誰にでも書ける #! /bin/sh 講座
第 1 回
「who am i は alias でできない」

安岡孝一

yasuoka : root さん、 root さん。
root : 何だい？
yasuoka : 教えてほしいことがあるんですけど、時間ありますか？
root : ああいいよ。
yasuoka : 最近、皆んなの間で whoami を alias するのが流行ってますよね。
root : ほんと？
yasuoka : あれ、知らなかったんですか？ 僕はこんな風にしているんですけど。

```
% whoami （ぽこ）
You are YASUOKA.
% 
```

root : ほう、どうやってるのかな？

```
% alias whoami （ぽこ）
echo You are YASUOKA.
% 
```

なかなかやるね。で？
yasuoka : でも who am i は alias でできないんです。

```
% who am i （ぽこ）
ginkaku!yasuoka  console Feb 28 16:02
% 
```

何かいい方法はないでしょうか？
root : ようするに who am i と実行した時にも、 You are YASUOKA. と出力してほしいと、そういうわけだね？
yasuoka : そうです。
root : じゃあ、 who を alias したらどう？ whoami と同じように。
yasuoka : でもそれじゃあ、普通の who が使えなくなっちゃうでしょう？

```
% who （ぽこ）
yasuoka  console Feb 28 16:02
takahash ttyp0   Feb 28 13:12   (kinkaku:0.0)
% 
```

root : すると、 who のオプションが am i のときには You are YASUOKA. を出して、何もないときには普通に who してほしいと？
yasuoka : そういうわけです。
root : じゃあ、自分専用の who を作るしかないね。
yasuoka : えー、 C でですかー？
root : いやいや、そういう時のためにシェルがあるんだよ。
yasuoka : シェル？

> シェルは Unix システム上で動作するコマンド・インタプリタである。
> シェル用に書かれたプログラムをシェル・スクリプトという。

root : そう、シェル。
yasuoka : どうやって使うんですか？
root : まあ、そうあせらずに。ちょっとごめんね。

```
% cd （ぽこ）
% ls （ぽこ）
Mail    News
% 
```

うーん、まずは自分専用のコマンドを入れておくディレクトリを作らなきゃね。

```
% mkdir bin （ぽこ）
% ls （ぽこ）
Mail    News    bin
% 
```

yasuoka : その bin の中に、コマンドを入れるんですか？
root : うん、そう。それから、えーっと。

```
% echo $PATH （ぽこ）
/usr/local/bin:/usr/ucb:/bin:/usr/bin:.
% 
```

コマンドパスは.cshrc で設定しているのかい？

yasuoka : ええ。
root : じゃーっと。

```
% fgrep path .cshrc（ぽこ）
set path=(/usr/local/bin /usr/ucb /bin /usr/bin .)
% ■
```

yasuoka : その fgrepって何ですか？
root : ファイルの中から、ある文字列を含む行を探し出すコマンドだよ。

> fgrep 文字列 ファイル名
> 文字列を含む行を全て出力する。入力はファイル、出力は標準出力。ただしファイル名が省略された場合は、入力は標準入力。
> fgrep -v 文字列 ファイル名
> 文字列を含まない行を全て出力する。他は上に同じ。
> エグジットステイタスは、出力行があった時には 0、なかった時には 1 となる。

root : じゃエディタか何かで、.cshrc のこの行の (と /usr/local/bin との間に、~/bin を入れてくれるかい？
yasuoka : はい。

（間）

yasuoka : できました。
root : なかなかエディタは使い馴れてるみたいだね。

```
% fgrep path .cshrc（ぽこ）
set path=(~/bin /usr/local/bin /usr/ucb /bin /usr/bin .)
% ■
```

じゃあ、.cshrc を読み込んでっと。

```
% source .cshrc（ぽこ）
% echo $PATH（ぽこ）
/home/yasuoka/bin:/usr/local/bin:/usr/ucb:/bin:/usr/bin:.
% ■
```

OK、 OK、準備完了。
yasuoka : これで、どうなったんですか？
root : ~/bin の中にあるファイルをコマンドとして使えるようになったんだ。
yasuoka : やった。
root : さっそく~/bin の中にファイルを作ってくれるかい？ 名前は who だ。

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

if test "$1" = am
then if test "$2" = i
     then echo You are YASUOKA.
          exit 0
     fi
fi

/bin/who $*
exit 0
```

（間）

yasuoka : できました。
root : どれどれ。

```
% cd ~/bin（ぽこ）
% ls -l（ぽこ）
total 1
-rw-r--r--  1 yasuoka       145 Feb 28 16:19 who
% ■
```

このままじゃ、まだ実行できないな。

```
% chmod 755 who（ぽこ）
% ls -l（ぽこ）
total 1
-rwxr-xr-x  1 yasuoka       145 Feb 28 16:19 who
% ■
```

yasuoka : その chmodってのは何ですか？
root : ファイルのモードを変えるコマンドだよ。さっきの ls -l のいちばん最初に出てる -rwxr-xr-x とかいうのがあっただろ。あれの rwxr-xr-x の部分を 2 進数の 111101101 とみなして、それを 8 進数に変換した 755 を、1 つめのパラメータとするんだ。

chmod 3 桁の 8 進数 ファイル名
　　ファイルのモードを変える。ファイル名は複数書いてもよい。
モードの意味は以下の通り。
　　-rwxr-xr-x　　r は readable、 w は writable、 x は executable
　　　最初の桁は自分に対して、
　　　次の桁は自分と同じグループの人に対して、
　　　最後の桁は他の人に対して、それぞれ適用される。
　　なお、最初の - はファイルの種類を表し、 chmod とは無関係。

yasuoka : だいたいわかりました。
root : でも、まだこれだけじゃ、だめなんだな。

```
% rehash（ぽこ）
% 
```

yasuoka : それは？
root : 内部に持っているコマンドの一覧表を書き換えるんだ。コマンドを追加した時とか削除した時には必ず実行しなきゃならない。
yasuoka : 色々めんどくさいんですね。
root : 最初だけだよ。さて、実行してみようか。

```
% who am i（ぽこ）
You are YASUOKA.
% who（ぽこ）
yasuoka  console Feb 28 16:02
takahash ttyp0   Feb 28 13:12   (kinkaku:0.0)
% 
```

yasuoka : すごーい。
root : ふっふっふ。
yasuoka : どんな仕掛けなんですか？ よかったら説明して下さい。

```
% cat who（ぽこ）
#! /bin/sh
# "who" Version 1.0

if test "$1" = am
then if test "$2" = i
     then echo You are YASUOKA.
          exit 0
     fi
fi

/bin/who $*
exit 0
% 
```

root : いいよ。まず最初の #! /bin/sh だけど、これはこのファイルがシェル上で実行されるシェル・スクリプトだっていう宣言みたいなもんだ。ただし、これは BSD に限ったことで、 System V では最初の行には: とでも書いといた方がいい。
yasuoka : System V で #! /bin/sh と書いたらどうなるんですか？
root : C シェルのスクリプトと勘違いされてしまうね。文法が違うから、うまく実行されなくなる。
yasuoka : へーえ。
root : 次の # から始まってる行はコメント。 4 行目はちょっと説明がいるな。まずはシェルでの if 文の基本的な使い方から。

```
if test 条件                          if [ 条件 ]
then コマンド列          あるいは      then コマンド列
else コマンド列                       else コマンド列
fi                                    fi
```

　　else 節はなくてもよいが、 then、 else、 fi の前で必ず改行。
条件は以下の通り。

文字列 = 文字列	2 つの文字列が等しいとき真
文字列 != 文字列	2 つの文字列が等しくないとき真
-z 文字列	文字列がヌルのとき真
-n 文字列	文字列がヌルでないとき真
数 -eq 数	2 数が等しいとき真
数 -ne 数	2 数が等しくないとき真
数 -gt 数	左数が右数より大きいとき真
数 -ge 数	左数が右数以上のとき真
数 -lt 数	左数が右数未満のとき真

数 -le 数	左数が右数以下のとき真
-s ファイル名	ファイルが存在して、そのサイズが 0 でないとき真
-d ファイル名	ファイルが存在して、ディレクトリのとき真
-f ファイル名	ファイルが存在して、ディレクトリでないとき真
-r ファイル名	ファイルが存在して、 readable のとき真
-w ファイル名	ファイルが存在して、 writable のとき真
-t 0	標準入力が端末のとき真
-t 1	標準出力が端末のとき真
-t 2	エラー出力が端末のとき真
! 条件	条件が偽のとき真（-a や -o より優先）
条件 -a 条件	2 条件とも真のとき真（-o より優先）
条件 -o 条件	2 条件のどちらかが真のとき真
'(' 条件 ')'	条件が真のとき真、優先度を変える

yasuoka ： たいていのことはできるんですね。すると 4 行目の if は "$1"と am とが等しいとき真になるわけですか。この "$1"ってのがコマンドの第 1 パラメータなんですか？

root ： まあそうだけど。ついでだから、シェルでの基本的な変数の読み出しを教えてあげよう。

$0	実行コマンド
$1	第 1 パラメータ
$2	第 2 パラメータ
以下、「$パラメータ番号」で $9 まで使用可能。	
$*	全パラメータ、ただし "$*"は "$1 $2 … "と展開
$@	全パラメータ、ただし "$@"は "$1" "$2" …と展開
$#	パラメータ数
$$	プロセス番号
$?	直前に実行したコマンドからのエグジットステイタス
$HOME	ホームディレクトリ
$PATH	コマンドパス
$USER	ユーザ ID （一般に BSD）
$LOGNAME	ユーザ ID （一般に System V）

yasuoka ： どうして 4 行目では、 $1 じゃなくて "$1"なんですか？

root ： じゃあ、試しに $1 にしてごらん。

yasuoka ： はい。

```
% cp who who~ （ぽこ）
% 
```

（間）

yasuoka ： できました。

```
% cat who （ぽこ）
#! /bin/sh
# "who" Version 1.0

if test $1 = am
then if test $2 = i
     then echo You are YASUOKA.
          exit 0
     fi
fi

/bin/who $*
exit 0
% who am i （ぽこ）
You are YASUOKA.
% 
```

別に大丈夫みたいですけど？

root ： そうかな？

```
% who （ぽこ）
who: test: argument expected
% 
```

yasuoka ： あら？ どうしてですか？

root ： つまり $1 がヌルだったら、 4 行目は

```
if test = am
```

っていう風に評価されてしまうんだよ。それで左文字列がないと思われてエラーになっちゃうんだ。でも元のやつなら

```
 if test "" = am
```

っていう風に評価されるから、エラーにはならない。

yasuoka ： そうか。

```
 % mv who~ who （ぽこ）
 % 
```

ところで、パラメータは 9 個までしか使えないんですか？

root ： いやいや、そんなことはないよ。

yasuoka ： じゃ、どうやって 10 番目のパラメータを読み出すんですか？

root ： shift を使うんだ。

```
shift
      $2 $3 $4 …を $1 $2 $3 …に移す。このとき、 $* や $@ や $# も同時に
      書き換えられる。
```

yasuoka ： shift を実行すると、第 10 パラメータが $9 に入るわけですね。

root ： そう。それを繰り返せば、第 11 パラメータも第 12 パラメータも読み出せる。

yasuoka ： なかなかうまくできてるんですね。ところで、元のプログラムに戻りますけど、 7 行目と最後の exitってのは、終了っていう意味ですか？

root ： うん、そうだよ。

yasuoka ： その後ろについてる数字は何ですか？

root ： ああ、これはエグジットステイタスといって、このプログラムを呼び出したプロセスに返す値だよ。普通は 0、エラー終了の時は 1 というのが一般的だな。

```
exit 数
      プログラムを終了し、エグジットステイタスを返す。数が省略された時には、 $? をエグジットステイタスとして返す。
```

yasuoka ： $?って、何でしたっけ？

root ： 直前に実行したコマンドからのエグジットステイタスだよ。

yasuoka ： あ、そうでしたね。で、下から 2 行目は /bin/whoっと。ここは who じゃいけないんですか？

root ： who だと、これ自身を呼び出してしまうからね。

yasuoka ： あ、そうですね。

root ： ついでだから System V 版も見せたげようか？

```
:
# "who" Version 1.0 for System V

if test "$1" = am
then if test "$2" = i
     then echo You are YASUOKA.
          exit 0
     fi
fi

/bin/who $*
exit 0
```

root ： 最初以外は全然変わりないな。

yasuoka ： その最初の:ってのは、実際にはどういう意味なんですか？

root ： :も実はシェルのコマンドの一種で、何もしない、っていう意味だよ。

```
:
      何もしない。
```

yasuoka ： 何もしないってコマンドが、何かの役に立つんですか？

root ： まあね。例えば if 文で、 then の方は何もしなくって else の方だけ何かしたい時とかね。

yasuoka ： ええ？ だったら条件で！を使えばいいじゃないですか。

root ： いやまあ、そうもいかないときもあるんだよ。くわしくはまた第 4 回あたりで話すから。

yasuoka ： そうですか。ところでもしかしたら、この who の条件の部分は -a を使って書き直せるんじゃないですか？

root ： よくわかったね。今日はもう時間がないから、それは次回までの宿題ということにしよう。

yasuoka ： はい。どうもありがとうございました。