Command	Flags	BuiltIn	Description	Examples
man		N	Access documentation (manual pages) - Takes command you want to look up as an argument	man ls
info		N	An alternative documentations system with a builtin menu system	info bash
help		Y	Access the help page for builtin features and commands of the bash shell.	help help echo
pwd		Y	Prints shell’s current working directory	pwd
cd		Y	Changes the shell's current working directory to the path given	cd /usr/games cd ..
ls		N	Lists the contents of a specified directory to the standard output (if no arguments then list of contents of working directory)	ls ls / ls /usr/bin
	-l		Long listing option - Prints information about the metadata -- ids, permissions, timestamps, etc	ls -l
	-a		Includes information about hidden files and directories (names beginning with **.**)	ls -a ls -la
	-R		Recursive call to list all of the tree below	ls -R ls -R /usr
mkdir		N	Creates new directories	mkdir myDirectory
rmdir		N	Remove/deletes directories	rmdir myDirectory
touch		N	Creates empty files or updates the access time	touch myfile
rm		N	Removes files (support recursive removal -r, see man rm)	rm myfile
cp		N	Copy files and directories (supports recursive copy -r, see man cp)	cp /var/log/*.log /home/jovyan
mv		N	Can move files and directories. It can also be used to rename files.	mv source desitination mv oldname newname
cat		N	Reads bytes from standard input and writes them to the standard output. If arguments are given, then the bytes will be read from the arguments instead of standard input	cat /etc/passwd cat <in >out cat /etc/passwd > ~/mycopy
file		N	Inspects the specified files and attempts to tell the “type”	file /bin/ls file /etc/passwd file /usr/include/stdio.h
chmod		N	Changes the mode bits of a file or directory, i.e. parts of the meta data such as permissions and if the file is marked as executable or not	chmod ugo-rw myFile
	+x		Marks the specified files as executable	chmod +x myFile
chgrp		N	Changes the group meta data information of a file or directory	chgrp games myFile
chown		N	Changes the ownership meta data information of a file or directory	chown jovyan myFile
wc		N	Counts the characters, words and lines of an ASCII file (by default counts all three). If no arguments are given, wc will read data from its standard input	wc /etc/passwd cat /etc/passwd | wc
	-l		Counts lines only	wc -l /etc/passwd
	-w		Counts words only	wc -w /etc/passwd
	-c		Counts characters only	wc -c /etc/passwd
grep		N	Very powerful ASCII search tool that can be used to filter data. If no files given it will read data from its standard input (see man grep)	grep jovyan /etc/passwd cat /etc/passwd | grep jovyan
	-i		Ignores lower/upper case distinction while searching	grep -i MySeaRch /usr/include
	-R		Searches recursively through all the files of a subtree specified	grep -R mysearch /usr/include
find		N	Very powerful tool for recursively finding files and directories that match various criteria (see man find)	find /usr -type d -name include
date		N	Prints the current date and time to standard output	date
echo		Y	Prints the arguments to the standard output with a newline at the end	echo hello echo $HOME
	-n		Does NOT include a newline at the end	echo -n hello
	-e		Expands backslash-escaped characters (see help echo)	echo '\thello\n\ngoodbye'
history		Y	Shows the history of the commands lines executed on the terminal	history
!n		Y	After checking history, you can use "!" along with the number of the command to re-execute it	!5
!!		Y	Shortcut to re-execute last command	!!
read		Y	Reads a single line from standard input (to scan in user input) and splits input into words - assigning input to variables if provides (i.e. after example $var1 is "Hello", $var2 is "World!")	read var1 var2 Hello World!
test		Y	Evaluate conditional expression (supports lots of string and path predicates/tests; see help test) Exit status indicates if the expression is true or false	test -n $X test -z $Y test -a ~/readme
[[		Y	More general use of the *test* command. Supports all predicate conditions and combing with &&, ||, and ()	[[(-n $X && -z $Y) || -a ~/readme]]
((		Y	Integer arithmetic expression evaluation (same as builtin *let* command see help '((' and help let)	((x > 10 && x <20)) ((i++))

id		N	Print UNIX credential information (user id, username, group ids and names)	id
ps		N	Powerful command for listing currently running processes (many options see man ps)	ps ps axf
jobs		Y	List this shell's "jobs", commands/pipelines, started by this shell (see help jobs)	jobs
bg		Y	Put specified job into the background, as if it had been started with & (see help bg)	bg %1
fg		Y	Move specified job into the foreground (see help fg)	fg %3
kill		Y	Send a signal to a process or job. Most common use is to kill (destroy) a currently running process or job. You can, however, use it to send any signal to any process (see kill -L and man kill)	kill %4 kill 32 kill -SIGSTOP 35

Expansions: Before running any commands the shell will scan the input line and replace certain sequences of characters in "interesting" way. These are called expansions. Below are five of most commonly use.

Parameter and Variable Expansion:
The shell supports a notion of variables some special variables.
1. Create and set the a variable use =: Eg. x=hello
   To set a variable to string composed of multiple words we can use quotes
   Eg. x="hello and goodbye" or x='hello and goodbye'
2. Expand/recall the value of a variable prefix name with $ (optionally enclose name in with {}). Eg. $x or ${x}
3. Combining, setting, and expanding to update variables

```
   Eg. y="walk into a bar.  "
          x="A cat"
          x="$x and a dog, "
          y="${x}$y"
          echo "$y"
```

4. There are many special variables and parameters that the shell can expand that you don't need to set but will expand to, or let you control, useful things
   a random integer: RANDOM : Eg. echo $RANDOM
   path of home directory: HOME: Eg. echo $HOME
   exit status of last command: ?: Eg. echo $?
  Some can be set to control the behaviour of the shell:
    The list of locations to look for external commands: PATH: Eg.
      echo $PATH
      PATH=$PATH:/home/jovyan
   The prompt string PS1: Eg.
      echo $PS1
      PS1="nice> "
  See help variables and man bash for more details

Arithmetic Expansion:
The shell supports the ability to do some basic arithmetic that you can use. Simply prefix you expression with $((and suffix it with)). Eg.

```
   echo $((1+1))
   x=12
   y=13
   echo $((x + y * x))
```

Note you can also do math without expanding by using the syntax for arithmetic expressions ((Eg. ((x=x+25)) or ((i++)) or ((x<10)). All of these will do the specified arithmetic but not print any result and only set an exit status code indicating success or failure.

Process Substitution:
The shell can let's us substitute the standard output from a command as part of a command we want to execute. To do this we prefix the command we want the output from with $(and suffix it with). The command can be as complicated as we like. Eg.

```
   x=$(ls /)
   echo $x
   x=$(echo -n "contents of / is: "; ls /)
   echo $x
   cnt=$(cat /etc/passwd | grep nologin | wc -l)
   echo $cnt
```

Filename Expansion:
Often times we want to name files or directories as part of our commands the shell will expand certain patterns with names of directories and files that match the pattern (this is often thought of as wildcards). Eg.
  * will be replaced will the name of all the files and directories in the current working directory: Eg. echo *
  * can be used in arbitrary ways as a wildcard: Eg.
      echo /b*
      ls -l /usr/s*/python*

Tilde Expansion:
As a short cut the shell will substitute the value of the HOME variable for the ~ character (this is a convenient shortcut) Eg.
    cp /etc/passwd ~/mycopy

<u>I/O Redirection:</u>
All commands have three default standard I/O Streams automatically opened. Command line syntax can be used to "redirect" them to different locations than the default.

Standard Input (FD=0) Redirection ('<'):
  cmd < file   : the command will be run with its standard input connected to the specified file
        Eg. cat < /etc/passwd

Standard Output (FD=1) Redirection ('>' and '>>'):
  cmd > file   : the command's standard output will be connected to the file. If the file exits its contents will be completely replaced
        Eg. cat > /home/jovyan/myFile
  cmd >> file : the command's standard output will be connected to the file. If the file exists the output will be **<u>appended</u>** to the end

Standard Error (FD=2) redirection ('2>' and '2>>'):
  cmd 2> file   : the command's standard error will be connected to the file. If the file exits its contents will be completely replaced
        Eg. cat > /home/jovyan/myFile
  cmd 2>> file : the command's standard error will be connect to the file. If the file exists the output will be **<u>appended</u>** to the end

Combining:
  cmd < file1 > file2 2> file3 : All three of the command's outputs will be redirected to the specified files
        Eg. cat < /etc/passwd > /home/jovyan/foo 2> /home/jovyan/errors

Bash Programming Support:

For Loop Examples:
    Integers: for ((i=0; i<10; i++)); do echo $i; done
    List of words: for i in a b c d e; do echo $i; done

While Loop Examples:
    Integers: while ((i<10)); do echo $i; done
    Combined with Read: while read i; do echo $i; done
                         ls | while read i; do echo $i; done

If Statement Examples: The three common usages
  Use Exit Status of an external command:
      if grep jovyan /etc/passwd; then echo Found; else echo NOT Found; fi
  *Use Exit status of Builtin <u>Integer</u> Test Command***:**
      if ((x<10)); then echo x is just right; else echo x is too large; fi
  Use Exit status of Builtin <u>General</u> Test Command:
      if [[$x = "hello world"]]; then echo goodbye; else echo what?; fi

Pipelines :
  Chain commands together -- output of one is the input to the other:
    ls /usr/include | grep stdio.h
  A pipeline has an exit status and can be used as if it where single command:
    if ls /usr/include | grep stdio.h; then echo "something stdio.h exists in /usr/include"; fi

Use the standard output of a command as a value:
    echo "The current date and time is: $(date)"
    x=$(ls | grep my | wc -l)

Quoting: The shell supports the use of single and double quotes to control expansion and word splitting.
Single quotes: no expansion and no word splitting (spaces are ignored) Eg.
    x='$HOME is a nice string'
    echo $x
Double quotes: Some expansions but no word splitting Eg.
    x="$HOME is a nice string"
    echo $x

Keyboard Shortcuts:
Up and down arrows: go back and forward in command history
Tab: command and file completion

Special Directories:
Current Directory: .
Parent Directory: ..

Path (File/Directory) Permissions:
A Path can be marked for read (r) write (w) or execute (x). Permissions can be set for its owner, group and all other users. Additionally: The type of a path is either a regular file, directory, pipe or other form of special file. *<u>ls -l</u>* can be used to inspect the type and permissions information. The first character indicates the type: **-** means the path is a regular file, **d** directory, **p** pipe, **l** link, **b** block special, **c** character special, etc. The next nine characters indicate the user, group and other permissions.