CS 279 - Homework 4

Deadline:

Due by 11:59 pm on **FRIDAY**, September 19.

How to submit:

Submit your files using `~ah270/279submit` on nrs-labs, with a homework number of `4`, by the deadline shown above.

Purpose

To learn more about `vi` and `emacs`, to experiment with hard and soft links, and to see how environment variables are inherited by child processes but cannot be changed in the parent process by changing them in the child process.

Important notes:

- If I don't specify which mode -- octal or symbolic -- to use in specifying a file's permissions, then you may use either mode, your choice.
- It is possible that your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1:

You practiced using `vi` and `emacs` in lab on Tuesday. Recall that there are links to several references for each from the public course web page, linked from the "References" section.

Look over these references (or search the web for others, if you wish).

Find at least one command that interests you that we did not demonstrate in lab for each of `vi` and `emacs`, and try each of them out.

In a file `hw4-1.txt`, include:

- your name
- at least one `vi` command that you found interesting that we did not demonstrate in lab
  - why you found that command to be interesting
  - when you tried that command in `vi`, were you able to successfully use it?
- at least one `emacs` command that you found interesting that we did not demonstrate in lab
  - why you found that command to be interesting

- when you tried that command in `emacs`, were you able to successfully use it?

Submit your resulting `hw4-1.txt` .

Problem 2:

You'll play with links a bit in this problem. (Remember that we found that `emacs` made new copies of files unexpectedly, and that is why use of `vi` is specified for this problem.)

- Create a subdirectory, make its permissions 700, and `cd` to it -- I'd like to keep the output files for this problem uncluttered.
- Create a short-but-non-empty text file of your choice in this subdirectory. Use `cat` with the name of this file, redirecting the result into a file `hw4-2-part1-orig.txt`, so I'll be able to see the original state of the original file you started with.
- Create a hard link to this text file in this subdirectory.
- Create a symbolic/soft link to this text file in this subdirectory.
- Look at the output of `ls -li` -- see how the hard link and the soft link compare to the text file you originally linked to. Then do:

  `ls -li > hw4-2-part2-links.txt`

  ...so I can see that you created these links.
- Use `vi` with the name of your original text file, and noticeably change it in some fashion. Use `cat` or `more` to then look at your text file, your hard link, and your soft link. Then use `cat` with the names of your text file, your hard link, and your soft link, redirecting the result to `hw4-2-part3-chg1.txt`
- Use `vi` with the name of your hard link, and noticeably change it in some fashion. Use `cat` or `more` to then look at your text file, your hard link, and your soft link. Then use `cat` with the names of your text file, your hard link, and your soft link, redirecting the result to `hw4-2-part4-chg2.txt`
- Use `vi` with the name of your soft link, and noticeably change it in some fashion. Use `cat` or `more` to then look at your text file, your hard link, and your soft link. Then use `cat` with the names of your text file, your hard link, and your soft link, redirecting the result to `hw4-2-part5-chg3.txt`
- Now, use the `rm` command to remove your original text file.
- Do the command:

  `ls -li > hw4-2-part6-rm.txt`

  ...so I can see that you removed the original text file.
- Now do the `cat` command with the name of your hard link, redirecting the result into `hw4-2-part7-hard.txt`

  And, do the `cat` command with the name of your soft link, redirecting the result using the `2>` symbol into `hw4-2-part8-soft.txt`

  (Rhetorical question, that you don't have to turn in an answer for: can you figure out why I asked you to use `2>` for the command involving the soft link here?)

Submit your files `hw4-2-part?-*.txt`

Problem 3:

Create a file `hw4-3.txt` that contains your name, and the answers to these questions (each preceded by the part name).

If you did the following three commands:

```
echo "stuff" > stuffy.txt
ln stuffy.txt file1
cp stuffy.txt file2
ls -li
```

3 part a

...how do the i-node numbers of `stuffy.txt` and `file1` compare?

3 part b

...how do the i-node numbers of `stuffy.txt` and `file2` compare?

3 part c

If you used `vi` to now change `stuffy.txt`, would `file1` change? ...would `file2` change?

3 part d

If you used `vi` to change `file1`, would `stuffy.txt` change? ...would `file2` change?

3 part e

If you used `vi` to change `file2`, would `file1` change? ...would `stuffy.txt` change?

Submit your resulting `hw4-3.txt` .

Problem 4:

Use the `env` command to see the environment variables currently set by your shell.

It turns out that you can create a new environment variable by using:

`export` *NAME*=*value*

- interestingly, note that there should NOT be a blank between the variable name and the equal sign, and there should NOT be a blank between the equal sign and the value
- also note that, if the value contains blanks, then it should be in quotes
- the lifetime of this environment variable is from now until this shell terminates

- you can now see this variable's value using the `env` command. You can also see it by using:

  `echo $`*`NAME`*

For example:

```
$ export SOUND=tardis
$ env | grep SOUND
SOUND=tardis
$ echo $SOUND
tardis
```

4 part a

Create a new environment variable of your choice in your current shell.

Show that you succeeded by piping the result of the `env` command to a `grep` for your environment variable name (as I did for `SOUND` above), redirecting the result into `hw4-4a.txt`

Also `echo` the result of putting a `$` in front of your environment variable name (as I did for `SOUND` above), APPENDING the result (`>>`) to `hw4-4a.txt`

4 part b

Create a small bash shell script `env-play.sh` :

- include a descriptive opening comment block including your name and the last modified date
- `echo` a message saying that this is the beginning of `env-play.sh`
- give the result of piping the result of the `env` command to a `grep` for your environment variable name
- call `echo` with the result of putting a `$` in front of your environment variable name
- Now CHANGE your environment variable value (*`NAME=value`* should do it).
- `echo` a message saying this is after changing your environment variable value, and then
- give the result of piping the result of the `env` command to a `grep` for your environment variable name
- call `echo` with the result of putting a `$` in front of your environment variable name
- `echo` a message saying that this is the end of `env-play.sh`

Test this, and verify that environment variables do indeed behave as discussed in class. (Child process does inherit them, and can change its copy, BUT because it gets a copy, its changes don't change the parent's environment variables.)

4 part c

For the new environment variable you created in part a, do the following:

- pipe the result of the `env` command to a `grep` for your environment variable name (as I did for SOUND above), redirecting the result into `hw4-4c-1.txt`
- Also `echo` the result of putting a `$` in front of your environment variable name (as I did for `SOUND` above), APPENDING the result (`>>`) to `hw4-4c-1.txt`

Now run your shell script `env-play.sh`, redirecting the output to `hw4-4c-2.txt`

After running your shell script,

- pipe the result of the `env` command to a `grep` for your environment variable name (as I did for SOUND above), redirecting the result into `hw4-4c-3.txt`
- Also `echo` the result of putting a `$` in front of your environment variable name (as I did for `SOUND` above), APPENDING the result (`>>`) to `hw4-4c-3.txt`

You should see that your shell's environment variable values in `hw4-4c-1.txt` and `hw4-4c-3.txt` are the same.

Now run your shell script a bit differently:

```
source env-play.sh > hw4-4c-4.txt
```

After running your shell script,

- pipe the result of the `env` command to a `grep` for your environment variable name (as I did for SOUND above), redirecting the result into `hw4-4c-5.txt`
- Also `echo` the result of putting a `$` in front of your environment variable name (as I did for `SOUND` above), APPENDING the result (`>>`) to `hw4-4c-5.txt`

Rhetorical question, that you don't have to turn in an answer for: why do the contents of `hw4-4c-3.txt` differ from the contents of `hw4-4c-5.txt`?

Submit your resulting `env-play.sh` and `hw4-4*.txt` files.