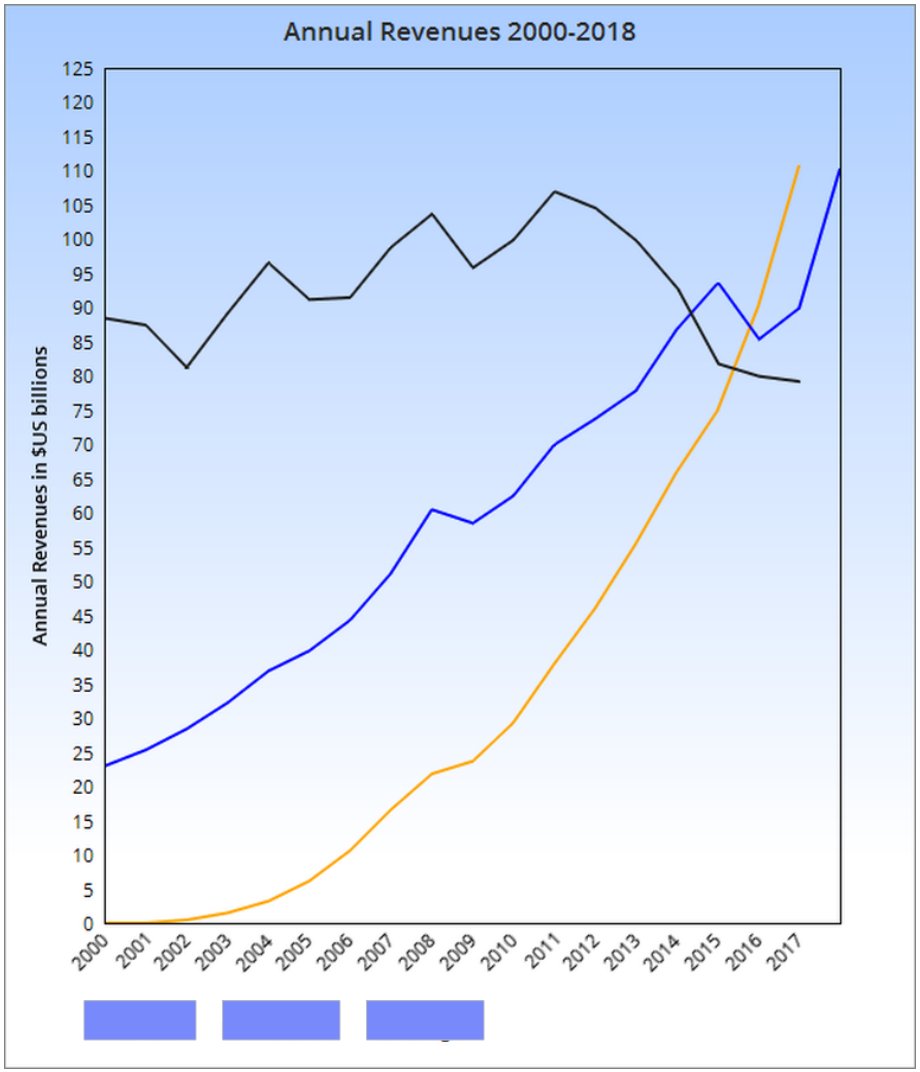
Introduction to System Programming

Prof. Seokin Hong

Kyungpook National University

Fall 2019

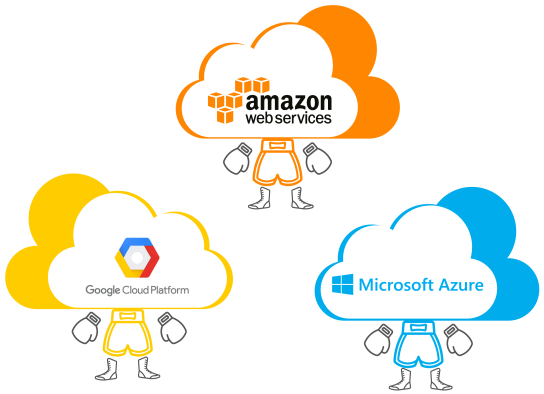
Microsoft is now bigger than IBM has ever been …

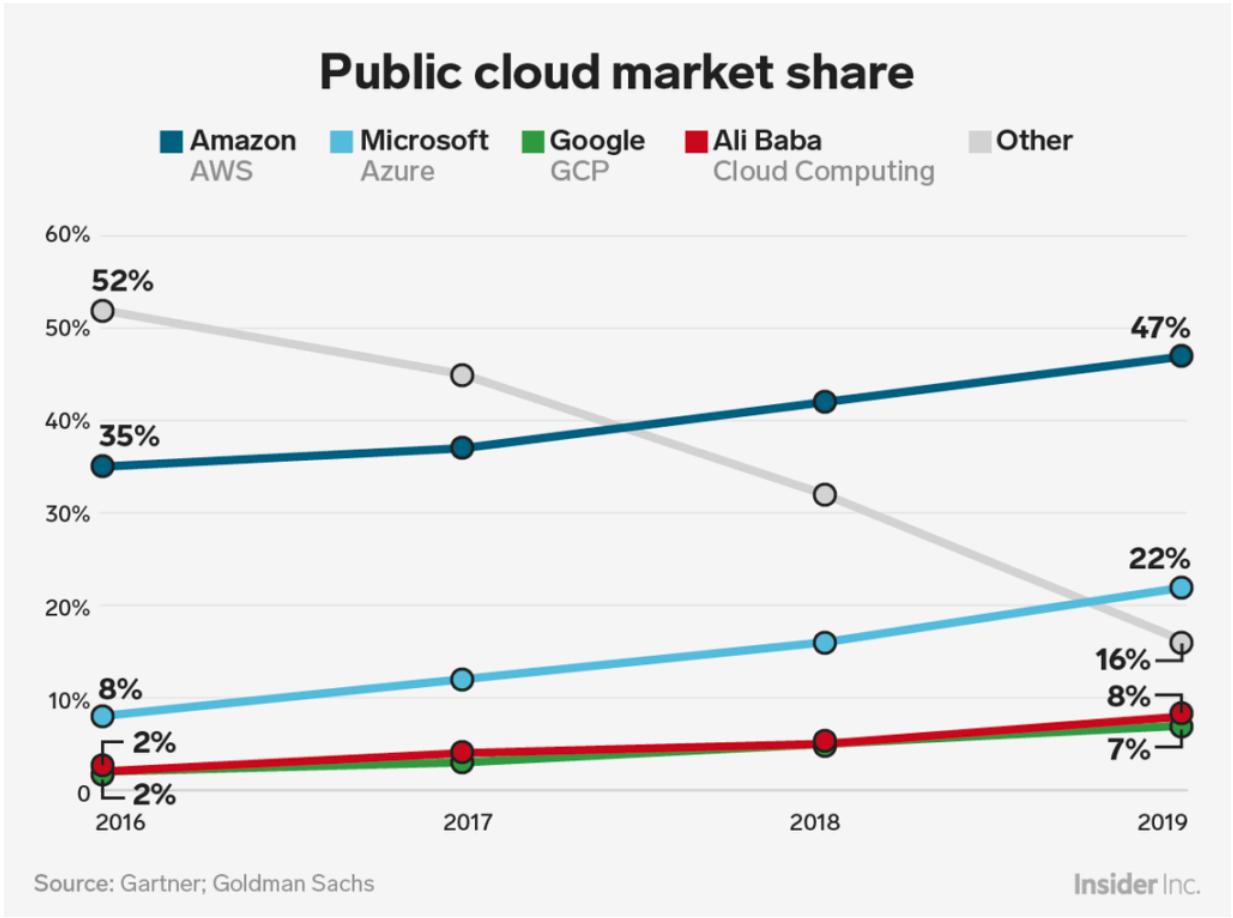


Microsoft Hits Record High as Cloud Revenues Drive Q4 Earnings Beat, Jul 19, 2019

- **Microsoft's Business Model**
  - **More Personal Computing** (2019 revenue of $46 billion, contributed 36% to total revenue)
  - **Intelligent cloud** (2019 revenue of $39 billion, contributed 31% to total revenue)
  - **Productivity and Business Processes** (2019 revenue of $41 billion, contributed 33% to total revenue)

Who is the King in Public Cloud?

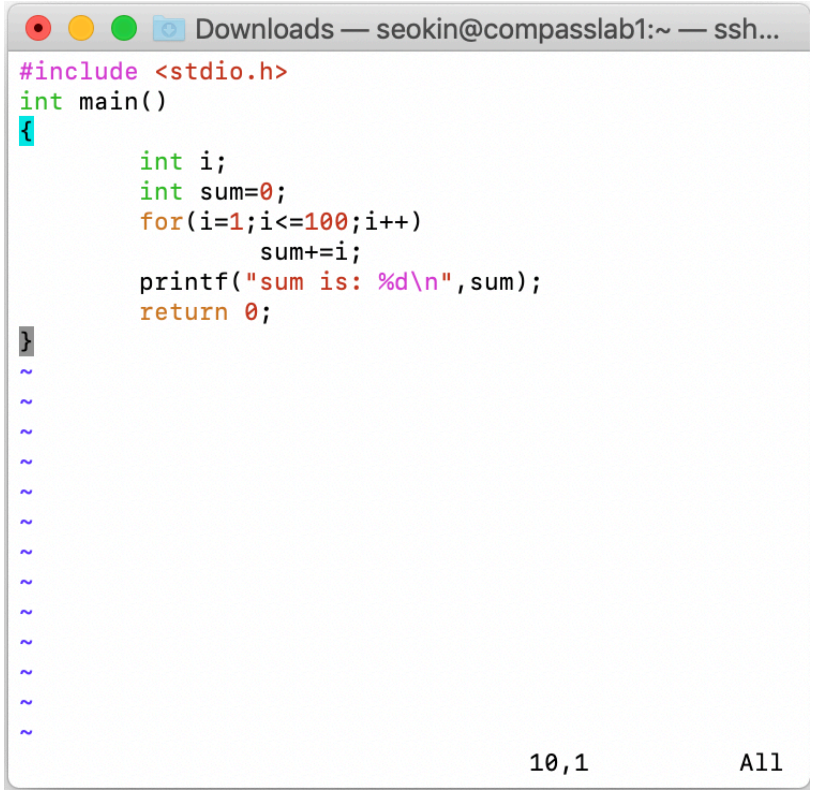




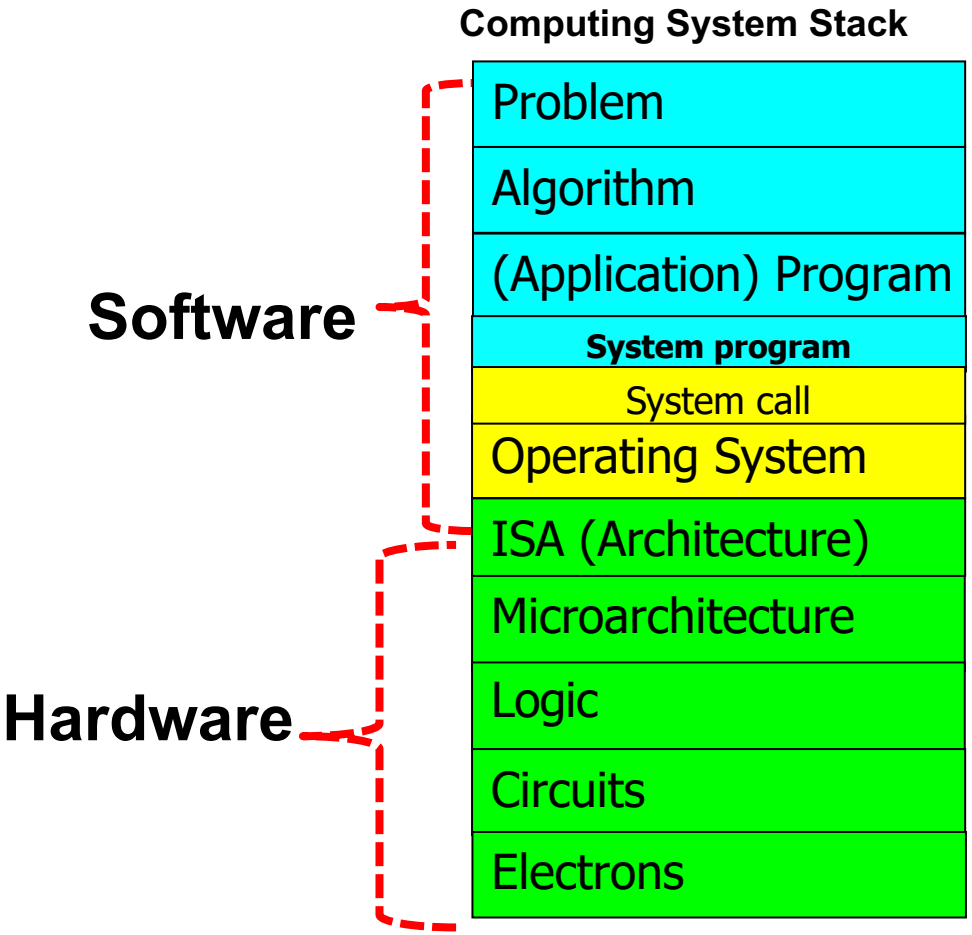
AWS Still King in Public Cloud, While Azure Grows Fastest, IBM Falls

You can be …

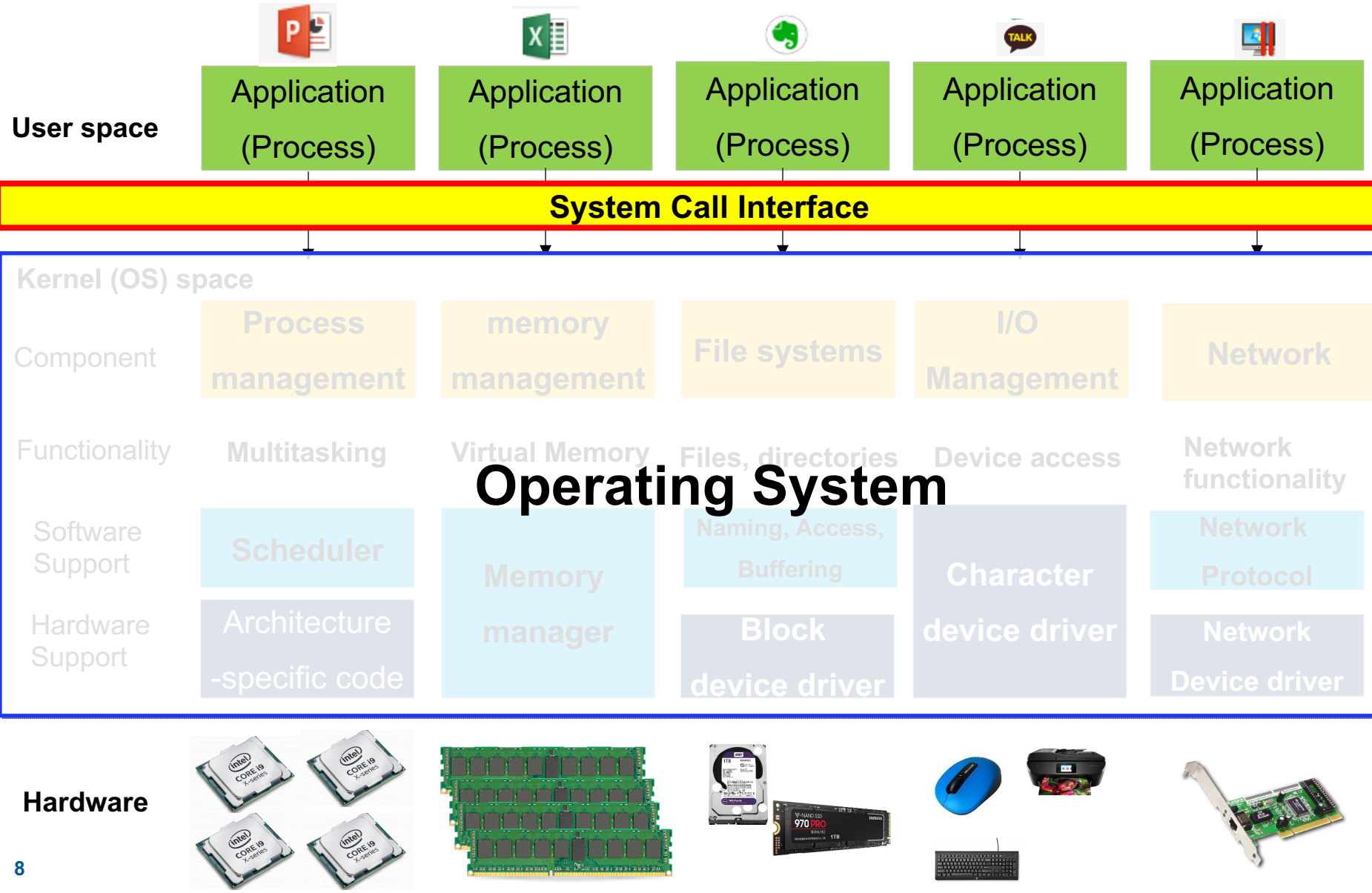
Problem: Write a code that calculates the sum from 1 to 100!



Computing System Stack



How to use Operating System?



How to use Operating System?

Application Program

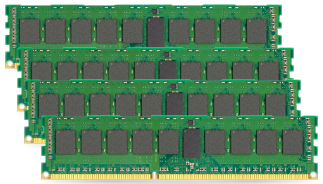
System Programs (low-level library)

System Call Interface (open, read, write, close, etc)

Kernel (OS) space

Component	Process management	memory management	File systems	I/O Management	Network
Functionality	Multitasking	Virtual Memory	Files, directories	Device access	Network functionality
Software Support	Scheduler	Memory manager	Naming, Access, Buffering	Character device driver	Network Protocol
Hardware Support	Architecture -specific code		Block device driver		Network Device driver

Hardware

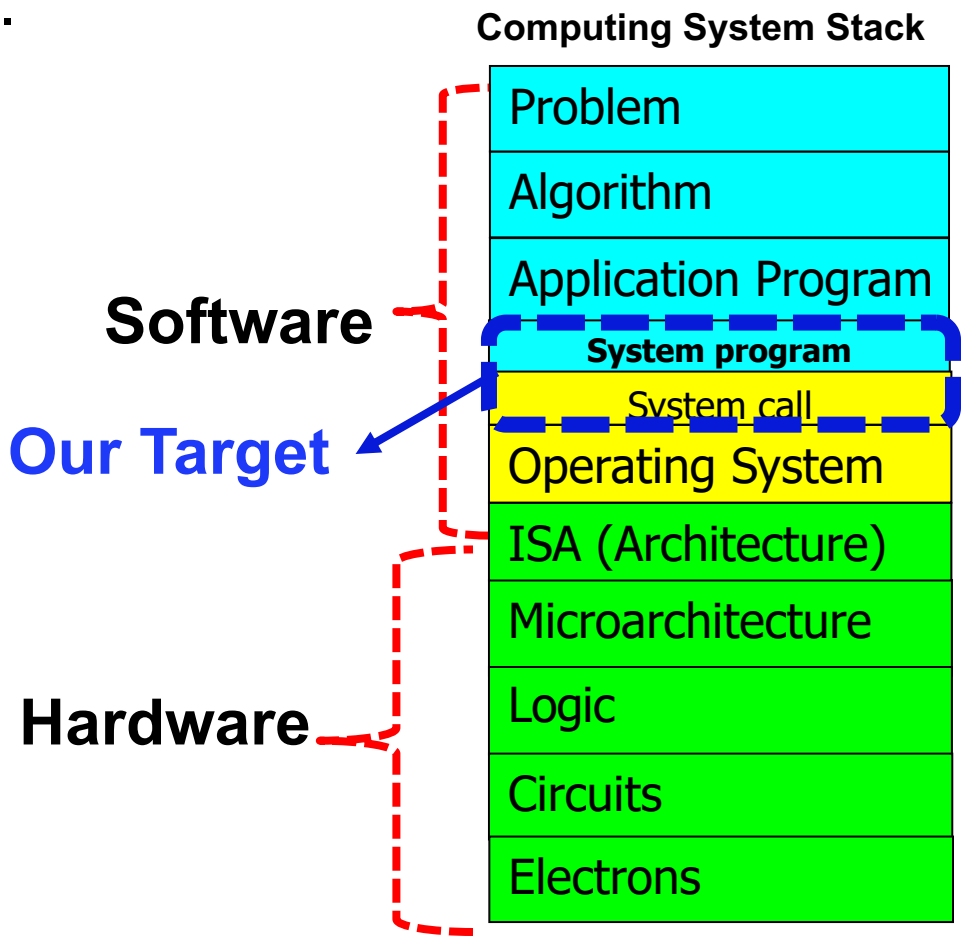




Hope you are here for this,

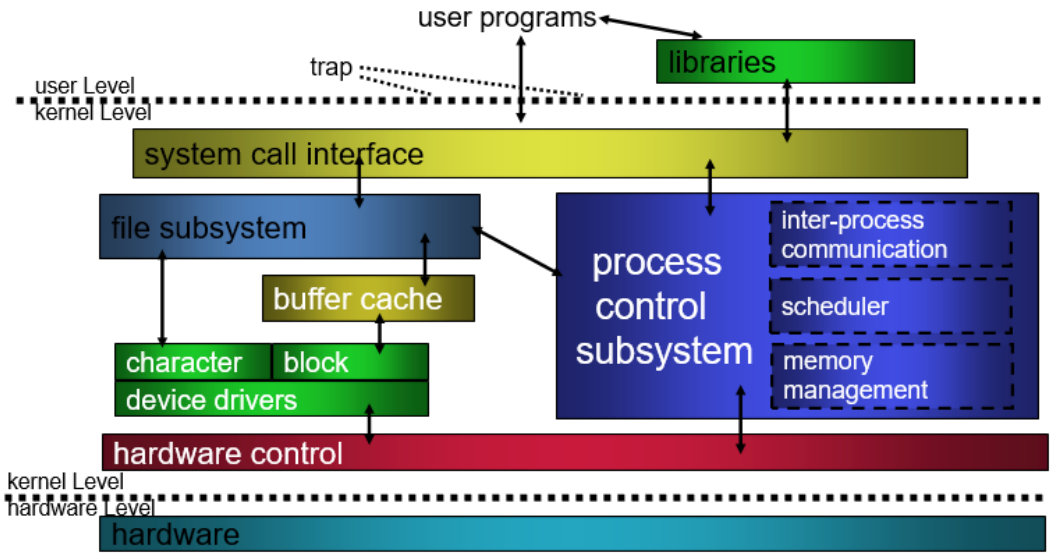
- To learn a way to **write programs at the system-call level** in the UNIX/LINUX system.

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



System program

- The kernel provides **services** for accessing 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 System 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 Linux 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 : Wed** 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?

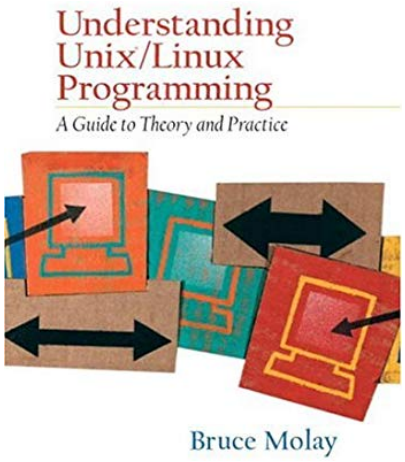
Teaching Assistants

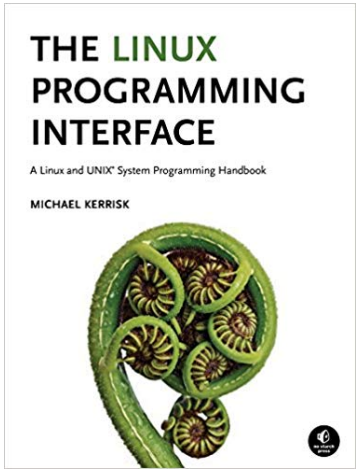
- **남상윤**
  - nsun9505@naver.com
  - System Programming, : A+

- **권오상**
  - kos_0915@daum.net
  - 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
  - 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, Q&A

How Will You Be Evaluated?

- Midterm : 30%
- Final : 30%
- Project : 20%
- Homework : 20%

- Grade will be determined solely by the score (non-negotiable)

Linux for Beginners

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

Prof. Seokin Hong

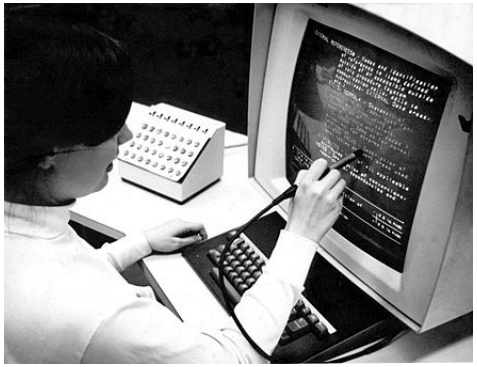
Kyungpook National University

Fall 2019

Terminal

- A electronic device that is used for entering data into, and displaying or printing data from, a computer or a computing system

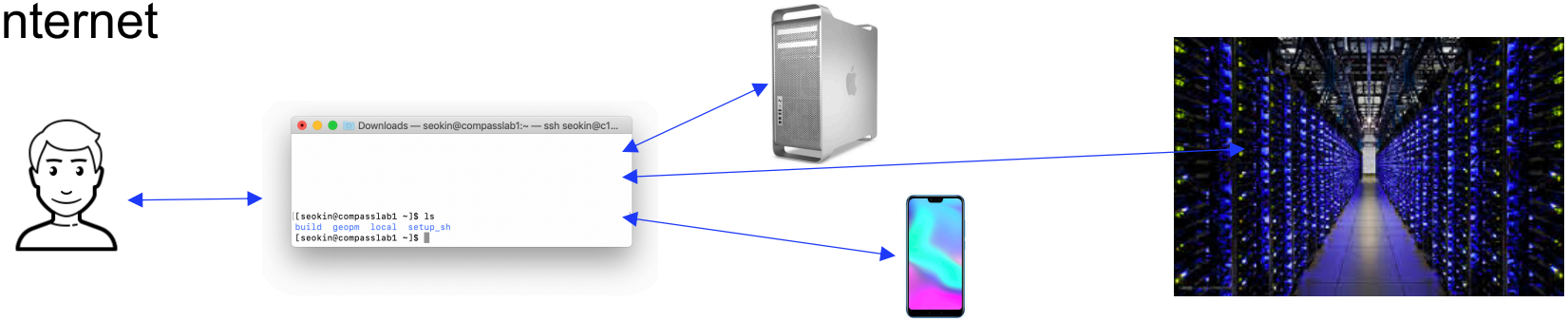
IBM 2741 printing terminal



IBM 2250 Model 4

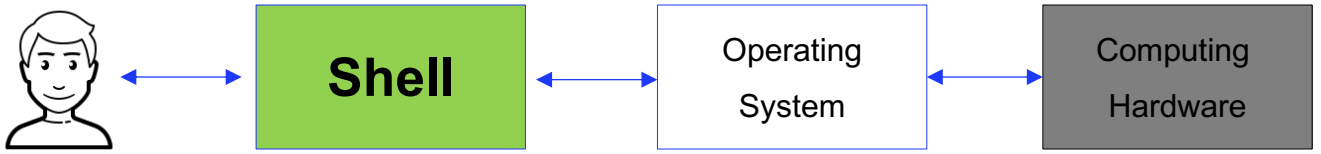
MAC OSX Terminal Emulator

- With terminal, we can access any computing system connected to internet

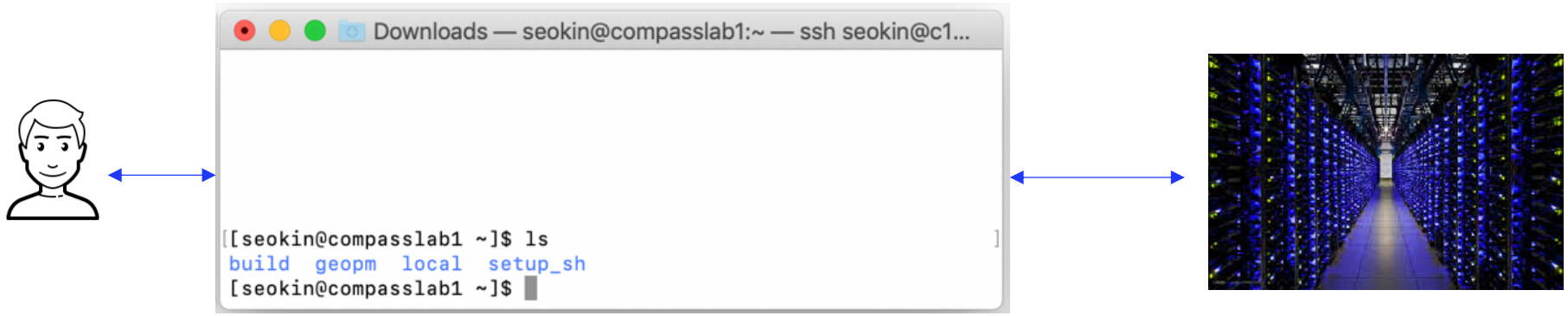


Shell

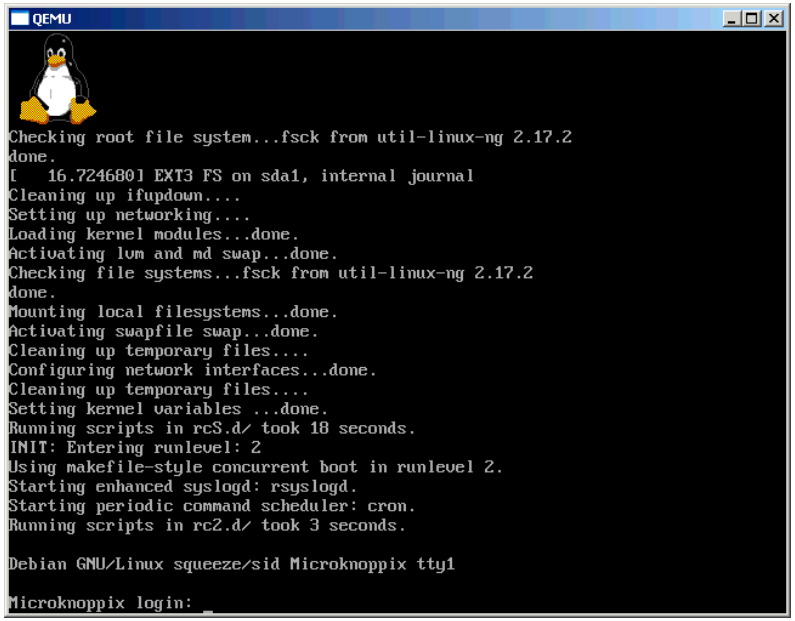
- Command-line interpreter and program launcher for Unix-like operating systems
- Shell is an interface between user and the operating system

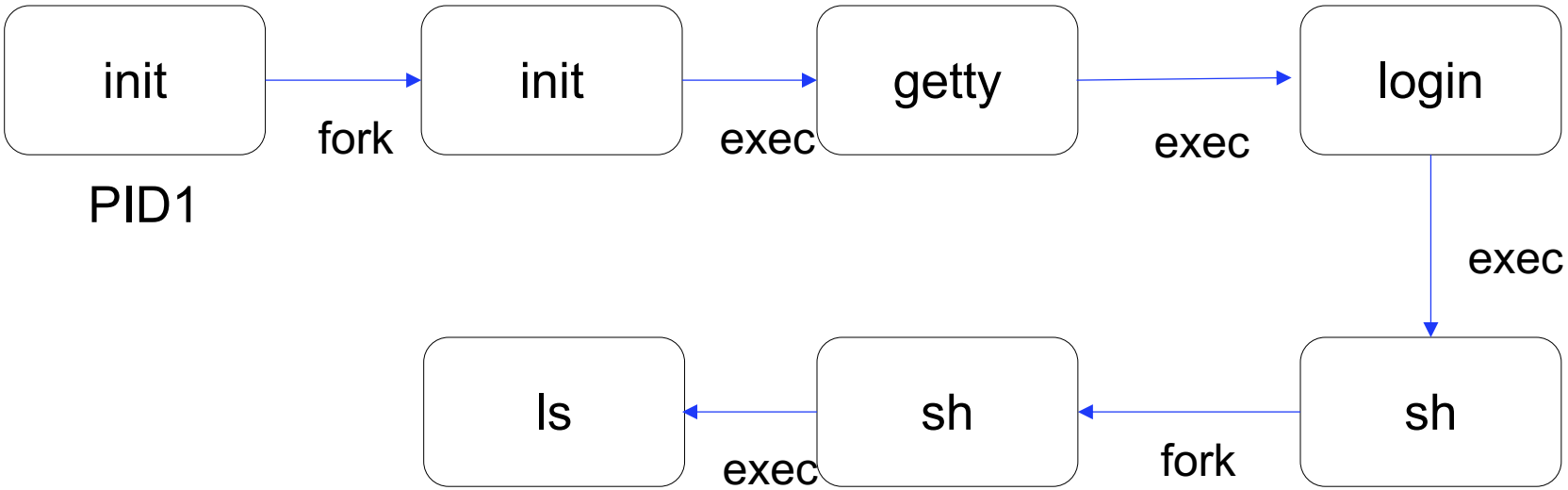


- When a user logs in to a computing system, a login program executes the shell program



Login Sequence in Linux





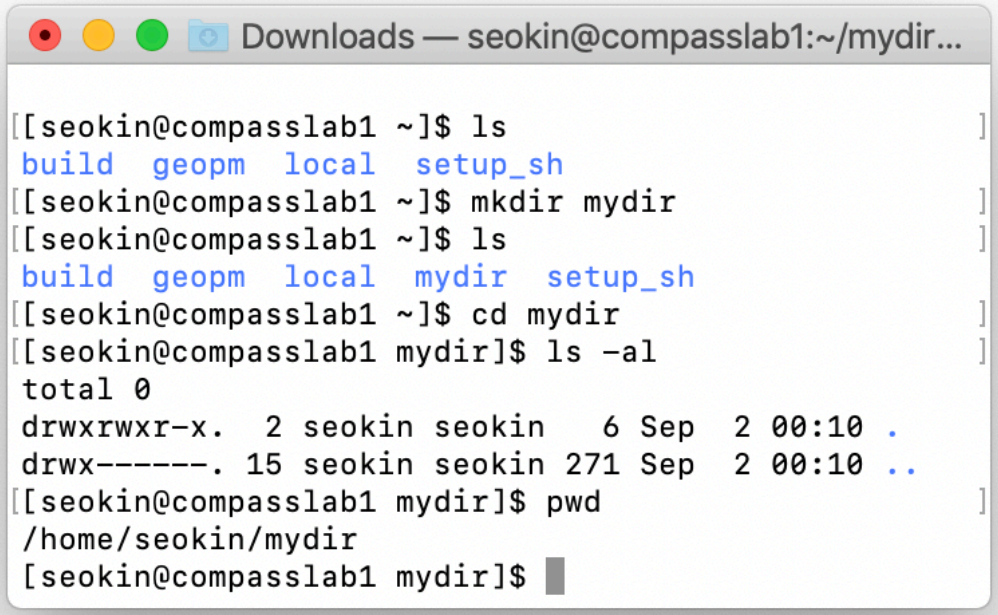
Listing files and directories

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

  - **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, changing Directories

- **mkdir (make directory)**
  - $ mkdir mydir
- **cd (change directory)**
  - $ cd mydir
- **The directories . and ..**
  - . : the current directory
  - .. : the parent directory
- **Pathname**
  - **$ pwd**



Copying, relocating, deleting file

- **cp (copy)**
  - $ cp ~/.bashrc ~/.bashrc.bk
- **mv (move or rename file)**
  - $ mv ~/.bashrc.bk ~/.bashrc.bk2
- **rm (remove), rmdir (remove directory)**
  - $ rm ~/.bashrc.bk2
  - $ mkdir test_dir
  - $ rmdir test_dir

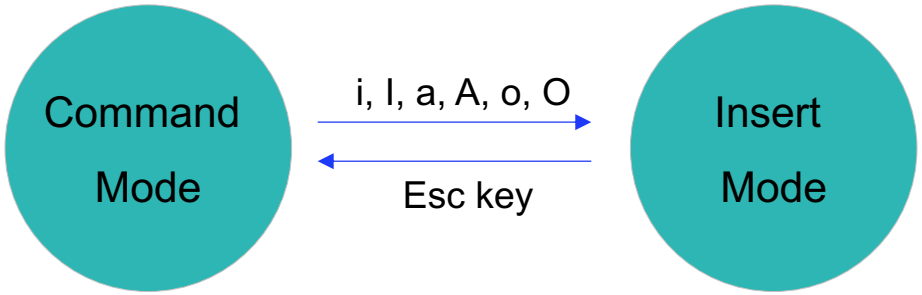
About Vi(m) Editor

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

Vi Editor

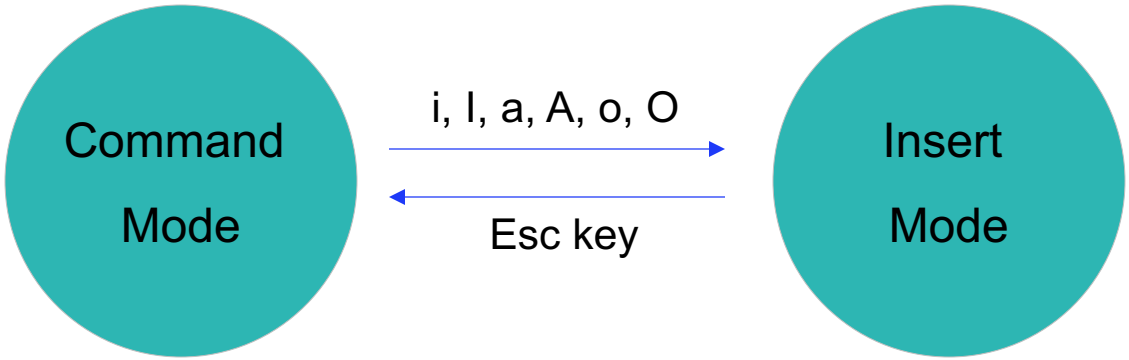
- Installing vim **: $ apt-get install vim**
- **Starting the vi Editor**
  - $vim testfile

- **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



Inserting or Adding Text

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 the end of a line
o	Open a new line below the current line
O	Open a new line above the current line



Deleting Text

Key	Description
x	delete single character under cursor
Nx	*delete N characters, starting with character under cursor*
dw	*delete the single word beginning with character under cursor*
dNw	*delete N words beginning with character under cursor*
D	*delete the remainder of the line, starting with current cursor position*
dd	*delete entire current line*
Ndd	*delete N lines, beginning with the current line; e.g., 5dd deletes 5 lines*

Cutting and Pasting Text

Key	Description
yy	copy the current line into the buffer
Nyy	Copy the next N lines, including the current line, into the buffer
p	put the line in the buffer into the text after the current line

Undo & redo

Key	Description
u	copy the current line into the buffer
Nyy	Copy the next N lines, including the current line, into the buffer
p	put the line in the buffer into the text after the current line

Moving the Cursor

Key	Description
j	*move cursor down one line*
k	*move cursor up one line*
h	*move cursor left one character*
l	*move cursor left one character*
0 (zero)	*move cursor to start of current line*
$	Move cursor to end of current line
w	Move cursor to beginning of next word
b	Move cursor back to beginning of preceding word
:0	Move cursor to first line in file
:n	Move cursor to line n
:$	Move cursor to last line in file

Searching Text

Key	Description
/string	*search forward for occurrence of string in text*
$string	*search backward for occurrence of string in text*
n	*move to next occurrence of search string*
N	*move to next occurrence of search string in opposite direction*

Saving and Reading Files

Key	Description
:r filename<return>	*read file named filename and insert after current line*
:w<return>	*write current contents to file named in original vi call*
:w newfile<return>	*write current contents to a new file named newfile*

Exiting Vi

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

Advanced vi

visit shell

Key	Description
:sh	visit shell
$exit (in the shell)	Return to vi

Splitting window and switching between windows

Key	Description
:sp filename	Horizontally split a window and open filename file
:vsp filename	Vertically split a window and open filename file
Ctrl-ww	Switch between windows

Show line numbers

Key	Description
:set nu	Show line numbers

Advanced vi

Customizing the vim

- Option 1) Install the Ultimate vimrc

```
$ git clone --depth=1 https://github.com/amix/vimrc.git ~/.vim_runtime
$ sh ~/.vim_runtime/install_awesome_vimrc.sh
```

- Option 2) Add .vimrc to ~/

```
set backspace=2
syntax on
filetype indent on
set autoindent
set number
colorscheme desert
```

Vi Editor

- **Practice #1 – hello world!**
  - Create “**hello.c**” that prints “Hello world”

```
#include <stdio.h>

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

Compile C/C++ Program

- gcc filename
  - GNU C Compiler from the GCC (GNU Compiler Collection)
- g++ filename
  - GNU C++ Compiler from the GCC

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
  - $gcc –c hello.c
  - $ls
- Create execute file
  - $cc –o hello hello.c
  - $./hello

Practice #2

- 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 hello.c -o hello
- **Install gdb**
  - $ apt-get install gdb
- **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
  - **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
  - Display the value of a variable or expression
    - (gdb) display 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

Practice #3 : gdb

gdb_test.c

```
#include <stdio.h>

 int main()
{
    int i;
    int sum=0;

    for(i=1;i<=100;i++)
       sum+=i;
    printf(”sum is: %d\n”,sum);

    return 0;
 }
```

~$ gcc –g gdb_test.c –o gdb_test

~$ gdb gdb_test

Next Time

- Chapter 2 : Users, Files, and the Manual