オプションは無視されます。また、これらのオプションは互いに上書きしあうので、コマンドのアクションは最後に指定されたオプションによって決まります。

chmod はオブジェクトに対する i5/OS のデータ権限を変更することに注意してください。CHGAUT CL コマンドを使用して、オブジェクトに対する i5/OS のオブジェクト権限を変更します。

オプション

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。シンボリック・リンクにはモードがないので、 **chmod** はシンボリック・リンク上で無効です。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。シンボリック・リンクにはモードがないので、 **chmod** はシンボリック・リンク上で無効です。

-R *file* にディレクトリーが指定されている場合、その時点で連結されている全サブツリー中の各ファイルのモードが **chmod** によって変更されます。

-h シンボリック・リンクはたどられません。シンボリック・リンクにはモードがないので、 **chmod** はシンボリック・リンク上で無効です。

オペランド

mode は絶対モードでもシンボリック・モードでも構いません。絶対モードは、次の値を OR 結合して得られる 3 または 4 桁の 8 進数です。

4000 実行ビット上にユーザー ID を設定

2000 実行ビット上にグループ ID を設定

1000 ディレクトリーの制限付き削除ビット

0400 所有者による読み取り許可

0200 所有者による書き込み許可

0100 所有者による実行または検索許可

0040 グループによる読み取り許可

0020 グループによる書き込み許可

0010 グループによる実行または検索許可

0004 その他のユーザーによる読み取り許可

0002 その他のユーザーによる書き込み許可

0001 その他のユーザーによる実行または検索許可

シンボリック・モードの構文規則は以下のようになります。

- *mode* ::= clause [, clause ...]
- *clause* ::= [who ...] [action ...] last_action
- *action* ::= op [perm ...]
- *last_action* ::= op [perm ...]
- *who* ::= a | u | g | o
- *op* ::= + | - | =
- *perm* ::= r | w | x | X | s | t | u | g | o

who 記号には、以下のように許可が付与または拒否されているユーザーが指定されます。

u 所有者許可ビット

g グループ許可ビット

o その他のユーザーの許可ビット

a 所有者、グループ、その他のユーザーの許可ビット。これは、**ugo** 記号を一緒に指定することと同じです。

op 記号は、以下のように実行される操作を表します。

+ 指定された許可を付与します。 *perm* に値が指定されていない場合は、"+" 操作は無効です。 *who* に値が指定されていない場合に、 *perm* に指定されている各許可ビットに対して、ファイル・モード作成マスク中の対応するビットがクリアされていれば、その許可ビットが設定されます。それ以外の場合は、*who* および *perm* の指定値で表されるモード・ビットが設定されます。

- 指定された許可を拒否します。 *perm* に値が指定されていない場合は、"-" 操作は無効です。 *who* に値が指定されていない場合に、 *perm* に指定されている各許可ビットに対して、ファイル・モード作成マスク中の対応するビットがクリアされていれば、その許可ビットがクリアされます。 それ以外の場合は、*who* および *perm* の指定値で表されるモード・ビットがクリアされます。

= 選択された許可フィールドをクリアし、そこに指定された許可を設定します。 *who* 値で指定されたモード・ビットがクリアされているか、または *who* 値が指定されていない場合は、所有者、グループ、およびその他のモード・ビットがクリアされます。その後、*who* に値が指定されない場合は、*perm* に指定されている各許可ビットに対応するファイル・モード作成マスクがクリアされていれば、その許可ビットが設定されます。それ以外の場合は、*who* および *perm* の指定値で表されるモード・ビットが設定されます。

perm 記号は、以下のようにモード・ビットの一部を表します。

r 読み取りビット。

w 書き込みビット。

x 実行/検索ビット。

X 対象ファイルがディレクトリーの場合、またはオリジナル (変更前の) モードで実行/検索ビットが設定されている場合は、実行/検索ビット。この記号による操作が有効になるのは *op* 記号 "+" と併用した場合のみで、その他の場合はすべて無視されます。

s 所有者許可ビットが設定されている場合は、実行ビット上にユーザー ID を設定し、グループ許可ビットが設定されている場合は実行ビット上にグループ ID を設定します。

t オブジェクトがディレクトリーの場合の制限付き削除ビット。 *who* 記号が **a** であるか、または *who* 記号がない場合に使用できます。ファイルがディレクトリーでないか、*who* 記号が **u**、**g**、または **o** である場合、無視されます。

各 *clause* には、モード・ビットに対して実行する 1 つまたは複数の操作が指定されます。各操作は指定された順序でモード・ビットに適用されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. 絶対モードを使用して、所有者に対する読み取りおよび書き込み許可を認可し、グループおよびその他のユーザーに対する読み取り許可を認可する場合:

   ```
   chmod 644 myfile
   ```

2. グループおよびその他のユーザーに対する書き込み許可を否定する場合:

```
chmod go-w myfile
```

3. 現在設定されているすべての許可をクリアし、所有者、グループ、およびその他のユーザーに対する読み取りおよび書き取り許可を認可する場合:

   ```
   chmod =rw myfile
   ```

4. ディレクトリー上での検索許可が所有者、グループ、およびその他のユーザーのいずれか 1 つに対して設定されている場合に、所有者、グループ、およびその他のユーザーに対する検索許可を認可する場合:

   ```
   chmod +X mydir
   ```

5. 絶対モードを使用して、所有者に対する読み取り、書き込み、および実行許可を認可し、グループおよびその他のユーザーに対する読み取りおよび実行許可を認可する場合:

   ```
   chmod 755 myfile
   ```

6. グループおよびその他のユーザーのすべての許可をクリアする場合:

   ```
   chmod go= myfile
   ```

7. グループ許可を所有者許可と等しくなるよう設定するが、グループに対する書き込み許可を否定する場合:

   ```
   chmod g=u-w myfile
   ```

8. 絶対モードを使用して、実行ビット上にユーザー ID を設定し、所有者に対する読み取り、書き込み、および実行許可を認可し、その他のユーザーに対する実行許可を認可する場合:

   ```
   chmod 4701 myfile
   ```

関連タスク:

91 ページの『chgrp - ファイル・グループの所有権を変更する』

chown - ファイルの所有権を変更する

構文

chown [-R [-H | -L | -P]] [-h] *owner*[*:group*] *file ...*

説明

chown を使うと、*file* の所有者を *owner* で指定されているユーザー ID またはプロファイルに設定できます。 オプションで、**chown** を使って *file* のグループを *group* で指定されているグループ ID またはプロファイルに設定することもできます。

ファイルの所有者を変更するには、次のいずれかの権限が必要です。

- 現行ユーザーには *ALLOBJ 特殊権限がある。
- 現行ユーザーがそのファイルまたはディレクトリーの所有者である。

ファイル・グループを変更するには、次のいずれかの権限が必要です。

- 現行ユーザーには *ALLOBJ 特殊権限がある。
- 現行ユーザーが *file* の所有者であり、また次のいずれかである。
  - ジョブの基本グループが *group*。
  - ジョブの補足グループのうちの 1 つが *group*。

さらに、現行ユーザーには新規ユーザー・プロファイルまたはグループ・プロファイルに対する *USE 権限が必要です。

デフォルトでは **chown** はシンボリック・リンクをたどって、シンボリック・リンクが指すファイルの所有者とグループを変更します。

ファイルのグループを、ファイルの所有者と同じにすることはできません。

オプション

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。

-R *file* にディレクトリーが指定されている場合、その時点で連結している全サブツリー中の各ファイルの所有者およびグループが **chown** によって反復して変更されます。

-h シンボリック・リンクが指すファイルではなく、シンボリック・リンクの所有者とグループを変更します。

オペランド

owner オペランドには、ユーザー ID 番号またはユーザー・プロファイル名のいずれかを指定します。 *group* オペランドには、グループ ID 番号またはグループ・プロファイル名のいずれかを指定します。 *file* オペランドには、オブジェクトへのパス名を指定します。

終了状況

- 0 正常終了し、要求されたすべての変更が行われた。
- >0 エラー発生。

例

1. ファイル "personal.file" の所有者をユーザー・プロファイル "sam" に変更する。

   ```
   chown sam personal.file
   ```

2. サブディレクトリー "moe.dir" とその下のすべてのファイルとサブディレクトリーで、所有者をユーザー・プロファイル "larry" に変更する。

   ```
   chown -R larry moe.dir
   ```

3. ファイル "your.file" の所有者をユーザー ID "500" に変更する。

   ```
   chown 500 your.file
   ```

4. ファイル "memo.txt" の所有者をユーザー・プロファイル "sam" に変更し、グループをグループ・プロファイル "abbey" に変更する。

   ```
   chown sam:abbey memo.txt
   ```

関連タスク:

91 ページの『chgrp - ファイル・グループの所有権を変更する』

109 ページの『ls - ディレクトリーの内容をリスト表示する』

131 ページの『setccsid - ファイルの CCSID 属性を設定する』

compress - データを圧縮する

構文

compress [-cfv] [-b *bits***]** [*file* ...]

説明

compress ユーティリティーは、適応する Lempel-Ziv コーディングを使って *file* のサイズを縮小します。どの *file* も、拡張子「.Z」を付加された名前に変更されます。許可で認められたのと同じ数の変更時刻、アクセス時刻、ファイル・フラグ、ファイル・モード、ユーザー ID、およびグループ ID が新しいファイル内に保存されます。圧縮してもサイズが縮小されないファイルの場合、*file* は無視されます。

file の名前を変更するとファイルが上書きされてしまう場合に、標準入力装置が端末であると、標準エラーを確認するよう指示するプロンプトが表示されます。プロンプトを出せない場合や確認が得られない場合、ファイルは上書きされません。

オプション

-b *bits* ビット・コード限度を指定します (詳細は以下を参照してください)。

-c 圧縮済みの出力が標準出力に書き込まれます。ファイルは変更されません。

-f 実際に圧縮されなくても、*file* を強制的に圧縮します。しかも、確認を求めないままファイルは上書きされます。

-v 各 *file* の縮小パーセントを出力します。

オペランド

各 *file* は、圧縮されるファイルのパス名です。 *files* を指定しない場合、標準入力は標準出力に圧縮されます。入力ファイルも出力ファイルも正規のファイルでない場合、サイズの縮小とファイルの上書きの検査は実行されず、入力ファイルは除去されず、入力ファイルの属性は保存されません。

補足説明

compress ユーティリティーでは、修正された Lempel-Ziv アルゴリズムが使われます。ファイル内の共通サブストリングは、最初に 9 ビット・コード 257 以上に置き換えられます。このアルゴリズムは、コード 512 に達すると 10 ビット・コードに切り替え、 **-b** フラグで指定された限度 (デフォルトは 16) に達するまでさらにビットを使用しつづけます。ビットは 9 - 16 でなければなりません。

ビット限度に達した後は、圧縮率が **compress** によって定期的に検査されます。それが増えていると **compress** は既存のコード辞書を使用し続けます。これに対して圧縮率が減っていると **compress** はサブストリング・テーブルを破棄してから、スクラッチからそのテーブルを再作成します。こうしてアルゴリズムは、ファイルの次の「ブロック」を適合させられるようになります。

実現できる圧縮量は、入力のサイズ、コードあたりのビット数、および共通サブストリングの配布によって異なります。通常、ソース・コードや英語などのテキストは 50 - 60% 縮小されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

117 ページの『pax - ポータブル・アーカイブを交換する』

137 ページの『uncompress - 圧縮データを圧縮解除する』

138 ページの『zcat - データを拡張および連結する』

cp - ファイルをコピーする

構文

cp [-r | -R [-H | -L | -P]] [-fhipt] *source_file target_file*

cp [-r | -R [-H | -L | -P]] [-fhipt] *source_file ... target_directory*

説明

最初の構文形式では、**cp** ユーティリティーによって *source_file* の内容が *target_file* にコピーされます。

2 番目の構文形式では、**cp** ユーティリティーによって各 *source_file* の内容が宛先の *target_directory* にコピーされます。ファイルの名前自体は変更されません。 ディレクトリーである *source_file* が 1 つだけ指定されていて、さらに **-R** フラグが指定されている場合を除いて、*target_directory* が存在していなければなりません。

ファイルをそのファイル自体にコピーしようとすると、 **cp** のコピー操作は失敗します。

target_file が存在していない場合は、*target_file* の作成時に、ファイル作成マスクで変更された *source_file* のモードが使用されます。新規ファイルの作成時には、S_ISUID および S_ISGID ファイル許可ビットが設定されません。

target_file がすでに存在していて **-t** オプションが指定されていない場合は、その内容は 2 進データとして上書きされ、CCSID 属性は *source_file* の CCSID 属性と一致するように変更されます。 *target_file* のファイル許可ビット、所有者、およびグループは変更されません。 **-t** オプションを使用すると、データを強制的にテキスト・データとしてコピーできます。 **-p** オプションによって、ファイル許可ビット、所有者、およびグループを強制的にコピーできます。

QSYS.LIB ファイル・システムのメンバーをコピーする場合は、*source_file* の多くの属性が保持されないことに注意してください。それらは、メンバーではなくオブジェクトに関連しているからです。

-h オプションが指定されているか、**-H** または **-L** オプションと共に **-R** オプションが指定されていない限り、常にシンボリック・リンクに従います。 **-R** オプションを指定しない限り、 **-H**、**-L**、および **-P** の各オプションは無視されます。また、これらのオプションは互いに上書きしあうので、コマンドのアクションは最後に指定されたオプションによって決まります。

オプション

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に発見されたシンボリック・リンクには従わず、シンボリック・リンクの指すファイルの代わりにそのシンボリック・リンクがコピーされます。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。シンボリック・リンクの指すファイルの代わりに、ツリー走査中に発見されたシンボリック・リンクがコピーされます。

-R *source_file* によってディレクトリーが指定される場合は、 **cp** では、そのディレクトリーと、その時点で連結しているサブツリー全体がコピーされます。 このオプションを使用すると、**cp** は特殊

ファイルを通常のファイルとしてコピーするのではなく、特殊ファイルを作成します。作成されたディレクトリーのモードは、ファイル作成マスクによって変更されていないときの、対応するソース・ディレクトリーのモードと同じになります。

-f *target_file* が書き込み走査用に開けない場合にそれを削除します。データがコピーされる前に新規ファイルが作成されます。

-h シンボリック・リンクが指すファイルの代わりに、シンボリック・リンクをコピーします。

-i 既存のファイルを上書きするファイルをコピーする前に、標準エラーにプロンプトを出します。標準入力からの応答が現行ロケールにおける YES 応答の先頭文字で始まっている場合は、ファイルのコピーが行われます。

-p 許可の範囲内の数の変更時刻、アクセス時刻、ファイル許可ビット、所有者、グループがコピー中に保持されます。

所有者とグループを保持できない場合でも、エラー・メッセージは表示されず、終了値も変更されません。

S_ISUID および S_ISGID ファイル許可ビットがコピーされるのは、ファイルの所有者とグループの両方が正常にコピーされる場合のみです。

注:	QSYS.LIB ファイル・システムへのコピーを行う場合には、このオプションは無効です。

-r **-R** と同じですが、特殊ファイルを正規ファイルと同じようにコピーするという点が異なります。**-R** フラグは **-r** フラグより優先されます。

-t target_file が存在する場合には、*source_file* 中のデータをテキスト・データとして処理し、そのデータのコピー時に *target_file* に関連する CCSID へ変換します。 *target_file* の CCSID 属性は変更されません。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイル file1 をサブディレクトリー data.dir にコピーする場合:

   ```
   cp file1 data.dir
   ```

2. code サブディレクトリー内にあって拡張子が .java であるすべてのファイルを、サブディレクトリー code/old_code.dir にコピーし、サブディレクトリー code/old_code.dir の中に同じファイルがすでにある場合に限り、ユーザーに上書きの確認を求めるプロンプトを表示する場合 :

   ```
   cp -i code/*.java code/old_code.dir
   ```

関連タスク:

108 ページの『ln - ファイルをリンクする』

109 ページの『ls - ディレクトリーの内容をリスト表示する』

114 ページの『mv - ファイルを移動する』

130 ページの『rm - ディレクトリー項目を削除する』

131 ページの『rmdir - ディレクトリーを削除する』

136 ページの『umask - ファイル・モード作成マスクを入手または設定する』

dirname - パス名のディレクトリー部分を戻す

構文

dirname *string*

説明

dirname を使用すると、最後のスラッシュ文字 (/)から始まり *string* の最後で終わるファイル名部分を削除し、結果を標準出力に書き込むことができます。 *string* の処理には、次の規則が使用されます。

- *string* 全体がスラッシュ文字から成る場合は、単一のスラッシュ文字が標準出力に書き込まれ、処理が終了します。
- *string* 内に後書きスラッシュ文字がある場合、そのスラッシュは削除されます。
- *string* 内にスラッシュ文字が残っていない場合は、ピリオドが標準出力に書き込まれ、処理が終了します。
- *string* 内にスラッシュ以外の後書き文字がある場合、その文字は削除されます。
- *string* 内に後書きスラッシュ文字がある場合、そのスラッシュは削除されます。
- 残りのストリングが空の場合は、*string* は 1 つのスラッシュ文字に変換されます。

オペランド

string オペランドはパス名であり、このパス名のディレクトリー部分が **dirname** によって戻されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

シェル変数 FOO を /usr/bin に設定する場合 :

```
FOO=$(dirname /usr/bin/trail)
```

関連タスク:

87 ページの『basename - パス名のディレクトリー以外の部分を戻す』

file - ファイル・タイプを判別する

構文

file [-m *MagicFile*] **[-f** *ListFile*] [*file* ...]

file [-c] [-m *MagicFile*]

説明

最初の構文形式では、指定された *file* のオブジェクト・タイプが **file** ユーティリティーによって判別されます。 **file** ユーティリティーは、最も近いと思われるタイプを判別します。次にそのファイル・タイプは標準出力に書き込まれます。パス名が通常のファイルであると判断されると、 **file** は最初の 1024 バイトを調べてタイプを判別します。デフォルトでは、**file** ユーティリティーは /etc/magic ファイルを使って、オブジェクト内に指定されたバイト・オフセットで定義されているパターンをもつファイルを特定します。

2 番目の構文形式では **file** ユーティリティーは、指定された *MagicFile* にフォーマット・エラーがないかどうかを調べます。

オプション

-c 指定された MagicFile にフォーマット・エラーがあるかどうかを調べます。

-f *ListFile*
テスト対象のファイル名のリストが入っているファイルを指定します。この *ListFile* では、1 行に 1 ファイルのみ指定可能で、ファイル名の前後にスペースを付加することはできません。

-m *MagicFile*
使用する MagicFile の名前を指定します。デフォルトの MagicFile は /etc/magic です。

オペランド

各 *file* は、テストされるファイルのパス名です。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

『find - ファイルを検索する』

find - ファイルを検索する

構文

find [-H | -L | -P] [-Xdx] [-f *file***]** *file ...* [*expression*]

説明

find ユーティリティーは、指定された各 *file* に対するディレクトリー・ツリーを再帰的に下降しながら、ツリー内の各ファイルへの *expression* (以下に示す「プライマリー」と「オペランド」から構成される) を評価します。

オプション

-H コマンド行上のシンボリック・リンクに従って操作が実行されます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。コマンド行に指定された各シンボリック・リンクについては、リンクによって参照されるファイルに関する情報とファイル・タイプが戻されます。参照されているファイルが存在しない場合は、リンク自体のファイル情報およびタイプが戻されます。

-L コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。コマンド行に指定された各シンボリック・リンクごとに、リンクで参照されるファイルに関する情報とファイル・タイプが戻されます。参照されているファイルが存在しない場合は、リンク自体のファイル情報およびタイプが戻されます。

-P シンボリック・リンクには従いません。各シンボリック・リンクについては、リンク自体のファイル情報とファイル・タイプが戻されます。

-X **find** を **xargs** と共に安全に使用できるように変更します。 **xargs** で使用される区切り文字のどれかがファイル名に入っている場合は、標準エラーに診断メッセージが表示され、そのファイルはスキップされます。区切り文字には、一重引用符 (') と二重引用符 (")、円記号 (¥)、スペース文字、タブ文字、および改行文字があります。

-d **find** の走査は下位レベル優先で行われます。ディレクトリーは後順で走査され、ディレクトリー内のすべての項目の処理の終了後、ディレクトリー自体が処理されます。デフォルト設定では、 **find** は、ディレクトリーを先順で (それぞれの内容より先に) 走査します。デフォルト値は横方向優先の走査ではないので注意してください。

-f **find** を使って走査するファイル階層を指定します。ファイル階層は、オプション群の直後のオペランドとして指定することもできます。

-x **find** は、下降の起点になるファイルとは装置番号が異なるディレクトリーの中は検索しません。

プライマリー

-atime *n*

ファイルに最後にアクセスした時刻と **find** が開始した時刻との時間差を 24 時間単位で切り上げた値が、24 時間の *n* 倍である場合は、真になります。

-ctime *n*

ファイル状況情報を最後に変更した時刻と **find** が開始された時刻との時間差を 24 時間単位で切り上げた値が、24 時間の *n* 倍である場合は、真になります。

-exec *utility* **[***argument* **...] ;**

utility に指定したプログラムが戻り状況として値 0 を戻した場合は、真になります。指定したユーティリティーには、オプションの引数を渡すことができます。式はセミコロン (;) で終わっていなければなりません。ユーティリティー名または引数のどこかにストリング {} がある場合は、現行ファイルのパス名で置き換えられます。 ユーティリティーは **find** の実行が開始されたディレクトリーから実行されます。セミコロンもシェルに対して特殊文字であるので、引数として **find** に渡されるようにセミコロンをエスケープすることが必要な場合があります。

-group *gname*

ファイルがグループ *gname* に属している場合に、真になります。 *gname* が数値であり、それに該当するグループ名がない場合は、*gname* はグループ ID と見なされます。

-inum *n*

ファイルの inode 番号が *n* の場合に、真になります。

-links *n*

ファイルのリンク数が *n* 個の場合には、真になります。

-ls このプライマリーは、常に真として評価されます。標準出力に書き出される現行ファイル情報は以下のとおりです。

- inode 番号
- サイズ (KB)
- ファイル許可
- ハード・リンク数
- 所有者
- グループ
- サイズ (バイト)
- 最後の変更時刻
- パス名

ファイルがブロック特殊ファイルまたは文字特殊ファイルである場合は、バイト単位のサイズではなく、メジャー番号とマイナー番号が表示されます。ファイルがシンボリック・リンクである場合は、リンク先ファイルのパス名の前に `->' が表示されます。

-mtime *n*

ファイルの最後の変更時刻と **find** が開始された時刻との時間差を 24 時間単位で切り上げた値が 24 時間の *n* 倍である場合は、真になります。

-ok *utility* **[***argument***...] ;**

-ok プライマリーは、**find** が標準エラーにメッセージを出力しその応答を読み取ることによって、*utility* 実行の承認をユーザーに要求することを除いて、**-exec** と同じです。応答が現行ロケールにおける YES 応答の先頭文字以外である場合は、*utility* は実行されず、ok 式の値は偽になります。

-name *pattern*

評価対象のパス名の最後の構成要素が *pattern* に一致した場合は、真になります。特殊シェル・パターン照合文字 ([、]、*、および ?) を *pattern* の一部として使用できます。エスケープ文字である円記号 (¥) を付けてこれらの文字を指定すれば、本来の文字として明示指定して照合することができます。

-newer *file*

現行ファイルの最後の変更時刻が *file* の変更時刻よりも新しい場合に、真になります。

-nouser

ファイルが不明なユーザーに属している場合に、真になります。

-nogroup

ファイルが不明なグループに属している場合に、真になります。

-path *pattern*

評価対象のパス名が *pattern* に一致した場合に、真になります。特殊シェル・パターン照合文字 ([、]、*、および ?) を *pattern* の一部として使用できます。エスケープ文字である円記号 (¥) を付けてこれらの文字を指定すれば、本来の文字として明示指定して照合することができます。スラッシュ (/) は通常の文字と見なされるので、明示指定して照合する必要はありません。

-perm [-]*mode*

mode は記号でも chmod コマンドでサポートされる 8 進数形式でも構いません。 *mode* が記号の場合、開始値は 0 と見なされ、プロセス・ファイル・モード作成マスクに関係なく、許可を設定またはクリアします。 *mode* が 8 進数の場合は、ファイルのモード・ビット 00777 (S_IRWXU | S_IRWXG | S_IRWXO) だけが比較に使用されます。 *mode* の前にダッシュ (-) が付いている場合は、少なくともモード内のすべてのビットがファイルのモード・ビット内で設定されているときに、このプライマリーは真として評価されます。 *mode* の前にダッシュが付いていない場合は、*mode* 内のビットがファイルのモード・ビットと一致しているときに、このプライマリーは真として評価されます。シンボリック・モードの最初の文字は、ダッシュ (-) であってはならないという点に注意してください。

-print このプライマリーは、常に真として評価されます。現行ファイルのパス名がこのプライマリーによって標準出力に表示されます。 **-exec**、**-ls**、**-ok** のどれも指定されていない場合、式はユーザー指定の式に追加されます。

-prune このプライマリーは、常に真として評価されます。これは、**find** が現行ファイルの中まで下降しないようにします。 **-d** オプションを指定した場合は、**-prune** プライマリーは無効になることに注意してください。

-size *n[c]*

512 バイト・ブロック単位に切り上げたファイル・サイズが *n* の場合に、真になります。 *n* の後に *c* が続く場合は、ファイル・サイズが *n* バイトの場合に、このプライマリーは真になります。

-type *t* ファイルが指定したタイプの場合に、真になります。可能なファイル・タイプは次のとおりです。

- b ブロック特殊

- c 文字特殊
- d ディレクトリー
- f 正規ファイル
- l シンボリック・リンク
- p FIFO
- s ソケット

-user *uname*
ファイルがユーザー *uname* に属している場合に、真になります。 *uname* が数値であり、それに該当するユーザー名がない場合には、*uname* はユーザー ID と見なされます。

数値引数をとるすべてのプライマリーは、前に正符号 (+) または負符号 (-) を付けることができます。先行正符号は「n より大」を意味し、先行負符号は「n より小」を意味します。どちらも「n に等しい」の意味にはなりません。

演算子

プライマリーは、次の演算子と組み合わせて使用できます。演算子は、優先順位の高いものから順に列記してあります。

(expression)
これは、括弧で囲まれた式が真として評価された場合に、真になります。

!expression
これは単項 NOT 演算子です。式が偽の場合に、真と評価されます。

expression -and expression
-and 演算子は論理 AND 演算子です。この演算子は、2 つの式を並置すれば暗黙に指定されるので、明示的に指定しなくても構いません。式は、両方の式が真である場合に、真と評価されます。第 1 の式が偽である場合は、第 2 の式は評価されません。

expression -or expression
-or 演算子は論理 OR 演算子です。式は、第 1 または第 2 のいずれかの式が真であれば、真と評価されます。 第 1 の式が真である場合は、第 2 の式は評価されません。

すべてのオペランドおよびプライマリーは、**find** ユーティリティーに対して個別の引数である必要があります。それ自身が引数をとるプライマリーでは、各引数がそれぞれ **find** にとって別個の引数であるものと見なされます。**注**

find で使用される特殊文字は、多くのシェル・コマンドにとっても特殊文字です。特に、*、 [、]、?、(、)、!、および ; の各文字は、シェルからエスケープしておくことが必要な場合があります。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ディレクトリー /project/java/class から始めて、すべての *.class ファイルを検索する場合 :
   ```
   find /project/java/class -name '*.class'
   ```
2. ディレクトリー /project/java/code から始めて、ストリング import java.awt; を含むすべての *.java ファイルを検索する場合 :

```
find /project/java/code -name '*.java' -exec grep 'import java.awt;' {} ¥;
```

3. ディレクトリー /project/java/class から始めて、すべての *.class ファイルを検索し、ファイルを除去する場合 :

```
find /project/java/class -name '*.class' -exec rm {} ¥;
```

4. ディレクトリー /project から始めて、ユーザー abbey に属するすべてのファイルを検索する場合 :

```
find /project -user abbey
```

関連概念:

100 ページの『file - ファイル・タイプを判別する』

関連タスク:

60 ページの『xargs - 引数リストを作成してユーティリティーを起動する』

92 ページの『chmod - ファイル・モードを変更する』

gencat - フォーマットされたメッセージ・カタログを生成する

構文

gencat **[-C** *ccsid***]** **[-m** *mode* **]** **[-t** *text***]** *catfile* *msgfile* ...

説明

gencat ユーティリティーは、メッセージ・テキスト・ソース・ファイル *msgfile* から、フォーマットされたメッセージ・カタログ *catfile* を生成します。 300 までのメッセージ・テキスト・ソース・ファイルを指定できます。メッセージ・テキスト・ソース・ファイルは指定された順序で処理されます。連続したソース・ファイルがそれぞれカタログを変更します。ソース・ファイルのメッセージ番号がメッセージ・カタログにすでに存在している場合、ソース・ファイルで定義されている新規メッセージ・テキストがメッセージ・カタログ・ファイルにある古いメッセージ・テキストと置き換えられます。ソース・ファイルのメッセージ番号がメッセージ・カタログにまだ存在していない場合、メッセージ情報がメッセージ・カタログに追加されます。

オプション

-C *ccsid*
: メッセージ・カタログを作成し、指定した *ccsid* でメッセージ・テキストを保管します。

-m *mode*
: メッセージ・カタログのファイル許可ビットを、指定した *mode* に設定します。 mode 引数は、chmod コマンドでサポートされているどのフォーマットでも指定できます。シンボリック・モードを指定した場合は、演算文字 + および - は、 a=rw の初期モードを基準として解釈されます。

-t *text*
: 指定した *text* をメッセージ・カタログ・オブジェクトに割り当てます。テキストのオブジェクトへの割り当ては、メッセージ・カタログに使用するファイル・システムまたはオブジェクト・タイプによって提供されるサポートに依存しています。

オペランド

catfile オペランドは、変更または作成するメッセージ・カタログへのパスを指定します。 **-m** オプションが指定されない場合、メッセージ・カタログは、現行のファイル作成マスクで変更したデフォルト・モード (所有者、グループ、およびその他に読み書きのアクセス権を許可するモード: 0666) を使用して作成されます。

各 *msgfile* は、入力メッセージ・テキスト・ソース・ファイルへのパスを指定します。メッセージ・テキスト・ソース・ファイルは 300 が限度です。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. 1 つのメッセージ・テキスト・ソース・ファイルを使用してメッセージ・カタログを作成する場合:

   ```
   gencat product.cat msg.src
   ```

2. 複数のメッセージ・テキスト・ソース・ファイルを使用してメッセージ・カタログを作成する場合:

   ```
   gencat product.cat msg1.src msg2.src msg3.src
   ```

3. メッセージ・カタログを作成し、モードおよび ccsid を設定する場合:

   ```
   gencat -C 37 -m a-w product.cat msg.src
   ```

関連タスク:

92 ページの『chmod - ファイル・モードを変更する』

139 ページの『dspmsg - メッセージ・カタログのメッセージを表示する』

getconf - 構成値を取得する

構文

getconf [*name* [*pathname*]]

説明

getconf ユーティリティーは POSIX 構成変数を表示します。 *name* を指定すると、**getconf** は、構成変数の値を標準出力に表示します。構成変数がパス名に応じて決まる場合は、*pathname* を指定する必要があります。

引数を何も指定しなかった場合は、**getconf** は、すべての構成変数とその値のリストを表示します。パス名に応じて決まる構成変数の場合は、**getconf** は / を使って表します。

オプション

なし。

オペランド

name に指定できる値は次のいずれかです。

CCSID
: 統合ファイル・システムのパス名に内部使用されるデフォルトのコード化文字セット ID (CCSID) を表します。

CHOWN_RESTRICTED
: *pathname* によって表されるオブジェクトに対して **chown** を使用できるのは、適切な特権を持っているジョブに限られます。

CLK_TCK
: 1 秒間に刻むクロック目盛りの数。

LINK_MAX
pathname によって表されるオブジェクトが持つ最大リンク数。

NAME_MAX
ファイル名 (パス名の最後のコンポーネント) の最大バイト数。

NGROUPS_MAX
ジョブに関連付けることができる補足グループ ID の最大数。

NO_TRUNC
ファイル名が NAME_MAX より長い場合にエラーが生成されます。

OPEN_MAX
1 つのジョブ内で同時にオープンできる最大ファイル数。

PAGE_SIZE
システム・ハードウェアのページ・サイズを表します。

PAGESIZE
システム・ハードウェアのページ・サイズを表します。

PATH_MAX
完全パス名の最大バイト数。

PIPE_BUF
一度にパイプに書き込むことのできる最大バイト数。

STREAM_MAX
1 つのジョブ内で同時にオープンできる最大ストリーム数。

THREAD_SAFE
pathname で表されるオブジェクトはスレッド・セーフ・ファイル・システムに入っています。

終了状況

- 0 正常終了。
- >0 成功の場合。

例

1. ディレクトリー /home がスレッド・セーフ・ファイル・システムに入っているかどうかを判別する場合:

   ```
   getconf THREAD_SAFE /home
   ```

2. ファイル名の最大バイト数を表示する場合:

   ```
   getconf NAME_MAX
   ```

3. 構成変数をすべて表示する場合:

   ```
   getconf - 構成値を取得する
   ```

head - ファイルの先頭部分をコピーする

構文

head [-n *count*] [*file* ...]

説明

head ユーティリティーは、指定した個々のファイルまたは (ファイルが指定されていない場合は) 標準出力について、 *count* に指定された数の先頭行を表示します。 **-n** を指定しなかった場合は、ファイルの最初の 10 行が表示されます。

複数の *file* を指定した場合は、各 *file* の前に、==> XXX <== の形式のストリングから成るヘッダーが付けられます。XXX はファイルの名前です。

オプション

-n *count* に指定された数の行を表示します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

ファイル myfile の最初の 20 行を表示する場合 :

```
head -n 20 myfile
```

関連タスク:

88 ページの『cat - ファイルを連結して出力する』

132 ページの『tail - ファイルの末尾部分を表示する』

ln - ファイルをリンクする

構文

ln [-fs] *source_file* [*target_file*]

ln [-fs] *source_file ...* [*target_dir*]

説明

ln ユーティリティーは、オリジナルのファイルと同じモードの新しいディレクトリー項目 (リンク・ファイル) を作成します。これは、コピー用に記憶域を使用せずに同一ファイルの複数コピーを同時に多数の箇所で保守する場合に便利です。ストレージを使用するかわりに、リンクによってオリジナル・コピーを「指し示し」ます。リンクには、ハード・リンクとシンボリック・リンクの 2 つのタイプがあります。ハード・リンクとシンボリック・リンクの違いの 1 つは、そのリンクがファイルをどのように「指し示す」かという点にあります。

デフォルト設定では、**ln** ではハード・リンクが作成されます。ファイルへのハード・リンクは、オリジナルのディレクトリー項目とは見分けがつきません。ファイルに対する変更は、どの名前を使ってそのファイルを参照するかに関係なく、すべて有効になります。 ハード・リンクは、通常は複数のディレクトリーを参照せず、複数のファイル・システムにまたがることはできません。

シンボリック・リンクには、リンク先のファイルの名前が入っています。シンボリック・リンクは、複数のファイル・システムにまたがることができ、また複数のディレクトリーを参照することもできます。

ln に 1 つまたは 2 つの引数を指定すると、既存ファイル *source_file* へのリンクが作成されます。*target_file* を指定した場合は、リンクにその名前が付きます。 *target_file* は、リンクを入れるディレクトリーの場合もあります。 ディレクトリーを指定しなければ、リンクは現行ディレクトリーに入れられます。ディレクトリーのみを指定した場合は、*source_file* の最後の構成要素へのリンクが作成されます。

ln に 3 つ以上の引数を指定すると、*target_dir* の中に、指定したすべてのソース・ファイルへのリンクが作成されます。 作成されるリンクは、それぞれのリンク先のファイルと同じ名前になります。

オプション

-f 既存のファイルのリンクを解除して、リンクを作成できるようにします。

-s シンボリック・リンクを作成します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイル /usr/bin/perl5 からファイル /usr/bin/perl へのシンボリック・リンクを作成する場合 :

   ```
   ln -s /usr/bin/perl5 /usr/bin/perl
   ```

2. ファイル /usr/bin/qsh からファイル /bin/qsh への新しいリンクを作成し、ファイル /bin/qsh への既存のリンクがあれば、それを解除する場合 :

   ```
   ln -f /usr/bin/qsh /bin/qsh
   ```

関連タスク:

98 ページの『cp - ファイルをコピーする』
『ls - ディレクトリーの内容をリスト表示する』
114 ページの『mv - ファイルを移動する』
130 ページの『rm - ディレクトリー項目を削除する』
131 ページの『rmdir - ディレクトリーを削除する』

ls - ディレクトリーの内容をリスト表示する

構文

ls [-ACFLRSTacdfiloqrstu1] [*file* ...]

説明

file に指定するオペランドの中にタイプがディレクトリー以外のものがあった場合は、**ls** は、要求された関連情報に加えてその名前も表示します。 タイプがディレクトリーであるファイルを指定したオペランドについては、**ls** は、要求された関連情報に加えて、そのディレクトリーに置かれているファイルの名前を表示します。

オペランドをまったく指定しなかった場合は、現行ディレクトリーの内容が表示されます。複数のオペランドを指定した場合は、ディレクトリー以外のオペランドが最初に表示されます。ディレクトリー・オペランドとディレクトリー以外のオペランドは、別個のグループとしてアルファベット順にソートされます。

オプション

-A 「.」と「..」を除くすべての項目がリスト表示されます。

-C 複数列の出力を強制します。これは、出力先が端末装置の場合のデフォルト値です。

-F ディレクトリーを示す各パス名の後ろにはスラッシュ (/)、実行可能ファイルを示すパス名の後ろにはアスタリスク (*)、各シンボリック・リンクの後ろにはアットマーク (@) が表示されます。

-L 引数がシンボリック・リンクである場合に、リンク自体ではなく、そのリンクによって参照されるファイルまたはディレクトリーがリスト表示されます。

-R サブディレクトリーが再帰的にリスト表示されます。

-S ファイルの CCSID 属性が表示されます。または、ファイルをサイズ別に分類し表示します。振る舞いは環境変数 QIBM_ZSH_LS_SORT_BY_SIZE によって異なります。

-T **-l** オプションも指定した場合に、ファイルに関する完全な日時情報が表示されます。これには、月、日、時、分、秒、および年が含まれます。

-a ドット (.) で始まる名前を持つディレクトリー項目も含めて表示されます。.

-c ソートまたは出力の際に、ファイルの状況が最後に変更されたときの時刻が使用されます。

-d ディレクトリーがプレーン・ファイルとしてリスト表示され (再帰的に検索されない)、引数リスト内のシンボリック・リンクは間接参照されません。

-f 出力はソートされません。

-i 各ファイルについて、ファイルのシリアル番号 (i ノード番号) が出力されます。

-l (小文字の L) 長形式でリスト表示されます。詳細については、このあとの『補足説明』を参照してください。 出力先が端末である場合は、長形式のリストの前に、すべてのファイルのサイズの合計が 1 行に出力されます。

-o 長形式 (**-l**) 出力にファイル・フラグが含まれます。

-q ファイル名の中の図形文字以外の文字を疑問符 (?) で表示します。これは、出力先が端末である場合のデフォルト値です。

-r ソート順序を反転させ、逆アルファベット順、または古い項目から新しい項目への順で表示します。

-s 各ファイルに実際に割り振られているバイト数が、1024 バイトを 1 単位として表示されます。部分的に使用されている単位は、すぐ上の整数値に切り上げられます。

-t オペランドをアルファベット順にソートする前に、変更時刻でソートします (変更日時の最も新しいものが最初になります)。

-u ファイルの最後の変更時刻ではなく、最後のアクセス時刻を使用して、ソート (-t) または表示 (-l) されます。

-1 (数字の 1 です。) 1 行に 1 項目ずつ出力します。これは、出力先が端末でない場合のデフォルト値です。

-1、**-C**、および **-l** の各オプションは、互いに一時変更します。使用されるフォーマットは、最後に指定したオプションによって決まります。

-c および **-u** オプションは、互いに一時変更します。 使用されるファイル時刻は、最後に指定したオプションによって決まります。

デフォルト設定では、**ls** は、1 行に 1 項目ずつ標準出力にリスト表示されます。ただし、出力先が端末の場合、または **-C** オプションを指定した場合は、例外です。

ファイル情報を表示するとき、**-i**、 **-s**、**-l**、および **-S** オプションに関連した情報は、それぞれ 1 つまたは複数のブランクで区切られます。

補足説明

-l オプションを指定した場合は、各ファイルごとに次の長形式情報が表示されます。

- ファイル・モード
- リンク数
- 所有者名
- グループ名
- ファイルのバイト数
- ファイルが最後に変更された時刻
- パス名

現在の日付からさかのぼって 6 か月以内にファイルが変更されていた場合、日時は省略形で月、日、時刻および分が表示されます。それ以外の場合、日時は月、日、および 4 桁の年で表示されます。

さらに、内容が表示される各ディレクトリーごとに、そのディレクトリー内のファイルで使用されているバイトの合計数が、そのファイルに関する情報のすぐ前の行に単独で表示されます。

所有者名またはグループ名が、既知のユーザー名またはグループ名ではない場合は、数値 ID が表示されます。

ファイルが文字特殊ファイルまたはブロック特殊ファイルである場合は、そのファイルのメジャーおよびマイナーの両方の装置番号がサイズ・フィールドに表示されます。ファイルがシンボリック・リンクである場合は、リンク先ファイルのパス名の前に、-> が表示されます。

ファイル・モードは、項目タイプ、所有者許可、グループ許可、およびその他の許可で構成されます。項目タイプ文字は、次のようにファイルのタイプを示します。

- b ブロック特殊ファイル
- c 文字特殊ファイル
- d ディレクトリー
- l シンボリック・リンク
- p パイプ
- s ソケット
- - 正規ファイル

所有者許可、グループ許可、およびその他の許可は、それぞれ 3 文字です。 各フィールドに、それぞれ 3 個の文字位置があります。

- 最初の位置の値が r の場合は、そのファイルは読み取り可能です。その値が - の場合は、読み取り可能ではありません。
- 2 番目の位置の値が w の場合は、そのファイルは書き込み可能です。その値が - の場合は、書き込み可能ではありません。
- 3 番目の位置については次のとおりです。
  - 所有者許可の値が S の場合、ユーザー ID の設定モードが設定されます。グループ許可の値が S の場合、グループ ID の設定モードが設定されます。
  - 所有者許可の値が s の場合、そのファイルは実行可能で、ユーザー ID の設定モードが設定されます。グループ許可の値が s の場合、そのファイルは実行可能で、グループ ID の設定モードが設定されます。
  - 値が x の場合、ファイルが実行可能であるか、ディレクトリーが検索可能です。

– 値が - の場合、オブジェクトが実行不能であるか、検索不能です。

環境変数

ls は、次の環境変数の影響を受けます。

COLUMNS
この変数に 10 進整数を表すストリングが入っている場合は、その数値は、複数テキスト列の出力を表示するときの列幅として使用されます。 **ls** ユーティリティーは、与えられた幅に基づいて、表示するパス名テキスト列の数を計算します。 **-C** オプションを参照してください。

QIBM_ZSH_LS_SORT_BY_SIZE
ls -S を表示する ls ユーティリティーにより生じる結果を制御するのに、この環境変数を設定します。変数の値が '1' の場合、ls -S はファイルをサイズ別に分類し表示します。それ以外の場合、ls -S はファイルの CCSID 属性を表示します。デフォルト値はありません。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. 現行ディレクトリー内のファイルのリストを長形式で表示する場合 :

```
ls -l
```

2. ファイル myfile の日付と時刻に関するすべての詳細情報を表示する場合 :

```
ls -lT myfile
-rwxrwxrwx  1 abbey  0      592 Sep 12 22:47:01 1998 myfile
```

関連タスク:

91 ページの『chgrp - ファイル・グループの所有権を変更する』
92 ページの『chmod - ファイル・モードを変更する』
95 ページの『chown - ファイルの所有権を変更する』
98 ページの『cp - ファイルをコピーする』
108 ページの『ln - ファイルをリンクする』
『mkdir - ディレクトリーを作成する』
114 ページの『mv - ファイルを移動する』
130 ページの『rm - ディレクトリー項目を削除する』
131 ページの『rmdir - ディレクトリーを削除する』

mkdir - ディレクトリーを作成する

構文

mkdir [-p] [-m *mode***]** *directory ...*

説明

mkdir ユーティリティーは、現行のファイル作成マスクで変更したモード rwxrwxrwx (0777) を使用して、オペランドに指定された名前のディレクトリーを指定した順序で作成します。

ユーザーは、親ディレクトリーへの書き込み許可を持っている必要があります。

オプション

-m 作成された最後のディレクトリーのファイル許可ビットを、指定した *mode* に設定します。 mode 引数は、chmod コマンドでサポートされているどのフォーマットでも指定できます。シンボリック・モードを指定した場合は、演算文字 + および - は、 a=rwx の初期モードを基準として解釈されます。

-p 必要に応じて中間ディレクトリーが作成されます。このオプションを指定しない場合は、各オペランドの絶対パス・プレフィックスがすでに存在していなければなりません。中間ディレクトリーの許可は、現行のファイル作成マスクで変更されたとおりの rwxrwxrwx (0777) の許可ビットで決まります。所有者の場合は、これに書き込み許可と検索許可が加わります。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

ディレクトリー new、java、test、dir、4、および bob を作成し、モードを所有者用の読み取り、書き込み、および実行に設定する場合 :

```
mkdir -p -m 700 /new/java/test/dir/4/bob
```

関連タスク:

92 ページの『chmod - ファイル・モードを変更する』

109 ページの『ls - ディレクトリーの内容をリスト表示する』

131 ページの『rmdir - ディレクトリーを削除する』

136 ページの『umask - ファイル・モード作成マスクを入手または設定する』

『mkfifo - FIFO 特殊ファイルを作成する』

mkfifo - FIFO 特殊ファイルを作成する

構文

mkfifo [-p] [-m *mode*] *file ...*

説明

mkfifo ユーティリティーは、オペランドに指定された名前の FIFO 特殊ファイルを、指定された順序で作成します。そのときには、現行のファイル作成マスクで変更したデフォルト・モード (所有者、グループ、およびその他に読み書きのアクセス権を許可するモード: 0666) を使用します。

ユーザーは、親ディレクトリーへの書き込み許可を持っている必要があります。

オプション

-m *mode*
FIFO 特殊ファイルのファイル許可ビットを、指定した *mode* に設定します。 mode 引数は、chmod コマンドでサポートされているどのフォーマットでも指定できます。シンボリック・モードを指定した場合は、演算文字 + および - は、 a=rw の初期モードを基準として解釈されます。

-p 必要に応じて中間ディレクトリーが作成されます。このオプションを指定しない場合は、各 *file* の絶対パス・プレフィックスがすでに存在していなければなりません。中間ディレクトリーは、現行

のファイル作成マスクで変更されたとおりのデフォルト・モード (所有者、グループ、そしてその他に読み取り、書き込み、および検索のアクセス権を許可するモード: 0777) で作成されます。

オペランド

それぞれの *file* は、FIFO 特殊ファイルのパス名です。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. FIFO 特殊ファイル "fifo1" および "fifo1" を作成する場合:

   ```
   mkfifo fifo1 fifo2
   ```

2. FIFO 特殊ファイル "fifo1" を作成し、所有者に読み取り、書き込み、および実行許可を設定する場合:

   ```
   mkfifo -m 700 myfifo
   ```

3. FIFO 特殊ファイル "/dir1/dir2/fifo1" と、そのパスに含まれている、存在していない各ディレクトリーを作成する場合:

   ```
   mkfifo -p /dir1/dir2/fifo1
   ```

関連タスク:

92 ページの『chmod - ファイル・モードを変更する』
112 ページの『mkdir - ディレクトリーを作成する』
136 ページの『umask - ファイル・モード作成マスクを入手または設定する』

mv - ファイルを移動する

構文

mv [-f | -i] *source_file target_file*

mv [-f | -i] *source_file ... target_dir*

説明

最初の形式では、**mv** ユーティリティーは *source_file* オペランドで指定されたファイル名を、*target_file* オペランドで指定された宛先パス名に変更します。最後のオペランドに既存のディレクトリーの名前が指定されていない場合は、この形式と見なされます。

2 番目の形式では、**mv** は *source_file* オペランドで指定されている各ファイルを、*target_dir* オペランドで指定されている既存のディレクトリー内の宛先ファイルに移動します。各 *source_file* オペランドの宛先パスは、*target_dir*、スラッシュ、*source_file* のパス名の最後の構成要素を連結したものです。

source_file オペランドと宛先パスの両方が共にディレクトリーである場合を除き、いずれかにディレクトリーを指定するとエラーになります。

宛先パスに書き込みを許可するモードがない場合、 **mv** から **-i** オプションの指定に従ってユーザーに確認を求めるプロンプトが表示されます。

オプション

-f 宛先パスを上書きする前に、確認を求めるプロンプトは表示されません。 **-f** オプションを指定した場合は、 **-i** オプションは無視されます。

-i 既存のファイルを上書きするファイルを移動する前に、標準エラーにプロンプトを出します。標準入力から応答が現行ロケールにおける YES 応答の先頭文字で始まっている場合は、移動が行われます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

ファイル perl5 をディレクトリー /usr/bin に移動し、同名のファイルが存在する場合に上書きするかどうかをユーザーに尋ねるプロンプトを表示する場合 :

```
mv -i perl5 /usr/bin
```

関連タスク:

98 ページの『cp - ファイルをコピーする』

108 ページの『ln - ファイルをリンクする』

109 ページの『ls - ディレクトリーの内容をリスト表示する』

130 ページの『rm - ディレクトリー項目を削除する』

od - 様々なフォーマットのファイルをダンプする

構文

od [-A *address_base*] **[-j** *skip*] **[-N** *count*] **[-t** *type_string*] **[-Cbcdosvx]** [*file...*]

説明

od ユーティリティーは、指定された *files* の内容をユーザー指定形式で標準出力に書き出します。 *file* パラメーターを指定しないと **od** コマンドによって標準入力が読み取られます。これのフォーマットは **-t** フラグで指定します。フォーマット・タイプを指定しない場合のデフォルトは **-t oS** です。

オプション

-A *address_base*

出力オフセット・ベースのフォーマットを指定します。 *address_base* は次のいずれかの値です。

- 10 進数の場合は **d**
- 8 進数の場合は **o**
- 16 進数の場合は **x**
- なしの場合は **n**

n の場合、オフセット・ベースは表示されません。 **-A** を指定しない場合のデフォルトは **-A o** になります。

-b バイトを 8 進数で出力します。これは、**-t 01** と同じです。

-C 残りの出力を書き出す前にファイルの CCSID を標準出力に表示します。

-c バイトを文字として出力します。これは、**-t c** と同じです。

-d バイトを符号なしの 10 進数で出力します。これは、**-t u2** と同じです。

-j *skip* 出力の表示を開始する前にスキップするバイト数を指定します。複数のファイルを指定すると、このバイト数は、指定するすべてのファイルの連結入力で使われます。その数が連結入力のサイズより大きいとエラーが起きます。この値は、16 進数 (前に 0x または 0X が付きます)、8 進数 (前に 0 が付きます)、または 10 進数 (デフォルト) で指定することができます。

-N *count*
書き出すバイト数を指定します。デフォルトでは、ファイル全体が書き出されます。この値は、16 進数 (前に 0x または 0X が付きます)、8 進数 (前に 0 が付きます)、または 10 進数 (デフォルト) で指定することができます。

-o バイトを 8 進数で出力します。これは、**-t o2** と同じです。

-s バイトを符号付き 10 進数で出力します。これは、**-t d2** と同じです。

-t *type_string*
1 つ以上の出力タイプを指定します。指定するタイプは、指定したいすべてのフォーマット・タイプの入っているストリングでなければなりません。 *type_string* には次の値を入れることができます。

- 文字の場合は **a**
- 文字の場合は **c**
- 符号付き 10 進数の場合は **d**
- 浮動小数点の場合は **f**
- 8 進数の場合は **o**
- 符号なしの 10 進数の場合は **u**
- 16 進数の場合は **x**

a および **c** のタイプを指定すると、想定外の結果を生じることがあります。これらのタイプはデータの CCSID によって異なるからです。 **a** 型指定子は、指定された文字として印刷不能文字を表示します。 **c** 型指定子は、3 桁の 8 進数として印刷不能文字を表示します。

d、**o**、**u** および **x** のタイプを指定する場合、その後に **1**、**2**、**4**、**C**、**S**、**I**、または **L** を付けることができます。それは、出力タイプの各インスタンスで変換されるバイト数を指定します。値 **C**、**S**、**I**、および **L** は char、short、int、および long に対応します。

f のタイプを指定する場合、その後に **4**、**8**、**F**、**D**、または **L** を付けることができます。それは、出力タイプの各インスタンスで変換されるバイト数を指定します。値 **F**、**D**、および **L** は float、double、および long double に対応します。 **-t** を指定しない場合、デフォルトは **-t oS** になります。

-v すべての入力データを書き出します。このオプションを指定しないと、反復出力行は書き出されません。反復が起きた場合、アスタリスク (*) だけが書き出されます。

-x バイトを 16 進数で出力します。これは、**-t x2** と同じです。

オペランド

各 *file* は、標準出力に書き出されるオブジェクトのパス名です。 *file* オペランドを指定しないと、標準入力が使われます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイルを 16 進形式でダンプする場合:

   ```
   od -tx output.txt
   ```

2. ファイルの最初の 50 バイトをダンプする場合:

   ```
   od -N50 output.txt
   ```

3. ファイルの最初の 100 バイトをスキップして、残りをダンプする場合:

   ```
   od -j100 output.txt
   ```

4. ファイルを 16 進形式と文字フォーマットの両方でダンプする場合:

   ```
   od -tx1 -tc output.txt
   ```

関連タスク:

88 ページの『cat - ファイルを連結して出力する』

124 ページの『pr - ファイルを出力する』

pax - ポータブル・アーカイブを交換する

構文

pax [-cdnv] [-E *limit*] **[-f** *archive*] **[-s** *replstr* **...] [-U** *user* **...] [-G** *group* **...] [-T** [*from_date*][,*to_date*] **...]** [*pattern* **...]**

pax -r [-cdiknuvDYZ] [-C *ccsid* **] [-E** *limit*] **[-f** *archive*] **[-o** *options* **...] [-p** *string* **...] [-s** *replstr* **...] [-U** *user* **...] [-G** *group* **...] [-T** [*from_date*][,*to_date*] **...]** [*pattern* **...]**

pax -w [-dituvHLPX] [-b *blocksize*] **[[-a] [-f** *archive*]] **[-x** *format*] **[-B** *bytes*] **[-s** *replstr* **...] [-o** *options* **...]** **[-U** *user* **...] [-G** *group* **...] [-T** [*from_date*][,*to_date*]**[/[c][m]] ...]** [*file* **...]**

pax -r -w [-diklntuvDHLPXYZ] [-p *string* **...] [-s** *replstr* **...] [-U** *user* **...] [-G** *group* **...] [-T** [*from_date*][,*to_date*]**[/[c][m]] ...] [***file* **...]** *directory*

説明

pax ユーティリティーは、アーカイブ・ファイルのメンバーを読み取り、書き込み、および一覧表示し、さらにディレクトリー階層をコピーします。 **pax** 操作は特定のアーカイブ・フォーマットに従属することはなく、多種多様のアーカイブ・フォーマットをサポートします。サポートされているアーカイブ・フォーマットのリストは、**-x** オプションの解説の項に記載されています。

-r および **-w** オプションがあると、リスト表示、読み取り、書き込み、およびコピーのうちのどの機能モードで **pax** が稼働するかが指定されます。

<none>List

pax は、指定されたパターンに一致するパス名をもったアーカイブ・ファイルから読み取ったそのファイルのメンバーの目次を書き込みます。この目次には、行ごとに 1 つずつファイル名が入っており、単一行バッファリングを使って書き込まれます。

-r 読み取り

pax は、指定された *patterns* に一致するパス名をもったアーカイブ・ファイルから読み取ったそのファイルのメンバーを抽出します。アーカイブ・フォーマットとブロック化は入力で自動的に決定されます。抽出するファイルがディレクトリーであると、そのディレクトリーをルートとするファイル階層全体が抽出されます。抽出ファイルはすべて、現在のファイル階層に対して相対的に作成

されます。抽出ファイルの所有権の設定、アクセスおよび変更の回数、およびファイル・モードについて詳しくは、 **-p** オプションの項に説明されています。

-w 書き込み

pax は、指定されたアーカイブ *format* を使って、 *file* オペランドの入ったアーカイブを標準出力に書き込みます。 *file* オペランドを指定しないと、行ごとに 1 つずつファイルの入ったコピー対象のファイル・リストが標準入力から読み取られます。 *file* オペランドがディレクトリーでもあると、そのディレクトリーをルートとするファイル階層全体が取り込まれます。

-r -w コピー

pax は *file* オペランドを宛先 *directory* にコピーします。 *file* オペランドを指定しないと、行ごとに 1 つずつファイルの入ったコピー対象のファイル・リストが標準入力から読み取られます。 *file* オペランドがディレクトリーでもあると、そのディレクトリーをルートとするファイル階層全体が取り込まれます。このコピーの場合、コピーされたファイルがアーカイブ・ファイルに書き込まれてから抽出されたのと同じ結果を生じます。ただし、オリジナル・ファイルとコピー・ファイルをつなぐハード・リンクがある可能性があることを除きます (以下の **-l** オプションを参照)。

重要:	宛先 *directory* は、*file* オペランドのうちの 1 つであったり、 file オペランドのうちの 1 つにルートをもつファイル階層のメンバーであったりしてはなりません。そのような条件下でコピーを行うと想定外の結果を生じます。

注:	アーカイブ・ファイルは、他のプラットフォームに移植できるよう、CCSID 819 でなければなりません。

pax は読み取りまたはリスト表示操作で、損傷のあるアーカイブを処理するときはメディア損傷からのリカバリーを試みてから、アーカイブ全体を通して検索を行って、できる限り多数のアーカイブ・メンバーを見つけ出して処理します (エラー処理について詳しくは、 **-E** オプションの項を参照)。

オプション

-r 標準入力からアーカイブ・ファイルを読み取って、指定された *files* を抽出します。アーカイブ・メンバーを抽出するのに中間ディレクトリーが必要な場合、モード引数として S_IRWXU、S_IRWXG、および S_IRWXO のビット単位の包含 OR を使って **mkdir** が呼び出された場合に行われるように、その中間ディレクトリーが作成されます。選択されたアーカイブの *format* が、リンクされたファイルの指定をサポートする場合に、アーカイブの抽出時にそのファイルをリンクできないときは、**pax** は操作の完了時に標準エラーに診断メッセージを書き込んで、ゼロ以外の終了状況とともに終了します。

-w 指定されたアーカイブ *format* で標準出力にファイルを書き込みます。 *file* オペランドを指定しないと、前にも後にも <ブランク> の付いていない、行ごとに 1 つずつパス名の入ったパス名リストが標準入力から読み取られます。

-a 事前に書き込まれているアーカイブの末尾にファイルを付加します。 **-x** オプションを使ってアーカイブの *format* が指定されないと、現在アーカイブ内で使われているフォーマットが選択されます。すでにアーカイブ内で使われているのと異なるフォーマットをアーカイブに付加しようとすると、 **pax** は非ゼロの状況でただちに終了します。書き込みを開始したアーカイブ・ボリューム内で使ったブロック・サイズが、そのアーカイブ・ボリュームの残りでもそのまま使われます。

-b *blocksize*

アーカイブの書き込み時に、アーカイブ・ファイルに対する書き込みあたり正数の 10 進整数のバイト数ごとに出力をブロック化します。 *blocksize* は、512 バイトの倍数でなければなりません。

また、最大値は 32256 バイトです。 *blocksize* を k または b で終了すれば、1024 (1K) または 512 の乗算を指定することができます。 *blocksizes* のペアを x で区切って積を示すことができます。ブロック化を指定しない場合のデフォルトのブロック・サイズは、使用している個々のアーカイブ・フォーマットによって異なります (**-x** オプションを参照)。

-c *pattern* および *file* オペランドで指定されたものを除き、すべてのファイルまたはアーカイブのメンバーを突き合わせます。

-d タイプがディレクトリーのファイルがコピーまたはアーカイブされるか、またはタイプがディレクトリーのアーカイブ・メンバーが抽出されて、ディレクトリー・ファイルまたはアーカイブ・メンバーのみが突き合わせられ、そのディレクトリーをルートとするファイル階層は突き合わせられません。

-f *archive*
入力または出力のアーカイブのパス名として *archive* を指定します。それによって、デフォルトの標準入力 (リスト表示または読み取りの場合) または標準出力 (書き込みの場合) がオーバーライドされます。 1 つのアーカイブが複数のファイルと様々なアーカイブ装置にまたがっていてもかまいません。必要があれば、アーカイブ内の次のボリュームのファイルまたは装置のパス名を **pax** からプロンプトで尋ねられます。

-i ファイルまたはアーカイブ・メンバーの名前を対話式で変更します。 *pattern* オペランドに一致する各アーカイブ・メンバーごとにか、または *file* オペランドに一致する各ファイルごとに、ファイルの名前、そのファイル・モード、およびその変更時刻を示すプロンプトが **pax** によって端末に表示されます。次に **pax** は端末から行を読み取ります。その行がブランクであると、そのファイルまたはアーカイブ・メンバーはスキップされます。その行が 1 つのピリオドから成る場合、そのファイルまたはアーカイブ・メンバーは、名前を変更されないで処理されます。それ以外の場合、その名前は、行の内容に置き換えられます。応答の読み取り時に EOF を検出すると **pax** は非ゼロの終了状況でただちに終了します。LC_TIME 環境変数が設定されている場合、変更時刻は、指定されたロケールの LC_TIME カテゴリーで d_t_fmt キーワードによって指定されたフォーマットでフォーマット設定されます。

-k 既存ファイルを上書きしません。

-l (英字の小文字の「エル」) ファイルをリンクします。コピー・モード (-r -w) では、可能であればソース・ファイルと宛先ファイルをつなぐハード・リンクが確立されます。

-n 各 *pattern* オペランドに一致する最初のアーカイブ・メンバーを選択します。どの *pattern* にも、複数のアーカイブ・メンバーが一致することはありません。タイプがディレクトリーのメンバーが一致すると、そのディレクトリーをルートとするファイル階層も一致します (ただし、 **-d** も同時に指定しないことを前提とします)。

-o **-x** で指定されたアーカイブ・フォーマットに固有のアーカイブ・ファイルの抽出または書き込み用のアルゴリズムを変更するための情報。一般的にオプションは、name=value という形式になります。

-p *string*
1 つ以上のファイル特性オプション (特権) を指定します。 *string* は、抽出時に保持または廃棄されるファイル特性を指定するためのストリングです。このストリングは、a、e、m、o、および p で構成されます。1 つのストリング内で複数の特性を連結することができ、複数の **-p** オプションを指定することができます。指定文字の意味は次のとおりです。

a ファイルへのアクセス時刻を保存しません。デフォルトで、ファイルのアクセス時刻は、可能であれば常に保存されます。

e ユーザー ID、グループ ID、ファイル・モード・ビット、ファイル・アクセス時刻、およ

びファイル変更時刻をすべて保存します。この用途としては、適切な特権をすべて備えたユーザーが、ファイルの属性すべてを、アーカイブに格納された状態のままで保存することを想定しています。 e フラグは、o および p フラグを合計したものになります。

m ファイル変更時刻を保存しません。デフォルトで、ファイルの変更時刻は、可能であれば常に保存されます。

o ユーザー ID とグループ ID を保存します。

p ファイル・モード・ビットを保存します。この用途としては、正規の特権を備えたユーザーが、所有権以外のファイル属性すべてを保存することを想定しています。ファイルの各時刻はデフォルトで保存されますが、他の 2 つのプラグはこれを無効にし、代わりに抽出時の時刻を使用できるよう用意されています。

上記のリストでは、保存とは、呼び出し側プロセスの許可に応じて、アーカイブに保管されている属性を抽出ファイルに与えることを指します。それ以外の場合、抽出ファイルの属性は、通常のファイル作成アクションの一環として決定されます。これらのアイテムのうちのいずれかの保存が何らかの理由で失敗した場合、標準エラーに診断メッセージが **pax** によって書き込まれます。アイテムの保存が失敗すると、最終的な終了状況は影響を受けますが、抽出ファイルの削除にはつながりません。いずれかの *string* 内のファイル特性文字が互いに重複しているかあるいは対立する場合、最後に指定したものが優先されます。たとえば、**-p eme** と指定すると、ファイル変更時刻は保存されます。

-s 正規の式の構文を使って、置換式 *replstr* に従って *pattern* または *file* オペランドで指定されたファイルまたはアーカイブ・メンバーを変更します。正規の式のフォーマットは次のとおりです。

```
/old/new/[gp]
```

old は基本の正規表現であり、new には、& 記号 (&)、n (n は数字) 逆参照、または副次式の突き合わせを指定することができます。また old ストリングには、<改行> 文字を入れることもできます。非ヌル文字 (ここでは / が示されています) を区切り文字として使用することができます。複数の **-s** 式を指定することができます。この式は、コマンド行に指定された順に適用され、置換が最初に正常に完了したら終了します。オプションのトレーラーの g は引き続き、最後に正常に完了した置換の末尾に続く最初の文字で始まるパス名サブストリングに対して、置換式を適用します。最初に失敗した置換によって g オプションの操作は停止されます。オプションのトレーラーの p は、正常に完了した置換の最終結果を次のようなフォーマットで標準エラーに書き込みます。

```
<original path name> >> <new path name>
```

空ストリングに置き換えられるファイルまたはアーカイブ・メンバーの名前は選択されず、スキップされます。

-t **pax** によって読み取られたかまたはアクセスされたファイルまたはディレクトリーのアクセス時間を、 **pax** によって読み取られるかまたはアクセスされる前と同じ値にリセットします。

-u 同一名のついた既存のファイルまたはアーカイブ・メンバーよりも古い (ファイル変更時刻の値がより小さい) ファイルを無視します。ファイル・システム内のファイルと同じ名前がアーカイブ・メンバーに付いている場合に、アーカイブ・メンバーのほうがそのファイルよりも新しいと、そのアーカイブ・メンバーが読み取り中に抽出されます。ファイル・システム・メンバーにアーカイブ・メンバーと同じ名前が付いている場合に、ファイル・システム・メンバーのほうがアーカイブ・メンバーよりも新しいと、そのファイル・システム・メンバーが書き込み中にアーカイブに書き込まれます。ソース階層内のファイルのほうが宛先階層内のファイルよりも新しい場合、コピー中に宛先階層内のファイルは、ソース階層内のファイルに置きかえられるか、またはソース階層内のファイル・リンクに置きかえられます。

-v リスト操作中に、**-l** オプションを指定した **ls** ユーティリティーのフォーマットを使って、冗長目次を作成します。アーカイブの前のメンバーへのハード・リンクを表すパス名の場合、その出力は、 <ls -l listing> == <link name> というフォーマットになり、シンボリック・リンクを表すパス名の場合、その出力は、<ls -l listing> = ><link name> というフォーマットになります。ただし <ls -l listing> は、 **-l** オプションと一緒に使われた場合に **ls** ユーティリティーで指定される出力フォーマットです。それ以外の場合、他のどの操作モード (読み取り、書き込み、およびコピー) でもパス名は、該当するファイルまたはアーカイブ・メンバーに対する処理が開始すると同時に、トレーラー改行なしで書き込まれるかまたは標準エラーにフラッシュされます。トレーラー改行は、バッファーに入れられません。また、ファイルの読み取りまたは書き込みが終わった後にしか書き込まれません。LC_TIME 環境変数が設定されている場合、出力時刻は、指定されたロケールの LC_TIME カテゴリーで d_t_fmt キーワードによって指定されたフォーマットでフォーマット設定されます。

-x 出力アーカイブのフォーマットを指定します。デフォルト・フォーマットは ustar です。現在 **pax** では次のようなフォーマットがサポートされています。

cpio 1003.2 標準に指定されている拡張 cpio 交換形式。このフォーマットのデフォルトのブロック・サイズは 5120 バイトです。

bcpio cpio の旧 2 進フォーマット。このフォーマットのデフォルトのブロック・サイズは 5120 バイトです。このフォーマットはあまり移植性が高くないので、他のフォーマットを使えるときは使わないでください。

sv4cpio
システム 5 リリース 4 の cpio。このフォーマットのデフォルトのブロック・サイズは 5120 バイトです。

sv4crc ファイル crc チェックサムを備えたシステム 5 リリース 4 の cpio。このフォーマットのデフォルトのブロック・サイズは 5120 バイトです。

tar - ファイル・アーカイバー
BSD4.3 にあるとおりの旧 BSD tar フォーマット。このフォーマットのデフォルトのブロック・サイズは 10240 バイトです。このフォーマットで保管されるパス名は、100 文字以下の長さでなければなりません。正規ファイル、ハード・リンク、ソフト・リンク、およびディレクトリーしかアーカイブできません (他のファイル・システム・タイプはサポートされません)。ディレクトリーのストレージを省略するためにアーカイブに書き込むときに、旧 tar フォーマットに対する逆方向の互換性を保つには **-o** オプションを使用することができます。このオプションは `-o -Cm -write_opt=nodir` の形式をとります。

ustar 1003.2 標準に指定されている拡張 tar 交換形式。このフォーマットのデフォルトのブロック・サイズは 10240 バイトです。このフォーマットで保管されるパス名は、250 文字以下の長さでなければなりません。

アーカイブ・フォーマットに関する制限が原因で保管または抽出できないファイルは、 **pax** で検出されてレポートされます。個々のアーカイブ・フォーマットごとに、さらに別の使用上の制限事項があることもあります。アーカイブ・フォーマットの一般的な制限事項には、ファイル・パス名の長さ、ファイル・サイズ、リンク・パス名の長さ、およびファイル・タイプなどがありますが、これらのみに限りません。

-A **pax** を旧 tar として実行します。

-B 1 つのアーカイブ・ボリュームに書き込まれるバイト数を制限します。バイト制限を m、k、または b で終了すれば、 1048576 (1M)、1024 (1K)、または 512 の乗算を指定することができます。バイト制限のペアを x で区切って積を示すことができます。

-C *ccsid*

指定された *ccsid* でアーカイブから抽出したファイルを作成します。 CCSID 819 から指定された *ccsid* への有効な変換がなければなりません。このオプションは、QIBM_CCSID 環境変数の値をオーバーライドします。

-D このオプションは **-u** オプションと同じです。ただし、ファイル変更時刻ではなくファイルの i ノード変更時刻が検査されることを除きます。宛先ディレクトリー内のファイル・コピーよりも新しい i ノード情報 (uid や gid など) をもつファイルを選択するときにファイルの i ノード時刻を使います。

-E 不具合のあるアーカイブの読み取りを試みるときの連続した読み取り障害の数を制限します。正数の制限が使用されると **pax** は、アーカイブ読み取りエラーからリカバリーしてから、アーカイブに保管されている次のファイルから処理の再開を試みます。 0 の制限の場合 **pax** は、アーカイブ・ボリューム上で最初の読み取りエラーが検出されると操作を停止します。 NONE の制限では **pax** は、永続的に読み取りエラーからのリカバリーを試みます。デフォルトの制限は、少数の正数の再試行です。

重要:	このオプションで NONE を使用する場合はかなりの慎重さが必要です。不具合のあるアーカイブごとに **pax** が無限ループに陥る可能性があるからです。

-G ファイルをそのグループ名を基に選択するか、または # (数値の gid) を使って始動するときに選択します。 # を拡張するのに " を使うことができます。複数個の -G オプションを指定すれば、最初の一致個所で検査を停止することができます。

-H 物理ファイル・システムを走査するときに、コマンド行のシンボリック・リンクだけを追います。

-L シンボリック・リンクを追いながら、論理ファイル・システムを走査します。

-P シンボリック・リンクを追わずに、物理ファイル・システムの走査を実行します。これがデフォルト・モードです。

-T *from_date* から *to_date* までの (date を含む) 指定の時間範囲内に入るファイル変更または i ノード変更時刻を基にファイルを選択できるようにします。 *from_date* だけを指定すると、それに等しいかまたはそれより新しい変更または i ノード変更時刻をもつすべてのファイルが選択されます。 *to_date* だけを指定すると、それに等しいかまたはそれより古い変更または i ノード変更時刻をもつすべてのファイルが選択されます。 *from_date* と *to_date* が等しいと、ちょうどその時刻の変更時刻または i ノード変更時刻をもつファイルだけが選択されます。

pax が書き込みまたはコピー・モードにあるときに、オプションのトレーラー・フィールド [c][m] を使うと、どのファイル時刻 (i ノード変更とファイル変更のどちらか一方または両方) を比較に使うかを指定することができます。どちらも指定しないと、デフォルトどおりにファイル変更時刻のみが使用されます。 m は、ファイル変更時刻 (ファイルに最後に書き込まれた時刻) の比較を指定します。 c は、i ノード変更時刻 (たとえば所有者、グループ、モードなどの変更のように、ファイル i ノードが最後に変更された時刻) の比較を指定します。 c と m を両方指定すると、変更時刻と i ノード変更時刻の両方が比較されます。 i ノード変更時刻の比較が便利なのは、最近変更された属性をもつファイルを選択する場合や、最近作成されて、しかもその変更時刻をそれより古い時刻にリセットされた (アーカイブからファイルが抽出され、変更時刻が保存されるときに行われる内容) ファイルを選択する場合です。両方のファイル時刻を使った時刻比較が便利なのは、 **pax** を使って時刻ベースの増分アーカイブを作成する (指定時刻範囲内に変更されたファイルのみがアーカイブされます) 場合です。

時刻範囲は、7 つのフィールドから成りますが、各々のフィールドに 2 つの数字が入っていなければなりません。そのフォーマットは次のとおりです。

[cc[yy[mm[dd[hh]]]]]mm[.ss]

cc は世紀、yy は年の最後の 2 桁の数字、最初の mm は月 (01 - 12)、dd は日 (01 - 31)、hh は 1 日のうちの時間 (00 - 23)、 2 番目の mm は分 (00 - 59)、そして ss は秒数 (00 - 59) です。分のフィールドは必須ですが、他のフィールドはオプションであり、hh、dd、mm、yy、cc の順に追加しなければなりません。

ss フィールドは、他のフィールドにとらわれずに追加することができます。時刻範囲は現在の時刻に対する相対範囲であり、したがって -T 1234/cm の場合、本日の 12:34 PM またはそれ以後の変更時刻または i ノード変更時刻をもつすべてのファイルが選択されます。複数個の -T 時刻範囲を指定すれば、最初の一致個所で検査を停止することができます。

-U ファイルをそのユーザー名を基に選択するか、または数値 uid である # を使って始動するときに選択します。 # を拡張するのに " を使うことができます。複数個の **-U** オプションを指定すれば、最初の一致個所で検査を停止することができます。

-X パス名で指定されたファイル階層をたどるときに、別のデバイス ID をもつ下位ディレクトリーには進みません。

-Y このオプションは **-D** オプションと同じです。ただし、ファイル名の変更がすべて完了した後に作成されたパス名を使って i ノード変更時刻が検査されることを除きます。

-Z このオプションは **-u** オプションと同じです。ただし、ファイル名の変更がすべて完了した後に作成されたパス名を使って変更時刻が検査されることを除きます。

ファイルまたはアーカイブ・メンバーの名前に対して実行されるオプション (**-c**、**-i**、**-n**、**-s**、**-u**、**-v**、**-D**、**-G**、**-T**、**-U**、**-Y**、および **-Z**) は以下のように対話を行います。

- 読み取り操作中のファイルの抽出ではアーカイブ・メンバーは、 **-c**、**-n**、**-u**、**-D**、**-G**、**-T**、**-U** オプションによって変更されたユーザー指定の *pattern* オペランドにのみ基づいて選択されます。次に、すべての **-s** および **-i** オプションが、そのように選択されたファイルの名前を上記の順序で変更します。さらに **-Y** および **-Z** オプションが、最終パス名に基づいて適用されます。最後に **-v** オプションが、変更で生成された名前を書き込みます。
- 書き込み操作中のファイルのアーカイブまたはコピー操作中のファイルのコピーではアーカイブ・メンバーは、 **-n**、**-u**、**-D**、**-G**、**-T**、および **-U** オプションによって変更された (**-D** オプションはコピー操作中のみ適用されます) ユーザー指定パス名にのみ基づいて選択されます。次に、すべての **-s** および **-i** オプションが、そのように選択されたファイルの名前を上記の順序で変更します。さらにコピー操作中に **-Y** および **-Z** オプションが、最終パス名に基づいて適用されます。最後に **-v** オプションが、変更で生成された名前を書き込みます。
- **-u** と **-D** オプションのどちらか一方または両方を **-n** オプションと一緒に指定すると、ファイルは、比較のために照らし合わされるファイルよりも新しくない限り、選択済みと見なされません。

オペランド

directory オペランドは、宛先ディレクトリーのパス名を指定します。 *directory* オペランドが存在しないか、またはユーザーがこれに書き込めない場合、あるいはタイプ・ディレクトリーでない場合、**pax** は非ゼロの終了状況で終了します。

pattern オペランドは、アーカイブ・メンバーの 1 つ以上のパス名を選択するのに使用します。 *pattern* オペランドを指定しないと、アーカイブのすべてのメンバーが選択されます。パターンがディレクトリーに一致すると、そのディレクトリーをルートとするファイル階層全体が選択されます。 *pattern* オペランドによって少なくとも 1 つのアーカイブ・メンバーが選択されないと、 **pax** はその *pattern* オペランドを診断メッセージに入れて標準エラーに書き込んでから、非ゼロの終了状況で終了します。

file オペランドは、コピーまたはアーカイブしようとするファイルのパス名を指定します。 *file* オペランドによって少なくとも 1 つのアーカイブ・メンバーが選択されないと、 **pax** はその *file* オペランドのパス名を診断メッセージに入れて標準エラーに書き込んでから、非ゼロの終了状況で終了します。

環境変数

pax は、次の環境変数の影響を受けます。

LANG LC_ で始まる変数を使って明示的に設定されていないロケール・カテゴリー用のデフォルト値を提供します。

LC_TIME
ファイル時刻の表示で使用される日付および時刻形式を定義します。

QIBM_CCSID
pax は、環境変数の値により指定される CCSID のアーカイブから抽出されたファイルを作成します。

終了状況

- 0 すべてのファイルの処理は正常に完了。
- 1 エラー発生。

例

1. 現在のディレクトリーの内容をアーカイブ・ファイルにコピーする場合:

   ```
   pax -w -f saved.ar
   ```

2. アーカイブ・ファイルの冗長目次を表示する場合:

   ```
   pax -r -v -f saved.ar
   ```

3. 以下のコマンドは、 /home/abbey/olddir に固定されたディレクトリー・ツリー全体を /home/abbey/newdir へコピーします。

   ```
   mkdir /home/abbey/newdir
   cd /home/abbey/olddir
   pax -rw . /home/abbey/newdir
   ```

4. 現在のディレクトリーからディレクトリー宛先にコピーするファイルを対話式で選択する場合:

   ```
   pax -rw -i . destination
   ```

5. ユーザー root およびグループ bin によって所有されているアーカイブ・ファイルからすべてのファイルを抽出し、すべてのファイル許可を保存する場合:

   ```
   pax -r -pe -U root -G bin -f saved.ar
   ```

6. ソース・ディレクトリー /sourcecode にあるものと同じ名前の付いたファイルよりも古い宛先ディレクトリー /backup 内のファイルだけをリストおよび更新する場合:

   ```
   pax -r -w -v -Y -Z /sourcecode /backup
   ```

関連タスク:

96 ページの『compress - データを圧縮する』

133 ページの『tar - ファイル・アーカイバー』

pr - ファイルを出力する

構文

pr **[+***page***]** **[-***column***]** **[-adFmrt]** **[-e** [*char*][*gap*]**]** **[-h** *header***]** **[-i[***char***][***gap***]]** **[-l** *line***]** **[-n[***char***][***width***]]** **[-o** *offset***]** **[-s[***char***]]** **[-w** *width***]** [-] [*file* ...]

説明

pr ユーティリティーは、テキスト・ファイルを出力しページ編集するためのフィルターです。複数の入力ファイルが指定されている場合は、各ファイルが読み取られ、形式化され、標準出力に書き込まれます。デフォルト設定では、入力は 66 行構成のページに分かれます。各ページには、ページ番号、日付、時刻、およびファイルのパス名が入った 5 行のヘッダーと、ブランク行から成る 5 行のトレーラーが備えられています。 LC_TIME 環境変数が設定されている場合、ヘッダーの日付および時刻は、指定されたロケールの LC_TIME カテゴリーで d_t_fmt キーワードによって指定されたフォーマットでフォーマット設定されます。

複数列の出力が指定されている場合は、すべてのテキスト列が同じ幅になります。デフォルト設定では、テキストの列と列の間は、少なくとも 1 つの <space> で区切られます。テキスト列に収まらない入力行は切り捨てられます。単一列の出力の場合は、行は切り捨てられません。

出力がリダイレクトされている場合は出力プロセスの実行中に、端末装置で出力している場合は、正常なファイルの出力がすべて完了したあとで、エラー・メッセージが標準エラーに書き込まれます。

端末装置での出力中に **pr** が割り込みシグナルを受け取ると、**pr** は、それまでに累積しているエラー・メッセージをすべて画面に表示してから、終了します。

オプション

注:

1. 次のオプションの説明では、*column*、*lines*、*offset*、 *page*、および *width* は正の 10 進整数で、 *gap* は負ではない 10 進整数です。
2. **-s** オプションを指定するとき、オプション文字と引数の間を空けてはなりません。
3. **-e**、**-i**、および **-n** オプションに引数を指定する場合、引数は両方ともオプション文字と離さずに指定する必要があります。

+*page*　形式化された入力の、*page* に指定したページ番号から出力が開始されます。

-*column*
　列の幅 (デフォルト値は 1) で出力が生成されます。入力ファイルからテキストを受け取る順に、各列に上から下へと書き込みが行われます。オプション **-e** および **-i** が指定されているものと見なされます。このオプションは **-m** オプションと同時に使用しないでください。 **-t** オプションと共に使用した場合は、出力の表示に最小の行数が使用されます。

-a　**column** オプションの効果が変更され、ページの各列にラウンドロビン方式の順序でテキストが埋め込まれます (たとえば、column が 2 の場合、最初の入力行は列 1 の 1 行目から始まり、2 番目は列 2 の 1 行目、3 番目は列 1 の 2 行目から始まります)。このオプションでは、**column** オプションを併用する必要があります。

-d　ダブルスペースの出力が生成されます。入力内で検出された個々の <newline> 文字のあとに、余分な <newline> 文字が出力されます。

-e [*char*][*gap*]
　個々の入力 <tab> を、数式 n**gap*+1 で指定される次に大きい値の桁位置まで展開します。ここで、*n* は 0 より大きい整数です。*gap* の値が 0 かまたは省略された場合、そのデフォルト値は 8 です。入力中のすべての <tab> 文字は、相当数の <space> に展開されます。数字以外の文字、つまり *char* が指定されている場合は、その文字が入力タブ文字として使用されます。

-F　一連の <newline> 文字を使用するデフォルト動作の代わりに、改ページ用に <form-feed> 文字が使用されます。

-h *header*
ストリング *header* を使用して、ヘッダー行の中のファイル名が置換されます。

-i [*char*][*gap*]
出力時に、複数の隣接する <space> が *gap*+1、 2**gap*+1 などで表される桁位置まで達していると、複数の <space> が常に <tab> に置換されます。 *gap* が 0 または省略されている場合には、8 桁ごとのデフォルトの <tab> 設定が使用されます。数字以外の文字、つまり *char* が指定されている場合は、その文字が出力の <tab> 文字として使用されます。

-l *lines* デフォルト値 66 行が一時変更され、ページ長が *lines* の値にリセットされます。 *lines* が、ヘッダーおよびトレーラーの深さ (行数) の合計より大きくない場合は、**-t** オプションが有効である場合と同様に、**pr** ユーティリティーでヘッダーとトレーラーの両方の出力が抑制されます。

-m 複数のファイルの内容を組み合わせます。 file オペランドで指定した個々のファイルから 1 行ずつ取り出され、固定幅の (つまり幅の桁数が等しい) 複数のテキスト列に、各列に対応するファイルからの行が横並びになるように書き込まれます。テキスト列の数は、正常にオープンできた file オペランドの数によって決まります。マージされるファイルの最大数は、ページ幅および 1 プロセスあたりオープン可能なファイルの最大数によって決まります。オプション **-e** および **i** が指定されているものと見なされます。

-n [*char*][*width*]
width 桁を使用して行番号が付けられます。 *width* が指定されなかった場合のデフォルト値は 5 です。行番号は、各テキスト列または **-m** 出力の各行の最初の *width* 桁を占めます。 *char* (数字以外の任意の文字) を指定した場合は、行番号をそのあとのテキストと区切るために、行番号のあとに *char* が付加されます。 *char* のデフォルト値は <tab> です。 *width* の桁数より長い行番号は切り捨てられます。

-o *offset*
出力の各行の前に、*offset* <spaces> が設定されます。このオプションを指定しなかった場合のデフォルト値は 0 です。 設定されるスペースは、出力行の幅です。

-r ファイルのオープンに失敗しても、診断報告書は書き込まれません。

-s *char*
テキストの列と列の間を、該当数の <space> で区切らずに、1 文字の *char* によって区切ります (*char* を指定しなかった場合は、<tab> 文字が使用されます)。

-t 通常は各ページに出力される 5 行の識別ヘッダーも 5 行のトレーラーも、出力されません。各ファイルの最終行を出力したら、ページの終わりまでスペースを入れずに、処理が終了します。

-w *width*
複数テキスト列の出力の場合に限り、行幅を *width* の桁数に設定します。 このオプションを指定せず、**-s** オプションも指定しなかった場合は、デフォルトの幅は 72 です。 このオプションを指定せず、**-s** オプションを指定した場合は、デフォルトの幅は 512です。

オペランド

各 *file* は、出力されるファイルのパス名です。 *file* オペランドを指定しなかった場合、または *file* オペランドが - である場合は、標準入力が使用されます。

環境変数

pr は、次の環境変数の影響を受けます。

LANG LC_ で始まる変数を使って明示的に設定されていないロケール・カテゴリー用のデフォルト値を提供します。

LC_TIME
ヘッダー行の書き込みに使用される日付および時刻のフォーマットを定義します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイルを 3 ページから出力する場合:

```
pr +3 source.java
```

2. 各 *.java ファイルを出力し、ヘッダー・メッセージを変更する場合:

```
pr -h 'JDK source files and examples' code/*.java
```

関連タスク:
88 ページの『cat - ファイルを連結して出力する』
115 ページの『od - 様々なフォーマットのファイルをダンプする』

pwd - 作業ディレクトリー名を戻す

構文

pwd

説明

pwd を使うと、標準出力に作業ディレクトリーを表示することができます。

オプション

なし。

オペランド

なし。

終了状況

- 0 正常終了。

関連タスク:
90 ページの『cd - 作業ディレクトリーを変更する』
『pwdx - 展開された作業ディレクトリーを出力する』

pwdx - 展開された作業ディレクトリーを出力する

構文

pwdx

説明

pwdx を使用すると、シンボリック・リンクを展開した形で、作業ディレクトリーを標準出力に表示することができます。

終了状況

- 0 正常終了。

関連タスク:

90 ページの『cd - 作業ディレクトリーを変更する』

127 ページの『pwd - 作業ディレクトリー名を戻す』

Rfile - レコード・ファイルの読み書きをする

構文

Rfile -r | -w | -h [-abKlqQs] [-c *CL-command*] **[-C** *CL-command*] *file ...*

説明

Rfile ユーティリティーは、i5/OS レコード・ファイル (データベースまたは装置ファイル) を読み取って、そのデータを標準出力に書き込む処理、または標準入力を読み取って、そのデータをレコード・ファイルに書き込む処理を行います。

注:	これは i5/OS に固有のユーティリティーです。

オプション

-a 標準入力の内容をレコード・ファイルに追加します。このオプションは、**-w** が指定されている場合にのみ適用されます。 **-w** が指定されていて、**-a** の指定がない場合、ストリームの内容を書き込む前に、物理ファイル・メンバーはいずれも消去されます。

-b バイナリー・データを処理します。このオプションを指定すると、入力ストリームまたは出力ストリーム内の改行文字の通常の処理が停止します。 **-b** を省略すると、レコード・ファイルに書き込まれる標準入力行から改行文字が除去されたり、標準出力に書き込まれるレコードの末尾に改行文字が挿入されたりします。

-c *CL-command*
何らかのレコード・ファイルを処理する前に、ユーティリティー・プロセスの CL コマンドを実行します。このオプションは、レコード・ファイル用の装置依存のパラメーターを指定する CL オーバーライド・コマンドを実行するために使用されます。複数の **-c** オプションを指定すると、レコード・ファイルを処理する前に、CL コマンドが順番に処理されます。

-C *CL-command*
すべてのレコード・ファイルを処理した後に、ユーティリティー・プロセスの CL コマンドを実行します。複数の **-C** オプションを指定すると、すべてのレコード・ファイルを処理した後に CL コマンドが順番に処理されます。

-h 標準エラーにコマンド構文の要旨を書き込みます。

-K ジョブ終了時にジョブ・ログを保存します。通常、システムは QShell ユーティリティーを実行した後、そのジョブ・ログを削除します。このオプションを設定すると、**Rfile** を実行するジョブが終了したとき、強制的にシステムはジョブ・ログ・リストを作成します。(これは問題判別の助けになるかもしれません。)

-l 長いテキスト行を切り捨てません。このオプションはテキスト・データにのみ適用されます。 **-l** を指定すると、1 つの出力レコードに収まらない長さの標準入力行は、折り返されて必要な数のレコードに分割されます。標準出力に書き込まれるレコードから、末尾ブランクは除去されません。

-q 警告メッセージを抑止します。このオプションを使用すると、出力ファイル内で長いテキスト行が切り捨てられたり、折り返されたりした場合に、通常標準エラーに書き込まれるメッセージが抑止されます。

-Q ファイル名として i5/OS 修飾名構文を使用します。このオプションを指定すると、コマンド・オペランドとして指定されるファイル名は、(統合ファイル・システムのパス名ではなく) i5/OS 修飾名になります。

-r 指定されたレコード・ファイルを読み取り、それらの内容を標準出力に書き込みます。 **-r** または **-w** のいずれか一方だけを指定しなければなりません。

-s ソース・シーケンス番号および日付フィールドをテキストとして処理します。このオプションは、FILETYPE(*SRC) レコード・ファイルのテキスト処理にのみ適用されます。 **-s** を指定すると、すべてのレコードの内容全体は、テキスト行として処理されます。 **-s** を省略すると、読み取られるすべてのソース・レコードから最初の 12 バイトが取り除かれ、書き込まれるすべてのソース・レコードの最初の 12 バイトが、シーケンス番号とゼロ (日付フィールドに相当する) で埋められます。

-w 標準入力から読み取り、その内容を、指定したレコード・ファイルに書き込みます。出力ファイルはすでに存在しているものでなければなりません。さもないとエラーが報告されます (ファイルは作成されません)。 **-r** または **-w** のいずれか一方だけを指定しなければなりません。

オペランド

少なくとも 1 つの i5/OS レコード・ファイルを指定する必要があります。複数のファイルを指定すると、ファイルが順番に処理されます。各入力ソースはファイル終わりまで処理されます。オプション **-Q** を省略すると、ファイルは統合ファイル・システムのパス名により識別されます。オプション **-Q** を指定すると、ファイル名は以下の形式のいずれかにより指定されます。

```
file
library/file
'file(member)'
'library/file(member)'
```

ライブラリー名を省略した場合、またはライブラリー名として *LIBL を指定した場合、そのファイルはジョブ・ライブラリー・リストを使用して探し出されます。メンバー名を省略した場合、またはメンバー名として *FIRST を指定した場合、データベース・ファイルの最初のメンバーがオープンします。メンバー名として *LAST を指定すると、データベース・ファイルの最後のメンバーがオープンします。メンバー名 *ALL をオプション **-r** と共に使用すると、データベース・ファイルのすべてのメンバー (最初から最後まで) を読み取ることができます。装置ファイルのメンバー名は無視されます (i5/OS 修飾名形式で指定した場合)。

例

1. ソース・データベース・メンバー QSYSINC/H(SQLCLI) の内容を読み取り、それを標準出力に書き込みます。 (シーケンス番号および日付情報を含む) 各行の最初の 12 文字と、末尾ブランクは除去されます。

   ```
   Rfile -rQ 'qsysinc/h(sqlcli)'
   ```

2. ストリーム・ファイル mydoc.ps の内容を、未変換の ASCII データとしてスプール・プリンター・ファイル QPRINT に書き込み、その後 CL LPR コマンドを使用してスプール・ファイルを別のシステムに送信します。

   ```
   before='ovrprtf qprint devtype(*userascii) spool(*yes)'
   after="lpr file(qprint) system(usrchprt01) prtq('rchdps') transform(*no)"
   cat -c mydoc.ps | Rfile -wbQ -c "$before" -C "$after" qprint
   ```

3. ライブラリー QGPL にある保管ファイル INSAVF の内容を、ジョブ・ライブラリー・リストを使用して見つけた OUTSAVF という名前の保管ファイルにコピーします。 ASCII/EBCDIC 変換および改行処理を避けるため、データはバイナリー・モードで読み書きされるという点に注意してください。

```
Rfile -rb /qsys.lib/qgpl.lib/insavf.file | Rfile -wbQ outsavf
```

関連タスク:

89 ページの『catsplf - スプール・ファイルを連結して出力する』

rm - ディレクトリー項目を削除する

構文

rm [-f | -i] [-dPRr] *file* ...

説明

rm ユーティリティーは、コマンド行で指定したディレクトリー以外のタイプの *file* を削除します。 *file* の許可で書き込みが許可されていない場合は、標準入力装置が端末であれば、標準エラー時の確認を求めるプロンプトが表示されます。

rm ユーティリティーは、リンクによって参照されるファイルではなく、シンボリック・リンクを除去します。

ファイル . および .. を削除しようとすると、エラーになります。

オプション

-d 他のタイプのファイルと同様に、ディレクトリーも削除の対象になります。

-f ファイルの許可に関係なく、確認のためのプロンプトなしに *file* が削除されます。該当ファイルが存在しなくても、診断メッセージは表示されず、終了状況がエラーを反映する値に変更されることもありません。それより前の **-i** オプションはすべて、 **-f** オプションによって指定変更されます。

-i ファイルの許可、または標準入力装置が端末かどうかに関係なく、各 *file* を削除する前に確認のプロンプトが表示されます。 標準入力からの応答が現行ロケールにおける YES 応答の先頭文字で始まっている場合は、*file* が削除されます。それより前の **-f** オプションはすべて、 **-i** オプションによって指定変更されます。

-P 削除の前に正規ファイルが上書きされます。これらのファイルは、削除の前に 3 回上書きされます。最初はバイト・パターン 0xff、次に 0x00、最後に再び 0xff で上書きされます。

-R 各 *file* 引数をルートとするファイル階層が削除されます。 **-R** オプションでは、**-d** オプションが暗黙に指定されています。 **-i** オプションを指定すると、各ディレクトリーの内容を処理する前に (そしてディレクトリーを削除しようとする前と同様)、確認を求めるプロンプトが表示されます。ユーザーが肯定の応答をしなかった場合は、そのディレクトリーをルートとするファイル階層は処理対象から外されます。

-r **-R** と同じです。

終了状況

- 0 指定したすべてのファイルまたはファイル階層が削除されたか、**-f** オプションが指定されていたため、既存のすべてのファイルまたはファイル階層が削除されました。
- >0 エラー発生。

例

1. 確認を求めるプロンプトを表示しないで、すべてのファイルおよびディレクトリー java、そしてすべてのサブディレクトリーまたはファイル (あるいはその両方) を削除する場合 :

   ```
   rm -r -f /home/bob/examples/code/java
   ```

2. ファイル　file1、file2、および　file3 を除去する場合:

   ```
   rm file1 file2 file3
   ```

関連タスク:

98 ページの『cp - ファイルをコピーする』
108 ページの『ln - ファイルをリンクする』
109 ページの『ls - ディレクトリーの内容をリスト表示する』
114 ページの『mv - ファイルを移動する』
『rmdir - ディレクトリーを削除する』

rmdir - ディレクトリーを削除する

構文

rmdir *directory ...*

説明

rmdir ユーティリティーは、各 *directory* 引数で指定されているディレクトリー項目が空の場合に、その項目を削除します。

引数は指定した順序で処理されます。親ディレクトリーとそのサブディレクトリーを削除するには、最初にサブディレクトリーを指定して、**rmdir** による削除の際に親ディレクトリーが空になっているようにする必要があります。

終了状況

- 0 指定した各ディレクトリー項目によって空のディレクトリーが参照されていて、そのディレクトリーの削除が正常に完了。
- >0 エラー発生。

関連タスク:

98 ページの『cp - ファイルをコピーする』
108 ページの『ln - ファイルをリンクする』
109 ページの『ls - ディレクトリーの内容をリスト表示する』
112 ページの『mkdir - ディレクトリーを作成する』
130 ページの『rm - ディレクトリー項目を削除する』

setccsid - ファイルの CCSID 属性を設定する

構文

setccsid [-R [-H | -L | -P]] [-h] *ccsid file ...*

説明

setccsid ユーティリティーは、指定された *files* の CCSID 属性を、指定された *ccsid* に設定します。 *file* に入っているデータは変更されません。

オプション

-H **-R** オプションを指定すると、コマンド行のシンボリック・リンクがたどられます。ツリー走査中に検出されたシンボリック・リンクは対象にはなりません。

-L **-R** オプションが指定されている場合は、コマンド行のシンボリック・リンクとツリー走査中に発見されたシンボリック・リンクの両方に従って操作が実行されます。

-P **-R** オプションが指定されていれば、シンボリック・リンクはたどられません。

-R *file* にディレクトリーが指定されている場合、その時点で連結している全サブツリー中の各ファイルの CCSID が **setccsid** によって設定されます。

-h シンボリック・リンクが指すファイルではなく、シンボリック・リンクの CCSID を設定します。

オペランド

ccsid は、コード化文字セット ID を識別する整数です。各 *file* は、CCSID を設定されるファイルのパス名です。

例

"file1" および "file2" のファイルの CCSID を 819 に設定する場合:

```
setccsid 819 file1 file2
```

関連タスク:

67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
67 ページの『sed - ストリーム・エディター』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
75 ページの『split - ファイルを分割する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』
82 ページの『attr - ファイルの属性を取得または設定する』
92 ページの『chmod - ファイル・モードを変更する』
95 ページの『chown - ファイルの所有権を変更する』
135 ページの『touch - ファイルのアクセス時刻および変更時刻を変更する』

tail - ファイルの末尾部分を表示する

構文

tail [-f | -r] [-b *number* **| -c** *number* **| -k** *number* **|-n** *number***]** [*file* ...]

説明

tail ユーティリティーは、*file* (デフォルト設定では標準入力) の内容を標準出力に表示します。

表示は、入力の中の特定のバイト、行、512 バイト、または K バイト・ブロックの位置から始まります。*number* の前に正符号 (+) を付けると、入力の始めを基準とする相対位置を指定できます。たとえば、-c +2 と入力すると、入力の 2 バイト目から表示されます。 *number* の前に負符号 (-) を付けるか、または符号を明記しなかった場合は、入力の終わりを基準とする相対位置となります。たとえば、-n 2 と指定すると、入力の最後の 2 行が表示されます。デフォルトの開始位置は、-n 10、つまり入力の最後の 10 行です。

複数の *file* を指定した場合は、各 file の前に、==> XXX <== の形式のストリングから成るヘッダーが付けられます。XXX はファイルの名前です。

注:	**tail** では、大きいファイル (サイズが 2GB を超えるファイル) は処置できません。

オプション

-b *number*
位置は *number* 番目の 512 バイト・ブロックです。

-c *number*
位置は *number* 番目のバイトです。

-f ファイルの終わりに達しても **tail** は終了しないで、入力に後続のデータが追加されるのを待ちます。 **-f** オプションは、標準入力がパイプの場合は無視されますが、FIFO の場合は無視されません。

-k *number*
位置は *number* 番目の K バイトです。

-n *number*
位置は *number* 番目の行です。

-r 入力は行単位で、逆の順序で表示されます。さらに、このオプションを指定すると、**-b**、 **-c**、**-n** オプションの意味も変化します。 **-r** オプションを指定した場合、これらのオプションは、表示を開始する位置を入力の始めまたは終わりからのバイト数、行数、またはブロック数で指定するのではなく、表示するバイト数、行数、または 512 バイト・ブロック数を指定するものとなります。 **-r** オプションのデフォルト設定では、入力のすべてが表示されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

ファイル donkeys の最後の 100 行を表示する場合: ただし、ファイル donkeys が 100 行未満の場合、**tail** はファイル全体を表示します。

```
tail -n 100 donkeys
```

関連タスク:

88 ページの『cat - ファイルを連結して出力する』
107 ページの『head - ファイルの先頭部分をコピーする』

tar - ファイル・アーカイバー

構文

tar -crtux[befmopvwHLPX] [*archive***] [***blocksize***]** *file ...*

説明

tar ユーティリティーは、アーカイブ・ファイルのファイルの読み取り、書き込み、およびリスト表示を行います。

オプション

以下のオプションで、**tar** が実行する機能を選択します。以下のオプションの 1 つを指定する必要があります。

-c 新しいアーカイブを作成します。

-r 指定された *file* をアーカイブの末尾に追加します。

-t アーカイブにあるファイル名を標準出力にリストします。

-u アーカイブの中の *file* が前回アーカイブに書き込まれてから変更されている場合は、それを更新します。*file* がアーカイブの中にない場合は、それをアーカイブに追加します。

-x 指定された *file* をアーカイブから抽出します。 *file* を指定しないと、アーカイブからすべてのファイルが抽出されます。

以下のオプションは、**tar** の操作に影響します。

-b 第 1 オペランド (**f** がすでに指定されている場合は第 2 オペランド) を、アーカイブのブロック・サイズとして使用します。

-e 最初のエラーが検出された後、終了します。

-f 第 1 オペランド (**b** がすでに指定されている場合は第 2 オペランド) を、アーカイブの名前としてデフォルトの名前に替えて使用します。ファイル名が - の場合、**tar** は機能に応じて、標準出力への書き込みまたは標準入力からの読み取りを行います。

-m 変更時刻を復元しません。ファイルの変更時刻は、抽出された時刻になります。

-o 抽出されたファイルの所有者およびグループの設定を、アーカイブと一緒に保管されたユーザーおよびグループに代えて、**tar** を実行しているユーザーにします。

-p アーカイブから抽出されたファイルの所有者、グループ、ファイル・モード、アクセス時間、および変更時刻を保存します。

-v 冗長モード。処理されているそれぞれのファイルの名前を標準エラーに書き込みます。 **t** 機能が指定されている場合、その出力には、それぞれのファイルのモード、リンクの数、所有者、グループ、サイズ、および変更日付が含まれます。

-w 実行するアクションとファイル名を書き出した後、ユーザーの確認を待ちます。肯定の応答が返された場合、その処理が実行されます。それ以外の入力があった場合、その処理は行われません。

-H 物理ファイル・システムを走査するときに、コマンド行のシンボリック・リンクだけを追います。

-L シンボリック・リンクを追いながら、論理ファイル・システムを走査します。

-P シンボリック・リンクを追わずに、物理ファイル・システムの走査を実行します。これがデフォルト・モードです。

-X パス名で指定されたファイル階層をたどるときに、別のデバイス ID をもつ下位ディレクトリーには進みません。

オペランド

それぞれの *file* は、その機能に応じ、アーカイブに追加されるオブジェクト、またはアーカイブから抽出されるオブジェクトになります。

環境変数

tar は、次の環境変数の影響を受けます。

QIBM_CCSID
環境変数の値は、アーカイブから抽出されるファイルを作成する時に使用された CCSID です。CCSID 819 から、指定された CCSID への有効な変換がなければなりません。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

117 ページの『pax - ポータブル・アーカイブを交換する』

touch - ファイルのアクセス時刻および変更時刻を変更する

構文

touch [-acfm] [-r *ref_file*] **[-t** [[CC]YY]MMDDhhmm[.SS] **] [-C** *ccsid*] *file* ...

説明

touch ユーティリティーは、ファイルの変更時刻およびアクセス時刻を現在の時刻に設定します。 *file* が存在しない場合は、デフォルトの許可を持つものとしてそのファイルが作成されます。

オプション

-a *file* のアクセス時刻を変更します。 **-m** フラグが指定されていない場合は、ファイルの変更時刻は変更されません。

-C *ccsid*
file が存在しない場合は、指定した *ccsid* を持つファイルが作成されます。このオプションは、QIBM_CCSID 環境変数の値をオーバーライドします。

-c *file* が存在しない場合には、それは作成されません。 **touch** ユーティリティーでは、この状況はエラーとは見なされません。エラー・メッセージは表示されず、終了値には影響はありません。

-f 現時点でファイル許可では許されていない場合でも、更新の強制実行を試みます。

-m *file* の変更時刻を変更します。 **-a** フラグが指定されていない場合は、ファイルのアクセス時刻は変更されません。

-r *ref_file*
現在の時刻ではなく、指定した *ref_file* のアクセス時刻および変更時刻を使用します。

-t アクセス時刻および変更時刻が、指定した時刻に変更されます。この引数の形式は、次の形式で指定してください。

```
[[CC]YY]MMDDhhmm[.SS]
```

対になっているそれぞれの文字の意味は次のとおりです。

CC 年の最初の 2 桁 (世紀)。

YY 年の後半の 2 桁。 CC を指定せずに YY だけを指定した場合、YY の値が 69 - 99 の範囲内であれば、CC 値は 19 と見なされます。その他の場合は、20 の CC 値が使われます。

MM 月を表す 1 - 12 の値。

DD 日を表す 1 - 31 の値。

hh 時を表す 0 - 23 の値。

mm 分を表す 0 - 59 の値。

SS 秒を表す 0 - 59 の値。

CC と YY のどちらも指定しなかった場合は、デフォルト値として現在の年が使用されます。 SS を指定しなかった場合、デフォルト値は 0 です。

環境変数

touch は、次の環境変数の影響を受けます。

QIBM_CCSID
file が存在しない場合、**touch** は、環境変数の値で指定される CCSID のファイルを作成します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイル myfile の時刻・日付スタンプを、ファイル yourfile の時刻・日付スタンプに変更する場合 :
   ```
   touch -r yourfile myfile
   ```
2. ファイル myfile の時刻・日付スタンプを特定の時刻・日付スタンプに変更する場合 :
   ```
   touch -t 200001010000.00 myfile
   ```

関連タスク:

82 ページの『attr - ファイルの属性を取得または設定する』

131 ページの『setccsid - ファイルの CCSID 属性を設定する』

『umask - ファイル・モード作成マスクを入手または設定する』

umask - ファイル・モード作成マスクを入手または設定する

構文

umask [**-S**] [*mask*]

説明

umask を使用すると、ファイル作成マスクを設定または表示できます。マスクは、ファイルまたはディレクトリーの作成時に設定されたファイル許可ビットを制御するために使用します。

mask が指定されると、**qsh** はファイル作成マスクを *mask* に設定します。 *mask* が指定されなかった場合は、**qsh** は、現行のファイル作成マスクを標準出力に表示します。

オプション

- **-S** シンボリック許可を使用します。

オペランド

シンボリック許可を使用しているときは、*mask* は、削除してはならない許可を定義する式です。シンボリック許可は、[*who*] *op* [*permission*] の形式の式で、意味は次のとおりです。

- *who* は次の英字の組み合わせです。

  - **u** (所有者許可)
  - **g** (グループ許可)
  - **o** (その他 (共通)) の許可
  - **a** (すべての許可 (デフォルト値))
- *op* は次のどちらかです。
  - **-** (負符号) (許可を削除する場合)
  - **+** (正符号) (許可を追加する場合)
- *permission* は次のとおりです (複数指定も可)。
  - **r** (読み取り許可)
  - **w** (書き込み許可)
  - **x** (実行または検索許可)

終了状況

- 0 正常終了。
- >0 *mask* が無効。

例

1. 現行のファイル作成マスクをシンボリック形式で表示する場合: **umask -S**
2. 現行のファイル作成マスクを表示する場合: **umask**
3. ファイル作成マスクを設定、他のユーザーの読み取り許可を削除する場合: **umask** 004
4. ファイル作成マスクを設定してグループの書き込み許可を削除する場合: **umask -S** g-w

関連タスク:

92 ページの『chmod - ファイル・モードを変更する』
98 ページの『cp - ファイルをコピーする』
112 ページの『mkdir - ディレクトリーを作成する』
113 ページの『mkfifo - FIFO 特殊ファイルを作成する』
135 ページの『touch - ファイルのアクセス時刻および変更時刻を変更する』
195 ページの『ulimit - リソース限界を設定または表示する』

uncompress - 圧縮データを圧縮解除する

構文

uncompress [-cv] [-b *bits*] [*file* ...]

説明

uncompress ユーティリティーは、圧縮された *files* をオリジナルの形式に復元し、そのときに拡張子 .Z を削除してファイル名を変更します。

file の名前を変更するとファイルが上書きされてしまう場合に、標準入力装置が端末であると、標準エラーを確認するよう指示するプロンプトが表示されます。プロンプトを出せない場合や確認が得られない場合、ファイルは上書きされません。

オプション

-b *bits* ビット・コード限度を指定します (詳細は以下を参照してください)。

-c 未圧縮の出力が標準出力に書き込まれます。ファイルは変更されません。

-v 各ファイルの縮小パーセントを出力します。

オペランド

各 *file* は、圧縮解除されるファイルのパス名です。 *files* を指定しない場合、標準出力への標準入力は未圧縮になります。入力ファイルも出力ファイルも正規のファイルでない場合、サイズの縮小とファイルの上書きの検査は実行されず、入力ファイルは除去されず、入力ファイルの属性は保存されません。

補足説明

uncompress ユーティリティーでは、修正された Lempel-Ziv アルゴリズムが使われます。ファイル内の共通サブストリングは、最初に 9 ビット・コード 257 以上に置き換えられます。このアルゴリズムは、コード 512 に達すると 10 ビット・コードに切り替え、 **-b** フラグで指定された限度 (デフォルトは 16) に達するまでさらにビットを使用しつづけます。ビットは 9 - 16 でなければなりません。

実現できる圧縮量は、入力のサイズ、コードあたりのビット数、および共通サブストリングの配布によって異なります。通常、ソース・コードや英語などのテキストは 50 - 60% 縮小されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

96 ページの『compress - データを圧縮する』

『zcat - データを拡張および連結する』

zcat - データを拡張および連結する

構文

zcat [*file* ...]

説明

zcat ユーティリティーは、指定された *files* にある圧縮データを展開します。するとその圧縮解除出力は標準出力に書き込まれます。

オペランド

各 *file* は、圧縮データの入ったファイルのパス名です。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

88 ページの『cat - ファイルを連結して出力する』

89 ページの『catsplf - スプール・ファイルを連結して出力する』

96 ページの『compress - データを圧縮する』

137 ページの『uncompress - 圧縮データを圧縮解除する』

入出力の読み書き用のユーティリティー

入出力の読み書き用のユーティリティーを表示します。

dspmsg - メッセージ・カタログのメッセージを表示する

構文

dspmsg [-n] [-s *set***]** *catalog msgid* **[** *defaultMsg* **[** *arguments ...* **]]**

説明

dspmsg ユーティリティーは、GENCAT CL コマンドで作成されたメッセージ・カタログのメッセージを表示します。メッセージは標準出力に書き込まれます。複数の言語に翻訳されるメッセージをスクリプトが表示する必要があるときは、 **echo** または **print** の代わりに **dspmsg** ユーティリティーを使うことができます。

オプション

-n 指定されたメッセージを置換なしに表示します。

-s *set* メッセージ・カタログ内に指定された *set* でメッセージを検索します。 *set* のデフォルト値は 1 です。

オペランド

catalog オペランドは、メッセージ・カタログへのパス名を指定します。相対パス名を使ってカタログを指定すると、カタログの検索には NLSPATH 変数と LC_MESSAGES ロケール・カテゴリーが使われます。

msgid オペランドは、メッセージ・カタログで検索するメッセージ ID を指定します。

指定した *catalog* または *msgid* が見つからないと、代わりにオプションの *defaultMsg* が表示されます。 *defaultMsg* オペランドを指定しないと、システム生成メッセージが表示されます。

オプションの *arguments* に %s、%n$s、%ld、または %n$ld printf() 変換指定が入っていると、その arguments は出力メッセージに置き換えられます。その他の変換指定は無効です。ただし通常の制御文字エスケープ (たとえば、¥n) はサポートされます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

カタログ mycat のメッセージ 5 を表示する場合:

```
dspmsg mycat 5 "Message not found" hello
```

関連タスク:

105 ページの『gencat - フォーマットされたメッセージ・カタログを生成する』
140 ページの『echo - 引数を標準出力に書き込む』
140 ページの『print - 出力を書き出す』
141 ページの『printf - 形式化された出力を書き出す』
142 ページの『read - 標準入力から行を読み取る』

echo - 引数を標準出力に書き込む

構文

echo [*arg* ...]

説明

echo を使うと、各 *arg* を空白文字で区切り、後ろに改行文字を付けて、標準出力に表示することができます。

オペランド

各 *arg* は標準出力にエコー表示されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

60 ページの『xargs - 引数リストを作成してユーティリティーを起動する』
139 ページの『dspmsg - メッセージ・カタログのメッセージを表示する』
『print - 出力を書き出す』
141 ページの『printf - 形式化された出力を書き出す』
194 ページの『tee - 標準入力を複製する』

print - 出力を書き出す

構文

print [**-nrR**] [**-u** [*n*]] [*argument* ...]

説明

print を使うと、各 *argument* を <space> 文字で区切り、末尾に <newline> 文字を付けて、標準出力に表示することができます。

-r または **-R** を指定した場合を除き、print は次の規則に従って出力の形式を設定します。

- **¥a** ベル。
- **¥b** バックスペース。
- **¥c** 改行文字を追加しないで印刷。残りの *arguments* は無視されます。
- **¥f** 書式送り。
- **¥n** 改行。
- **¥r** 復帰。
- **¥t** タブ。
- **¥v** 垂直タブ。
- **¥¥** 円記号。
- **¥0x** 1 桁、2 桁、または 3 桁の 8 進数の EBCDIC コードを持つ文字。

オプション

-n 出力の末尾に改行文字を追加しません。

-r 上記の規則を使用しません。

-R 上記の規則を使用しません。

-u *n* 出力を、指定された記述子 *n* (指定されていない場合のデフォルト値は記述子 1) に書き出します。 記述子は、1、2、または **exec** を使ってオープンした記述子でなければなりません。

オペランド

各 *argument* は標準出力で印刷されます。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

139 ページの『dspmsg - メッセージ・カタログのメッセージを表示する』

140 ページの『echo - 引数を標準出力に書き込む』

『printf - 形式化された出力を書き出す』

142 ページの『read - 標準入力から行を読み取る』

printf - 形式化された出力を書き出す

構文

printf *format*[*argument* ...]

説明

printf を使うと、出力を形式化して標準出力に表示することができます。構文は ILE C printf() 関数に似ています。 **printf** は、次の変換制御文字列構文を使って出力を形式化します。

%[*flags*][*width*].[*precision*]*conversion*

conversion は、該当の *argument* をどのように表示するかを指定します。 次の変換文字のいずれかを指定する必要があります。

c 符号なし文字

d 符号付き 10 進数

e,E 浮動小数

f 浮動小数点数

g,G 有効数字を持つ浮動小数

i 符号付き 10 進数

o 符号なし 8 進数。

s ストリング。

u 符号なし 10 進数。

x 0123456789abcdef を数字として使用する符号なし 16 進数。

X 0123456789ABCDEF を数字として使用する符号なし 16 進数。

flags は、*argument* の表示方法を次のように制御します。

- (負符号)
argument をフィールド内で左寄せします。

+ (正符号)
すべての数字の前に + または - を付けます。

space 正数の前に <space> を付け、負数の前に - を付けます。

0 **d**、**e**、**E**、**f**、**g**、または **G** の場合に、フィールド幅を先行ゼロで埋めます。

変換文字に応じた別の出力形式を使用します。 **o** の場合、8 進数の前に 0 を付けます。 **x** の場合、16 進数の前に 0x を付けます。 **X** の場合、16 進数の前に 0X を付けます。 **e**、**E**、**f**、**g**、または **G** の場合、小数点を表示します。 **g** または **G** の場合、後続ゼロを表示します。

width は、表示される最小桁数です。 width にアスタリスク (*) 文字を使用した場合は、次の *argument* の値がフィールド幅となります。

precision の意味は変換文字によって異なります。

- **d**、**i**、**o**、**u**、**x**、または **X** *precision* は、表示する最小桁数を指定します。
- **e**、**E**、または **f** の場合 *precision* は、小数点の後に表示する桁数を指定します。
- **g**、または **G** *precision* は、有効数字の最大桁数を指定します。
- **s** の場合 *precision* は、表示する最大文字数を指定します。

オプション

なし。

オペランド

各 *argument* は、*format* に指定したとおりに変換され表示されます。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

139 ページの『dspmsg - メッセージ・カタログのメッセージを表示する』
140 ページの『echo - 引数を標準出力に書き込む』
140 ページの『print - 出力を書き出す』

read - 標準入力から行を読み取る

構文

read [**-r**] [**-p** *prompt*] [**-u** [*n*]] [*name* ...]

説明

read を使うと、行を 1 つ読み取り、 **IFS** 変数内の文字を区切り文字として使って、その行をいくつかのフィールドに分割することができます。デフォルトでは、行の終わりに円記号 **(¥)** があると、その行は次の行に継続します。 **qsh** は、円記号と <newline> の両方を削除します。

オプション

-p *prompt*
対話式オプションが設定されているときに、標準エラーに *prompt* を表示します。

-r 行末の円記号は行の継続を意味しません。

-u *n* 指定された記述子 *n* (指定されていない場合のデフォルト値は記述子 0) から読み取ります。記述子は、0 、または **exec** を使ってオープンした記述子でなければなりません。

オペランド

各 *name* は、入力行の対応するフィールドに割り当てられます。 残ったフィールドは、すべて最後の *name* に割り当てられます。 デフォルトの name は **REPLY** 変数です。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. プロンプトを表示したあとで stdin から行を読み取る場合: **read** -p `Enter a name: ' firstname lastname
2. 記述子 5 から行を読み取る場合: **read** -u5

関連概念:

50 ページの『exec - コマンドを実行し、記述子をオープン、クローズ、またはコピーする』

関連タスク:

139 ページの『dspmsg - メッセージ・カタログのメッセージを表示する』
140 ページの『print - 出力を書き出す』

Java プログラム開発用のユーティリティー

Java プログラム開発用のユーティリティーを表示します。

ajar - 代替 Java アーカイブ

構文

ajar {-h | --help}

ajar {-l | --list} [-v | --verbose] [-q | --quiet] *jarfile* **[**{*file* | *pattern*} ...**] [{-x | -i}** {*file* | *pattern*} ...**] ...**

ajar {-x | --extract} [-v | --verbose] [-q | --quiet] [-N | --neverWrite] [-p | --pipe] *jarfile* **[**{*file* | *pattern*} ... **] [{-x | -i}** {*file* | *pattern*} ...**] ...**

ajar {-c | --create} [-0 | --store-only] [-v | --verbose] [-r | --recurse] [-@ | --stdin] [-D | --nodirs] [-q | --quiet] [{-m | --manifest} *mffile*] **[-M | --no-manifest] [{-n | --no-deflate}** *suffix..*] *jarfile file ...* **[{-x | -i}** {*file* | *pattern*} ...**] ...**

ajar {-a | --add} [-0 | --store-only] [-v | --verbose] [-r | --recurse] [-@ | --stdin] [-D | --nodirs] [-q | --quiet] [{-m | --manifest} *mffile*] **[-M | --no-manifest] [{-n | --no-deflate}** *suffix..*] *jarfile file ...* **[{-x | -i}** {*file* | *pattern*} ...] **...**

ajar {-d | --delete} [-v | --verbose] [-q | --quiet] [{-m | --manifest} *mffile*] **[-M | --no-manifest]** *jarfile* {*file* | *pattern*} ... **[{-x | -i}** {*file* | *pattern*} ...] **...**

説明

ajar は、Java[tm] Archive (JAR) ファイルを作成および操作するための別のインターフェースとして使用できます。 **ajar** ユーティリティーは、zip/unzip ツールに搭載されているいくつかの機能と、 IBM Developer Kit for Java という **jar** ツールの機能とを組み合わせたものです。 zip または unzip に似たインターフェースが必要な場合は、jar でなく、**ajar** を使用してください。

jar ツールと同様に、**ajar** は、jar ファイルのコンテンツのリスト作成、jar ファイルからの抽出、新しい jar ファイルの作成に使用でき、多数の zip 形式をサポートしています。さらに、**ajar** は、既存の jar 形式のファイルの追加と削除をサポートしています。

アクション

-h | --help
コマンド構文を stdout に書き込みます。

-l | --list
目次を stdout に書き込みます。

-x | --extract
ファイルを抽出して現行ディレクトリーに書き込みます。

-c | --create
新しいアーカイブを作成します。

-a | --add
アーカイブに新しいファイルを追加し、既存のファイルを置換します。

-d | --delete
アーカイブからファイルを削除します。

オプション

-@ | --stdin
stdin からファイル・リストを読み取ります。このファイル・リストは、コマンド行で通常 *jarfile* パラメーターの次に続くパラメーターで構成されます。ファイル・リストは複数行で構成することができます。この場合、項目を 1 行ずつ指定し、先行ブランクは入れません。注記は '#' で始まり、1 行に入力します。

-0 | --store-only
保管のみ。ファイルの圧縮はしません。ファイルの追加および jar の作成時に使用されます。

-m | --manifest
指定されたファイルから MANIFEST 情報のみを組み込みます。

-n | --no-deflate
指定されたサフィックスが付いたファイルを圧縮しません。サフィックスのリストは、別のオプションまたは "--" を使って終了させなければなりません。次の例を参照してください。

-p | --pipe
抽出先を stdout にします。

-q | --quiet
静止モード。情報メッセージも警告メッセージも書き込みません。

-r | --recurse
ディレクトリー内に再帰します。ファイルの追加および jar の作成時に使用されます。

-v | --verbose
冗長モード。診断情報を stderr に書き込みます。

-D | --nodirs
ディレクトリー項目を抑止します。ファイルの追加および jar の作成時に使用されます。

-M | --no-manifest
MANIFEST を作成しません。

-N | --neverWrite
抽出時に、ファイルの上書きが行われません。

オペランド

jarfile オペランドで、処理対象の jar ファイルのパス名が指定されます。 *jarfile* は、統合ファイル・システム (IFS) 名でなければなりません。

file オペランドで、ファイルまたはディレクトリーのパス名が指定されます。 *file* は、IFS 名でなければなりません。

pattern オペランドで、ファイルおよびディレクトリーのパス名と一致するパターンが指定されます。 *pattern* は、IFS 名に一致するものです。 *pattern* は、次のメタ文字 (1 つまたは複数) を備えた文字列です。

***** 0 文字の場合にも、1 文字以上のどのような文字にも一致します。

? すべての単一の文字に一致します。

[...] ブラケット内にある任意の単一の文字に一致します。... は一連の文字を表します。範囲は、開始文字、ハイフン、および終了文字で指定します。感嘆符 ('!') または脱字記号 ('^') が左ブラケットに続いている場合は、ブラケット内の文字が補足される (ブラケット内の文字以外のすべての文字に一致する) ことを意味します。

Qshell で解釈されないようにするには、パターンを引用符で囲むか、またはメタ文字の前に円記号 ('¥') を付ける必要があります。

file オペランドと *pattern* オペランドは、アクションの対象となるファイルを選択するために使用されます。選択されるファイルは、3 つのファイル・セット、つまり候補セット、除外セット、組み込みセットを用いて判別されます。

候補セット
候補セットは、*jarfile* の後ろ、かつ任意の -x または -i の前に表示されているオペランドを使用して決定されます。リスト表示および抽出アクションのデフォルト設定では、候補セットは jar ファイルに入っているすべてのファイルに設定されます。その他のアクションの場合はすべて、候補セットのデフォルト値はありません。

除外セット

除外セットは、-x から次の -x、-i、またはそのコマンド文字列の最後までの間にあるすべての *file* オペランドおよび *pattern* オペランドのリストを使用して決定されます。除外セットのデフォルト値は空のセットです。

組み込みセット

組み込みセットは、-i から次の -i、-x、またはそのコマンド文字列の最後までの間にあるすべての *file* オペランドおよび *pattern* オペランドのリストを使用して決定されます。組み込みセットのデフォルト値は、候補セットの中のすべてのファイルです。

組み込みセット内にあって除外セットにはないすべての候補ファイルが選択されます。

終了状況

- 0 すべてのファイルは正常に処理。
- >0 エラー発生。

例

1. 現行ディレクトリーにある myjar という名前の jar ファイルの中のすべてのファイルをリスト表示する場合: ajar -l myjar
2. myjar の中のすべての .java ファイルをリスト表示する場合: ajar -l myjar ¥*.java
3. myjar のすべてのファイルを抽出して現行ディレクトリーに入れる場合: ajar -x myjar
4. 現行ディレクトリーをルートとするファイル・システム階層内のすべてのディレクトリーおよびファイルの入った、myjar という名前の jar を作成する場合 (この例では、Qshell は '*' を解釈し、それを展開して、現行ディレクトリー内のすべてのファイルおよびディレクトリーを候補ファイルのリストに入れます。) : ajar -c -r myjar *
5. 現行ディレクトリー内のファイルのみに関する項目を備えた、myjar という名前の jar を作成する場合 : ajar -c -D myjar *
6. MANIFEST なしに同じ jar ファイルを作成する場合 (これはすべての実用に対応する ZIP ファイルです。): ajar -c -D -M myjar *
7. 現行ディレクトリーをルートとするファイル・システム階層内の .java ファイルを除くすべてのファイルを備えた、myjar という名前の jar を作成する場合: ajar -c -r myjar * -x ¥*.java
8. 現行ディレクトリーをルートとするファイルシステム階層内の .class ファイルのみの入った、myjar という名前の jar を作成する場合: ajar -c -r myjar * -i ¥*.class
9. .java ファイルを圧縮しないで、myjar という名前の jar を作成する場合: ajar -c -r -n java -- myjar *
10. stdin からファイル・リストを読み取って myjar という名前の jar を作成する場合: ajar -@ -c -r myjar

    サンプル stdin データは次のとおりです。

    ```
    docs
    source
    classes
    -x
    docs/foo/*
    ```

11. bar という名前のファイルを myjar という jar に追加する場合: ajar -a myjar bar
12. foo/bar という名前のファイルを myjar という jar から削除する場合: ajar -d myjar foo/bar

注:

1. 短いオプションはひとまとめにして指定できます (たとえば、-c -v -D と -cvD は同じです)。長いオプション (--create、--verbose、--nodirs など) は、省略した文字列が固有である限り、短くすることができます。
2. jar を作成する場合や jar にファイルを追加する場合は、ファイル名を変更することができます。たとえば、"ajar -c x.jar bin/foo : bin/bar" では、bin/bar という単一の項目があるファイル bin/foo から、jar ファイル x.jar が作成されます。 stdin に次のものが入っていれば、stdin に "ajar -c@ x.jar" を使用して上記と同じ処理ができます。

   ```
   bin/foo : bin/bar
   ```
3. **ajar** を使用するには、QIBM_MULTI_THREADED 環境変数を 'Y' に指定しておく必要があります。

appletviewer - Java アプレットを表示する

appletviewer ツールを使うと、Web ブラウザーを使わないでアプレットを実行できます。 これは、Sun Microsystems, Inc. の appletviewer ツールと互換性があります。

appletviewer ツールは、Qshell インタープリターを使って利用できます。

関連情報:

appletviewer ツール

extcheck - JAR 競合検出ユーティリティー

extcheck ツールを使用して、ターゲット JAR ファイルと現在インストールされている拡張 JAR ファイルとのバージョン競合を検出します。このツールは、Sun Microsystems, Inc. から提供されている keytool と互換性があります。

extcheck ツールの運用には、Qshell インタープリターが必要です。

関連情報:

extcheck ツール

jar - Java ファイルをアーカイブする

jar ツールは、複数のファイルを結合して 1 つの Java アーカイブ (JAR) ファイルにします。このツールは、Sun Microsystems, Inc. から提供されている jar ツールと互換性があります。

jar ツールの運用には、Qshell インタープリターが必要です。

関連概念:

『統合ファイル・システムのファイル』

関連情報:

統合ファイル・システム情報

jar ツール

統合ファイル・システムのファイル: 統合ファイル・システムには、Javatm 関係のクラス・ファイル、ソース・ファイル、ZIP ファイル、および JARファイルが階層ファイル構造の形式で保管されています。ソース・ファイルも、統合ファイル・システムに保管できます。 次の統合ファイル・システムにファイルを保管できます。

- "root" (/) ファイル・システム
- オープン・システム・ファイル・システム (QOpenSys)
- ユーザー定義のファイル・システム

- ライブラリー・ファイル・システム (QSYS.LIB)
- OS/2 Warp Server for IBM i ファイル・システム (QLANSrv)
- 光ファイル・システム (QOPT)

注: その他の統合ファイル・システムは、スレッド・セーフではないためサポートされていません。

関連概念:

147 ページの『jar - Java ファイルをアーカイブする』

jarsigner - JAR 署名と検証

jarsigner ツールを使用して、 JAR ファイルに署名し、署名付き JAR ファイルの署名を検証します。 jarsigner ツールは、JAR ファイルへの署名のための秘密鍵を見つける必要があるときに、keytool が作成し管理する鍵ストアにアクセスします。 J2SDK では、javakey ツールが jarsigner および keytool ツールに置き換わります。 このツールは、Sun Microsystems, Inc. から提供されている jarsigner ツールと互換性があります。

jarsigner ツールの運用には、Qshell インタープリターが必要です。

関連情報:

jarsigner ツール

java - Java インタープリターを実行する

java Qshell コマンドは、Java プログラムを実行します。このコマンドは、Sun Microsystems, Inc. から提供されている java ツールと互換性がありますが、次に示す例外があります。

IBM Developer Kit for Java では、java Qshell コマンドの次のオプションは無視されます。

無視されるオプション	説明
-cs	サポートされていません。
-checksource	サポートされていません。
-debug	システム内部デバッガーでサポートされています。
-noasyncgc	IBM Developer Kit for Java では、ガーベッジ・コレクションは常に実行されています。
-noclassgc	IBM Developer Kit for Java では、ガーベッジ・コレクションは常に実行されています。
-prof	システムには、独自のパフォーマンス・ツールがあります。
-ss	適用されません。
-oss	適用されません。
-t	システムは、独自のトレース機能を使用します。
-verify	システムでは検証は常に実行されます。
-verifyremote	システムでは検証は常に実行されます。
-noverify	システムでは検証は常に実行されます。

java Qshell コマンドは、新規オプションをサポートしています。次のオプションがサポートされます。

サポートされるオプション	説明
-secure	CLASSPATH に指定されたディレクトリーへの共通書き込みアクセスを検査します。
-gcfrq	ガーベッジ・コレクションの頻度を指定します。
-gcpty	ガーベッジ・コレクションの優先順位を指定します。
-opt	最適化レベルを指定します。
-verbosegc	ガーベッジ・コレクションをスイープするたびに、メッセージを表示します。

CL 解説書の「Java の実行 (RUNJVA)」コマンドの項に、これらの新規オプションの詳細な説明があります。 Java プログラムの管理については、CL 解説書の「Java プログラムの作成 (CRTJVAPGM)」コマンド、「Java プログラムの削除 (DLTJVAPGM)」コマンド、および「Java プログラムの表示 (DSPJVAPGM)」コマンドの項を参照してください。

java Qshell コマンドの運用には、Qshell インタープリターが必要です。

関連情報:

java ツール

javac - Java プログラムをコンパイルする

javac ツールは Java プログラムをコンパイルします。このツールは、Sun Microsystems, Inc. から提供されている javac ツールと互換性があります。

javac ツールの運用には、Qshell インタープリターが必要です。

関連情報:

javac ツール

javadoc - Java 文書を生成する

javadoc ツールは API 文書を生成します。 このツールは、Sun Microsystems, Inc. から提供されている javadoc ツールと互換性があります。

javadoc ツールの運用には、Qshell インタープリターが必要です。

関連情報:

javadoc ツール

javah - C ヘッダーまたはスタブ・ファイルを生成する

javah ツールは、Java ネイティブ・メソッドの実装を容易にします。このツールは、Sun Microsystems, Inc. から提供されている javah ツールと、多少の例外を除いて互換性があります。

注:	ネイティブ・メソッドを書くと、アプリケーションは 100% Pure Java ではなくなります。また、プラットフォーム間で直接移植できなくなります。 ネイティブ・メソッドは、その特性として、プラットフォーム固有またはシステム固有のものになります。 ネイティブ・メソッドを使用すると、アプリケーションの開発および保守のコストが増える可能性があります。

javah ツールの運用には、Qshell インタープリターが必要です。このツールは、Java クラス・ファイルを読み取って、現行作業ディレクトリー内に C 言語ヘッダー・ファイルを作成します。作成されるヘッダー・ファイルは、ストリーム・ファイル (STMF) です。このヘッダー・ファイルを i5/OS の C プログラムに組み込むには、最初にファイル・メンバーにこのファイルをコピーしておく必要があります。

javah ツールは、Sun Microsystems, Inc. が提供するツールと互換性があります。ただし、次のオプションは、指定してもシステムはこれを無視します。

無視されるオプション	説明
-td	javah ツールでは、一時ディレクトリーは不要です。
-stubs	IBM i で Java がサポートするのは、ネイティブ・メソッドの Java ネイティブ・インターフェース (JNI) 形式のみです。スタブが必要なのは、JNI 以前の形式のネイティブ・メソッドの場合のみです。
-trace	これは .c スタブ・ファイル出力に関連するオプションです。IBM i の Java ではこの出力はサポートされていません。
-v	サポートされていません。

注 :	**-jni** オプションは常に指定する必要があります。システムは、JNI より前のネイティブ・メソッドの実装をサポートしていません。

関連情報:

javah ツール

javakey - Java セキュリティー・キーおよび証明書を管理する

javakey ツールは、暗号鍵、証明書の生成と管理、およびアプレット用のディジタル署名の生成に使用できます。 このツールは、Sun Microsystems, Inc. から提供されている javakey ツールと互換性があります。

アプレットのパッケージ化および署名方法は、ブラウザーによって異なります。ご使用のブラウザーの資料を参照して、Java™ JAR ファイル形式および javakey アプレット署名と互換性があることを確認してください。

注 :	javakey ツールによって作成されたファイルには、機密情報が入っています。公開鍵ファイルおよび秘密鍵ファイルの保護には、統合ファイル・システムの適切なセキュリティー機能が使われています。

javakey ツールの運用には、Qshell インタープリターが必要です。

関連情報:

統合ファイル・システム

javakey ツール

javap - コンパイル済みの Java プログラムを逆アセンブルする

javap ツールは、コンパイル済みの Java ファイルを逆アセンブルして、Java プログラムのソース表現を印刷します。オリジナルのソース・コードがもうシステムにない場合は、このツールが役立ちます。

このツールは Sun Microsystems, Inc. から提供されている javap ツールと互換性がありますが、次に示す例外があります。

無視されるオプション	説明
-b	このオプションは無視されます。後方互換性は不要です。
-p	IBM i プラットフォームでは、**-p** は有効なオプションではありません。**-private** と、フルスペルで指定する必要があります。
-verify	このオプションは無視されます。 javap ツールは、検証を実行しません。

javap ツールの運用には、Qshell インタープリターが必要です。

注 :	javap ツールを使用したクラスの逆アセンブルは、それらのクラスのライセンス契約に違反する場合があります。javap ツールを使用する前に、ライセンス契約の内容をよく確認してください。

関連情報:

javap ツール

keytool - キーと証明書の管理ツール

keytool を使用して、公開鍵と秘密鍵のペアや自己署名付き証明書を作成し、鍵ストアを管理します。J2SDK では、javakey ツールが jarsigner および keytool ツールに置き換わります。 このツールは、Sun Microsystems, Inc. から提供されている keytool と互換性があります。

keytool ツールの運用には、Qshell インタープリターが必要です。

関連情報:

keytool - キーと証明書の管理ツール

native2ascii - ネイティブ文字を ASCII に変換する

native2ascii ツールは、ネイティブ・コード文字 (Latin 1 でも Unicode でもない文字) のファイルを、Unicode コード文字のファイルに変換します。 このツールは、Sun Microsystems, Inc. から提供されている native2ascii ツールと互換性があります。

native2ascii ツールの運用には、Qshell インタープリターが必要です。

関連情報:

native2ascii ツール

policytool - ポリシー・ファイルの作成および管理ツール

policytool でユーザーのインストールにおける Java セキュリティー・ポリシーを定義する外部ポリシー構成ファイルを作成および変更できます。このツールは、Sun Microsystems, Inc. から提供されている policytool と互換性があります。

関連情報:

「IBM Developer Kit for Java」の『Native Abstract Windowing Toolkit』

policytool - ポリシー・ファイルの作成および管理ツール

rmic - Java RMI スタブをコンパイルする

rmic ツールは、Java オブジェクト用のスタブ・ファイルおよびクラス・ファイルを生成します。 このツールは、Sun Microsystems, Inc. から提供されている rmic ツールと互換性があります。

rmic ツールの運用には、Qshell インタープリターが必要です。

関連情報:

rmic ツール

rmid - Java RMI 活動化システム

rmid ツールを使用して活動化システム・デーモンを開始します。その結果オブジェクトが Java 仮想マシンに登録され、活動化されます。このツールは、Sun Microsystems, Inc. から提供されている rmid ツールと互換性があります。

rmid ツールの運用には、Qshell インタープリターが必要です。

関連情報:

rmid ツール

rmiregistry - リモート・オブジェクト・レジストリーを開始する

rmiregistry ツールは、指定されたポート上でリモート・オブジェクト・レジストリーを開始します。 このツールは、Sun Microsystems, Inc. から提供されている rmiregistry ツールと互換性があります。

rmiregistry ツールの運用には、Qshell インタープリターが必要です。

関連情報:

rmiregistry ツール

serialver - シリアル・バージョンを戻す

serialver ツールは、1 つ以上のクラスのバージョン番号またはシリアル化固有の ID を戻します。 このツールは、Sun Microsystems, Inc. から提供されている serialver ツールと互換性があります。

serialver ツールの運用には、Qshell インタープリターが必要です。

関連情報:

serialver ツール

tnameserv - ネーム・サービス

tnameserv ツールはネーム・サービスへのアクセスを提供します。このツールは、Sun Microsystems, Inc. から提供されている tnameserv ツールと互換性があります。

tnameserv ツールの運用には、Qshell インタープリターが必要です。

関連情報:

tnameserv ツール

ジョブ管理用のユーティリティー

ジョブ管理用のユーティリティーを表示するには、このリンクを選択します。

getjobid - ジョブ情報を表示する

構文

getjobid [-csv] [*pid ...*]

getjobid -j [-csv] [*qualified-job ...*]

説明

getjobid ユーティリティーは、指定されたプロセス用の修飾ジョブ名およびプロセス ID を標準出力に書き込みます。修飾ジョブ名は、*number/user/name* 形式のストリングです。 *number* は 6 桁の 10 進数であり、*user* はジョブが開始されたユーザー・プロファイルであり、*name* はジョブの名前です。

最初の構文形式では、プロセスはプロセス ID を使用して識別されます。 2 番目の構文形式では、プロセスは修飾ジョブ名を使用して識別されます。

-v オプションが指定されている場合は、**getjobid** によって、各指定プロセスのプロセス ID、親プロセス ID、プロセス・グループ、現在の状況、修飾ジョブ名が表示されます。

注:	これは i5/OS に固有のユーティリティーです。

オプション

-c 指定プロセスの現行の子プロセスのすべてについての情報が表示されます。

-j プロセスは修飾ジョブ名を使用して識別されます。

-s 修飾ジョブ名だけ付いた短い形式を表示します。

-v プロセス ID、親プロセス ID、プロセス・グループ、現在の状況、修飾ジョブ名を含む、指定プロセスについての詳細情報を表示します。

オペランド

各 *pid* は、システム上のアクティブ・プロセスの 10 進プロセス ID です。 *pid* が指定されていない場合は、 **getjobid** では、現行のプロセスに関する情報が表示されます。

各 *qualified-job* は、システム上のアクティブ・プロセスの修飾ジョブ名です。修飾ジョブ名は、*number/user/name* 形式のストリングです。 *number* は 6 桁の 10 進数であり、*user* はジョブが開始されたユーザー・プロファイルであり、*name* はジョブの名前です。

終了状況

- 0 正常終了。
- >0 エラー発生。終了状況の値は、情報を取得できなかったプロセスの個数です。

例

1. 現行のプロセスの修飾ジョブ名を表示する場合:

   ```
   getjobid - ジョブ情報を表示する
   ```

2. 3 つのプロセスについての詳細情報を表示する場合:

```
getjobid -v 318 942 1130
```

3. 1 つのプロセスの修飾ジョブ名の短い形式を表示する場合:

```
getjobid -s 325
```

4. 修飾ジョブ名で識別されたプロセスについての詳細情報を表示する場合:

```
getjobid -jv 325411/SHELLTST/QZSHCHLD
```

関連タスク:

『jobs - 現行セッションのジョブの状況を表示する』

158 ページの『ps - プロセス状況を表示する』

hash - ユーティリティー・ロケーションを記憶または報告する

hash [-p *filename* **] [** *utility ...* **]**

hash -r

説明

hash ユーティリティーは、記憶済みユーティリティー・ロケーションのリストに *utility* を追加したり、記憶済みのすべてのユーティリティーをそのリストから削除したりします。デフォルトでは、**hash** は *utility* の検索にパス検索を使用します。

引数を指定しなかった場合は、**hash** はリストの内容を表示します。 前回の cd コマンド以後に検索されていない項目にはアスタリスクが付いています。この項目は無効になっている可能性があります。

オプション

-p *filename*
: *utility* の検索にパス検索を使用しません。 *utility* の場所として指定された *filename* を使用します。

-r
: 前に記憶したユーティリティー・ロケーションをすべて削除します。

オペランド

各 *utility* は、記憶されたユーティリティー・ロケーションのリストに追加されます。

終了状況

- 0 正常終了。

関連タスク:

90 ページの『cd - 作業ディレクトリーを変更する』

jobs - 現行セッションのジョブの状況を表示する

構文

jobs [-ln] [*job ...*]

説明

jobs を使うと、**qsh** で開始された活動状態のジョブについての情報を表示できます。 それぞれのジョブごとに、**qsh** は次の情報を表示します。

- ジョブ番号。大括弧 [] で囲んで表示されます。

- 状況 (Running (実行中)、Done (完了)、Terminated (終了) など)。
- ジョブの戻り値。戻り値が 0 より大きく、ジョブ状況が Done (完了) のときに、小括弧 () で囲んで表示されます。
- ジョブのコマンド行。

オプション

-l 指定されたジョブのプロセスごとに、状況を表示します。

-n 状況が変化したがまだ報告されていないジョブのみの状況を表示します。

オペランド

各 *job* は、活動状態のジョブを指します。 job には次のものを指定できます。

- プロセス ID を示す数値。
- ジョブ番号を示す %number。
- 文字列で始まる名前を持つジョブを示す %string。

job を指定しなかった場合は、**qsh** はすべての活動状態のジョブの状況を表示します。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. ジョブ番号 1 の状況を表示する場合: **jobs** %1
2. プロセス ID 16107 の状況を表示する場合: **jobs** 16107
3. ls ユーティリティーを実行しているジョブの状況を表示する場合: **jobs** %ls
4. 活動状態のすべてのジョブの状況を表示する場合: **jobs**

関連タスク:

153 ページの『getjobid - ジョブ情報を表示する』
『kill - プロセスを終了するかまたはシグナルを送る』
162 ページの『wait - プロセスの完了を待つ』
158 ページの『ps - プロセス状況を表示する』

kill - プロセスを終了するかまたはシグナルを送る

構文

kill [**-s** *signame*] *job* ...

kill [**-n** *signum*] *job* ...

kill [**-sig**] *job* ...

kill -l [*signal* ...]

説明

kill を使うと、指定した *jobs* にシグナルを送ることができます。 シグナルは次のいずれかで指定できます。

- *signame* - シグナル名。
- *signum* - シグナル番号。
- *sig* - 負符号 (**-**) のあとに、スペースを付けないでシグナル名またはシグナル番号を指定します。

注:	i5/OS では、有効なシグナル番号が他のシステムのシグナル番号と異なる場合があります。 **-l** オプションを指定すれば、有効なシグナル名をリスト表示することができます。可搬性を確保するために、シグナル名は必ず指定するようにしてください。

オプション

-l シグナル名をリストします。引数を指定しなかった場合は、**qsh** はすべてのシグナル名を表示します。 *signal* が名前である場合は、**qsh** はその名前に対応するシグナル番号を表示します。 *signal* が番号である場合は、**qsh** はその番号に対応するシグナル名を表示します。

-n シグナル番号。

-s 大文字または小文字のシグナル名。

オペランド

各 *job* は、活動状態のジョブを指します。 job には次のものを指定できます。

- プロセス ID を示す数値。
- ジョブ番号を示す %number。
- 文字列で始まる名前を持つジョブを示す %string。

終了状況

- 0 正常終了。
- >0 不成功の場合。 **-l** オプションを指定しなかった場合は、 **qsh** がシグナルを送信できなかったジョブの数が、終了状況に示されます。

例

1. USR1 シグナルをプロセス ID 16711 に送る場合：**kill** -s USR1 16711
2. USR1 シグナルをジョブ 1 に送る場合: **kill** -n 7 %1
3. 有効なシグナル名をリストする場合: **kill** -l

関連タスク:

154 ページの『jobs - 現行セッションのジョブの状況を表示する』
160 ページの『trap - シグナルをトラップする』
162 ページの『wait - プロセスの完了を待つ』

liblist - ライブラリー・リストを管理する

構文

liblist [-acdfl] [*library* ...]

説明

liblist を使うと、ライブラリー・リストのユーザー部分からライブラリーを削除したり、現行ライブラリーを設定したり、または現行ジョブ用のライブラリー・リストを表示したりすることができます。

ライブラリー・リストのユーザー部分にライブラリーを追加するには、**-a** オプションとライブラリーのリストを指定します。デフォルトでは、ライブラリーはライブラリー・リストの先頭のユーザー部分に追加されます。

ライブラリー・リストのユーザー部分からライブラリーを削除するには、**-d** オプションとライブラリーのリストを指定します。

-c オプションを指定すると、現行ライブラリーが *library* に設定されます。 **-c** オプションと **-d** オプションの両方を指定すると、現行ライブラリーの設定を解除できます。

引数を何も指定しなかった場合は、**qsh** は現行ライブラリー・リストを表示します。 出力の各行には、それぞれライブラリー名とライブラリーのタイプが示されます。 ライブラリーのタイプは次のいずれかです。

- SYS ライブラリー・リストのシステム部分を成すライブラリー。
- PRD ライブラリー・リストのプロダクト部分を成すライブラリー。
- CUR 現行ライブラリー。
- USR ライブラリー・リストのユーザー部分を成すライブラリー。

オプション

-a ライブラリー・リストのユーザー部分に *library* を追加します。

-c 現行ライブラリーを *library* に追加します。

-d ライブラリー・リストのユーザー部分から *library* を削除します。ただし **-c** オプションも同時に指定した場合は、現行ライブラリーの設定が解除されます。

-f **-a** オプションと共に指定した場合、ライブラリー・リストのユーザー部分の先頭に *library* を追加します。

-l **-a** オプションと共に指定した場合、ライブラリー・リストのユーザー部分の末尾に *library* を追加します。

オペランド

library は、指定したオプションに応じて、ライブラリー・リストのユーザー部分に追加されるか、またはそこから削除されるライブラリーです。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. ライブラリー MYLIB をライブラリー・リストに追加する場合: **liblist** -a MYLIB
2. ライブラリー MYLIB をライブラリー・リストから削除する場合: **liblist** -d MYLIB
3. 現行ライブラリーを MYLIB に設定する場合: **liblist** -c MYLIB
4. 現行ライブラリーの設定を解除する場合: **liblist** -cd
5. ライブラリー・リストを表示する場合: **liblist**

ps - プロセス状況を表示する

構文

ps [-Aaefjlt] [-o *format* **] [-p** *pidlist* **] [-s** *sbslist* **] [-u** *userlist* **]**

説明

ps ユーティリティーはプロセスに関する情報を表示します。 **ps** からの出力には、以下のフィールドが含まれます。

CGROUP
プロセスの現行の 1 次グループ・プロファイル。

CMD そのプロセスにより最後に実行が行われたプログラム、メニュー、またはコマンド。

CUSER
プロセスの現行のユーザー・プロファイル。

DEVICE
プロセスと関連した装置記述オブジェクトの名前。

ETIME プロセスが開始されてからの経過時間。時間は [[*dd*-]*hh*:]*mm*:*ss* 形式で表示され、 *dd* は日数、*hh* は時間数、*mm* は分数、*ss* は秒数です。

FUNCTION
そのプロセスにより最後に実行が行われたプログラム、メニュー、またはコマンド。

JOBID プロセスの修飾ジョブ名。修飾ジョブ名は、*number*/*user*/*name* 形式のストリングです。 *number* は 6 桁の 10 進数であり、*user* はジョブが開始されたユーザー・プロファイルであり、*name* はジョブの名前です。

JOBNAME
修飾ジョブ名のジョブ名コンポーネント。

JOBNUM
修飾ジョブ名のジョブ番号コンポーネント。

NTHREADS
プロセス内で現在実行されているスレッド数 (10 進数)。

PCPU パーセントとして表示される、使用可能な CPU 時間に対する最近使用された CPU 時間の比率。

PGID プロセス・グループ ID 番号 (10 進数)。

PID プロセス ID 番号 (10 進数)。

PPID 親プロセス ID 番号 (10 進数)。

PRI プロセスの現行の優先順位 (10 進数)。小さい数ほど優先順位が高いことを意味します。

SBS プロセスが稼働しているサブシステム。

STATUS
プロセスの現行の状況。

STIME プロセスが開始された日付および時刻。デフォルトでは、日付および時刻は *mm-dd-yyyy hh*:*nn*:*ss* 形式で表示されます。 *mm* は月、*dd* は日、*yyyy* は年、*hh* は時、*nn* は分、*ss* は秒です。
LC_TIME 環境変数が設定されている場合、日付および時刻は、指定されたロケールの LC_TIME カテゴリーで d_t_fmt キーワードによって指定されたフォーマットで表示されます。

THCOUNT
プロセス内で現在実行されているスレッド数 (10 進数)。

TIME プロセスで使用された CPU 時間 (秒数)。時間は [[*dd*-]*hh*:]*mm*:*ss* 形式で表示され、 *dd* は日数、*hh* は時間数、*mm* は分数、*ss* は秒数です。

TMPSZ
プロセスによって使用されるメガバイトの一時記憶域の量 (10 進数)。

TYPE プロセスのタイプ。

USER 修飾ジョブ名のユーザー・プロファイル・コンポーネント。

UID 修飾ジョブ名のユーザー・プロファイル・コンポーネントに対応するユーザー ID 番号。

デフォルトでは、**ps** は、現在のユーザーが所有するプロセスに関する
PID、DEVICE、TIME、FUNCTION、STATUS、および JOBID の各フィールドを表示します。 **ps** によって表示されるフィールドを選択するには、**-o** オプションを使用してください。

その他のプロセスについての情報を表示するためには、*JOBCTL 特殊権限が必要です。

オプション

-a 5250 端末に関連したすべてのプロセスの情報を表示します。

-A すべてのプロセスの情報を表示します。これには、アクティブなプロセス、ジョブ・キューにあるプロセス、または出力キューにあるプロセスが含まれます。

-e アクティブなプロセスを出力に組み込みます。

-f 全リストを表示します。出力には、USER、PID、PPID、STIME、DEVICE、TIME、および FUNCTION の各フィールドが含まれます。

-j ジョブ・キューにあるプロセスを出力に組み込みます。

-l 長形式のリストを表示します。出力には、USER、PID、PPID、PRI、STATUS、JOBID、STIME、DEVICE、TIME、および FUNCTION の各フィールドが含まれます。

-o *format*
format にあるフォーマット指定に基づいて情報を表示します。複数の **-o** オプションを指定することができます。

-p *pidlist*
プロセス ID 番号が *pidlist* に指定されているプロセスの情報を書き出します。 *pidlist* は、ブランクまたはコンマで区切られたリスト形式の単一引数である必要があります。

-s *sbslist*
sbslist で指定したサブシステムで実行中のプロセスの情報を書き出します。 *sbslist* は、ブランクまたはコンマで区切られたリスト形式の単一引数である必要があります。

-t アウト・キューにあるプロセスを出力に組み込みます。

-u *userlist*
ユーザー ID 番号またはユーザー名が *userlist* に指定されているプロセスの情報を書き出します。*userlist* は、ブランクまたはコンマで区切られたリスト形式の単一引数である必要があります。

環境変数

ps は、次の環境変数の影響を受けます。

LANG LC_ で始まる変数を使って明示的に設定されていないロケール・カテゴリー用のデフォルト値を提供します。

LC_TIME
日付および時刻属性の出力フォーマットを定義します。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

153 ページの『getjobid - ジョブ情報を表示する』

154 ページの『jobs - 現行セッションのジョブの状況を表示する』

sleep - 呼び出しを一定期間中断する

構文

sleep *time*

説明

sleep を使用すると、プロセスの実行を *time* に指定した秒数だけ中断できます。

オプション

なし。

オペランド

time の値は正整数でなければなりません。

終了状況

- 0 正常終了。
- >0 *time* が無効。

trap - シグナルをトラップする

構文

trap [*action condition ...*]

trap -p [*condition ...*]

trap -l

説明

trap ユーティリティーは、 *condition* が発生したときに **qsh** によって実行される *action* を設定します。**qsh** は、**trap** を実行しているときに *action* を展開し、*condition* が発生したときに再度展開します。

-p オプションを指定した場合、 **trap** は、指定した *conditions* についての現行の *action* を表示します。

-l オプションが指定されている場合、**trap** は、すべてのシグナル名とその対応する番号のリストを表示します。

引数を何も指定しないと、**trap** は現在定義されているトラップをリスト表示します。

オプション

-l すべてのシグナル名とその対応する番号のリストを表示します。

-p 各トラップを再入力可能な形式で表示します。

オペランド

action には次のものを指定できます。

- *condition* が発生したときにそれを無視するにはヌル。
- *condition* を元の値にリセットするには負符号 (**-**)。
- *condition* が発生したときに実行するコマンド。

condition には次のものを指定できます。

- シグナルの名前または番号。 **trap -l** を使用すると、有効なシグナルのリストを表示することができます。可搬性を確保するために、シグナル名は必ず指定するようにしてください。
- **0** または **EXIT**。 **qsh** は、シェルの終了時に *action* を実行します。
- **ERR**。 **qsh** は、コマンドの戻り状況が 0 以外の場合に *action* を実行します。
- **DEBUG**。 **qsh** は、各単純コマンドのあとで *action* を実行します。

複数の条件が同時に発生した場合は、**qsh** は次の順序でトラップを実行します。

1. **DEBUG** (指定されている場合)。
2. **ERR** (指定されていて該当する場合)。
3. その他の指定されているトラップ (シグナル番号順)。
4. **EXIT**。

終了状況

- 0 正常終了。
- >0 無効な *condition* が指定された。

例

1. ERR 条件についてトラップを設定する場合:
   ```
   trap `print Command failed' ERR
   ```
2. ERR 条件を無視する場合:
   ```
   trap "" ERR
   ```
3. ERR 条件を元の値にリセットする場合:
   ```
   trap - ERR
   ```
4. ERR 条件についての現行のアクションを表示する場合:
   ```
   trap -p ERR
   ```
5. 現在定義されているすべてのトラップを表示する場合:
   ```
   trap - シグナルをトラップする
   ```

関連タスク:

155 ページの『kill - プロセスを終了するかまたはシグナルを送る』

162 ページの『wait - プロセスの完了を待つ』

wait - プロセスの完了を待つ

構文

wait [*job* ...]

説明

wait を使うと、指定した *jobs* が完了するまで待つように指定することができます。*job* の指定がない場合は、**qsh** はすべての子プロセスが終了するまで待ちます。

オプション

なし。

オペランド

各 *job* は、活動状態のジョブを指します。 job には次のものを指定できます。

- プロセス ID を示す数値。 **qsh** は、指定されたプロセスが終了するまで待ちます。
- ジョブ番号を示す %number。 **qsh** は、このジョブ内のすべてのプロセスが終了するまで待ちます。
- 文字列で始まる名前を持つジョブを示す %string。 **qsh** は、このジョブ内のすべてのプロセスが終了するまで待ちます。

終了状況

job が指定されなかった場合:

- 0 すべてのジョブ実行が終了。
- >0 不成功の場合。

少なくとも 1 つの *job* が指定された場合： 最後の *job* の終了状況。

例

1. プロセス ID 16825 が終了するまで待つ場合: **wait** 16825
2. ジョブ番号 5 が終了するまで待つ場合: **wait** %5

関連タスク:

154 ページの『jobs - 現行セッションのジョブの状況を表示する』
155 ページの『kill - プロセスを終了するかまたはシグナルを送る』
160 ページの『trap - シグナルをトラップする』

Kerberos 信任状およびキー・テーブル用のユーティリティー

Kerberos 信任状およびキー・テーブル用のユーティリティーを表示するには、このリンクを選択します。

- kdestroy - Kerberos 信任状キャッシュを破棄する
- keytab - Kerberos キー・テーブル・ファイルを管理する
- kinit - Kerberos 発券許可証を取得または更新する
- klist - Kerberos 信任状キャッシュまたはキー・テーブル・ファイルの内容を表示する
- ksetup - Kerberos レルム用の LDAP ディレクトリーにある Kerberos サービス・エントリーを管理する

LDAP ディレクトリー・サーバーのためのユーティリティー

LDAP ディレクトリー・サーバーのためのユーティリティーを表示するには、このリンクを選択します。

- ldapadd - LDAP 項目の追加ツール
- ldapmodify - LDAP 項目の変更ツール
- ldapchangepwd - LDAP パスワードの変更ツール
- ldapmodrdn - LDAP 相対識別名 (RDN) の変更ツール
- ldapdiff - LDAP 複製の同期の比較ツール
- ldapdelete - LDAP 項目の削除ツール
- ldapexop - LDAP 操作の拡張ツール
- ldapsearch - LDAP サーバーの検索ツール

パラメーターおよび変数を取り扱うユーティリティー

パラメーターおよび変数を取り扱うユーティリティーを表示します。

declare - 変数を宣言し、属性設定をする

構文

declare [-Eilrux] *name* **[**=*value***]** ...

declare [+Eilrux] *name* **[**=*value***]** ...

declare -fF [*name* ... **]**

declare -p *name* ...

declare

説明

declare ユーティリティーは、変数の宣言、変数への値の割り当て、変数の属性の設定または設定解除、さらにシェル関数の定義の表示を行います。シェル関数内で使用される場合、**declare** は、変数 *name* をその関数に対しローカルに設定します。

最初の構文形式では、**declare** は *name* 変数を宣言します。また、オプションでその変数に、指定された *value* を割り当てます。オプションが指定されると、対応する変数の属性はオンになります。

2 番目の構文形式では、**declare** は *name* 変数を宣言します。また、オプションでその変数に、指定された *value* を割り当てます。オプションが指定されると、対応する変数の属性はオフになります。

3 番目の構文形式では、*name* が指定されていない場合、あるいは *name* にシェル関数が指定されている場合、**declare** はすべてのシェル関数の名前と定義を表示します。

4 番目の構文形式では、**declare** は、*name* により指定された変数の属性と値を、再入力可能なフォーマットで表示します。

5 番目の構文形式では、**declare** はすべての変数の名前と値を表示します。

オプション

-E 変数に浮動小数点属性を設定します。変数に割り当てがなされると、その値は浮動小数点数として評価されます。

-f シェル関数の名前と定義を表示します。

-F シェル関数の名前を表示します。

-i 変数に整数属性を設定します。変数に割り当てがなされると、その値は整数として評価されます。

-l 変数に英小文字属性を設定します。変数に割り当てがなされると、その値は英小文字に設定されます。

-p 各変数を再入力可能な形式で表示します。

-r 変数に読み取り専用属性を設定します。その変数の値は、後続の割り当てによって変更することも、設定解除することもできません。 *value* も指定した場合は、読み取り専用属性を設定する前に変数の値が更新されます。

-u 変数に英大文字属性を設定します。変数に割り当てがなされると、その値は英大文字に設定されます。

-x 変数にエクスポート属性を設定します。その変数は、それ以降に実行されるコマンドの環境の中に自動的に配置されます。

オペランド

それぞれの *name* は、有効なシェル変数名でなければなりません。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連概念:

25 ページの『複合コマンド』
複合コマンドは、他のコマンドの制御フローを提供します。複合コマンドは予約語で始まり、それに対応する予約語で終わります。

9 ページの『変数』
qsh は、開始されると、定義済みの環境変数に基づいてシェル変数を初期設定します。変数は、データを格納するために使用されます。 .

関連タスク:

『export - 変数に対してエクスポート属性を設定する』

174 ページの『let - 算術式を評価する』

165 ページの『local - 関数内でローカル変数を割り当てる』

167 ページの『readonly - 変数に対して読み取り専用属性を設定する』

168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』

170 ページの『typeset - 変数を宣言し、属性設定をする』

171 ページの『unset - 変数および関数の値を設定解除する』

export - 変数に対してエクスポート属性を設定する

構文

export [-ps] [*name* [=*value*] ...]

説明

export を使うと、*name* で指定される変数に対してエクスポート属性を設定できます。 エクスポート属性を持つ変数は、それ以降に実行されるコマンドの環境の中に自動的に配置されます。

引数を何も指定しなかった場合は、**qsh** は、エクスポート属性を持つすべての変数とその値のリストを表示します。

オプション

-p 出力の各行の前に export という語を付けます。したがって、各行をそのまま再入力に使えます。

-s 変数を現行プロセスの中の環境変数としても設定します。

オペランド

各 *name* には、現行の環境での変数を指定します。 *value* も指定した場合は、変数の値が更新されます。

終了状況

- 0 正常終了。

例

1. 既存の変数についてエクスポート属性を設定する場合:

   ```
   export ALPHA
   ```

2. 新しい変数の値とエクスポート属性を設定する場合:

   ```
   export ALPHA=one
   ```

3. エクスポート属性を持つすべての変数をリスト表示する場合:

   ```
   export - 変数に対してエクスポート属性を設定する
   ```

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』
『local - 関数内でローカル変数を割り当てる』
167 ページの『readonly - 変数に対して読み取り専用属性を設定する』
168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』
171 ページの『unset - 変数および関数の値を設定解除する』
166 ページの『printenv - 環境変数の値を表示する』
170 ページの『typeset - 変数を宣言し、属性設定をする』

local - 関数内でローカル変数を割り当てる

構文

local [*name* [=*value*] ...]

説明

local を使うと、変数を関数に対し、ローカルにすることができます。 ローカル変数は、周囲のスコープ内に同じ名前の変数があれば、その変数から、初期値と、エクスポートされた属性および読み取り専用の属性を継承します。 同名の変数がない場合は、その変数は最初は未設定になります。

qsh は動的スコープを使用します。したがってたとえば変数 *alpha* を関数 *foo* 用のローカル変数にした場合、この関数が次に関数 *bar* を呼び出したときには、*bar* 内での変数 *alpha* に対する参照は、*alpha* という名前のグローバル変数を指さないで、*foo* の中で宣言されている変数を指すことになります。

特殊パラメーター **-** は、ローカルにすることができる唯一の特殊パラメーターです。 **-** をローカルにすると、関数の内部で **set** によって変更されたシェル・オプションは、その関数から戻った時点ですべて元の値に戻ります。

オプション

なし。

オペランド

各 *name* には、現行の環境での変数を指定します。 *value* も指定した場合は、変数の値が更新されます。

終了状況

- 0 正常終了。
- >0 関数の外部から呼び出された場合。

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』
164 ページの『export - 変数に対してエクスポート属性を設定する』
167 ページの『readonly - 変数に対して読み取り専用属性を設定する』
168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』
170 ページの『typeset - 変数を宣言し、属性設定をする』
171 ページの『unset - 変数および関数の値を設定解除する』

printenv - 環境変数の値を表示する

構文

printenv [-s] [*name* **]**

説明

printenv ユーティリティーは、環境変数 *name* の値を表示します。 *name* が指定されていない場合、**printenv** はすべての現行環境変数を "name=value" のフォーマットで 1 行に 1 つずつ表示します。デフォルトでは、**printenv** はジョブ環境変数を表示します。

オプション

-s システム環境変数を表示します。

オペランド

name は、現行環境またはシステム環境変数内の環境変数名です。

終了状況

- 0 正常終了。
- >0 *name* が現在定義されていない場合

関連タスク:

164 ページの『export - 変数に対してエクスポート属性を設定する』
49 ページの『env - コマンド呼び出し用の環境を設定する』

readonly - 変数に対して読み取り専用属性を設定する

構文

readonly [**-p**] [*name* [=*value*] ...]

説明

readonly を使うと、*name* に指定する変数に対して読み取り専用属性を設定できます。読み取り専用属性を持つ変数の値は、後続の割り当てによって変更することも、設定解除することもできません。

読み取り専用属性を持つ変数の値は **qsh** で変更されることがあります。たとえば、 **PWD** が読み取り専用属性を持っている場合に、現行作業ディレクトリーを変更すると、この変数の値が変更されます。

引数を指定せずに **qsh** を入力すると、読み取り専用属性を持つ変数とその値のリストが表示されます。

オプション

-p 出力の各行の前に readonly という語を付けます。したがって、各行をそのまま再入力に使えます。

オペランド

各 *name* には、現行の環境での変数を指定します。 *value* も指定した場合は、読み取り専用属性を設定する前に変数の値が更新されます。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. 既存の変数について読み取り専用属性を設定する場合:

   ```
   readonly ALPHA
   ```

2. 新しい変数の値と読み取り専用属性を設定する場合:

   ```
   readonly ALPHA=one
   ```

3. 読み取り専用属性を持つすべての変数をリスト表示する場合:

   ```
   readonly - 変数に対して読み取り専用属性を設定する
   ```

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』
164 ページの『export - 変数に対してエクスポート属性を設定する』
165 ページの『local - 関数内でローカル変数を割り当てる』
168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』
170 ページの『typeset - 変数を宣言し、属性設定をする』
171 ページの『unset - 変数および関数の値を設定解除する』

set - オプションおよび定位置パラメーターを設定または設定解除する

構文

set [**-abCefFjlmntuvx-**] [**-o** *option*] [*argument* ...]

set [**+abCefFjlmntuvx-**] [**+o** *option*] [*argument* ...]

説明

set ユーティリティーは、次のことを実行できます。

- オプションも引数も指定しないと、すべてのシェル変数の名前と値が表示されます。
- オプション を指定せずに **-o** option を指定すると、オプションの設定内容が表示されます。
- **-** (負符号) に続けてオプション文字を指定した場合、または **-o** *option* を指定した場合は、オプションが設定されます。
- **+** (正符号) に続けてオプション文字を指定した場合、または **+o** *option* を指定した場合は、オプションの設定が解除されます。
- *arguments* を指定すると、定位置パラメーターが設定されます。
- *argument* を指定せずに **--** を指定すると、定位置パラメーターの設定が解除されます。

オプション

すべての単一文字オプションには、それぞれ対応する **-o** *option* があります。 次の各文字オプションのあとの括弧内に示されているのが、オプション値です。 **qsh** は次のオプションをサポートしています。

-a (allexport)
値が代入されている各変数にエクスポート属性を設定します。

-b (notify)
バックグラウンド・ジョブ完了の非同期通知を使用可能にします。

-C (noclobber)
リダイレクト演算子 **>** を使って既存のファイルを上書きしません。

-e (errexit)
対話式オプションが設定されていない場合に、テストされていないコマンドが失敗すると即時に終了します。 **if**、**elif**、**while**、または **until** を制御するためにコマンドを使用した場合、または、**&&** 演算子または || 演算子の左側のオペランドとしてコマンドを使用した場合は、そのコマンドの終了状況は明示的にテスト済みであると見なされます。

-f (noglob)
パス名展開を使用不可にします。

-F (float)
算術式 の中で浮動小数点演算を使用できるようにします。

-j (jobtrace)
ジョブのトレースを使用可能にします。 **qsh** が i5/OS ジョブを開始するごとに、完全修飾のジョブ名とプロセス ID を添付したメッセージを標準エラーに表示します。

-l (logcmds)
コマンドのロギングを使用可能にします。コマンドが実行される前に各コマンドをジョブ・ログのメッセージに書き込みます。

-m (monitor)
ジョブの完了後メッセージを表示します。対話式オプションが設定されると、**qsh** はこのオプションを暗黙でオンにします。

-n (noexec)
対話式オプションが設定されていない場合に、コマンドを読み取りますが、実行はしません。 これは、シェル・スクリプトの構文を検査するときに使用するオプションです。

-t (trace)
内部トレースを使用可能にします。 **qsh** は、**TRACEFILE** 変数で指定されたファイルか、ユーザーのホーム・ディレクトリー内の qsh_trace ファイルに、内部情報をトレースします。

-u (nounset)
設定されていない変数を展開しようとしたときに標準エラーにメッセージを書き込みます。対話式オプションが設定されていない場合は即時に終了します。

-v (verbose)
読み取った入力をそのまま標準エラーに書き込みます。

-x (xtrace)
PS4 変数の展開後に実行前の各コマンドを標準エラーに書き込みます。

オペランド

各 *argument* は、順番に定位置パラメーターに代入されます。

終了状況

- 0 正常終了。

例

1. すべての変数とその値をリスト表示する場合:
   ```
   set - オプションおよび定位置パラメーターを設定または設定解除する
   ```
2. すべてのオプションの設定内容をリスト表示する場合:
   ```
   set -o
   ```
3. 定位置パラメーター $1、$2、$3 を設定する場合:
   ```
   set alpha beta gamma
   ```
4. allexport オプションと notify オプションを設定する場合:
   ```
   set -o allexport -o notify
   ```
5. verbose オプションと xtrace オプションを設定する場合:
   ```
   set -xv
   ```
6. xtrace オプションの設定を解除する場合:
   ```
   set +x
   ```
7. notify オプションの設定を解除する場合:
   ```
   set +o notify
   ```
8. すべての定位置パラメーターの設定を解除する場合:
   ```
   set --
   ```

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

164 ページの『export - 変数に対してエクスポート属性を設定する』

165 ページの『local - 関数内でローカル変数を割り当てる』
167 ページの『readonly - 変数に対して読み取り専用属性を設定する』
53 ページの『qsh - Qshell コマンド言語インタープリター』
『shift - 定位置パラメーターをシフトする』
『typeset - 変数を宣言し、属性設定をする』
171 ページの『unset - 変数および関数の値を設定解除する』

shift - 定位置パラメーターをシフトする

構文

shift [*n*]

説明

shift を使用すると、定位置パラメーターを *n* 個分だけ左にシフトできます。定位置パラメーター 1 には定位置パラメーター (1+*n*) の値が代入され、定位置パラメーター 2 には定位置パラメーター (2+*n*) の値が代入されます (以下同様です)。 特殊パラメーター # は、新しい定位置パラメーター数に更新されます。

オプション

なし。

オペランド

n の値は、特殊パラメーター # 以下の符号なし整数でなければなりません。 *n* を指定しなかった場合のデフォルト値は 1 です。*n* が 0 の場合は、定位置パラメーターは変化しません。

終了状況

- 0 正常終了。
- >0 *n* が無効。

例

定位置パラメーターを 2 個分シフトする場合: **shift** 2

関連タスク:

168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』

typeset - 変数を宣言し、属性設定をする

構文

typeset [-Eilrux] *name* **[**=*value***]** ...

typeset [+Eilrux] *name* **[**=*value***]** ...

typeset -fF [*name* ... **]**

typeset -p *name* ...

typeset

説明

typeset ユーティリティーは、変数の宣言、変数への値の割り当て、変数の属性の設定、さらにシェル関数の定義の表示を行います。これは、declare ユーティリティーと同じ働きをします。

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

164 ページの『export - 変数に対してエクスポート属性を設定する』

165 ページの『local - 関数内でローカル変数を割り当てる』

167 ページの『readonly - 変数に対して読み取り専用属性を設定する』

168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』

『unset - 変数および関数の値を設定解除する』

unset - 変数および関数の値を設定解除する

構文

unset [**-fv**] [*name* ...]

説明

unset を使うと、name に指定した各変数または関数の設定を解除できます。 オプションの指定がなかった場合は、*name* は変数を表します。 読み取り専用属性を持つ変数は設定解除できません。

オプション

-f *name* は関数を表します。

-v *name* は変数を表します。

オペランド

各 *name* は変数または関数です。

終了状況

- 0 正常終了。
- >0 見つからない *name* が 1 つ以上あった。値は、見つからなかった *names* の数を示します。

例

1. 変数 alpha を設定解除する場合: unset alpha
2. 関数 foo を設定解除する場合: unset -f foo

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

164 ページの『export - 変数に対してエクスポート属性を設定する』

165 ページの『local - 関数内でローカル変数を割り当てる』

167 ページの『readonly - 変数に対して読み取り専用属性を設定する』

168 ページの『set - オプションおよび定位置パラメーターを設定または設定解除する』

170 ページの『typeset - 変数を宣言し、属性設定をする』

スクリプト作成用のユーティリティー

スクリプト作成用のユーティリティーを表示するには、このリンクを選択します。

break - for、while、または until ループを終了する

構文

break[*n*]

説明

break を使うと、一番内側の **for**、**while**、または **until** ループを終了することも、または一番内側から *n* 番目のループを終了することもできます。処理はループの直後のコマンドから再開されます。

オプション

なし。

オペランド

n の値は、1 以上でなければなりません。

終了状況

- 0 正常終了。

関連タスク:

『continue - for、while、または until ループを継続する』

コロン (:) - ヌル・ユーティリティー

構文

: [*argument* ...]

説明

コマンドを指定しなければならないがそのコマンドに何の処理も実行させたくない、という場合に、**コロン** を使用できます。 たとえば、**if** コマンドの **then** 条件の中で使用します。

オプション

なし。

オペランド

各 *argument* が展開されます。

終了状況

- 0 正常終了。

continue - for、while、または until ループを継続する

構文

continue [*n*]

説明

continue を使うと、一番内側の **for**、**while**、または **until** ループの先頭に進むか、一番内側から *n* 番目のループの先頭に進むことができます。処理は、ループの先頭にある最初のコマンドから再開されます。

オプション

なし。

オペランド

n の値は、1 以上でなければなりません。

終了状況

- 0 正常終了。

関連タスク:

172 ページの『break - for、while、または until ループを終了する』

false - 偽の値を戻す

構文

false

説明

false は、0 以外の終了コードを伴って戻ります。

オプション

なし。

オペランド

なし。

終了状況

- >0 不成功の場合。

関連タスク:

177 ページの『true - 真の値を戻す』

getopts - ユーティリティー・オプションを解析する

構文

getopts *optstring varname*

説明

getopts を使うと、有効なオプションの定位置パラメーターを検査できます。 オプション引数は、負符号 **(-)** で始まります。 負符号で始まらない引数が最初に出てきたときや引数 **--** が出てきたときはオプション引数の終わりです。

getopts は、呼び出されるたびに、次のオプション文字を検出して *varname* に入れます。 **qsh** は、処理対象とする次のパラメーターの索引を、変数 **OPTIND** に格納します。 オプションに引数が必要なときは、**qsh** はその引数を変数 **OPTARG** に格納します。

オプション

なし。

オペランド

getopts で認識されるオプション文字は、 *optstring* の中に指定します。文字の後ろにコロン **(:)** を付けると、そのオプションは引数を持っているものと見なされます。オプション文字と引数の間を <space> で区切ることもできます。

getopts を呼び出すたびに、*varname* はオプション文字を使用して更新されます。

終了状況

- 0 正常終了。
- >0 不成功の場合。

let - 算術式を評価する

構文

let *arg* ...

説明

let を使うと、各 *arg* をそれぞれ 1 つの算術式 として評価できます。算術演算子の多くは **qsh** にとって特殊な意味を持つので、場合によっては各 *arg* を引用符で囲む必要があります。

オペランド

各 *arg* は、それぞれ 1 つの算術式として評価されます。

終了状況

- 0 最後の式の値が 0 以外の値。
- 1 最後の式の値が 0。

例

変数 x に 1 を加算する場合 :

```
let x=x+1
```

関連タスク:

163 ページの『declare - 変数を宣言し、属性設定をする』

return - 関数から戻る

構文

return [*n*]

説明

return を使うと、関数またはドット・スクリプトから呼び出し元のシェル・スクリプトに戻ることができます。 関数またはドット・スクリプトの外部で **return** を呼び出した場合は、**exit** と同じ結果になります。

オプション

なし。

オペランド

n の値は、0 - 255 の整数です。

終了状況

n を指定した場合は、*n*。 n を指定しなかった場合は、直前のコマンドの終了状況。

関連タスク:

51 ページの『exit - シェルを終了する』

test - 式を評価する

構文

test *expression*

[*expression*]

説明

test ユーティリティーは、ファイルのタイプのチェック、ファイルの許可のチェック、2 つのストリングの比較、または 2 つの算術式の比較を行うことができます。

test ユーティリティーは、以下のプライマリーを使用してファイルの状態を検査します。

-b *file* *file* が存在し、それがブロック特殊ファイルである場合に真になります。

-c *file* *file* が存在し、それが文字特殊ファイルである場合に真になります。

-d *file* *file* が存在し、それがディレクトリーである場合に真になります。

-e *file* *file* が存在すれば、そのタイプに関係なく真になります。

-f *file* *file* が存在し、それが正規ファイルである場合に真になります。

-g *file* *file* が存在し、そのグループ ID 設定フラグが設定されている場合に真になります。

-G *file* *file* が存在し、それが有効なグループ ID の所有である場合に真になります。

-h *file* *file* が存在し、それがシンボリック・リンクである場合に真になります。

-k *file* *file* が存在し、その制限付き削除フラグが設定されている場合に真になります。

-L *file* *file* が存在し、それがシンボリック・リンクである場合に真になります。

-N *file* *file* が存在し、それがネイティブ・オブジェクトである場合に真になります。

-O *file* *file* が存在し、それが有効なユーザー ID の所有である場合に真になります。

-p *file* *file* が存在し、それがパイプである場合に真になります。

-r *file* *file* が存在し、それが読み取り可能である場合に真になります。

-s *file* *file* が存在し、そのサイズがゼロより大きい場合に真になります。

-S *file* *file* が存在し、それがソケットである場合に真になります。

-u *file* *file* が存在し、そのユーザー ID 設定フラグが設定されている場合に真になります。

-w *file* *file* が存在し、それが書き込み可能である場合に真になります。

-x *file* *file* が存在し、それが実行可能である場合に真になります。これは、実行ビットがオンであることを示すだけです。 *file* がディレクトリーである場合は、そのディレクトリーが検索されます。

*file1***-ef***file2*
file1 と *file2* が、名前は異なるものの、同じファイル (デバイス番号と inode 番号が同じ) である場合に真になります。

*file1***-nt***file2*
file1 が *file2* より新しい、または *file2* が存在しない場合に真になります。

*file1***-ot***file2*
file1 が *file2* より古い、または *file2* が存在しない場合に真になります。

test ユーティリティーは、以下のプライマリーを使用して状況検査の状態を検査します。

-o *optname*
シェル・オプション *optname* が使用可能である場合に真になります。

-t *fd* ファイル記述子 *fd* がオープン状態で、端末に関連付けられている場合に真になります。

test ユーティリティーは、以下のプライマリーを使用してストリング比較の状態を検査します。

-n *string*
string の長さがゼロではない場合に真になります。

-z *string*
string の長さがゼロである場合に真になります。

string *string* が空ストリングではない場合に真になります。

string1 **=** *string2*
2 つのストリングが同じである場合に真になります。

*string1***==***string2*
2 つのストリングが同じである場合に真になります。

*string1***!=***string2*
2 つのストリングが同じではない場合に真になります。

string1 **<** *string2*
現行のロケールの照合順序で、*string1* が *string2* の前にソートされる場合に真になります。

string1 **>** *string2*
現行のロケールの照合順序で、*string1* が *string2* の後にソートされる場合に真になります。

test ユーティリティーは、以下のプライマリーを使用して算術式比較の状態を検査します。

*exp1***-eq***exp2*
算術式が等しい場合に真になります。

*exp1***-ne***exp2*
算術式が等しくない場合に真になります。

*exp1***-gt***exp2*
最初の算術式が 2 番目の算術式より大きい場合に真になります。

*exp1***-ge***exp2*
最初の算術式が 2 番目の算術式より大きいまたは等しい場合に真になります。

*exp1***-lt***exp2*
最初の算術式が 2 番目の算術式より小さい場合に真になります。

*exp1***-le***exp2*
最初の算術式が 2 番目の算術式より小さいまたは等しい場合に真になります。

次の演算子を使って上記のプライマリーを組み合わせれば、複合式を作成できます。

- **!** *expr expr* が偽の場合に真になります。
- *expr1* **-a** *expr2* 両方の式が真の場合に真になります。
- *expr1* **&** *expr2* 両方の式が真の場合に真になります。
- *expr1* **&&** *expr2* 両方の式が真の場合に真になります。
- *expr1* **-o** *expr2* どちらかの式が真である場合に真になります。
- *expr*1 | *expr2* どちらかの式が真である場合に真になります。
- *expr*1 || *expr2* どちらかの式が真である場合に真になります。
- **(***expr***)** 括弧はグループ化のために使用されます。

-a 演算子、**&** 演算子、および **&&** 演算子は、**-o** 演算子、| 演算子、および || 演算子より優先されます。

オプション

上記を参照。

オペランド

すべての演算子およびフラグは、それぞれ単独の引数として扱われます。

終了状況

- 0 *expression* が真。
- 1 *expression* が偽。
- >1 エラー発生。

例

1. /home がディレクトリーかどうかを確認する場合:
   ```
   test -d /home
   ```
2. ある整数が他の整数以下かどうかを確認する場合:
   ```
   test "$index" -le "$count"
   ```
3. 2 つの文字列が等しいかどうかを確認する場合:
   ```
   test "$REPLY" = "Yes"
   ```

true - 真の値を戻す

構文

true

説明

true は、終了コード 0 を戻します。

オプション

なし。

オペランド

なし。

終了状況

0

関連タスク:
173 ページの『false - 偽の値を戻す』

その他のユーティリティー

その他のユーティリティーを表示します。

clrtmp - /tmp ディレクトリーをクリアする

構文

clrtmp [-c]

説明

clrtmp ユーティリティーは、/tmp ディレクトリーからすべてのオブジェクトを除去することにより、/tmp ディレクトリーを消去します。他のシステムでは、/tmp ディレクトリーはシステムが起動する度に消去されますが、 i5/OS では、/tmp ディレクトリーはシステムが起動する度に消去されるわけではありません。i5/OS の起動時に /tmp ディレクトリーを消去するには、QSTRUPPGM システム値により指定される始動プログラムに、**clrtmp** ユーティリティーへの呼び出しを含めることができます。

/tmp ディレクトリーからオブジェクトを除去するためには、**clrtmp** の呼び出し元は、/tmp に含まれている各サブディレクトリーに対して *WX 権限を持っていなければならず、さらに各オブジェクトに対して *OBJEXIST 権限を持っている必要があります。呼び出し元が必要な権限を持っていない場合、それらのオブジェクトは /tmp ディレクトリーから除去されません。

システムが稼働中に **clrtmp** が呼び出された場合、予測できない結果になることがあります。たとえば、別のプログラムが /tmp ディレクトリーにあるファイルに書き込みをしている場合、そのファイルのパスは除去され、そのファイルを使用することができなくなります。

注:	これは i5/OS に固有のユーティリティーです。

オプション

-c /tmp が存在していない場合は、/tmp を作成します。

終了状況

- 0 正常終了。
- >0 エラーが発生したか、または少なくとも 1 つのオブジェクトが /tmp ディレクトリーから除去できなかった。

dataq - i5/OS データ待ち行列からメッセージを送受信する

構文

dataq -c [-l] *queue*

dataq -r [-lp] [-n *number*] **[-t** *seconds*] *queue*

dataq -w [-l] [-n *number*] *queue* **[** *data ...* **]**

説明

dataq ユーティリティーは、データ待ち行列からのメッセージの消去、データ待ち行列からのメッセージの読み取り、またはデータ待ち行列へのメッセージの書き込みを行います。

最初の構文形式では、**dataq** はすべてのメッセージを *queue* から消去します。

2 番目の構文形式では、**dataq** はメッセージを *queue* から読み取り、それらを標準出力に書き込みます。デフォルトでは、*queue* から読み取るメッセージは 1 つです。 *queue* からのメッセージが入手できない場合、**dataq** はメッセージを待ちます。

3 番目の構文形式では、**dataq** はメッセージを *queue* に書き込みます。 *data* が指定された場合、それは 1 つのメッセージとして *queue* に書き込まれます。それ以外の場合は、標準入力から読み取られた行がそれぞれ、1 つのメッセージとして *queue* に書き込まれます。

オプション

-c すべてのメッセージを *queue* から消去します。

-l 相対パス名が指定されている場合、ライブラリー・リストを使用して *queue* を検索します。

-n *number*
-r オプションが指定されている場合、*number* で示される数のメッセージを *queue* から読み取ります。 **-w** オプションが指定されている場合、*number* で示される数のメッセージを *queue* に書き込みます。

-p peek モード。メッセージを読み取る時、そのメッセージを *queue* に残します。

-r メッセージを *queue* から読み取ります。

-t *seconds*
メッセージを読み取るとき、*seconds* 秒間待機してもメッセージを受け取らなかった場合に終了します。

-w *queue* からのメッセージを書き込みます。

オペランド

queue はデータ待ち行列のパス名です。データ待ち行列は、QSYS.LIB ファイル・システムにしか存在できません。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

180 ページの『datarea - i5/OS(TM) データ域の読み書きをする』

128 ページの『Rfile - レコード・ファイルの読み書きをする』

datarea - i5/OS(TM) データ域の読み書きをする

構文

datarea -r [-l] [-s *substring*] *data-area*

datarea -w [-l] [-s *substring*] *data-area* **[** *data ...* **]**

説明

datarea ユーティリティーは、データ域の読み書きをします。

最初の構文形式では、**datarea** は、*data-area* の内容を読み取り、それを標準出力に書き込みます。デフォルトでは、データ域全体を読み取ります。

2 番目の構文形式では、**datarea** は、*data-area* に書き込みをします。 *data* が指定された場合、それが *data-area* に書き込まれます。指定されなかった場合は、標準入力から 1 行読み取り、*data-area* に書き込みます。

オプション

-l 相対パス名が指定されている時、ライブラリー・リストを使用して *data-area* を検索します。

-r *data-area* から読み取りをします。

-s *substring*
文字タイプ・データ域の作業のため、*substring* で指定される文字位置を読み書きします。 *substring* で、番号、ダッシュ (-)、および 2 番目の番号で構成される番号の範囲として指定すると、1 番目の番号から 2 番目の番号まで (これらを含む) の各文字位置を選択できます。最初の番号を省略すると、1 から 2 番目の間の番号の文字位置が選択されます。 2 番目の番号を省略すると、1 番目の番号からデータ域の最後までの文字位置が選択されます。

-w *data-area* に書き込みをします。

オペランド

data-area はデータ域のパス名です。データ域は、QSYS.LIB ファイル・システムにのみ存在します。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連概念:

179 ページの『dataq - i5/OS データ待ち行列からメッセージを送受信する』

関連タスク:

128 ページの『Rfile - レコード・ファイルの読み書きをする』

date - 日付と時刻を書き込む

構文

date [-u] [+*format*]

説明

date ユーティリティーは、標準出力に日付と時刻を書き込みます。デフォルトでは、現在の日付と時刻が書き込まれます。

オプション

-u 世界標準時 (UTC) で時刻が提供されます。正確な時刻が返されるようにするには、QUTCOFFSET システム値を **date** に正しく設定する必要があります。

オペランド

+*format* オペランドで **date** コマンドからの出力の形式が指定されます。各フィールド記述子は、それぞれの対応する値に置き換えられて標準出力に出力されます。その他すべての文字は、変更されずにコピーされて出力されます。 出力は、常に改行文字で終了します。

以下のフィールド記述子を使用することができます。

%a ロケール指定に従い、曜日名を省略形で挿入します。

%A ロケール指定に従い、曜日名を完全な形式で挿入します。

%b ロケール指定に従い、月名を省略形で挿入します。

%B ロケール指定に従い、月名を完全な形式で挿入します。

%c ロケール指定に従い、日付と時刻を挿入します。

%d 日付 (01 - 31) を挿入します。

%H 時刻 (24 時間制) を 10 進数 (00 - 23) で挿入します。

%I 時刻 (12 時間制) を 10 進数 (01 - 12) で挿入します。

%j 年間通算日 (001 - 366) を挿入します。

%m 月 (01 - 12) を挿入します。

%M 分 (00 - 59) を挿入します。

%p ロケール指定に従い、AM または PM に相当する値を挿入します。

%S 秒 (00 - 61) を挿入します。

%U 日曜日を週の起点として、年間の週番号 (00 - 53) を挿入します。

%w 日曜日を 0、つまり週の起点として、曜日 (0 - 6) を挿入します。

%W 月曜日を週の起点として、年間の週番号 (00 - 53) を挿入します。

%x ロケール指定に従い、データ表示を挿入します。

%X ロケール指定に従い、時刻表示を挿入します。

%y 西暦年の最後の 2 桁 (00 - 99) を挿入します。

%Y 年を挿入します。

%Z 時間帯の名称を挿入します。時間帯が指定されていない場合には、文字は挿入されません。

%% % を挿入します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. 完全な曜日名、完全な月名、日、4 桁の年を出力する場合 :

```
date +@(#) 89 1.41@(#), 0 %d%, %Y
Friday, August 14, 1998
```

2. 日、省略形式の月名、2 桁の年を出力する場合 :

```
date +%d%.%b%.%y
14.Aug.98
```

3. 数字で表した月、日、2 桁の年を出力する場合 :

```
date +%m%/%d%/%y
08/14/98
```

expr - 引数を式として評価する

構文

expr *operand ...*

説明

expr ユーティリティーは、*operands* によって形式化された式を評価し、結果を標準出力に書き出します。

オペランド

評価される式のフォーマットは次のとおりです。 *expr*、*expr1*、および *expr2* は、10 進整数または文字列です。

注:	両方の引数が整数の場合、6 つの関係式では 10 進整数の比較の結果が戻されます。それ以外の場合は、文字列の比較の結果が戻されます。個々の比較の結果は、指定された関係が真ならば 1 に、偽ならば 0 になります。

式	説明
expr1 | *expr2*	*expr1* の評価が 0 でもヌルでもない場合には、その評価が戻されます。それ以外の場合は *expr2* の評価が戻されます。
expr1 & *expr2*	どちらの式の評価もヌルまたは 0 でない場合は、*expr1* の評価が戻されます。それ以外の場合は 0 が戻されます。
expr1= *expr2*	等しい
expr1 > *expr2*	より大きい
expr1 >= *expr2*	より大きいまたは等しい
expr1 < *expr2*	より小さい
expr1 <= *expr2*	より小さいまたは等しい
expr1 != *expr2*	等しくない
expr1 + *expr2*	10 進整数値の加算
expr1 - *expr2*	10 進整数値の減算
expr1 * *expr2*	10 進整数値の乗算
expr1 / *expr2*	10 進整数値の除算

式	説明
expr1 % *expr2*	整数除算の余り
expr1 : *expr2*	式の突き合わせ
(*expr*)	グループ化記号

終了状況

- 0 式の評価がヌルでも 0 でもない。
- 1 式の評価がヌルまたは 0。
- 2 式が無効。
- >2 エラー発生。

例

1. 演算式を評価する場合 :

```
expr 10+10*10/10-10
```

2. 真条件または偽条件を評価する場合 :

```
expr 10 = 10
```

hostname - 現在のホスト・システムの名前を表示する

構文

hostname [-is]

説明

hostname ユーティリティーは、現在のホスト・システムの名前を標準出力に書き込みます。

オプション

-i ホスト・システムの IP アドレスも表示します。

-s ホスト・システムの短縮名をドメイン情報なしで表示します。

終了状況

- 0 正常終了。
- >0 エラー発生。

id - ユーザー ID を戻す

構文

id [*user*]

id -G [-n] [*user*]

id -g [-nr] [*user*]

id -p [*user*]

id -u [-nr] [*user*]

説明

id ユーティリティーは、呼び出しプロセスのユーザー名、グループ名、および数字値 ID を標準出力に表示します。 実 ID と実効 ID が異なる場合は、両方とも表示されます。同じ場合は、実 ID のみが表示されます。

user (ログイン名またはユーザー ID) が指定されている場合は、そのユーザーのユーザー ID とグループ ID が表示されます。この場合、実 ID と実効 ID は同じものと見なされます。

オプション

-G 異なるグループ ID (実効 ID、実 ID、および補助 ID) を、スペースで区切った数字で、特別の順序によらずに表示します。

-g 実効グループ ID を数字で表示します。

-n **-G**、**-g**、および **-u** オプションの場合に、ユーザー ID またはグループ ID を、数字でなく名前で表示します。 名前への対応付けができない ID 番号の場合は、その数字がそのまま表示されます。

-p 人間が理解できる形式の出力を作成します。ユーザー ID は、前にキーワード uid を付けた名前で表示されます。実効ユーザー ID が実ユーザー ID と異なる場合は、実ユーザー ID が、前にキーワード euid を付けた名前で表示されます。実効グループ ID が実グループ ID と異なる場合は、実グループ ID が、前にキーワード rgid を付けた名前で表示されます。ユーザーが属しているグループのリストは、前にキーワード groups を付け、名前で表示されます。項目ごとに、それぞれ 1 行に表示されます。

-r **-g** および **-u** オプションの場合は、実効 ID ではなく、実 ID が表示されます。

-u 実効ユーザー ID を数字で表示します。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

ユーザー SAM に属するすべてのユーザー ID およびグループ ID を表示する場合 :

```
id -p SAM
uid      SAM
groups     500, 1
```

関連タスク:

193 ページの『logname - ユーザーのログイン名を表示する』

ipcrm - プロセス間通信 ID を削除する

構文

ipcrm [**-m** *shmid*] [**-M** *shmkey*] [**-q** *msgid*] [**-Q** *msgkey*] [**-s** *semid*] [**-S** *semkey*]

説明

ipcrm ユーティリティーは、呼び出し元に IPC 項目への必要な権限がある場合、プロセス間通信 (IPC) 項目を削除します。呼び出し元は、キーまたは ID のいずれかにより項目を指定することができます。呼び出し元は一度に複数の項目を削除できます。

オプション

-M *shmkey*
指定したキーを持つ共用メモリー・セグメントを削除します。

-m *shmid*
指定した ID を持つ共用メモリー・セグメントを削除します。

-Q *msgkey*
指定したキーを持つメッセージ待ち行列を削除します。

-q *msgid*
指定した ID を持つメッセージ待ち行列を削除します。

-S *semKey*
指定したキーを持つセマフォー・セットを削除します。

-s *semid*
指定した ID を持つセマフォー・セットを削除します。

オペランド

オペランドはありません。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

- キーが 1283 のセマフォーを削除し、ID が 10 のメッセージ待ち行列を削除する場合:

```
ipcrm -S 1283 -q 10
```

関連タスク:
『ipcs - プロセス間通信状況を報告する』

ipcs - プロセス間通信状況を報告する

構文

ipcs [-ETabcjmnopqstu]

説明

ipcs ユーティリティーは、システム上にすでにあるプロセス間通信 (IPC) についての情報を報告します。**ipcs** ユーティリティーは、共通権限が *EXCLUDE に設定された状態で出荷されます。ユーザーは、*SERVICE 特殊権限がないと **ipcs** を実行できません。

ipcs は、指定された IPC メカニズムと一致するすべての項目情報を自動的に報告します。追加情報は指定されたオプションに基づいて報告されます。

IPC メカニズムが指定されていない場合は、5 つのすべてのメカニズムが報告されます。 IPC メカニズムは、**-m** オプションで共用メモリーを、**-n** オプションで名前付きセマフォーを、**-s** オプションでセマフォー・セットを、**-q** オプションでメッセージ待ち行列を、または **-u** オプションで名前なしセマフォーを指定します。

すべての共用メモリー、セマフォー・セット、およびメッセージ待ち行列項目に関して以下の情報が報告されます。

- メカニズムのタイプ (T 欄)。
- 項目 ID (10 進数形式) (ID 欄)。
- 項目キー (16 進数形式) (KEY 欄)。
- 項目のアクセス・モードおよびフラグ (MODE 欄)。
- 項目所有者のユーザー・プロファイル (OWNER 欄)。
- 項目所有者のグループ・プロファイル (GROUP 欄)。

すべての名前付きセマフォー項目について以下の情報が報告されます。
- メカニズムのタイプ (T 欄)。
- セマフォーの名称 (TITLE 欄)。
- 項目のアクセス・モードおよびフラグ (MODE 欄)。

すべての名前なしセマフォー項目について以下の情報が報告されます。
- メカニズムのタイプ (T 欄)。
- セマフォーの名称 (TITLE 欄)。

重要: ipcs を実行すると、システムのリソースをロックするので、他の IPC 操作のパフォーマンスに影響を与える可能性があります。

オプション

報告する IPC メカニズムを選択するために以下のオプションが使用されます。

-m システム上の共用メモリー項目を表示します。

-n システム上の名前付きセマフォー項目を表示します。

-q システム上のメッセージ待ち行列項目を表示します。

-s システム上のセマフォー・セット項目を表示します。

-u システム上の名前なしセマフォー項目を表示します。

以下のオプションで、IPC メカニズムの報告に含める追加情報を選択します。

-a **-b**、**-c**、**-o**、**-p**、および **-t** の各オプションが指定された場合のように、すべての情報を報告します。

-b 最大許容サイズを表示します。メッセージ待ち行列が指定されている場合、報告書には QBYTES 欄が組み込まれます。共用メモリーが指定されている場合、報告書には SEGSZ 欄が組み込まれます。セマフォー・セットが指定されている場合、報告書には NSEMS 欄が組み込まれます。名前付きセマフォーまたは名前なしセマフォーが指定されている場合、報告書には VALUE 欄および NWAITERS 欄が組み込まれます。

-c 項目の作成者のユーザー・プロファイルおよびグループ・プロファイルを表示します。すべてのメカニズムに関して、報告書には CREATOR 欄および CGROUP 欄が組み込まれます。

-E 拡張情報を表示します。メッセージ待ち行列が指定されている場合、報告書には WPID、WTID、MSGTYPE、および SIZE の各欄が組み込まれます。共用メモリーが指定されている場合、報告書には APID 欄、NUMATT 欄、および PAGESZ 欄が組み込まれます。セマフォー・セットが指定されている場合、報告書には SEMNUM、SEMVAL、LOPID、WAITZ、 WAITP、および WAITVAL の各欄が組み込まれます。名前付きセマフォーが指定されている場合、報告書には

NAME、LPOST、LWAIT、 WAITER、JOB、および THREAD の各欄が組み込まれます。名前なしセマフォーが指定されている場合、報告書には LPOST、LWAIT、WAITER、JOB、および THREAD の各欄が組み込まれます。

他のシステムでは、これほど詳細なレベルの設定ができないため、このオプションは **-a** オプションと分離されています。このオプションを指定すると、項目ごとに最低 1 行が追加されます。

-j **-E** オプションも指定される場合、プロセス ID の代わりに修飾ジョブ名を表示します。メッセージ待ち行列が指定されている場合、報告書には WPID ではなく WJOBID 欄が組み込まれます。共用メモリーが指定されている場合、報告書には APID ではなく AJOBID 欄が組み込まれます。セマフォー・セットが指定されている場合、報告書には LOPID ではなく LOJOBID 欄、WAITZ ではなく WAITZJID 欄、および WAITP ではなく WAITPJID 欄 が組み込まれます。

-o 未解決の使用法についての情報を表示します。メッセージ待ち行列が指定されている場合、報告書には CBYTES 欄および QNUM 欄が組み込まれます。共用メモリーが指定されている場合、報告書には NATTCH 欄が組み込まれます。

-p プロセス ID 情報を表示します。メッセージ待ち行列が指定されている場合、報告書には LSPID 欄および LRPID 欄が組み込まれます。共用メモリーが指定されている場合、報告書には CPID 欄および LPID 欄が組み込まれます。

-t 時刻情報を表示します。メッセージ待ち行列が指定されている場合、報告書には CTIME 欄、RTIME 欄、および STIME 欄が組み込まれます。共用メモリーが指定されている場合、報告書には CTIME 欄、ATIME 欄、および DTIME 欄が組み込まれます。セマフォー・セットが指定されている場合、報告書には CTIME 欄および OTIME 欄が組み込まれます。

-T スレッド情報を表示します。メッセージ待ち行列が指定されている場合、報告書には LSTID 欄および LRTID 欄が組み込まれます。共用メモリーが指定されている場合、報告書には CTID 欄および LTID 欄が組み込まれます。セマフォー・セットが指定されている場合や、**-E** オプションが指定されている場合、報告書には LOTID 欄、WAITZTID 欄、および WAITPTID 欄が組み込まれます。

オペランド

オペランドはありません。

補足説明

下記のリストは、出力に報告することのできるすべての欄の説明です。欄の名前の後、その欄を表示するためのオプションを示します。「デフォルト」という値は、どのオプションが指定されていてもその欄は常に表示されるという意味です。

AJOBID (-Ej)
共用メモリー・セグメントに付加されているジョブの修飾ジョブ名。

ATIME (-t, -a)
ジョブが最後に共用メモリー・セグメントに付加した時刻。

APID (-E)
共用メモリー・セグメントに付加されているジョブのプロセス ID。

CBYTES (-o, -a)
現在メッセージ待ち行列上にあるメッセージの合計バイト数。

CGROUP (-c, -a)
項目の作成者のグループ・プロファイル。

CPID (-p, -a)
共用メモリー・セグメントを作成したジョブのプロセス ID。

CTID (-T)
共用メモリー・セグメントを作成したスレッドのスレッド ID。

CREATOR (-c, -a)
項目の作成者のユーザー・プロファイル。

CTIME (-t, -a)
項目が作成された時刻、または所有者または許可 (あるいはその両方) が変更された時刻のうち、最後のもの。

DTIME (-t, -a)
ジョブが最後に共用メモリー・セグメントから切り離された時刻。

GROUP (デフォルト)
項目所有者のグループ・プロファイル。

ID (デフォルト)
項目 ID (10 進数形式)。

JOB (-E)
名前付きセマフォーまたは名前なしセマフォーを待機しているジョブの完全修飾ジョブ名。

KEY (デフォルト)
項目キー (16 進数形式)。

LOJOBID (-Ej)
semop() を使用して、セマフォーの値を最後に変更したジョブの修飾ジョブ名。

LOPID (-E)
semop() を使用して、セマフォーの値を最後に変更したジョブのプロセス ID。

LOTID (-TE)
semop() を使用して、セマフォーの値を最後に変更したスレッドのスレッド ID。

LPID (-p, -a)
共用メモリー・セグメントに接続したジョブまたはそこから切り離しをしたジョブ、またはセマフォーの値を変更したジョブのうち、最後のもののプロセス ID。

LPOST (-E)
名前付きセマフォーまたは名前なしセマフォーを最後に追加したスレッドの完全修飾ジョブ名、およびスレッド ID。

LRPID (-p, -a)
msgrcv() を使用して、メッセージ・キューからメッセージを最後に受信したジョブのプロセス ID。

LRTID (-T)
msgrcv() を使用して、メッセージ・キューからメッセージを最後に受信したスレッドのスレッド ID。

LSPID (-p, -a)
msgsnd() を使用して、メッセージ・キューへメッセージを最後に送信したジョブのプロセス ID。

LSTID (-T)
msgsnd() を使用して、メッセージ・キューへメッセージを最後に送信したスレッドのスレッド ID。

LTID (-T)
共用メモリー・セグメントに接続、またはそこから切り離しをした最後のスレッドのスレッド ID。

LWAIT (-E)
名前付きセマフォーまたは名前なしセマフォーを最後に待機したスレッドの完全修飾ジョブ名、およびスレッド ID。

MODE (デフォルト)
項目の状態および許可に関する情報を与える 11 文字のフィールド。

最初の文字は次のいずれかです。

D その項目は損傷を受けているため、その項目を操作することはできません。内部エラーが発生した場合に限って、項目にはマークが付けられます。

T その項目は共用メモリー・セグメントであり、セグメントはテラスペース・ストレージを使用します。

Y その項目は共用メモリー・セグメントであり、セグメントはテラスペース・ストレージを使用します。また、その項目は損傷を受けています。

-

2 番目の文字は次のいずれかです。

R その項目はメッセージ待ち行列であり、スレッドが msgrcv() への呼び出しを待っている。

S その項目はメッセージ待ち行列であり、スレッドが msgsnd() への呼び出しを待っている。

D その項目は共用メモリー・セグメントであり、共用メモリー・セグメントは、すべてのジョブが共用メモリーから切り離されるとき、除去されるようにマークされます。

- 上記のいずれも適用されません。

次の 9 文字は、3 つずつの許可が 3 セットあるものとして解釈されます。最初のセットは所有者の許可、2 番目のセットはグループの許可、3 番目のセットはその他の許可を示します。各セット内で、先頭文字は読み取り許可を表し、2 番目の文字は書き込み許可を表し、最後の文字は現在未使用です。

許可の意味は次のとおりです。

r 読み取り許可が認可されている。

w 書き込み許可が認可されている。

- 当該の許可が認可されていない。

MSGTYPE (-E)
現在メッセージ待ち行列上にあるメッセージのタイプ。

NAME (-E)
名前付きセマフォーのパス名。

NATTCH (-o, -a)
共用メモリー・セグメントへの現在の接続数。

NUMATT (-E)
ジョブが共用メモリー・セグメントに接続された回数。

NSEMS (-b, -a)
セマフォー・セット中のセマフォーの数。

NWAITERS (-b, -a)
名前付きセマフォーまたは名前なしセマフォーを待機しているスレッドの数。

OTIME (-t, -a)
セマフォー・セットを使用して semop() が最後に呼び出された時刻。

OWNER (デフォルト)
項目所有者のユーザー・プロファイル。

PAGESZ (-E)
共用メモリー・セグメントをバッキングするストレージのページ・サイズ (バイト単位)。

QBYTES (-b, -a)
メッセージ待ち行列上に許可されている最大のバイト数。

QNUM (-o, -a)
現在メッセージ待ち行列上にあるメッセージの数。

RTIME (-t, -a)
メッセージ待ち行列を使用して msgrcv() が最後に呼び出された時刻。

SEGSZ (-b, -a)
共用メモリー・セグメントのサイズ。

SEMNUM (-E)
セマフォー・セット中のセマフォー番号。

SEMVAL (-E)
セマフォーの値。

SIZE (-E)
メッセージ待ち行列上にあるメッセージのサイズ。

STIME (-t, -a)
メッセージ待ち行列を使用して msgsnd() が最後に呼び出された時刻。

T (デフォルト)
項目タイプ。その値は、共用メモリー・セグメントの場合は M、名前付きセマフォーの場合は N、メッセージ待ち行列の場合は Q、セマフォー・セットの場合は S、または名前なしセマフォーの場合は U となります。

THREAD (-E)
名前付きセマフォーまたは名前なしセマフォーを待機しているスレッドのスレッド ID。

TITLE (デフォルト)
名前付きセマフォーまたは名前なしセマフォーの名称。

VALUE (-b, -a)
名前付きセマフォーまたは名前なしセマフォーの現行値。

WAITER (-E)
名前付きセマフォーまたは名前なしセマフォーを待機しているスレッドのインデックス番号。

WAITP (-E)
セマフォーの値が正数になるのを待機しているジョブのプロセス ID。

WAITPJID (-Ej)
セマフォーの値が正数になるのを待機しているジョブの修飾ジョブ名。

WAITPTID (-ET)
セマフォーの値が正数になるのを待機しているスレッドのスレッド ID。

WAITVAL (-E)
スレッドは、セマフォーがこの値になるまで待機します。

WAITZ (-E)
セマフォーの値がゼロになるのを待機しているジョブのプロセス ID。

WAITZJID (-Ej)
セマフォーの値がゼロになるのを待機しているジョブの修飾ジョブ名。

WAITZTID (-ET)
セマフォーの値がゼロになるのを待機しているスレッドのスレッド ID。

WJOBID (-Ej)
メッセージの受信を待っているジョブの修飾ジョブ名。

WPID (-E)
メッセージの受信を待っているジョブのプロセス ID。

WTID (-E)
メッセージの受信を待っているスレッドのスレッド ID。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

184 ページの『ipcrm - プロセス間通信 ID を削除する』

locale - ロケール特定情報を入手する

構文

locale [-a]

locale [-ck] *name ...*

説明

locale ユーティリティーは、現行の環境に関する情報を標準出力に表示します。

最初の構文形式では、**locale** はロケール環境変数の名前および値を書き込みます。 **-a** オプションを指定すると、 **locale** は、システム上で使用可能なすべてのロケール名を書き込みます。

2 番目の構文形式では、**locale** は、*name* で指定されたロケール・カテゴリーまたはロケール・キーワードに関する詳細情報を書き込みます。

オプション

-a 使用可能なすべてのロケールに関する情報を書き込みます。

-c ロケール・カテゴリーの名前を表示します。

-k ロケール・キーワードの名前を表示します。

オペランド

name オペランドは、次のいずれかのロケール・カテゴリーまたはロケール・キーワードです。

- カテゴリー LC_CTYPE では、キーワードには、alnum、alpha、blank、cntrl、digit、graph、lower、print、punct、space、upper、xdigit、および codeset が含まれます。
- カテゴリー LC_MESSAGES では、キーワードには、yesexpr、noexpr、yesstr、および nostr が含まれます。
- カテゴリー LC_MONETARY では、キーワードには、int_curr_symbol、currency_symbol、mon_decimal_point、mon_grouping、mon_thousands_sep、positive_sign、negative_sign、int_frac_digits、frac_digits、p_cs_precedes、p_sep_by_space、n_cs_precedes、n_sep_by_space、p_sign_posn、n_sign_posn、debit_sign、credit_sign、left_parenthesis、right_parenthesis、および crncystr が含まれます。
- カテゴリー LC_NUMERIC では、キーワードには、decimal_point、thousands_sep、grouping、および radixchar が含まれます。
- カテゴリー LC_TIME では、キーワードには、abday、abday_1、abday_2、abday_3、 abday_4、abday_5、 abday_6、 abday_7、day、day_1、day_2、day_3、day_4、day_5、day_6、day_7、abmon、ab_mon1、abmon_2、abmon_3、abmon_4、abmon_5、abmon_6、abmon_7、abmon_8、abmon_9、abmon_10、abmon_11、abmon_12、mon、mon_1、mon_2、mon_3、mon_4、mon_5、mon_6、mon_7、mon_8、mon_9、mon_10、mon_11、mon_12、d_t_fmt、d_fmt、t_fmt、am_pm、am_str、pm_str、era、era_d_fmt、era_year、t_fmt_ampm、era_t_fmt、era_d_t_fmt、および alt_digits が含まれます。

終了状況

- 0 正常終了。
- >0 不成功の場合。

例

1. ロケール環境変数の現行値を表示する場合:

   ```
   locale - ロケール特定情報を入手する
   ```

2. システム上で使用可能なロケールのリストを表示する場合:

   ```
   locale -a
   ```

関連タスク:

67 ページの『iconv - ある CCSID から別の CCSID に文字を変換する』
67 ページの『sed - ストリーム・エディター』
73 ページの『sort - テキスト・ファイルをソート、マージ、またはシーケンス検査する』
75 ページの『split - ファイルを分割する』
78 ページの『uniq - ファイル内の繰り返し行を報告または抽出する』
76 ページの『tr - 文字を変換する』

関連情報:

ロケールの概要

logger - メッセージをログに記録する

構文

logger [-is] [-f *file*] **[-t** *tag*] [*message* ...]

説明

logger ユーティリティーは、QHST システム・ログにメッセージを書き込むためのシェル・コマンド・インターフェースを提供します。 *message* を指定せず、**-f** フラグも設定されていなかった場合は、標準入力がログに書き込まれます。

オプション

-i **logger** プロセスのプロセス ID を各行にログ記録します。

-s *message* を、システム・ログのほかに標準エラーにもログ記録します。

-f 指定した *file* をログ記録します。

-t ログの各行に、指定した *tag* を付けます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. ファイル test.output.log をシステム・ログに送る場合 :

```
logger -f test.output.log
```

2. メッセージをシステム・ログと標準エラーに送り、タグを付ける場合 :

```
logger -s -t 'Tag your are it' My message is simple
```

logname - ユーザーのログイン名を表示する

構文

logname

説明

logname ユーティリティーは、ユーザーのログイン名を標準出力に書き込み、その後ろに改行を入れます。

logname ユーティリティーは、環境に信頼性がない場合に **LOGNAME** 環境変数および **USER** 環境変数を明示的に無視します。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

183 ページの『id - ユーザー ID を戻す』

sysval - システム値またはネットワーク属性を検索する

構文

sysval [-p] *systemValue ...*

sysval -n [-p] *networkAttr ...*

説明

sysval ユーティリティーは、ネットワーク属性用の i5/OS システム値を表示します。出力では行ごとにシステム値またはネットワーク属性が 1 つずつ表示されます。

注:	これは i5/OS に固有のユーティリティーです。

オプション

-n ネットワーク属性を表示します。

-p システム値または値をもったネットワーク属性名を表示します。

オペランド

有効なシステム値の名前と説明は、Retrieve System Values API を参照してください。有効なネットワーク属性の名前と説明は、Retrieve Network Attributes API を参照してください。

例

1. QDATE システム値を表示する場合:

```
sysval QDATE
```

2. SYSNAME ネットワーク属性を表示する場合:

```
sysval -n SYSNAME
```

tee - 標準入力を複製する

構文

tee [-ai] [*file* ...]

説明

tee ユーティリティーは、標準入力を標準出力にコピーして、0 個以上の *file* の中にコピーを作成します。出力はバッファーには格納されません。

tee ユーティリティーでは、**-i** オプションを指定した場合を除き、すべてのシグナルに対してデフォルトのアクションがとられます。

オプション

-a ファイルの内容を上書きしないで、ファイルの末尾に出力が追加されます。

-i SIGINT シグナルは無視されます。

環境変数

tee は、次の環境変数の影響を受けます。

QIBM_CCSID
tee によって作成されるファイルは、環境変数の値により指定される CCSID で作成されます。

終了状況

- 0 正常終了。
- >0 エラー発生。

例

1. コマンドの出力を 3 つの別ファイルに保管する場合 :

```
grep 'off_set=' code/*.java | tee file1 file2 file3 > logfile
```

2. ファイル back9 の作業用バックアップ・コピーを作成する場合 :

```
cat back9 | tee pro.tees pro.tees.bak
```

関連タスク:

140 ページの『echo - 引数を標準出力に書き込む』

ulimit - リソース限界を設定または表示する

構文

ulimit [-HS] [-acdfmnst] [*limit* **]**

説明

ulimit ユーティリティーは、リソース限界を設定または表示します。リソース限界は、現行のプロセスおよびリソース限界が設定された後に開始されたすべてのプロセスに適用されます。

それぞれのリソースごとに、ハード限界つまり最大限界、およびソフト限界つまり現行限界があります。ソフト限界は、ハード限界以下のどのような値にでも変更することができます。ハード限界は、ソフト限界以上のどのような値にでも変更することができます。ハード限界を大きくすることができるのは、*JOBCTL 特殊権限を持っているユーザーのみです。

i5/OS 上で設定できるリソース限界は、ファイル・サイズ (**-f**) および記述子の数 (**-n**) だけです。すべてのリソース限界はすべて表示可能です。

オプション

-a すべてのリソース限界を表示します。

-c コア・ファイルの最大サイズのリソース限界を表示します (K バイト単位)。

-d プロセスのデータ・セグメントの最大サイズのリソース限界を表示します (K バイト単位)。

-f ファイルの最大サイズのリソース限界を設定または表示します (K バイト単位)。

-H リソースのハード限界を設定または表示します。

-m プロセスの合計使用可能ストレージの最大サイズのリソース限界を表示します。

-n プロセスがオープンすることができるファイル記述子の最大数のリソース限界を設定または表示します。

-s プロセスのスタックの最大サイズのリソース限界を表示します (K バイト単位)。

-S リソースのソフト限界を設定または表示します。

-t CPU 時間の最大量のリソース限界を表示します (秒数)。

オペランド

limit が指定しないと、リソース限界の値が表示されます。 **-H** オプションを指定すると、ハード限界が表示されます。その他の場合は、ソフト限界が表示されます。

limit を指定すると、リソース限界の値が設定されます。 *limit* は、算術式、またはストリング「unlimited」(限界なしを示す) とすることができます。 **-H** および **-S** のいずれのオプションも指定されていない場合、ハード限界およびソフト限界の両方が指定されます。

リソースが指定されていない場合、デフォルトはファイル・サイズ (**-f**) リソース限界になります。

終了状況

- 0 正常終了。
- >0 不成功の場合。

関連タスク:

136 ページの『umask - ファイル・モード作成マスクを入手または設定する』

『uname - システム名を戻す』

uname - システム名を戻す

構文

uname [-amnrsv]

説明

uname ユーティリティーは、オペレーティング・システムの実装の名前を標準出力に書き込みます。 オプションを指定すると、1 つまたは複数のシステム特性を表す文字列が標準出力に書き込まれます。

-a フラグを指定するか、複数のフラグを指定した場合は、すべての出力がスペースで区切られて、1 行に書き込まれます。

オプション

-a **-m**、**-n**、**-r**、**-s**、および **-v** オプションをすべて指定した場合と同じ動作をします。

-m システムのハードウェアのタイプ名が標準出力に書き込まれます。

-n システムの名前が標準出力に書き込まれます。

-r オペレーティング・システムの現行リリース・レベルが標準出力に書き込まれます。

-s オペレーティング・システム実装の名前が標準出力に書き込まれます。

-v オペレーティング・システムのこのリリースのバージョン・レベルが標準出力に書き込まれます。

終了状況

- 0 正常終了。
- >0 エラー発生。

関連タスク:

195 ページの『ulimit - リソース限界を設定または表示する』

アプリケーション・プログラミング・インターフェース

これらのアプリケーション・プログラミング・インターフェース (API) は Qshell で提供されます。

QzshSystem() - QSH コマンドを実行する

構文

```
#include <qshell.h>

int QzshSystem( const char *command );
```

スレッド・セーフ: あり

QzshSystem() 関数では、子プロセスを作成してその子プロセス内で **qsh** を呼び出すことによって、指定したシェル・コマンドが実行されます。 **qsh** は、*command* を解釈し、実行して終了します。

子プロセスが終了すると、**QzshSystem()** 関数から戻ります。 **QzshSystem()** では、子プロセスが終了するまで待っている間は、SIGQUIT シグナルと SIGINT シグナルは無視され、SIGCHLD シグナルはブロックされます。呼び出しプロセスによって開始された他の子プロセスの状況情報は、**QzshSystem()** 関数の影響を受けることはありません。

パラメーター

**command* (入力) 実行するシェル・コマンドを備えたヌル終了ストリングを指すポインター。

権限

参照するオブジェクト	必要な権限	errno
パス名の中で該当の実行可能ファイルの前にある各ディレクトリー	*X	EACCES
実行可能ファイル	*X	EACCES
実行可能ファイルがシェル・スクリプトの場合	*RX	EACCES

戻り値

値 **QzshSystem()** は正常に終了しました。戻り値は、**waitpid()** 関数から戻された状況値です。アプリケーションでは、sys/wait.h ヘッダー・ファイルに備えられたマクロを使用して、子プロセスからの状況情報を解釈することができます。戻り値は負の値のこともあります。

-1 **QzshSystem()** は失敗しました。*errno* が、該当のエラーを示す値に設定されます。

エラー条件

QzshSystem() が失敗した場合、*errno* には、一般に次のエラーが示されます。状況によっては、*errno* に下記以外のエラーが示されることもあります。

[EACCES]
許可が拒否されました。

オブジェクト・アクセス許可では禁止されている方法でオブジェクトにアクセスしようとしました。

スレッドには、指定したファイル、ディレクトリー、コンポーネント、またはパスに対するアクセス権がありません。

[ECHILD]
呼び出しプロセスには、待機操作を実行できる子プロセスは残っていません。

[EFAULT]

引数として使用されたアドレスが正しくありません。

呼び出しで引数を使おうとしたら、システムで無効なアドレスが検出されました。

この関数に渡されたパラメーターにアクセスしようとしたときに、システムで無効なアドレスが検出されました。

[EINVAL]

引数に指定した値が正しくありません。

関数に正しくない引数値が渡されたか、オブジェクトに対する操作の際に、そのオブジェクトのタイプ用としてサポートされていない操作を指定しました。

[ENOMEM]

記憶域割り振り要求が失敗しました。

関数が記憶域の割り振りを必要としているときに、使用可能な記憶域がありませんでした。

要求された関数を実行するための十分なメモリーがありません。

[ENOSYSRSC]

要求を実行するために必要なだけの使用可能なシステム・リソースがありません。

[EUNKNOWN]

不明なシステム状態。

不明なシステム状態が原因で操作が失敗しました。ジョブ・ログ内にメッセージがあればそれを読み、そこに示されているエラーを訂正します。 その後、操作を再試行してください。

例: QzshSystem() 関数と QzshCheckShellCommand() 関数の使用

次の例は、**QzshSystem()** 関数と **QzshCheckShellCommand()** 関数の使用方法を示しています。

```
#include <stdio.h>
#include <qshell.h>
#include <sys/wait.h>
#include <errno.h>

int main(int argc, char *argv[])
{
   int status;
   char *command = "ls";

   /* Verify the user has access to the specified command. */
   if (QzshCheckShellCommand(command, NULL) == 0) {
      /* Run the specified command. */
      status = QzshSystem(command);
      if (WIFEXITED(status)) {
         printf("Command %s completed with exit status %d.\n",
                command, WEXITSTATUS(status));
      }
      else if (WIFSIGNALED(status)) {
         printf("Command %s ended with signal %d.\n",
                command, WTERMSIG(status));
      }
      else if (WIFEXCEPTION(status)) {
         printf("Command %s ended with exception.\n", command);
      }
   }
   else
```

```
      printf("Error %d finding command %s\n", errno, command);

   return(0);
}
```

出力

ls コマンドを戻り状況値 0 を戻して完了します。

関連概念:

『QzshCheckShellCommand() - QSH コマンドを検索する』

関連情報:

spawn() - プロセスを作成

waitpid() - 特定の子プロセスの待機

QzshCheckShellCommand() - QSH コマンドを検索する

構文

```
#include <qshell.h>

int QzshCheckShellCommand( const char *command, const char *path );
```

スレッド・セーフ: あり

QzshCheckShellCommand() 関数では、次のものを検索すれば、指定したシェル・コマンドが見つかります。

- 最初に、組み込みユーティリティー
- 次に、*path* または **PATH** 環境変数で指定されたリスト内の各ディレクトリー

アプリケーションで **QzshCheckShellCommand()** を使用すれば、 *command* が存在し、しかもユーザーが *command* に対する権限を持っていることを実行の前に確認することができます。

パラメーター

**command* (入力) 検索するシェル・コマンドを備えたヌル終了ストリングを指すポインター。

path* (入力) 検索するディレクトリーのリスト (コロンで区切ったもの) を備えたヌル終了ストリングを指すポインター。このパラメーターが NULL の場合は、QzshCheckShellCommand()** では **PATH** 環境変数の値が使用されます。

権限

コマンドが実行可能ファイルである場合は、ユーザーは次の権限を持っている必要があります。

参照するオブジェクト	必要な権限	errno
パス名の中で該当の実行可能ファイルの前にある各ディレクトリー	*X	EACCES
実行可能ファイル	*X	EACCES
実行可能ファイルがシェル・スクリプトの場合	*RX	EACCES

戻り値

0 **QzshCheckShellCommand()** は正常に終了しました。 *command* は現行環境の中で見つかりました。

-1 **Qp0zCheckShellCommand()** は失敗しました。 *errno* が、該当のエラーを示す値に設定されます。

エラー条件

QzshCheckShellCommand() が失敗した場合、 *errno* には、通常次のエラーが示されます。状況によっては、*errno* に下記以外のエラーが示されることもあります。

[EACCES]
許可が拒否されました。

オブジェクト・アクセス許可では禁止されている方法でオブジェクトにアクセスしようとしました。

スレッドには、指定したファイル、ディレクトリー、コンポーネント、またはパスに対するアクセス権がありません。

[EFAULT]
引数として使用されたアドレスが正しくありません。

呼び出しで引数を使おうとしたら、システムで無効なアドレスが検出されました。

この関数に渡されたパラメーターにアクセスしようとしたときに、システムで無効なアドレスが検出されました。

[EINVAL]
引数に指定した値が正しくありません。

関数に正しくない引数値が渡されたか、オブジェクトに対する操作の際に、そのオブジェクトのタイプ用としてサポートされていない操作を指定しました。

[ENOMEM]
記憶域割り振り要求が失敗しました。

関数が記憶域の割り振りを必要としているときに、使用可能な記憶域がありませんでした。

要求された関数を実行するための十分なメモリーがありません。

[ENOENT]
該当のパスまたはディレクトリーがありません。

指定されたパス名のディレクトリーまたは構成要素が存在しません。

指定されたファイルまたはディレクトリーが存在しないか、空ストリングです。

[EUNKNOWN]
不明なシステム状態。

不明なシステム状態が原因で操作が失敗しました。ジョブ・ログ内にメッセージがあればそれを読み、そこに示されているエラーを訂正します。 そして操作を再試行してください。

例: QzshCheckShellCommand() 関数の使用

この関数の使用例については、QzshSystem() 関数を参照してください。

関連概念:

197 ページの『QzshSystem() - QSH コマンドを実行する』

例: qsh セッションに接続するリモート・クライアントの使用

この例は、対話式 Qshell セッションを開始するためのリモート・クライアントおよびサーバーを示します。

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に対する責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

例: サーバー・プログラム

```
/*********************************************************************/
/*                                                                   */
/* Name:           server.c                                          */
/*                                                                   */
/* Description:    This program is a server for starting interactive  */
/*                 qsh sessions on remote clients.  The program      */
/*                 listens for connections from clients.  When a     */
/*                 connection is accepted, it reads the user name    */
/*                 and password of the client.  It then swaps to the */
/*                 the specified user profile and spawns a new       */
/*                 process running the qsh shell interpreter that    */
/*                 handles the connection.                           */
/*                                                                   */
/* Parameters:  1. Port number to listen for connections on.         */
/*                                                                   */
/* Notes:       1. The user name and password are sent as plain text */
/*                 from the client.                                  */
/*              2. The user profile running this program must have   */
/*                 authority to the QSYGETPH, QSYRLSPH, and          */
/*                 QWTSETP APIs.                                     */
/*              3. You will need to change the value of the NLSPATH  */
/*                 environment variable if your system is using a    */
/*                 different language than 2924.                     */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/* Includes                                                          */
/*********************************************************************/

#include <stdio.h>          /* fopen(), vfprintf() */
#include <sys/socket.h>     /* socket(), bind(), and so on. */
#include <netinet/in.h>     /* sockaddr_in, INADDR_ANY, and so on */
#include <arpa/inet.h>      /* inet_ntoa() */
#include <spawn.h>          /* spawn() */
```

```
#include <unistd.h>          /* close(), read(), and so on */
#include <stdlib.h>          /* exit()*/
#include <stdarg.h>          /* va_start(), va_end() */
#include <qp0z1170.h>        /* Qp0zInitEnv() */
#include <qsygetph.h>        /* QSYGETPH() */
#include <qwtsetp.h>         /* QWTSETP() */
#include <qsyrlsph.h>        /* QSYRLSPH() */
#include <qusec.h>           /* Qus_EC_t */
#include <pwd.h>             /* getpwnam() */
#include <ctype.h>           /* toupper() */
#include <time.h>            /* ctime(), time() */
#include <except.h>          /* Exception and cancel handling */
#include <errno.h>           /* errno and constants */

/*********************************************************************/
/* Constants                                                         */
/*********************************************************************/

#define DEFAULT_BUF 4096
#define DEFAULT_PORT 6042
#define NULL_PH "\0\0\0\0\0\0\0\0\0\0\0\0"
#define PH_SIZE 12
#define NAME_SIZE 11
#undef PATH_MAX
#define PATH_MAX 4096

/*********************************************************************/
/* Global Variables                                                  */
/*********************************************************************/

/* For logging errors */
FILE *log_fp;
char log_file[] = "/tmp/qsh_server.log";
char log_buffer[DEFAULT_BUF];

/*********************************************************************/
/* Function Prototypes                                               */
/*********************************************************************/

int strtoupper(char *);
int GetString(int, char *, size_t);
void LogError(char *, ...);
void SendError(int, char *, ...);
void CleanupHandler(_CNL_Hndlr_Parms_T *);

int main(int argc, char *argv[])
{
  int sfd;                    /* Server's listening socket */
  int cfd;                    /* Socket connected to client */
  int on=1;                   /* Flag for setsockopt() */
  struct sockaddr_in my_addr; /* Address server binds to */
  struct sockaddr_in client_addr; /* Addrress of connected client */
  int client_addr_len;        /* Length of client's socket address */
  unsigned short port;        /* Server's TCP port */
  char server_ph[PH_SIZE+1] = NULL_PH; /* Server's profile handle */
  char client_ph[PH_SIZE+1] = NULL_PH; /* Client's profile handle */
  char profile[NAME_SIZE];    /* User profile read from client */
  char password[NAME_SIZE];   /* Password read from client */
  char sy_profile[NAME_SIZE]; /* User profile for i5/OS APIs */
  char sy_password[NAME_SIZE]; /* Password for i5/OS APIs */
  char server_profile[NAME_SIZE] = "*CURRENT  ";
  char no_pwd[NAME_SIZE]         = "*NOPWD    ";
  struct passwd *cpw;         /* User information for client */
  Qus_EC_t error = { sizeof(Qus_EC_t), 0 }; /* Error code for SPIs */

  /* Parameters for spawn() to shell process */
  char qsh_pgm[] = "/QSYS.LIB/QSHELL.LIB/QZSHSH.PGM";
```

```
char *args[5];              /* Argument array */
char *envs[10];             /* Environment variable array */
int fd_count;               /* Number of descriptors */
int fd_map[3];              /* Map of descriptors */
struct inheritance inherit; /* Inheritance options */
char server_dir[] = "/";    /* Default current working directory */

/* Environment variables */
char home_var[PATH_MAX+10];
char logname_var[NAME_SIZE+10];
char path_var[] = "PATH=/usr/bin:.:/QOpenSys/usr/bin";
char stdio_var[] = "QIBM_USE_DESCRIPTOR_STDIO=I";
char terminal_type_var[] = "TERMINAL_TYPE=REMOTE";
char nlspath_var[] = "NLSPATH=/QIBM/ProdData/OS400/Shell/MRI2924/%N";

volatile _INTRPT_Hndlr_Parms_T ca; /* For exception handler */

/*********************************************************************/
/* Process the input parameters. */
/*********************************************************************/

/* Use the default port if one is not specified. */
if (argc < 2) {
  port = DEFAULT_PORT;
}

else {
  port = atoi(argv[1]);
}

/*********************************************************************/
/* Initialize the server environment. */
/*********************************************************************/

/* Initialize for environment variables. */
Qp0zInitEnv();

/* Change to default directory. */
chdir(server_dir);

/* Initialize the server's profile handle. */
QSYGETPH(server_profile, no_pwd, server_ph, &error);
if (error.Bytes_Available != 0) {
  LogError("Could not get profile handle for server, "
           "QSYGETPH() failed with exception %7.7s\n",
           error.Exception_Id);
  exit(1);
}

/*********************************************************************/
/* Set up the listening socket. */
/*********************************************************************/

/* Create a socket. */
if ((sfd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP)) < 0) {
  LogError("socket() failed, errno=%d\n", errno);
  exit(1);
}

#pragma cancel_handler(CleanupHandler, sfd)
#pragma exception_handler(Cleanup, ca, _C1_ALL, _C2_ALL)

/* Allow re-use of this socket address. */
if (setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR, (char *)&on,
               sizeof(int)) != 0) {
  LogError("setsockopt() failed, errno=%d\n", errno);
  exit(1);
```

```
}

/* Bind to a port. */
memset(&my_addr, '\0', sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_port = port;
my_addr.sin_addr.s_addr = INADDR_ANY;
if (bind(sfd, (struct sockaddr *)&my_addr, sizeof(my_addr)) != 0) {
  LogError("bind() failed for port %d, errno=%d\n", port, errno);
  close(sfd);
  exit(1);
}

/* Make this a listening socket. */
if (listen(sfd, 10) != 0) {
  LogError("listen() failed, errno=%d\n", errno);
  close(sfd);
  exit(1);
}

/*********************************************************************/
/* Accept connections from clients. */
/*********************************************************************/

while (1) {
  if ((cfd = accept(sfd, NULL, 0)) < 0) {
    LogError("accept() failed, errno=%d\n", errno);
    close(sfd);
    exit(1);
  }

  /* Read the user profile and password from the client.  The client
     sends two null-terminated strings - the first one is the user
     profile and the second one is the password. */
  if (GetString(cfd, profile, 11) != 0) {
    getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
    LogError("Could not read profile from client at %s, port %hu\n",
             inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
    close(cfd);
    continue;
  }

  if (GetString(cfd, password, 11) != 0) {
    getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
    LogError("Could not read password from client  at %s, port %hu\n",
             inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
    close(cfd);
    continue;
  }

  /* Check for the special values that turn off password checking in QSYGETPH(). */
  if ((profile[0] == '*') || (password[0] == '*')) {
    getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
    LogError("Invalid password sent from client at %s, port %hu\n",
             inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
    close(cfd);
    continue;
  }

  /* QSYGETPH() requires that the profile be exactly ten characters,
     left-aligned in the field, and padded with blanks. */
  strtoupper(profile);
  sprintf(sy_profile, "%-10.10s", profile);

  /* Get the profile handle for the client's user profile. */
  QSYGETPH(sy_profile, password, client_ph, &error, strlen(password), 0);
  if (error.Bytes_Available != 0) {
```

```
  LogError("Could not get profile handle for profile %s, "
           "QSYGETPH() failed with exception %7.7s\n",
           sy_profile, error.Exception_Id);
  SendError(cfd, "Could not get profile handle for profile %s\n",
            sy_profile);
  close(cfd);
  continue;
}

/* Switch to client's user profile. */
QWTSETP(client_ph, &error);
if (error.Bytes_Available != 0) {
  LogError("Could not switch to profile %s, "
           "QWTSETP() failed with exception %7.7s\n",
           sy_profile, error.Exception_Id);
  SendError(cfd, "Could not switch to profile %s\n", sy_profile);
  QSYRLSPH(client_ph, NULL);
  close(cfd);
  continue;
}

/* Get the info for this user profile. */
if ((cpw = getpwnam(profile)) == NULL) {
  /* Log error. */
  LogError("Could not retrieve information for profile %s, "
           "getpwnam() failed with errno=%d\n",
           profile, errno);
  SendError(cfd, "Could not retrieve information for profile %s\n",
            profile);

  /* Switch back to the server's user profile. */
  QWTSETP(server_ph, &error);
  if (error.Bytes_Available != 0) {
    LogError("Could not switch back to server's profile, "
             "QWTSETP() failed with exception %7.7s\n",
             error.Exception_Id);
    break;
  }

  /* Release the client's profile handle. */
  QSYRLSPH(client_ph, NULL);
  if (error.Bytes_Available != 0) {
    LogError("Could not release client's profile handle, "
             "QSYRLSPH() failed with exception %7.7s\n",
             error.Exception_Id);
    break;
  }
  close(cfd);
  continue;
}

/* Build the file descriptor map for the child. */
fd_count = 3;
fd_map[0] = cfd;
fd_map[1] = cfd;
fd_map[2] = cfd;

/* Build the argv array for the child. */
args[0] = qsh_pgm;
args[1] = "-login";      /* Do login processing */
args[2] = "-s";          /* Take input from stdin */
args[3] = "-i";          /* Run as an interactive shell */
args[4] = NULL;

/* Build the environ array for the child. */
sprintf(home_var, "HOME=%s", cpw->pw_dir);
sprintf(logname_var, "LOGNAME=%s", cpw->pw_name);
```

```
    envs[0] = home_var;
    envs[1] = logname_var;
    envs[2] = path_var;
    envs[3] = stdio_var;
    envs[4] = terminal_type_var;
    envs[5] = nlspath_var;
    envs[6] = NULL;

    /* Set up the inheritance structure. */
    memset(&inherit, '\0', sizeof(struct inheritance));
    inherit.flags = SPAWN_SETTHREAD_NP;
    inherit.pgroup = SPAWN_NEWPGROUP;

    /* Change to the home directory for the client.  The child process
       inherits this as its current working directory. */
    chdir(cpw->pw_dir);

    /* Start a child process running the shell interpreter. */
    if (spawn(args[0], fd_count, fd_map, &inherit, args, envs) < 0) {
      LogError("Could not start qsh process, spawn() failed with "
               "errno=%d\n", errno);
      SendError(cfd, "Could not start qsh process\n");
    }

    /* Clean up for the next connection. */
    chdir(server_dir);
    close(cfd);

    /* Switch back to server's user profile. */
    QWTSETP(server_ph, &error);
    if (error.Bytes_Available != 0) {
      LogError("Could not switch back to server's profile, "
               "QWTSETP() failed with exception %7.7s\n",
               error.Exception_Id);
      break;
    }

    /* Release the client's profile handle. */
    QSYRLSPH(client_ph, &error);
    if (error.Bytes_Available != 0) {
      LogError("Could not release client's profile handle, "
               "QSYRLSPH() failed with exception %7.7s\n",
               error.Exception_Id);
      break;
    }
  } /* End of while */

  /* Clean up. */
  close(sfd);

  #pragma disable_handler /* Exception handler */
  #pragma disable_handler /* Cancel handler */

  exit(0);
  return 0;

  /* Exception handler */
  Cleanup:

  LogError("Unexpected exception %7.7s\n", ca.Msg_Id);
  close(sfd);
  exit(1);
} /* End of main() */


/*
 * Convert a string to uppercase.
```

```
 */

int
strtoupper(char *string)
{
  for ( ; *string != '\0'; ++string)
    *string = toupper(*string);

  return 0;
} /* End of strtoupper() */


/*
 * Read a string from a socket.
 */

int
GetString(int fd, char *buffer, size_t nbytes)
{
  char c;
  do {
    if (read(fd, &c, 1) != 1) {
      return -1;
    }
    *buffer++ = c;
    if (--nbytes == 0) {
      return 0;
    }
  } while (c != '\0');

  return 0;
} /* End of GetString() */


/*
 * Write an error message to the log file.
 */

void LogError(char *format, ...)
{
  va_list ap;
  time_t now;                /* Time stamp */

  /* If needed, open the log file. */
  if (log_fp == NULL) {
    log_fp = fopen(log_file, "w");
    if (log_fp == NULL) {
      return;
    }
  }

  /* Write timestamp to the log file. */
  now=time(NULL);
  fprintf(log_fp, "\n%s", ctime(&now));

  /* Write the formatted string to the log file. */
  va_start(ap, format);
  vfprintf(log_fp, format, ap);
  va_end(ap);

  /* Flush output to log file. */
  fflush(log_fp);

  return;
} /* End of LogError() */
```

```
/*
 * Send an error message to the client.
 */

void SendError(int fd, char *format, ...)
{
  va_list ap;

  /* Build the formatted string. */
  va_start(ap, format);
  vsprintf(log_buffer, format, ap);
  va_end(ap);

  /* Write the formatted string. */
  write(fd, log_buffer, strlen(log_buffer));

  return;
} /* End of SendError() */


/*
 * Handler to clean up when the program is canceled.
 */

void CleanupHandler(_CNL_Hndlr_Parms_T *cancel_info)
{
  int sfd;
  sfd = *((int *)cancel_info->Com_Area);
  close(sfd);
} /* End of CleanupHandler() */
```

例: クライアント・プログラム

```
/*********************************************************************/
/*                                                                   */
/* Name:           qshc.c                                            */
/*                                                                   */
/* Description:    This program is a client for an interactive qsh   */
/*                 session running on a server. The program          */
/*                 first connects to a server on the specified       */
/*                 server and sends the user name and password of    */
/*                 the client. After the qsh session is started,     */
/*                 the program takes input from stdin and sends it   */
/*                 to the server and receives output from the server */
/*                 and displays it on stdout.                        */
/*                                                                   */
/* Parameters:  1. Host running the qsh server (either host name or  */
/*                 IP address).                                      */
/*                                                                   */
/* Options:     1. -n to force prompt for user name and password.    */
/*              2. -p to specify port of qsh server.                 */
/*                                                                   */
/* Notes:       1. The user name and password are sent as plain text */
/*                 to the server.                                    */
/*              2. All translations from ASCII to EBCDIC are done by */
/*                 this program on the client.                       */
/*              3. The program includes translation tables for       */
/*                 converting between EBCDIC code page 37 (US English)*/
/*                 and ASCII code page 850 (US English). You can     */
/*                 modify these tables to support other code pages.  */
/*                 Or if your system supports the iconv APIs, you    */
/*                 can define USE_ICONV to translate using iconv().  */
/*              4. This program has been tested on AIX 4.1.5 and     */
/*                 Linux 2.0.29.                                     */
/*                                                                   */
/*********************************************************************/
```

```
/* Remove the comments from the following line to use iconv(). */
/* #define USE_ICONV 1 */

/*********************************************************************/
/* Includes                                                          */
/*********************************************************************/

#include <stdio.h>           /* perror() */
#include <sys/socket.h>      /* socket(), bind(), and so on */
#include <netinet/in.h>      /* sockaddr_in, INADDR_ANY, and so on */
#include <unistd.h>          /* close(), read(), write() and so on */
#include <stdlib.h>          /* exit() */
#include <stdlib.h>          /* exit(), memset() */
#include <sys/ioctl.h>       /* ioctl() */
#include <errno.h>           /* errno and values */
#include <string.h>          /* strlen() */
#include <arpa/inet.h>       /* inet_addr() */
#include <netdb.h>           /* gethostbyname() */
#include <pwd.h>             /* getpwuid() */
#include <signal.h>          /* sigaction(), and so on */

#ifdef _AIX
#include <sys/select.h>      /* select() */
#include <strings.h>         /* bzero() for FD_ZERO */
#endif
#ifdef __linux__
#include <sys/time.h>        /* FD_SET(), select */
#endif

#ifdef USE_ICONV
#include <iconv.h>           /* iconv(), and so on */
#endif

/*********************************************************************/
/* Constants */
/*********************************************************************/

#define QSH_PORT 6042
#define DEFAULT_BUF 4096

/*********************************************************************/
/* Types */
/*********************************************************************/

typedef unsigned char uchar;

/*********************************************************************/
/* Global Variables                                                  */
/*********************************************************************/

char *sysname;          /* Long host name of server system */

#ifdef USE_ICONV
iconv_t ecd;            /* Conversion descriptor for ASCII to EBCDIC */
iconv_t acd;            /* Conversion descriptor for EBCDIC to ASCII */

#else
/* EBCDIC to ASCII translation table */
static uchar AsciiTable[256] =
{
  0x00,0x01,0x02,0x03,0x20,0x09,0x20,0x7f, /* 00-07 */
  0x20,0x20,0x20,0x0b,0x0c,0x0d,0x0e,0x0f, /* 08-0f */
  0x10,0x11,0x12,0x13,0x20,0x0a,0x08,0x20, /* 10-17 */
  0x18,0x19,0x20,0x20,0x20,0x1d,0x1e,0x1f, /* 18-1f */
  0x20,0x20,0x1c,0x20,0x20,0x0a,0x17,0x1b, /* 20-27 */
  0x20,0x20,0x20,0x20,0x20,0x05,0x06,0x07, /* 28-2f */
```

```
  0x20,0x20,0x16,0x20,0x20,0x20,0x20,0x04, /* 30-37 */
  0x20,0x20,0x20,0x20,0x14,0x15,0x20,0x1a, /* 38-3f */
  0x20,0x20,0x83,0x84,0x85,0xa0,0xc6,0x86, /* 40-47 */
  0x87,0xa4,0xbd,0x2e,0x3c,0x28,0x2b,0x7c, /* 48-4f */
  0x26,0x82,0x88,0x89,0x8a,0xa1,0x8c,0x8b, /* 50-57 */
  0x8d,0xe1,0x21,0x24,0x2a,0x29,0x3b,0xaa, /* 58-5f */
  0x2d,0x2f,0xb6,0x8e,0xb7,0xb5,0xc7,0x8f, /* 60-67 */
  0x80,0xa5,0xdd,0x2c,0x25,0x5f,0x3e,0x3f, /* 68-6f */
  0x9b,0x90,0xd2,0xd3,0xd4,0xd6,0xd7,0xd8, /* 70-77 */
  0xde,0x60,0x3a,0x23,0x40,0x27,0x3d,0x22, /* 78-7f */
  0x9d,0x61,0x62,0x63,0x64,0x65,0x66,0x67, /* 80-87 */
  0x68,0x69,0xae,0xaf,0xd0,0xec,0xe7,0xf1, /* 88-8f */
  0xf8,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f,0x70, /* 90-97 */
  0x71,0x72,0xa6,0xa7,0x91,0xf7,0x92,0xcf, /* 98-9f */
  0xe6,0x7e,0x73,0x74,0x75,0x76,0x77,0x78, /* a8-a7 */
  0x79,0x7a,0xad,0xa8,0xd1,0xed,0xe8,0xa9, /* a8-af */
  0x5e,0x9c,0xbe,0xfa,0xb8,0x15,0x14,0xac, /* b0-b7 */
  0xab,0xf3,0x5b,0x5d,0xee,0xf9,0xef,0x9e, /* b8-bf */
  0x7b,0x41,0x42,0x43,0x44,0x45,0x46,0x47, /* c0-c7 */
  0x48,0x49,0xf0,0x93,0x94,0x95,0xa2,0xe4, /* c8-cf */
  0x7d,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f,0x50, /* d0-d7 */
  0x51,0x52,0xfb,0x96,0x81,0x97,0xa3,0x98, /* d8-df */
  0x5c,0xf6,0x53,0x54,0x55,0x56,0x57,0x58, /* e0-e7 */
  0x59,0x5a,0xfc,0xe2,0x99,0xe3,0xe0,0xe5, /* e8-ef */
  0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37, /* f0-f7 */
  0x38,0x39,0xfd,0xea,0x9a,0xeb,0xe9,0xff  /* f8-ff */
};

/* ASCII to EBCDIC translation table */
static uchar EbcdicTable[256] =
{
  0x00,0x01,0x02,0x03,0x37,0x2d,0x2e,0x2f, /* 00-07 */
  0x16,0x05,0x25,0x0b,0x0c,0x0d,0x0e,0x0f, /* 08-0f */
  0x10,0x11,0x12,0x13,0x3c,0x3d,0x32,0x26, /* 10-17 */
  0x18,0x19,0x3f,0x27,0x22,0x1d,0x1e,0x1f, /* 18-1f */
  0x40,0x5a,0x7f,0x7b,0x5b,0x6c,0x50,0x7d, /* 20-27 */
  0x4d,0x5d,0x5c,0x4e,0x6b,0x60,0x4b,0x61, /* 28-2f */
  0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7, /* 30-37 */
  0xf8,0xf9,0x7a,0x5e,0x4c,0x7e,0x6e,0x6f, /* 38-3f */
  0x7c,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7, /* 40-47 */
  0xc8,0xc9,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6, /* 48-4f */
  0xd7,0xd8,0xd9,0xe2,0xe3,0xe4,0xe5,0xe6, /* 50-57 */
  0xe7,0xe8,0xe9,0xba,0xe0,0xbb,0xb0,0x6d, /* 58-5f */
  0x79,0x81,0x82,0x83,0x84,0x85,0x86,0x87, /* 60-67 */
  0x88,0x89,0x91,0x92,0x93,0x94,0x95,0x96, /* 68-6f */
  0x97,0x98,0x99,0xa2,0xa3,0xa4,0xa5,0xa6, /* 70-77 */
  0xa7,0xa8,0xa9,0xc0,0x4f,0xd0,0xa1,0x07, /* 78-7f */
  0x68,0xdc,0x51,0x42,0x43,0x44,0x47,0x48, /* 80-87 */
  0x52,0x53,0x54,0x57,0x56,0x58,0x63,0x67, /* 88-8f */
  0x71,0x9c,0x9e,0xcb,0xcc,0xcd,0xdb,0xdd, /* 90-97 */
  0xdf,0xec,0xfc,0x70,0xb1,0x80,0xbf,0x40, /* 98-9f */
  0x45,0x55,0xee,0xde,0x49,0x69,0x9a,0x9b, /* a8-a7 */
  0xab,0xaf,0x5f,0xb8,0xb7,0xaa,0x8a,0x8b, /* a8-af */
  0x40,0x40,0x40,0x40,0x40,0x65,0x62,0x64, /* b0-b7 */
  0xb4,0x40,0x40,0x40,0x40,0x4a,0xb2,0x40, /* b8-bf */
  0x40,0x40,0x40,0x40,0x40,0x40,0x46,0x66, /* c0-c7 */
  0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x9f, /* c8-cf */
  0x8c,0xac,0x72,0x73,0x74,0x89,0x75,0x76, /* d0-d7 */
  0x77,0x40,0x40,0x40,0x40,0x6a,0x78,0x40, /* d8-df */
  0xee,0x59,0xeb,0xed,0xcf,0xef,0xa0,0x8e, /* e0-e7 */
  0xae,0xfe,0xfb,0xfd,0x8d,0xad,0xbc,0xbe, /* e8-ef */
  0xca,0x8f,0x40,0xb9,0xb6,0xb5,0xe1,0x9d, /* f0-f7 */
  0x90,0xbd,0xb3,0xda,0xea,0xfa,0x40,0x40  /* f8-ff */
};
#endif /* USE_ICONV */

/*********************************************************************/
```

```
/* Function Prototypes                                               */
/*******************************************************************/

int ConvertToEBCDIC(char *, size_t, char *, size_t);
int ConvertToASCII(char *, size_t, char *, size_t);
int GetPassword(char *, char *, char *);
int Translate(uchar *, size_t, uchar *, uchar *);
void MySignalHandler(int);
void usage(void);

int main (int argc, char *argv[])
{
  struct sigaction sigact;   /* Signal action */
  int c;                     /* Option letter */
  int nflag=0;               /* True when -n option is specified */
  int port=QSH_PORT;         /* Port to connect to on server */
  int sd;                    /* Socket to server */
  fd_set read_set;           /* For select() */
  int rc;                    /* Return code */
  struct sockaddr_in svr_addr; /* AF_INET socket address */
  long ip_addr;              /* IP address of server system */
  struct in_addr host_addr;  /* Host address for gethostbyaddr() */
  char *hostname;            /* Short host name of server system */
  size_t len;                /* Length of input string */
  char *ascii_user;          /* Username in ASCII */
  char *ebcdic_user;         /* Username in EBCDIC */
  char *ascii_pwd;           /* Password in ASCII */
  char *ebcdic_pwd;          /* Password in EBCDIC */
  struct hostent *host_p;    /* Pointer to hostent structure returned by
                                gethostbyname() */
  char *ascii_buf;           /* Buffer for ASCII text */
  char *ebcdic_buf;          /* Buffer for EBCDIC text */
  int buf_size;              /* Amount of data read from server */

  /*******************************************************************/
  /* Initialization. */
  /*******************************************************************/

  #ifdef USE_ICONV
  /* Open the conversion descriptors for converting between ASCII and
     EBCDIC. Assume the server job is running in CCSID 37.
     This must be changed if the server job is running in a
     different CCSID. The input parameters to iconv_open() may need to
     be changed depending on the operating system. This ioonv_open() is
     coded for AIX. */
  if ((acd = iconv_open("IBM-850", "IBM-037")) < 0) {
    perror("qshc: iconv_open() failed for ASCII to EBCDIC");
    exit(1);
  }

  if ((ecd = iconv_open("IBM-037", "IBM-850")) < 0) {
    perror("qshc: iconv_open() failed for EBCDIC to ASCII");
    exit(1);
  }
  #endif /* USE_IOONV */

  /* Set up a signal handler for SIGINT. The signal is sent to the
     process when the user presses <ctrl>c. */
  sigemptyset(&sigact.sa_mask);
  sigact.sa_flags = 0;
  sigact.sa_handler = MySignalHandler;
  if (sigaction(SIGINT, &sigact, NULL) != 0) {
    perror("qshc: sigaction(SIGINT) failed");
    exit(1);
  }

  /*******************************************************************/
```

```
/* Process the input parameters. */
/*********************************************************************/

if (argc < 2) {
  usage();
}

/* Process the options. */
while ((c = getopt(argc, argv, "hnp:")) != EOF) {
  switch (c) {
    case 'n':
      nflag = 1;
      break;
    case 'p':
      port = atoi(optarg);
      break;
    case 'h':
    default:
      usage();
      break;
  } /* End of switch */
} /* End of while */

/* Convert a dotted decimal address to a 32-bit IP address. */
hostname = argv[optind];
ip_addr = inet_addr(hostname);

/* When inet_addr() returns -1 assume the user specified
   a host name. */
if (ip_addr == -1) {
  /*  Try to find the host by name. */
  host_p = gethostbyname(hostname);
  if (host_p) {
    memcpy(&ip_addr, host_p->h_addr, host_p->h_length);
    sysname = host_p->h_name;
  }

  else {
    fprintf(stderr, "qshc: Could not find host %s\n", hostname);
    exit(1);
  }
} /* End of if */

/* The user specified a IP address. */
else {
  /* Try to find the host by address. */
  host_addr.s_addr = ip_addr;
  host_p = gethostbyaddr((char *)&host_addr.s_addr, sizeof(host_addr),
                         AF_INET);
  if (host_p) {
    sysname = host_p->h_name;
  }

  else {
    fprintf(stderr, "qshc: Could not find host %s\n", hostname);
    exit(1);
  }
} /* End of else */

/*********************************************************************/
/* Connect to the qsh server on the specified system. */
/*********************************************************************/

/* Create a socket. */
if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP)) < 0) {
  perror("qshc: socket() failed");
  exit(1);
```

```
}

/* Connect to the qsh server on the specified system. */
memset(&svr_addr, '\0', sizeof(svr_addr));
svr_addr.sin_family = AF_INET;
svr_addr.sin_port = htons(port);
svr_addr.sin_addr.s_addr = ip_addr;
if (connect(sd, (struct sockaddr *)&svr_addr, sizeof(svr_addr)) != 0) {
  perror("qshc: connect() failed");
  exit(1);
}

/*********************************************************************/
/* Send the user name and password to the server. */
/*********************************************************************/

/* Allocate buffers for translating input and output. */
ascii_buf = (char *)malloc(DEFAULT_BUF);
memset(ascii_buf, '\0', DEFAULT_BUF);
ebcdic_buf = (char *)malloc(DEFAULT_BUF);
memset(ebcdic_buf, '\0', DEFAULT_BUF);

ascii_user = ascii_buf;
ascii_pwd = ascii_buf + 100;
ebcdic_user = ebcdic_buf;
ebcdic_pwd = ebcdic_buf + 100;

/* Prompt the user for the user name and password. */
if (nflag) {
  printf("Enter user name: ");
  gets(ascii_user);
  ascii_pwd = getpass("Enter password: ");
}

/* Get the user name and password from the ~/.netrc file. */
else {
  if (GetPassword(hostname, ascii_user, ascii_pwd) != 0) {
    fprintf(stderr, "qshc: Could not find user or password in ~/.netrc\n");
    exit(1);
  }
}

/* Convert the user name and password to EBCDIC. */
if (ConvertToEBCDIC(ascii_user, strlen(ascii_user)+1, ebcdic_user, 11) < 0) {
  fprintf(stderr, "qshc: Could not convert user %s to EBCDIC\n", ascii_user);
  exit(1);
}

if (ConvertToEBCDIC(ascii_pwd, strlen(ascii_pwd)+1, ebcdic_pwd, 11) < 0) {
  fprintf(stderr, "qshc: Could not convert password %s to EBCDIC\n",
          ascii_pwd);
  exit(1);
}

/* Send the user name and password to the qsh server. Note that the
   user name and password are sent as plain text. */
if ((rc = write(sd, (void *)ebcdic_user, strlen(ebcdic_user)+1)) < 0) {
  perror("qshc: write() failed sending username\n");
  close(sd);
  exit(1);
}

if ((rc = write(sd, (void *)ebcdic_pwd, strlen(ebcdic_pwd)+1)) < 0) {
  perror("qshc: write() failed sending password\n");
  close(sd);
  exit(1);
}
```

```
printf("Started qsh session on %s\n\n", sysname);

/*********************************************************************/
/* Process input and output between the user and the remote shell. */
/*********************************************************************/

/* Loop forever. */
while (1) {
  /* Select on stdin and the socket connected to the remote shell. */
  FD_ZERO(&read_set);
  FD_SET(0, &read_set);
  FD_SET(sd, &read_set);

  rc = select(sd+1, &read_set, NULL, NULL, NULL);

  if ((rc < 0) && (errno != EINTR)) {
    perror("qshc: select() failed");
    exit(1);
  }

  if (rc == 0) {
    continue;
  }

  /* Process data entered by the terminal user. */
  if (FD_ISSET(0, &read_set)) {
    /* Read the data from the terminal. */
    gets(ascii_buf);

    /* Convert the string to EBCDIC. */
    len = strlen(ascii_buf);
    if (ConvertToEBCDIC(ascii_buf, len, ebcdic_buf, DEFAULT_BUF) < 0) {
      fprintf(stderr, "qshc: Could not convert input string to EBCDIC\n");
      continue;
    }

    /* Put a newline on the end of the string. */
    *(ebcdic_buf+len) = 0x25;

    /* Send the data to the remote shell. */
    if (write(sd, ebcdic_buf, len+1) < 0) {
      perror("qshc: write() failed sending input");
    }
  }

  /* Process data from the remote shell. */
  if (FD_ISSET(sd, &read_set)) {
    /* Read the data from the remote shell. */
    buf_size = read(sd, ebcdic_buf, DEFAULT_BUF-1);

    /* There was a failure reading from the remote shell. */
    if (buf_size < 0) {
      perror("\nqshc: error reading data from remote shell");
      printf("Ended qsh session on %s\n", sysname);
      exit(0);
    }

    /* The remote shell process ended. */
    else if (buf_size == 0) {
      printf("\nEnded qsh session on %s\n", sysname);
      exit(0);
    }

    /* Process the data from the remote shell. */
    else {
      /* Convert to ASCII. */
      *(ebcdic_buf+buf_size) = '\0';
```

```
        if (ConvertToASCII(ebcdic_buf, buf_size+1, ascii_buf,
                           DEFAULT_BUF) >= 0) {
          write(1, ascii_buf, buf_size);
        }
       }
     }
  } /* End of while */

  exit(0);
} /* End of main() */


/*
 * Convert a string from ASCII to EBCDIC.
 */

int
ConvertToEBCDIC(char *ibuf, size_t ileft, char *obuf, size_t oleft)
{
  int rc;

  #ifdef USE_ICONV
  rc = iconv(ecd, (const char**)&ibuf, &ileft, &obuf, &oleft);
  #else
  rc = Translate((uchar *)ibuf, ileft, (uchar *)obuf, EbcdicTable);
  #endif
  if (rc < 0)
    perror("qshc: error converting to EBCDIC");

  return rc;
} /* End of ConvertToEBCDIC() */


/*
 * Convert a string from EBCDIC to ASCII.
 */

int
ConvertToASCII(char *ibuf, size_t ileft, char *obuf, size_t oleft)
{
  int rc;

  #ifdef USE_ICONV
  rc = iconv(acd, (const char**)&ibuf, &ileft, &obuf, &oleft);
  #else
  rc = Translate((uchar *)ibuf, ileft, (uchar *)obuf, AsciiTable);
  #endif
  if (rc < 0)
    perror("qshc: error converting to ASCII");

  return rc;
} /* End of ConvertToASCII() */


/*
 * Get the user name and password for the specified system from the
 * ~/.netrc file.
 */

int
GetPassword(char *sysname, char *logname, char *password)
{
  #define BUFSIZE 256
  char buffer[BUFSIZE];
  char *systag, *logtag;
  int logflag = 0, pwdflag = 0;
  FILE *netrc;
```

```
  struct passwd *pwdbuf;
  int rc=0;

  /* Get user's home directory. */
  pwdbuf = getpwuid(getuid());

  /* Does user have a .netrc file in their home directory? */
  strcat(strcpy(buffer, pwdbuf->pw_dir), "/.netrc");

  if ((netrc = fopen(buffer, "r")) == NULL) {
    perror("qshc: open() failed for ~/.netrc file");
    return -1;
  }

  while (!(logflag || pwdflag) && fgets(buffer, BUFSIZE, netrc) != NULL) {
    /* Find system name in ~/.netrc. */
    if ((systag = (char*)strtok(buffer, " \t\n")) != NULL &&
        !strncmp(systag, "machine", 7)) {
      systag = (char *)strtok(NULL, " \t\n");
      if (!strcmp(systag, sysname)) {
        /* Find login and password. */
        while (!logflag || !pwdflag) {
          if ((logtag = (char *)strtok(NULL, " \t\n")) == NULL) {
            /* Nothing else on that line... get another. */
            while (!logtag) {
              fgets(buffer, BUFSIZE, netrc);
              logtag = (char *)strtok(buffer, " \t\n");
            }
          }

          if (!strncmp(logtag, "login", 5)) {
            strcpy(logname, strtok(NULL, " \n\t"));
            ++logflag;
          }
          else if (!strncmp(logtag, "password", 8)) {
            strcpy(password, strtok(NULL, " \n\t"));
            ++pwdflag;
          }
          else
            ;
        } /* while flags not set */
      } /* if found login and passwd in .netrc */
    } /* if machine in .netrc */
  } /* while fgets */

  fclose(netrc);

  /* Login and password not found for system. */
  if (!(logflag && pwdflag)) {
    rc = -1;
  }

  return rc;
} /* End of GetPassword() */


#ifndef USE_ICONV
/*
 * Translate bytes using the specified translation table.
 */

int
Translate(uchar *ip, size_t ilen, uchar *op, uchar *table)
{
  int index;
  for (index = 0; index < ilen; ++index) {
    *op = table[*ip];
```

```
    ip++;
    op++;
  }

  return 0;
} /* End of Translate() */
#endif


/*
 * Signal handler.
 */

void
MySignalHandler(int signo)
{
  switch (signo) {
    case SIGINT:
      printf("\nqshc: <ctrl>c ends this program\n");
      printf("Ended qsh session on %s\n", sysname);
      exit(0);
      break;

    default:
      exit(1);
      break;
  } /* End of switch */

  return;
} /* End of MySignalHandler() */


/*
 * Display usage message.
 */

void usage(void)
{
  fprintf(stderr, "Usage: qshc [-n] [-p port] hostname\n");
  exit(1);
} /* End of usage() */
```

例: サーバー・プログラムを作成および実行する

サーバー・プログラムの作成

次の例は、i5/OS 上でサーバー・プログラムを作成する方法を示しています。

この例では、サーバー・プログラムのソースは、ファイル QGPL/QCSRC の中のメンバー SERVER 内にあることを前提とします。サーバー・プログラムの所有者は最小限の権限を持つ特殊ユーザー・プロファイル QSHSVR ですが、このユーザー・プロファイルには QSYGETPH()、QSYRLSPH()、および QWTSETP() に対する専用認可があります。 QSHSVR ユーザー・プロファイルを使ってサインオンすることはできません。サーバー・プログラムは QSHSVR の権限を借用しているので、クライアントのユーザー・プロファイルに切り替えることができます。

```
CRTBNDC   PGM(QGPL/SERVER)
          SRCFILE(QGPL/QCSRC)
          SRCMBR(SERVER)
          OPTIMIZE(40)
          SYSIFCOPT(*IFSIO)
          LOCALETYPE(*LOCALE)
          USRPRF(*OWNER)
          AUT(*USE)
          TEXT('Shell server')
```

```
CRTUSRPRF USRPRF(QSHSVR)
          PASSWORD(*NONE)
          USRCLS(*USER)
          TEXT('Shell server profile')
CHGOBJOWN OBJ(QGPL/SERVER)
          OBJTYPE(*PGM)
          NEWOWN(QSHSVR)
GRTOBJAUT OBJ(QSYS/QSYGETPH)
          OBJTYPE(*PGM)
          USER(QSHSVR)
          AUT(*USE)
GRTOBJAUT OBJ(QSYS/QSYRLSPH)
          OBJTYPE(*PGM)
          USER(QSHSVR)
          AUT(*USE)
GRTOBJAUT OBJ(QSYS/QWTSETP)
          OBJTYPE(*PGM)
          USER(QSHSVR)
          AUT(*USE)
```

サーバー・プログラムの実行

サーバー・プログラムと、サーバーが開始する子プロセスを、それぞれのサブシステム内で実行する必要がある場合があります。次の例は、下記のオブジェクトを作成する方法を示しています。

- 非スレッド・ジョブとマルチスレッド・ジョブの両方のためのサブシステムの記述および関連する経路指定項目および事前開始ジョブ項目。
- クラス。
- ジョブ記述。
- ジョブ待ち行列。

```
CRTSBSD   SBSD(QGPL/SHELL)
          POOLS((1 *BASE))
          AUT(*USE)
          TEXT('Shell server subsystem')
CRTCLS    CLS(QGPL/SHELL)
          RUNPTY(20)
          TIMESLICE(2000)
          DFTWAIT(30)
          AUT(*USE)
          TEXT('Shell server class')
CRTJOBQ   JOBQ(QGPL/SHELL)
          AUTCHK(*DTAAUT)
          AUT(*USE)
          TEXT('Shell server job queue')
CRTJOBD   JOBD(QGPL/SHELL)
          JOBQ(QGPL/SHELL)
          AUT(*USE)
          TEXT('Shell server job description')
ADDJOBQE  SBSD(QGPL/SHELL)
          JOBQ(QGPL/SHELL)
          MAXACT(*NOMAX)
ADDRTGE   SBSD(QGPL/SHELL)
          SEQNBR(1)
          CMPVAL(*ANY)
          PGM(*LIBL/QCMD)
ADDPJE    SBSD(QGPL/SHELL)
          PGM(QSYS/QP0ZSPWP)
          USER(QSHSVR)
          STRJOBS(*YES)
          INLJOBS(10)
          THRESHOLD(2)
          ADLJOBS(3)
          MAXJOBS(*NOMAX)
```

```
          JOBD(QGPL/SHELL)
ADDPJE    SBSD(QGPL/SHELL)
          PGM(QSYS/QP0ZSPWT)
          USER(QSHSVR)
          STRJOBS(*YES)
          INLJOBS(10)
          THRESHOLD(2)
          ADLJOBS(3)
          MAXJOBS(*NOMAX)
          JOBD(QSYS/QAMTJOBD)
```

サブシステムの開始

次の例は、上記の例に示したサブシステム、およびサーバー・プログラムを開始する方法を示しています。

```
STRSBS    SBSD(QGPL/QSHELL)
SBMJOB    CMD(CALL QGPL/SERVER)
          JOB(SERVER)
          JOBD(QGPL/SHELL)
          JOBQ(QGPL/SHELL)
          USER(QSHSVR)
```

例: クライアント・プログラムを作成および実行する

クライアント・プログラムの作成

次の例は、xlc を使って AIX 上でクライアント・プログラムを作成する方法を示しています。この例では、クライアント・プログラムのソースが現行作業ディレクトリーのファイル qshc.c の中にあることを前提にしています。 このクライアント・プログラムは、AIX 4.1.5 では xlc を使用し、 Linux 2.0.29 では gcc 2.7.2.1 を使用して、コンパイルされテストされています。

```
xlc -o qshc qshc.c
```

クライアント・プログラムの実行

次の例は、クライアント・プログラムを実行してシステム myas400 で実行されているサーバーに接続する方法を示しています。コマンドを実行するには、指定されたシステム用の項目が ~/.netrc ファイルの中に存在し、サーバーが開始されており TCP/IP ポート 6042 で listen していなければなりません。

```
qshc myas400
```

Qshell の関連情報

IBM Redbooks® 資料や Information Center のトピック・コレクションなどのソースには、Qshell のトピック・コレクションに関連した情報があります。いずれの PDF ファイルも表示または印刷することができます。

IBM Redbooks 資料

- Building AS/400® Internet-Based Applications with Java 

その他の情報

Information Center のトピック・コレクションは以下のとおりです。

- IBM Developer Kit for Java
- IBM Toolbox for Java

印刷資料は以下のとおりです。

- Qshell for iSeries®

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書「論理区画」には、プログラムを作成するユーザーが IBM i のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、『www.ibm.com/legal/copytrade.shtml』をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

IT Infrastructure Library は英国 Office of Government Commerce の一部である the Central Computer and Telecommunications Agency の登録商標です。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Intel Centrino、Intel Centrino ロゴ、Celeron、Intel Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

ITIL は英国 Office of Government Commerce の登録商標および共同体登録商標であって、米国特許商標庁にて登録されています。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Cell Broadband Engine は、Sony Computer Entertainment, Inc.の米国およびその他の国における商標であり、同社の許諾を受けて使用しています。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

プログラム番号: 5770-SS1

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21