続・誰にでも使える Unix 講座
第 2 回
「大きなファイル、どーんどーん」

安岡孝一

yasuoka ： root さん、 root さん。
root ： 何だい？
yasuoka ： tar して compress して uuencode したんですけど、ファイルが大きすぎるんです。

```
~/tmp% ls -l（ぽこ）
total 214
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
-rw-r--r--  1 yasuoka    120812 Mar 19 14:13 lfh.uu
~/tmp% ■
```

root ： この lfh.uu をメールで送りたいのかい？
yasuoka ： はい。
root ： うーん、確かに 120812 バイトは大きすぎるな。メール 1 通 50000 バイトが限界だからな。
yasuoka ： どうしましょう。
root ： ファイルを分割して送るしかないだろう。
yasuoka ： 分割ですか？
root ： そう。方法は 2 つある。1 つはこの lfh.uu を分割して送る方法。もう 1 つは lfh.tar.Z を分割した後、それぞれを uuencode して送る方法。どっちがいい？
yasuoka ： 両方教えて下さい。
root ： 何だ、欲張りだな。じゃ、まずは lfh.uu を分割しよう。

> split -数 ファイル名 文字列
> ファイルを「数」行ずつに分割し、「文字列aa」、「文字列ab」、…というファイルに出力する。文字列が省略された場合は、 x が指定されたものとみなす。入力はファイル、出力は上記のファイル。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。

root ： この lfh.uu、何行ある？
yasuoka ： えっと

```
~/tmp% wc lfh.uu（ぽこ）
    1952    5814  120812 lfh.uu
~/tmp% ■
```

1952 行ですね。
root ： 3 で割ると？
yasuoka ： 650.66 …。
root ： じゃ、 split -651 lfh.uu を実行してごらん。
yasuoka ： はい。

```
~/tmp% split -651 lfh.uu（ぽこ）
~/tmp% ■
```

これでどうなったんですか？
root ： ls -l してごらん。
yasuoka ： はい。

```
~/tmp% ls -l（ぽこ）
total 334
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
-rw-r--r--  1 yasuoka    120812 Mar 19 14:13 lfh.uu
-rw-r--r--  1 yasuoka     40320 Mar 19 15:07 xaa
-rw-r--r--  1 yasuoka     40362 Mar 19 15:07 xab
-rw-r--r--  1 yasuoka     40130 Mar 19 15:07 xac
~/tmp% ■
```

xaa と xab と xacってファイルが出来てます。
root ： それが lfh.uu を分割したものだよ。それぞれ 651 行づつあるはずだ。
yasuoka ： そうなんですか？

```
~/tmp% wc x*（ぽこ）
     651    1920   40320 xaa
     651    1793   40362 xab
     650    2101   40130 xac
    1952    5814  120812 total
~/tmp% ■
```

あ、だいたいそうだ。

root : で、それを 1 ファイルづつメールで送って、向こうで一つにくっつけてもらって、 uudecode してもらえばいいんだよ。

yasuoka : 一つにくっつけるって？

root : cat xaa xab xac > x.uu としてごらん。

yasuoka : はい。

```
~/tmp% cat xaa xab xac > x.uu （ぽこ）
~/tmp% ls -l （ぽこ）
total 462
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
-rw-r--r--  1 yasuoka    120812 Mar 19 14:13 lfh.uu
-rw-r--r--  1 yasuoka    120812 Mar 19 15:10 x.uu
-rw-r--r--  1 yasuoka     40320 Mar 19 15:07 xaa
-rw-r--r--  1 yasuoka     40362 Mar 19 15:07 xab
-rw-r--r--  1 yasuoka     40130 Mar 19 15:07 xac
~/tmp% 
```

root : それで元通りだ。

```
~/tmp% cmp lfh.uu x.uu （ぽこ）
~/tmp% 
```

ね。

yasuoka : ね？

cmp ファイル名 ファイル名
: 2 つのファイルを先頭から較べ、最初に異なっていたところを標準出力に出力する。

cmp -l ファイル名 ファイル名
: 2 つのファイルの違っているバイトを、バイト位置（10 進数）、前のファイルの内容（8 進数）、後のファイルの内容（8 進数）の順に標準出力に出力する。

いずれも、前のファイル名として - が指定された場合には、標準入力が用いられる。エグジットステイタスは、違いがなかった時には 0、あった時には 1 となる。

yasuoka : ああ、こういうことですか。

root : そういうことだ。でも split で lfh.uu を分割してメールで送るこのやり方だと、実はちょっと問題がある。

yasuoka : 問題？

root : うん。 xaa と xab と xac をそれぞれメールで送ったとして、メールの最初には From とか色々な行が付くだろ。

yasuoka : はい。

root : それを向こうでちゃんと削り取ってもらわないといけない。

yasuoka : うーん。

root : しかもメールは出した順番に着くとは限らない。

yasuoka : じゃ、どうしたらいいんですか？

root : もう 1 つの方法、つまり lfh.tar.Z を分割した後、それぞれを uuencode して送る方法を使うんだ。まずは、さっき作ったファイルを全部消してくれるかい？

yasuoka : はい。

```
~/tmp% ls （ぽこ）
lfh.tar.Z       x.uu            xab
lfh.uu          xaa             xac
~/tmp% rm x* （ぽこ）
rm: remove x.uu? y （ぽこ）
rm: remove xaa? y （ぽこ）
rm: remove xab? y （ぽこ）
rm: remove xac? y （ぽこ）
~/tmp% ls （ぽこ）
lfh.tar.Z       lfh.uu
~/tmp% 
```

消しました。

root : その lfh.uu も要らないな。それも消して。

yasuoka : はい。

```
~/tmp% rm lfh.uu （ぽこ）
rm: remove lfh.uu? y （ぽこ）
~/tmp% ls -l （ぽこ）
total 86
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
~/tmp% 
```

root : よし OK。じゃ、~/bin の中に次のシェル・スクリプトを作ってくれるかい？ ファイル名は bsplit だ。

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

case "$1" in
-[1-9]*) COUNT=‘expr "$1" : ’-\([0-9]*\)$’ ’|’ 10000‘
         shift ;;
*) COUNT=10000 ;;
esac

IF=/tmp/bsplit$$
trap "rm -f $IF ; exit 2" 1 2 3 15
cat $1 > $IF
X=${2-x}

N=0
SKIP=0
set ‘wc $IF‘
while [ $SKIP -lt $3 ]
do OF=$X‘echo $N | awk ’{printf("%c%c",$1/26+97,$1%26+97);}’‘
   dd if=$IF of=$OF bs=1 skip=$SKIP count=$COUNT 2> /dev/null
   N=‘expr $N + 1‘
   SKIP=‘expr $SKIP + $COUNT‘
done

rm -f $IF
exit 0
```

（間）

yasuoka ： できました。
root ： じゃ、さっきの lfh.tar.Z の大きさを見せてよ。
yasuoka ： はい。

```
~/bin% pushd（ぽこ）
~/tmp ~/bin
~/tmp% wc lfh.tar.Z（ぽこ）
     462    1336   87665 lfh.tar.Z
~/tmp% 
```

root ： 50000 バイトのメールは、 uuencode 前だったら 36000 バイトくらいだから…。それじゃ単純に bsplit -36000 lfh.tar.Z を実行してみよう。
yasuoka ： bsplit -36000 lfh.tar.Z ですね。

```
~/tmp% bsplit -36000 lfh.tar.Z（ぽこ）
~/tmp% 
```

どうなったんですか？
root ： lfh.tar.Z が 36000 バイトづつ 3 つのファイルに分割されたはずだよ。
yasuoka ： え？

```
~/tmp% ls -l（ぽこ）
total 174
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
-rw-r--r--  1 yasuoka     36000 Mar 19 15:35 xaa
-rw-r--r--  1 yasuoka     36000 Mar 19 15:35 xab
-rw-r--r--  1 yasuoka     15665 Mar 19 15:36 xac
~/tmp% 
```

この xaa と xab と xac がそうですか？
root ： うん。
yasuoka ： で、どうすればいいんですか？
root ： まずは uuencode xaa lfh.tar.Z1of3 > uu1 かな。
yasuoka ： はい。

```
~/tmp% uuencode xaa lfh.tar.Z1of3 > uu1（ぽこ）
~/tmp% 
```

root ： 同様のことを xab は lfh.tar.Z2of3 と uu2 で、 xac は lfh.tar.Z3of3 と uu3 でやる。
yasuoka ： あ、そういうことですか。

```
~/tmp% uuencode xab lfh.tar.Z2of3 > uu2（ぽこ）
~/tmp% uuencode xac lfh.tar.Z3of3 > uu3（ぽこ）
~/tmp% ls -l（ぽこ）
total 294
-rw-r--r--  1 yasuoka     87665 Mar 19 14:09 lfh.tar.Z
-rw-r--r--  1 yasuoka     49630 Mar 19 15:40 uu1
-rw-r--r--  1 yasuoka     49630 Mar 19 15:42 uu2
```

```
-rw-r--r--  1 yasuoka     21616 Mar 19 15:43 uu3
-rw-r--r--  1 yasuoka     36000 Mar 19 15:35 xaa
-rw-r--r--  1 yasuoka     36000 Mar 19 15:35 xab
-rw-r--r--  1 yasuoka     15665 Mar 19 15:36 xac
~/tmp% ■
```

で、この uu1 と uu2 と uu3 とをそれぞれメールで送るわけですね。

root ： そう。そして向こうで各メール毎に uudecode してもらう。

yasuoka ： それで出来た 3 つのファイルを cat で一つにまとめた後、 uncompress して tar ですね。

root ： そういうこと。

yasuoka ： うーん、なかなか。でもどうやってファイルを 36000 バイトづつに分割したんですか？

root ： bsplit を見てごらん。基本は 20 行目の dd だ。

> dd if=ファイル名 of=ファイル名 bs=1 skip=数 count=数
> if で指定されたファイルの最初から skip で指定されたバイト数を読みとばし、そこから count で指定されたバイト数を of で指定されたファイルに出力する。それと同時に、入出力したバイト数が標準エラーに出力される。 if が省略された場合は、入力は標準入力。 of が省略された場合は、出力は標準出力。 skip が省略された場合は、 skip=0 とみなす。 count が省略された場合は、入力の最後までを出力する。

root ： ちょっとやってみせようか？

```
~/tmp% head -1 uu1 （ぽこ）
begin 644 lfh.tar.Z1of3
~/tmp% ■
```

この uu1ってファイルの最初から 16 バイト目までを、 tempってファイルに出力してみよう。

```
~/tmp% dd if=uu1 of=temp bs=1 skip=0 count=16 （ぽこ）
16+0 records in
16+0 records out
~/tmp% cat temp （ぽこ）
begin 644 lfh.ta~/tmp% ■
```

改行コードが無いんで、 cat するとちょっと気持ち悪いな。

```
begin 644 lfh.ta~/tmp% （ぽこ）
~/tmp% ■
```

でも 16 バイト目までだろう？

yasuoka ： b、 e、 g、 i、 n、空白、 6、 4、 4、空白、 l、 f、 h、 .、 t、 a。はい、16 バイトです。

root ： bsplit ではこれを繰り返して、ファイルを指定されたバイト数づつに分割してるんだ。

```
~/tmp% rm temp （ぽこ）
rm: remove temp? y （ぽこ）
~/tmp% ■
```

yasuoka ： その繰り返しが 18 行目から 23 行目までの while ですね。

root ： そういうこと。出力ファイル名の末尾の aa とか ab とかは、 19 行目の awk で作ってる。 awk の命令は読めるね？

yasuoka ： はい。「誰にでも書ける #! /bin/awk -f 講座」で勉強しました。

> awk 文字列 ファイル名
> 文字列を awk の命令列とみなして、 awk を実行する。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 ： じゃ、ここは説明しなくてもいいな。あと bsplit で、わかりにくいところはと…。 expr "$1" : '-\([0-9]*\)$' '|' 10000 は、ちょっとむずかしいかな。

yasuoka ： どこですか？

root ： 5 行目。

> expr 式
> 式を計算した結果を標準出力に出力する。なお、計算結果が 0 もしくはヌルのときエグジットステイタスは 1、それ以外のとき 0 となる。
> 式は以下の通り。
>
> | | |
> |---|---|
> | 数 | 数（整数のみ有効） |
> | 文字列 | 文字列 |
> | 式 '\|' 式 | 左式が 0 もしくはヌルのとき右式、さもなくば左式
ただし、両式とも 0 もしくはヌルのときは 0 |
> | 式 '&' 式 | 2 式のどちらかが 0 かヌルならば 0、さもなくば左式 |

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 : 正規表現	式（文字列）が正規表現にマッチした文字数 あるいは \(\) の部分にマッチした文字列
'(' 式 ')'	優先度を変える

使用できる正規表現は、以下の通り。

$	式の末尾
.	任意の 1 文字
[複数の文字]	複数の文字のいずれか 1 文字、 - は省略を意味する
[^複数の文字]	複数の文字以外の 1 文字、 - は省略を意味する
*	直前の 1 文字の 0 回以上の繰り返し
\(正規表現\)	正規表現にマッチした文字列を返すようにする
\$	$
\.	.
\[	[
\]	]
*	*
\\	\
その他の文字	その文字自身

正規表現のマッチングは、必ず式の先頭からおこなわれる。

yasuoka : exprって、加減乗除のコマンドじゃなかったんですか？

root : ま、そういう機能が基本だけどね。正規表現を使って文字列の取り出しとかもできる。ちょっとやってみせよう。

```
~/tmp% expr abcdef : '[^d]*d' （ぽこ）
4
~/tmp% 
```

これは、 abcdef の先頭から何文字が [^d]*dっていう正規表現にマッチしたか、つまり dっていう文字が abcdef の何文字目にあるかを返してる。見ての通り 4 文字目だね。 sed の正規表現なんかと違うところは、マッチングが必ず文字列の先頭からおこなわれることと、基本的にはマッチした文字数が返されることだ。

yasuoka : 正規表現の先頭に必ず^ が付いてる、と思えばいいんですか？

root : まあ近い線だな。当然だけど expr の正規表現では、^ は意味を持たない。

yasuoka : ふーん。

root : それから \(\) は 1 回だけ使えて、もし使われた時には、マッチした文字数じゃなく、 \(\) の部分にマッチした文字列を返す。

```
~/tmp% expr `pwd` : '.*/\([^/]*\)$' （ぽこ）
tmp
~/tmp% 
```

だからこういう basename もどきも簡単にできる。

yasuoka : なーかなか。

root : これを使って bsplit の 5 行目では、オプションの -36000 から 36000 を取り出してる。

```
~/tmp% expr "-36000" : '-\([0-9]*\)$' '|' 10000 （ぽこ）
36000
~/tmp% 
```

yasuoka : 取り出してるのはわかったんですけど、その後ろの'|' 10000 は？

root : -8ca0 みたいな数字とそれ以外の混ざったものを指定された時に、デフォルトを 10000 にするためだよ。

```
~/tmp% ^-36000^-8ca0^ （ぽこ）
expr "-8ca0" : '-\([0-9]*\)$' '|' 10000
10000
~/tmp% 
```

'|'ってのは、前の式の値がヌルになった時には後の式の値を返すからね。ああ、もうこんな時間だ。メールも分割して送れるようになったし、今日はここまでにしようか。

yasuoka : はい。どうもありがとうございました。