



PROGRAMMING & DATA STRUCTURE

Instructor: Sourav Kumar Dandapat

TEXT/REFERENCE BOOKS & NOTES

○ C Programming

- Programming with C (Second Edition) B.S. Gottfried, Schaum's Outline Series, Tata McGraw-Hill, 2006.
- Kernighan & Ritchie, *The C Programming Language*, 2nd Ed.

○ Data structures

- Data structures using C and C++ (Second Edition) Y. Langsam, M.J. Augenstein, A.M. Tanenbaum,
- S. Lipschutz, Schaum's Outline Series, Tata McGraw-Hill, 2006.

○ http://172.16.1.252/~sourav/CS_2020



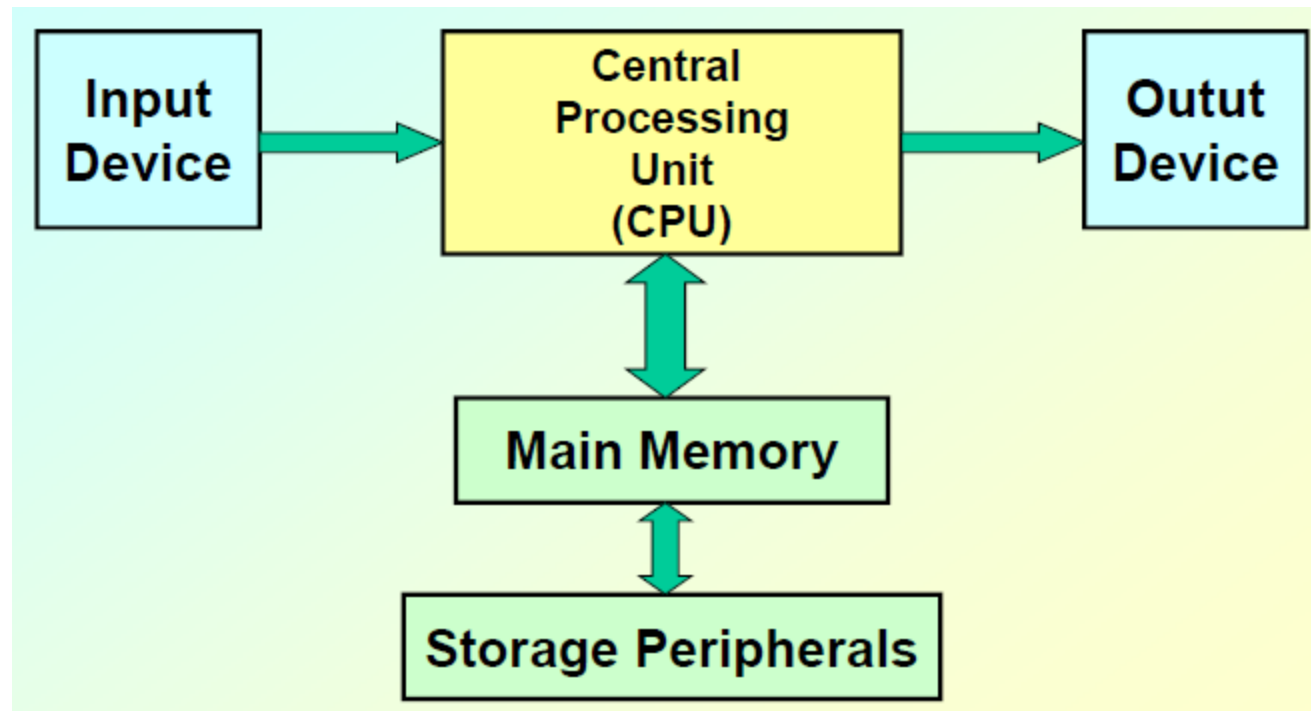
ABOUT THE COURSE AND EVALUATION

- Evaluation in the theory course:
 - Mid-semester 30%
 - End-semester 40%
 - Six quizzes 30%
 - Tentative quiz dates: 15th January, 29th January, 12th February, 18th March, 1st April, 15th April
- Minimum attendance criteria 75%



WHAT IS A COMPUTER?

- It is a machine which can accept data, process them, and output results.



CENTRAL PROCESSING UNIT (CPU)

- ❖ All computations take place here in order for the computer to perform a designated task.
- ❖ It has a number of registers which temporarily store data and programs (instructions).
- ❖ It has circuitry to carry out arithmetic and logic operations, take decisions, etc.
- ❖ It retrieves instructions from the memory (fetch), interprets (decode) them, and performs the requested operation (execute).



MAIN MEMORY


- ❖ It is typically known as Random Access Memory or (RAM)
- ❖ It is much faster compared to storage (hard disk) however much more costlier compared to storage
- ❖ Memory sizes in the range of 1 Gbytes to 16 Gbytes are typical today.
- ❖ Some measures to be remembered
 - 1 K (kilo) = 2^{10} (= 1024)
 - 1 M (mega) = 2^{20} (= one million approx.)
 - 1 G (giga) = 2^{30} (= one billion approx.)



- ❖ **Input Device**
 - **Keyboard, Mouse, Scanner, Touchpad**
- ❖ **Output Device**
 - **Monitor, Printer**
- ❖ **Storage Peripherals**
 - **Magnetic Disks: hard disk,**
Allows direct access
 - **Optical Disks: CDROM, CD-RW, DVD, BlueRay**
Allows direct access
 - **Flash Memory: pen drives**
Allows direct access
 - **Magnetic Tape: DAT**
Only sequential access



HOW DOES A COMPUTER WORK?

- ❖ **Stored program concept.**
 - Main difference from a calculator.
 - ❖ **What is a program?**
 - Set of instructions for carrying out a specific task.
 - ❖ **Where are programs stored?**
 - In secondary memory when first created
 - Brought into main memory, during execution.
- 

CONCEPT OF BITS AND BYTES

❖ Bit

- A single binary digit (0 or 1).

❖ Nibble

- A collection of four bits (say, 0110).

❖ Byte

- A collection of eight bits (say, 01000111).

❖ Word

- Depends on the computer.
- Typically 4 or 8 bytes (that is, 32 or 64 bits).



CONTD..

- ❖ **An k-digit decimal number**
 - Can express unsigned integers in the range 0 to $10^k - 1$.
- ❖ **An k-bit binary number**
 - Can express unsigned integers in the range 0 to $2^k - 1$.
 - For k=8, from 0 to 255



CLASSIFICATION OF SOFTWARE

❖ Application Software

- Used to solve a particular problem.
- Editor, financial accounting, weather forecasting, mathematical toolbox, etc.

❖ System Software

- Helps in running other programs.
- Compiler, operating system, etc.



ALGORITHM

Precise step-by-step instructions to solve a problem.

Characteristics

- ❑ Accept Inputs.
- ❑ Instructions are precise and unambiguous.
- ❑ Takes finite time.
- ❑ Produces one or more output.



COMPUTER LANGUAGES

Machine Language

- ❖ Expressed in binary.
- ❖ 10110100 may mean ADD, 01100101 may mean SUB, etc.
- ❖ Directly understood by the computer.
- ❖ Not portable; varies from one machine type to another.
- ❖ Program written for one type of machine will not run on another type of machine.
- ❖ Difficult to use in writing programs.



Assembly Language

- ❖ Mnemonic form of machine language.
- ❖ Easier to use as compared to machine language.
- ❖ For example, use “ADD” instead of “10110100”.
- ❖ Not portable (like machine language).
- ❖ Requires a translator program called *assembler*.
- ❖ Every assembler has its own assembly language which is designed for one specific computer architecture.



❖ **Assembly language is also difficult to use in writing programs.**

– **Requires many instructions to solve a problem.**

– **Example: Find the average of three numbers.**

MOV A,X ; A = X

ADD A,Y ; A = A + Y

ADD A,Z ; A = A + Z

DIV A,3 ; A = A / 3

MOV RES,A ; RES = A

❖ **In C, $RES = (X + Y + Z) / 3$**

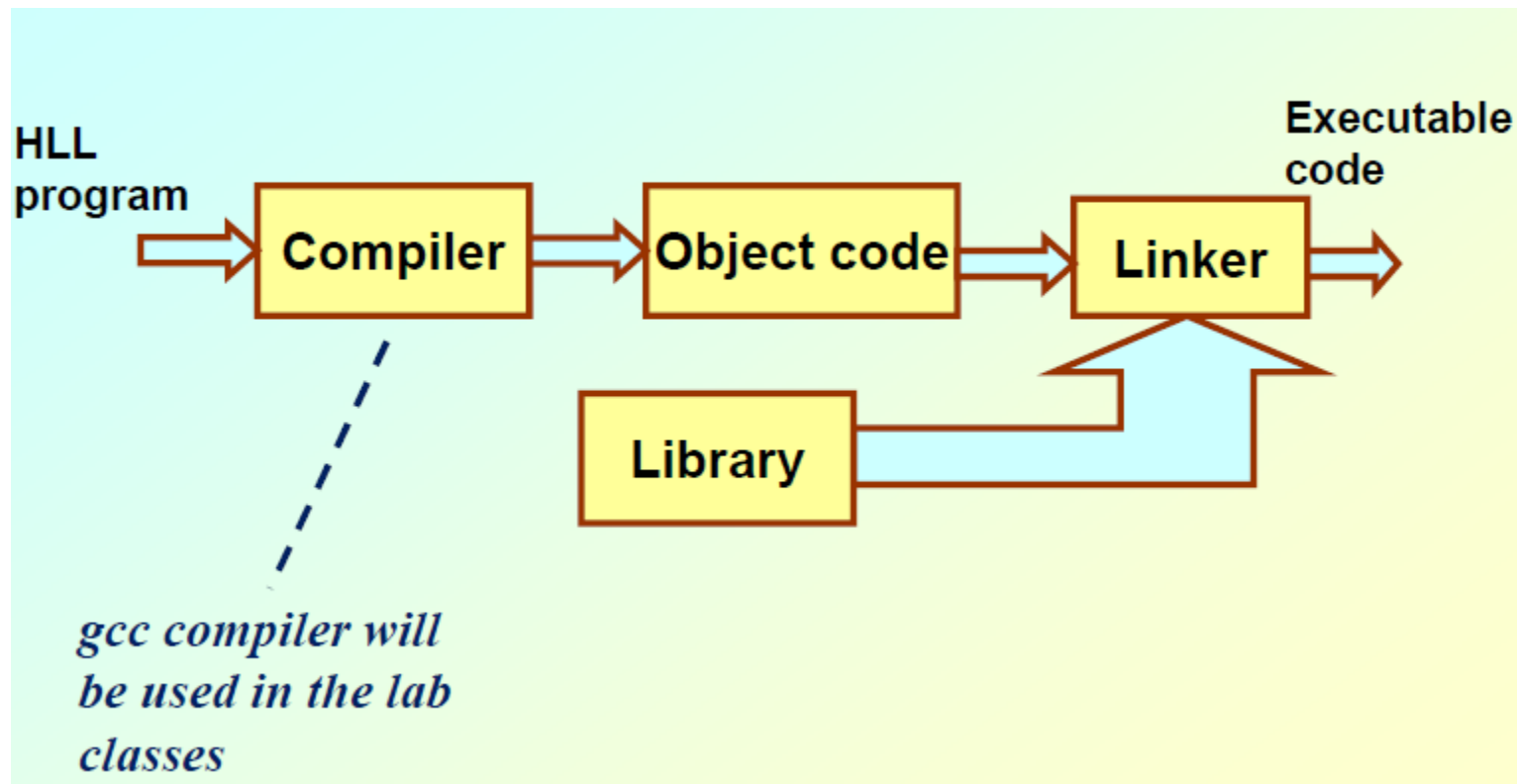


HIGH-LEVEL LANGUAGE

- ❖ Machine language and assembly language are called low-level languages.
- ❖ They are closer to the machine and difficult to use.
- ❖ High-level languages are easier to use and they are closer to the programmer.
- ❖ Examples: Fortran, C, C++, Java.
- ❖ Requires an elaborate process of translation.
- ❖ Using a software called *compiler*.
- ❖ They are portable across platforms.



COMPILATION



IMPERATIVE VS. DECLARATIVE PROGRAMMING

- ❑ Declarative : Describes computation in terms of what you want to do but not how you want to do. Eg: SQL
- ❑ Declarative Programming is like asking your friend to draw a landscape. You don't care how they draw it, that's up to them.
- ❑ Imperative – Describes computation in terms of program state and statements that change program state. Eg. C, etc
- ❑ Imperative Programming is like your friend listening to Bob Ross tell them how to paint a landscape. While good ole Bob Ross isn't exactly commanding, he is giving them step by step directions to get the desired result.



HISTORY OF C

Programming Language	Year	Remarks
Algol	1958	Lack of standard input/output interface, used mainly by scientific community, vendors were not interested
CPL (Combined Programming Language)	1960	CPL was intended for a wider application area than scientific calculations and was therefore much more complex and not as elegant as ALGOL, it included industrial process control, business data processing and possibly some early command line games.
BCPL	1967	BCPL introduced several features of modern programming languages, including using curly braces to delimit code blocks; and the world's first 'hello world' demonstrator program. Conservative and hence can deal with specific set of problems.
B	1969	B was derived from BCPL, and its name may be a contraction of BCPL. Basis of C . Conservative and hence can deal with specific set of problems.
C	1972	Widely used programming language

LINUX BASICS

- Logging in /Authentication.

Two modes – text (quicker) and graphical (requires more system resources).

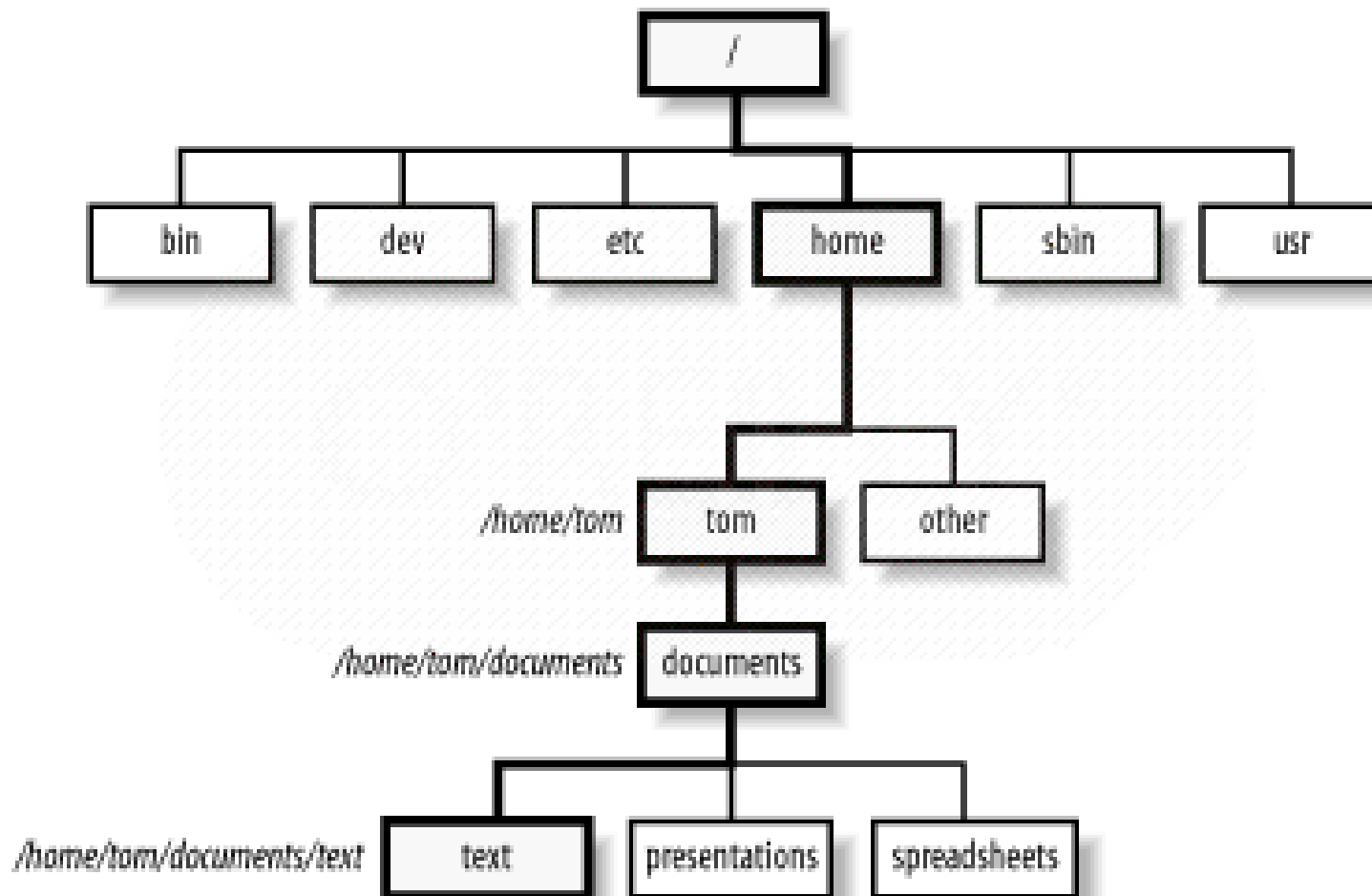
- Remotely login to the server

- If you are trying to connect from Windows system use ssh client (putty, SSH secure shell client)
- From linux system use the following command
ssh user_name@remote_terminal_ip
- Xmanager is used to remotely login using graphical mode.



LINUX DIRECTORY STRUCTURE

Linux directory structure



○ / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.

○ /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.



- /dev – Device Files
 - Contains device files.
 - These include terminal devices, usb, or any device attached to the system.
 - For example: /dev/tty1
- /etc – Configuration Files
 - Contains configuration files required by all programs.
 - This also contains startup and shutdown shell scripts used to start/stop individual programs.
- /home – Home Directories
 - Home directories for all users to store their personal files.
 - For example: /home/john, /home/nikita



○ /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.

○ /usr – User data

- Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications



USEFUL COMMANDS

- ❑ `passwd` – Change password
- ❑ `exit` or `logout` – Leave the session
- ❑ `ls` – List files and directories
- ❑ `mkdir <name>` - Make Directory
- ❑ `cd <name>` - Change Directory
- ❑ `rm -rf <name>` - Remove Directory recursively
- ❑ `rm <name>` - remove file
- ❑ `touch <name>` - create an empty file
- ❑ `cp file1 file2` copy content of file1 to file2, if file2 is not there then it is created
- ❑ `mv file1 file2` - rename file1 to file2
- ❑ `pwd` – shows present working directory
- ❑ `cat <name>` - Display contents of a file



- Lets say you open a terminal in linux and typed “pwd” and then press enter it shows /home/damodar
- Then you typed “ls” and press enter, it shows nothing
- then you typed mkdir CS102 and press enter
- You typed then “ls” and press enter. What will you see?
- Then you typed cd CS102 and press enter. What would happen?
- Then you typed “pwd” and press enter. What will you see?



- Then you typed “touch a” press enter. What would happen?
- Then you typed “ls” and press enter what would you see?
- Then you typed “cp a b” and press enter what would happen?
- Then you typed “ls” and press enter what will you see?
- Then you typed “mv a c” and press enter what would happen?
- Then you typed “ls” and press enter what will you see?



- Then you typed “rm b” what would happen?
- Then you typed “ls” and press enter what will you see?
- Then you typed “cd ..” what would happen?
- Then you typed “ls” and press enter what will you see?
- Then you typed “pwd” and press enter what will you see?
- Then you typed “rm -rf CS102” what would happen



USEFUL COMMANDS

Getting Help

- ❑ `man <command>` - Show manual
- ❑ `info <command>; whatis <command>`



FILE EDITING

- vi <name> - Text editor.
- ❑ Two modes – command and insert.
- ❑ Enter “i” for insert mode and type text.
- ❑ “Esc” for command mode
- ❑ x : delete letter
- ❑ dw: delete word
- ❑ <n> yy : Yank/Copy n lines to buffer
- ❑ p: Paste the lines




FILE EDITING

- w: write/save file
- x: exit editor
- q: quit editor
- wq: write and quit
- w!: Forceful write
- q!: Quit without saving changes



FILE PERMISSION

- ❑ **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
 - ❑ **Write:** The right permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
 - ❑ **Execute:** In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.
- 

OWNERSHIP OF LINUX FILES

- **User:** A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.
- **Group:** A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file.
- **Other:** Any other user who has access to a file.



FILE PERMISSION

○ Checking file permission

- `ls -l`
- Example `ls -l hello.c`
- `-rw-r--r-- 1 sourav fac 68 Jan 3 18:09 hello.c`

○ Setting file permission

- `chmod 755 hello.c`
- `ls -l hello.c`
- `-rwxr-xr-x 1 sourav fac 68 Jan 3 18:09 hello.c`

