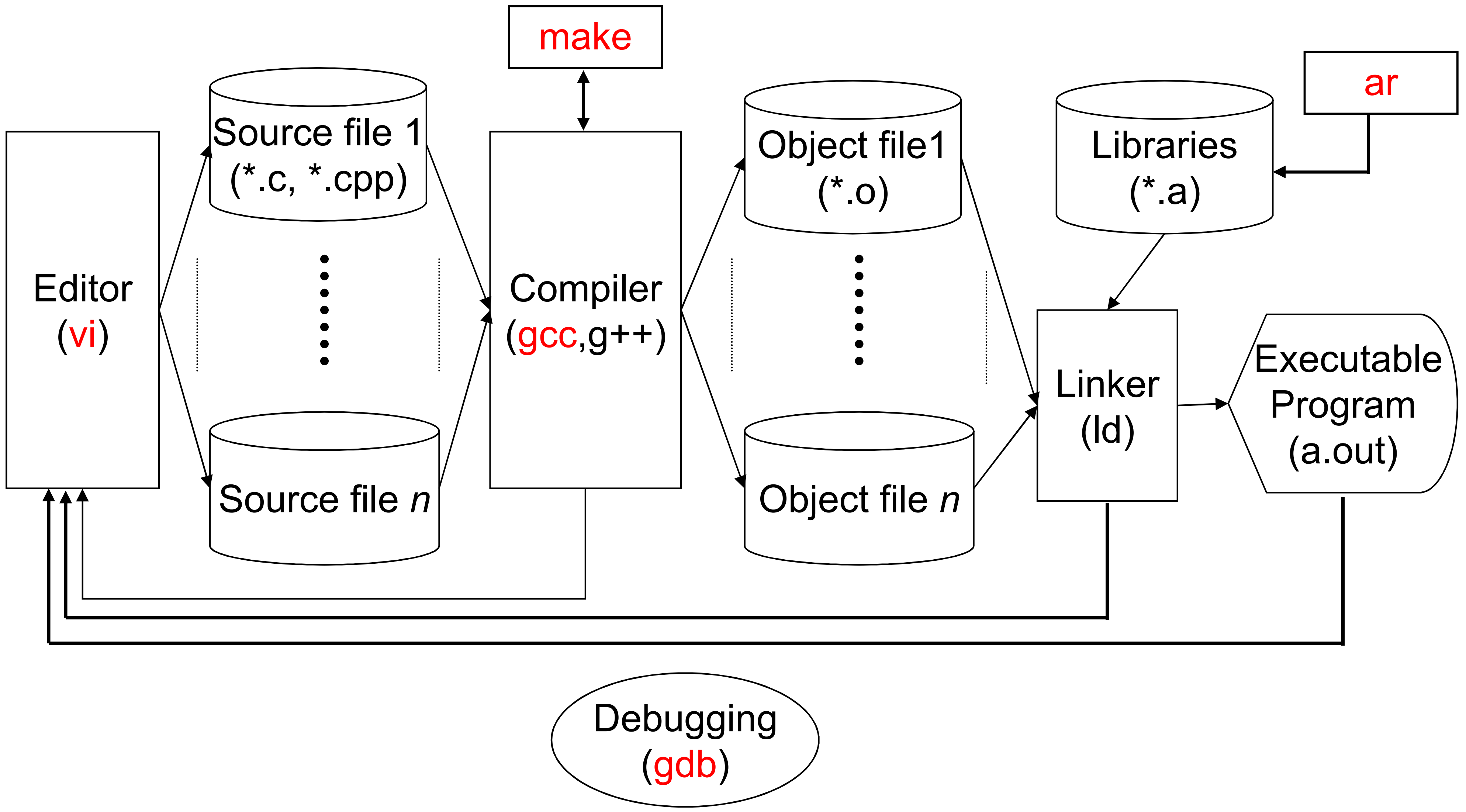
Development Tools in Linux

경희대학교 컴퓨터공학과

조 진 성

Program Development in Linux



vi – Text Editor in Linux

Developed by BSD

Text editor

- ✓ System configuration files, etc.
- ✓ Program source files

Terminal settings

```
$ set TERM = vt100; export TERM (bash)
% setenv TERM vt100 (csh)
```

Other text editors

- ✓ Emacs, nano, gedit

You have to be skillful with vi

- ✓ if you want to be an Linux expert
- ✓ if you want to get ‘A’ in this class

vi (Cont'd)

Execution

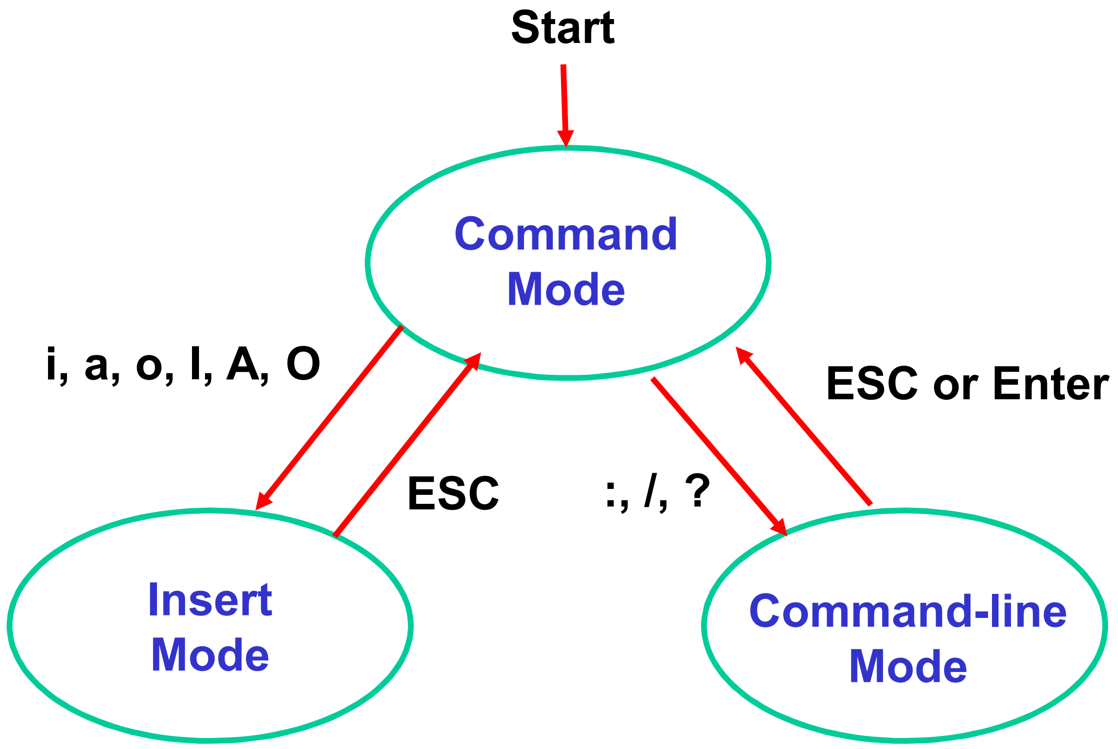
```
$ vi main.c
```

Termination

```
ZZ
:wq
:x
:q!
```

Mode

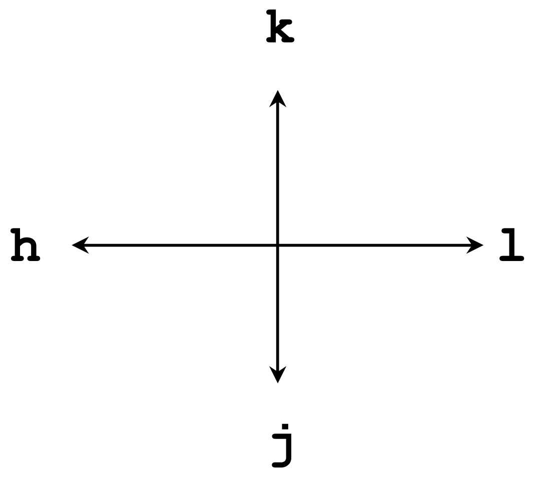
- ✓ Command mode
- ✓ Command line mode
- ✓ Insert mode



vi (Con't)

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 (Cont'd)

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 (Cont’d)

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
✓ `:help [word]`	: help about word

Miscellaneous command in command mode

- ✓ `^L` : refresh
- ✓ `^G` : summary

vi (Cont'd)

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
$ g++ [options] source-files
```

Options

✓	`-c`	: compile only
✓	`-D`*`name`*`[=def]`	: define a symbol *name*
✓	`-g`	: for debug
✓	`-help`	: for help
✓	`-I`*`pathname`*	: add *pathname* for #include files
✓	`-l`*`library`*	: 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 (Cont'd)

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`	: standard directory for library files
✓ `/usr/lib/libc.a`	: standard C library
✓ `/usr/lib/libm.a`	: mathematics library

gcc (Cont'd)

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

```
$ gcc hello.c mystrcpy.c
$ ./a.out
$ gcc -o hello hello.c mystrcpy.c
$ ./hello
```

gcc (Cont'd)

[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 (Cont'd)

Examples

```
$ ar -help
$ ar rcv archive file.o
$ ar rav next.o archive file.o
$ ar xv archive file.o
$ ar tv archive
```

Exercise

```
$ 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] [targets] [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 (Cont'd)

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 (Cont’d)

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 header.h
    gcc -c output.c
clean :
    rm -f core *.o prog
```

make (Cont'd)

Internal macros

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

Special target names

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

make (Cont'd)

Typical Makefile

```
# 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 (Cont'd)

Exercise

```
$ 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

Cf) xxgdb

gdb (Cont'd)

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 (Cont'd)

Exercise

```
$ gcc -g bug.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 in Linux

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