

SYSTEM & NETWORK TECHNOLOGIES

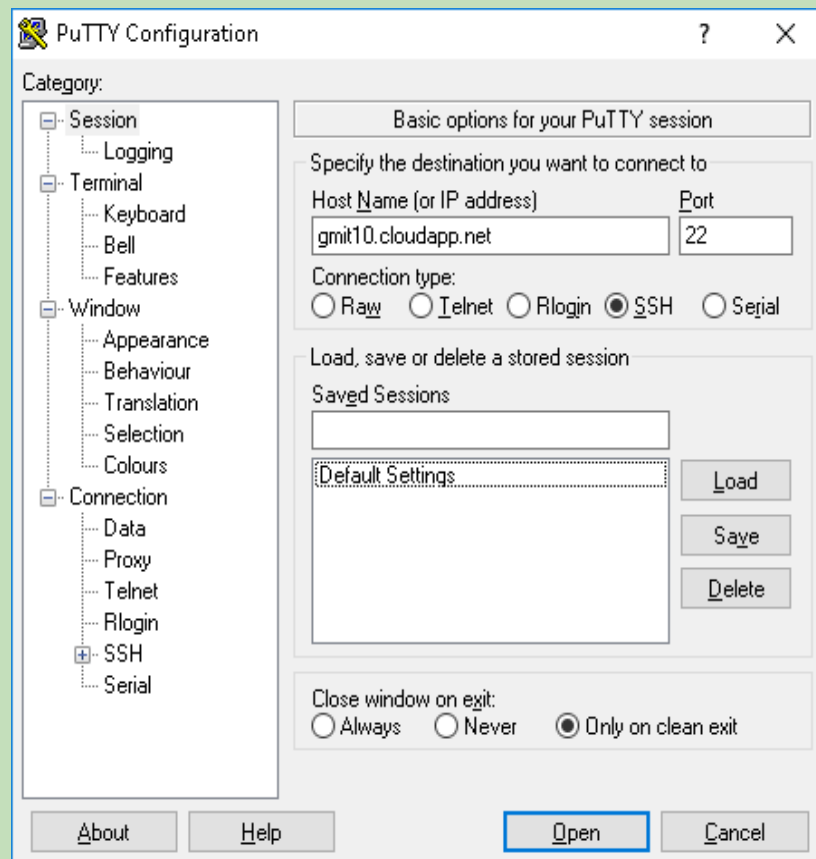
LAB EXERCISES APRIL 12TH 2016

UNIX/LINUX

Objective: Login to a Unix/Linux system via terminal such as putty

Install putty if required (Try ninite.com)

1: Enter credentials as required in putty (or alternative) and click open.



You may be prompted about a security alert. Click “yes” to continue. You will be prompted for your credentials. Note that both the username and password are case-sensitive. After providing the correct credentials, you should be logged on. The dollar sign prompt is Unix’s way of telling you that it’s ready for you to enter a command.

Who am I?

1. The easiest way to find out “who you are” is to enter the whoami command:

whoami

Try it on your system. The command lists the account name associated with the current login.

Unix is case sensitive, so the exit command is not the same as EXIT. If you enter a command all in uppercase, the system won’t find any such program or command and instead will respond with the complaint command not found. Get in the habit of using all lowercase for commands and Unix input. Lowercase is the natural Unix style.

Finding Out What Other Users Are Logged In to the System

Three commands are available to get you this information, and the one you choose depends on how much you'd like to learn about the other users: `users`, `who`, and `w`.

1. The simplest of the commands is the `users` command, which lists the account names of all people using the system:

users

Try this on your computer and see if other users are logged in to your computer system.

2. A command that you've encountered earlier can be used to find out who is logged on to the system, what line they're on, and how long they've been logged in. That command is `who`:

who

Here, you can see that the `who` command provides additional information over the `users` command.

3. The "`w`" command will display additional information.

w

This is a much more complex command than `users` or `who`, and it offers more information. Notice that the output is broken into different areas. The first line summarizes the status of the system and, somewhat cryptically, the number of programs that the computer is running at one time. The output indicates the username, when the user logged in to the system, how long it's been since the user has done anything (in minutes and seconds), the combined CPU time of all jobs the user has run, and the amount of CPU time taken by the current job. The last field tells you what the users doing.

Try the `date` command - this should display both the date and time.

date

absolute filename: Any filename that begins with a leading slash (/); it uniquely describes a single file in the file system.

binary or executable: A binary is a file format that is intended for the computer to work with directly rather than for humans to peruse. (aka executable)

device driver: All peripherals attached to the computer are called devices in Unix, and each has a control program always associated with it, called a device driver. Examples are the device drivers for the display, keyboard, mouse, and all hard disks.

directory: This type of Unix file is used to group other files, equivalent to a folder. Files and directories can be placed inside other directories, to build a hierarchical system.

directory separator character: On a hierarchical file system, there must be some way to specify which parts of a full filename are directories and which part is the actual filename. In Unix, the directory separator character is the slash (/), so a filename like `/tmp/testme` is easily interpreted as a file called `testme` in a directory called `tmp`.

dot: This is shorthand notation for the current directory.

dot-dot: This is shorthand notation for the directory one level higher up in the hierarchical file system from the current location.

dot file (hidden file): A dot file is a configuration file used by one or more programs. These files are called dot files because the first letter of the filename is a dot, as in `.profile` or `.login`. By default, the `ls` command doesn't list dot files, making them also hidden files in Unix.

home directory This is your private directory and is also where you start out when you log in to the system.

kernel: The kernel is the underlying core of the Unix operating system itself. This is akin to the concrete foundation under a modern skyscraper.

preference file: These are what dot files (hidden files) really are: They contain your individual preferences for many of the Unix commands you use.

relative filename: Any filename that does not begin with a slash (/) is a relative filename, a filename whose exact meaning depends on where you are in the file system. For example, the file `test` might exist in both your home directory and the root directory: `/test` is an absolute filename

and leaves no question about which version is being used, but `test` could refer to either copy, depending on your current directory.

root directory: The directory at the very top of the file system hierarchy is known as the root, or slash, directory.

search path: A search path is a list of directories used to find a command. When a user enters the command `ls`, the shell looks in each directory in the search path to find a file `ls`, either until it is found or until the list is exhausted.

slash: The directory at the very top of the file system hierarchy is known as the root, or slash, directory.

symbolic link: A file that contains a pointer to another file rather than contents of its own. This can also be a directory that points to another directory rather than having files of its own. It is a useful way to have multiple names for a single program or to allow multiple people to share a single copy of a file.

user environment: This is a set of values that describe the user's current location and modify the behavior of commands.

working directory: This is the directory where the user is working.

The Unix Hierarchical File System

A hierarchy is a system organized by graded categorisation. A familiar example is the organisational structure of a company, where workers report to supervisors and supervisors report to middle managers, who, in turn, report to senior managers, and senior managers report to vice-presidents, who report to the president of the company. That's exactly what a hierarchical file system is all about. You want to have your files located in the most appropriate place in the file system, whether at the very top, in a folder, or in a nested series of folders. With careful use, a hierarchical file system can contain thousands of files and still allow users to find any individual file quickly.

The top level of the Unix file structure (/) is known as the root directory or slash directory, and it always has a certain set of subdirectories, including bin, dev, etc, lib, mnt, tmp, and usr. There can be a lot more, however. You can obtain a listing of the files and directories in your own top-level directory by using the **ls -F /** command. Any filename that ends with a slash (/) is a folder (Unix calls these directories). Note: Any filename that ends with an asterisk (*) is a program. Anything ending with the at sign (@) is a symbolic link (a pointer to another file or directory elsewhere in the file system - similar to a shortcut in Windows), and everything else is a normal, plain file.

Directories

The bin Directory

The bin directory is where all the executable binaries were kept in early Unix. Over time, as more and more executables were added to Unix, having all the executables in one place proved unmanageable, and the bin directory split into multiple parts with different purposes (/bin, /sbin, /usr/bin).

The dev Directory

Device drivers are stored in the standard Unix dev (devices) directory.

The etc Directory

Unix administration can be quite complex, involving management of user accounts, the file system, security, device drivers, hardware configurations, and more. To help, Unix designates the etc directory as the storage place for all administrative files and information.

The lib Directory

This is a central storage place for function and procedural libraries. If programs want to include certain features, they can reference the shared copy in the Unix library (lib) rather than having a duplicate copy.

The tmp Directory

The tmp directory (pronounced "temp") is used by many programs as a temporary file-storage space. If you're editing a file, for example, the editor makes a copy of the file, saves it in tmp, and you work directly with that, saving the new file back to your original only when you've completed your work. The tmp directory can end up littered with various files and executables left by programs that don't remove their own temporary files.

The usr Directory

The usr (pronounced "user") directory was intended to be the central storage place for all user-related commands and data—the stuff directly relevant to users, as opposed to the system. After decades of evolution, however, /usr has come to encompass everything from libraries to system-wide applications, just because its boundaries have always been ill defined.

Find Where You Are with pwd

In Unix, the command pwd tells you the present working directory.

pwd

Pwd (print working directory) lets you know where you are. It also tells you the names of all directories above you because it always lists your current location as an absolute directory name.

List

TO list the contents of the directory you are currently in, try the following command:

ls

Moving to Another Location with cd

Try moving to the very top level of the file system and entering pwd to see whether the computer shows that you've moved:

cd /

pwd

ls

Notice that cd doesn't produce any output. Many Unix commands operate silently like this, unless an error is encountered. The system then indicates the problem.

Enter cd without specifying a directory.

cd

pwd

Without any directory specified, cd moves you back to your home directory automatically. If you get lost, it's a fast shorthand way to move to your home directory.

Move up one level by using cd .. and check the results with pwd:

cd

pwd

cd ..

pwd

Use the ls -C -F command to list all the directories contained at this point in the file system:

ls -C -F

Try using a combination of cd and pwd to move about your file system. Remember that without any arguments, **cd** always brings you back to your home directory.

Experimenting

Enter these commands at the Unix terminal, and try to interpret the output. Ask questions and don't be afraid to experiment (as a normal user you cannot do much harm):

echo hello world

hostname

arch

uname -a

clear

uptime

last

top

(you may need to press q to quit)

cal 2000

cal 9 1752

(do you notice anything unusual?)

yes please

(you may need to press Ctrl-c to quit)

time sleep 5

history

mkdir MyFolder

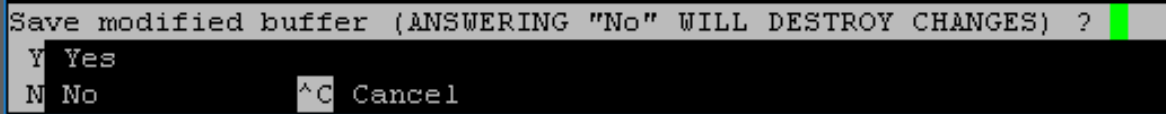
(Followed by ls command)

Creating a File

Nano is a text editor for Unix-like operating environments using a command line interface. To create a file using nano, try the following command:

nano file1.txt

This will create a file called file1.txt and open it for editing. Add some text. To close the file, press “ctrl” and “x” together. You should see a prompt at the bottom of the screen:

A screenshot of the nano text editor's save prompt. The prompt is displayed on a black background with white text. It reads: "Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?". Below the prompt, there are three options: "Y Yes", "N No", and "^C Cancel". A green cursor is visible at the end of the prompt line.

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No      ^C Cancel
```

Press “y”, and then “enter” to confirm the filename. Once saved, try the ls command to confirm that the file exists.

To edit a file through nano, type nano followed by the filename. For example, to edit a file called file2.doc you would type the following command:

nano file2.doc

Experimenting some more

Enter these commands at the Unix terminal:

uptime >myfile
(Followed by nano myfile)

ifconfig

ping google.com

lynx google.com