Unix CMD – Commands

~/directory/etc ~ represents home directory instead of starting from base

stdin input stream with default value of 0 (keyboard)

stdout output stream (display)

stderr error stream (display)

permissions every file has set read/write/execute permissions for the “user” “group” and “others” the “user” class determines the permissions that the owner of the file has, the “group” class is the permissions that any group of user has, the “others” class is all others. Using ls -l will show permissions in groups of three for “user” “group” and “others” by displaying “r” “w” “x” if permissions are allowed, if there is a “-“ then the permission is not allowed.

shebang command at beginning of script which tells computer which interpreter to use, defined by #!/locationofinterpreter. in the case of cloud9 it would be “#!/home/ec2-user/.rvm/rubies/ruby-2.6.3/bin/ruby”, however as the former is specific to cloud9, it is better to direct the shebang to the environment path of the interpreter using “#!/usr/bin/env ruby”

env var environment variables are key/value pairs where the information of the current environment is stored, these values are those such as the username, tempfiles location, home location, and ruby interpreter location. All programs have access to the environment variables. environment variables can be set using “export <var\_name>=<value>”. environment variables can be useful for storing data which is specific to the user, such as passwords, simply make the ruby code read the env\_var of the password instead of inputting the password itself – “secret\_key = ENV['PASSWORD']”. env\_var added are only available in the current instance of the shell, profile files are required for permanent env\_vars.

PATH PATH is a colon separated list of directories where shell will look for programs you ask it to run. when typing a command/program without specifying a path, the shell will search the PATH directories for the command/program. Versions of ruby can be switched using “rvm use <version>”, which changes the PATH directory. PATHS can be modified using “export PATH=$PATH:<path>” which will add <path> to the end of PATHS

Profile Files Each time the shell is loaded, the environment variables from the ~/.bash\_profile are loaded. to make permanent env\_vars add the values to the bash\_profile, e.g. “echo "export SEASON=winter" >> ~/.bash\_profile”

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Switches** |
| date | Displays time and date |  |
| ls | list contents of current directories | * -A - to give hidden files * -l – to give long format inc permissions |
| cd | changes directory | * .. - goes to parent * ./directory – goes to directory in current folder * ~/directory – goes to directory in home folder |
| pwd | print full working directory |  |
| touch <filename> | creates blank file with filename in working directory |  |
| rm <filename> | removes file named filename from working directory) | * -r – will delete directory even if not empty * -i – will give interactive delete, confirming each item to delete * -f – force deletes all |
| mkdir <dirname> | makes directory with dirname in working directory |  |
| rmdir <dirname | removes directory with dirname in working directory (will not delete if not empty) |  |
| cp <filename> <newfilename> | copys <filename> to <newfilename> |  |
| mv <filename> <filename> | moves <filename> to <filename>, can also be used to rename by changing second filename |  |
| cat <file> | concatenate file, ie displays what is inside file directly in terminal |  |
| cat > <filename> | prompts user to add text to new file called filename |  |
| cat <file1> <file2> > <newfile> | adds contents of file1 and file2 to newfile |  |
| less <textfile> | shows scrollable contents of textfile, close using q | * search in less using /<query> |
| head -<number> <textfile> | shows first <number> lines of <textfile> |  |
| tail -<number> <textfile> | shows last <number> lines of <textfile> | * -f <logfile> - stores terminal inputs on in logfile |
| sed -n <expression>p <file> | selects line depending on expression |  |
| man <command> | shows manual for command |  |
| <command1> | <command2> | pipelines the output (stdout) of command1 into the input <stdin> of command2. outputs of command1 are not shown on screen. |  |
| <command> > <file> | redirect output to file |  |
| <command> < <file> | redirect input of command to come from file |  |
| <command> >> <file> | append to file |  |
| \* .txt  \*n\*  new\*.txt | \* acts as a wildcard meaning all files with ending .txt will be selected. |  |
| find . -name “\*.txt” -print | finds all files with name \*.txt and prints, piping find (to list a directory) into grep allows for full regular expression use |  |
| grep <query> <file> | global regular expression print - searches for query(regex) and highlights, can search in specific file or files given in file argument | * -i – ignores case distinctions * -v – displays lines which do not match * -n – precede each matching line with a line number * -c – print only total count of matched lines * -e regex option to allow for more than one condition |
| wc <file> | provides output of linecount, wordcount and charatercount | * -l – provides number of lines |
| whoami | outputs username |  |
| chmod <person><+/-><permission> <file> | changes permissions of file | <person>   * “u” user * “g” group * “o” others * “a” all   <permission>   * “r” read * “w” write * “x” execute |
| echo “<text>” | outputs text to the display, useful for creating short text files by piping. |  |
| ps | shows processes running | * x – shows all processes running on computer |
| kill <processnumber> | kill specific process |  |
| sleep <seconds> | stops command prompt for returning after x seconds |  |
| <processorcommand> & | & runs job in background allowing access to the command prompt immediately for long processes. To force background after process has started running “^Z” then type “bg” |  |
| jobs | show tasks which are running in background |  |
| fg %<jobnumber> | bring job into foreground. fg with no job number will suspend the last process |  |
| kill %<jobnumber> | kill job |  |
| vim <filename> | similar to nano however more advanced, requires command to insert text – use vim cheatsheet |  |
| clear | clears cmd |  |
| df | space left on filesystem |  |
| history | shows command line history as list | * set history=<number> to change the length of the history buffer |