情報工学実験I-3
Make,CVSの使い方

055702B 池野谷克俊
担当教員:當間愛晃

実験日 2006年5月6日
提出期限日 2006年5月8日
提出した日2006年5月8日

1 問題

Level 1.1

上記サンプルプログラムをダウンロードし，make を実行せよ．make 実行時，端末上にその動作が表示されるはずである．表示された内容を記録し，その内容が何を意味するのか説明せよ

Level 1.2

どれか 1 つ以上のソースファイルを編集・保存し，make を実行せよ（コンパイルが通るのであれば編集内容は問わない）．編集対象毎に make の動作は異なるはずである．どのファイルを編集するとどう動作するのかを確認せよ．

Level 1.3

touch コマンドにより 1 つ以上のソースファイルのタイムスタンプを更新し，make を実行せよ．この時の make の動作は Level 1.2 と比較して同じはずである．何故このような動作になっているのか考察せよ．

Level 2.1

サンプルには幾つかの Makefile 例があり，makefile2 は詳細動作まで記述した例である．makefile2 の source や command を削除し，make を実行せよ．この例では全く同一の動作になっているはずである．これは make に推論機能が実装されているためである．各自削除した source / command について，具体的に何が推論により補われているのかを示せ．

Level 3.1

余り（%計算）を求める関数 mod(x,y) を mod.c として作成し，main 関数内でその関数を実行するように修正せよ． 動作確認をした後で，新規に追加すべきファイル（mod.c）を追加しつつ，修正した sample.c の修正バージョンの 2 点を，新バージョンとして CVS に登録せよ． なお，レポートには cvs commit 時に出力されたログを示すこと．

Level 3.2

現在，make-sample プロジェクトには以下に示すバージョンが含まれているはずである.
1. 初期バージョン (ダウンロードしたままのもの. 多少の修正はされてても良い)
2. 割り算を追加したバージョン
3. 余剰計算を追加したバージョン
上記の 3 バージョン時のソースファイル一式を取り出せ. レポートには取り出した際のコマンドと, その出力結果を引用すること.

2 解答及び考察

2.1 Level1.1

1. 解答

実行結果

```
[j05002@make-sample]% make
cc    -c -o sample.o sample.c
cc    -c -o add.o add.c
cc    -c -o multi.o multi.c
cc   sample.o add.o multi.o   -o sample
```

2. 考察

   - 実行結果の 2〜4 行目は cc というコンパイラを使い,'.c' ファイルから'.o' ファイル (オブジェクトファイル) を作成している.
     5 行目は sample.o, add.o, multi.o の 3 つのファイルを統合し, 実行ファイルを作成している.

2.2 Level1.2

1. 解答 sample.c の変数の値を変更してもう一度 make を実行してみた

実行結果

```
[j05002@make-sample]% make
cc    -c -o sample.o sample.c
cc   sample.o add.o multi.o   -o sample
```

2. 考察

   - ファイルの内容を変更して make を実行した場合, 変更されたファイルが再度コンパイルされる. その他のファイルはコンパイルされない. そして再度実行ファイルを作成する.

2.3 Level1.3

1. 解答

実行結果

```
[j05002@make-sample]% touch multi.c
[j05002@make-sample]% make
cc    -c -o multi.o multi.c
cc   sample.o add.o multi.o   -o sample
[j05002@make-sample]% touch multi.o
[j05002@make-sample]% make
cc   sample.o add.o multi.o   -o sample
```

2. 考察

   - touch コマンドは指定したファイルやディレクトリのタイム・スタンプを変更するコマンドである.touch コマンドを使い'.c' ファイルのタイム・スタンプを変更して,make を実行すると変更した'.c' ファイルを再度コンパイルし, 実行ファイルを作成する.
     一方,touch コマンドを使い, オブジェクトファイルのタイムスタンプを変更した場合は実行ファイルを再度作成するだけである. これは, 実行ファイルがオブジェクトファイルに依存し, オブジェクトファイルが'.c' ファイルに依存しているためである.

2.4 Level2.1

1. 解答 makefile2 の multi.o の部分を削除して以下のようなソースにして実行してみた

ソースコード

```
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c

clean:
        rm -f *.o sample
```

実行結果

```
[j05002@make-sample]% make
cc    -c -o sample.o sample.c
cc    -c -o add.o add.c
cc    -c -o multi.o multi.c
cc   sample.o add.o multi.o   -o sample
```

2. 考察

   - コンパイルし,multi.o を作成させるコマンドを削除しても make はうまく実行できている. この結果から, 実行ファイルを作成する時に足りない multi.o を,multi.c をコンパイルして作成するという作業を推論により補われているということになる.

2.5 Level3.1

1. 解答

実行結果

```
[j05002@make-sample]% cvs add mod.c
cvs add: scheduling file ‘mod.c’ for addition
cvs add: use ’cvs commit’ to add this file permanently

cvs commit をする際にいろいろ間違えたまま cvs commit してしまったので間違
いを修正して cvs commit しなおしました

[j05002@make-sample]% cvs commit
cvs commit: Examining .
Checking in mod.c;
/Users/j05002/CVS_DB/make-sample/mod.c,v  <--  mod.c
new revision: 1.2; previous revision: 1.1
done
Checking in multi.c;
/Users/j05002/CVS_DB/make-sample/multi.c,v  <--  multi.c
new revision: 1.4; previous revision: 1.3
done
Checking in sample.c;
/Users/j05002/CVS_DB/make-sample/sample.c,v  <--  sample.c
new revision: 1.4; previous revision: 1.3
done
```

2. 考察

   - cvs add で mod.c を追加し,make を実行するために sample.c, Makefile, makefile1, makefile2 を少し変更した. その後 make を実行し, うまく実行できたので cvs commit で更新した.multi.c が更新されているのは, 間違えて少し multi.c を変更してしまったためである. その後間違っていたことに気づき multi.c, mod.c, sample.c を修正して cvs commit しなおした.

2.6 Level3.2

1. 解答

実行結果-初期バージョン-

```
[j05002@make-sample]% cvs checkout -r 1.1.1.1 -p make-sample
cvs checkout: Updating make-sample
===================================================================
Checking out make-sample/0readme.txt
RCS:  /Users/j05002/CVS_DB/make-sample/0readme.txt,v
```

```
VERS: 1.1.1.1
***************
[make-sample: 0readme.txt]

makefile1: 最もシンプルな例
makefile2: 詳動作細例
makefile3: マクロ例 + clean
=============================================================================
Checking out make-sample/Makefile
RCS:  /Users/j05002/CVS_DB/make-sample/Makefile,v
VERS: 1.1.1.1
***************
sample: sample.o add.o multi.o
=============================================================================
Checking out make-sample/add.c
RCS:  /Users/j05002/CVS_DB/make-sample/add.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>

/* add: x と y 和を返す */
int add(int x, int y){
  return (x+y);
}
=============================================================================
Checking out make-sample/makefile1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile1,v
VERS: 1.1.1.1
***************
#最もシンプルな例
sample: sample.o add.o multi.o
=============================================================================
Checking out make-sample/makefile2
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2,v
VERS: 1.1.1.1
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
clean:
        rm -f *.o sample
=============================================================================
Checking out make-sample/makefile2_1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2_1,v
VERS: 1.1.1.1
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
multi.o: multi.c
          $(CC) -c multi.c

clean:
        rm -f *.o sample
=============================================================================
Checking out make-sample/makefile3
RCS:  /Users/j05002/CVS_DB/make-sample/makefile3,v
VERS: 1.1.1.1
```

```
***************
# マクロ例 + clean
CC      = gcc
CFLAGS  = -Wall -O2
OBJS    = sample.o add.o multi.o

sample: $(OBJS)

clean:
        rm -f $(OBJS) *~ sample
===================================================================
Checking out make-sample/multi.c
RCS:  /Users/j05002/CVS_DB/make-sample/multi.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>

/* multi: x と y の積を返す */
int multi(int x, int y){
  return (x*y);
}
===================================================================
Checking out make-sample/sample.c
RCS:  /Users/j05002/CVS_DB/make-sample/sample.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>

// 関数宣言
int add(int x, int y);
int multi(int x, int y);

int main()
{
  int x, y;

  x = 4;
  y = 5;
  fprintf(stdout,"add(%d,%d)=%d\n",x,y,add(x,y));
  fprintf(stdout,"multi(%d,%d)=%d\n",x,y,multi(x,y));
}
```

実行結果-割り算を追加したバージョン-

```
[j05002@make-sample]% cvs checkout -D "2006-04-29" -p make-sample
cvs checkout: Updating make-sample
===================================================================
Checking out make-sample/0readme.txt
RCS:  /Users/j05002/CVS_DB/make-sample/0readme.txt,v
VERS: 1.1.1.1
***************
[make-sample: 0readme.txt]

makefile1: 最もシンプルな例
makefile2: 詳動作細例
makefile3: マクロ例 + clean
===================================================================
Checking out make-sample/Makefile
RCS:  /Users/j05002/CVS_DB/make-sample/Makefile,v
VERS: 1.1.1.1
***************
sample: sample.o add.o multi.o
===================================================================
Checking out make-sample/add.c
RCS:  /Users/j05002/CVS_DB/make-sample/add.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>

/* add: x と y 和を返す */
int add(int x, int y){
  return (x+y);
}
```

```
===================================================================
Checking out make-sample/makefile1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile1,v
VERS: 1.1.1.1
***************
#最もシンプルな例
sample: sample.o add.o multi.o
===================================================================
Checking out make-sample/makefile2
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2,v
VERS: 1.1.1.1
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
clean:
        rm -f *.o sample
===================================================================
Checking out make-sample/makefile2_1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2_1,v
VERS: 1.1.1.1
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
multi.o: multi.c
          $(CC) -c multi.c

clean:
        rm -f *.o sample
===================================================================
Checking out make-sample/makefile3
RCS:  /Users/j05002/CVS_DB/make-sample/makefile3,v
VERS: 1.1.1.1
***************
# マクロ例 + clean
CC      = gcc
CFLAGS  = -Wall -O2
OBJS    = sample.o add.o multi.o

sample: $(OBJS)

clean:
        rm -f $(OBJS) *~ sample
===================================================================
Checking out make-sample/multi.c
RCS:  /Users/j05002/CVS_DB/make-sample/multi.c,v
VERS: 1.2
***************
#include <stdio.h>

/* multi: x と y の積を返す */
int multi(int x, int y){
  return (x*y);
}

/* divide: x/y を計算 */
int divide(int x, int y){
```

```
  return (x/y);
}
===================================================================
Checking out make-sample/sample.c
RCS:  /Users/j05002/CVS_DB/make-sample/sample.c,v
VERS: 1.2
***************
#include <stdio.h>

// 関数宣言
int add(int x, int y);
int multi(int x, int y);
int divide(int x, int y);

int main()
{
  int x, y;

  x = 4;
  y = 5;
  fprintf(stdout,"add(%d,%d)=%d\n",x,y,add(x,y));
  fprintf(stdout,"multi(%d,%d)=%d\n",x,y,multi(x,y));
  fprintf(stdout,"divide(%d,%d)=%d\n",x,y,divide(x,y));
}
```

実行結果-余剰計算を追加したバージョン-

```
[j05002@make-sample]% cvs checkout -D "2006-05-06 20:51:29" -p
% make-sample
cvs checkout: Updating make-sample
===================================================================
Checking out make-sample/0readme.txt
RCS:  /Users/j05002/CVS_DB/make-sample/0readme.txt,v
VERS: 1.1.1.1
***************
[make-sample: 0readme.txt]

makefile1: 最もシンプルな例
makefile2: 詳動作細例
makefile3: マクロ例 + clean
===================================================================
Checking out make-sample/Makefile
RCS:  /Users/j05002/CVS_DB/make-sample/Makefile,v
VERS: 1.2
***************
sample: sample.o add.o multi.o mod.o
===================================================================
Checking out make-sample/add.c
RCS:  /Users/j05002/CVS_DB/make-sample/add.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>

/* add: x と y 和を返す */
int add(int x, int y){
  return (x+y);
}
===================================================================
Checking out make-sample/makefile1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile1,v
VERS: 1.2
***************
#最もシンプルな例
sample: sample.o add.o multi.o mod.o
===================================================================
Checking out make-sample/makefile2
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2,v
VERS: 1.2
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2
```

```
sample: sample.o add.o multi.o mod.o
        $(CC) sample.o add.o multi.o mod.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
multi.o: multi.c
         $(CC) -c multi.c
mod.o: mod.c
       $(CC) -c mod.c
clean:
        rm -f *.o sample
===================================================================
Checking out make-sample/makefile2_1
RCS:  /Users/j05002/CVS_DB/make-sample/makefile2_1,v
VERS: 1.1.1.1
***************
# 詳動作細例
CC      = gcc
CFLAGS  = -Wall -O2

sample: sample.o add.o multi.o
        $(CC) sample.o add.o multi.o -o sample

sample.o: sample.c
          $(CC) -c sample.c
add.o: add.c
          $(CC) -c add.c
multi.o: multi.c
          $(CC) -c multi.c

clean:
        rm -f *.o sample
===================================================================
Checking out make-sample/makefile3
RCS:  /Users/j05002/CVS_DB/make-sample/makefile3,v
VERS: 1.1.1.1
***************
# マクロ例 + clean
CC      = gcc
CFLAGS  = -Wall -O2
OBJS    = sample.o add.o multi.o

sample: $(OBJS)

clean:
        rm -f $(OBJS) *~ sample
===================================================================
Checking out make-sample/mod.c
RCS:  /Users/j05002/CVS_DB/make-sample/mod.c,v
VERS: 1.2
***************
#include <stdio.h>

/* mod: x/y を計算 */
int mod(int x, int y){
  return (x%y);
}
===================================================================
Checking out make-sample/multi.c
RCS:  /Users/j05002/CVS_DB/make-sample/multi.c,v
VERS: 1.4
***************
#include <stdio.h>

/* multi: x と y の積を返す */
int multi(int x, int y){
  return (x*y);
}

/* divide: x と y の割り算の商を返す */
int divide(int x, int y){
  return (x/y);
```

```
}
===================================================================
Checking out make-sample/sample.c
RCS:  /Users/j05002/CVS_DB/make-sample/sample.c,v
VERS: 1.4
***************
#include <stdio.h>

// 関数宣言
int add(int x, int y);
int multi(int x, int y);
int divide(int x, int y);
int mod(int x, int y);

int main()
{
  int x, y;

  x = 4;
  y = 5;
  fprintf(stdout,"add(%d,%d)=%d\n",x,y,add(x,y));
  fprintf(stdout,"multi(%d,%d)=%d\n",x,y,multi(x,y));
  fprintf(stdout,"divide(%d,%d)=%d\n",x,y,divide(x,y));
  fprintf(stdout,"mod(%d,%d)=%d\n",x,y,mod(x,y));
}
```

2. 考察

   - 任意のバージョンの内容を呼び出す場合は
     cvs checkout -r バージョン -p プロジェクト名
     とすればよい.-p オプションを外せば, ファイル復元ができる.
     任意の日時の内容を呼び出す場合は
     cvs checkout -D ” 日時” -p プロジェクト名
     とすればよい. 例えば 2006 年 4 月 28 日 12 時の内容を呼び出す場合は,” 日時” の所を”2006-04-28 12:00:00” とすればよい.

3 感想

今回の実験は前の実験に比べれば楽だった.make は一年の時に C 言語でやったことがあるので, けっこうスラスラ進んだが,CVS は知らなかったので, けっこう手こずった. いろいろ試してみるうちに CVS の有効性がちょっと分かってきた.

参考文献

[1] NAL 研
http://www.eva.ie.u-ryukyu.ac.jp/t̃nal/

[2] バージョン管理システム CVS を使う
http://radiofly.to/nishi/cvs/cvs.html