CS 279 - Exam 2 Review Suggestions - Fall 2022

last modified: 2022-11-01

Exam 2 BONUS Opportunity

- You can receive (a maximum) ***5 POINTS BONUS*** on Exam 2 if you do the following:
  - Make a **hand-written** Exam 2 study sheet (a single sheet of paper, no larger than 8.5" by 11", on which you have hand-written as much as you would like on one or both sides)
  - Submit a photo or scan of it saved as a .pdf, .png, .jpg, .gif, or .tiff to Canvas **by 9:00 am on Wednesday, November 9** such that I can read at least some significant CS 279 Exam 2 material on it.
  - Please let me know if you have any questions about this, and I hope it helps you in reviewing course concepts more effectively before Exam 2.
  - You are **encouraged** to have this **at hand** as you are taking Exam 2.

Exam 2 Set-up

- You will take Exam 2 in TA 011 on Wednesday, November 9.
  - You are expected to work **individually** on the exam -- it is not acceptable during the exam to discuss anything on the exam with anyone else.
  - You may have your Exam 2 study sheet on hand during the exam. Otherwise, the exam is closed-note, closed-book, and closed-computer/closed-electronic-devices.
  - I expect there will be some multiple-choice questions, and the rest will be short- to medium-answer questions.
- Your studying should include careful study of posted examples and notes thus far.
- You are responsible for material covered in class sessions, lab exercises, and homeworks through and including the Week 10 Lab Exercise (Thursday, October 27) and Homework 7 (due 11:59 pm on Friday, November 4).
  - This review handout is intended to be a quick overview of especially important material.
  - TIP: It is **recommended** to retake/read over the short-answer questions in Canvas from Homeworks 1-7 (and especially Homeworks 4-7) as you are studying for Exam 2!

    These are set up for unlimited retakes, and only keep the highest score, so you will not hurt your grade by doing so!
- Remember that Linux/UNIX commands are **case sensitive** - for example, `Ls` is not the same command type as `ls`. You are expected to use the correct case in your answers.
- You are also expected to follow CS 279 course style guidelines in your answers.

more Bash shell features

- you should be comfortable with more `if` tests (such as those discussed during Week 5)
  - using `(( ... ))` to evaluate and test numerical expressions
  - using extended test `[[ ... ]]` for options such as `=~`
- you should be able to read, write, and understand `while` loops
  - how can you read the lines from a file with a `while` loop?

environment variables

- every process (including your current shell) has a collection of environment variables associated with it, that can be used or set by that process;
- environment variables and local variables
  - what is the scope/lifetime of each?
  - what environment variables does a subshell have? how does it get them? what is the impact on the calling shell if they are changed in a subshell?
- know how to use the `env` command to see those currently in effect
- know how to view the value of a given environment variable using `echo` and `$`
- be aware that a child process has its environment variables set to be a *copy* of its parent process's environment variables
  - (so, it can use its copy of these, and can change them, but when it is complete its environment variables disappear and any changes thus disappear, too)
- you should know how you can change the value of an environment variable
- environment variables you should be familiar with at this point: `SHELL` (contains your login shell), `$0` (your currently-running shell), `TERM` (your terminal type), `HOME` (your home directory), `PATH` (the search path -- list of directories to be searched -- for commands)
- should be able to use the `export` command to specify that a variable will be copied to a shell's subshells
- what does the `source` command do? why might one choose to use it?

shell initialization files such as `.bashrc` and `.bash_profile`

- how can you create command aliases in an initialization file? You should be able to read, write, and understand `alias` commands.
- how can you set your Bash command prompt?
- what does the `PATH` environment variable contain? How is it used? You should know how to reset this environment variable.

processes and jobs and job control

- 3 possible outcomes of a process: success, failure, abnormal termination
  - can tell which via its exit status: 0 - success, < 128 - normal "failure" exit status, >=128 - abnormal termination (e.g.., because of an interrupt or other signal) -- often 128 + the signal code number
  - know how to see the exit status of the last completed command using `$?`
- know the three file descriptors associated with each process -- standard input, standard output, and standard error
  - by default, what is standard input set to come from? ...what is standard output set to go to? ...what is standard error set to go to?
  - how can these each be redirected in the command line?
- what does `ps x` output? what does `jobs` output? What are some of the differences between what these output?
  - which would you use to see the background jobs for the current shell?
  - which would you use to see all of the current processes you own, regardless of the shell they were started in?
  - which contains process ids?
- Foreground and Background Execution
- Starting, Listing, and Killing Jobs; should be able to kill processes you own as well
- should be familiar with several ways to indicate a background job that you would like to now execute in the foreground; should be familiar with job identifiers for background jobs
- should be able to describe and write commands for starting a process in the background, putting it in the background from the foreground, and changing the status of a `Stopped` job to `Running`
- job - a collection of one or more processes [in the current shell]
  - should understand: foreground process, background process, running background process, stopped background process, current job, previous job
  - significance of &, +, -

more UNIX/Linux commands: `sort`, `diff`, `cmp`, `wc`, `which`, `tee`, `touch`

- remember that you can use the `which` command to show which version of a command you are running (to locate it in your `PATH`)
- you should understand and be comfortable calling these commands; you should know how they can be useful, when you might use them, and their effects and/or output when called

pipes - |

- should understand what this is, how to do it, why it is useful, and should be able to read and write commands using this
- what is a filter? What are the benefits of filters?
  - Note that the `sort` command is an example of a filter (it accepts standard input, and outputs the lines given to it in sorted order);
  - `grep` is also an example of a filter, because it can accept standard input, and can, for example, output only those lines containing the given pattern
- how can you use the `tee` command to both write the current pipeline's contents to a file as well as pass it on down the pipeline?

more file globbing options

- `*`, `?`, `[ ]`, etc.
  - example possibility: given a pattern and several file names, you would say for each whether they would be matched or not
  - or, given a directory's contents and a pattern, and you would list which would be output by an `ls` command using that pattern
  - or, given a pattern, you might need to give an example of a filename that it would match, and one that it would not match;
  - and of course you might have to give a pattern that would match specified files

`grep` command

- using `grep` with a BRE and a file name, in which case it outputs the lines in that file containing that pattern
- using `grep` with a BRE in a pipeline, in which case it outputs lines in the previous command's output that contain that pattern
- be familiar with discussed options such as `-l` for names of files with the pattern, `-E` to use EREs, `-c` to give a COUNT of matching lines

regular expressions

- You are responsible for being able to read, write, and understand both basic regular expressions (BREs) and extended regular expressions (EREs)
- You should be able to use regular expressions with the `grep` command, and within a shell script with the =~ operator
- be familiar with important BRE features such as:
  - what `*` means when it follows something

  - `[ ... ]`
  - `^` at the beginning of a pattern, `$` at the end of a pattern
  - subexpressions `\( \)` and backreferences `\1`, `\2`, etc.
  - interval expressions `\{m\}`, `\{m,\}`, `\{m,n\}`
- be familiar with important ERE features such as:
  - what + and ? mean when they follow something
  - unquoted parentheses for grouping subexpressions
  - using | for or
  - unquoted { } for interval expressions

Bash arrays and the `BASH_REMATCH array`

- you should be able to read, write, and understand bash arrays
  - you should be able to create an array, add array elements, modify array elements, access array elements
  - you should know how to find out how many elements are in an array
  - you should be able to obtain all of the indices for an array; you should be able to obtain all of the elements in an array
- you should be able to use the `BASH_REMATCH` array to obtain what matched a subexpression in a previous regular expression; you should be able to use it to obtain what matched that regular expression as well

`/dev/tty` and `/dev/null`

- what are device files?
- what are these special device files?
- what is special about `/dev/null`? How is `/dev/null` often used?