誰にでも使える csh 講座
第 3 回
「昔のことは忘れた」

安岡孝一

yasuoka ： root さん、 root さん。
root ： 何だい？
yasuoka ： 前に `#! /bin/sh` で書いた `users`ってコマンドを

```
% cat users~ （ぽこ）
#! /bin/sh
# "users" Version 2.0

who |
( while read LINE
  do set $LINE
     echo $1
  done
) | sort | uniq | pr -t -8 -w80 -l1

exit 0
% ■
```

頑張って `#! /bin/csh` で書き直したんですけど。

```
% cat users （ぽこ）
#! /bin/csh
# "users" Version 3.0

who |
( set LINE=($<)
  while ($#LINE)
    echo $LINE[1]
    set LINE=($<)
  end
) | sort | uniq | pr -t -8 -w80 -l1

exit (0)
% ■
```

root ： ふーん。で？
yasuoka ： でも全然動かないんです。

```
% users （ぽこ）
Invalid null command.
% ■
```

どうしてでしょう？
root ： そりゃ、だめだろうな。 4 行目の `who` の後のパイプが、どこにも繋がってないからね。
yasuoka ： え？ 次の行の `( set LINE=($<)` に繋がるんじゃないですか？
root ： シェルならそうだけど。 C シェルは行の概念が厳しいから、パイプの後のコマンドは同じ行に書かなきゃならない。
yasuoka ： そうなんですか。
root ： でも、そこを直しても、まだ動かないだろうな。
yasuoka ： まだ間違ってるんですか？
root ： うん。 `( )` を使ってサブ C シェルを立ち上げてるみたいだけど、 C シェルでは `(と)` は同じ行に書かなきゃならない。
yasuoka ： そうなんですか。
root ： でも、 `while` の後とか、 `end` の前後には必ず改行が要る。
yasuoka ： えー、どうすればいいんですか？
root ： ようするに、サブ C シェルの中には `while` は書けないってことだよ。
yasuoka ： 困ったなぁ。何とかなりませんか？
root ： うーん、 `who` の出力からユーザ名を切り出してくるところか。 `colrm` でも使うしかないな。
yasuoka ： `colrm`？

```
#! /bin/csh
# "users" Version 3.1

who | colrm 9 | sort | uniq | pr -t -8 -w80 -l1
exit (0)
```

（間）

root ： ここまで短くなっちゃうと、C シェルでもシェルでも同じだな。
yasuoka ： うーん。

```
% users （ぽこ）
takenaga  yasuoka
% who （ぽこ）
yasuoka  console Mar  7 16:01
takenaga ttyp0   Mar  7 13:58   (kinkaku)
% who | colrm 9 （ぽこ）
yasuoka
takenaga
% 
```

こんな便利なコマンドがあるのに、どうして今まで教えてくれなかったんですか？

`colrm 数 数`
: BSD のみ。各行の左「数」文字目から右「数」文字目までを削除する。右「数」が省略された場合は、左「数」文字目以降を削除する。入力は標準入力、出力は標準出力。

`cut -c数-数 ファイル名`
: System V のみ。各行の左「数」文字目から右「数」文字目を残し、他の部分を削除する。右「数」が省略された場合は、左「数」文字目以降を残し、他の部分を削除する。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。

root ： BSD と System V でコマンドがひっくり返ってるからね。
yasuoka ： ひっくり返ってるって？
root ： これの System V 版を見ればわかるよ。

```
#! /bin/csh
# "users" Version 3.1 for System V

who | cut -c1-8 | sort | uniq | pr -t -8 -w80 -l1
exit (0)
```

yasuoka ： `colrm 9` のところが `cut -c1-8` になってますね。
root ： だろ。まあ 1 文字目から 8 文字目を残す分には、これでいいけどね。

yasuoka ： 10 文字目から 19 文字目を残すことって、できます？
root ： System V なら `cut -c10-19`。BSD なら `colrm 20 | colrm 1 9` かな。
yasuoka ： 逆に 10 文字目から 19 文字目を削除するには？
root ： BSD なら `colrm 10 19`。System V なら `cut -c1-9,20-` かな。`cut` の方はコンマを使って複数の範囲を書けるからね、何とか同じにできる。
yasuoka ： そうですか。
root ： でも行の中にタブが入ってると、ちょっと動作が違ってくるかな。`colrm` はタブを複数の空白に数えるけど、`cut` はタブを 1 文字だと思っちゃうみたいだから。
yasuoka ： タブを `expand` で空白に展開しておけばいいんでしょう？
root ： System V に `expand` はないよ。

`expand ファイル名`
: BSD のみ。タブを必要な数の空白に置き換える。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。

`newform -i ファイル名`
: System V のみ。タブを必要な数の空白に置き換える。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。

yasuoka ： すると、もし `who` の出力にタブが使われてたとしたら、この `users` はうまく動かないんですか？
root ： ありうるね。
yasuoka ： 入力の各行をパラメータとみなして、その第 1 パラメータを取り出してくるっていうコマンドはないんですか？
root ： あるにはあるんだけど…。

`join -o 1.数 -a1 ファイル名 /dev/null`
: 各行の第「数」フィールドを取り出す。「1.数」は複数書いてもよい。入力はファイル、出力は標準出力。ただしファイル名として - が指定された場合は、入力は標準入力。

yasuoka ： `join` ですか。

```
% who （ぽこ）
yasuoka  console Mar  7 16:01
takenaga ttyp0   Mar  7 13:58   (kinkaku)
% who | join -o 1.1 -a1 - /dev/null （ぽこ）
yasuoka
```

```
takenaga
% who | join -o 1.2 -a1 - /dev/null （ぽこ）
console
ttyp0
% who | join -o 1.2 1.5 1.1 -a1 - /dev/null （ぽこ）
console 16:01 yasuoka
ttyp0 13:58 takenaga
% 
```

お、これはこれは。でも変なコマンドですね。

root ： 元々こういうことに使うためのものじゃないからね。

```
join -o フィールド指定 ファイル名 ファイル名
      2 つのファイルの第 1 フィールドが一致した行を、フィールド指定に従って
      標準出力に出力する。ファイルはそれぞれ sort されていなければならな
      い。前のファイル名として - が指定された場合は、標準入力が用いられる。
フィールド指定は以下の通り。複数書いてもよい。
       1.数    前のファイルの第「数」フィールド
       2.数    後のファイルの第「数」フィールド
```

root ： ところで yasuoka くん。

yasuoka ： 何ですか？

root ： 今日は C シェルの history を、みっちりやるはずじゃなかったかい？

yasuoka ： 昔のことは忘れました。

root ： そうそう、忘れないために history があるんだよ。

```
% history （ぽこ）
    28  users
    29  vi users
    30  users
    31  who
    32  who | colrm 9
    33  who
    34  who | join -o 1.1 -a1 - /dev/null
    35  who | join -o 1.2 -a1 - /dev/null
    36  who | join -o 1.2 1.5 1.1 -a1 - /dev/null
    37  history
% 
```

まずは基本的な history の使い方から。

!!	直前のコマンド
!文字列	文字列から始まる最近のコマンド
!?文字列?	文字列を含む最近のコマンド
!数	「数」番目のコマンド
!-数	「数」回前のコマンド

yasuoka ： ええ、これぐらいは知ってます。

```
% !-2 （ぽこ）
who | join -o 1.2 1.5 1.1 -a1 - /dev/null
console 16:01 yasuoka
ttyp0 13:58 takenaga
% 
```

root ： じゃあ、 history の前後に他の文字列が書けるのは知ってるかい？

yasuoka ： いえ、それは知りませんでした。

root ： 例えばね、こんな風にパイプを伸ばしていったりできる。

```
% !! | fgrep yasuoka （ぽこ）
who | join -o 1.2 1.5 1.1 -a1 - /dev/null | fgrep yasuoka
console 16:01 yasuoka
% 
```

yasuoka ： ははーん。面白いですね。

root ： それから、コマンドの一部を取り出すこともできる。

ヒストリ:数	第「数」パラメータ
ヒストリ:数-数	左「数」パラメータから右「数」パラメータまで
ヒストリ:数-	左「数」パラメータから最後の 1 つ前のパラメータまで
ヒストリ:*	第 1 パラメータから最後のパラメータまで

いずれも数に $ を用いた場合、最後のパラメータを意味する。

root ： 例えば

```
% !36 （ぽこ）
who | join -o 1.2 1.5 1.1 -a1 - /dev/null
console 16:01 yasuoka
ttyp0 13:58 takenaga
% 
```

の時刻のところだけやめようと思ったら

```
% !36:0-4 !36:6-$ （ぽこ）
who | join -o 1.2 1.1 -a1 - /dev/null
console yasuoka
ttyp0 takenaga
% 
```

てな風にすればいい。

yasuoka ： 1.5 を切りとったんですね。

root ： うん。ただ注意しなきゃいけないのは、パラメータが 0 から始まることと、パイプなんかも 1 パラメータとして数えられてしまうことだ。

yasuoka ： はい。

root ： でも、 1.5 を削り取るにはもっと簡単な方法があるんだよ。

```
% !36:s/1.5// （ぽこ）
who | join -o 1.2 1.1 -a1 - /dev/null
console yasuoka
ttyp0 takenaga
% 
```

yasuoka ： それ、何ですか？

ヒストリ:s/文字列/文字列/	左文字列を右文字列に置換する
ヒストリ:t	最後の / 以前を削除する
ヒストリ:r	最後の . 以後を削除する
ヒストリ:h	最後の / 以後を削除する
ヒストリ:e	最後の . 以前を削除する
ヒストリ:p	表示をおこなうだけで実行しない
ヒストリ:gs/文字列/文字列/	全パラメータで左文字列を右文字列に置換する

いずれもヒストリはパラメータ指定を含んでいてもよい。

root ： 置き換えだよ。例えば

```
% head -2 ~/bin/users （ぽこ）
#! /bin/csh
# "users" Version 3.1
% 
```

とした後で「あっしまった、 head じゃなく tail だった」と思ったなら

```
% !!:s/head/tail/ （ぽこ）
tail -2 ~/bin/users
who | colrm 9 | sort | uniq | pr -t -8 -w80 -l1
exit (0)
% 
```

で済む。

yasuoka ： へえ。

root ： 区切りの / の代わりに? や^ を使ってもかまわないよ。

```
% !!:s^-2^-1^ （ぽこ）
tail -1 ~/bin/users
exit (0)
% 
```

yasuoka ： 便利ですね。

root ： 便利だろ。それから:gs だけど、これは例えば

```
% ls （ぽこ）
bow     dow     hb~     users~  where~  who~
bow~    hb      users   where   who
% mv bow~ bow.old （ぽこ）
% 
```

とした後で、さらに hb~ についても同じように

```
% !!:gs/bow/hb/ （ぽこ）
mv hb~ hb.old
% 
```

としたい時なんかに便利だね。

```
% ls （ぽこ）
bow     dow     hb.old  users~  where~  who~
bow.old hb      users   where   who
% mv !m:2 !m:1 （ぽこ）
mv hb.old hb~
% !!:gs/hb/bow/ （ぽこ）
mv bow.old bow~
% ls （ぽこ）
bow     dow     hb~     users~  where~  who~
```

```
bow~    hb      users   where   who
% 
```

yasuoka : この:h なんかはどうやって使うんですか？

root : いちばんよくやるのが

```
% cat /home/yasuoka/bin/users （ぽこ）
#! /bin/csh
# "users" Version 3.1

who | colrm 9 | sort | uniq | pr -t -8 -w80 -l1
exit (0)
% 
```

とかした後に「あ、この users のあるディレクトリに行きたいな」と思った時。

```
% cd !!:1:h （ぽこ）
cd /home/yasuoka/bin
% 
```

こーんなに簡単。

yasuoka : すごーい。

root : それから、 history の短縮形をいくつか覚えておくと便利だろうな。

^文字列^文字列^	「!!:s^文字列^文字列^」の短縮形、 直前のコマンドの左文字列を右文字列に置き換える
!*	「!!:*」の短縮形、 直前のコマンドの第 1 パラメータから最後のパラメータまで
!$	「!!:$」の短縮形、直前のコマンドの最後のパラメータ
!:	「!!:」の短縮形

root : 例えばさっきやった

```
% !t:p （ぽこ）
tail -1 ~/bin/users
% 
```

の -1 を -2 に変えるのでも

```
% ^-1^-2^ （ぽこ）
tail -2 ~/bin/users
who | colrm 9 | sort | uniq | pr -t -8 -w80 -l1
exit (0)
% 
```

と短縮できる。

yasuoka : すみません、:pって何でしたっけ？

root : 表示するだけで、コマンドを実行しないってやつだよ。実はヒストリの:s だの:p だのは組み合わせて使えるから、置き換えとかで自信がない時は、:p で一回表示してから実行するってのも手だ。

yasuoka : へーえ。

root : それから、どこまでが history だかわかりにくいと思った時には、 ! の後を { } でくくるといい。

yasuoka : どうするんですか？

root : 例えば

```
% !{t:s/-2/-3/}~ （ぽこ）
tail -3 ~/bin/users~
) | sort | uniq | pr -t -8 -w80 -l1

exit 0
% 
```

とかね。さあて、調子にのって話してたらもうこんな時間か。 history については大体こんなところだけどね。

yasuoka : すごく勉強になりました。

root : まあ、コマンドを打つ時にできるだけ history を使うようにすれば、自然に身体が覚えると思うよ。

yasuoka : はい。どうもありがとうございました。