

CS 279 - Homework 12

Deadline:

Due by 11:59 pm on **FRIDAY**, December 7.

How to submit:

Submit your files using `~st10/279submit` on nrs-labs, with a homework number of 12, by the deadline shown above.

Purpose

To practice more with the bash shell `case` statement, `RCS`, `crontab`, `uptime`, `top`, `at`, `batch`, and `IFS` with `read`.

Important notes:

- **Each** bash shell script that you write is expected to include a descriptive opening comment block including your name and the last modified date.
 - each bash shell script `FUNCTION` you write should also have an descriptive opening comment block including at least descriptions of what it expects and what it produces and/or its side-effects when run
- It is possible that some of your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1

Write a bash shell script `get-details.sh` that appropriately uses a `case` statement such that, for each file in the current directory, it **FIRST** outputs its name on one line, and **THEN**, starting on the next line, follows that with additional output based on the file's name as described below.

Note that you are permitted to add additional spacing and "underlines" or borders as you wish.

- if the file's name ends in `.txt`, it then outputs the result of the `wc` command for that file.
- if the file's name ends in `.sh`, it then outputs each line in that file (if any) that happens to contain a command-line argument
- if the file's name ends in `.gz`, it then outputs the results of calling `gzip -l` on that file (this gives a lovely summary of its compression)
- if the file's name ends in `.v`, it then outputs the result of calling `rlog` on that file but piping its result to `grep` such that you only get lines containing the word `revision`

- if the file's name ends in `.tar`, it then outputs the result of calling `tar tf` on that file, so that the archive's contents are displayed.
- otherwise, it then outputs the long listing for that file.

Don't submit your resulting `get-details.sh` yet...

Problem 2

Consider your shell script `get-details.sh` from Problem 1.

FIRST: in a file `hw12-2.txt`, put your name, and then the command requested for each of the following. But you'll also be doing some additional actions, as noted.

2 part a

In `hw12-2.txt`, write an RCS command whose effect will be to create an RCS revision group file, `get-details.sh,v`, for `get-details.sh` that includes its current version as the initial version.

Then, actually do this command.

2 part b

In `hw12-2.txt`, write an RCS command whose effect will be to check out, WITH the lock, the latest version of `get-details.sh` from its RCS revision group file `get-details.sh,v`.

Then, actually do this command.

2 part c

Assume that you have modified `get-details.sh`. In `hw12-2.txt`, write an RCS command to now check in this modified version as the latest version.

Now -- actually make the following revision to `get-details.sh`:

- if a file name with no suffix happens to be a directory, it then outputs a message including the number of files in that directory. (Otherwise, it then outputs the long listing for that file.)

Once you have added this and tested it to your satisfaction, actually do this part's RCS command.

2 part d

In `hw12-2.txt`, write an RCS command whose effect will be to check out a READ-ONLY version -- WITHOUT the lock -- of the latest version of `get-details.sh` from its RCS revision group file `get-details.sh,v`.

Actually do this command. Do `ls -l get-details.sh` -- see how it is, indeed, read-only? But once you've checked this out, go ahead and check out a copy with the lock, so you won't have a problem if you decide to make further changes to this script later.

2 part e

In `hw12-2.txt`, write an RCS command that will give you a listing of versions and log entries for `get-details.sh`.

Then, actually do this command, redirecting its output to `hw12-2e.txt`.

2 part f

In `hw12-2.txt`, write an RCS command that will output the differences between the original version of `get-details.sh` -- the version you checked in with your answer to Problem 2 part a -- and the latest version, the version you checked in with your answer to Problem 2 part c.

Then, actually do this command, redirecting its output to `hw12-2f.txt`.

Submit your resulting `get-details.sh`, `get-details.sh,v`, `hw12-2.txt`, `hw12-2e.txt`, and `hw12-2f.txt`.

Problem 3

In a file `hw12-3.txt`, put your name, and then your answers for each of the following.

3 part a

Consider this crontab entry:

```
1 2 3,5 6-8 4 /home/abc1/glarble.sh
```

What will this cause to happen, and when/how often?

3 part b

Consider this crontab entry:

```
30 21 * * 5 find /home/def2 -name "*~" -exec rm -f {} \;
```

What will this cause to happen, and when/how often?

3 part c

Write a crontab entry that will run a script `/home/ghi3/monthly.sh` at 12:05 am on the first day of each month.

3 part d

A system administrator with username `jdk14` wants to gather some system performance statistics snapshots.

Write a crontab entry that will append the result of the `uptime` command to file `/home/jdk14/fall-aft-stats.txt` every Monday and Wednesday from September to December at 5:00 pm each day

3 part e

Oh dear -- you discover, using the `uptime` command, that the system load is quite high. Give the command you could type to see, in real-time, which processes are using the most CPU until you type `q` or `control-d`. (Don't make this harder than it is -- I just want to get this command into this homework, even if trivially!)

3 part f

Here is an odd thing I have discovered -- if you would like the results of the `time` command to be appended to a file, not only does it turn out that its output goes to standard error, and so you have to redirect standard error, but you also have to enclose the command in parentheses before that redirecting for it to actually work!

That is,

```
time ls 2>> looky.txt      # does NOT redirect ls's execution times to looky.txt
(time ls) 2>> looky.txt    # DOES redirect ls's execution times to looky.txt
```

That said...

...assume that it is currently 4 pm on an unspecified date. User `mno5` wants to arrange to have a shell script, `/home/mno5/big-big-task.sh`, run at 2 am tomorrow morning, such that the amount of time it takes to run is also appended to a file `/home/mno5/task-stats.txt`. Write up a command that `mno5` could do now to set up this task to be done then.

3 part g

User `mno5` wants to have another shell script, `/home/mno5/low-priority.sh`, run with a priority 19 lower than the norm when the system chooses to run it. Write a command that `mno5` could do to make this happen.

3 part h

User `mno5` wants to see all of the jobs requested using the commands from parts f and g. Write a command that `mno5` could now type that will do this.

3 part i

In part h's result, user `mno5` sees a job 4 that he/she decides not to have run, after all. Write the command that `mno5` could use to remove this job (so it won't be done after all).

Submit your resulting `hw12-3.txt`.

Problem 4

While there are varying, and even formal, definitions for a csv (comma-separated values) file, let's keep it very simple for this problem, and say it is a text file in which the values on each line are separated by commas.

Also for our purposes in this problem, we'll assume each line in such a file contains a set number of values about some thing. For example, each line might contain a product name and its sales for the last 4 weeks, separated by commas; or, each line might contain the name, operating system, price, and date of release of a computer game, separated by commas; or, each line might contain an HSU username, web alias, and a file from that user's `public_html` directory on nrs-labs, separated by commas.

Decide on some kind of data you would like to store in such a csv file; you need to have at least 3 comma-separated values expected per line.

Now write a shell-script `csv-play.sh` that meets the following specifications:

- it should expect a csv file of the type you have decided you want to use as a command-line argument. It should complain and exit with a non-zero exit status if the file doesn't exist or is not readable; otherwise, you are allowed to assume it is a well-formed csv file of the type you expect.
- it should use `IFS` and `read` appropriately to read the contents of this file, and do something appropriate with it that results in output -- here are a few ideas:
 - for example, if each line contains a product name and its sales for the last 4 weeks, it might output its name and its average sales;
 - or if each line contains information about a computer game -- its name, operating system, price, and date of release -- it might output the information for each in some attractive manner;
 - or if each line contains an HSU username, web alias, and a file in their `public_html` directory, it might verify that that file is readable, and if so outputs the URL for that particular file;
 - ...but you get to choose, as long as it somehow transforms/does something with the input that results in output based on what has been read.

Submit your resulting `csv-play.sh` along with at least two sample csv input files with at least 10 lines each and with the redirected output from running your `csv-play.sh` on those input files.