CS 279 - Week 6 Lab Exercise

Deadline

Due by the end of lab on 2022-09-29.

How to submit

Submit the files specified below on https://canvas.humboldt.edu.

Purpose

Seeing the difference it can make to `export` a local variable, playing around a bit with processes and job control, and trying out `cat` with multiple files.

Important notes

- Remember: On the public course web site, at the end of the **References** section, there is a handout about how to use **`ssh`** to connect to `nrs-projects.humboldt.edu` and how to use **`sftp`** to transfer files to and from `nrs-projects.humboldt.edu`.
- Work in PAIRS for this lab exercise:
  - two people at one computer,
  - one typing (driver),
  - one saying what to type (navigator),
  - both discussing along the way!

  When done, the driver should e-mail the files to the navigator, so BOTH of you can EACH submit them.

Lab Exercise setup

- use `ssh` to connect to the one of your accounts on `nrs-projects.humboldt.edu`
- make and protect a directory `279lab6` using the commands:

  ```
  mkdir 279lab6
  chmod 700 279lab6
  ```
- go into that directory using:

  ```
  cd 279lab6
  ```

Problem 1

Along with this lab exercise handout, you should find a file `279lab06-prob1.txt`. Copy its contents into a file with this same name in your `279lab6` directory.

It contains a number of questions. Type your names and your answers to those questions within this file and create the additional files described, and submit your resulting files:

(continued on next page!)

- `279lab06-prob1.txt`
- `lab6-1` or `lab6-1.sh`
- `demo-lab6-1.txt`

Problem 2

You will now try out some commands involving processes and job control.

Do the following from your current `bash` session on nrs-projects.

- The `ps` (process status) command has MANY options! But, when called with NO arguments, it displays a list of running processes in the currently-running shell.
  - Type:
    ```
    ps
    ```
    ...and see the list of running processes in your current `bash` shell on nrs-projects.
  - Then type:
    ```
    echo "problem 2 - 1" > prob2-out.txt
    ps >> prob2-out.txt
    ```
- We mentioned in class that you can start a process in the background by ending its command with an `&`.

  That is,
  ```
  nano play.txt &
  ```
  ...will start `nano`, except as a background job.
  - Type the above command, starting `nano` editing a file `play.txt` as a background job.
- FUN FACT #1: Many (not all) programs currently running in the foreground can be suspended and put into the background by typing the control key and the letter z at the same time (in Linux style, this is often written as ^Z). (`nano`, sadly, is one of the exceptions to this!)
  - But, you can do this in the `man` command!
  - Start the command `man ps`, but then type ^Z to put it into the background before you have gone through all of its pages.
- FUN FACT #2: Another example command that you CAN put into the background by using ^Z is the `top` command, which displays (from its `man` page): "...a list of processes or threads currently being managed by the Linux kernel."
  - Start the `top` command, and then type ^Z to put it into the background.
- Now that you have some jobs in the background, the `jobs` command becomes interesting!
  - Type:
    ```
    jobs
    ```
    ......and see the list of current jobs in your current `bash` shell on nrs-projects.
  - Compare that to what you see when you now type:
    ```
    ps
    ```

- Then type:

  ```
  echo "problem 2 - 2" >> prob2-out.txt
  jobs >> prob2-out.txt
  ps >> prob2-out.txt
  ```

- As we mentioned, there are quite a few ways to request that a background job be brought to the foreground. Two of the many possibilities:
  - `fg` with no arguments brings the most recent stopped job into the foreground, and
  - `fg` with an argument of `%` and the job number -- or even just `%` and the job number! -- brings the job with that job number into the foreground.

  Bring your background `nano` job (it probably has a job number of 1) to the foreground. Type your names into `play.txt`, then save and exit `nano`.
  - Try `ps` and `jobs` now -- the `nano` process and job is no longer listed.
  - Then type:

  ```
  echo "problem 2 - 3" >> prob2-out.txt
  jobs >> prob2-out.txt
  ps >> prob2-out.txt
  ```

- Bring your background `man` job (it probably has a job number of 2) to the foreground. Skim as much of it as you would like, noting that you can type `q` to quit before reading or skimming all of its entry.
  - Try `ps` and `jobs` now -- the `man` process and job is no longer listed.
  - Then type:

  ```
  echo "problem 2 - 4" >> prob2-out.txt
  jobs >> prob2-out.txt
  ps >> prob2-out.txt
  ```

- Bring your background `top` job (it probably has a job number of 3) to the foreground. The traditional control combo for interrupting and ending a command in Linux is typing the control key and letter c at the same time (abbreviated as ^C) -- this works to end this `top` command, but so does typing `q`.
  - Use whichever of these you would like to now end the `top` command.
  - Then type:

  ```
  echo "problem 2 - 5" >> prob2-out.txt
  jobs >> prob2-out.txt
  ps >> prob2-out.txt
  ```

Submit your resulting files `play.txt` and `prob2-out.txt`.

Problem 3

You should know at this point that the `cat` command lets you see the contents of a file.

I am not sure if it has been pointed out that the command name `cat` is short for **concatenate**, and indeed when `cat` is called with *multiple* file-name arguments, their contents are all concatenated together and output to the screen.

(And if you redirect that output, the result is that all of the `cat` arguments' files are concatenated together in that redirected output file!)

BUT -- you cannot redirect output to a file you are redirecting from. That is, this does not work and has unfortunate results:

```
$ cat looky.txt looky.txt > looky.txt
cat: looky.txt: input file is output file
cat: looky.txt: input file is output file
```

...and also, `looky.txt`'s contents have been deleted, even though the output redirect failed!!

You CAN, however, redirect to a different file, and then move or copy the result back to the desired file:

```
$ cat looky.txt looky.txt > temp
$ mv temp looky.txt
```

...and now `looky.txt` contains two copies of its previous contents.

So -- try this out!

On `nrs-projects`, copy the following files from my home directory to your current working directory:

```
cp ~st10/lab6*.txt .      # note that dot at the end, nickname for your
cp ~st10/letter*   .      #     current working directory!
```

Oh no! It has been declared that all C++ files related to some project must start with a particular header and footer comment!

File `lab6-prelude.txt` contains the required header comment, and `lab6-postscript.txt` contains the required footer comment.

Write a script `label-proj` or `label-proj.sh` that has the expected first line and comments including at least the name of this script, both of your names, and today's date, and also meets the following requirements:

- For each file in the current working directory whose name ends in `.cpp` or `.h`, your script should:
  - create a temporary file containing the result of concatenating the contents of `lab6-prelude.txt`, that file's current contents, and `lab6-postscript.txt`
  - then write a `mv` command so that the temporary file is changed to have the name of that file

When you run your script, if it has worked, the three copied files `letter_match.h`, `letter_match.cpp`, and `letter_match_test.cpp` should now all have the desired header and footer comment added to them. (And if the first version(s) of your script do not work, you can copy over new versions of files `letter*` from `~st10` as you did for your original versions.)

Submit:

- your script `label-proj` or `label-proj.h`
- the resulting versions of your copies of `letter_match.h`, `letter_match.cpp`, and `letter_match_test.cpp`

BOTH of you should then submit copies of these problems' files to Canvas for this lab exercise.

Once both of you have submitted these lab exercise files, you may leave lab if you wish. Or, you can ask questions, read the course text, etc. But note that questions about today's lab exercise will get first priority.