情報工学実験Ⅰ
Make,cvsの使い方

実 験 日 ：2008年5月30日(金)

提 出 日 ：2008年6月 6日(金)

担当教員名：赤 嶺 有 平

学籍番号 ： 075730G

氏 名 ：澤 岨 千 明

level 1.1

演習 1-1、1-2 の実行結果を記録し、考察せよ

演習 1-1

サンプルプログラムをダウンロードし、make を実行せよ

実行結果

```
% ls
0readme.txt     add.c           makefile2       multi.c
Makefile        arithmetic.h    makefile3       sample.c
% make
gcc -o sample sample.c add.c multi.c
% ls
0readme.txt     add.c           makefile2       multi.c         sample.c
Makefile        arithmetic.h    makefile3       sample*
% ./sample
add(2,3)=5
multi(2,3)=6
```

考察

・make は Makefile を実行するコマンドである。

・make コマンドによって、sample.c、add.c、multi.c をまとめてコンパイルした後 sample という名前の実行ファイルが作られる。

・makefile2、makefile3 を実行したい場合は引数 -f をつけて実行すればよい。

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)。

変更後の sample.c のソース

```
1   #include <stdio.h>
2   #include "arithmetic.h"
3
4   int main()
5   {
6     int x, y;
7
8     x = 6;
9     y = 9;
10    fprintf(stdout,"add(%d,%d)=%d\n",x,y,add(x,y));
11    fprintf(stdout,"multi(%d,%d)=%d\n",x,y,multi(x,y));
12  }
```

実行結果

```
% make
gcc -o sample sample.c add.c multi.c
[C-T:~/make-sample] e075730% ./sample
add(6,9)=15
multi(6,9)=54
```

考察

・sample.c の x、y の値を変更したものが上のソースである。

・make を実行すると、演習 1-1 と同様にコンパイルされた。

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を考察せよ。

実行結果

```
% make
make: `sample' is up to date.
% touch sample.c
% make
gcc -o sample sample.c add.c multi.c
```

考察

・touch コマンドはファイルの修正時刻を書き換えるコマンドである。
・“make: `sample' is up to date.” とはすでにされているという意味である。
・既に最新の状態であったが、touch コマンドによってファイルの修正時刻が更新されたので、再びコンパイルできるようになったのである。

level 2

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。

実行結果

```
% make -f makefile2
gcc -c sample.c
gcc -c add.c
gcc -c multi.c
gcc sample.o add.o multi.o -o sample
% ls
0readme.txt     add.o           makefile3       sample*
Makefile        arithmetic.h    multi.c         sample.c
add.c           makefile2       multi.o         sample.o
```

考察

・makefile2 の実行結果、Makefile には無かった add.o、multi.o、sample.o というファイルが出来ている。
・sample.c、add.c、multi.c をコンパイルしてできたこれらのオブジェクト（*.o）は sample という実行ファイルへコンパイルする為に作られたものである。

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

```
% emacs -nw arithmetic.h
% make -f makefile2
gcc -c sample.c
gcc -c add.c
gcc -c multi.c
gcc sample.o add.o multi.o -o sample

% emacs -nw sample.c
% make -f makefile2
gcc -c sample.c
gcc sample.o add.o multi.o -o sample
```

考察

・通常、一つのファイルを編集したときはそのファイルのみがコンパイルされる。
・arithemetic.h はコンパイルされる三つのファイルにも関わっているため、このファイルを編集・保存したので、タイムスタンプが更新され再びコンパイルできるようになった。

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。

実行結果

```
% make -f makefile3
gcc  -c sample.c
gcc  -c add.c
gcc  -c multi.c
gcc -o sample sample.o add.o multi.o
% make -f makefile3
make: `sample' is up to date.
% touch sample.c
% make -f makefile3
gcc  -c sample.c
gcc -o sample sample.o add.o multi.o
% touch *.c
% make -f makefile3
gcc  -c sample.c
gcc  -c add.c
gcc  -c multi.c
gcc -o sample sample.o add.o multi.o
```

考察
・sample は最新状態なので touch コマンドでタイムスタンプを更新し再度 make コマンドを入力した。
・ここで、sample.c のみを更新した場合と全てを更新した状態でコンパイルしてみた。

演習 3-2
　　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事が多い）。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かを確認しなさい。

makefile3 のソース

```
1   # マクロ例 + clean
2   CC      = gcc
3   CFLAGS  = -Wall -O2
4   OBJS    = sample.o add.o multi.o
5
6   sample: $(OBJS)
7           $(CC) -o $@ $(OBJS)
8
9   .c.o:
10          $(CC) $(CFLGAS) -c $<
11
12  sample.o: arithmetic.h
13  add.o: arithmetic.h
14  multi.o: arithmetic.h
15
16  clean:
17          rm -f *.o *~
```

実行結果

```
% ls
0readme.txt     add.o           makefile2       multi.c         sample.c
Makefile        arithmetic.h    makefile3       multi.o         sample.c~
add.c           arithmetic.h~   makefile3~      sample*         sample.o

% make -f makefile3 clean
rm -f *.o *~
```

```
% ls
0readme.txt     add.c           makefile2       multi.c         sample.c
Makefile        arithmetic.h    makefile3       sample*
```

考察
・makefile3 で clean に“rm -f *.o *~”を定義した。
・makefile3 を実行する時に clean を入力することで、“rm -f *.o *~”が実行され、オブジェクト（*.o）とバックアップファイル（*~）が削除されるようになっている。
・実行時、編集時に生成されるオブジェクトやバックアップファイルを消す時に使われる。

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 の追加バージョンの 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 のソース

```
1   #include <stdio.h>
2   #include "arithmetic.h"
3
4   /* mod:x/y の余りを求める */
5   int mod(int x, int y){
6     return(x%y);
7   }
```

変更後の sample.c

```
1   #include <stdio.h>
2   #include "arithmetic.h"
3
4   int divide(int x, int y);
5   int main()
6   {
7     int x, y;
8
9     x = 2;
10    y = 3;
11    fprintf(stdout,"add(%d,%d)=%d\n",x,y,add(x,y));
12    fprintf(stdout,"multi(%d,%d)=%d\n",x,y,multi(x,y));
13    fprintf(stdout,"divide(%d,%d)=%d\n",x,y,divide(x,y));
14    fprintf(stdout,"mod(%d,%d)=%d\n",x,y,mod(x,y));
15
16  }
```

変更後の Makefile

```
1   sample: sample.c add.c multi.c mod.c
2           gcc -o sample sample.c add.c multi.c mod.c
```

実行結果

```
% make
gcc -o sample sample.c add.c multi.c mod.c
% ./sample
add(2,3)=5
multi(2,3)=6
divide(2,3)=0
mod(2,3)=2

% cvs add mod.c
cvs add: scheduling file `mod.c' for addition
cvs add: use 'cvs commit' to add this file permanently

% cvs commit
cvs commit: Examining .
```

```
Checking in Makefile;
/Users/e075730/CVS_DB/make-sample/Makefile,v  <--  Makefile
new revision: 1.2; previous revision: 1.1
done
RCS file: /Users/e075730/CVS_DB/make-sample/mod.c,v
done
Checking in mod.c;
/Users/e075730/CVS_DB/make-sample/mod.c,v  <--  mod.c
initial revision: 1.1
done
Checking in sample;
/Users/e075730/CVS_DB/make-sample/sample,v  <--  sample
new revision: 1.3; previous revision: 1.2
done
Checking in sample.c;
/Users/e075730/CVS_DB/make-sample/sample.c,v  <--  sample.c
new revision: 1.3; previous revision: 1.2
done

% cvs add sample.c mod.c
cvs add: sample.c already exists, with version number 1.3
cvs add: mod.c already exists, with version number 1.1
```

考察

・mod.c は x/y の余剰を求めるファイルで、sample.c、Makefile は mod.c の分を新たに記述したものである。

・make で正常に働いていることがわかる。

・“cvs add mod.c”で mod.c の情報を追加保存した。

・“cvs commit”で編集したファイルを検出して保存している。

・Makefile、sample、sample.c は新たなバージョンに変更され、mod.c は今回初めて追加されたのでバージョンが 1.1 である。

・最後にもう一度“cvs add”コマンドを行った。どちらも既に存在しており、バージョンも更新されていることがわかる。

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

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。

実行結果 1.初期バージョン

```
% cvs checkout -r 1.1.1.1 -p make-sample
cvs checkout: Updating make-sample
===================================================================
Checking out make-sample/0readme.txt
RCS:  /Users/e075730/CVS_DB/make-sample/0readme.txt,v
VERS: 1.1.1.1
***************
[make-sample: 0readme.txt]

makefile1: ?G??Y????
makefile2: ??u?????
makefile3: ?  ?????
===================================================================
Checking out make-sample/Makefile
RCS:  /Users/e075730/CVS_DB/make-sample/Makefile,v
VERS: 1.1.1.1
***************
sample: sample.c add.c multi.c
        gcc -o sample sample.c add.c
multi.c=============================================================
```

```
Checking out make-sample/add.c
RCS:  /Users/e075730/CVS_DB/make-sample/add.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>
#include "arithmetic.h"

/* add: x??y?¤???¸ */
int add(int x, int y){
  return (x+y);
}
==============================================================
Checking out make-sample/arithmetic.h
RCS:  /Users/e075730/CVS_DB/make-sample/arithmetic.h,v
VERS: 1.1.1.1
***************
int add(int x, int y);
int multi(int x, int y);
==============================================================
Checking out make-sample/makefile2
RCS:  /Users/e075730/CVS_DB/make-sample/makefile2,v
VERS: 1.1.1.1
***************
# ??ư?????

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

sample.o: sample.c arithmetic.h
          gcc -c sample.c
add.o: add.c arithmetic.h
          gcc -c add.c
multi.o: multi.c arithmetic.h
          gcc -c multi.c

==============================================================
Checking out make-sample/makefile3
RCS:  /Users/e075730/CVS_DB/make-sample/makefile3,v
VERS: 1.1.1.1
***************
# ?  ????? + clean
CC      = gcc
CFLAGS  = -Wall -O2
OBJS    = sample.o add.o multi.o

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

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

sample.o: arithmetic.h
add.o: arithmetic.h
multi.o: arithmetic.h

clean:
        rm -f *.o *~
==============================================================
Checking out make-sample/multi.c
RCS:  /Users/e075730/CVS_DB/make-sample/multi.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>
#include "arithmetic.h"
```

```
/* multi: x??y???Ⱶ???̨ */
int multi(int x, int y){
  return (x*y);
}
============================================================
Checking out make-sample/sample
RCS:  /Users/e075730/CVS_DB/make-sample/sample,v
VERS: 1.1.1.1
***************
????
?__cstring__TEXT?p?__textcoal_nt__TEXT??
                                        ?__DATA __data__DATA
__dyld__DATA__common__DATA  4?__IMPORT0 __pointers__IMPORT0
__jump_table__IMPORT0?8__LINKEDIT@0\
                                                              /usr/lib/dyld
                                                                        4
??FH
X/usr/lib/libSystem.B.dylib0"?1?
                                ??P?j???????]?\$??L$????¯\??U??WVS??,?}
                                                                   ?]??
                                                                        ? ?
???u?????? ????????u??p?0???t?0???t?????E??D$?$??]?U??E?D$?$??G?E???$??0??t$
                                                                  ?\?|
$??$??$?^??h?% ??% U??S??4?*?E??E??E?D$?E??$?z????:??RX?L$?E?D$
                                                 ?
E??D???D$?$???E?D$?E??$?E????:??RX?L$?E?D$

?E??D???D$?$???4[]?U???E
                  ??U????E
                           ??
__dyld_make_delayed_module_initializer_calls__dyld_mod_term_funcsadd(%d,%d)=%d
multi(%d,%d)=%d
?$?????????????????????? ?3?D?Q??

                                      ?ol?  ?$ ?( , &0 @4 [8 x< ?@ ?D ?H ?L ;
:?}z@P ]?
ci??!t'{?????*?!NXArgc_NXArgv___progname__dyld_func_lookup__start_environdyld_st
ub_binding_helperstart___keymgr_dwarf2_register_sections__cthread_init_routine__
mh_execute_header_atexit_catch_exception_raise_catch_exception_raise_state_catch
_exception_raise_state_identity_clock_alarm_reply_do_mach_notify_dead_name_do_ma
ch_notify_no_senders_do_mach_notify_port_deleted_do_mach_notify_send_once_do_seq
nos_mach_notify_dead_name_do_seqnos_mach_notify_no_senders_do_seqnos_mach_notify
_port_deleted_do_seqnos_mach_notify_send_once_errno_exit_mach_init_routine_main_
receive_samples___i686.get_pc_thunk.bx___sF_add_fprintf_multi==================
================================================
Checking out make-sample/sample.c
RCS:  /Users/e075730/CVS_DB/make-sample/sample.c,v
VERS: 1.1.1.1
***************
#include <stdio.h>
#include "arithmetic.h"

int main()
{
  int x, y;

  x = 2;
  y = 3;
  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 2. 割り算追加バージョン

```
% cvs checkout -r 1.2 -p make-sample
cvs checkout: Updating make-sample
```

```
===================================================================
Checking out make-sample/Makefile
RCS:  /Users/e075730/CVS_DB/make-sample/Makefile,v
VERS: 1.2
***************
sample: sample.c add.c multi.c mod.c
        gcc -o sample sample.c add.c multi.c
mod.c===================================================================
Checking out make-sample/multi.c
RCS:  /Users/e075730/CVS_DB/make-sample/multi.c,v
VERS: 1.2
***************
#include <stdio.h>
#include "arithmetic.h"

/* multi: x??y???H???? */
int multi(int x, int y){
  return (x*y);
}

/* divide: x/y??  ? */
int divide(int x, int y){
  return(x/y);
}
===================================================================
Checking out make-sample/sample
RCS:  /Users/e075730/CVS_DB/make-sample/sample,v
VERS: 1.2
***************
????
?__cstring__TEXTx?x__textcoal_nt__TEXT??
                                        ?__DATA __data__DATA
__dyld__DATA__common__DATA  4?__IMPORT0 __pointers__IMPORT0
__jump_table__IMPORT0?8__LINKEDIT@0p
                                                                /usr/lib/dyld
                                                                          4
??FH
X/usr/lib/libSystem.B.dylib0#?1?
                                ?P4j???????]?\$??L$????¯\??U??WVS??,?}
                                                                    ?]??
                                                                         ? ?
???u?x????< ????????u??p?0???t?0???t???9?E??D$?$|?]?U??E?D$?$??G?E???$??0??t$
                                                                  ?\?|
$??$??$????h?% ??% U??S??4??E??E??E?D$?E??$????????RX?L$?E?D$
                                        ?
E??D??f?D$?$?p?E?D$?E??$????????RX?L$?E?D$

?E??D??v?D$?$?-?E?D$?E??$?T???????RX?L$?E?D$
                                   ?E??D????D$?$????4[]?U???E
                                                        ??U????E
                                                                ??U?????E
                                                                          ?
E?ЋM???9?E?E???__dyld_make_delayed_module_initializer_calls__dyld_mod_term_func
sadd(%d,%d)=%d
multi(%d,%d)=%d
divide(%d,%d)=%d
?$?????????????????????? L3^D@Q??

                                    ?o;?  ?$ ?( , tX&0 @4 [8 x< ?@ ?D ?H ?L ;
:R?I@P ]4ci??!|'{?????*?"
NXArgc_NXArgv___progname__dyld_func_lookup__start_environdyld_stub_binding_helpe
rstart___keymgr_dwarf2_register_sections__cthread_init_routine__mh_execute_heade
r_atexit_catch_exception_raise_catch_exception_raise_state_catch_exception_raise
_state_identity_clock_alarm_reply_do_mach_notify_dead_name_do_mach_notify_no_sen
ders_do_mach_notify_port_deleted_do_mach_notify_send_once_do_seqnos_mach_notify_
```

```
dead_name_do_seqnos_mach_notify_no_senders_do_seqnos_mach_notify_port_deleted_do
_seqnos_mach_notify_send_once_errno_exit_mach_init_routine_main_receive_samples_
__i686.get_pc_thunk.bx___sF_add_divide_fprintf_multi============================
========================================
Checking out make-sample/sample.c
RCS:  /Users/e075730/CVS_DB/make-sample/sample.c,v
VERS: 1.2
***************
#include <stdio.h>
#include "arithmetic.h"

int divide(int x, int y);
int main()
{
  int x, y;

  x = 2;
  y = 3;
  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 3．余剰計算追加バージョン

```
% cvs checkout -r 1.3 -p make-sample
cvs checkout: Updating make-sample
========================================================================
Checking out make-sample/sample
RCS:  /Users/e075730/CVS_DB/make-sample/sample,v
VERS: 1.3
***************
????
    P? 8__PAGEZERO__TEXT__text__TEXT???
                                        ?__cstring__TEXTh?h__textcoal_nt__TEXT??
                                                                              ?
__DATA __data__DATA __dyld__DATA__common__DATA  4?__IMPORT0 __pointers__IMPORT0
__jump_table__IMPORT0?8__LINKEDIT@0?
              /usr/lib/dyld
                            4??FH
X/usr/lib/libSystem.B.dylib0$?1?
                                 ??P?j???????]?\$??L$????¯\??U??WVS??,?}
                                                                    ?]??
                                                                         ? ?
???u?h???? ????????u??p?0???t?0???t????E??D$?$l?]?U??E?D$?$??G?E???$?v?0??t$
                                                                  ?\?|
$??$??$?B??h?% ??% U??S??4??E??E??E?D$?E??$???????RX?L$?E?D$
                                        ?
E??D????D$?$???E?D$?E??$????????RX?L$?E?D$

?E??D????D$?$??E?D$?E??$???????RX?L$?E?D$
                                 ?E??D????D$?$?Z?E?D$?E??$?s??????RX?L$?E?D$
                                                                       ?
E??D????D$?$???4[]?U???E
         ??U????E
                 ??U?????E
                          ?E?ЋM???9?E?E???U?????E
                                                 ?
E?ЋM???9????__dyld_make_delayed_module_initializer_calls__dyld_mod_term_funcsad
d(%d,%d)=%d
multi(%d,%d)=%d
divide(%d,%d)=%d
mod(%d,%d)=%d
?$?????????????????????? ?3?D?Q??

                                     ?o?  ?$ ?( , t+&0 @4 [8 x< ?@ ?D ?H ?L ;
```

```
:??J?@P ]?ci??!|'{?????*?# !
"_NXArgc_NXArgv___progname__dyld_func_lookup__start_environdyld_stub_binding_hel
perstart___keymgr_dwarf2_register_sections__cthread_init_routine__mh_execute_hea
der_atexit_catch_exception_raise_catch_exception_raise_state_catch_exception_rai
se_state_identity_clock_alarm_reply_do_mach_notify_dead_name_do_mach_notify_no_s
enders_do_mach_notify_port_deleted_do_mach_notify_send_once_do_seqnos_mach_notif
y_dead_name_do_seqnos_mach_notify_no_senders_do_seqnos_mach_notify_port_deleted_
do_seqnos_mach_notify_send_once_errno_exit_mach_init_routine_main_receive_sample
s___i686.get_pc_thunk.bx___sF_add_divide_fprintf_mod_multi======================
==========================================
Checking out make-sample/sample.c
RCS:  /Users/e075730/CVS_DB/make-sample/sample.c,v
VERS: 1.3
***************
#include <stdio.h>
#include "arithmetic.h"

int divide(int x, int y);
int main()
{
  int x, y;

  x = 2;
  y = 3;
  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));

}
```

考察

・“cvs checout -r バージョン -p 呼び出すもの”コマンドでバージョンファイルを呼び出し出力させた。

・初期バージョンは 1.1.1.1 である。新たに追加されたものは 1.1、そして更新される度に 1.2、1.3、と値が大きくなっていく。

・“Checking out”はそのファイルの元の場所、“RCS”更新されたファイルの情報、“VERS”はバージョン、そしてソースを出力している。

・実行ファイルはともかく、あるファイルで文字化けがでたが、直せなかったので放置した。

感想

make は 1 年ときにもやっていたが忘れていたのでもう一度やれて良かった。

cvs は多少理解できなかったが、使いこなすと非常に便利なので覚えておきたい。