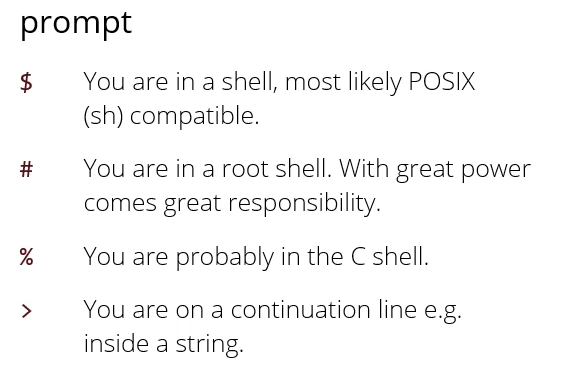
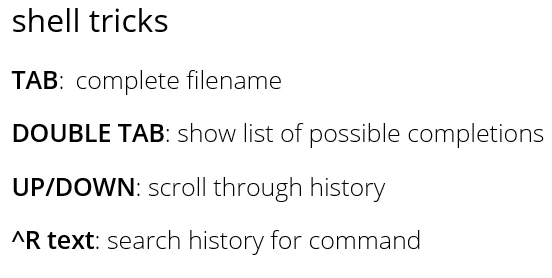
**The Shell**

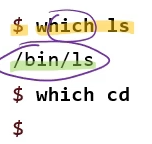
* Interact with Alpine Linux through the shell
  + Shell/terminal/console/command line/(command) prompt
* General work flow is a request/response system
  + You type a command, the Linux system does something, provides a result, you do that and then type the next command



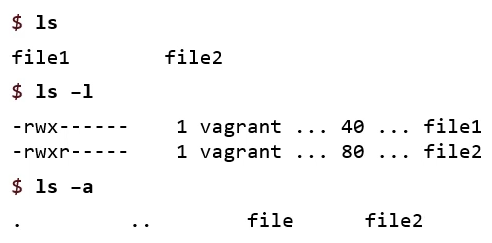
* C shell = alternative shell whose language looks a little like C
* Shell allows you to break strings over multiple lines, > will show you sign that you are inside the string
  + Press ^C to cancel
* To save type in typing:



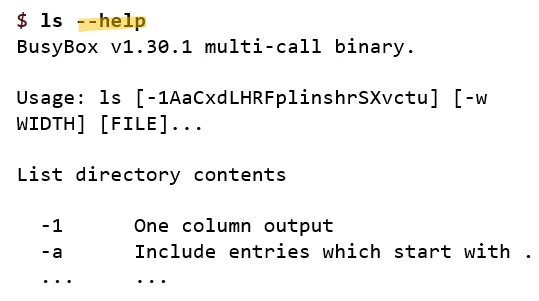
* “bin” is short for binaries, another name for programs
* “which” command helps you find out what happens when you type a command e.g. “which ls”



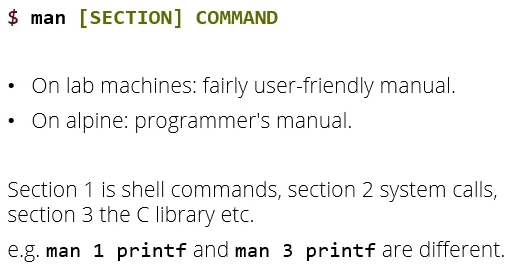
* “which cd” has no output because it is executed internally by the shell itself (not a program file)
* Programs can take options, of which there are two kinds:



* First kind: designated by a single minus sign followed by the letter
* E.g. ls alone will list the files, ls -l (long) will give you a longer list of the same files
* Ls -a (all) will print files with a name that start with a . which are normally hidden
  + . refers to the folder
  + .. refers to the parent folder
* As long as they’re single letter options you can combine them in any order, like ls -l -a or ls -la or ls-al
* A lot of these programs call into a C library behind the scenes called “getopts”, which handles parsing these options for you
  + If you are writing your own shell programs, using this will help
* Second kind:



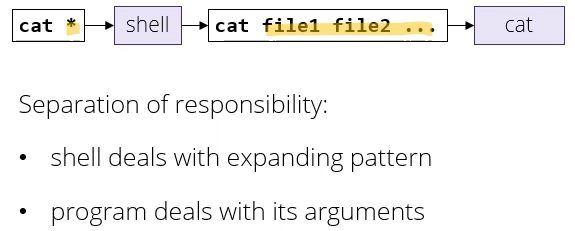
* More than one letter, convention to put two dashes - - then name
* Help will tell you what the command does (list directory contents), what all the options are (usage first [] ), then list them all line by line
* Because there are so many for ls, you will have to scroll
* If you have a terminal command that causes things to scroll and you want to see it a page at a time, you have to put a | (called a pipe) at the end of a command
  + E.g. ls -- help | less
  + Means: put the output of this command into the input of the next command
  + Next command called less
  + Less is a command that reads all of its input, and then displays it a page at a time, can use things like up and down arrow to scroll, and q to quit when you’re done
* More detailed help on all commands is in user manual



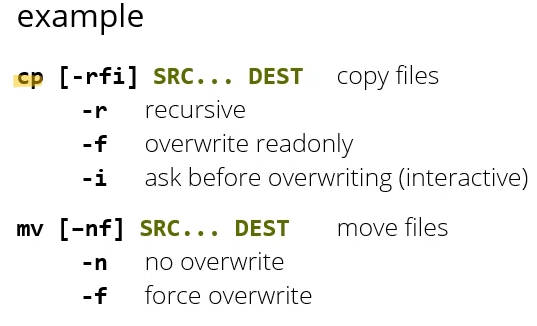
* Things in [ ] are optional (don’t type the [ ] ), green font means replace with something of your choice

**Shell expansion**

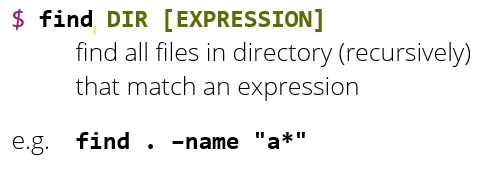
* When you first type something in your shell and press enter, the shell first does shell expansion, where it makes a few changes and then passes on to the actual commands you’re calling
* Sometimes you need the shell not to expand something
* For example, the first thing the shell expands if you have an argument to a program which is \* the shell will expand that to the list of all files in the current folder
* Cat \* (cat stands for canonicate) and will print the contents of a file to the terminal
* If you type cat \*, it will first go to the shell and the shell replaces that with a list of all files in current folder, then goes to cat command
* Cat reads arguments one by one, takes each file and prints to the terminal
* Each command has different responsibilities
* Responsibility of cat command is to read files and print them to the terminal – not to find out what files exist in the current folder: that’s dedicated to the shell



* If you have a word in a command line argument with stars in it, e.g. a\*, the shell will expand that accordingly
  + E.g. a\* stands for any files that start with a and \* stands for any number of letters
* ? matches a single character, which is often useful e.g. if you are importing a lot of images from a digital camera
  + E.g. image???.jpg matches image001.jpg etc.
* If you put a list of characters in [ ] the shell expands that to a single character
  + Like a restricted version of ?
  + You can use – to indicate range
  + E.g. [ab]
  + E.g. image[0-9].jpg would match image0.jpg etc.
* $ is used for any variable name expansion
  + The shell, like any programming language, has variables
  + If you start a word with a dollar sign, it tell the shell to expand to the value of that variable
* If you don’t want something to be expanded in the shell, you can put it in quotes
  + If you put the string in “double quotes” it puts the whole string as a single argument and turns off a lot of pattern matching
    - Turns off \* and ? etc. but still keeps variable interpolation on
    - So can still put $ inside double quotes and it will still expand to a variable value
  + To turn off pretty much everything including variable expansion, you can put a string in ‘single quotes’ and it will pass the string along to the program pretty much unchanged
  + Can put a \ in front of single characters to escape them e.g. \\* \? \[ \$
* In Linux, most important commands have two letter names, e.g. cp (copy)



* Cp takes a list of source files and one destination, and copies from the source to the destination
  + If the destination is a folder, you can take more than one source file and put all those files into that folder
  + Most important options for it:
    - -r (recursive) : if one of the source files is a folder, copy all its source files and subfolders
    - -f (force) : overwrite files in the destination even if they’re read only
    - I (interactive) : even if there’s a file in the destination that isn’t marked as read only, warn me before overwriting just in case
* Mv has pretty much the same syntax but it moves instead of copies (like copy and then delete original)
  + -f force overwrite of files in destination
  + -n no overwrite (if try to move and in destination file already exists, will give you an error)
* E.g. **cp index.html style.css web**
  + Will take the files and then put them in the destination folder on the end (web)
  + Or **cp \* web** will copy all files in the current folder into destination folder web
    - If you use in an empty folder, will provide an error
* Find command will search the current folder or any subfolders



* Find, followed by directory you want to start searching from, followed by expression(s) that say the file(s) you want to look for
  + If you don’t give an expression, it just gives all the files and subfolders starting from where you are searching
  + . means search in current directory
  + – name means the following arguments indicate file names to search for
  + “a\*” means don’t expand a\*, just pass onto command – here the command file itself can deal with a\* as a pattern for files starting with a