

Introduction to System Programming

Prof. Seokin Hong

Kyungpook National University

Fall 2018

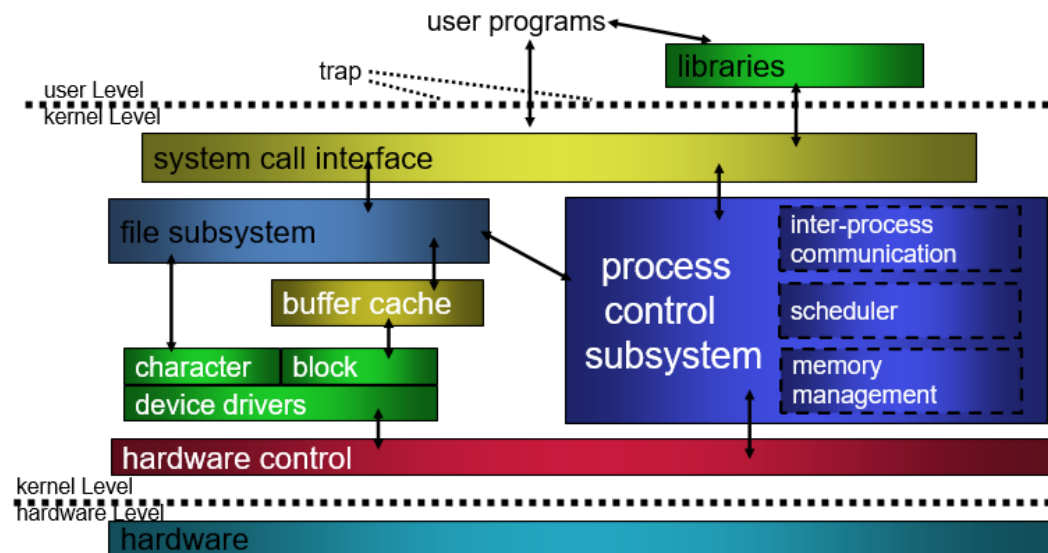
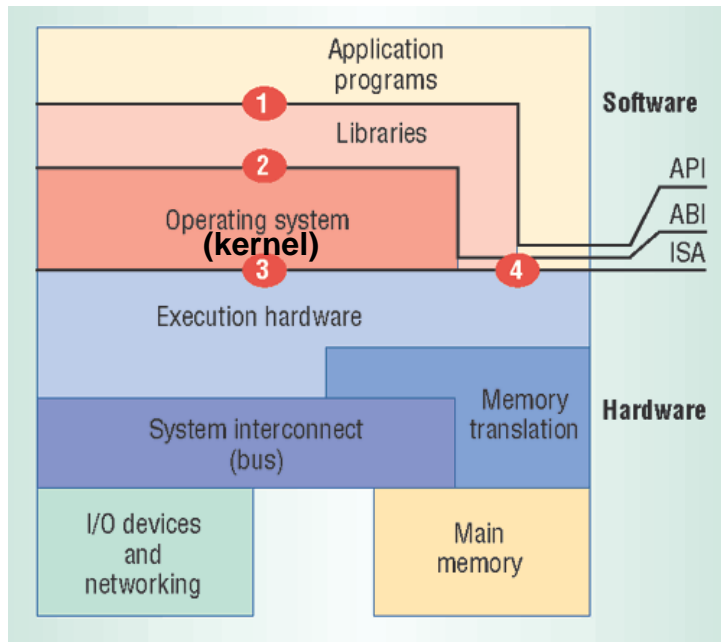
Hope you are here for this,

- To learn a way to write programs at the system-call level in the UNIX/LINUX system.

- **Topics includes**
 - File I/O
 - Device I/O
 - Timers
 - Process management
 - Pipes
 - Network programming
 - Semaphores
 - ...

UNIX architecture and System program

- The kernel provides **services** for accessing to system resources (Processors, I/O, Process management, memory, devices, timers, networking)
- **System programs** use those services directly to access the system resources



Our Goal: Understanding Systems Programming

- To find the answers for
 - What are the details of each type of kernel service?
 - How does data get from a device to the program and back?
 - What is the kernel doing, how it does it
 - And, how to write programs that use those services.

Three steps to learn..

- 1. Looking at “real” programs
 - We will study standard Unix programs to see what they do and how they are used in practice
 - We will see what **system services** these programs use.
- 2. Looking at the system calls
 - We will next learn about the **system calls** we use to invoke **these services**
- 3. Writing our own version of the programs
 - to write our own system programs.

Who are We?

Instructor : Seokin Hong, Ph.D.

- **seokin@knu.ac.kr**
- **Office : 공대9호관 5층, 525**
- **Office Hours : Thu 14:00~15:00**
- <https://sites.google.com/view/seokinhong>
- **PhD from KAIST, worked at IBM, Samsung, ETRI**



▪ Research Interests

- Computer architecture
- Memory and storage systems
- Near-data processing for big data analytics and machine learning
- Architectural supports for security
- Computer system modeling and simulation

Who are We?

Teaching Assistants

■ Hoseung Lee

- astrohsy@gmail.com
- System Programming, : A+



■ Jeong woo Woo

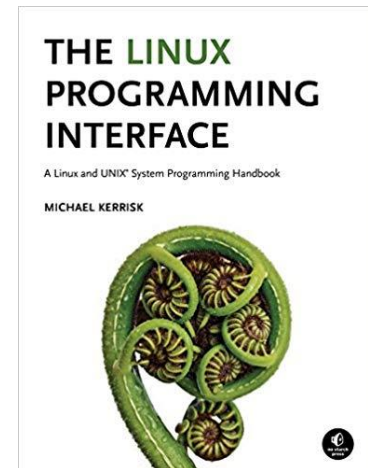
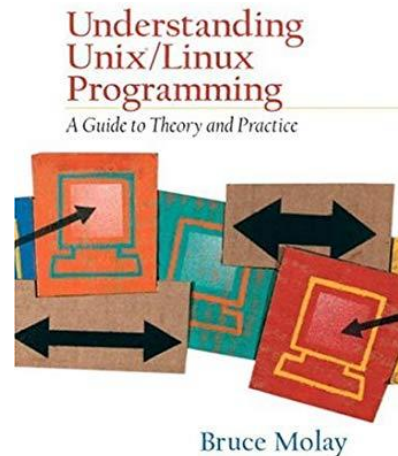
- dnwjddn52@naver.com
- System Programming : A+



Textbook and Course Schedule

■ Textbook

- Primary: Understanding Unix/Linux Programming, Bruce Molay
- Secondary: The Linux Programming Interface, Michael Kerrisk



■ Course Schedule

- Tentative schedule is in syllabus
- But don't believe the “static” schedule

Notice to Students

- Prerequisites

- Students must be able to write programs in C.
- Experience on Linux or other operating systems would be helpful

- Grading for students repeating (retaking) this course or taking “Open Source Programming (2017 Spring)”

- Max. Grade will be limited to A-

Where to Get Course Information?

- Website (LMS: Learning Management System) :
<http://lms.knu.ac.kr/>
 - Announcement
 - Lecture notes
 - Homework
 - Project information
 - Course schedule, handouts, papers, Q&A

How Will You Be Evaluated?

- Homeworks : 25%
 - Team Project : 10%
 - Midterm : 30%
 - Final : 30%
 - Attendance : 5%
-
- Grade will be determined solely by the score (non-negotiable)

Linux for Beginners

- basic shell command,
- vi editor
- gcc compiler
- gdb

Prof. Seokin Hong

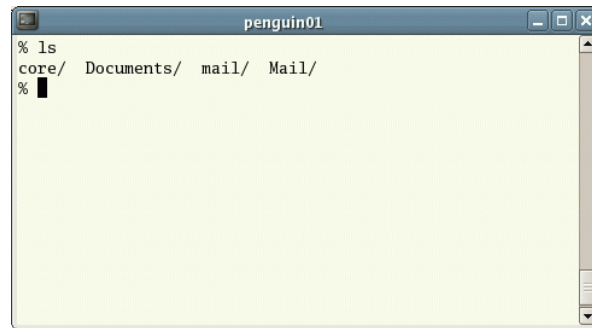
Kyungpook National University

Fall 2018

Listing files and directories

■ ls (list)

- The **ls** command (lowercase L and lowercase S) lists the contents of your current working directory.

A terminal window titled 'penguin01' with a yellow background. It shows the command '% ls' being entered, followed by the output 'core/ Documents/ mail/ Mail/' on the next line. A cursor is visible on the line '% '.

○ Options

- -a : lists files that are normally hidden
- -l : list with long format
- -S : sort by file size
- -t : sort by time & date
- more options : \$ man ls

Making, change Directories

- `mkdir` (make directory)
 - `% mkdir unixstuff`
- `cd` (change directory)
 - `% cd unixstuff`
- The directories `.` and `..`
 - `.` : the current directory
 - `..` : the parent directory
- Pathname
 - `% pwd`

Command	Meaning
<code>ls</code>	list files and directories
<code>ls -a</code>	list all files and directories
<code>mkdir</code>	make a directory
<code>cd <i>directory</i></code>	change to named directory
<code>cd</code>	change to home-directory
<code>cd ~</code>	change to home-directory
<code>cd ..</code>	change to parent directory
<code>pwd</code>	display the path of the current directory

Copying files

- **cp (copy)**

- % cp ~/.bashrc ~/.bashrc.bk

- **mv (move or rename file)**

- % mv ~/.bashrc.bk ~/.bashrc.bk2

- **rm (remove), rmdir (remove directory)**

- % rm ~/.bashrc.bk2

About Vi(m) Editor

- vi (pronounced "vee-eye")
 - A screen-oriented text editor
 - The standard Unix editor
 - There are lots of clones (vim, nvi, elvis, macvim)

Vi Editor

■ Starting the vi Editor

- \$vim testfile

■ Operation Modes

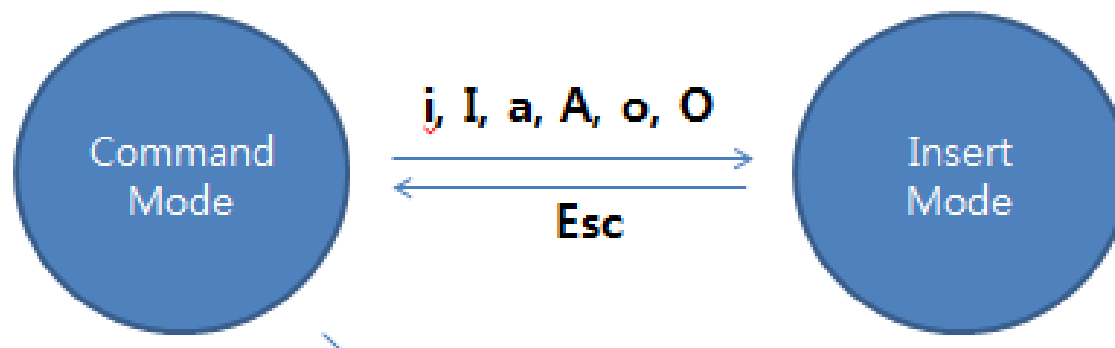
- **Command mode** : perform administrative tasks such as saving the files, executing the commands, moving the cursor, cutting (yanking) and pasting the lines or words, as well as finding and replacing.
- **Insert mode** : This mode enables you to insert text into the file.
- vi always starts in the **command mode**.
- To enter text, you must be in the insert mode for which simply type **i**.
- To come out of the insert mode, press the **Esc** key

■ Getting Out of vi

- The command to quit out of vi is **:q**.
- The command to save the contents of the editor is **:w**
- Save and quit : **wq**

From Command Mode to Insert Mode

Key	Description
i	Insert text before current character
a	Append text after current character
I	Begin text insertion at the beginning of a line
A	Append text at end of a line
o	Open a new line below current line
O	Open a new line above current line



Writing & Exiting in Command/Ex mode

■ Writing

Key	Description
:w	<i>Write current file</i>
:w fileName	<i>Write current file to fileName</i>

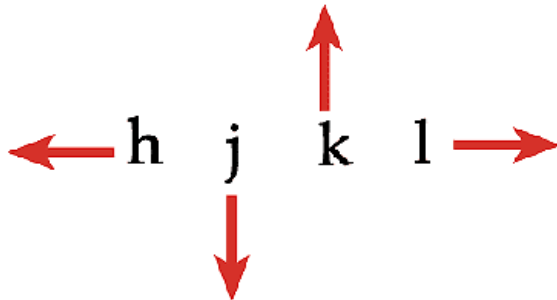
■ Exiting vi

Key	Description
:q	<i>Quit (will only work if file has not been changed)</i>
:q!	<i>Quit without saving changes to file</i>
:wq	<i>Write, then quit</i>
:wq fileName	<i>Write to fileName , then quit</i>

Command mode

■ Basic Cursor Movement

- h,j,k,l
- Arrow key
 - Only support in vim



key	Description
k	<i>Up one line</i>
j	<i>Down one line</i>
l	<i>Right one character (or use <Spacebar>)</i>
h	<i>Left one character</i>
w	<i>Right one word</i>
b	<i>Left one word</i>

Advanced vi

■ Deleting, and Changing Text (in command mode)

Key	Description
x	<i>Delete a character</i>
dd	<i>Delete a line</i>
r	<i>Replace a character</i>
R	<i>Overwrite text, press <Esc> to end</i>

• Others

Key	Description
:set nu	<i>Show line Numbers</i>
:set nonu	<i>Hide line numbers</i>
:!ls	<i>See a list of files in your current directory</i>
:linenum	<i>Go to linenum</i>

Vi Editor

■ Exercise – hello world!

- Create “**hello.c**” that prints “Hello world”

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

Compile C/C++ Program

- `cc filename`
 - C compiler
- `gcc filename`
 - GNU C Compiler from the GCC (GNU Compiler Collection)
- `g++ filename`
 - GNU C++ Compiler from the GCC
- (In general) use `gcc` → call `cc`

Compile C/C++ Program

■ Option

- -c : create object file
- -o : create output(executable) file
- (Without this option, the execute file name is a.out)
- -g : add debugging info

■ Create object file

- `$cc -c hello.c`
- `$ls`

■ Create execute file

- `$cc -o hello hello.c`
- `$./hello`

Practical exercise #1

- Write a code that calculates the sum from 1 to 100.
- The sample output
 - The sum of 1 to 100 is <result>.

The GNU Debugger

- GDB

- GNU debugger

- Compile option for debug

- Caution : You have to use `-g` option when you compile a code
- `$ gcc -g filename -o program`

- Run gdb

- `$ gdb ./program`

gdb commands

■ **Setting breakpoints**

- Set a breakpoint at the beginning of a function
 - (gdb) b main
- Set a breakpoint at a line of the current file during debugging.
 - (gdb) b 10
- Set a breakpoint at the beginning of a class member function.
 - (gdb) b list::erase
- Listing breakpoints.
 - (gdb) info b
- Deleting a breakpoint.
 - (gdb) delete 2

gdb commands

■ **Running the program being debugged**

- Start the program being debugged.
 - (gdb) r
- Execute a single statement. If the statement is a function call, just single step *into* the function.
 - (gdb) s
- Execute a single statement. If the statement is a function call, execute the entire function and return to the statement just after the call; that is, step *over* the function.
 - (gdb) n
- Execute from the current point up to the next breakpoint if there is one, otherwise execute until the program terminates.
 - (gdb) c

gdb commands

■ **Examining Variables**

- Print the value of a variable or expression.
 - (gdb) print count

■ **List source code**

- list lines of source code.
 - (gdb) l
- List lines of source code centered around a particular line.
 - (gdb) l 41

■ **backtrace**

- A backtrace is a summary of how your program got where it is.
 - (gdb) bt

■ **quit gdb**

- (gdb) q

gdb commands

■ Summary of commands

Gdb Command	Description
set listsize <i>n</i>	Set the number of lines listed by the list command to <i>n</i> [set listsize]
b <i>function</i>	Set a breakpoint at the beginning of <i>function</i> [break]
b <i>line number</i>	Set a breakpoint at <i>line number</i> of the current file. [break]
info b	List all breakpoints [info]
delete <i>n</i>	Delete breakpoint number <i>n</i> [delete]
r <i>args</i>	Start the program being debugged, possibly with command line arguments <i>args</i> . [run]
s <i>count</i>	Single step the next <i>count</i> statements (default is 1). Step <i>into</i> functions. [step]
n <i>count</i>	Single step the next <i>count</i> statements (default is 1). Step <i>over</i> functions. [next]
finish	Execute the rest of the current function. Step <i>out of</i> the current function. [finish]
c	Continue execution up to the next breakpoint or until termination if no breakpoints are encountered. [continue]
p <i>expression</i>	print the value of <i>expression</i> [print]
l <i>optional_line</i>	List next <i>listsize</i> lines. If <i>optional_line</i> is given, list the lines centered around <i>optional_line</i> . [list]
where	Display the current line and function and the stack of calls that got you there. [where]
h <i>optional_topic</i>	help or help on <i>optional_topic</i> [help]
q	quit gdb [quit]

gdb Example

test.c

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

```
int main()
```

```
{
```

```
    int a;
```

```
    int b;
```

```
    a = 0;
```

```
    b = 10;
```

```
    int sum = a + b;
```

```
    a = 5;
```

```
    int arr[5];
```

```
    arr[sum] = 1;
```

```
    return 0;
```

```
}
```

```
~$ gcc test.c -o test
```

```
~$ ./test
```

Segmentation fault (core dumped)

```
~$ gcc -g test.c -o test
```

```
~$ gdb test
```

gdb Example (1)

```
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 /home/hjsong/test...done.
(gdb) run
Starting program: ...
```

Program received signal SIGSEGV, Segmentation fault.

0x00007fff00000001 in ?? ()

(gdb) bt

gdb Example (2)

```
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 /home/hjsong/test...done.
```

```
(gdb) break 5
```

Breakpoint 1 at xxxx: file test.c, line 5.

```
(gdb) run
```

```
(gdb) break 7
```

Breakpoint 2 at xxxx: file test.c, line 7.

```
(gdb) c
```

gdb Example (3)

```
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 /home/hjsong/test...done.
```

```
(gdb) break 5
```

Breakpoint 1 at xxxx: file test.c, line 5.

```
(gdb) run
```

```
(gdb) break 7
```

Breakpoint 2 at xxxx: file test.c, line 7.

```
(gdb) s
```

Next Time

- Chapter 2 : Users, Files, and the Manual