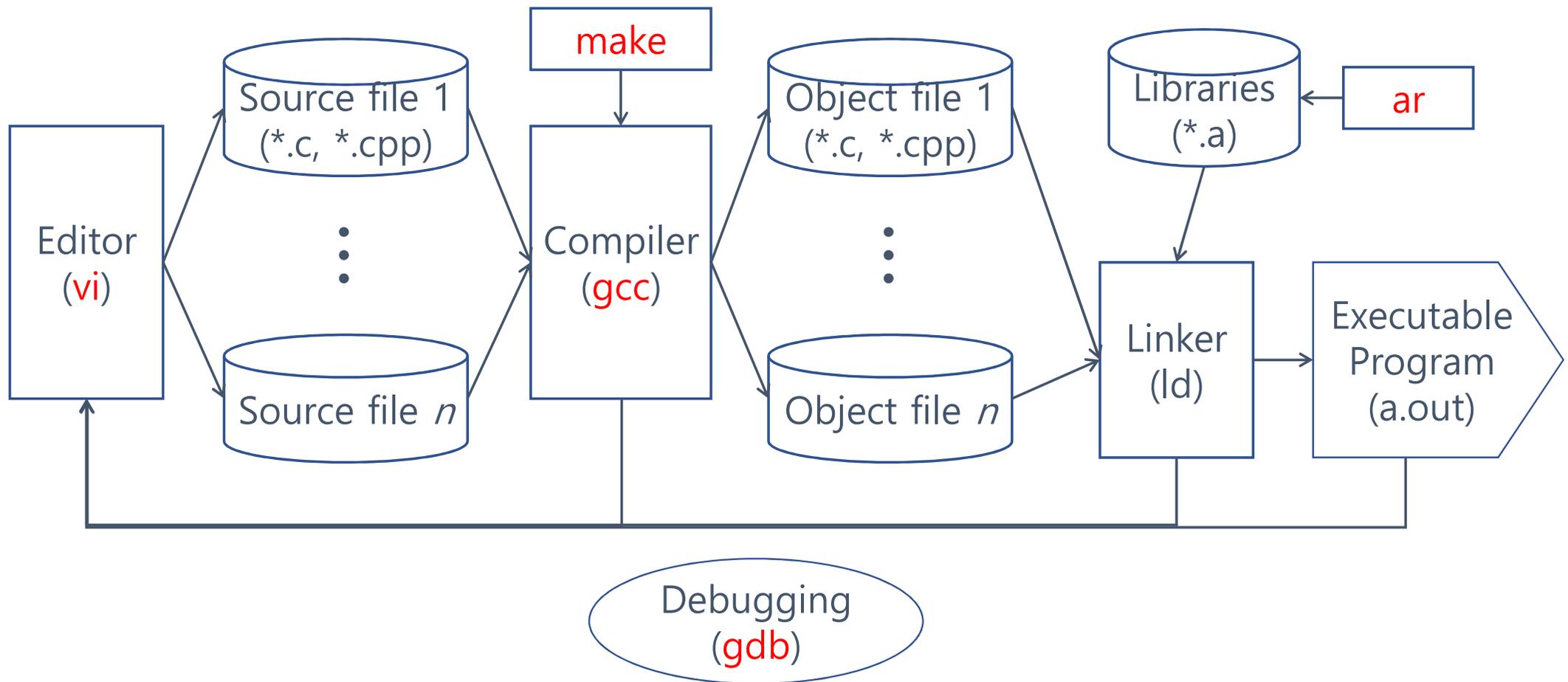




Development Tools

Program Development



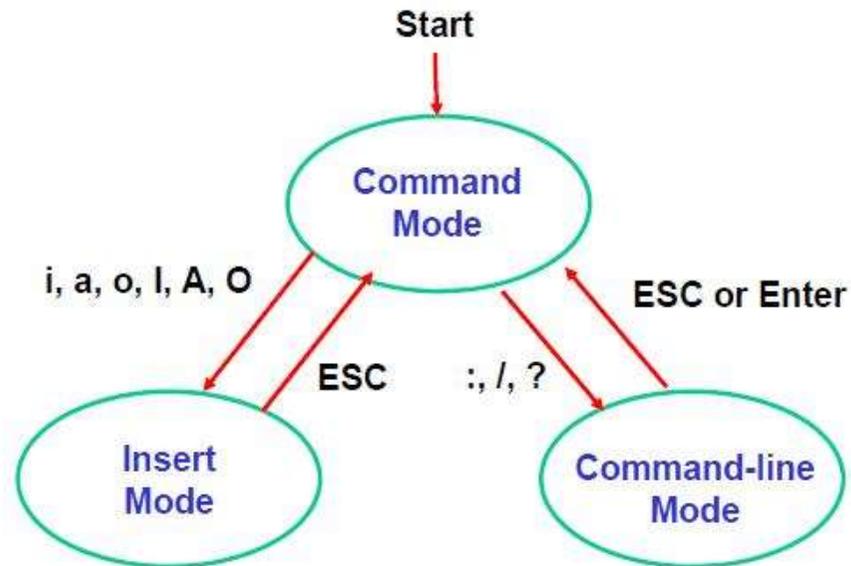


vi/vim – Text Editor

- Text editor
 - ✓ System configuration files
 - ✓ Program source files
- You have to be skillful with vi/vim
 - ✓ If you want to be a system expert
 - ✓ If you want to get good grade in this class

vi/vim

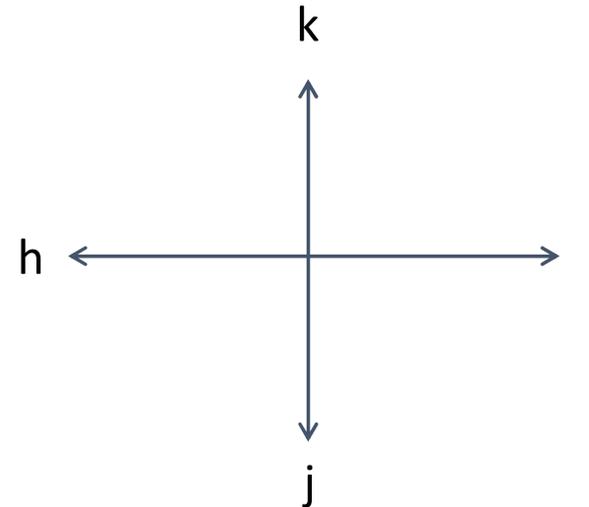
- Execution
 - ✓ `$vi main.c`
- Termination
 - ✓ `ZZ`
 - ✓ `:wq`
 - ✓ `:x`
 - ✓ `:q!`
- Mode
 - ✓ Command mode
 - ✓ Command line mode
 - ✓ Insert mode



vi/vim

▪ Cursor movement in command mode

- ✓ l : right
- ✓ h : left
- ✓ j : down
- ✓ k : up
- ✓ \$: end of line
- ✓ ^ : begin of line
- ✓ ^F : next page
- ✓ ^B : previous page
- ✓ ^D : next half-page
- ✓ ^U : previous half-page
- ✓]] : next function
- ✓ [[: previous function





vi/vim

- Edit command in command mode
 - ✓ x : delete a character
 - ✓ dw : delete a word
 - ✓ dd : delete a line
 - ✓ 5dd : delete 5 lines
 - ✓ yy : yank a line (copy)
 - ✓ 3yy : yank 3 lines
 - ✓ p : put (paste)
 - ✓ r : replace a character
 - ✓ cw : change a word
 - ✓ ~ : convert to upper-case character (and vice versa)
 - ✓ J : join two lines to one line
 - ✓ u : un-do
 - ✓ . : repeat

vi/vim

- Command in command-line mode
 - ✓ :w : write
 - ✓ :r filename : read & insert a file
 - ✓ :e filename : edit a new file
 - ✓ : n : edit next file (when vi *.c)
 - ✓ /string : search string in forward direction
 - ✓ ?string : search string in backward direction
 - ✓ n : search next string in forward direction
 - ✓ N : search next string in backward direction
 - ✓ :1, \$s/cnt/count/g : substitute one string with another
 - ✓ :10, .w! filename : write into a new file
 - ✓ :., +10y : yank (copy)
 - ✓ :!ls : execute shell command ls
 - ✓ :help [word] : help about word
- Miscellaneous command in command mode
 - ✓ ^L : refresh
 - ✓ ^G : summary



vi/vim

- Settings in command-line mode
 - ✓ **: set option [=value]**

 - ✓ Options
 - ai, noai : autoindent
 - sm, nosm : show matching paranthesis
 - nu, nonu : show line number
 - showmode, noshowmode : show mode
 - Tabstop [=value] : tab size

 - ✓ Default settings: ~/ .exrc

 - ✓ Examples
 - :set tabstop=4
 - :set ai

gcc – GNU C Compiler

- Usage

- ✓ `$ gcc [options] source-files`

- Options

- ✓ `-c` : compile only
 - ✓ `-Dname [=def]` : define a symbol *name*
 - ✓ `-g` : for debug
 - ✓ `-help` : for help
 - ✓ `-Ipathname` : add *pathname* for #include files
 - ✓ `-llibrary` : link with *library*
 - ✓ `-L directory` : add *directory* for library directories
 - ✓ `-o output-file` : name the output file *output-file*
 - ✓ `-O[level]` : optimize the object code
 - ✓ `-S` : produce an assembly source file
 - ✓ `-temp=directory` : set *directory* for temporary files
 - ✓ `-w` : do not print warnings
 - ✓ `-static` : static linking (dynamic linking by default)



gcc

▪ Files

- ✓ a.out : executable output file (by default)
- ✓ file.a : library of object files (archive file)
- ✓ file.c : C source file
- ✓ file.C .cc .cpp : C++ source file
- ✓ file.i : C source file after preprocessing
- ✓ file.o : object file
- ✓ file.s : assembler source file
- ✓ /usr/include : standard directory for #include files
- ✓ /usr/lib/libc.a : standard C library
- ✓ /usr/lib/libm.a : mathematics library

gcc

- Examples

- ✓ \$ gcc -help
- ✓ \$ gcc test.c
- ✓ \$ gcc -o test test.c
- ✓ \$ gcc -c test1.c test2.c
- ✓ \$ gcc -o test test.c test1.o test2.o libtmp.a
- ✓ \$ gcc -DDEBUG=1 -g debug.c
- ✓ \$ gcc -I./include -L./lib math.c -lm
- ✓ \$ gcc -O -S -temp=. asm.c
- ✓ \$ gcc -static static.c

- Exercise (<http://inslab.sch.ac.kr/~kimdy/lec/tool.html>)

- ✓ \$ gcc hello.c mystrcpy.c
- ✓ \$./a.out
- ✓ \$ gcc -o hello hello.c mystrcpy.c
- ✓ \$./hello

gcc

[mystrcpy.c]

```
#include <string.h>

void mystrcpy(char *dst,
              char *src)
{
    while (*src) {
        *dst++ = *src++;
    }
    *dst = *src;
}
```

[hello.c]

```
#include <stdio.h>

main()
{
    char str[80];

    mystrcpy(str, "Hello");
    puts(str);
}
```

ar – Library Archives

- Usage
 - ✓ **\$ ar [options] archive [member-files]**
- Options
 - ✓ d : delete
 - ✓ m : move to the end of the archive
 - ✓ p : print
 - ✓ q : quick append
 - ✓ r : replace
 - ✓ t : table of contents
 - ✓ x : extract
- Modifiers
 - ✓ c : create
 - ✓ u : update
 - ✓ v : verbose
 - ✓ *a*position-name : place new files after *position-name*
 - ✓ *b*position-name : place new files before *position-name*



ar

- Examples
 - ✓ \$ ar -help
 - ✓ \$ ar rcv archive file.o
 - ✓ \$ ar rav next.o archive file.o
 - ✓ \$ ar xv archive file.o
 - ✓ \$ ar tv archive

- Exercise (<http://inslab.sch.ac.kr/~kimdy/lec/tool.html>)
 - ✓ \$ gcc -c mystrcpy.c
 - ✓ \$ ar ruv libmine.a mystrcpy.o
 - ✓ \$ gcc -o hello -L. hello.c -lmine
 - ✓ \$./hello

make – Maintain Related Programs

- Usage
 - ✓ **\$ make [-f makefile] [options] [target] [macro definitions]**
 - ✓ Macro definition: **name=string**
 - ✓ Default makefile: Makefile or makefile

- Options
 - ✓ -d : debug mode
 - ✓ -e : environment variables override assignments within makefiles
 - ✓ -i : ignore error codes
 - ✓ -s : silent mode
 - ✓ -t : touch target files

- ✓ Cf) \$ touch [-c] file-name



make

- Makefile lines
 - ✓ Dependency line
target: [prerequisites] [; command]
 - ✓ Command line
tab [@] command
 - ✓ Macro definition
name = string
 - ✓ Include statement
include file-name
 - ✓ Comment
this is comment

make

- Basic Makefile

```
prog : main.o input.o output.o
```

```
    gcc -o prog main.o input.o output.o
```

```
main.o : main.c
```

```
    gcc -c main.c
```

```
input.o : input.c header.h
```

```
    gcc -c input.c
```

```
output.o : output.c
```

```
    gcc -c output.c
```

```
clean :
```

```
    rm -f core *.o prog
```



make

- Internal macros

- ✓ `$?` : the list of prerequisites that are newer than the target
- ✓ `$@` : the name of the current target
- ✓ `$<` : the name of the current prerequisite
- ✓ `$*` : the name –without the suffix- of the current prerequisite
- ✓ `$%` : the name of the corresponding .o file

- Special target names

- ✓ `.IGNORE:`
- ✓ `.SILENT:`
- ✓ `.SUFFIXES:`

make

```
# This is a typical Makefile
CC = gcc
CFLAGS = -I../inc -g
LDFLAGS = -L../lib -lm

TARGET = qsim
OBJS = qsim.o event.o queue.o

.SUFFIXES : .c .o

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

$(TARGET) : $(OBJS)
    $(CC) -o $@ $(OBJS) $(LDFLAGS)

qsim.o : qsim.c ../inc/common.h
event.o : event.c header.h
queue.o : queue.c header.h

clean :
    rm -f core $(TARGET) $(OBJS)
```

make

- Exercise

(<http://inslab.sch.ac.kr/~kimdy/lec/tool.html>)

- ✓ \$ make
- ✓ \$./hello

[Makefile]

```
CC = gcc
CFLAGS =
LDFLAGS = -L. -lmine
TARGET = hello
OBJS = hello.o

.SUFFIXES : .c .o

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

$(TARGET) : $(OBJS) libmine.a
    $(CC) -o $@ $(OBJS) $(LDFLAGS)

libmine.a : mystrcpy.o
    ar ruv libmine.a mystrcpy.o

clean :
    rm -rf libmine.a $(TARGET) *.o
```

gdb – GNU Source-Level Debugger

- Usage
 - ✓ **\$ gdb [options] obj-file**

- Options
 - ✓ -h : for help
 - ✓ -x *file* : execute GDB commands from *file*
 - ✓ -d *directory* : add *directory* for source files
 - ✓ -q : do not print the introductory and copyright messages

gdb

- Execution commands

- ✓ `break [file:]line-number` : breakpoint at *file:line-number*
- ✓ `break [file:]function` : breakpoint at *file:function*
- ✓ `run [arglist]` : run with *arglist*
- ✓ `print expression` : print *expression* for one time
- ✓ `display expression` : display *expression* forever
- ✓ `undisplay` : stop to display
- ✓ `c[ont]` : continue to run
- ✓ `n[ext] [n]` : execute next *n* lines
- ✓ `s[tep]` : step into next line
- ✓ `help [name]` : help for *name*
- ✓ `q[uit]` : stop debugging

gdb

- Exercise

(<http://inslab.sch.ac.kr/~kimdy/lec/tool.htm>)

```
$ gcc -g debug.c
```

```
$ gdb a.out
```

```
(gdb) break 10
```

```
(gdb) run
```

```
(gdb) s
```

```
(gdb) n
```

```
(gdb) ...
```

```
(gdb) display i
```

```
(gdb) print j
```

```
(gdb) ...
```

```
(gdb) c
```

```
(gdb) quit
```

```
[bug.c]
```

```
1 #include <stdio.h>
```

```
2 #include <string.h>
```

```
3
```

```
4 main()
```

```
5 {
```

```
6     int i;
```

```
7     double j;
```

```
8     char *bug = NULL;
```

```
9
```

```
10    for (i = 0 ; i < 5 ; i++) {
```

```
11        j = i/2 + i;
```

```
12        printf("j is %lf \n", j );
```

```
13    }
```

```
14
```

```
15    strcpy(bug, "hi");
```

```
16    printf("bug is %s \n", bug);
```

```
17 }
```



Summary

- Development tools
 - ✓ Text editor: **vi**
 - ✓ Compiler: **gcc**
 - ✓ Library archives: **ar**
 - ✓ Maintain related programs: **make**
 - ✓ Debugger: **gdb**

- Homework
 - ✓ 단계별 수행한 실습 과정에 대한 화면을 캡처하여 한글파일에 붙여서 제출
 - ✓ Tools_학번_이름.hwp



E N D

