情報工学実験Ⅰ

「Make , CVS の使い方」

担当教員名: 赤嶺 有平

提　出　日: ２０１０年6月1日
学 籍 番 号: ０９５７０７B
氏　　名: 大城　佳明

Level 1.1

上記演習１－１，１－２の実行結果を記録し，考察せよ.

演習 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.

<Makefile>

```
01 sample: sample.c add.c multi.c
02         gcc -o sample sample.c add.c multi.c
```

<実行結果>

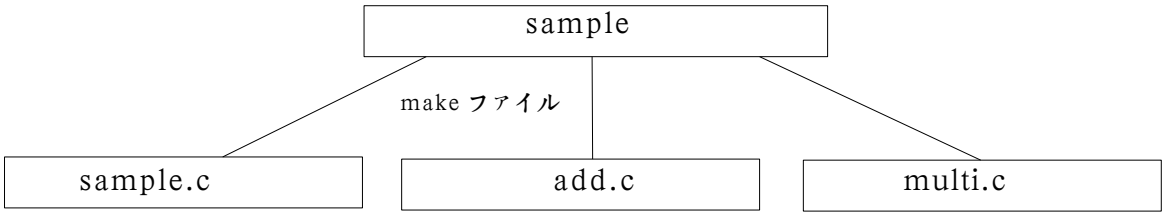
```
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
02 gcc -o sample sample.c add.c multi.c
```

<考察>

1.一行のみ表示された。

2.make とコマンドを打つと、「Makefile」が実行される

3. make で起きた事をかんたんに図で表した。



ソース（C 言語）

4.「gcc」とは C 言語をコンパイルする時に使うコマンドである

5.「-o」は出力するファイルを指定している。

6. sample という make フィルが作成されている

7.「sample.c add.c multi.c」は合体するプログラムファイルである。

演習 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）

<実行結果>

```
 ① add.c のソースファルを編集・保存した。
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
02 gcc -o sample sample.c add.c multi.c

 ②編集・保存しない
03 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
04 make: `sample' is up to date.
```

<考察>

1.① ソースを編集・保存したものは最初と同じ結果が表れた

2.② 編集・保存していないものはコンパイルされなかった

3.変更されなかったファイルには「up to date」と出る

4.これより、ソーフの中が変更なければ新しい make フィアルは作成されな事がわかる

Level 1.2

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．挙動の変化を確認し，何故このような動作になっているのか考察せよ

<実行結果>

```
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch add.c
02 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
03 gcc -o sample sample.c add.c multi.c
```

<考察>

1.touch コマンドで add.c のファイルが保持している最終アクセス日時と最終更新日時を変更できる

2.実行結果より、最終アクセス日時と最終更新日時を変更することで、make ファイルを作成出来る

3.最終アクセル日時と最終変更日時どちらを見て make ファイルが作られているのかを調べてみた

<3.実行結果>

```
  ①最終アクセス日時の変更
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -a add.c
02 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
03 make: `sample' is up to date.

  ②最終変更日時の変更
04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -m add.c
```

```
05 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
06 gcc -o sample sample.c add.c multi.c
```

<3.考察>

A.01～03 は最終アクセス日時の変更した結果を示している

B.「touch -a」は最終アクセス日時を変更するコマンドである

C.03 より make ファイルが作成されなかったことがわかる

D.04～06 は最終変更日時の変更した結果を示している

E.「touch -m」は最終変更日時を変更するコマンドである

F.06 より最終変更日時を変更を行う事により、make ファイルが作成される事がわかった

4.ファイルの最終変更日時を過去にしてみたらどうなるかやってみた

<4.実行結果>

```
  ①実行前の最終変更日時
01 -rw-r--r-- 1 yoshiaki  staff   109  5 27 17:55 add.c

02 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271750 add.c

   ②変更後の最終変更日時
03 -rw-r--r-- 1 yoshiaki  staff   109  5 27 17:50 add.c

04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
05 make: `sample' is up to date.
```

<4.考察>

A.01 より変更前は最終変更日時は「5/27 17:55」だとわかる

B.02 の「touch -mt （日付時間）」で最終変更日時を変更出来る

C.03 より「5/27 17:50」に変更されたのがわかる

D.05 より make ファイルは作成されなかった事がわかる

E.したがって、最終変更日時を変えるだけでは作成されない事がわかる。

5.次に sample（実行ファイル）の最終変更日時を過去にしてみた

<5.実行結果>

```
  ①実行前の最終変更日時
01 -rwxr-xr-x  1 yoshiaki  staff  12652  5 27 17:55 sample

02 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271050 sample

  ②変更後の最終変更日時
03 -rwxr-xr-x  1 yoshiaki  staff  12652  5 27 10:50 sample
```

```
04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
05 gcc -o sample sample.c add.c multi.c
```

<5.考察>
A.01 より変更前は最終変更日時は「5/27 17:55」だとわかる

B.03 より「5/27 10:50」に変更されたのがわかる

C.05 より make ファイルが作成出来たことがわかる

D.実行ファイルの日時を過去にすると make ファイルが作成できる

6.以上の結果より make ファイルを作成するとき実行ファイルの最終変更日時と合体するファイルの最終変更日時を比較して、作成するかどうかの判断をしていると思われる

<6.実行結果>

```
① sample が最終変更日時が遅い
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271050 sample
02 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271051 add.c
03 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
04 gcc -o sample sample.c add.c multi.c

① add.c が最終変更日時が遅い
05 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271050 sample
06 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ touch -mt 05271049 add.c
07 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make
08 make: `sample' is up to date.
```

<6.考察>
A.01 ~ 04 は sample の方が変更日時を遅めに設定した結果である

B.最終変更日時を　sample は「5/27 10:50」　add.c は「5/27 10:51」　にした

C.04 より作成出来たのがわかる

D.05 ~ 08 は add.c の方が変更日時を遅めに設定した結果である

E.最終変更日時を　sample は 5/27 10:50」　add.c は「5/27 10:49」　にした

F.08 より作成出来なかったのがわかる

G.実行ファイルが合体するファイルよりも古かったら更新される仕組みになっている

7.以上の事よりまとめる

A.make ファイルは最終変更日時で判断している

B.実行ファイルと合体するファイルの変更日時を比べている

C.実行ファイルが合体するファイルよりも古かったら make ファイルが作成される

Level2

演習 2 における make の挙動を記録し考察せよ

演習 2-1

makefile2 は，ファイル間の依存関係も含めて詳細に記述した例である．-f オプションで makefile2 を指定し make を実行せよ．

<makefile2>

```
01 # 詳動作細例

02 sample: sample.o add.o multi.o
03        gcc sample.o add.o multi.o -o sample

04 sample.o: sample.c arithmetic.h
05          gcc -c sample.c
06 add.o: add.c arithmetic.h
07          gcc -c add.c
08 multi.o: multi.c arithmetic.h
09          gcc -c multi.c
```

<実行結果>

```
01 oshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
02 gcc -c sample.c
03 gcc -c add.c
04 gcc -c multi.c
05 gcc sample.o add.o multi.o -o sample
```

<考察>

1.実行結果 02 ~ 04 よりファイルはそれぞれコンパイルされたを示している

2.「-c」は中間ファイルの事を示している

3.05 ではコンパイルして出来たオブジェクトファイルをすべてリンクさせている

4.「sample」が実行ファイルとなる

演習 2-2

makefile2 を用いて，演習 1 と同様に，ファイルの編集後，make の挙動がどう変化するか観察せよ．特に arithmetic.h を編集した際の挙動に注目する．

<実行結果>

```
  ① sample.c を編集
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
02 gcc -c sample.c
03 gcc sample.o add.o multi.o -o sample

  ② add.c を編集
04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
```

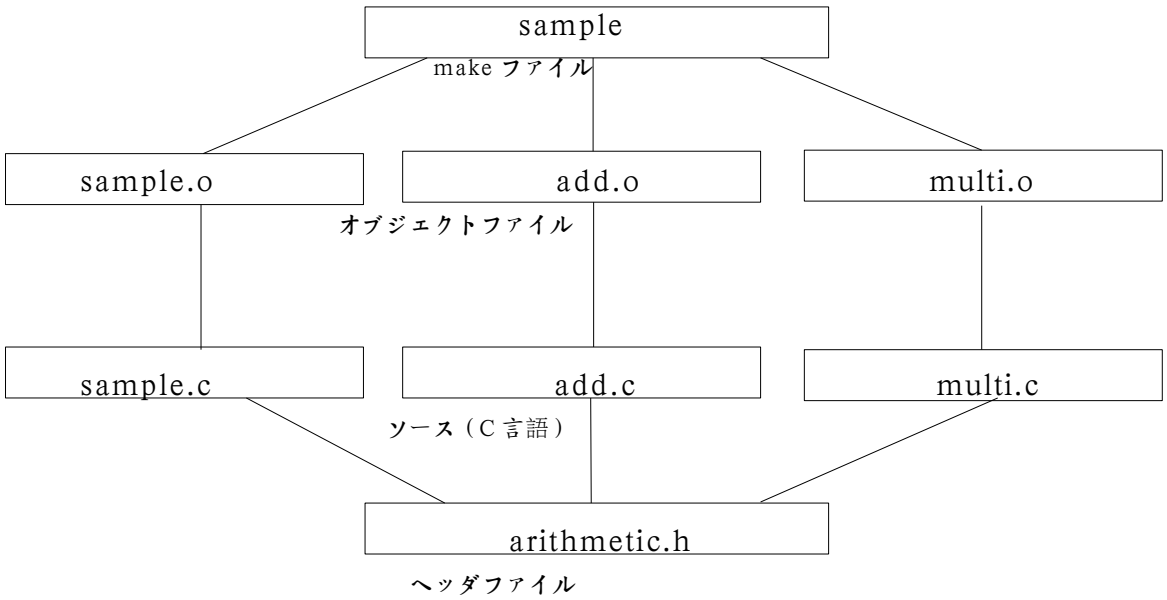
```
05 gcc -c add.c
06 gcc sample.o add.o multi.o -o sample

  ③ multi.c を編集
07 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
08 gcc -c multi.c
09 gcc sample.o add.o multi.o -o sample

   ④ arithmetic.h を編集
10 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
11 gcc -c sample.c
12 gcc -c add.c
13 gcc -c multi.c
14 gcc sample.o add.o multi.o -o sample
```

<考察>

1.ファイルの関係性を図にした



2.sample.c を編集すると、02 より sample.c だけがコンパイルされているのがわかる

3.add.c、muti.c の時も同じである。

4.変更されたファイルだけがコンパイルされているのがわかる

5.コンパイルした後は変更したファイルのオブジェクトファイルを合体させる必要がある

6.arithmetic.h を変更すると 11,12,13 よりすべてのファイルがコンパイルされている

7.3 つのファイルはすべて arithmetic.h が含まれているためだと思われる

8.arithmetic.h のファイルが変更されるとそれに依存している sample.c、add.c、multi.c も変更された事になるからである

Level 3

演習 3 の実行結果を記録し，考察せよ．

演習 3-1

makefile3 を用いて make を実行し，挙動を確認せよ．各ソースファイルを，touch でタイムスタンプを更新し，make の挙動の変化を観察せよ．

<makefile3>

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

05 sample: $(OBJS)
06         $(CC) -o $@ $(OBJS)

07 .c.o:
08         $(CC) $(CFLGAS) -c $<

09 sample.o: arithmetic.h
10 add.o: arithmetic.h
11 multi.o: arithmetic.h
```

<実行結果>

```
  ① sample.c を編集
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile3
02 gcc -c sample.c
03 gcc sample.o add.o multi.o -o sample

  ② add.c を編集
04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile3
05 gcc -c add.c
06 gcc sample.o add.o multi.o -o sample

  ③ multi.c を編集
07 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile3
08 gcc -c multi.c
09 gcc sample.o add.o multi.o -o sample

  ④ arithmetic.h を編集
10 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile2
11 gcc -c sample.c
12 gcc -c add.c
13 gcc -c multi.c
14 gcc sample.o add.o multi.o -o sample
```

<考察>

1 .実行結果は makefile2 と変わらない

2 .makefile3 のソースは makefile2 より簡潔に書かれている

3.数が多くなるとmakefile3の方がいい

演習3-2

上記のcleanは大抵のmakefileに定義されている内容である（コマンドやその引数は必要に応じてマクロを使うことが多い）．これは何をしているのか．makefile3に類似の機能を記述し，実行せよ．その際，clean実行前後で何がどう変わるのかを確認しなさい．

<makefile3>（追加）

```
12 clean:
13   <tub>   rm -f a.out *.o *~
```

<実行結果>

```
  ①実行前
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ ls
02 0readme.txt  add.o           makefile3       multi.o         sample.o
03 Makefile     arithmetic.h    makefile3~      sample
04 add.c                makefile2       multi.c         sample.c

  ②実行
05 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ make -f makefile3 clean
06 rm -f a.out *.o *~

  ③実行後
07 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ ls
08 0readme.txt  add.c           makefile2       multi.c         sample.c
09 Makefile     arithmetic.h    makefile3       sample
```

<考察>

1.ソールより、「rm」はファイルの削除である

2.「a.out」「*.o」「*~」を削除するという意味になる

3.実行結果より実行前と実行後を比べる

A.「add.o」「multi.o」「sample.o」が消えている。

B.「makefile3~」が消えている

4.以上の事より「*.o」「*~」が削除されている事がわかる

5.05ではファイル名の後ろにtargetを引数として指定しているためclean以外は実行されていない

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．

<mod.c>

```
01 #include <stdio.h>
02 #include "arithmetic.h"
03 /* mod: x と y の余剰を返す */
04 int mod(int x, int y){
05   return (x % y);
06 }
```

<sample.c>（追加）

```
13   fprintf(stdout,"mod(%d,%d)=%d\n",x,y,mod(x,y));
```

<arithmetic.h>（追加）

```
03 int mod(int x, int y);
```

<makefile3>（変更）

```
04 OBJS    = sample.o add.o multi.o mod.o
```

<実行結果>

```
  ① mod.c の追加
01 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ cvs add mod.c
02 cvs add: scheduling file `mod.c' for addition
03 cvs add: use `cvs commit' to add this file permanently

  ② commit
04 yoshiaki-oshiro-no-macbook-2:make-sample yoshiaki$ cvs commit
05 cvs commit: Examining .
06 /Users/yoshiaki/CVS_DB/make-sample/arithmetic.h,v  <--  arithmetic.h
07 new revision: 1.2; previous revision: 1.1
08 /Users/yoshiaki/CVS_DB/make-sample/makefile3,v  <--  makefile3
09 new revision: 1.2; previous revision: 1.1
10 /Users/yoshiaki/CVS_DB/make-sample/mod.c,v  <--  mod.c
11 initial revision: 1.1
12 /Users/yoshiaki/CVS_DB/make-sample/sample,v  <--  sample
13 new revision: 1.3; previous revision: 1.2
14 /Users/yoshiaki/CVS_DB/make-sample/sample.c,v  <--  sample.c
15 new revision: 1.3; previous revision: 1.2
```

<考察>

1.「cvs add （ファイル名）」で cvs に追加が出来る

2.02,03 より追加の準備ができたこと示している

3 .commit を使い、編集、追加したファイルを CVS_DB/make-sample に保存した
4 .11 より mod.c は 1 回目の保存だとわかる

5 .13,15 の「new revision:1.3」より sample.c、sample は 3 回目の保存だとわかる

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

<バージョン調べ>

```
revision 1.3
date: 2010-05-28 12:40:32 +0900;  author: yoshiaki;  state: Exp;  lines: +1 -0;  commitid:
g5B6Uzk7dnboSuAu;
renew2
----------------------------
revision 1.2
date: 2010-05-28 12:31:37 +0900;  author: yoshiaki;  state: Exp;  lines: +2 -1;  commitid:
QezJ1unRo47kPuAu;
renew1
----------------------------
revision 1.1
date: 2010-05-28 12:28:03 +0900;  author: yoshiaki;  state: Exp;  commitid: mSUaMljKmHt7OuAu;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2010-05-28 12:28:03 +0900;  author: yoshiaki;  state: Exp;  lines: +0 -0;  commitid:
mSUaMljKmHt7OuAu;
start
```

上記から表にまとめてみた

バージョン	コメント	revision	日時
初期バージョン	start	1.1	2010/5/28 12:28
割り算を追加したバージョン	renew1	1,2	2010/5/28 12:31
余剰計算を追加したバージョン	renew2	1,3	2010/5/28 12:40

1.初期バージョン

<実行結果>

```
01 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ cvs checkout -r 1.1 make-sample
02 cvs checkout: Updating make-sample
03 U make-sample/0readme.txt
04 U make-sample/Makefile
        <省略>
13 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ls ./make-sample/
14 0readme.txt  add.c           makefile3       sample
15 CVS          arithmetic.h    mod.c           sample.c
16 Makefile     makefile2       multi.c

17 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ./make-sample/sample
```

```
18 add(2,3)=5
19 multi(2,3)=6
```

2.割り算を追加したバーション

<実行結果>

```
01 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ cvs checkout -D "2010-05-28 12:31:37"  make-sample
02 cvs checkout: Updating make-sample
03 U make-sample/0readme.txt
        <省略>
12 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ls ./make-sample/
13 0readme.txt  Makefile        arithmetic.h    makefile3       sample
14 CVS          add.c           makefile2       multi.c         sample.c

15 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ./make-sample/sample
16 add(2,3)=5
17 multi(2,3)=6
18 divide(2,3)=0
```

3.余剰計算を追加したバージョン

<実行結果>

```
01 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ cvs checkout make-sample
02 cvs checkout: Updating make-sample
03 U make-sample/0readme.txt
        <省略>
13 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ls ./make-sample/
14 0readme.txt  add.c           makefile3       sample
15 CVS          arithmetic.h    mod.c           sample.c
16 Makefile     makefile2       multi.c

17 yoshiaki-oshiro-no-macbook-2:d1 yoshiaki$ ./make-sample/sample
18 add(2,3)=5
19 multi(2,3)=6
20 divide(2,3)=0
21 mod(2,3)=2
```

<考察>

A. 3パターンの取り方で実行した

B. 1.初期バージョンでは「cvs checkout -r <revision> <ファイル名>」で取り出した

ⅰ.revertion 1.1である

ⅱ.14~16よりすべてのファイルが取り出されたことがわかる

ⅲ.17~19より初期バージョンだとわかる

C. 2.割り算を追加したバーションでは「cvs checkout -D "日付" <ファイル名>」で取り出した

ⅰ.「2010-05-28 12:31:37」である

ii.14～16 より mod.c 以外取り出されたことがわかる

iii.これは mod.c だけ後から追加したため revesion1.2 にはないと考えられる

iv.15～18 より割り算を追加したバージョンであることがわかる

D. 3 .余剰計算を追加したバージョン(最新)では「cvs checout <ファイル名>」で取り出した

ⅰ.一番最新のが取り出される

ii.14～16 よりすべて取り出されているのがわかる

iii.17～21 より 余剰計算を追加したバージョンであることがわかる

Level 3.3

make や cvs は共同作業時により有効に機能する．しかし，cvs では checkout 時にロック制御を行わないため，「複数人が現バージョンのコピーを取得し，同じファイルに修正を加える状況（競合）」が起こり得る．この場合，cvs は誰の修正が正しいのかを判断することが出来ないため，ユーザ同士で修正箇所を確認し合い，人手により競合を解決する（正しい修正を通知する）必要がある．実際にそのような状況を作り出し，実験してみよ．

～流れ～

① checkout をし、同じ revision の「make-sample1」「make-sample2」を作成する
② make-sample1 の中の sample.c の中を編集・保存する
③ make-sample2 の中の sample.c と sample の内容を編集・保存する
④ make-sample1 を commit する
⑤ make-sample2 を commit する
⑥ make-sample2 の status を見る

<実験結果>

```
  ① checkout をし、同じ revision の「make-sample1」「make-sample2」を作成
01 yoshiaki-oshiro-no-macbook-2:working yoshiaki$ ls
02 make-sample1          make-sample2


  ② make-sample1 の中の sample.c の中を編集・保存した
03 yoshiaki-oshiro-no-macbook-2:make-sample1 yoshiaki$ cvs status
       ...
04 0File: sample.c       Status: Locally Modified
       ...


  ③ make-sample2 の中の sample.c と sample の内容を編集・保存した
05 yoshiaki-oshiro-no-macbook-2:make-sample2 yoshiaki$ cvs status
       ...
06 File: sample          Status: Locally Modified
       ...
07 File: sample.c        Status: Locally Modified
       ...


  ④ make-sample1 を commit した
08 yoshiaki-oshiro-no-macbook-2:make-sample1 yoshiaki$ cvs commit
09 cvs commit: Examining .
10 /Users/yoshiaki/CVS_DB/make-sample/sample.c,v  <--  sample.c
11 new revision: 1.6; previous revision: 1.5


  ⑤ make-sample2 を commit した
12 yoshiaki-oshiro-no-macbook-2:make-sample2 yoshiaki$ cvs commit
13 cvs commit: Examining .
14 cvs commit: Up-to-date check failed for `sample.c'
15 cvs [commit aborted]: correct above errors first!


  ⑥ make-sample2 の status を見てみた
16 yoshiaki-oshiro-no-macbook-2:make-sample2 yoshiaki$ cvs status
       ...
17 File: sample          Status: Locally Modified
18
19 Working revision:    1.4     2010-05-29 11:43:29 +0900
20 Repository revision: 1.4     /Users/yoshiaki/CVS_DB/make-sample/sample,v
       ...
20  File: sample.c       Status: Needs Merge
21
22 Working revision:    1.5     2010-05-28 23:23:21 +0900
23 Repository revision: 1.6     /Users/yoshiaki/CVS_DB/make-sample/sample.c,v
       ...
```

<考察>

1.①～④までは特に変わった点はなかった

2.⑤ではエラーが発生した

A.15 より 'sample.c' が失敗した事がわかる

B.これは「make-sample1」の 'sample.c' だけ編集・保存したからだ

3.⑥で詳しく調べてみた

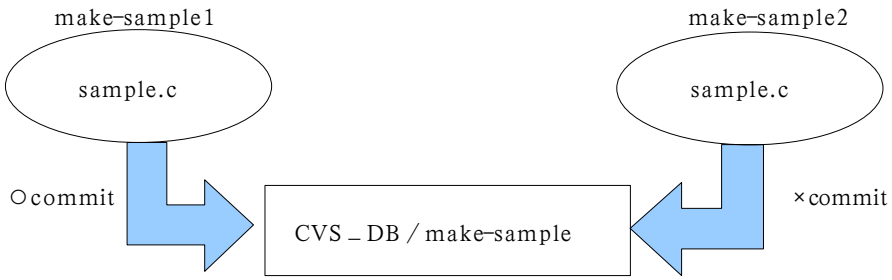
A.'sample' は変化は見られなかった

B.'sample.c' が「Status: Needs Merge」と表示されていた

C. 22「Working revision:1.5」、23「Repository revision:1.6」と revesion が一致していない

D.11 の「 new revision: 1.6; previous revision: 1.5」より revision が変わっている。

E.revesio が違うので commit が出来なかったと考えられる



4.以上の事より「make-sample2」は commit 出来なかったのである

参 考 文 献

（CVS コマンド一覧）
http://www.ne.jp/asahi/hishidama/home/tech/cvs/command.html

（任意のバージョンのプロジェクト全体を取り出す）
http://www.gfd-dennou.org/arch/cc-env/cvs/old_2006-08-29/old_2004-10-01/www-vox.dj.kit.ac.jp/nishi/cvs/cvs-08.html