CS 352 (Fall 16): System Programming and UNIX

Project #1

Basic UNIX Commands

due at 9pm, Wed 31 Aug 2016

1 Overview

This project is a quick overview of the UNIX tools that you'll need to write your first C program. We'll ask a series of questions; you'll write a text file, and turn it in using `turnin`.

2 Format

You will turn in a text file, containing your answers to the questions below. If you prefer to work on your own computer, you may do so (and then upload the file to Lectura); however, we'd recommend that you write the file on Lectura directly. After all, things will go a lot faster if you don't have to upload your source code, over and over.

Make sure that the file you are creating is a text file. On Windows, we recommend Notepad++ (`https://notpad-plus-plus.org`) - an open source editor. (The standard Notepad **almost** works, but has some quirks that you'll understand better later in the semeter.) Mac, no doubt, has a similar application - although I don't use Mac, so I can't tell you its name.

Be careful using word processors. Most word processors produce binary files (not text files).

3 The Questions

For the following questions, give the UNIX command (with any required options and/or arguments). Some may require that you pipe two commands together.

While every one of the commands you need has been mentioned in the slide deck, some will require options which we have not discussed. Consult the man page for the commands to find the new options that you need.

NOTE: Every answer you submit should be a **single command**, which might include pipes, but which must not include multiple steps. So the following answers are OK:

- `cmd1 | cmd2`
- `cd foo/bar`

This is not:

- ```
  cd foo
  cd bar
  ```
  ```

1. List all of the files in `/usr/bin`, except for the hidden files; print out the "long format" so that we can see the permissions, usernames, etc.

2. Get documentation on the command `xargs`.

3. Make a subdirectory named `myDir` inside the current directory.

4. Leave the current directory; move out to the directory which contains it.

5. Suppose that your current directory is `/a/b/c`. Move to the directory `/a/b/x/y` using an **absolute path.**

6. Suppose that your current directory is `/a/b/c`. Move to the directory `/a/b/x/y` using a **relative path.**

7. Copy the file `orig`; name the duplicate `copy`.

8. Copy the **directory** `dir1`; name the duplicate `dir2`. (This is mentioned in passing in the slides; if you don't recall it, check the man page.)

9. List all of the files in the current directory which have names beginning with `b`. Just list the names, don't give any details.

10. List all of the files in the current directory (except hidden files) which have the extension `.txt` (all lowercase). Give the "long format" output.

11. As was shown in class, the `ls` command lists the contents of a directory to `stdout`. Use file redirection to save the list of file names in the current directory to a file called `files`.

12. Compare the file `expected` to `actual`, and show the differences (if any).

13. Print out the number of lines in the file `asdf`.

    Hint: A single command is used to count words, characters, and/or lines. Options control exactly what it prints out.

14. Search through the file `records`, and print out any lines in the file which contain the word `HELLO` (in all caps).

    HINT: There are two ways to do this. One is to use a certain command, and pass the file name as a parameter. The other is to have two commands, connected with a pipe: the first prints out the file, and the second searches through the stream of data. **You may use either version.**

15. Write a command which uses a pipe. The first command should search through the file `records`, and output any lines that contain the word `HELLO` (in all caps). The second should count the number of those lines.

16. Read the man page for the `cat` command. Show how to print out the contents of **two** files with a single command; print out `foo`, then `bar`.

17. Change the name of the file `old` to `new`; do not make a copy.

18. Move the file `misplaced` (which is in the current directory) into the sub-directory `putHere` of the current directory.

19. Remove the file `badFile` from the current directory.

4 Turnin

When you have filled out your text file, you are ready to turn it in. However, take note: there is one detail which may be new to you.

In this class, every assignment requires that you turn in your files inside a specially-named directory.

We do this because, as the programs get more complex, we will be turning in several files at once. Directories make things more sane! **If you do not name the directory properly, or if you do not name the files *inside* it properly, then you will lose points, as we will have to manually grade your code - instead of using our testing scripts.**

Turn in the following directory structure using the assignment name `cs352_f16_proj01`:

```
proj01/
    commands.txt
```

Turn in your directory using the assignment name `cs352_f16_proj01`. Make sure that you turn in the directory - **NOT** the file inside it!

5 If You Haven't Used `turnin` Before

If you've never used `turnin` before, don't worry, it's pretty easy (in 127A, the students learn this during the first Section meeting). I'll break it down into several steps.

Step 1: Connecting to Lectura

We connect to Lectura through a protocol known as `ssh` (Secure Shell). `ssh` allows us to connect to Lectura from any computer, anywhere in the world, and type commands on it as if we were sitting in front of it, at the keyboard.

The best way to use `ssh` is on the command line. If you are using a Mac or Linux box (at home, or in the labs), open up the Terminal, and type the following command:

```
ssh username@lectura.cs.arizona.edu
```

(Replace "username" with your CS username, which is the same as your Net ID.)

However, if you are using Windows, then you don't have `ssh` on the command line, and you'll have to install a program called "Putty" to do this. The University provides information about how to install that here:

`http://softwarelicense.arizona.edu/ssh-clients-windows-and-mac`

Step 2: Creating a Directory

When you log on to Lectura through `ssh`, it will automatically take you to your home directory. You need to create a directory to hold the file. For simplicity, let's create it right in your home directory for now; however, in the long term, it's wise to organize your data better. In the future, create a directory for all of your 352 work - and then put your directories inside of that!

To create a directory, use the `mkdir` command. Simply type:

```
mkdir directoryName
```

Replace `directoryName` with the name of the directory you want. It can include uppercase and lowercase letters, numbers, underscores, and a few other characters, but it **MUST NOT** include spaces!

Once you have created a directory, you can move into it with `cd` command, like this:

```
cd directoryName
```

Directory and file names in UNIX are CaSe SeNsItIvE, so make sure that you type the directory exactly like you did when you ran `mkdir`! Also, note that you can get "out" of any directory (that is, just move out one step) using the ".." directory, like this:

```
cd ..
```

Step 3: Creating a File

The **best** way to create a file on Lectura is to write it there! However, there are other options. If you would like to edit your file right on Lectura, try one of these programs:

- `vi` is an old classic; it's what Russ and Eric use. It's optimized for running over `ssh`, but many people think that it is hard to use. Google for a tutorial if you want to try it.
- `emacs` is a popular alternative. It also works well over `ssh`, and some people think it's easier to use. Again, Google for a tutorial if you want.
- `pico` is a good starting point for novices. To use it, simply type

```
pico filename
```

and you'll enter the editor.

No need for a tutorial here - the commands are all listed along the bottom of the screen. Just remember that something like `^O` ("caret-oh") means "type Control-o".

However, if you don't want to edit your files on Lectura, you may write them on your own machine, and upload them. To upload a file, you use `scp` (Secure Copy). Again, on Mac or Linux, you can do this on the command line:

```
scp filename username@lectura.cs.arizona.edu:~/directoryName/
```

This copies the file `filename` (on your computer) to the directory `directoryName`, inside your home directory on lectura. (Of course, also replace `username` with your CS username.)

But if you hate to use the command line (shame on you!), there are graphical programs you can use: WinSCP for Windows, and Fugu for Mac. You can download them from the same place where you can get Putty:

```
http://softwarelicense.arizona.edu/ssh-clients-windows-and-mac
```

Step 4: Use `turnin`

`turnin` is a utility (which only works on Lectura), which you can use to turn in your homework. In order to turn something in, you must know what "assignment" you are using; you can then turn in some files - or, in this class, an entire directory.

To turn in something, first use `cd` to navigate to where the item is. Do **NOT** go into the directory you created - instead, go to the place which **contains** that directory. If you created the directory inside your home directory, then you don't have to go anywhere - you'll be inside your home directory as soon as you log in with `ssh`.

Next, you use the `turnin` command to turn in the files:

```
turnin assignmentName directoryName
```

In this case, the command you type should be **exactly**:

```
turnin cs352_f16_proj01 proj01
```

If it works correctly, then you should get output which looks like this:

```
Turning in:
 proj01/test.txt -- ok
All done.
```

After you have turned in your files, use `turnin -ls` to confirm that your files were uploaded. **Do this double-check! You'll regret it if you make a mistake and don't catch it!**

```
turnin -ls cs352_f16_proj01
```

You should expect something which looks a lot like this:

```
.:
total 4
drwxrwx--- 2 russelll cs252f16 4096 Aug  4 14:57 proj01

./proj01:
total 4
-rw-rw---- 1 russelll cs252f16 29 Aug  4 14:57 test.txt
```

Possible Error Messages

Here is an error message. If you get this, then either you mis-typed the assignment name, or you are trying to turn it in after the due date:

```
"cs252_f16_proj01": Unknown assignment.
```

Here is another error message. If you get this, then you may have accidentally added a trailing forward-slash after the name of your directory:

```
Turning in:
File(s) to be turned in must be in current directory.
All files were not turned in successfully.
```