Notes on Linux operating system

Written by Jan Mrázek for the MIBO(BCMB)8270L course
last updated: Jan 9, 2007

UNIX: operating system
Linux: free version of UNIX

Basic communication with a Linux computer is through a command line terminal (no graphics) or "shell". That is, the commands are typed on the keyboard instead of clicking on things on the screen with a mouse.

Most modern UNIX/Linux platforms offer a graphical user interface similar to Windows but its use over the network can be problematic. We will use the shell.

One of the advantages of using Linux and Linux shell is that you can access the computer from anywhere through the Internet.

We will access our Linux server via secure shell (ssh) and secure file transfer protocol (sftp), which have to be installed on your computer (PC or Mac). If you want to use your home computer for access to our server download and install SSH 3.2.9 at http://www.sitesoft.uga.edu/ (SSH 3.2.9 includes both ssh ad sftp, no need to install a separate sftp). For the purpose of this course you have been provided with accounts on our lab server willow.mib.uga.edu (type it in when SSH asks you for a host name).

ssh allows you to open a Linux shell window on your desktop. In the Linux shell, you can type in commands, run programs and manipulate files on the remote computer. ssh itself does not allow you to move files between your computer and the remote Linux computer. That's done by sftp.

Linux file system

Files in Linux are organized in directories (analogous to folders in Windows). The root directory is simply "/". Users have their files in their home directories in "/home/". For example, my home directory (user name "jan") is "/home/jan/".
Special directory names: "./" refers to the current directory; "../" refers to the directory one level above the current directory; "~/" refers to your home directory.

File names: unlike Windows, Linux differentiates upper case and lower case letters in file names. That is, the file names "MyFile", "Myfile", "myfile", and "MYFILE" relate to four different files.

Hidden files: filenames that begin with "." (period) are hidden files. These are usually system files that don't show up when you list directory content. Under normal situations you don't have to worry about the hidden files. Just remember not to use "." at the start of a file name.

Files are assigned "permissions" that define who has access to them and what kinds of access. Basic types of access are read, write and execute. Read access allows you to read the content (e.g., make your own copy) of a file. Write access allows you to delete, modify or overwrite files.

Execute access is required to execute programs or for directories to be able to access their content. You have write access only to your home directory, that is, unless specifically given access you will not be able to store files elsewhere on the file system. At the same time, you have read and execute (but not write) access to all system files and programs that you will need. File permissions can be changed with the "chmod" command but it is unlikely that you will need it in this course.

Linux does not use extensions to recognize the type of a file (like Windows) but you can include "." in file names. I often give files Windows-like extensions (like .txt, .pdf, .bat, etc.) in order to remind myself what kind of a file it is and also that I don't have to change the extension when I transfer the file to Windows.

I recommend that you use directories to keep your files organized.

Wildcards: Many Linux commands allow wildcards in the file names. Most useful are "*" (matches any text) and "?" (matches any single character). For example, "*" matches all files in a directory, "a*" matches all files that start with "a", "a*z" matches all files that start with "a" and end with "z", "a?z" will match things like "atz", "a2z", "a.z" but not "a2.z", ????? will mach all files with names exactly 5 characters long. You can use it with directories, too, e.g., the command "ls ~/*/*.txt" (see also below) will display file names of all files ending with ".txt" in directories one level inside your home directory.

Linux Shell

Linux commands have none, one or more parameters. For example, "ls /home/jan" will list the content of my home directory ("/home/jan" is a parameter). "ls" with no parameters will list content of the current directory. "cp file1 file2" will read "file1" and make a copy named "file2". As you see, the order of the parameters is usually significant.

In addition to parameters, most Linux commands have options. Options start with "-". For example, "ls -l /home/jan" will list additional information about each file (without the -l it will only list the file names). Options modify behavior of the command.

There are several types of shells that have some minor differences. The shell we use is called "bash".

List of Basic UNIX/Linux shell commands

Following is just the very basic list of some useful commands. More can be found on the internet (e.g., type "linux shell commands" in Google). Check http://linux.org.mt/article/terminal for a beginner's tutorial including some more advanced tricks.

Ctrl-C: pressing ctrl-C will stop the running program or command

Output redirection: ">" will redirect the output normally going on the screen into a file. E.g., "ls > List" will list all files in the directory but store it in a file named "List" instead of displaying it on the screen. This can be used with any command or program.

Files and directories

- pwd: shows current directory.
- cd directoryname: makes directoryname your current directory. cd with no parameters switches to your home directory
- ls directoryname: lists contents of directories. Use ls -l for more information about the files. You can limit the list with wildcards (e.g., "ls /home/mydirectory/*.txt")
- mkdir directoryname: creates a new directory.
- cp source destination: makes a copy of a file named "source" to "destination".
- cp –r source destination: copies a directory and its content
- mv source destination: moves a file or directory.
- rm filenamelist: removes/deletes file(s). Be careful with wildcards.
- rm –r directory: removes directory (-ies) including its content

Viewing files

- cat file: prints the whole file on the screen. Can be used with redirection, e.g., "cat Small1 Small2 Small3 > Big" will concatenate files Small1, 2 and 3 into a single file named Big. You can also use wildcards, e.g., "cat *.txt > VeryBigFile".
- more file: shows a file page by page. When viewing the files press "Enter" to move forward one line, "Space" to move forward one page, "b" to move one page backward, "q" to end viewing the file, type "/patern" to jump to the next occurrence of the text "pattern". These are just few things you can do with more.

Getting more information about commands

- man command: will show the manual page (complete reference) for the command. Scroll up and down as in more.
- info command: similar to man but contains more verbose information.
- Many commands will show basic description when run with -h or --help option.

Other

- passwd: change your password
- exit: close the terminal
- chmod: change permissions for a file/directory. This allows you to set access to your files for other users or turn a text file into an executable. You will hardly need it in this course.
- cc program.c –o ~/bin/program: compiles a program written in C language and creates an executable file in your bin directory. ~/bin/ is a standard place for storing executable (binary) files. –lm option has to be included if the program uses math library. –O option will optimize the program (make it run faster).

Editing

You can use command-line editors like vi, emacs or ed to modify contents of text files. However, they take a while to get used to and unless you already know them you may be better off using sftp to open the file in Notepad on your (Windows) computer. Just make sure that the sftp is set to ASCII (text) mode when you do this.

Running programs

Once created, executable programs can be run as any regular command just by typing the program name and any parameters and/or options. In fact, the shell commands are programs.

Executable program files can be created in different ways (included for your reference):

- When written in a language like C, C++, or Fortran, the "source code" (human-readable text file prepared by a programmer in an appropriate syntax) can be converted into an "executable" (machine-readable binary file) by a special program called compiler (see cc command above). Compiling complex program packages may be complicated and programmers often create "make" files, which contain all instructions how the program should be compiled and assembled in a single file and are executed with the "make" command. Program packages are increasingly often distributed as rpm files where the installation is completely automatic making it very easy for the user (generally only the system administrator can install such packages).
- You can create shell scripts (text files containing lists of commands to be executed) and give them executable permission. That will effectively turn them into executable programs.
- Interpreted languages such as Perl differ from compiled ones in that you do not prepare an executable file but instead each command is "interpreted" at the time of execution. This makes such programs slower to run but eliminates need for compilation. Perl programs (scripts) are run by typing "perl program" followed by parameters and options.