

Quick Unix Tutorial

CSE8A - Fall 2006

Running a Program in the Bash Shell

- When you open a new Unix terminal you are using a special program called a *shell*.
- A shell is a program that can help you start other programs. The name of the shell we'll use is 'Bash.'
- The shell provides you with a *prompt*. At the prompt, you can enter in the name of the program you'd like to run. This type of interface is called a *command-line interface*.

The prompt on your system is '-bash-2.05b\$ ', which we abbreviate as '\$ '. To run a program, type in the name of the program at the prompt and hit enter. For example, 'date' is the name of a program that prints out the current time and date:

```
$ date
Thu Sep 21 15:16:40 PST 2006
```

- Sometimes programs in Unix are referred to as 'commands.'

Making and Listing Directories

- Within your Unix account is your *home directory*. Your home directory is like the 'My Documents' folder in Windows. It's your space to put in your own files and other folders.
- The words 'folder' and 'directory' are synonymous. A directory is simply something that has a name and can store files and other directories in it (and those other directories can themselves store other files and directories and so on...).
- Your shell keeps track for you what your *current directory* is. The current directory is the folder you are currently 'in'. The shell tells all of the programs you execute what the current directory is, so that they can know what it is. (See page 2 for the section on how to change the current directory.)
- When you are first logged in, the current directory is your home directory.

To find out what directory you are in, use the 'pwd' command, which stands for 'print working directory':

```
$ pwd
/home/linux/ieng6/cs8f/cs8fzz
```

The '/' slashes are used to separate directory names. This means that your home directory is named 'cs8fzz'. That directory resides in the 'cs8f' directory. The 'cs8f' directory resides in 'ieng6', which resides in 'linux', which resides in 'home', which resides in '/'.

- The directory '/' is called the *root directory*. Every directory and file in Unix is directly or indirectly contained in the root directory.

Let's make a new directory, using the mkdir command:

```
$ mkdir psal
```

This made a new directory named 'psal' inside the current directory, but it didn't print anything out for us see.

To find out if it worked, we can ask for a listing of the current directory, which we can do with the 'ls' command (that's ls as in 'list'):

```
$ ls
psal
```

Command Line Arguments

That psal directory looks pretty lonely, but there are actually hidden files in your home directory too. You can tell ls to show the hidden files by giving it additional 'input'. Additional input that you specify on the command-line is called a *command-line argument*. To see all files, pass the '-a' (for 'all') *argument* to 'ls':

```
$ ls -a
.  .. .bash_history .bash_profile
.bashrc .cshrc .login .procmailrc psal
```

An argument is a piece of information given to a program. (The name 'psal' given to the mkdir command in the previous section is also an argument.)

The general format executing a program on the command line is:

\$ command-name the-arguments

Where '*command-name*' is the name of a command (such as 'date', 'pwd', 'mkdir', or 'ls') and '*the-arguments*' is a space separated list of zero or more arguments (such as 'psal' and '-a').

You can see a 'long' listing by passing the '-l' argument:

```
$ ls -l
total 4
drwxr-x--- 3 cs8fzz other 4096 Sep 21 21:19 psal
```

- The 'd' means that 'psal' is a directory. If there is no 'd' then that means it's a regular file.
- When an argument begins with a '-' hyphen (as in '-a' or '-l') it is sometimes called an *option*. An option is a special argument that changes the behavior of the program. In this example, the '-a' option lists all files, even hidden ones, and the '-l' option forces a more detailed output.

For ls, you can use both options at the same time:

```
$ ls -l -a
```

When options are only a single letter, you can use just a single hyphen and you don't need the extra space:

```
$ ls -la
```

This turns the previously two '-l' and '-a' arguments into a single '-la' argument. You can expect to see this shortened form quite often.

The shell always tells the program what its arguments are, in addition to telling it what the current directory is.

- When an argument does not begin with a '-' it is typically a file name or a directory name. However, it is up to the program to decide what the arguments given to it means.
- When an argument is given to a program, it is said that the argument is 'passed' to the program.

Changing the Current Directory

- You can tell the shell to change the current directory by using the 'cd' command, for 'change directory'.
- You pass the cd command the name of a new directory to use as the current directory:

```
$ cd psa1
```

Now if you do a 'pwd' you'll see the directory has changed:

```
$ pwd
/home/linux/ieng6/cs8f/cs8fzz/psa1
```

To get back to the directory we were in before (the cs8fzz directory) we can give its full name to the 'cd' command:

```
$ cd /home/linux/ieng6/cs8f/cs8fzz
```

- The 'full name' of a directory or file is called its *path*.

There is also a short cut. To switch 'up' a directory level, you can use the special name '..'. The name '..' is called the *parent directory*.

```
$ cd ..
```

Creating New Files and Deleting Files

You can start a text editor by naming the text editor and the file you'd like to edit. If the file already exists, the text editor will open it for you. If the file does not exist, the text editor will start a new file for you.

```
$ cd psa1
$ emacs syllabusread.txt
```

If you save in the text editor that file will be created. To delete the file, use the rm command (for 'remove'):

```
$ rm syllabusread.txt
```

To remove a directory, you need to pass in the '-r' option:

```
$ rm -r psaone
```

You should only remove a directory or file if you are sure it doesn't have anything you'd want! E.g., if you just created a directory but made a typo. So, no deleting your homework.

Special Shell Characters

To switch to your home directory at any time, you can also use another special form:

```
$ cd ~
```

- The '~' tilde character is a special character for the shell: it's the shorthand notation for your home directory. Instead of the shell passing in '~' to programs (when it appears as the argument) the shell actually passes in the full path to your home directory.

To check what the shell is actually passing to programs, use the echo command:

```
$ echo what is ~ this
what is /home/linux/ieng6/cs8f/cs8fzz this
```

There are other special shell characters:

- Use '?' as a wildcard character. For example, 'psa?' matches all files or directories that begin with the letters 'psa' and have a fourth character. The shell will replace the single argument 'psa?' with a series of all matching files/directories.

For example, if there were four directories named psa1, psa2, psa3, and psa4, then 'psa?' would be expanded to those four entries:

```
$ echo psa?
psa1 psa2 psa3 psa4
```

- Use '*' as another wildcard character. The '*' is like the '?' but it can match zero to more characters instead of just one.

Typing in

```
$ echo *.txt
```

displays the names of all files that end with the suffix '.txt'.

By default, the 'ls' command lists the current directory. If you provide it with other file or directory names it will list those instead.

- `ls *.txt` Lists all text files in the current directory
- `ls ..` Lists the parent's contents

Also Helpful

- `javac *.java` Compiles all Java files that are in the current directory
- Don't type in long file- or directory-names! Use the 'tab' character and Bash will type in the rest of the name for you. (This is an excellent technique to also prevent misspelling the names of existing files!)
- If you don't want the shell to expand a special character, put them inside of 'quotes' or "double quotes". E.g.: `echo '~'`
- Beware of using wildcards with the rm command. A simple `rm *` will delete all of the files in the current directory!
- To view (but not edit) a file, use the 'cat' or 'more' commands. E.g. `more syllabusread.txt`