Introduction to HPC Programming

3. Introduction to Linux: The Supercomputers' OS

Valentin Pavlov <vpavlov@rila.bg>

About these lectures

- This is the third of series of six introductory lectures discussing the field of High-Performance Computing;
- The intended audience of the lectures are high-school students with some programming experience (preferrably using the C programming language) having interests in scientific studies, e.g. physics, chemistry, biology, etc.
- This lecture provides an overview of Linux OS and provides basic information which every user of a supercomputer ought to know.
- Some of the slides provide only pointers for the lecturer (and the readers) to elaborate on since its not feasible to describe all-things-Linux here!

Why Linux?

- At the end of the last lecture we showed the 10 fastest supercomputers on this planet.
- They are made by different teams, have different architectures and one thing in common – their Operating system is Linux;
- Moreover—94% (or 469) of the Top 500 supercomputers run Linux;
- Of the remaining 31, 28 run other Unix variants and only 3 run Windows OS;

What is the Operating System?

- The Operating System (OS) is the software that controls the hardware resources of a given machine and provides common view of that hardware to the rest of the programs.
- Usually the OS is divided into two big areas — “kernel” and “shell”.
- The above definition in greatest extent is applicable to the “kernel”, while in the “shell” consists of a large set of well known programs and libraries, which the users and other programs (application software) use.

GNU, Linux, Unix and other *x-es

- Linux stems from UNIX – multitasking and multiuser commercial operating system created by AT&T in 1969;
- Strictly speaking, Linux is just the **kernel** of the operating system which we call Linux and whose proper name is GNU/Linux.
- The **shell** of this operating system is made up of a collection of system programs provided by the GNU initiative;

GNU, Linux, Unix and other *x-es

- GNU is a recursive acronym—**G**NU is **N**ot **U**nix and in cnontrast to the original UNIX it is free of charge, as is the Linux kernel itself.
- The GNU initiative has nothing to do with Linux and in principle the GNU software stack can be used on other operating systems;
- Other UNIX variants include: Android, Mac OS X, AIX, FreeBSD, NetBSD, OpenBSD, Solaris, HP/UX, etc.

Distributions

- Everything in GNU/Linux is free and will always be;
- In theory, one can download all necessary programs from Internet and build his own variant of GNU/Linux;
- This is usually not happening [anymore], and users rely on ready-made collections of Linux-based software, so called “distributions”;

Distributions

- The distributions differ in the installation procedure, configuration and updates, but otherwise the differences are insignificant;
- If you are familiar with certain distribution, getting used to another one is very easy;
- Some of the widely used distributions are: Debian (including Ubuntu), SuSE (including SuSE Linux Enterprise Server), Red Hat (including Red Hat Enterprise Linux, CentOS, Fedora), Slackware, etc.

Terminal and graphical user interface

- All distributions offer the users the possibility to work in graphical environment, as well as in terminal mode;
- The Graphical User Interface (GUI) is easier to use for everyday tasks, including usage by non-specialists, but because of this it is quite limited;
- The terminal mode provides access to the full range of functionality of the OS and is also better suited for remote access;
- In the rest of the lecture we will provide some of the more widely used commands applicable in terminal mode and will not pay attention to the GUI

Login/Logout

- In order to be able to use a Linux (or other UNIX) machine and the software installed on it, you need to get access—an user account.
- Any given OS has a number of accounts, including one special user called `root`, who is concerned with administrative functions and has access to everything in the OS;
- Every machine has an administrative procedure through which user accounts are opened. You should follow whatever policy the site enforces to get your user and password.

Login/Logout

- The access is realised through a `login` procedure, regardless of wether the access is local (using the keyboard and monitor of the machine itself) or remote, through the use of some remote access protocol as telnet or ssh;
- When the login succeeds, you get access to the *shell prompt*;
- By using the `logout` command, you can disconnect from the machine;

shell prompt

- In terminal mode the interaction with the OS is through the so called *shell*;
- There are different shell programs (sh, bash, ash, csh, ksh, etc.). They are mostly equivalent in functionality and its up to the user to get aquainted with one or the other. Lately, `bash` is usually the shell of choice, but again this is a matter of user preference.

shell prompt

- Simplisticly put, the shell runs an endless loop in which it reads commands and then executes them. This is repeated forever until a `logout` command is issued.
- The indication that the shell is ready to accept a new command is called *the prompt*.
- Example:
- `vpavlov@metis:/home/vpavlov/Documents$`

Basic commands

- `ls` – shows the list of files in the current directory
- `pwd` – displays the current directory;
- `cd` *directory* – change the current directory;
- `cat` *textfile* – displays the contents of a text file;
- `man` *command* – shows help about a command;
- `info` *command* – shows extended help about a command;
- `passwd` – changes the account password;

Help!

- Your best friend in Linux land (after your favourite Internet search engine) is the command `man`, which you can use to get information (help) about all commands (try `man man`);
- You can use the arrows and PgUp/PgDn for navigation;
- In order to return to the shell prompt, use `q`
- Warning: everything in Linux is case-sensitive; this means that `q` is different from `Q` and `man` is different from `Man`;

More help!

- You can get even better help by using the `info` command; try `info info`
- It offers a cross-referenced hyper-textual help with hierarchical organization, navigation, etc. Again – use `q` to exit to the shell prompt;
- Observation: everything in Linux (UNIX) is very logical and minimalistic, so you don't waste too much energy typing. All frequently used commands are only two letters long!

What is a file?

- A file is a resource for storing data, usually on some permanent memory (e.g. hard disk)
- However, in Linux (UNIX) things are more general: everything is a file; if it is not a file, it is process. File types:
  - directory: a list of other files;
  - link: allows the same file to be seen under a different name and/or place;
  - special file (e.g. device): provides mechanism for input/output to hardware devices and process control;
  - socket: means for interprocess and network communication;
  - pipe: another interprocess communication means, but only between processes on the same machine;

File system

- The file system is an hierarchical database in which each record is a file;
- More or less its a tree in which every node is a file. If the file is a directory, it can be the root of a sub-tree, etc.
- In Linux, the file system has a single root whose name is `/`; compare this to Windows, in which each device has its own root, e.g. `C:`, `D:`, etc.
- On the directories of a file system you can *mount* other file systems. Thus, altough there is a single root of the file system, it can actually contain a mixture of several file systems, residing on several devices and even remote servers.

Paths

- Each file in the file system has an address so we can point it out and differentiate it from all other files;
- This address is called *a path*. It is formed by concatenating the directories we need to pass through in order to reach the file, followed by the file name; the different directories and the file name are separated by `/`
- There are two kinds of paths: absolute and relative. The absolute paths start with the root of the file system, and the relative – from the **current working directory**
- Example of an absolute path: `/usr/bin/date`
- Example of a relative path: `Documents/SC-COURSE-2013/lecture-03.pdf`

Important things to elaborate on

- Special directories: `~`, `.` and `..`;
- Important directories in the root of the file system: `/usr`, `/etc`, `/lib`, `/dev`, `/proc`, `/home`, `/var`, etc.
- Frequently used `ls` arguments;
- Hidden files;
- Detailed file information (type, access modes, owner, size, modification date, name);
- Different file systems: ext4, reiserfs, nfs, gpfs, pvfs, lustre, etc.
- `df`, `du`, `/etc/fstab`, `mount`, `$PATH`, `which`, `find`, `cp`, `mv`, `ln`, `mkdir`, `rmdir`, `rm`, `grep`, `more`, `less`, `chmod`, `chown`

Processes

- As already stated, if something is not a file, it is a process;
- A process is a program that is being executed at some moment in time;
- Some processes end quickly (e.g. `ls`);
- Others take a lot of time to finish (some intensive calculation);
- Some are interactive (e.g. your favourite browser);
- There are resident processes that run in the background (daemons), automatic processes and periodic processes, which run at certain times of the day/week/month;

Processes

- Linux is a multitasking OS—it allows multiple tasks (processes) to run simultaneously, but this is an illusion created by quickly switching the process that “owns” the CPU;
- The more CPUs the system has, the better the illusion for simultaneousity is;
- Linux is also a multiuser OS—the processes that run simultaneously can be run by different users at the same time. This is what allows a computer or a supercomputer to be used independently by many people!

Important things to elaborate on

- stdin, stdout, stderr, redirection and pipes;
- `bg`, `fg`, `&`;
- `ps`, `pstree`, process attributes;
- `top`, `uptime`;
- `kill`, `killall`, `SIGTERM`, `SIGKILL`, `SIGHUP`;
- `cron`, `crontab`, `at`;

Environment

- The *Environment* includes variables which can be set/reset/unset and through which the behaviour of future processes can be controlled;
- Environment variables can be set/unset for the extent of the whole shell session, or just for the next process to run;
- There are many standard environment variables: PATH, PS1, CC, CFLAGS, LD_LIBRARY_PATH, etc.
- The user and application programmers are free to use whatever environments they deem necessary.

Other useful commands to elaborate on

- `ssh`, `netstat`, `ifconfig`;
- `vi`, `vimtutor`;
- `env`, `export`;
- `sudo`, `apt-get`, `yast`;
- `llsubmit`, `llq`, `llcancel`;