15-131 Jobs, Globs, Man pages - Sample Answers

Task 1. You've been working really hard on a 112 assignment, but your code is taking forever to run. It's 7:30 pm now, WAY past your bedtime, so you decide to time the program overnight and see how long it takes. After waking up from your normal 13.5 hour nightly sleep, your program is still running! You're now convinced you must have an infinite loop somewhere. You type `vim homework.py` into the terminal and press enter, but nothing happens.
What's happening here, and how can you open the file again?

Answer 1. Your program is still running in the foreground, so any command you type (`vim homework.py` for example) will not run through bash. A few options are avaiable: `^Z` to stop the program and put it in the background to run later, or `^C` to just stop it outright.

Task 2. After eating breakfast and coming back from the gym, you sit down to start debugging your code. You make a TON of changes, maybe a bit too many, but you run your code and it terminates! Problem is, you basically erased everything except `print "Hello, World!"`. Luckily, you never actually quit your vim, you just put it into the background. Why could this be helpful and how can you open the file again?

Answer 2. To bring vim back to the foreground (assuming it's your most recent job), all you have to do is `fg`. If it's not your most recent job, use the `jobs` command to view all your active jobs, and `fg %<job number>` to bring vim back to the front. Stopping vim with `^Z` makes it easier to jump back into the file you're working on, and it also preserves your undo history. If you had closed vim, your changes wouldn't be undo-able and you'd have to start over from the beginning!

Task 3. After working a bit more, you get an array out of bounds error. You have no idea which file it could be in! How could you search through all the files in your current directory for array accesses?

Answer 3. Using `grep`, you could run `grep '\[.*\]' *`. If you didn't include the backslahes and tried to do `grep '[.*]' *`, you would match to any line that has the characters `.` or `*` in them, which is probably not what you want.

Task 4. You're finally ready to submit the assignment! Well done! You knew the hard work would pay off. After working so long, you want to make sure nothing goes wrong in this submission process. You've heard that using `tar` to compress files incorrectly can actually result in you losing all of your progress. You want to absolutely make sure you understand how to use it correctly before you throw away so many hours of work. How can you do this without leaving the terminal?

Answer 4. The simplest solution would be to run `man tar` and read the manual page for the tar command. Remember, arguments surrounded by `[]` are optional.