Faculty of Computer Science, Dalhousie University *10-Sep-2018*

CSCI 2132 — Software Development

Lecture 3: Unix Shells and Other Basic Concepts

Location: Chemistry 125 Instructor: Vlado Keselj
Time: 12:35 – 13:25

Previous Lecture

- **Introduction to UNIX**
- Reading: Chapter 1 of the UNIX book
- Operating System overview
  - Onion skin model
- OS functionalities
- UNIX history
- UNIX philosophy and main features

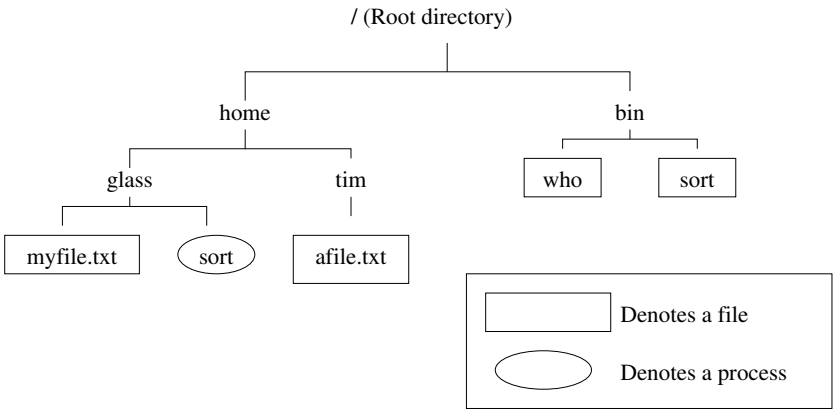
Some Hardware Concepts

- Central Processing Unit (CPU)
- Random-Access Memory (RAM)
- Read-Only Memory (ROM)
- Disk Memory (Hard Disk Drive, CD-ROM, etc.)
- Graphics card
- Network card (Ethernet card)
- Peripherals: keyboard, monitor, mouse, etc.

Some Important Unix Concepts

- **File:** collection of data (sequence of bytes) stored on disk (or other external storage)
- **Program:** a file containing machine code and data that can be loaded into memory and run
- **Process:** Program that is loaded in memory and running
- **Owner** of a File or Process: Processes and files have owners and may be protected against unauthorized access
- **Hierarchical directory structure** is suppored in Unix file system
- **Location** of a file or process: Files and processes have a location within the directory hierarchy. A process may change its location and the location of a file
- **System Calls:** Unix provides services (system calls) for creation, modification, and destruction of files and processes

Directory Hierarchy



Shells

- a program used by the user to interact with the system
- used to run other programs in the system
- built-in commands and utilities (external commands)
- used to automate many tasks
- A UNIX characteristic is existence of multiple shells (even Window managers); e.g.:
  - **Bourne Shell:** sh, bash (Bourne-Again Shell); we will focus on this one
  - Korn Shell: ksh
  - C Shell: csh, tcsh
  - Z Shell: zsh

3 Getting Started in Unix

Getting Started in Unix

- Covered in the lab
- Mandatory exercise: login to your bluenose account using ssh
- Options: putty, MobaXterm, Mac terminal, Linux terminal
- More options: use of VirtualBox
- Main learning objective:
  - be able to open one or more Terminals with ssh login to your bluenose account in each of them
- ask TAs, use Learning Center if there are any issues

3.1 Logging In

Logging In

- You can choose Windows or Mac environment in some labs
- Windows: you can use the `putty` program
- On Mac: open a Terminal and type:
  `ssh` *CSID*`@bluenose.cs.dal.ca`
- Instead of *CSID* use your CS userid (CSID)

– On Linux: similarly to Mac, you open the terminal and type the same command:
`ssh` *CSID*`@bluenose.cs.dal.ca`

Your first step is to login into the lab computer and after that to login into the `bluenose` server. Some labs have only Windows machines (as in Teaching Lab 3), or you have a choice of logging in into the Windows or Mac environment. You can choose any of them. You may have a chance to use a Linux computer, and that would be very convenient since your local operating system would be very similar to the server operating system.

On Windows: If you use a Windows environment, you will need to use a program named `putty` to login to the `bluenose.cs.dal.ca` server. This will be explained in the next step. The PuTTY program has already been installed in the lab machines. You can also install it on any other machine since it is freely available on the Internet.

A security note about downloading PuTTY: It is important to have in mind that since PuTTY provides secure access to computers we should make sure that we have the original program installed and not a copy that may have been potentially changed in a malicious way. The current authoritative link for downloading PuTTY seems to be `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

On Mac: In Mac OS environment, you can click on the search image in the upper right corner and type 'Terminal' to find the `Terminal` application. Once you open the terminal, you can login to the `bluenose` server by typing:

```
ssh CSID@bluenose.cs.dal.ca
```

where `CSID` is your CS userid.

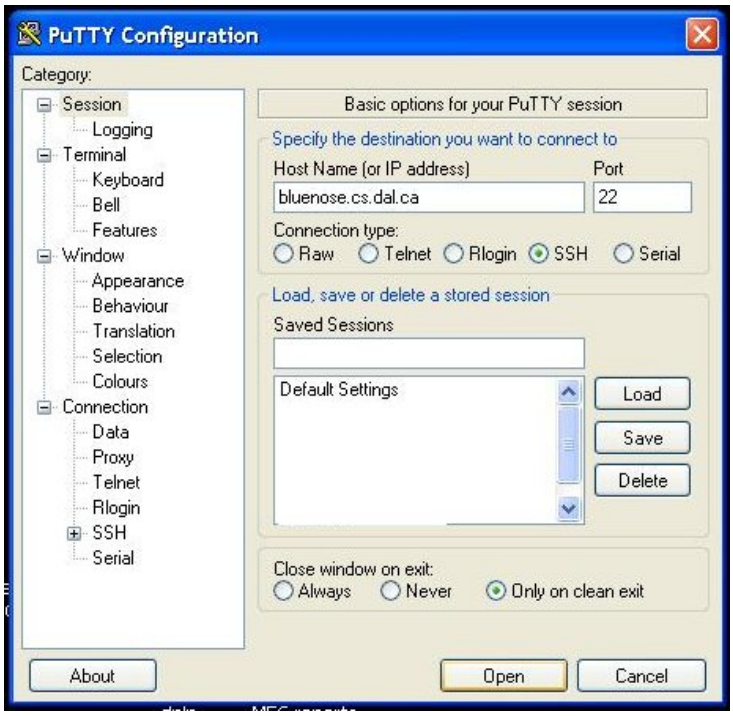
On Linux: In a Linux environment, you should, similarly to Max, open a terminal and type:

```
ssh CSID@bluenose.cs.dal.ca
```

where `CSID` is your CS userid.

Running PuTTY

– Double-click the PuTTY icon, and the following window should appear:



You should fill in the basic information: `bluenose.cs.dal.ca` for the Host Name. Make sure that the port number is 22; i.e., Connection type is SSH. SSH stands for Secure SHell protocol, which uses encryption to keep your password and communication with the server secure.

You click 'Open' and the login process should start. You are likely to receive a warning about an unknown host key. Normally, this is something that you should be careful about and try to make sure that the offered fingerprint matches the fingerprint of the server, but in a secure network it should be safe to accept this connection. Once accepted, the host key is stored with PuTTY and this warning should not appear again.

More details will be given in the Labs.

Shell Prompt

- After login, we communicate with a shell program, which responds with a *prompt* string; e.g.: `CSID@bluenose:~$`
- or, some shells respond with `%` or `#`
- User can change the prompt string
- We type commands, followed by enter key, and the shell will respond
- We can edit the command line before pressing Enter

After opening a PuTTY connection, a terminal window opens and we are prompted to enter a login userid and password, after which are logged in and can interact with the system. We will actually communicate with a shell program, which typically respondes with a prompt string, such as

```
username@bluenose:~$
```

indicating that it is waiting for some input from us. In the previous prompt string, the word `username` would actually be your username (also called userid). The prompt string may be different from shell to shell, and it may be customized. The characters `%` and `#` are also found as prompt strings.

We can now enter a command using keyboard. The shell allows us to edit the command line before submitting it by pressing the Enter key.

Running a Utility

- Enter a command, which can be a utility or a built-in command
- A utility is a program with the same name: shell finds it and runs it
- Example: `date`
- Output:
  `Tue Sep 10 22:35:49 ADT 2013`
- More examples:
  - `clear`
  - `passwd`

The commands that we can enter in a shell are generally divided into *internal shell commands* and *external commands.* The *internal shell commands,* also known as built-in commands, are commands that shell directly interprets and executes. the *external commands,* also known as utilities, are external programs and shell executes them by running this programs.

A shell command, internal or external, is executed by typing it on a line, followed by optional arguments, and pressing "Enter". Some examples of commands are `date`, `clear`, and `passwd`. The command 'date' can be used to display current time and date, the command 'clear' clears the screen, and the command 'passwd' is used to change password of a user.

Command-line Arguments

- Utilities can accept arguments
- For example, date utility allows us to choose format of the output, as in:
  `date +%Y-%m-%d-%H-%M-%S`
- Output:
  `2013-09-10-22-35-49`
- To explore usage of 'date', type:
  `man date`
  (use 'q' to exit)

The 'man' Utility

- `man` comes from 'manual pages'
- the command 'man' is the standard way of providing usage descriptions for various commands in Unix
- examples:
  - `man man`
    to read about the command 'man'
  - `man -k directory`
    to search commands using keyword 'directory'
  - `man 2 rmdir`
    to specify the section number of the man page in case of an ambiguity

The Unix command 'man' is used to get the manual page about a command or a system function. The 'man' command has different options or flags, which can be used. The simplest way to use the man command is:

`man` *a_command_name*

After this command we will get the manual page of the command '*a_command_name*', and we can read it on the screen. Some examples are:

```
man clear
man man
```

We can also specify a keyword and request that the man command gives a list of utilities, whose descriptions include the keyword, with a short description of each utility. We can then look for the utility that performs the task that we need to be done, and we can use the man command again to read about this utility. This syntax of the man command is:

```
man -k keyword
```

or, as an example:

```
man -k password
```

Special Shell Characters (Metacharacters)

- Characters which are interpreted in a special way when typed in a Unix terminal
- They are usually control characters obtained by pressing Ctrl key and then typing another key while Ctrl is pressed, e.g., Ctrl-C
- We will learn two of them:
  - Control-C (`^C`) is used to terminate a process
  - Control-D (`^D`) is used to signal end of file
- The command `stty -a` can be used to show many special characters

Metacharacters or Special Characters are characters that are interpreted in some special way by shell when typed. Two important special characters that we should mention now are Control-C (^C, or Ctrl-C) and Control-D (^D, or Ctrl-D). They are obtained by pressin the Ctrl key on keyboard, and while holding it down, we press the key 'c' or the key 'd'. The Control-C character can be used to terminate a running process, and Control-D can be used to indicate end-of-input to a program.

Input, Output, and Error Channels

- Remember that there are three default I/O channels for each UNIX program, i.e., three **standard I/O streams:**
  - stdin (standard input)
  - stdout (standard output)
  - stderr (standard error)
- Commands so far mostly used stdout only
- Command 'cat' frequently uses stdin and stdout

Each Unix process is initially assigned three I/O (input/output) communication channels:

1. stdin, or standard input, used to read textual data as input. For a program executed from shell (i.e., terminal or command line), the standard input is assinged to the keyboard by default.
2. stdout, or standard output, used to output data. For a program executed on a terminal, by default the output will be written to the terminal.
3. stderr, or standard error, used to output any error or warning messages from a C program. For a program executed on a terminal, by default the standard error is printed to the terminal, possibly intermixed with the standard output.

'cat' example

- Consider the following example (`^D` is Control-D):

```
$ cat > hamlet.txt
To be or not to be
that is the question
^D
$ cat hamlet.txt
```

Editors

- The 'cat' example was a way to create a textual file
- A more convenient way is to use a textual editor
- Unix offers several editors, for example:
  - emacs — to be used as main editor in this course,
  - vi or vim — also a major Unix editor,
  - pico, nano, and some others
- Editors are covered more in the labs

Logging Out

- On some shells typing Control-D is sufficient to log us out
- Sometimes it is disabled since a user can easily type it by mistake
- Commands 'logout' and 'exit' are available