The University of British Columbia

Physics 210 Assignment # 3:
SHELL SCRIPTS & PHP

Tue. 21 Sep. 2010 — finish by Tue. 28 Sep.

In Assignment 2 you created a directory `/home2/phys210/<you>/a02` in which to store the files you submitted for the second assignment.[1] You then used "`chmod -R o-r`" to make it and all its contents (including any possible subdirectories and their contents) inaccessible to `o` (for others). The second step will no longer be necessary, as we have set up a utility which periodically restores the correct ownership, group and permissions for the `/home2/phys210/` directory tree.

This week, create `/home2/phys210/<you>/a03` for your submissions on this assignment; and so on for the rest of the course. We won't remind you any more.

In the previous Assignment you learned how to customize your `bash` environment using the `bash` *resource* file `.bashrc` in your `$HOME` directory. By now you should have edited the template `.bashrc` file and the `.aliases` file it `source`s to suit your taste, including lots of personalized, idiosyncratic `alias`es that you find easier to remember (or more æsthetically pleasing) than the "raw" `bash` commands — which, while æsthetically offensive, are at least reasonably universal and thus worth remembering. This may impede your familiarization with `bash`, but for now, go ahead and indulge yourself with `alias`.

The goal of this Assignment is to make you reasonably familiar and comfortable with **shell scripts** and a few other important tools.

1. **GET STUCK IN `tar`**: In your `/home2/phys210/<you>/a03/` directory, write a short `bash` script called "`bck.sh`" that uses `tar`[2] to make a compressed "backup" of your entire `/home2/phys210/<you>` directory tree in a single file `~/HW.tar.gz` (note that this file should be in your `$HOME` directory). Make a symbolic link to `~/HW.tar.gz` in your `/home2/phys210/<you>/a03/` directory. Here's what `bck.sh` should do: First, if there is *already* a `~/HW.tar.gz` file, `gunzip` (decompress) it to `~/HW.tar` — then use `tar` to *append* to `~/HW.tar` any files in `~/HW/` (and its subdirectories) that have *changed* since the *last* time `bck.sh` was invoked — then `gzip ~/HW.tar` back to `~/HW.tar.gz` and you're done. Use "`man tar`" to find out how to do the middle part.[3] This is a valuable thing to know how to do.[4]

2. **PLAY WITH *ImageMagick***: Everyone likes graphics, so here's a treat for you: in your Terminal window, type "`man ImageMagick`" and learn about some of the things this free graphical manipulation library provides. You should also use your Web browser to consult the documentation in our **Manuals** link. Then, in your browser, right-click on the PHYS 210 logo at the top of our class homepage and use the "`Save Image As...`" menu item to save the file `P210logo.gif` — pay attention to *where* it gets saved, and then *move it* to your `/home2/phys210/<you>/a03/` directory. In that directory, use the `convert` command supplied by *ImageMagick* to make files `P210logo.jpg`, `P210logo.png`, `P210logo.tif` and `P210logo.pdf` (plus any other graphics formats you are especially fond of). *Which file is the smallest?* Is it the one you expected? Record (in `/home2/phys210/<you>/a03/readme.txt`) your answer and any comments. If you want to make your own graphics (maybe a personal logo?), try "`gimp`". But don't get too distracted by it!

---

[1] Any time I put something inside angle brackets, like "`<you>`", it stands for some (hopefully self-evident) string that will be different for each person. In this case "`<you>`" refers to your login ID on `hyper`, *without the angle brackets*. In my case, for instance, it would stand for "`jess`".

[2] Historically, `tar` is so named from "`t`ape `ar`chiver", although actual tape is rarely used any more.

[3] *Hint:* `tar -tzvf $HOME/HW.tar.gz` will show you the contents of your compressed backup file.

[4] A shell script is actually unnecessary for this task; the whole operation can be expressed in one line, using the "`;`" terminator to break up the line into several commands — which means you really should just create an `alias` in your `.bashrc` file to do it. But for this Assignment please put it in a `bck.sh` script.

3. WHAT'S IN THE SCRIPT? Copy the shell script `~phys210/bin/fib.sh` to your own `/home2/phys210/<you>/a03/` directory and (using your favourite editor) *add comment lines* (any line starting with "#") *to explain what is happening at each step.* Be ridiculously thorough; it will pay off in the end. Make sure that your copy of the file is executable ("`chmod +x fib.sh`") and check that it still works like the original despite all the comments you've added.[5]

4. BE THE BARD![6] WRITE A `fact`.[7] Make your own `bash` script to generate a sequence of **factorials**: $F_n = n! \equiv n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 3 \times 2 \times 1$ with the exception of $F_0 = 0! = 1$. Thus the first six factorials (F_0 through F_5) are $1, 1, 2, 6, 24, 120$ and so on. Of course, they go on forever, but they get very big very fast, so you don't want to try to generate *all* the factorials. Better if you specify the one to start on and how many more to print out.

So your script, which you should call `fact.sh` and which should be in your `/home2/phys210/<you>/a03/` directory, should look for two arguments, $F_{\rm start}$ and N, so that (for example) if you invoke it with "`fact.sh 2 3`" it should print out on your screen "`2 6 24 120`" (not necessarily all on the same line). This should be simple, eh?[8] To keep it that way, let's forbid using either of the first two factorials as $F_{\rm start}$. Thus if you enter `1` as your first argument you should get no response.

If you enter the wrong number of arguments, the script should print out a "`USAGE: ...` " message analogous to the one in `fib.sh` explaining how it is supposed to be used.

Oh, and one final feature: if you enter a first argument that is *not a factorial*, your script should print out a polite but perhaps slightly sarcastic message informing the user that they have entered an invalid starting point.[9]

You may want to start with a simpler version that just prints out the first 10 or so factorials, to make sure you have that part right; then add the bells and whistles involving arguments.

5. PUT IT ONLINE: In my humble opinion[10] shell scripts are not very elegant. There are much cleaner ways to do the same things, and then some! For instance, your personal homepage that you created last week need not be just a passive `HTML` file; it can actually *do* stuff, and show off the results to visitors, using scripting languages like PHP or *JavaScript*. I prefer PHP. The PHP script in `~phys210/public_html/fib.php` does the same thing as the `bash` shell script `fib.sh`, but it does it *online* for *anyone*. Try it out at `http://www.physics.ubc.ca/~phys210/fib.php` and then *convert it to make factorials* in a file called `fact.php` in your own `~/public_html/p210/` directory.[11]

---

[5] To run a program (or script) that is in your `PATH` (check this with "`echo $PATH`") you just type the name of the file. To run `fib.sh` in your current directory ("`.`"), assuming it is actually *in* that directory, use "`./fib.sh`" instead.

[6] Bard, playwright, writes scripts, get it? Oh, never mind.

[7] For `fact`orial, get it? Oh, never mind.

[8] Simple, until you have to *do* it! Note that the first two factorials (F_0 and F_1) are identical.

[9] *Hint:* No one said this should be *efficient*. If you just generate all the F_n from scratch every time, and stop when you reach a value greater than or equal to the first argument, you will have no troubles. But don't start printing until you get to the requested starting point.

[10] In Geek, "IMHO". For a list of others, Google "`Geek acronymns`".

[11] Note: in PHP, a double slash ("`//`") anywhere in a line makes everything to the right of the `//` a *comment* — *i.e.* ignored by the PHP processor.