Introduction to Computing & Programming (CSD 101)

Practice Lab – 1

For this practice lab, you will learn basic Linux commands. These commands are basic steps to edit and compile a C program in Ubuntu Linux distribution.

Instructions
There are commands in this assignment.

Due Date:

Submitting this Assignment
No submission required.

Grading Criteria
This is practice assignment with 0 points; however, you are required to complete this during the lab hours.

Basic Tips:

There are some things you need to know before heading into the deep waters of the Command Line:

1. Linux commands are cAse-sensitive (dedoimedo and Dedoimedo are two different files).

2. It is best to create folders and files in Linux WITHOUT spaces. For example: Red Gemini.doc is a valid Windows filename, but you might have problems accessing it from the command line in Linux; you should rename the file to RedGemini.doc.

3. Pressing TAB when typing a command will auto-complete the command. For example: if you have a single file in a certain folder that begins with the letter p, typing p then TAB will automatically complete the name regardless of its length; if you have more than one file, the command will complete the maximum available part of the string that matches all relevant filenames (s + TAB for smirk and smile will auto-complete only to smi; then you need to type r or l).

Practice Lab 1 – ICP (CSD-101)

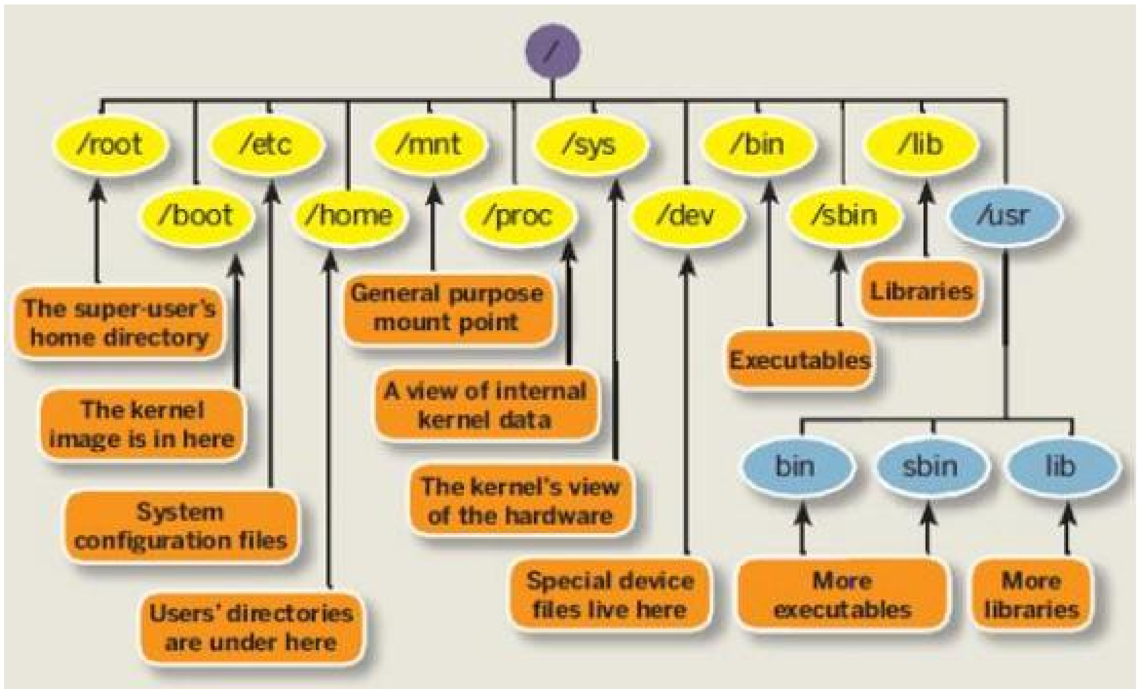
/
/root
/etc
/mnt
/sys
/bin
/lib
/boot
/home
/proc
/dev
/sbin
/usr
The super-user's home directory
The kernel image is in here
System configuration files
Users' directories are under here
General purpose mount point
A view of internal kernel data
The kernel's view of the hardware
Special device files live here
Executables
Libraries
bin
sbin
lib
More executables
More libraries

Fig. 1 GNU Linux File System

Do the following:

1. Boot your Lab computer in Ubuntu
2. Login using credentials provided by the TA in the lab
3. Open a terminal by pressing `Ctrl+Alt+t`
4. Study the Linux file system shown in the figure [1]
    - You should note that topmost directory in the filesystem is **/** (pronounced as root)
    - All the subdirectories are below root in the inverted tree of file-system
    - Absolute path name is complete path name of the file starting from / e.g. /root/jack/myfile.c is absolute path name
    - Relative path name is always with respect to current working directory (obtained using `pwd`)
5. Print working directory: `pwd`
    - Open a terminal
    - Type `pwd` and then enter
    - It displays the absolute path name of the current working directory
6. Create a directory called myCprograms: mkdir
    - Type `mkdir myCprograms` and then enter
7. Listing the contents of the directory
    - Type `ls -al` and then enter

8. Change directory : `cd`
   - `cd myCprograms` and then enter
   - Type `pwd and ls` to check as done earlier
   - To move from current folder to parent folder type `cd ..`
9. Create source code file prog1.c
   - Type `gedit prog1.c` and then enter
   - Type the following C program given below and save it

```
#include<stdio.h>
int main()
{
int a, b, sum;
printf("Enter the value of a and b");
scanf("%d%d", &a,&b);
sum=a+b;
printf("The result is %d",sum);
return 0;
}
```

10. GNU Compiler Collection (gcc) options and flags
    - `gcc prog1.c` If the program is compiled without errors, you can execute the program by typing `./a.out`
    - `gcc -o test test1.c` If the program is compiled without errors, you can execute the program by typing `./test`
    - gcc goes through a sequence of different intermediate steps before generating final executable. Those intermediate steps are the result of different tools which are invoked internally to complete the compilation of the source code.
    - The whole Compilation process is broken down into following phases:
      - Preprocessing
      - Compilation
      - Assembly
      - Linking
    - Please refer to figure [2]

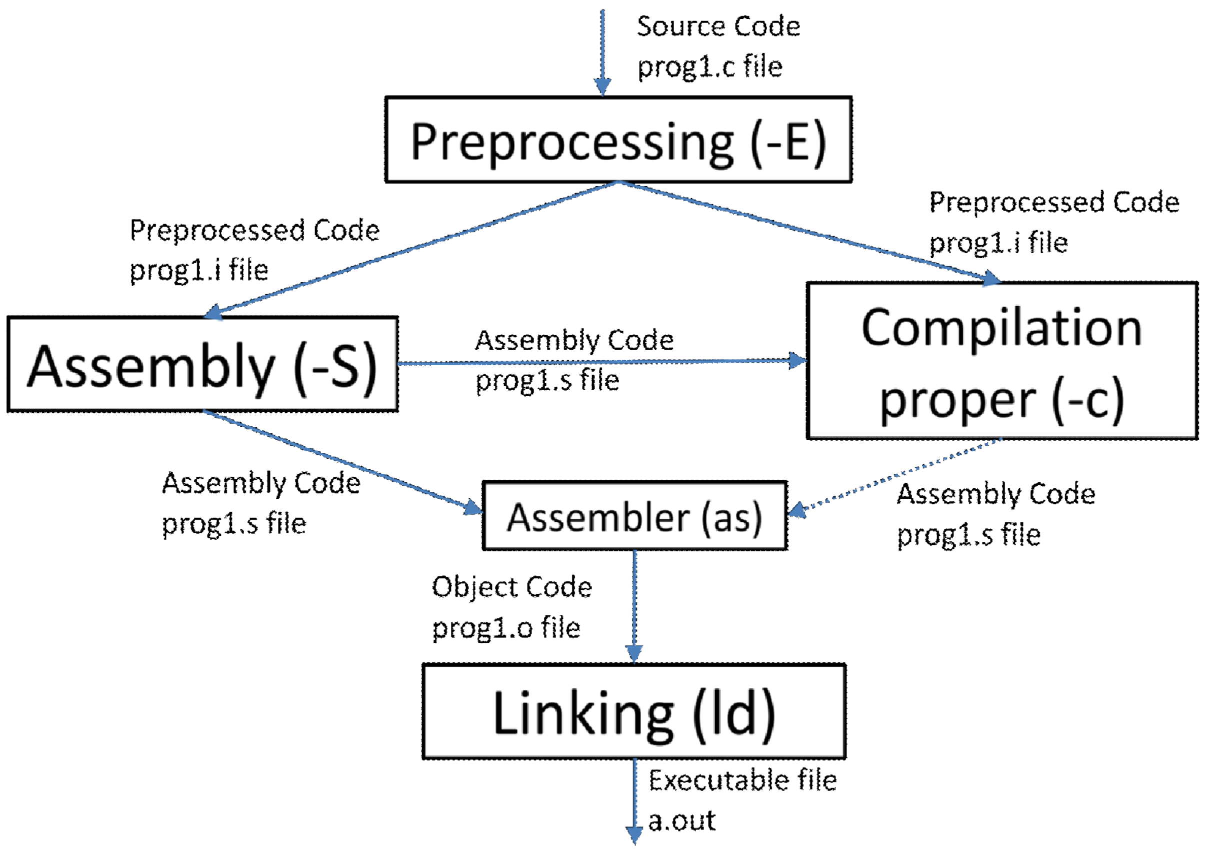

Fig. 2 Stages of Compilation in GCC

11. Put `prog1.c` file in an empty directory. Now, compile the prog1.c using gcc but with an extra flag `-save-temps`
    - Open a terminal and go to the folder where prog1.c is saved.
    - Type `ls` and then enter it will show prog1.c
    - `gcc -Wall -save-temps prog1.c -o prog1`
    - `ls` and then enter it will display `prog1.c prog1.i prog1.o prog1.s`
    - `GCC` flag `-save-temps` stores the usual “temporary” intermediate files permanently; place them in the current directory and name them based on the source file.
12. Preprocessing Steps
    - Basically C Preprocessor is responsible for 3 tasks namely:
        i. Text Substitution
        ii. Stripping of Comments
        iii. File Inclusion

Practice Lab 1 – ICP (CSD-101)

- Text Substitution and File Inclusion is requested in our source code using Preprocessor Directives.
- The lines in our C code that begin with the "#" character are preprocessor directives.
- The intermediate file that is generated after this stage is a .i file.
- Type `gcc -E prog1.c > prog1.i`

13. Compilation Steps
    - It will convert the preprocessed source code into assembly language code of the local CPU.
    - By using "-S" flag with gcc we can convert the pre-processed C source code into assembly language without creating an object file:
    - `gcc  -Wall -S  prog1.i -o  prog1.s`

14. Assembler
    - MACHINES (i.e. a computer) can understand only Machine-Level Code.
    - So we require an ASSEMBLER that converts assembly code in prog1.s file into machine code.
    - The Assembler as in gcc can be invoked by typing
      `as prog1.s -o prog1.o`
    - The resulting file 'prog1.o' contains the object code for 'prog1.c' program.
    - Alternatively, by using "**-c**" flag of gcc we can convert the assembly code into object code: `gcc -c prog1.s`

15. Linker
    - In the linking step makes sure that all the undefined symbols in code are resolved.
    - An undefined symbol is one for which there is no definition available. e.g. in our code, there is no definition of printf ( ) function.
    - So in order to make our program execute correctly, the definition of this function need to included or at least linked to our code.
    - Type `gcc prog1.o`
    - Alternatively, you may use linker directly with all required library by typing `ld prog1.o -lc --entry main` Do not worry if `a.out` is not created here. Your system may require more libraries to be included. Move on .

16. cat command
    - `cat prog1.c` and enter
    - It displays the content of the file

17. To print date

Practice Lab 1 – ICP (CSD-101)

- Type `date` and then enter
- It will display the current date

18. find command
    - Type `find /home -name prog1.c`
    - If you're in your /home directory, it will go through every directory and find file that has the name 'prog1.c' in it

19. whoami
    - Type `whoami` and then enter
    - It will display your username.

20. whatis
    - Type `whatis grep`
    - grep (1) - print lines matching a pattern
    - `whatis cp`

21. Copies files and directories(cp)
    - Open a terminal
    - Write `cp Source Dest`

22. Remove files and directories
    - Type `rm filename`

23. Log you out from the system
    - Type `exit/quit`

24. man command
    - Type `man command-name`
    - Read man pages of the following commands
      - i. nice
      - ii. touch
      - iii. chmod
      - iv. chgrp
      - v. more
      - vi. less
      - vii. cat
      - viii. ar
      - ix. ls
      - x. gcc option for creating shared object file (*.so) dynamic Link Library

Please Shut Down the system before leaving the lab