Useful Linux commands, creating, editing, compiling and running a C program

Here, we describe some useful Linux commands that you can also use in macOS. These commands are to be typed in a terminal. Further, these commands are helpful for organizing the files and directories in your system. Similar commands are also available in windows (please refer to a suitable document). *Please note that only basic features of the commands are discussed. To learn more about these commands, please see the manual page for each command. For example, to learn about "cp" command, type "man cp" in the terminal.*

$ pwd ↩

The "pwd" command stands for "Print Working Directory." It displays the full path of the current directory (or folder) that you are currently working in the terminal. Below is the output when I open a terminal and execute the "pwd" command in my macOS:

```
/Users/sghorai
```

The displayed path is my home/default directory.

$ ls ↩

The "ls" command lists the files and directories in the current working directory. By default, it does not show hidden files (those starting with a dot). Below is the output of "ls" command in my home directory

```
a.f                     google-python-exercises mida.tex                PAPER
ContD.pdf               imp.pdf                 MNNIT                   Pictures
cs_proj                 imp.tex                 Movies                  PRAC
Desktop                 INC                     MTH658                  Public
Documents               Library                 Music                   RPP
Downloads               mida.pdf                NUF                     STAT
```

It shows that both directories (such as INC, MTH658, etc.) and files (such as a.f, mida.tex, etc.) are present.

$ ls -l ↩

```
total 24984
-rw-r--r--@   1 dhiraj  staff      3927 23 Nov  2024 a.f
-rw-r--r--@   1 dhiraj  staff  12625046  2 Oct  2024 ContD.pdf
drwxr-xr-x   10 dhiraj  staff       320 28 Mar  2025 cs_proj
drwx------+  17 dhiraj  staff       544 25 Oct 21:10 Desktop
drwx------+  10 dhiraj  staff       320 16 Aug 07:31 Documents
drwx------+ 229 dhiraj  staff      7328 24 Oct 22:05 Downloads
drwxrwxr-x@  10 dhiraj  staff       320 14 Oct 19:29 google-python-exercises
-rw-r--r--@   1 dhiraj  staff     37087 25 Aug 08:20 imp.pdf
-rw-r--r--@   1 dhiraj  staff      3391 25 Aug 08:20 imp.tex
drwxr-xr-x@  35 dhiraj  staff      1120 16 Jan  2025 INC
drwx------@ 109 dhiraj  staff      3488  2 Oct 20:01 Library
-rw-r--r--@   1 dhiraj  staff    107073  3 Sep 22:18 mida.pdf
-rw-r--r--@   1 dhiraj  staff      3616  3 Sep 22:18 mida.tex
drwxr-xr-x   11 dhiraj  staff       352 12 Sep 19:37 MNNIT
drwx------    4 dhiraj  staff       128  3 Mar  2024 Movies
drwxr-xr-x@  50 dhiraj  staff      1600 24 Aug 19:20 MTH658
drwx------+   6 dhiraj  staff       192  1 Sep  2024 Music
drwxr-xr-x   23 dhiraj  staff       736  8 Feb  2025 NUF
drwxr-xr-x    4 dhiraj  staff       128  2 Dec  2024 PAPER
drwx------+   5 dhiraj  staff       160  5 Nov  2024 Pictures
drwxr-xr-x    8 dhiraj  staff       256 28 Mar  2025 PRAC
drwxr-xr-x+   5 dhiraj  staff       160 20 Oct 18:22 Public
drwxr-xr-x   16 dhiraj  staff       512 25 Oct 20:24 RPP
drwxr-xr-x   10 dhiraj  staff       320 30 Aug 19:07 STAT
```

The "ls -l" command lists the files and directories of the current working directory in a "long listing" format, which includes information about each file and directory, such as permissions, ownership, size, and modification date. Again, by default, it does not show hidden files (those starting with a dot). Shown above is the output of "ls -l" command in my home directory. The permission field (first column) starts with 'd' for a directory.

```
$ mkdir directory_name ↩
```

The "mkdir directory_name" creates a new directory named "directory_name" provided the directory "directory_name" does not exist in the current path. From the output of "ls" command (shown earlier), we observe that there exists no directory named "Cprog". We thus create a new directory "Cprog" by executing the command

```
$ mkdir Cprog ↩
```

Now, the output of "ls" command shows the new directory "Cprog" (see the third entry in the leftmost column):

```
a.f                    google-python-exercises  MNNIT    PRAC
ContD.pdf              imp.pdf                  Movies   Public
Cprog                  imp.tex                  MTH658   RPP
cs_proj                INC                      Music    STAT
Desktop                Library                  NUF
Documents              mida.pdf                 PAPER
Downloads              mida.tex                 Pictures
```

```
$ cd directory_name ↩
```

The "cd" in "cd directory_name" stands for "change directory". The command "cd directory_name" changes the current working directory to directory_name (directory_name must exist in the current directory). For example, executing the command

```
$ cd Cprog ↩
```

leads us to the path

```
/Users/sghorai/Cprog
```

that can be verified by executing "pwd" command. Executing "ls" command shows empty content of the newly created directory. Let us make another directory "Intro" (execute "mkdir Intro") and execution of "ls" shows the previously created folder "Cprog" has the folder "Intro" as its content. Executing the command

```
$ cd Intro ↩
```

leads us to the path

```
/Users/sghorai/Cprog/Intro
```

which has empty content. We create the source file "hello.c" using the technique discussed in Lecture 1. Compiling with "gcc hello.c" produces default executable file "a.out" which can be run using "./a.out". Execution of "ls" shows the created folder "Intro" has the files "hello.c" and "a.out" as its content.

```
$ cd .. ↩
```

The "cd .." command changes the current working directory to its parent directory. For example, see the following self-explanatory commands:

```
$ pwd ↩
```

```
/Users/sghorai/Cprog/Intro
```

```
$ cd .. ↩
```

```
$ pwd ↩
```

```
/Users/sghorai/Cprog
```

```
$ cd ↩
```

The "cd" commands changes current path to your home (or default) directory. For example, see the following self-explanatory commands:

```
$ pwd ↩
```

```
/Users/sghorai/Cprog/Intro
```

```
$ cd ↩
```

```
$ pwd ↩
```

```
/Users/sghorai
```

```
$ rmdir directory_name ↩
```

The command "rmdir directory_name" removes the directory "directory_name" provided it is empty. In case of the directory "directory_name" is non-empty, you need to go inside the directory and remove the files (to be discussed shortly) and directories (if any). Then, remove "directory_name" from its parent directory.

```
$ cp src_file dest_file ↩
```

The command "cp src_file dest_file" copies the contents of the "src_file" to the "dest_file". If the file "dest_file" doesn't exist, it is created, and the content is copied into it. However, if the file "dest_file" already exists, it is overwritten without warning.

```
$ cp -i src_file dest_file ↩
```

The command "cp src_file dest_file" copies the contents of the "src_file" to the "dest_file". If the file "dest_file" doesn't exist, it is created, and the content is copied into it. However, if the file "dest_file" already exists, the command will pause and ask for confirmation before overwriting the existing file. Here, "-i" enables interactive mode.

```
$ mv old_file new_file ↩
```

The command "mv old_file new_file" rename the "old_file" to "new_file". If a file with the name "new_file" already exists, it will be overwritten without prompting for confirmation.

```
$ mv -i old_file new_file ↩
```

The command "mv -i old_file new_file" rename the "old_file" to "new_file". If a file with the name "new_file" already exists, the command will ask for confirmation before overwriting the existing file. If the file "new_file" doesn't exist, it will simply rename "old_file" without prompting.

```
$ rm file_name ↩
```

The command "rm file_name" is used to remove (delete) the file "file_name". Files deleted using "rm" are typically removed permanently and do not go to a "recycle bin" or "trash." By default, "rm" will not prompt for confirmation before deleting a file.

```
$ rm -i file_name ↩
```

The command "rm -i file_name" is used to remove (delete) the file "file_name". Its purpose is same as that of "rm", the only difference is that "rm -i" will prompt for confirmation before deleting a file.

```
$ cat file_name ↩
```

The command "cat file_name" will output the entire content of "file_name" directly to your terminal.

```
$ more file_name ↩
```

The command "more file_name" is used to view the contents of "file_name" one screenful (of terminal) at a time. This is particularly useful for large files that would scroll off the screen if displayed entirely. It also allows you to navigate through "file_name" using various keys (e.g., return/enter for next line, spacebar for the next screenful, q to quit).

Return to Lecture Topics