Column

ヒアドキュメントとヒアストリングの実際

ヒアドキュメントやヒアストリングがシェル内部でどのように扱われているかはシェルの種類によって異なりますが、bashの場合はシェル内部で一時ファイルを作成することによって実現されています。

具体的には、ヒアドキュメントなどに書かれた文字列を内容とする一時ファイルが**/tmp**ディレクトリ以下に作成され、そのファイルを所定のファイル記述子(通常は標準入力の0番)でオープンした状態でコマンドが実行されます。この一時ファイルはオープン直後に削除されるため、**/tmp**ディレクトリを見ても一時ファイルを見ることはできません。UNIX系OSではファイルを削除しても、そのファイルをオープン中のプロセスが終了するまではファイル実体が存在し続けるため、実行されたコマンドからはファイル記述子を通してヒアドキュメントなどの内容を読むことができます。

Linuxの場合は、**図a**のようにヒアドキュメント等を使用しコマンド(ls)自身で**/dev/fd/0**(または**/proc/self/fd/0**)のシンボリックリンクのリンク先を見ることにより、ヒアドキュメント等に使用されている一時ファイルの正体を確認できます。「(deleted)」の表示から、一時ファイルは削除済みであることがわかります。

なお、一時ファイルが作成されるディレクトリ(**/tmp**)は、シェル変数**TMPDIR**を設定することにより変更できます(bashのバージョンによっては変更できません)。また、ディスクレスマシンなどで**/tmp**ディレクトリなどがNFS(*Network File System*)マウントされている場合、NFSの仕様により、削除された一時ファイルは「/tmp/.nfs005f80f300002852」のような名前にリネームされ、この場合はファイルとしても見えてしまいます。

図a ヒアドキュメントとヒアストリングの一時ファイルの正体を調べる

```
$ ls -l /dev/fd/0 << EOF                          ヒアドキュメントを実行
Hello World
EOF

lr-x------ 1 guest guest 64 Mar  5 11:38 /dev/fd/0 -> /tmp/sh-th
d-1299306069 (deleted)                            一時ファイルへのリンクと、
                                                  (deleted)の表示（実際は1行）

$ ls -l /dev/fd/0 <<< 'Hello World'               ヒアストリングを実行
lr-x------ 1 guest guest 64 Mar  5 11:40 /dev/fd/0 -> /tmp/sh-th
d-1299303718 (deleted)                            一時ファイルへのリンクと、
                                                  (deleted)の表示（実際は1行）
```


>第12章 よく使う外部コマンド

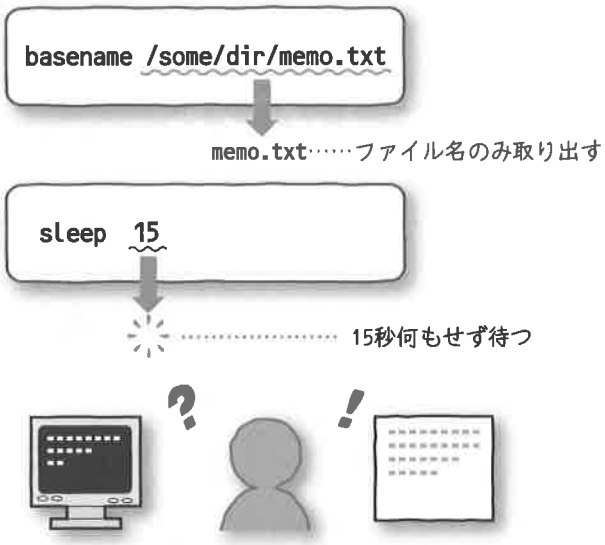
12.1 概要 260
12.2 シェルスクリプトならではのコマンド 261
12.3 一般コマンド 269

シェルスクリプトで使う外部コマンド

シェルスクリプト上に記述する外部コマンドも、コマンドラインにコマンド入力して実行するコマンドも、基本的には同じ外部コマンドです。したがって、どのようなコマンドでもシェルスクリプトに記述して実行することができます。

しかし、外部コマンドの中には、basenameやsleepのように、コマンドライン上で実行してもその意味がわからず、シェルスクリプトに記述するからこそ意味を持つコマンドが存在します(**図A**)。本章では、basenameやsleepなどコマンドをシェルスクリプトならではのコマンドとして解説し、さらにcp、mvなどの一般コマンドについても簡単に紹介します。

図A シェルスクリプトに記述するからこそ意味を持つ外部コマンド



コマンドライン上で実行してもその意味がわからず
シェルスクリプトに記述するからこそ意味を持つコマンドが存在する

expr

Linux (bash)
FreeBSD (sh)
Solaris (sh)

シェルスクリプトで数値計算を行う

書式 expr 式 ...

例
```
i=`expr "$i" + 1`  ……シェル変数iの値に1を加える
```

基本事項

exprコマンドは、与えられた式を評価し、その結果を標準出力に出力します。値や演算子はそれぞれ別個の引数として与える必要があります。演算子として、整数値の演算子である+(加算)、-(減算)、*(乗算)、/(除算)、%(剰余)などが使えます。

終了ステータス

exprコマンドの演算結果が「0」でも空文字列でもない場合、終了ステータスは「0」になります。

解説

シェルスクリプトでは、数値の入ったシェル変数に値を加算するなどの演算は、exprコマンドを呼び出すことによって行います。exprコマンドは、演算結果を標準出力に出力するため、冒頭の例のようにexprの出力をコマンド置換で取り込んでシェル変数に代入する使い方が一般的です。この時、exprの各引数は、数値と演算子との間をスペースで区切って、それぞれ別の引数として与えることに注意してください[注1]。

なおbashでは、(( ))を使った算術式の評価や$(( ))を使った算術式展開が使えるため、exprコマンドを使わなくても数値計算ができてしまいますが、移植性のためにはexprコマンドを使うのが基本です。

exprコマンドの使用例

exprコマンドの使用例を**図A**に示します。図を見ると、直接演算結果を表示する場合も、結果をコマンド置換で取り込んでシェル変数に代入する場合も、正しく計算が行われていることがわかります。なお、乗算の演算子の*は、パス名展開の*とみなされないように\でクォートして*とする必要があります。

注1 コマンド置換については9.3節を参照してください。

図A exprコマンドの使用例

```
$ expr 3 + 5                    3+5を計算
8                               答が表示される
$ expr 3 - 5                    3−5を計算
-2                              答が表示される
$ expr 3 \* 5                   3×5を計算（*はクォートして\*とする）
15                              答が表示される
$ expr 32 / 5                   32÷5を計算
6                               答が表示される（小数点以下は切り捨て）
$ expr 32 % 5                   32÷5の余りを計算
2                               答が表示される
$ i=3                           シェル変数iに3を代入
$ i=`expr "$i" + 1`             シェル変数iの値に1を加算（結果をコマンド置換で取り込み）
$ echo "$i"                     シェル変数iの値を表示
4                               たしかに4になっている
```

exprコマンドを使った文字列の演算

exprコマンドには、文字列の演算を行う「:」という演算子があり、**図B**のように文字列の一部を取り出すことができます。

「:」演算子は、:の左の項に書かれた文字列の先頭から、:の右の項に書かれた正規表現との一致を試み、一致した文字列のうち\(\)で囲まれた部分の文字列を標準出力に出力します。図Bでは、シェル変数**file**に代入されているファイル名に対して、その3〜5文字目のみを取り出すことと、拡張子のみを取り出すことを実行しています。

図B exprコマンドを使った文字列の演算

```
$ file='hello.txt'                  シェル変数fileにファイル名を代入
$ expr "$file" : '..\(...\)'        文字列の3〜5文字目のみを取り出す
llo                                 3〜5文字目が表示される
$ expr "$file" : '.*\(\..*\)'       ファイル名の拡張子部分を取り出す
.txt                                拡張子のみが表示される
```

参照

算術式の評価(( ))(p.80)　　算術式展開(p.232)

basename

Linux (bash)　FreeBSD (sh)　Solaris (sh)

ファイル名からディレクトリ名部分や拡張子を取り除く

書式 basename ファイル名 [拡張子]

例

```
name=`basename /some/dir/memo.txt`  ……ディレクトリ名を除いたmemo.txtのみnameに代入される
```

基本事項

basenameコマンドは、引数で指定されたファイル名から、先行するディレクトリ名部分を取り除き、/を含まないファイル名のみを標準出力に出力します。引数で拡張子が与えられた場合は、ファイル名からその拡張子が取り除かれます。

終了ステータス

basenameコマンドの終了ステータスは「0」になります。

解説

シェルスクリプト上で各種ファイル名を扱う際に、位置パラメータやシェル変数などに代入されているファイル名から、その先行ディレクトリ名を取り除いたり、拡張子を取り除いたりしたい場合があります。このような時、basenameコマンドを使います。basenameは、その結果のファイル名を標準出力に出力するため、冒頭の例のようにコマンド置換で取り込んで利用するのが普通です[注2]。

basenameの実行例

basenameコマンドの実行例を**図A**に示します。このように、先行ディレクトリ名の除去、拡張子の除去ができることがわかります。

図A basenameの実行例

```
$ basename /usr/local/bin/myprog        myprogの絶対パスを引数にすると
myprog                                  先行ディレクトリ名が除かれmyprogのみか表示される
$ basename picture.jpg .jpg             basenameの第2引数に.jpgを指定すると
picture                                 拡張子の.jpgを除いたファイル名のみか表示される
$ basename /some/dir/index.html .html   絶対パスのファイル名と拡張子を指定すると
index                                   先行ディレクトリ名と拡張子の両方が取り除かれ、
                                        indexのみか表示される
```

注2　コマンド置換については9.3節を参照してください。

basenameによる拡張子の変更

リストAは、カレントディレクトリ上にある拡張子**.txt**のファイルすべて(ただし.で始まるファイルを除く)を**.html**にリネームするシェルスクリプトです。ここでは、わかりやすいように、コマンド置換で取り込んだbasenameの結果をいったんシェル変数nameに代入してから、そのnameをmvコマンドの引数として利用しています。

リストA ファイルのリネーム

```
for file in *.txt ……カレントディレクトリ上の拡張子.txtのファイルについてループ
do ……ループの開始
  name=`basename "$file" .txt` ……ファイル名から.txtを除いたものをnameに代入
  mv "$name".txt "$name".html ……拡張子.txtを拡張子.htmlにリネーム
done ……ループの終了
```

Memo

- basenameとは逆に、先行ディレクトリ名のみを取り出すdirnameというコマンドがあります。
- bashの場合は、${name##*/}というパラメータ展開で、basenameと同様の処理を行うことができます。「${パラメータ#パターン}と${パラメータ##パターン}」(p.197)も合わせて参照してください。

参照

dirname(p.265)

dirname

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイル名から そのディレクトリ名部分のみを取り出す

書式 dirname [ファイル名]

例

```
name=`dirname /some/dir/memo.txt` ……basename部分を除いた/some/dirのみがnameに代入される
```

基本事項

dirnameコマンドは、引数で指定された[ファイル名]から、先行するディレクトリ名部分のみを標準出力に出力します。引数の[ファイル名]が/を含んでいない場合は、カレントディレクトリを表す「.」を出力します。

終了ステータス

dirnameコマンドの終了ステータスは「0」になります。

解説

dirnameコマンドは、basenameコマンドとは対照的に、先行するディレクトリ名部分を取り出します。これは、位置パラメータやシェル変数などに代入されている絶対パスのファイル名から、そのディレクトリ名部分を取り出したい場合に使います。dirnameは、その結果を標準出力に出力するため、冒頭の例のようにコマンド置換で取り込んで利用するのが普通です[注3]。

dirnameの実行例

dirnameコマンドの実行例を**図A**に示します。このように、先行ディレクトリ名が取り出せていることがわかります。

図A 先行ディレクトリ名の取り出し

```
$ dirname /usr/local/bin/myprog      myprogの絶対パスを引数にすると
/usr/local/bin                       先行ディレクトリ名のみが表示される
$ dirname picture.jpg                /を含まない、カレントディレクトリのファイル名の場合
.                                    カレントディレクトリを表す.が表示される
```

注3 コマンド置換については9.3節を参照してください。

dirnameによる拡張子の変更

リストAは、絶対パスで指定した複数のディレクトリ上にある拡張子**.txt**のファイルすべて(ただし.で始まるファイルを除く)を**.html**にリネームするシェルスクリプトです。ここで拡張子を取り除く処理にはbasenameコマンドを利用していますが、その際に先行ディレクトリ名も取り除かれてしまうため、先行ディレクトリ名を別途dirnameでシェル変数dirに取り込んでいます。dirの値はmvコマンドの引数上で利用し、これによって目的のリネームが行えます。

リストA ファイルのリネーム

```
for file in /some/dir/*.txt /other/dir/*.txt …複数ディレクトリの.txtファイルについてループ
do ……ループの開始
  dir=`dirname "$file"` ……ファイル名の先行ディレクトリ名をdirに代入
  name=`basename "$file" .txt` ……ファイル名から先行ディレクトリ名と.txtを除いたものをnameに代入
  mv "$dir"/"$name".txt "$dir"/"$name".html …拡張子.txtを拡張子.htmlにリネーム
done ……ループの終了
```

Memo

- bashの場合は${name%/*}というパラメータ展開で、dirnameに近い処理を行うことができます。「${パラメータ%パターン}と${パラメータ%%パターン}」(p.199)も合わせて参照してください。

参照

basename(p.263)

cat

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイルを連結し標準出力に出力する

書式 cat [ファイル ...]

例

```
var=`cat file` ……fileの内容をシェル変数varに取り込む
```

基本事項

catコマンドは、引数に指定した[ファイル名]を順に**連結**し、標準出力に出力します。引数に[ファイル名]を指定しなかった場合、またはファイル名が-の場合は、標準入力を読み込みます。

終了ステータス

指定のファイルが読み込めないなどのエラーが発生しないかぎり、終了ステータスは「0」になります。

解説

catコマンドは、本来は複数のファイルを連結するコマンドです。しかし、実際には1つのファイルを標準出力に出力するだけの目的で、つまり、単にファイルの内容を表示するために使われることが多いでしょう。シェルスクリプト上では、コマンド置換を使って、ファイルの内容をシェル変数に読み込む際にcatコマンドが利用されます[注4]。

ファイルの内容をシェル変数に読み込む

コマンド置換でcatコマンドの出力を取り込み、シェル変数に代入している例を**図A**に示します。シェル変数の値を表示すると、改行も含めてファイル内容がそのままシェル変数に取り込めていることがわかります。

図A ファイルの内容をシェル変数に読み込む

IPアドレスとホスト名を関連づける設定ファイル

```
$ var=`cat /etc/hosts`                          /etc/hostsの内容をシェル変数varに代入
$ echo "$var"                                   シェル変数varの内容を表示
127.0.0.1       localhost localhost.localdomain     /etc/hostsの内容が表示される
192.168.1.23    myhost myhost.localdomain
```

Memo

- bashでは、コマンド置換の際に`< file`または$(< file)と記述して、catコマンドを使わずにファイルの内容をシェル変数やコマンドの引数に取り込むことができます。

注4 コマンド置換については9.3節を参照してください。catコマンドには、行番号を付加する-nオプションほか、いくつかのファイル加工のためのオプションも存在します。

sleep

Linux (bash)
FreeBSD (sh)
Solaris (sh)

一定時間待つ

書式 sleep 秒数

例
```
sleep 15 ……15秒間待つ
```

基本事項

sleepコマンドは、引数に指定した秒数だけ待ってから、sleepコマンドを終了します。

終了ステータス

引数の指定に誤りがないかぎり、終了ステータスは「0」になります。

解説

sleepコマンドは、何もせずに単に一定時間だけ待つコマンドです。コマンド起動のタイミング待ちのためにsleepコマンドを挿入したり、ループ中で使用して一定時間ごとに何らかの動作を行うといった使い方ができます。

一定時間ごとにコマンドを実行

リストAは、while文のループ中にsleepコマンドを使い、1分おきにdfコマンドを実行して、ディスクの使用量を表示するようにしたシェルスクリプトです。sleepコマンドの引数の「60」の値を変えれば表示間隔を変更できます。

リストA 1分おきにディスク使用量をチェック

```
while : ……while文による無限ループ
do ……ループの開始
  df ……dfコマンドでディスク使用量を表示
  sleep 60 ……sleepコマンドで60秒間待つ
done ……ループの終了
```

Memo

- sleepコマンドのバージョンによっては、引数にs(秒)、m(分)、h(時間)、d(日)などの単位を付けたり、秒数などの数値に小数を使用して「sleep 0.1」(0.1秒待つ)のような指定が可能な場合があります。

参照

while文(p.63)

一般コマンド

Linux (bash)
FreeBSD (sh)
Solaris (sh)

よく使う一般コマンド

解説

そのほか、シェルスクリプトではさまざまなコマンドが使用されますが、中でも比較的よく使用され、Linux／FreeBSD／Solarisで共通して使用できるものを**表A**にまとめます。どれもUNIXの基本コマンドですが、詳しいコマンドの仕様やオプションなどについては、それぞれのオンラインマニュアルを参照してください。

続けて、表Aのコマンドからいくつか取り上げて、書式と使用例を紹介しておきます。

表A よく使う一般コマンド

コマンド	概要
ls	ファイルの一覧表示
cp	ファイルのコピー
mv	ファイルの移動またはリネーム
ln	ファイルのハードリンクまたはシンボリックリンク
rm	ファイルの削除
mkdir	ディレクトリの作成
rmdir	ディレクトリの削除
touch	ファイルの作成とタイムスタンプの更新
chmod	ファイルの属性(パーミッション)の変更
cmp	ファイルの比較
diff	ファイルの差分の抽出
date	日付と時刻の表示
wc	ファイルの行数、単語数、文字数の表示
head	ファイルの先頭部分の取り出し
tail	ファイルの末尾部分の取り出し
grep	ファイルから指定の文字列を含む行の抽出
sed	正規表現による文字列の置換など
awk	各種文字列処理ができるスクリプト言語
find	一定の条件のファイルの検索
xargs	標準入力を引数に取り込んでコマンドを起動

grep

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイルから特定の文字列を含む行を抜き出す

書式 **grep** [-オプション] パターン [ファイル ...]

grepコマンドは、引数で指定したファイルからパターンが含まれる行のみを取り出し、標準出力に出力します。**図A**は指定の文字列を含む行を取り出す例です。ファイルの指定を省略すると、標準入力からの入力になります。ファイルまたは標準入力中にパターンが含まれる行が存在した場合、grepコマンドの終了ステータスが「0」になるため、これをシェルスクリプトのif文などでの条件判断に利用できます。

図A resolv.confからnameserverを含む行を取り出す

DNSサーバの設定ファイル

```
$ grep nameserver /etc/resolv.conf          resolv.confのnameserverの行を取り出す
nameserver 192.168.10.1                     nameserverを含む行が表示される
```

wc

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイルの行数／単語数／ファイルサイズを表示する

書式 **wc** [-lwc] [ファイル ...]

wcコマンドを実行すると、引数で指定したファイルの行数、単語数、ファイルサイズを左から順に並べて標準出力に表示します。ファイルの指定を省略すると、標準入力からの入力になります。-l、-w、-cオプションを指定して、それぞれ行数、単語数、ファイルサイズのみを単独で表示させることもできます(**図B**)。

図B ファイルの行数、単語数、ファイルサイズを表示

```
$ wc memo.txt                    memo.txtのファイルの行数、単語数、ファイルサイズを表示させる
      66     915    6804 memo.txt     66行、915単語、6804バイトのファイルであることがわかる
$ wc -c memo.txt                 -cオプションを付けると
    6804 memo.txt                ファイルサイズのみが表示される
```


head

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイルの先頭から指定の行数だけを取り出す

書式 **head** [-行数] [ファイル ...]

headコマンドを実行すると指定したファイルの先頭から、引数で指定した行数だけを取り出し、標準出力に出力します(**図C**)。ファイルの指定を省略すると、標準入力からの入力になります。行数を省略した場合は10行を指定したものとみなされます。

図C ファイルの先頭から5行のみを取り出す

```
$ head -5 progressbar            progressbarというシェルスクリプトの先頭の5行を取り出す
#!/bin/sh                        1行目
                                 2行目
ECHO='echo -e'                   3行目
case `$ECHO` in                  4行目
  -e)                            5行目
```

tail

Linux (bash) / FreeBSD (sh) / Solaris (sh)

ファイルの末尾から指定の行数だけを取り出す

書式 **tail** [-行数] [ファイル ...]

tailコマンドを実行すると指定したファイルの末尾から引数で指定した行数だけを取り出し、標準出力に出力します(**図D**)。ファイルの指定を省略すると、標準入力からの入力になります。行数を省略した場合は10行を指定したものとみなされます。

図D ファイルの末尾の5行のみを取り出す

```
$ tail -5 progressbar            progressbarというシェルスクリプトの末尾の5行を取り出す
  print_bar "$i"                 ファイルの末尾から5行目
  i=`expr "$i" + 1`              ファイルの末尾から4行目
  sleep 1                        ファイルの末尾から3行目
done                             ファイルの末尾から2行目
echo                             ファイルの最終行
```

touch

Linux (bash)
FreeBSD (sh)
Solaris (sh)

ファイルの更新時刻を更新する

書式 **touch** [-オプション] ファイル ...

touchコマンドを実行すると、引数で指定したファイルの更新時刻(タイムスタンプ)が現在の時刻に更新されます(**図E**)。引数で指定したファイルが存在しない場合は、その名前の**サイズゼロ**のファイルが作成されます。

図E ファイル更新時刻を更新する

```
$ ls -l memo.txt                          lsコマンドでmemo.txtの更新時刻を表示
-rw-r-----   1 guest    user        13668 Jan 16 07:23 memo.txt
                                                    更新時刻が表示される
$ touch memo.txt                          touchコマンドでmemo.txtの更新時刻を更新
$ ls -l memo.txt                          再度lsコマンドでmemo.txtの更新時刻を表示
-rw-r-----   1 guest    user        13668 Jan 16 21:23 memo.txt
                                                    更新時刻が現在の時刻になる
$ date                                    念のため現在の時刻を表示
Sat Jan 16 21:23:32 JST 2038
```

sed

Linux (bash)
FreeBSD (sh)
Solaris (sh)

ファイルの中の文字列を置換する

書式 **sed** [-オプション] プログラム [ファイル ...]

sedコマンドは、引数で指定したファイルに対しプログラムで指定した置換などの処理を施し、その結果を標準出力に出力します(**図F**)。引数のファイルを省略した場合は標準入力からの入力になります。sedのプログラムの詳細についてはsedのオンラインマニュアルを参照してください。

図F 文字列を置換する

```
$ sed 's/January/February/g' old.txt > new.txt     old.txtの中のJanuaryという文字列を
                                                   Februaryに置換して、new.txtに書き込む
```


>第13章 配列

13.1 概要 274
13.2 配列への代入と参照 275
13.3 配列の一括代入と一括参照 277
13.4 bash以外のシェルで配列を使う方法 279