Prep for Linux Lab 1

* Let’s install Firefox
  + Open the Terminal
  + Update our package repositories. This ensures that we have an accurate list of package sources (repositories) with the latest versions of the packages in them.
    - **sudo apt update**
  + Depending on how often you do this, it could take a long time.
  + Do we have firefox installed? Let’s check.
    - Get firefox, if it exists, to do a version check
    - **firefox --version**
  + Nope, run the following to install firefox
    - **sudo apt install firefox**
  + This will download firefox from one of our repositories and install it.
  + Now let’s check the version
    - **firefox --version**
  + Looks good, let’s run it:
    - **Firefox**
  + Great, let’s visit our D2L page:
    - [**https://online.saskpolytech.ca/d2l/home/256478**](https://online.saskpolytech.ca/d2l/home/256478)
  + Uggh, can’t copy and paste. Let’s see if we can fix that.
  + Since we’re using VMWare, maybe we can install the VMWare tools package.
    - **sudo apt install open-vm-tools-desktop**
    - **sudo reboot**
  + Copy and paste should work now.
* Man
  + Linux has system reference manuals. We access them through the command **man**
    - **man man**
    - **man sudo**
    - **man apt**
* Talk about general filesystem layout – mounts, /, directories, home directory
  + The **mount** command attaches the filesystem of an external device to the filesystem of a system. It instructs the operating system that filesystem is ready to use and associate it with a particular point in the system's hierarchy. **Mounting** will make files, directories and devices available to the users.
  + **sudo mkdir /mnt/cdrom**
  + **mount**
  + **sudo mount /dev/sr0 /mnt/cdrom**
  + **cd /mnt/cdrom**
  + **ls**
  + **ll**
  + Mount it in Windows and compare
  + We’ll talk more about mounts, etc, later
* Talk about basic filesystem commands – ls, pwd, cd, touch, rm, mkdir, rmdir, mv
* Viewing files – more, less, cat
  + **Recreate your myFile**
  + **Have it a long file**
* Input and output redirection (<, >, >>) – there’ll be more on this later
  + **cat < myFile**
  + **cat myFile > catOutput**
  + **echo “test append” >> myFile**
  + **more myFile**

## Redirection and standard files

* Three standard files – stdin (0), stdout (1), stderr(2). These are pretty foundational to a lot of I/O in Linux in general.
* File descriptors always point to something – by default, it is your terminal, but you can redirect it.
  + A file descriptor is an entry in the kernel that describes what file is open and other information about that opened file
  + So an open file, that has a file scriptor, will also have information regarding how that file is being interacted with (ie: the terminal)
* “>” redirects” (by default, stdout), “X>&Y” redirects X to the same location as Y. By default it will “clobber” the file, though you can disable it by using the “Set –C” command
* Can redirect to /dev/null (a special file)if we don’t want to see it.
  + A black hole, so to speak
  + **cat myFile > /dev/null**
* ~~Use “ls . /foo” as an example that produces both stderr and stdout – what happens if we swp around the order.~~
  + ~~try > output 2>&1 vs 2>&1 >output~~
* ~~“&>” redirects stdout and stderr at once, and is a shortcut equivalent to “> 2>&1” (processed left to right remember)~~
* We can use “>>” to append instead of replace (use it for the redirect to file, not redirect to descriptor – it doesn’t make sense to append one and overwrite the other anyway)
* Can use exec to redirect forever, since exec without argumenets effects the current shell (instead of replacing it). We can also use something like “Exec 3<file” to open a file as fd3, so you can use it to read from. You can open them the same way, and close by redirecting to -.
* “<” redirects the contents of a file to stdin (“grep blah < file” is a non-useful example, more useful for scripting with read). A lot of < examples will be less useful because most shell commands you’d want to do it with can take in a filename as last argument.
* “<<MARKER blah blah blah MARKER” redirects from a here-is document
* “<<<” redirects a single line

## Bash tricks - Navigation, repeating commands, etc: (Can’t use in scripts)

* Arrow keys to go up through previous commands
* Ctrl-r to search history, Ctrl+p Ctrl+n
* Alt + . use the last word of the previous command
* !! for last command
* !# in combination with history
* !xyz repeats the last command starting with xyz
* !$ gives you the last word
* !!xyz:p prints it so you can check, :h removes trailing file names
* ^blad^blah replaces in the above command