誰にでも使える m4 講座
第 2 回
「マクロの繰り返し」

安岡孝一

yasuoka : root さん、 root さん。
root : 何だい？
yasuoka : 複雑な m4 のマクロ定義。
root : ああ、そうだったね。じゃ、 m4 を立ち上げて。
yasuoka : はい。

```
~% m4 （ぽこ）
```

root : まずは条件分岐から。 `define(‘test’,‘ifelse($1,z,OK,KO)’)` してみてごらん。
yasuoka : `define(‘test’,‘ifelse($1,z,OK,KO)’)` ですね。

```
define(‘test’,‘ifelse($1,z,OK,KO)’) （ぽこ）
```

root : それで `test(z)` してごらん。
yasuoka : はい。

```
test(z) （ぽこ）
OK
```

OK になりました。
root : `test(x)` は？
yasuoka : えっと

```
test(x) （ぽこ）
KO
```

KO です。どういうことですか？
root : `test(z)` は `ifelse(z,z,OK,KO)` に置き換えられるだろ。
yasuoka : はい。
root : で、 `ifelse`っていうのは、最初のパラメータと 2 番目のパラメータが同じだったら、 3 番目のパラメータを返すんだ。

> `ifelse(`文字列`,`文字列`,`文字列`,`文字列`)`
> 最初の文字列と 2 番目の文字列が同じならば、 3 番目の文字列を返す。さもなくば 4 番目の文字列を返す。 4 番目の文字列がない場合は、ヌルが指定されたものとみなす。
> `ifdef(`文字列`,`文字列`,`文字列`)`
> 最初の文字列がマクロ定義されていれば、 2 番目の文字列を返す。さもなくば 3 番目の文字列を返す。 3 番目の文字列がない場合は、ヌルが指定されたものとみなす。
> `ifelse(`A`,`B`,`C`,‘ifelse(`D`,`E`,`F`,`G`)’)` は `ifelse(`A`,`B`,`C`,`D`,`E`,`F`,`G`)` に簡略化できる。

root : この場合は OK が返ってくる。
yasuoka : 最初のパラメータと 2 番目のパラメータが違ったら？
root : その時は 4 番目のパラメータが返ってくる。例えばさっきの `text(x)` を置き換えると、 `ifelse(x,z,OK,KO)` だから、今度は KO になるんだ。
yasuoka : うーん、そういうことですか。
root : それから m4 には簡単な計算機能がある。 `eval(1+2)` してごらん。
yasuoka : `eval(1+2)` ですね。

```
eval(1+2) （ぽこ）
3
```

足し算ですか？
root : そういうこと。 Unix コマンドの `expr` と同じくらいの計算はできるよ。それに冪乗もできる。 `eval(5**3)` してごらん。
yasuoka : 5 の 3 乗ですか？

```
eval(5**3) （ぽこ）
125
eval(2**3**2) （ぽこ）
512
```

2 の 3 の 2 乗乗もちゃんとできますね。

eval(式)
　　式を計算した結果を返す。
式は以下の通り。

数	数（整数のみ有効）
式||式	2 式の両方が 0 のとき 0、さもなくば 1
式&&式	2 式のいずれかが 0 のとき 0、さもなくば 1
!式	式が 0 のとき 1、さもなくば 0
式==式	2 式が等しいとき 1、さもなくば 0
式!=式	2 式が等しくないとき 1、さもなくば 0
式>式	左式が右式より大きいとき 1、さもなくば 0
式>=式	左式が右式以上のとき 1、さもなくば 0
式<式	左式が右式未満のとき 1、さもなくば 0
式<=式	左式が右式以下のとき 1、さもなくば 0
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**式	左式の右式乗
(式)	優先度を変える

root ： で、これを利用すれば、もうちょっと複雑な定義ができるようになる。`define(‘S’,‘ifelse($1,1,1,‘$0(eval($1-1))+$1’)’)` してごらん。

yasuoka ： はい。

`define(‘S’,‘ifelse($1,1,1,‘$0(eval($1-1))+$1’)’)` （ぽこ）

root ： `S(10)` するとどうなる？

yasuoka ： `S(10)` ですね。

`S(10)` （ぽこ）
`1+2+3+4+5+6+7+8+9+10`

どういう仕掛けなんですか？

root ： まず `S(10)` が `ifelse(10,1,1,‘S(eval(10-1))+10’)` に置き換えられるだろ？

yasuoka ： はい。

root ： 次にパラメータが置き換えられて `ifelse(10,1,1,S(eval(10-1))+10)` になる。

yasuoka ： パラメータが置き換えられるって？

root ： 10 は 10 に、 1 は 1 に、その次の 1 も 1 に、最後の ‘`S(eval(10-1))+10`’ は `S(eval(10-1))+10` にそれぞれ置き換えられるんだ。それから `ifelse` 自身が置き換えられて `S(eval(10-1))+10` になる。 m4 でのマクロの置き換えは、まずマクロの中のパラメータを置き換えてから、マクロ自身を置き換えるんだ。

yasuoka ： うーん、そういうことなんですか。

root ： で、 `S(eval(10-1))+10` は、まず `eval(10-1)` が置き換えられるわけだけど、その前に実は `10-1` が `10-1` に置き換えられてから `eval(10-1)` が 9 になる。これで `S(9)+10` だ。

yasuoka ： だんだん分かってきました。

root ： `S(9)` が置き換えられて `ifelse(9,1,1,‘S(eval(9-1))+9’)+10` だから、`ifelse(9,1,1,S(eval(9-1))+9)+10` になって `S(eval(9-1))+9+10` になる。これを繰り返していって `S(1)+2+3+4+5+6+7+8+9+10` まで置き換えられていくわけだ。

yasuoka ： いよいよですね。

root ： これは `ifelse(1,1,1,‘S(eval(1-1))+1’)+2+3+4+5+6+7+8+9+10` に置き換わるから、 `ifelse(1,1,1,S(eval(1-1))+1)+2+3+4+5+6+7+8+9+10` になる。ここで `ifelse` の第 1 パラメータと第 2 パラメータが同じなので第 3 パラメータの 1 が返ってくるから、 `1+2+3+4+5+6+7+8+9+10` で置き換え終了。

yasuoka ： `S(10)` は結局 `1+2+3+4+5+6+7+8+9+10` になるわけですね。

root ： そういうこと。さらに `eval(S(10))` してごらん。

yasuoka ： `eval(S(10))` ですか？

`eval(S(10))` （ぽこ）
`55`

`eval(1+2+3+4+5+6+7+8+9+10)` ですね。

root ： そのとおり。だから `define(‘sum’,‘eval(S($1))’)` しておけば、総和

が簡単に求められるわけだ。

yasuoka : そうですね。

```
define(‘sum’,‘eval(S($1))’)dnl （ぽこ）
sum(100) （ぽこ）
5050
```

m4 の eval は expr と違って、文字列の取り出しとかはできないんですか？

root : できない。でも m4 には substr があるからね。

substr(文字列,数) 　　文字列の「数」文字目の次の文字以降を返す。 substr(文字列,数,数) 　　文字列の左「数」文字目の次の文字から右「数」文字を返す。

yasuoka : awk の substr と同じですか？

root : 実はちょっと違う。 substr(‘abcdefg’,3) とでもしてごらん。

yasuoka : はい。

```
substr(‘abcdefg’,3) （ぽこ）
defg
```

ん？

```
substr(‘abcdefg’,1,3) （ぽこ）
bcd
```

そうすると

```
substr(‘abcdefg’,0,7) （ぽこ）
abcdefg
```

左端が 0 なんですか？

root : そういうこと。実は index もそうで、 awk の index より 1 少ない値を返す。ためしてごらん。

yasuoka : えっと

```
index(‘abcdefg’,‘a’) （ぽこ）
0
index(‘abcdefg’,‘cd’) （ぽこ）
2
index(‘abcdefg’,‘ad’) （ぽこ）
-1
```

こういうことですね。

index(文字列,文字列) 　　「左文字列の何文字目に右文字列を含むか」から 1 を減じた数を返す。左端の場合は 0、含まない場合は -1。

yasuoka : ということは、 length もあるんですね。

```
length(‘abcdefg’) （ぽこ）
length(abcdefg)
```

あら、ない。

root : length じゃなくて len だよ。

len(文字列) 　　文字列の長さを返す。

yasuoka : あ、そうなんですか。

```
len(‘abcdefg’) （ぽこ）
7
```

len(‘abcdefg’) が len(abcdefg) に置き換えられて、長さは 7 文字っと。あ、そうだ。

```
len(S(10)) （ぽこ）
20
S(10) （ぽこ）
1+2+3+4+5+6+7+8+9+10
```

やっぱり思った通りですね。

root : len(‘S(10)’) はどうなる？
yasuoka : 5 じゃないんですか？

```
len(‘S(10)’) (ぽこ)
5
```

ほら、やっぱり。 S(10) は 5 文字ですからね。

```
‘S(10)’ (ぽこ)
S(10)
S(10) (ぽこ)
1+2+3+4+5+6+7+8+9+10
```

substr と index と len と、他にはないんですか？
root : translit がある。

> translit(文字列,文字列,文字列)
> 最初の文字列に対して、 2 番目の文字列の各文字を、 3 番目の文字列の同じ順番のところにある文字に置き換えた文字列を返す。 2 番目の文字列と 3 番目の文字列は同じ長さでなければならない。

root : translit(‘bee’,‘abcdefg’,‘ABCDEFG’) してごらん。
yasuoka : はい。

```
translit(‘bee’,‘abcdefg’,‘ABCDEFG’) (ぽこ)
BEE
```

あれ、大文字になった。
root : translit は、 Unix の tr とか、 sed の y とかと同じようなものなんだよ。
yasuoka : あ、そういうことですか。 a-z とかの省略はできるんですか？
root : できない。そこは sed の y と同じだ。ただし最近の m4 では、できるものもあるみたいだけどね。
yasuoka : わかりました。

```
translit(‘rum(10)’,‘r’,‘s’) (ぽこ)
55
```

rum(10) が sum(10) に置き換えられた後、さらに eval(S(10)) に置き換えられて、最終的に 55 までいくんですね。
root : そういうこと。
yasuoka : うーん、すると

```
translit(‘eval(S(10))’,‘+’,‘*’) (ぽこ)
55
```

あれ、違うなあ。

```
eval(translit(S(10),‘+’,‘*’)) (ぽこ)
3628800
```

お、うまくいった。
root : なかなかやるね。
yasuoka : へへー。

```
define(‘F’,‘translit(S($1),‘+’,‘*’)’) (ぽこ)

F(10) (ぽこ)
1*2*3*4*5*6*7*8*9*10
define(‘factorial’,‘eval(F($1))’) (ぽこ)

factorial(10) (ぽこ)
3628800
```

yasuoka : これで m4 の文字列関係の命令はおしまいですか？
root : ちょっと違うけど、あと changequoteってのがある。

> changequote(文字,文字)
> 置き換え抑止文字を変更する。何も返さない。
> changequote
> 置き換え抑止文字を ‘と’ に戻す。何も返さない。

root : changequote({,}) としてごらん。
yasuoka : changequote({,}) ですね。

```
changequote({,}) （ぽこ）

```

これでどうなったんですか？

root ： ‘ ’の代わりに { }が置き換えを抑止するようになったんだ。ためしに {unix} としてごらん。

yasuoka ： はい。

```
{unix} （ぽこ）
unix
```

うーん

```
‘unix’ （ぽこ）
‘’
```

あらら、すると

```
{S(10)} （ぽこ）
S(10)
‘S(10)’ （ぽこ）
m4: arg stack overflow
~%
```

あれ？ m4 が終了しちゃった。

root ： changequote({,}) で ‘ ’は置き換えを抑止しなくなるから、‘S(10)’ は ‘ifelse(10,1,1,‘S(eval(10-1))+10’)’ に置き換えられる。その次に ifelse の中のパラメータが先に置き換えられるけど、‘ ’は置き換えを抑止しなくなってるから、第 4 パラメータの ‘S(eval(10-1))+10’ は ‘ifelse(9,1,1,‘S(eval(9-1))+9’)+10’ になっちゃう。さらに中のパラメータが置き換えられて、っていう風に、 ifelse 自体が置き換えられる以前に中のパラメータがどんどん置き換えられていっちゃうので、最後は m4 が異常終了してしまうんだ。

yasuoka ： そういうことですか。じゃあ changequote は、 define する前にしなきゃいけませんね。

root ： そういうことかな。

yasuoka ： ところで root さん。

root ： 何だい？

yasuoka ： m4 のスクリプトってないんですか？

root ： うーん、あまりそういうのは書かないもんだけど…。ちょっとごめんね。

```
~% cat ~/bin/malias （ぽこ）
#! /bin/sh
# "malias" Version 1.1 for BSD
( tr 'a-z@%!.\055' A-Za-e < $HOME/.mailrc | awk '
$1=="A"{
  printf("define(‘z%s%c,‘z%s",$2,39,$3);
  for(i=4;i<=NF;i++)
    printf(" z%s",$i);
  printf("%c)dnl\n",39);
}'
  echo -n z
  echo "$*" | tr 'a-z@%!.\055' A-Za-e | sed 's/ / z/g'
) | m4 | tr A-Za-e 'a-z@%!.\055' | tr -d z
exit 0
~%
```

これが前回作った malias だっけ？

```
~% malias tm unix （ぽこ）
takahash matukawa unix
~% egrep '^a ' .mailrc （ぽこ）
a taka takahash
a matu matukawa
a tm taka matu
~%
```

メールの宛先の別名を展開するんだったね。

yasuoka ： そうです。

root ： じゃあ、今日はもう時間がないから、次回までに考えておくよ。次回はこの malias の m4 版スクリプトだ。

yasuoka ： 楽しみに待ってます。よろしくお願いします。