Development Environment 2

make

Jo, Heeseung

make

Definition

- make is utility to maintain groups of programs

Object

- If some file is modified, make detects it and update files related with modified one

make basic rules

3 components

- Target, dependency, command

File format

- File name: Makefile
- Format

```
target: dependency
[tab] command
```

make example

make 를 사용하지 않을때

- gcc -c main.c
- gcc -c add.c
- gcc -c sub.c
- gcc -o test main.o add.o sub.o

- 혹은, gcc -o test main.c add.c sub.c

Makefile example

Makefile (대소문자 유의)

```
all: main.o add.o sub.o
	gcc -o test main.o add.o sub.o
main.o: addsub.h main.c
	gcc -c main.c
add.o: add.c
	gcc -c add.c
sub.o: sub.c
	gcc -c sub.c
```

반드시 tab으로 입력

make example

어느 때 어떤 target이 다시 컴파일 될 것인가?

- main.c가 바뀌었을 경우
- add.c 혹은 sub.c가 바뀌었을 경우
- addsub.h가 바뀌었을 경우

make clean

To clean up

- make clean

```
all: main.o add.o sub.o
        gcc -o test main.o add.o sub.o
main.o: addsub.h main.c
        gcc -c main.c
add.o: add.c
        gcc -c add.c
sub.o: sub.c
        gcc -c sub.c

clean:
        rm *.o test
```

내부변수 사용

```
CC=gcc
SRC=main.c add.c sub.c
main: ${SRC}
        ${CC} -o test ${SRC}
```

미리 정해져 있는 내부변수

make -p 를 통하여 모든 값들을 확인 가능

- ASFLAGS : as 명령어의 옵션 세팅
  - AS = as
- CFLAGS : gcc 의 옵션 세팅
  - CC = cc (= gcc)
- CPPFLAGS : g++ 의 옵션
  - CXX = g++
- LDFLAGS : ld(linker) 의 옵션 세팅
  - LD = ld
- LFLAGS : lex 의 옵션 세팅
  - LEX = lex
- YFLAGS : yacc 의 옵션 세팅
  - YACC = yacc

Ex: Makefile 작성

실습목표

- 앞서 코딩한 99단 프로그램을 build 할 수 있는 Makefile 을 작성

Simple Makefile

```
CFLAGS = -Wall -O -g

bin=hello
t1=main
t2=funcs
obj=$(t1).o $(t2).o

all: $(bin)
$(bin): $(obj)
        $(CC) $(obj) -o $@
clean:
        rm -f $(bin) *.o
```

$(bin)을 나타내는 내부변수

```
CFLAGS = -Wall -O -g

bin=hello
t1=main
t2=funcs
obj=main.o funcs.o

all: hello
hello:  main.o funcs.o
        gcc main.o funcs.o -o hello
clean:
        rm -f hello *.o
```

Ex: Makefile 작성

실습목표

- 앞서 코딩한 $1^2 + 2^2 + 3^2 + ... + n^2$ 을 출력하는 프로그램을 build 할 수 있는 Makefile 을 작성
- 위의 예제를 이용하여 만들어 볼 것
- make, make all, make clean 이 정상적으로 동작하는지 확인
  - make 와 make all 은 동일 함

** 이후 모든 실습과 프로그램 작성시 Makefile을 만들어서 make로 빌드 할 것

GNU DeBugger (GDB)

GDB 기본 사용법

gcc 컴파일시 -g 옵션 추가 (-O 옵션은 제거)

gdb 실행

- gdb a.out
- gdb --args a.out 1 2 3 4

```
root@iter1:/tmp> gdb a.out
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /tmp/a.out...done.
(gdb)
```

GDB 기본 사용법

종료

- q / ctrl+d

소스 찾아가기 (list)

- l : main 함수를 기점으로 소스의 내용이 출력된다
- l 10 : 10 행 주변의 소스가 출력
- l func : func 함수의 소스를 출력
- l a.c:func : a.c 파일의 func 함수부분을 출력
- l a.c:10 : a.c 파일의 10행을 기준으로 출력

GDB 기본 사용법

중단점 사용하기 (breakpoint)

- b func : func 함수에 브레이크 포인트 설정
- b 10 : 10행에 브레이크 포인트 설정
- b a.c:func : a.c파일의 func함수에 브레이크 포인트 설정
- b a.c:10 : a.c파일의 10행에 브레이크 포인트 설정
- b +2 : 현재 행에서 2개 행 이후 지점에 브레이크 포인트 설정
- b -2 : 현재 행에서 2개 행 이전 지점에 브레이크 포인트 설정
- b *0x8049000 : 0x8049000 주소에 브레이크 포인트 설정 (어셈블리로 디버깅 시 사용)
- b 10 if var == 0 : 10행에 브레이크 포인트를 설정해되, var 변수 값이 0일 때 작동

GDB 기본 사용법

중단점 삭제하기 (clear, delete)

- cl func : func 함수의 시작 부분에 브레이크 포인트 지움
- cl 10 : 10행의 브레이크 포인트 지움
- cl a.c:func : a.c 파일의 func함수의 브레이크 포인트 지움
- cl a.c:10 : a.c 파일의 10행의 브레이크 포인트 지움
- cl : 모든 브레이크 포인트 지움

GDB 기본 사용법

프로그램 실행, 종료 (run, kill)

- r : 프로그램 수행 (재시작)
- r arg1 arg2 : arg1과 arg2를 인자로 프로그램 수행
- k : 프로그램 수행종료

역추적하기 (backtrace)

- bt : 오류가 발생한 함수를 역으로 찾아감

GDB 기본 사용법

변수 정보보기 (info, print)

- info locals : 현재 상태에서 어떤 지역변수들이 있으며, 값은 어떠한지를 알 수 있음
- info variables : 현재 상태에서의 전역변수 리스트를 확인
- p lval : lval 값을 확인
- p func : func 함수의 주소값을 확인
- p pt : pt가 구조체라면 구조체의 주소를 확인
- p *pt : pt가 구조체라면 구조체의 값을 확인
- p **pt : *pt가 구조체라면 구조체의 값을 확인
- info registers : 레지스트 값 전체를 한번에 확인

GDB 기본 사용법

디버깅 하기 (step, next, continue, until, finish, return, step instruction, next instruction)

- s : 현재 출력된 행을 수행하고 멈추지만, 함수의 경우 함수의 내부로 들어가서 수행
- s 5 : s를 5번 입력한 것과 동일
- n : 현재 행을 수행하고 멈추지만, 함수의 경우 함수를 수행하고 넘어감
- n 5 : n을 5번 입력한 것과 동일
- c : 다음 브레이크 포인트를 만날때 까지 계속 수행
- u : for 문에서 빠져나와서 다음 브레이크 포인트까지 수행

Ex: gdb

실습목표

- 다음 프로그램을 정상적인 결과값이 나오도록 debugging 하시오
- gcc -g -o a.out a.c

```
# include <stdio.h>

void main()
{
        int i, num, j;
        printf ("Enter the number: ");
        scanf ("%d", &num );

        for (i=1; i<num; i++)
                j=j*i;

        printf("The factorial of %d is %d\n", num, j);
}
```

Ex: gdb

실습목표

- 다음 프로그램은 왜 segment fault를 만드는지 설명하시오
- gcc -g -o a.out a.c

```
#include <stdio.h>

void main()
{
        char *temp= "Paras";

        int i;
        i=0;

        temp[0]='F';

        for (i=0 ; i < 5 ; i++ )
                printf("%c\n", temp[i]);
}
```

Ex: gdb

실습목표

- 다음 프로그램은 왜 segment fault를 만드는지 설명하시오
- gcc -g -o a.out a.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
        char *buf;

        buf = malloc(1<<31);
        strcpy(buf, "This is Test");
        printf("%s\n", buf);

        return 0;
}
```

Ex: gdb

실습목표

- 다음 프로그램이 정상적으로 동작하도록 수정하시오
- gcc -g -o a.out a.c

```
#include <stdio.h>

int main () {
        int x = 0;
        int inp = 0;
        int nums[20];
        int idx = 0;
        for (;;)         {
                printf ("Enter an integer: ");
                inp = scanf ("%d", &x);
                if (inp == 0) {
                        printf ("Error: not an integer\n");
                }
                else {
                        if (idx < 20) {
                                nums[idx] = x;
                                idx++;
                        }
                }
        }
        return 0;
}
```

Ex: gdb

실습목표

- 다음 프로그램이 정상적으로 동작하도록 수정하시오
- gcc -g -o a.out a.c

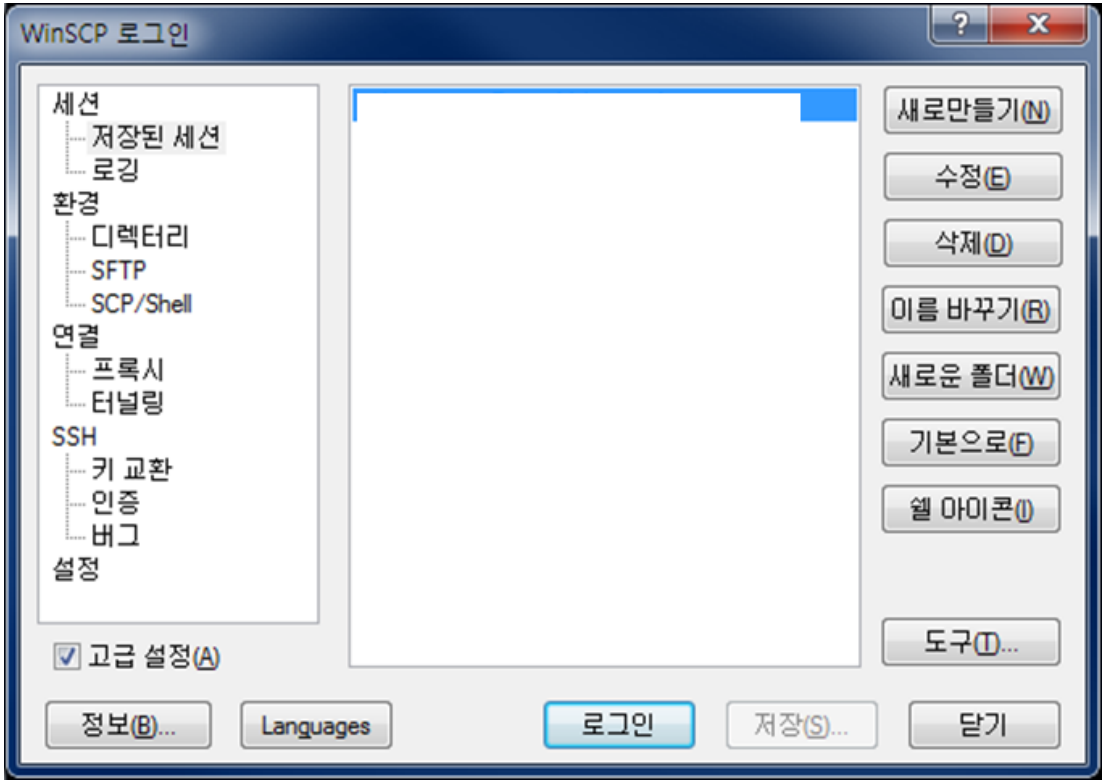
```
#include <stdio.h>

double fun(int i) {
  volatile double d[1] = {3.14};
  volatile long int a[2];
  a[i] = 1073741824;
  return d[0];
}

int main () {
        int i;
        double k;
        for (i=0; i<6 ; i++)
        {
                k=fun(i);
                printf("%f \n", k);
        }
        return 0;
}
```

File transfer using WinSCP

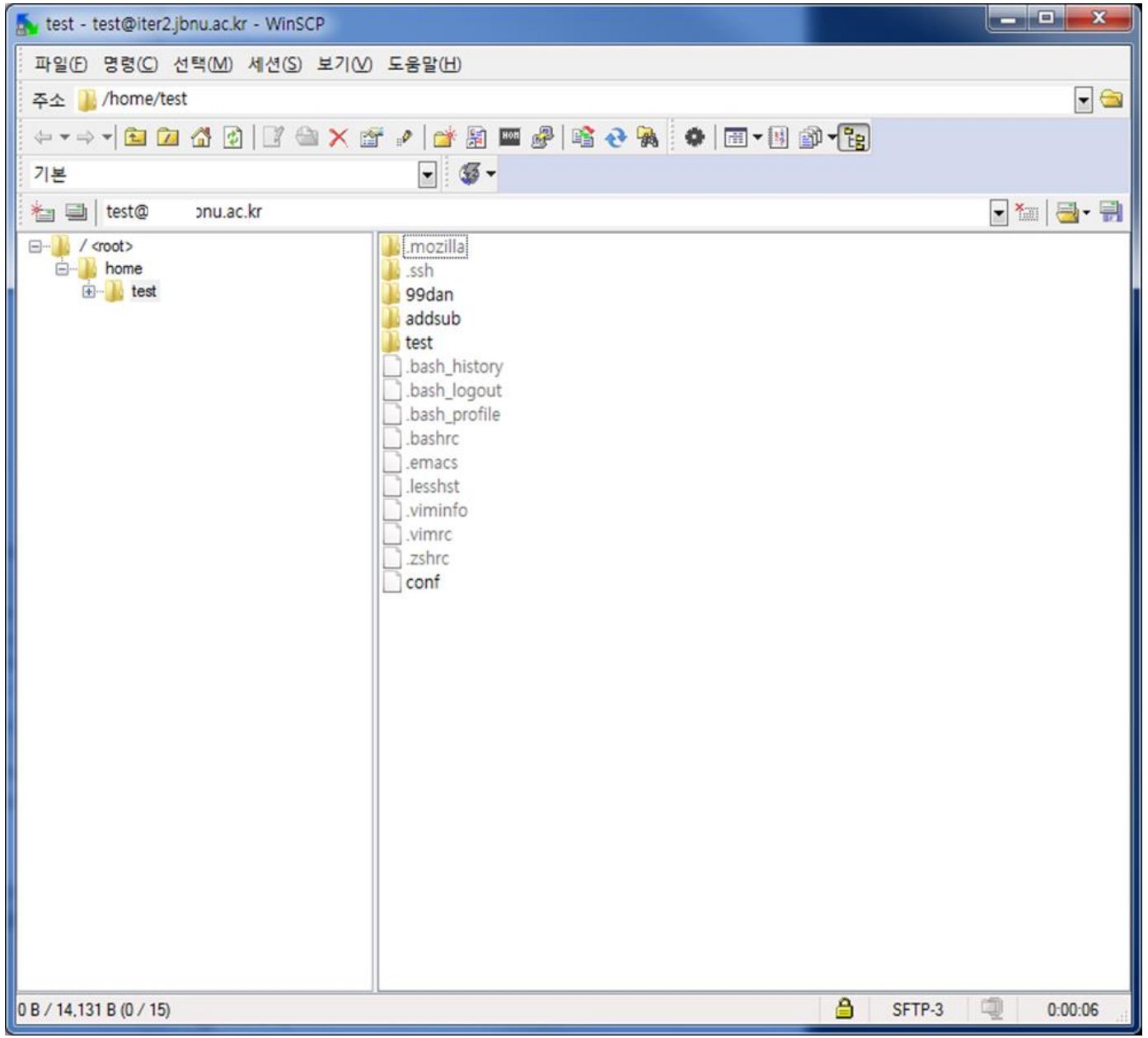
WinSCP

http://winscp.net

ssh를 이용한 파일 전송 tool



WinSCP


test - test@iter2.jbnu.ac.kr - WinSCP
파일(F) 명령(C) 선택(M) 세션(S) 보기(V) 도움말(H)
주소 /home/test
기본
test@ ɔnu.ac.kr
/ <root>
home
test
.mozilla
.ssh
99dan
addsub
test
.bash_history
.bash_logout
.bash_profile
.bashrc
.emacs
.lesshst
.viminfo
.vimrc
.zshrc
conf
0 B / 14,131 B (0 / 15)
SFTP-3
0:00:06

Exercise

Ex.

- 앞서 제작한 구구단 프로그램의 폴더를 모두 압축하여 Windows로 전송하자
- tar cvfz gugu.tar.gz gugu
- zip -r gugu.zip gugu

- WinSCP를 이용하여 다운로드