誰にでも書ける #! /bin/awk -f 講座
第 2 回
「再び半分のフィルタ」

安岡孝一

yasuoka : root さん、 root さん。

root : 何だい？

yasuoka : 今日は awk の if 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、何か例をあげようか。前に、入力の偶数行目だけを出力する sed のスクリプトって、作らなかったっけ。

yasuoka : evenhf ですか？

```
/home/yasuoka% cd bin （ぽこ）
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/sed -f
/ "evenhf" Version 1.1 /{}
1!n
d
/home/yasuoka/bin% evenhf evenhf （ぽこ）
/ "evenhf" Version 1.1 /{}
d
/home/yasuoka/bin% 
```

ああ、これだこれだ。これと同じ働きをする awk のスクリプトをあげよう。

```
#! /bin/awk -f
# "evenhf" Version 2.0
{
  if(NR%2==0)
    printf("%s\n",$0);
}
```

（間）

yasuoka : できました。

```
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/awk -f
# "evenhf" Version 2.0
{
  if(NR%2==0)
    printf("%s\n",$0);
}
/home/yasuoka/bin% !!~ （ぽこ）
cat evenhf~
#! /bin/sed -f
/ "evenhf" Version 1.1 /{}
1!n
d
/home/yasuoka/bin% 
```

sed のスクリプトの方が短いですね。

root : まあね。足し算とかをしなくてもいい場合には、だいたい sed のスクリプトの方が awk より短くなる。

yasuoka : ふーん。

```
/home/yasuoka/bin% evenhf evenhf （ぽこ）
# "evenhf" Version 2.0
  if(NR%2==0)
}
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/awk -f
# "evenhf" Version 2.0
{
  if(NR%2==0)
    printf("%s\n",$0);
}
/home/yasuoka/bin% 
```

この if(NR%2==0)って、 NR を 2 で割った余りが 0 と等しかったら、っていう意味ですか？

root : うん、そう。

yasuoka : ; がありませんけど。

root ： 次の行の; までが、ひとまとまりだ。NR が偶数なら printf("%s\n",$0) を実行する、って意味だよ。

```
if(条件)
  命令;
else
  命令;
```

条件が真ならば直後の命令を、さもなくば else の命令を実行する。else 節はなくてもよい。

条件は以下の通り。

文字列	ヌルでも "0"でもなければ真
式	結果が 0 でなければ真
式==式	2 式が等しいとき真、式は文字列でもよい
式!=式	2 式が等しくないとき真、式は文字列でもよい
式>式	左式が右式より大きいとき真
式>=式	左式が右式以上のとき真
式<式	左式が右式未満のとき真
式<=式	左式が右式以下のとき真
文字列~/正規表現/	文字列が正規表現にマッチするなら真
文字列!~/正規表現/	文字列が正規表現にマッチしないなら真
!条件	条件が偽のとき真（&& や || より優先）
条件&&条件	2 条件とも真のとき真（|| より優先）
条件||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

使用できる正規表現は、以下の通り。

^	文字列の先頭
$	文字列の末尾
.	任意の 1 文字
[複数の文字]	複数の文字のいずれか 1 文字、 - は省略を意味
[^複数の文字]	複数の文字以外の 1 文字、上と同じ省略が可能
*	直前の正規表現の 0 回以上の繰り返し
+	直前の正規表現の 1 回以上の繰り返し
?	直前の正規表現の 1 回以下の繰り返し
正規表現|正規表現	いずれかの正規表現
(正規表現)	正規表現それ自身、 * や | のおよぶ範囲を限定
\n	改行コード
\t	タブ
\^	^
\$	$
\.	.
\[	[
\]	]
*	*
\+	+
\?	?
\\|	|
\(	(
\)	)
\/	/
\\	\
その他の文字	その文字自身

yasuoka ： 条件の中に正規表現が使えるんですね。

root ： うん。でも sed の正規表現とは違うところがあるから、注意しなきゃね。

yasuoka ： どんなところが違うんですか？

root ： | が使えるところかな。例えば /abcd|axyzd/っていう正規表現は、 abcd か axyzd を含む文字列とマッチする。

yasuoka ： そんなの条件の方で、 || を使って書いたらいいんじゃないですか？

root ： まあそうだけど。これをもう少しまとめた /a(bc|xyz)d/ なんて、カッコいいと思わないかい？

yasuoka ： そうかなぁ。

root ： ただ、 sed の \1 とかは使えない。

yasuoka ： あら。あれ、便利だったのにぃ。

```
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/awk -f
# "evenhf" Version 2.0
{
```

```
  if(NR%2==0)
    printf("%s\n",$0);
}
/home/yasuoka/bin% 
```

if の後の命令は、1 つしか書けないんですか？

root ： いや、 sed と同じで { } を使えば、複数書くこともできるよ。

```
{
  命令;
  命令;
  …
}
```

複数の命令をひとまとめにする。

root ： 例えば、 NR が偶数なら printf("%s",$0) と printf("\n") を実行する、ってのは

```
#! /bin/awk -f
# "evenhf" Version 2.0
{
  if(NR%2==0){
    printf("%s",$0);
    printf("\n");
  }
}
```

って、書けばいい。

yasuoka ： { } は else にも使えるんですか？

root ： うん、どこでも使えるよ。

yasuoka ： わかりました。

root ： awk にはもう一つ if の書き方があるから、今度はそれで書いてみようか。

```
#! /bin/awk -f
# "evenhf" Version 2.1
NR%2==0{
  printf("%s\n",$0);
}
```

（間）

yasuoka ： 何か変な書き方ですね。

```
/home/yasuoka/bin% evenhf evenhf （ぽこ）
# "evenhf" Version 2.1
  printf("%s\n",$0);
/home/yasuoka/bin% 
```

でもちゃんと動いてるなぁ。

root ： これが awk の「パターン」って呼ばれるやつだよ。

yasuoka ： パターン？

root ： そう、パターン。 awk のスクリプトは基本的には

```
#! /bin/awk -f
パターン{
  命令列
}
パターン{
  命令列
}
…
```

っていう形を持ってるんだよ。で、入力の各行について、パターンが成立した時だけそれに対応する命令列を実行する。

yasuoka ： パターンって、どんなのが書けるんですか？

	常に実行する
条件	条件が真のとき実行する
条件,条件	左条件が真になってから、右条件が真になるまで
BEGIN	1 行目を入力する前、スクリプトの最初になければならない
END	最終行を入力した後、スクリプトの最後になければならない

条件は if のものとほぼ同じだが

文字列	ヌルでも "0"でもなければ真
式	結果が 0 でなければ真

が使用できない。

root ： ま、こんなとこかな。

yasuoka ： いちばん上の「常に実行する」ってのは？

root ： パターンに何も書かなかった時には、常に実行する、って意味だよ。

yasuoka : あ、左側は何もないんですか。誤植かと思っちゃった。で、さっきの evenhf ですけど

```
/home/yasuoka/bin% cat evenhf （ぽこ）
#! /bin/awk -f
# "evenhf" Version 2.1
NR%2==0{
  printf("%s\n",$0);
}
/home/yasuoka/bin% 
```

これは、 NR が偶数の行だけ printf("%s\n",$0) を実行する、って意味ですか？

root : そう。

yasuoka : NR が奇数の行はどうなるんですか？

root : NR%2==0 のパターンは成立しないし、このスクリプトには他のパターンはないから、結局何も実行されないで、すぐ次の行が読み込まれちゃう。

yasuoka : この evenhf の NR%2==0 のところを、 NR%2==1 に書き換えたら、逆に奇数行だけ取り出すようになりますよね。

```
#! /bin/awk -f
# "oddhf" Version 2.0
NR%2==1{
  printf("%s\n",$0);
}
```

root : うん、そうだよ。でも if を使えば、条件のところで手が抜ける。

```
#! /bin/awk -f
# "oddhf" Version 2.1
{
  if(NR%2)
    printf("%s\n",$0);
}
```

yasuoka : if とパターンとでは、条件の書き方が微妙に違うんですね。ところで root さん。

root : 何だい？

yasuoka : 前に sed で書いた、いちばん長い行を出力するスクリプト

```
/home/yasuoka/bin% cat longest （ぽこ）
#! /bin/sed -f
/ "longest" Version 1.0 /{}
$q
N
h
y/:/#/
s/\n/:/
s/[^:]/#/g
/^\(#*\):\1#/{
  g
  D
}
g
s/^\(.*\)\(\n\).*$/\2\1/
D
/home/yasuoka/bin% longest longest （ぽこ）
/ "longest" Version 1.0 /{}
/home/yasuoka/bin% 
```

これを awk に書き換えたら、どうなります？

root : うーん。

```
#! /bin/awk -f
# "longest" Version 2.0
length($0)>length(t){
  t=$0;
}
END{
  printf("%s\n",t);
}
```

（間）

yasuoka : できました。

```
/home/yasuoka/bin% longest longest （ぽこ）
# "longest" Version 2.0
/home/yasuoka/bin% 
```

お、なかなか。

```
/home/yasuoka/bin% !!~ （ぽこ）
longest longest~
/ "longest" Version 1.0 /{}
/home/yasuoka/bin% 
```

どういうしかけなんですか？

root ： しかけも何も、見たとおりだよ。入力行が変数 t より長かったら t に入力行を代入して、最後に t を出力してる。

yasuoka ： 3 行目の lengthって？

root ： 文字列の長さ。前回の式のところにあったろ。

yasuoka ： そうでしたっけ？

```
/home/yasuoka/bin% longest oddhf （ぽこ）
# "oddhf" Version 2.0
/home/yasuoka/bin% longest /dev/null （ぽこ）

/home/yasuoka/bin% 
```

あれ？

```
/home/yasuoka/bin% longest~ /dev/null （ぽこ）
/home/yasuoka/bin% 
```

root さん、これ変ですよ。

root ： /dev/null を入力した時の動きが違うな。どうしたらいいかな…。

```
#! /bin/awk -f
# "longest" Version 2.1
length($0)>=length(t){
  t=$0 "\n";
}
END{
  printf("%s",t);
}
```

（間）

root ： こんなもんでどうだい？

yasuoka ： ええ。

```
/home/yasuoka/bin% longest /dev/null （ぽこ）
/home/yasuoka/bin% 
```

うまくいってるみたいです。

```
/home/yasuoka/bin% longest longest （ぽこ）
# "longest" Version 2.1
/home/yasuoka/bin% 
```

どうなったんですか？

root ： t に $0 を代入する時に、必ず末尾に改行コードを付け加えるようにして、printf で改行コードを省いた。

yasuoka ： t=$0 "\n"ってのが、 t に $0 と改行コードをくっつけたのを代入する、って意味ですね。で、 printf("%s",t) は単純に t を出力する。でも、どうしてそれでうまくいくんですか？

root ： /dev/null が入力の時には、この longest では、いきなり END{のところが実行されるからね。その時 t は何も代入されてなくてヌルだから、printf の中に改行コードさえなければ何も出力されなくなるはずだ。さっきのだと、何が何でも改行コードだけは出力されたからね。そこで longest~ と動作が違ってきたんだ。

yasuoka ： あ、そうか。ところで root さん。

root ： 何だい？

yasuoka ： awk には入力各行毎の繰り返し以外に、もっと単純な繰り返しはないんですか？

root ： いや、ちゃんと while があるよ。でも今日はもう時間がないから、それはまた次回にしよう。次回は awk の while だ。

yasuoka ： はい、楽しみにしてます。どうもありがとうございました。