誰にでも書ける #! /bin/sed -f 講座
第 3 回
「半分のフィルタ」

安岡孝一

root ： yasuoka くん、 yasuoka くん。
yasuoka ： 何ですか？
root ： 改行コードをバッファの区切りだとみなす命令。
yasuoka ： あ、そうでしたね。
root ： まずは 1 つ例をあげよう。ファイル名は headhf だ。

```
#! /bin/sed -f
N
P
N
D
```

yasuoka ： 知らない命令ばっかりですね。どういうコマンドなんですか？
root ： 入力の前半分だけを取り出すコマンドだよ。もうできたかい？
yasuoka ： はい。
root ： じゃ、ちょっとごめんね。

```
/home/yasuoka/bin% ls （ぽこ）
bow     dow     headhf  rev     users   who
delcom  hb      number  tac     where   yasuoka
/home/yasuoka/bin% number bow （ぽこ）
     1  #! /bin/csh
     2  # "bow" Version 2.0
     3
     4  if ($#argv) then
     5    set TIME=$argv[1]
     6  else
     7    set TIME=3
     8  endif
     9
    10  while ($TIME > 0)
    11    echo BOWWOW
    12    @ TIME--
    13  end
    14
    15  exit (0)
/home/yasuoka/bin%
```

まだあるな。試しに number bow | headhf を実行してみてごらん。
yasuoka ： はい。

```
/home/yasuoka/bin% number bow | headhf （ぽこ）
     1  #! /bin/csh
     2  # "bow" Version 2.0
     3
     4  if ($#argv) then
     5    set TIME=$argv[1]
     6  else
     7    set TIME=3
/home/yasuoka/bin%
```

行数が奇数だと、 1 行少なくなるんですね。どういう仕掛けなんですか？
root ： そうだな。まずは N を説明しておこうか。

N
　バッファの末尾に改行コードを付加し、さらにその後に入力の次の行を付加する。ただし入力に次の行がないときには、実行を終了する。

root ： N は入力の次の行を読み込んでバッファに付加するけど、スクリプトの最初にはジャンプしない。ただし d と同じく、現在入力中の行番号は 1 進むから、注意するようにね。
yasuoka ： はい。
root ： それから P を教えとこう。

P
　バッファの先頭から最初の改行コードまでを、標準出力に出力する。もしバッファ内に改行コードがないときには、バッファの全内容に改行コードを付加して出力する。

root : P は、バッファに改行コードがないときには、 p を実行する、という風に覚えておくといい。次の D も同じで、バッファに改行コードがないときには、 d を実行する。

D
バッファの先頭から最初の改行コードまでを削除した後、スクリプトの最初にジャンプする。もしバッファ内に改行コードがないときには、入力の次の行をバッファに読み込み、スクリプトの最初にジャンプする。ただしその際、入力に次の行がないときには、実行を終了する。

root : ま、こんなところかな。

yasuoka : ふーん。

```
/home/yasuoka/bin% cat headhf （ぽこ）
#! /bin/sed -f
N
P
N
D
/home/yasuoka/bin%
```

でも、実際にどんな風に動くのかがわかりません。

root : そうか。じゃ

```
/home/yasuoka/bin% headhf headhf （ぽこ）
#! /bin/sed -f
N
/home/yasuoka/bin%
```

を例にして、説明しようか。

yasuoka : はい。

root : まず「#! /bin/sed -f」をバッファに読み込んで、実行開始。最初の N でバッファは「#! /bin/sed -f改N」になる。そして次の P でバッファの先頭から最初の改行コードまで、つまり

```
#! /bin/sed -f
```

が出力される。

yasuoka : これで入力の 1 行目が出力されたわけですね。

root : うん。さらに次の N でバッファは「#! /bin/sed -f改N改P」に、そして D で「N改P」になって、スクリプトの最初にジャンプする。

yasuoka : その D では、入力の次の行は読み込まれないんですか？

root : うん、 D はバッファ内に改行コードがあるときには、入力を読み込んだりはしない。

yasuoka : わかりました。

root : で、最初に戻って、 N でバッファは「N改P改N」になる。次の P で出力されるのは、バッファの先頭の

```
N
```

だ。これで入力の 2 行目が出力されたことになる。

yasuoka : はい。

root : N でバッファは「N改P改N改D」、 D で「P改N改D」になって、スクリプトの最初に戻る。

yasuoka : あれ、出力はもう全部済んでるんじゃないですか？

root : ま、そうあせらずに。次の N でもう入力行がないから、実行が終了する。これで終わり。

yasuoka : ふーん。

root : この headhf とは逆に、入力の後半分を出力するのが、次の tailhf だ。

```
#! /bin/sed -f
$q
N
$!N
D
```

yasuoka : qって何ですか？

root : バッファを出力して、実行終了。

q
バッファの内容に改行コードを付加して、標準出力に出力した後、実行を終了する。

root : もうできたかい？

yasuoka : はい。

root : じゃ、 yasuoka くん

```
/home/yasuoka/bin% tailhf tailhf （ぽこ）
N
$!N
```

```
D
/home/yasuoka/bin% ■
```

の動作を例にして、この tailhf を説明してくれるかい？

yasuoka ： はい。まず 1 行目の「#! /bin/sed -f」をバッファに読み込んで、実行開始。 $q は実行されなくて、 N でバッファは「#! /bin/sed -f改$q」、さらに $!N で「#! /bin/sed -f改$q改N」。 D で「$q改N」になって、スクリプトの最初にジャンプします。

root ： うん。

yasuoka ： また $q は実行されなくて、 N が 2 回でバッファは「$q改N改$!N改D」、さらに D で「N改$!N改D」になって、スクリプトの最初にジャンプします。

root ： OK。

yasuoka ： そして $q が、えーともう入力は最終行だから、実行されて

```
N
$!N
D
```

が出力されます。これで終わり。

root ： よくできました。

yasuoka ： 他に sed で覚えておくべき命令はないんですか？

root ： そうだな。 n は要るな。

> n
>
> バッファの内容に改行コードを付加して標準出力に出力した後、入力の次の行をバッファに読み込む。ただし入力に次の行がないときには、バッファの出力の後、実行を終了する。

yasuoka ： p と d を一緒にしたような命令ですね。

root ： まあね。でもスクリプトの最初にジャンプしたりはしないよ。

yasuoka ： あ、そうなんですか。この n を使った例で、何かありませんか？

root ： そうだなぁ。 oddhf なんかどうかな。

```
#! /bin/sed -f
n
d
```

yasuoka ： どういうコマンドなんですか？

root ： 当ててごらん。

yasuoka ： んーと。まず 1 行目を n で出力して 2 行目を読み込み、 d で 3 行目を読み込んでまた n、 4 行目は d で消されて…、わかった、入力の奇数行目だけを出力するんだ。

root ： 大当たりー。

yasuoka ： root さん、最近妙に明るくないですか？

root ： そうかな。じゃ、これの逆で、偶数行目だけを出力する evenhf を作ってごらん。

yasuoka ： はい。

（間）

yasuoka ： できました。

```
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/sed -f
1d
n
d
/home/yasuoka/bin% ■
```

こんなもんでどうです？

root ： まあまあ合格だけど。僕ならこうするな。

```
#! /bin/sed -f
1!n
d
```

この方が短いだろ？ なんなら

```
#! /bin/sed -f
N
s/^.*\n//
```

ってのも、お茶目だけどね。

yasuoka ： だったらさっきの oddhf だって

```
#! /bin/sed -f
N
s/\n.*$//
```

ってできるじゃないですか。

root : それは、うまく行かないと思うよ。
yasuoka : え？ どうしてですか？
root : 理由は自分でよく考えてごらん。
yasuoka : はい。

```
/home/yasuoka/bin% number oddhf | oddhf （ぽこ）
     1  #! /bin/sed -f
     3  d
/home/yasuoka/bin% 
```

他に sed で覚えておくべき命令はないんですか？
root : うん。{ } は必要かな。

```
{
  複数の命令
}
```

複数の命令をひとまとめにする。

root : {の前に条件を書けば、} までの命令全部に影響が及ぶんだ。
yasuoka : 何かいい例ありませんか？
root : じゃ、次のファイルを~/bin の中に作ってくれるかい？ ファイル名は、longest だ。

```
#! /bin/sed -f
$q
N
h
y/:/#/
s/\n/:/
s/[^:]/#/g
/^\(#*\):\1#/{
  g
  D
}
g
s/^\(.*\)\(\n\).*$/\2\1/
D
```

（間）

yasuoka : chmod してっと。

```
/home/yasuoka/bin% chmod 755 longest （ぽこ）
/home/yasuoka/bin% rehash （ぽこ）
/home/yasuoka/bin% 
```

できました。どういうコマンドなんですか？
root : 入力でいちばん長い行を出力する。
yasuoka : へーえ。

```
/home/yasuoka/bin% history （ぽこ）
   125  cp oddhf evenhf
   126  vi evenhf
   127  ls -l
   128  rehash
   129  number oddhf | oddhf
   130  cat evenhf
   131  vi longest
   132  chmod 755 longest
   133  rehash
   134  history
/home/yasuoka/bin% history | longest （ぽこ）
   129  number oddhf | oddhf
/home/yasuoka/bin% 
```

どういう仕掛けなんですか？
root : もう少し短い例がいいな。

```
/home/yasuoka/bin% ls ??????* （ぽこ）
delcom  evenhf  headhf  longest number  tailhf  yasuoka
/home/yasuoka/bin% 
```

うーん。

```
/home/yasuoka/bin% ls *n* （ぽこ）
evenhf  longest number
/home/yasuoka/bin% 
```

お、これいいな。

```
/home/yasuoka/bin% ls -1 *n* （ぽこ）
evenhf
```

```
longest
number
/home/yasuoka/bin% !! | longest （ぽこ）
ls -1 *n* | longest
longest
/home/yasuoka/bin% 
```

ls -1 *n* の出力を、 longest の入力にした場合を考えてみよう。

yasuoka : はい。

root : まずバッファに「evenhf」が読み込まれた状態で、実行開始。 N h で、バッファとコピーバッファは両方とも「evenhf改longest」になり、次の y s s で、バッファは「######:#######」になる。

yasuoka : ええと、まず: を # に全部置き換えて、それから改行コードを: に置き換えて、最後に: 以外の文字を全部 # に置き換えてるから、結局改行コードは: に、その他の文字は # に置き換わるわけですね。でもどうしてこんな面倒くさいことをしてるんですか？

root : [ ] の中に \n が書けないからだよ。一部の sed では、書けるものもあるみたいだけど。で、次の /^\(#*\):\1#/ は、: の前の # の数と、: の後の # の数を較べて、後の方が多ければ真になる。ここでは後の方が多いから、{の中が実行されるわけだ。

yasuoka : \1 には: の前の # が格納されるわけだから…、うーん、なかなか絶妙の比較方法ですね。

root : g D でバッファは「longest」になり、スクリプトの最初に戻る。また N h でバッファとコピーバッファは両方とも「longest改number」、さらに y s s でバッファは「#######:######」になる。今度は /^\(#*\):\1#/ は真にならないから、} の後に実行が移って、 g で「longest改number」をバッファに写し、 s で「改longest」、さらに D で「longest」となって、またスクリプトの最初に戻る。

yasuoka : あ、そうか。いちばん長い行が必ずバッファに残ってるんだ。

root : そうそう。で、最初の $q が実行されて

```
longest
```

が出力されて終了。

yasuoka : めでたし、めでたし。すごいですねー。

root : ま、これぐらいの sed のスクリプトが書けるようになれば、一人前かな。

```
/home/yasuoka/bin% number longest （ぽこ）
```

```
 1  #! /bin/sed -f
 2  $q
 3  N
 4  h
 5  y/:/#/
 6  s/\n/:/
 7  s/[^:]/#/g
 8  /^\(#*\):\1#/{
 9    g
10    D
11  }
12  g
13  s/^\(.*\)\(\n\).*$/\2\1/
14  D
/home/yasuoka/bin% !! | !* （ぽこ）
number longest | longest
    13  s/^\(.*\)\(\n\).*$/\2\1/
/home/yasuoka/bin% 
```

yasuoka : そういえば、コメントはどうやって書くんですか？

root : 一部の sed では、 # から始まる行がコメントになるみたいだけど、通常はコメントは書けないと思っておいた方がいい。

yasuoka : どうしても書きたい時は？

root : 意味のないコマンドを書くって手はある。例えば

```
/ "longest" Version 1.0 /{}
```

とかね。でも、実行が遅くなるのが難点だな。

yasuoka : そうですか。

root : よし、それじゃ sed はこのくらいにしよう。これからも頑張って、正規表現を使いまくった sed のスクリプトを書くんだよ。

yasuoka : はい。どうもありがとうございました。