誰にでも書ける #! /bin/sh 講座
第 3 回
「3 回まわってワン」

安岡孝一

yasuoka : root さん、 root さん。
root : 何だい？
yasuoka : 今日はシェルでの繰り返しのやり方を、教えてくれるっていう約束だったじゃないですか。
root : ああ、そうだったね。じゃあまずは一つ、例をあげよう。ファイル名は bow だ。

```
#! /bin/sh
# "bow" Version 1.0

TIME=$1
while test $TIME -gt 0
do echo BOWWOW
   TIME=‘expr $TIME - 1‘
done

exit 0
```

（間）

yasuoka : あと、それからっと。

```
% ls -l （ぽこ）
total 2
-rw-r--r--  1 yasuoka       116 Mar  2 16:17 bow
-rwxr-xr-x  1 yasuoka       240 Mar  1 20:29 who
% chmod 755 bow （ぽこ）
% ls -l （ぽこ）
total 2
-rwxr-xr-x  1 yasuoka       116 Mar  2 16:17 bow
-rwxr-xr-x  1 yasuoka       240 Mar  1 20:29 who
% rehash （ぽこ）
% ■
```

できました。
root : じゃ、試しに bow 3 で動かしてごらん。
yasuoka : はい。

```
% bow 3 （ぽこ）
BOWWOW
BOWWOW
BOWWOW
% ■
```

root : どうだい？
yasuoka : はい、うまくいきました。

```
% bow 7 （ぽこ）
BOWWOW
BOWWOW
BOWWOW
BOWWOW
BOWWOW
BOWWOW
BOWWOW
% bow 0 （ぽこ）
% ■
```

どういう仕掛けなんですか？
root : うん、まずは while の基本的な使い方から教えよう。

while test 条件		while [条件]
do コマンド列	あるいは	do コマンド列
done		done

条件が真である間、コマンド列を繰り返し実行する。 do と done の前で必ず改行。条件については if に同じ。

root : 先判定なので、ループを 1 回も実行しないこともあるよ。あと、これとよく似た繰り返し文に until がある。

yasuoka ： 後判定なんですか？

root ： いや、こっちも先判定で、条件がひっくり返ってるだけだ。

```
until test 条件                          until [ 条件 ]
do コマンド列          あるいは          do コマンド列
done                                     done
```

条件が偽である間、コマンド列を繰り返し実行する。do と done の前で必ず改行。条件については if に同じ。

yasuoka ： えーと、7 行目の ‘は何ですか？

root ： これはバッククウォートといって、コマンドの標準出力を文字列として取り込むためにあるんだ。ついでだから、シェルにおける引用符について、ここでまとめておこう。

’ … ’	内部をそのまま文字列として返す。
" … "	内部を評価してから、文字列として返す。
‘ … ‘	内部をコマンド列として実行し、その標準出力を文字列として返す。ただし、改行コードは空白に置き換えられる。

yasuoka ： すると、中の expr というのはコマンドなんですね。これはどういうコマンドなんですか？

root ： 基本的には、算術演算をおこなうコマンドだよ。

```
expr 式
      式を計算した結果を標準出力に出力する。なお、計算結果が 0 のときエグ
      ジットステイタスは 1、それ以外のとき 0 となる。
式は以下の通り。
       数              数（整数のみ有効）
       式 + 式         2 式の和
       式 - 式         左式から右式を引いた差
       式 ’*’ 式       2 式の積
       式 / 式         左式を右式で割った商（小数部切り捨て）
       式 % 式         左式を右式で割った余り
       ’(’ 式 ’)’      優先度を変える
’*’ や / や % は、 + や - より優先される。
```

yasuoka ： それで TIME を 1 ずつ減らして、繰り返しを制御してるわけですね。

```
% bow -4 （ぽこ）
% ■
```

あれ？ 無限に続くかと思ったのに。

root ： 判定のところが -gt で -ne じゃないからね。-ne だったら、無限に続いちゃうところだろう。

yasuoka ： あ、そうか。

```
% bow wow （ぽこ）
% ■
```

これは？

root ： test のところの条件判定は、整数じゃないものに対しては、0 だとみなすからだよ。

yasuoka ： へーえ。

```
% bow （ぽこ）
bow: test: argument expected
% ■
```

これは？

root ： しまった、$1 がセットされてない時を考えてなかった。ちょっと待ってくれるかい？

```
#! /bin/sh
# "bow" Version 1.1

TIME=${1-3}
while test "$TIME" -gt 0
do echo BOWWOW
   TIME=‘expr $TIME - 1‘
done

exit 0
```

root ： これでどうかな？

yasuoka ： えーっと。

```
% bow （ぽこ）
BOWWOW
```

```
BOWWOW
BOWWOW
% 
```

なーかなか。

root : ま、俗に言うところの、デフォルトは 3 というやつだな。何なら、こういう荒技もあるぞ。

```
#! /bin/sh
# "bow" Version 1.2

if [ -z "$1" ]
then set 3
fi

TIME=$1
while [ $TIME -gt 0 ]
do echo BOWWOW
   TIME=‘expr $TIME - 1‘
done

exit 0
```

yasuoka : setって何でしたっけ？

root : もう忘れたのかい？ もう一度、前回の資料を読んでおくこと。

yasuoka : はい。

root : それから、シェルには while と until 以外にも、もう一つ繰り返しをおこなうためのコマンドがあるんだ。

```
for 変数 in 文字列 文字列 …
do コマンド列
done
```

変数に文字列を順々に代入し、文字列の個数だけコマンド列を実行する。do と done の前で必ず改行。

root : これを使った例を一つ示そう。ファイル名は where だ。

```
#! /bin/sh
# "where" Version 1.0

for DIR in ‘echo $PATH | tr ":" " "‘
do if [ -f $DIR/$1 ]
   then echo $DIR/$1
   fi
done

exit 0
```

（間）

yasuoka : あとはっと。

```
% chmod 755 where （ぽこ）
% rehash （ぽこ）
% 
```

できました。

root : よし、じゃあ where who とでも実行してごらん。

yasuoka : はい。

```
% where who （ぽこ）
/home/yasuoka/bin/who
/bin/who
./who
% 
```

root : おや？ ちょっとごめん。

```
% echo $PATH （ぽこ）
/home/yasuoka/bin:/usr/local/bin:/usr/ucb:/bin:/usr/bin:.
% ls （ぽこ）
bow     where   who
% pwd （ぽこ）
/home/yasuoka/bin
% 
```

ああ、そういうことか。

yasuoka : どういうことですか？

root ： この where コマンドの意味はわかるかい？
yasuoka ： ええっとたぶん、パラメータとして与えたコマンドがどこのディレクトリにあるのか、教えてくれるんでしょ？
root ： 大正解。しかもよくある which なんかと違う点は、あるやつ全部を優先度順に出力してくれるっていう点だ。
yasuoka ： で、何がおかしかったんですか？
root ： いや、いきなり./who と出たんで、びっくりしただけだよ。よく考えたら今~/bin にいるわけだから、ダブって出るのは当然なんだな。
yasuoka ： . を展開できたらさらにカッコいいわけですね。
root ： まあ、そうだけど。
yasuoka ： じゃ、ちょっと待っててくれます？

（間）

```
#! /bin/sh
# "where" Version 1.1

for DIR in `echo $PATH | tr ":" " "`
do if [ $DIR = "." ]
   then DIR=`pwd`
   fi
   if [ -f $DIR/$1 ]
   then echo $DIR/$1
   fi
done

exit 0
```

yasuoka ： こんなもんでどうです？
root ： おお、もうバッククウォートを使いこなせるようになったのか。やってみせてくれるかい？
yasuoka ： はい。

```
% where who （ぽこ）
/home/yasuoka/bin/who
/bin/who
/home/yasuoka/bin/who
% 
```

root ： 同じのが 2 つも出てるね。こういうのをオミットできると、完璧なんだがな。
yasuoka ： うーん、それはむずかしい。
root ： それに、 pwd よりは /bin/pwd を使った方がいい。
yasuoka ： え、どうしてですか？

pwd
　　カレントディレクトリを標準出力に出力する。
シェルの内蔵コマンドである pwd はシンボリックリンクなどに対応していないため、 /bin/pwd を用いるのが望ましい。

root ： ま、こんなところだ。
yasuoka ： そうなんですか。
root ： で、さっきの話題なんだけど、オミットできるやつを作ってみようか。
yasuoka ： はい。

```
#! /bin/sh
# "where" Version 2.0

FILE=$1
set ""

for DIR in `echo $PATH | tr ":" " "`
do if [ $DIR = "." ]
   then DIR=`/bin/pwd`
   fi
   if [ -f $DIR/$FILE ]
   then for CHECK in $*
        do if [ $DIR = $CHECK ]
           then continue 2
           fi
        done
```

```
        echo $DIR/$FILE
        set $* $DIR
    fi
done

exit 0
```

（間）

yasuoka ： できました。

```
% where who （ぽこ）
/home/yasuoka/bin/who
/bin/who
% 
```

なーかなか。

root ： ふっふっふ。

yasuoka ： どんな仕掛けなんですか？

root ： 与えられたコマンドが存在するディレクトリを順々にパラメータに覚えていって、それ以前に同じものがなかったかどうかチェックしてるんだ。

yasuoka ： はあ。

root ： まず与えられたコマンドを 4 行目で FILE って変数に入れて、次に 5 行目でパラメータを全てクリアする。

yasuoka ： 5 行目は単に set だけじゃいけないんですか？

root ： それだと、全ての変数を標準出力に出力するって意味になってしまうからね。 6 行目からは、 PATH に書かれている各ディレクトリについて、 DIR に代入して繰り返す。これはさっきと同じだね。

yasuoka ： ええ。

root ： 与えられたコマンドが DIR で示されるディレクトリにあったなら、それ以前に同じディレクトリを調べてないかどうかチェックする。

yasuoka ： 12 行目から始まる for が、そのチェック用のループですね。すると、この 14 行目の continue 2 ってのは何ですか？

root ： 20 行目の done のところにジャンプしてるんだよ。

```
continue
      do … done ループの done にジャンプする。ループを繰り返す条件が成立していれば、再度ループを繰り返す。
```

```
continue 数
      「数」段分外のループの done にジャンプする。ループを繰り返す条件が成立していれば、再度ループを繰り返す。
```

root ： これとよく似たコマンドに break がある。

```
break
      do … done ループから飛び出す。
break 数
      「数」段の do … done ループから飛び出す。
```

yasuoka ： すると、それ以前に同じものがなかった時だけ、 17 行目と 18 行目を実行するわけですね。

root ： そう。 17 行目は見つかったコマンドの表示、 18 行目は DIR の内容をパラメータに追加している。

yasuoka ： この continue や break ってのは、 while、 until、 for、どれにでも使えるんですか？

root ： うん、そうだよ。

yasuoka ： だいたいわかりました。

root ： よし。ああ、もうこんな時間だ。続きはまた今度ということにしてくれるかい？

yasuoka ： はい。どうもありがとうございました。