誰にでも使える make 講座
第 1 回
「謎の Makefile」

安岡孝一

yasuoka : root さん、 root さん。
root : 何だい？
yasuoka : Makefileって何ですか？
root : make 用のファイルだけど。 Makefile がどうかしたのかい？
yasuoka : takahash くんからもらった C のプログラムに、ついてきたんです。

```
/home/yasuoka/src/kbanner% ls （ぽこ）
Makefile        font.h          font2.c         kbanner.c
README          font1.c         font3.c
/home/yasuoka/src/kbanner% cat Makefile （ぽこ）
SHELL = /bin/sh

kbanner: kbanner.o font1.o font2.o font3.o
        cc kbanner.o font1.o font2.o font3.o -o kbanner
        strip kbanner

clean:
        rm -f *.o core

kbanner.o: kbanner.c
        cc -c kbanner.c

font1.o: font1.c font.h
        cc -c font1.c

font2.o: font2.c font.h
        cc -c font2.c

font3.o: font3.c font.h
        cc -c font3.c
/home/yasuoka/src/kbanner% 
```

root : ふーん。じゃ、まずは make を実行してごらん。
yasuoka : make ですか？

```
/home/yasuoka/src/kbanner% make （ぽこ）
cc -c kbanner.c
cc -c font1.c
cc -c font2.c
cc -c font3.c
cc kbanner.o font1.o font2.o font3.o -o kbanner
strip kbanner
/home/yasuoka/src/kbanner% 
```

あれ、何か表示されましたけど。
root : make が実行したコマンドが、表示されたんだよ。たぶん kbannerってファイルが出来てるはずだな。 ls してごらん。
yasuoka : はい。

```
/home/yasuoka/src/kbanner% ls （ぽこ）
Makefile        font1.c         font2.o         kbanner
README          font1.o         font3.c         kbanner.c
font.h          font2.c         font3.o         kbanner.o
/home/yasuoka/src/kbanner% 
```

何か、いっぱい増えてます。どうなってるんですか？
root : cc は知ってるかい？
yasuoka : C コンパイラだってのは、聞いたことがあるんですけど…。
root : あまり自信がなさそうだな。じゃ、まずは cc から説明しておこう。

> cc -c .cで終わるファイル名
> .c で終わるファイルを、 C のプログラムとしてコンパイルする。入力は.c で終わるファイル、出力は.o で終わるファイル。
> cc .oで終わるファイル名 -o ファイル名
> .o で終わるファイルを、実行可能な 1 つのファイルにまとめる。.o で終わるファイル名は複数書いてもよい。「-o ファイル名」が省略された場合には「-o a.out」が指定されたものとみなす。

root : さっきのやつだと、まず

```
cc -c kbanner.c
```

で kbanner.c から kbanner.o を作って、次に

```
cc -c font1.c
```

で font1.c から font1.o を作って

```
cc -c font2.c
cc -c font3.c
```

font2.o と font3.o もそういう風に作ってから

```
cc kbanner.o font1.o font2.o font3.o -o kbanner
```

で kbannerってファイルにまとめて、最後は

```
strip kbanner
```

が実行されてる。

yasuoka : stripって何ですか？

root : 実行ファイルの中の要らない部分を削り取るんだよ。

> strip ファイル名
> 実行可能なファイルから、 cc でまとめるために必要だった情報を削り取る。入出力ともにファイル。ファイル名は複数書いてもよい。

yasuoka : だいたいわかったんですけど、どうしてそれが make 一つで実行されたんですか？

root : そうだな、 make についてちょっと説明しようか。

> make 文字列
> 文字列をターゲットとみなして、 Makefile を実行する。文字列を省略した場合には、 Makefile の最初のターゲットが指定されたものとみなす。

root : make は基本的には、指定されたファイルを作るコマンドなんだ。例えば make kbanner とすれば、 kbannerってファイルを作ってくれるし、 make font1.o とすれば、 font1.oってファイルを作ってくれる。

yasuoka : さっきは何も書かずに、ただ単に make でしたよ。

root : 何も書かなかった時については、後で説明するから、そう先走らないで。

yasuoka : はい。

root : で、指定されたファイルを実際にどうやって作るかだけど、 Makefile に作り方を書いておけば、 make が勝手にそれを読んでやってくれる。例えば make kbanner を実行すると

```
/home/yasuoka/src/kbanner% cat Makefile （ぽこ）
SHELL = /bin/sh

kbanner: kbanner.o font1.o font2.o font3.o
        cc kbanner.o font1.o font2.o font3.o -o kbanner
        strip kbanner

clean:
        rm -f *.o core

kbanner.o: kbanner.c
        cc -c kbanner.c

font1.o: font1.c font.h
        cc -c font1.c

font2.o: font2.c font.h
        cc -c font2.c

font3.o: font3.c font.h
        cc -c font3.c
/home/yasuoka/src/kbanner%
```

make は、これの 3 行目から 5 行目の「kbanner を作るには kbanner.o と font1.o と font2.o と font3.o が必要で、作るために実行すべきコマンドは cc kbanner.o font1.o font2.o font3.o -o kbanner の後 strip kbanner」っていう意味のことが書かれた部分を読んで、それに従う。

yasuoka : タブから始まる行がコマンドなんですね。

root : そう。そして make は、必要なファイルの作り方をさらに見に行く。例えばさっき、 kbanner を作るには kbanner.o が必要って言ったけど、さらにその kbanner.o を作るために、 10 行目から 11 行目の「kbanner.o を作るには kbanner.c が必要で、作るためのコマンドは cc -c kbanner.c」

を読むわけだ。で、 kbanner.c の作り方は書いてないけど、これは元々カレントディレクトリにあるファイルだから、別段何もしない。 font1.o、 font2.o、 font3.o も kbanner.o と同じだね。そんな風に順々に辿っていって、最終的には

```
cc -c kbanner.c
cc -c font1.c
cc -c font2.c
cc -c font3.c
cc -o kbanner.o font1.o font2.o font3.o -o kbanner
strip kbanner
```

の順番でコマンドが実行されるんだ。

yasuoka : だいたいわかりました。

root : で、何も書かずに単に make だけした時だけど、その時には Makefile のいちばん最初のが作られる。この Makefile だと kbanner だ。

yasuoka : あれ？ 1 行目の SHELL = /bin/sh は？

root : これは「各々のコマンドは /bin/sh で実行する」っていう意味なんだ。これを書いておくと、さっきの cc -c kbanner.c とか strip kbanner とかは、 1 行ずつ /bin/sh を通じて実行されるんだよ。

yasuoka : でもそれなら、作り方を書いたシェル・スクリプトを、置いておけばいいんじゃないですか？ 何も、こんなまどろっこしい Makefile みたいなものを書かなくても。

root : make は単なるシェル・スクリプトよりは、能力が高いんだよ。論より証拠、もう一度 make を実行してごらん。

yasuoka : え？

```
/home/yasuoka/src/kbanner% make （ぽこ）
‘kbanner’ is up to date.
/home/yasuoka/src/kbanner% ■
```

あれ、さっきと違う。

root : うん、 make は不必要だと判断したコマンドは、実行しない。

yasuoka : 不必要って？

root : kbanner を作るには、 kbanner.o と font1.o と font2.o と font3.o が必要だから、 kbanner の日付がこの 4 つより新しければ、 cc kbanner.o font1.o font2.o font3.o -o kbanner も strip kbanner も実行する必要はない、っていう風に判断するんだ。ためしに touch font2.o で、 font2.o の日付を新しくしてごらん。

root : touch font2.o ですか。

```
/home/yasuoka/src/kbanner% ls -l font2.o （ぽこ）
-rw-r--r--  1 yasuoka     78764 Mar 12 15:42 font2.o
/home/yasuoka/src/kbanner% touch font2.o （ぽこ）
/home/yasuoka/src/kbanner% !l （ぽこ）
ls -l font2.o
-rw-r--r--  1 yasuoka     78764 Mar 12 16:00 font2.o
/home/yasuoka/src/kbanner% ■
```

root : それで make してごらん。

yasuoka : はい。

```
/home/yasuoka/src/kbanner% make （ぽこ）
cc kbanner.o font1.o font2.o font3.o -o kbanner
strip kbanner
/home/yasuoka/src/kbanner% !l （ぽこ）
ls -l font2.o
-rw-r--r--  1 yasuoka     78764 Mar 12 16:00 font2.o
/home/yasuoka/src/kbanner% ls -l kbanner （ぽこ）
-rwxr-xr-x  1 yasuoka    253952 Mar 12 16:01 kbanner
/home/yasuoka/src/kbanner% ■
```

ふーむ。

```
/home/yasuoka/src/kbanner% make （ぽこ）
‘kbanner’ is up to date.
/home/yasuoka/src/kbanner% ■
```

root : touch kbanner.c も試してごらん。

yasuoka : touch kbanner.c ですか。

```
/home/yasuoka/src/kbanner% touch kbanner.c （ぽこ）
/home/yasuoka/src/kbanner% make （ぽこ）
cc -c kbanner.c
cc kbanner.o font1.o font2.o font3.o -o kbanner
strip kbanner
/home/yasuoka/src/kbanner% ■
```

あ、 cc -c kbanner.c も実行された。

root : kbanner.o を作るには kbanner.c が必要だからね。いくら kbanner.o が kbanner より古かったとしても、 kbanner.c が kbanner.o より新しかったら、そこから作り直すんだ。

yasuoka : でも、これがどういう役に立つんですか？

root : 例えば kbanner.c をエディットしたりすると、 kbanner.c の日付が変わるから、それに関係のあるファイルだけを make が作り直してくれる。

yasuoka : あっ、そうか。うーん、かしこい。

root : make のかしこさは、これだけじゃないよ。途中のコマンドでエラーが出たら、実行をそこで中止しちゃうんだ。

yasuoka : エラーって？

root : 正確に言うと、エグジットステイタスが 0 以外の時。

yasuoka : エグジットステイタスが 0 以外の時に、実行を中止すると、何かいいことがあるんですか？

root : うん、エグジットステイタスが 0 以外だったら、たいていの場合ファイルはうまく出来てないからね。それ以降は実行してもまずムダなんだよ。

yasuoka : そうなんですか。

```
/home/yasuoka/src/kbanner% cat Makefile （ぽこ）
SHELL = /bin/sh

kbanner: kbanner.o font1.o font2.o font3.o
        cc kbanner.o font1.o font2.o font3.o -o kbanner
        strip kbanner

clean:
        rm -f *.o core

kbanner.o: kbanner.c
        cc -c kbanner.c

font1.o: font1.c font.h
        cc -c font1.c

font2.o: font2.c font.h
        cc -c font2.c

font3.o: font3.c font.h
        cc -c font3.c
/home/yasuoka/src/kbanner%
```

ところでさっきから気になってるんですけど、この 7 行目の cleanって何ですか？

root : ためしに clean を作ってごらん。

yasuoka : はい。

```
/home/yasuoka/src/kbanner% make clean （ぽこ）
rm -f *.o core
/home/yasuoka/src/kbanner% ls （ぽこ）
Makefile        font.h          font2.c         kbanner
README          font1.c         font3.c         kbanner.c
/home/yasuoka/src/kbanner%
```

あれ、 clean なんてファイル出来てませんけど。それに font1.o とかが消えちゃった。

root : もう一度やってごらん。

yasuoka : もう一度？

```
/home/yasuoka/src/kbanner% make clean （ぽこ）
rm -f *.o core
/home/yasuoka/src/kbanner% ls （ぽこ）
Makefile        font.h          font2.c         kbanner
README          font1.c         font3.c         kbanner.c
/home/yasuoka/src/kbanner%
```

やっぱり出来ません。

root : うん、その cleanってのはダミーのターゲットなんだ。

yasuoka : ダミーのターゲット？

root : font1.o とか kbanner.o とかのファイルは、 kbanner が出来てしまったらもう必要ないだろう？

yasuoka : はい。

root : そこで、そういう不要なファイルを消すコマンドを実行する代わりに、こういう cleanってターゲットを書いておいて、 make clean を実行させるようにするんだ。見ての通り「clean を作るには何も必要なくて、作るためのコマンドは rm -f *.o core」って書いてあるけど、そんなコマンド

実行しても font1.o とかのファイルを消去するだけで、 clean なんてファイルは出来ない。だから何度実行しても、 rm -f *.o core でファイル掃除をするだけで、何も作らないんだ。

yasuoka : うーん、わかったような、わからないような。

root : じゃ、もう一つ installってターゲットを付け加えてみようか。

```
SHELL = /bin/sh

kbanner: kbanner.o font1.o font2.o font3.o
        cc kbanner.o font1.o font2.o font3.o -o kbanner
        strip kbanner

clean:
        rm -f *.o core

install: kbanner
        cp kbanner $$HOME/bin/kbanner

kbanner.o: kbanner.c
        cc -c kbanner.c

font1.o: font1.c font.h
        cc -c font1.c

font2.o: font2.c font.h
        cc -c font2.c

font3.o: font3.c font.h
        cc -c font3.c
```

（間）

yasuoka : できました。

root : じゃ、 make install を実行してごらん。

yasuoka : はい。

```
/home/yasuoka/src/kbanner% make install （ぽこ）
cc -c kbanner.c
cc -c font1.c
cc -c font2.c
cc -c font3.c
cc kbanner.o font1.o font2.o font3.o -o kbanner
strip kbanner
cp kbanner $HOME/bin/kbanner
/home/yasuoka/src/kbanner% ■
```

root : もう一度。

yasuoka : はい。

```
/home/yasuoka/src/kbanner% make install （ぽこ）
cp kbanner $HOME/bin/kbanner
/home/yasuoka/src/kbanner% ■
```

あ、今度は cp だけだ。

root : install を作るには kbanner が必要だけど、今の kbanner は十分新しいからね。もう strip kbanner とかは実行されない。でも installってファイルは、何度 make install しても出来ないから、何度でも cp kbanner $HOME/bin/kbanner が実行される。

yasuoka : だいたいわかりました。

```
/home/yasuoka/src/kbanner% make install （ぽこ）
cp kbanner $HOME/bin/kbanner
/home/yasuoka/src/kbanner% ■
```

でも、これちょっと変です。

root : 変って？

yasuoka : だって Makefile の 11 行目は、 cp kbanner $$HOME/bin/kbanner になってるのに、実行されたコマンドでは $ は一つです。

root : うん、いいところに気が付いたね。実行するコマンドの中に $ がある時には、 Makefile の中では $$ と書かなきゃいけないんだ。

yasuoka : え？ じゃ、プロセス番号の $$ が欲しい時には？

root : 当然 $$$$ となる。ま、細かいことは、また次回話すから、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。