***UNIX IMP Commands***

***BASIC Commands:***

1. **Current directory: Cd**

* cd: takes to current working directory
* cd directory path
* cd .. : takes one directory back
* cd - : takes to the last directory worked in
* cd /: Takes you back to the [root directory](https://www.computerhope.com/jargon/r/root.htm) of the current drive
* cd /: Takes you back to the [root directory](https://www.computerhope.com/jargon/r/root.htm) of the current drive.

1. **ls -lrt**: lists all the files in directory

* ls -a lists all contents in the working directory, including hidden files and directories

1. **wc**: Gives word counts in a file. Can also be combined with various commands like cat, grep etc. to generate customize output.

* cat volcanoes.txt | wc
* wc -l : To count the number of lines

1. **cat**:

* cat oceans.txt > continents.txt: > takes the standard output of the command on the left, and redirects it to the file on the right.
* cat glaciers.txt >> rivers.txt

>> takes the standard output of the command on the left and *appends*(adds) it to the file on the right.

1. **Copy command**: cp [*options*] *source* *dest*

* cp main.c def.h /home/usr/rapid/: copying 2 files
* cp -R dev bak: Copy all files and directories in dev recursively to subdirectory bak
* cp -i test.c bak : Interactive prompt before file overwrite

1. **Move** Command: mv

* mv file1 file2 (to move file1 to file2 after running this command file1 does

not exist)

* mv -i file1 file2 : Move file interactively

1. **Remove** file: rm

* rm filename : removes file from the current directory
* rm -r file: removes file recursively from all the subdirectories
* rm -I file: remove file interactively.
* rmdir: removes empty directory
* rmdir -p directory: **Nested remove (-p)** Removes nested directory if empty, removes in bottom up order

1. **Head:**

* **head filename :** Gives the top rows of the file
* **head -n filename:** To display first n lines of the file**.**

1. **Tail:**

* tail filename: Gives the bottom row of the file
* tail -n filename**:** To display last n lines of the file**.**
* tail -r filename : to print in reverse order

1. **touch**: create a 0 byte file.

* Touch filename

If the file exists, touch is used to update the modification time of the file.

1. **Cut** command:

Cut command in unix is used to select sections of text from each line of files. You can use the cut command to select fields or columns from a line by specifying a delimiter or you can select a portion of text by specifying the range or characters

**Syntax**: cut [-c] [-f] list [-d delim] file

* Cut command cannot print fields in any other order than given in the input file. For example if fields are desired in order 3,2,1 then cut will still give the results in the same order in which they appear in actual file.
* Cut command cannot handle files having multi-character delimiters. For example if file has delimiter double pipe then cut command cannot parse it. For both of the above limitations awk command can be used as an alternative.

1. **Paste** Command: Paste is a Unix command line utility which is used to join files horizontally (parallel merging) by outputting lines consisting of the sequentially corresponding lines of each file specified, separated by tabs(or any other delimiter), to the standard output. It is effectively the horizontal equivalent to the utility cat command which operates on the vertical plane of two or more files.

* **paste –s :** Concatenate all of the lines of each separate input file in command line order. The NEWLINE character of every line except the last line in each input file will be replaced with the TAB character, unless otherwise specified by the -d option
* **paste-d**delim-list : Consecutively use the characters in delim-list instead of TAB to separate merged lines. When delim-list is exhausted, start again at its beginning.

1. **du**: This command reports disk usage (i.e., the amount of space taken up by a group of files). The du command descends all subdirectories from the directory in which you enter the command, reporting the size of their contents, and finally reporting a total size for all the files it finds. To find out how much disk space your files take up, switch to your home directory with the [cd](https://kb.iu.edu/d/afsk#cd) command, and enter: du
2. **File transfer**:

* **SFTP:**

Go to the path where the file is present

**Step 1:** sftp (NTID)@(server to which you want to send the file)

**Step 2:** Give your unix password

**Step 3:** cd (the path where you want to send the file)

**Step 4:** place the file “put (filename)”

**Step 5:** Toquit sftp

Bye

* **FTP:**

Steps to Use FTP command:

**Step 1:** Login to the server and go the directory where the files need to be copied.

St**ep 2:** ftp <server name>

**Step 3:**  It prompts to enter the username and then password of the new server. Enter all the details.

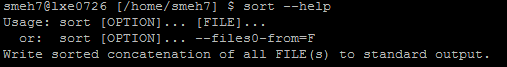
**Step 4:** cd <directory name from where files need to be copied>

**Step 5:** get <file name>

**Step 6:**  File will be copied to the required directory.

***Advanced UNIX Commands for testing:***

1. **SORT Command:**



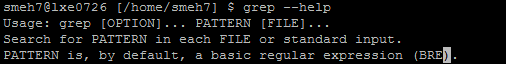
1. **sort [Filename]** :- The sort tool will sort lines alphabetically by default.
2. **sort –r [Filename]** :- To sort in reverse order pass the -r option to sort.
3. **sort –n [Filename]** :- To sort by number pass the -n option to sort.
4. **sort –f [Filename] :-** If a file has uppercase and lowercase content sort will order uppercase first, to sort and ignore case use the -f option.
5. **sort –fs [Filename]:** The -s option in the second command stabilizes the sort, ensuring that identical lines are output in the same order in which they occurred in the input.
6. **sort –c [Filename] :-** To check if a file is already sorted pass the -c option to sort. The sort tool can be used to understand if this file is sorted and which lines are out of order. If there is no output then the file is considered to be already sorted.
7. **sort –u [Filename] :-** By using the -u option the file can be sorted and stripped of duplicates.
8. **sort –k [fieldvalue on which sorting needs to be done] [Filename] :-**To sort by items not at the beginning of the line pass the -k option to sort along with a number of value of the field to sort on. This will write the result to standard output.
9. **sort -t [delimiter] [Filename]:-** To sort by a delimiter pass the -t option to sort along with the delimiter value.
10. **sort –m [Filename]:-** To merge already sorted files; do not sort
11. **sort –o [Filename]:-** To write result to FILE instead of standard output
12. **Uniq:**

The uniq command takes input and removes repeated lines. Because uniq only removes identical adjacent lines, it is often used in conjunction with sort to remove non-adjacent duplicate lines.

Example: sort ~/roster.txt | uniq

1. **Sort filename | uniq -ic:** To count duplicate lines while ignoring case.
2. **Sort filename | uniq -Id:**The -D option inverts the behavior of uniq, and prints only the duplicated lines.
3. **GREP Command:**

The **grep** command in UNIX is a command line utility for printing lines that match a pattern. It can be used to find text in a file and search a directory structure of files recursively. It also supports showing the context of a match by showing lines before and after the result and has support for regular expressions in pattern matching.



1. **grep 'text to be found' [file]** :- To find text in a file pass the string you are looking for to grep followed by the name of the file or files.
2. **grep 'text to be found' [file]| wc –l**:- To find the count of the matches. This can be useful if you are looking to edit a file and want to launch vi editor and go straight to the line (vi +40565 File).
3. **grep -H 'computer' /usr/share/dict/words/:** To print the filename for a match use the -H option. This is automatically invoked when grep is given more than one file to search.
4. **grep –r 'to search' /home :-** To search a pattern recursively in a given directory
5. **grep –v ‘pattern’ filename :-** To search for the inverse of a pattern use the -v option. This will print inverse matches to standard output.
6. **grep –i ‘pattern’ filename :-** To ignore case when searching use the -i option. By default grep will respect case.
7. **EGREP Command**: Search for PATTERN in each FILE or standard input. PATTERN is, by default, a basic regular expression (BRE).



**Syntax**: egrep [OPTION]... PATTERN [FILE]..

Example: egrep -i 'hello world' menu.h main.c

**Also**, can be used for searching by using multiple patterns using a pipe operator.

egrep ‘hello | world | bye’ menu.h main.c

1. **Tr Command :** The **tr** command deletes or substitutes characters from standard input and writes the result to standard output. The **tr** command performs three kinds of operations depending on the strings specified by the String1 and String2 variable and on the flags specified.

**Syntax:**

**tr** [  [**-c**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111430) | **-cds**| **-cs** | [**-C**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111375) | **-Cds** | **-Cs** | **-ds** |  [**-s**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111526)] [  [**-A**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111369)] [*String1*](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111538) [*String2*](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111544)

**tr** { **-cd**| **-cs**|[**-Cd**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111375) | **-Cs** |  [**-d**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111482)|  [**-s**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111526)} [  [**-A**](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111369)] [*String1*](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.cmds5/tr.htm?view=kc#tr__row-d3e111538)



1. To translate braces into parentheses, type:

**tr '{}' '()' < textfile > newfile**

This translates each { (left brace) to ( (left parenthesis) and each } (right brace) to ) (right parenthesis). All other characters remain unchanged.

1. To translate braces into brackets, type:

**tr '{}' '\[]' < textfile > newfile**

This translates each { (left brace) to [ (left bracket) and each } (right brace) to ] (right bracket). The left bracket must be entered with a \ (backslash) escape character.

1. To translate lowercase characters to uppercase, type:

**tr 'a-z' 'A-Z' < textfile > newfile**

1. To delete all NULL characters from a file, type:

**tr -d '\0' < textfile > newfile**

1. To replace every sequence of one or more new lines with a single new line, type:

**tr -s '\n' < textfile > newfile**

1. To replace every sequence of characters in the <space> character class with a single # character, type:

**tr -s '[:space:]' '[#\*]'**

1. **FIND Command:**

The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.

By using the - exec other UNIX commands can be executed on files or folders found. The -exec or -execdir options run without further prompts. If you prefer to be prompted before action is taken, replace -exec with -ok or -execdir with -okdir.

**Usage:**

* 1. **find ./foo -name foo.txt** : To find a single file by name pass the -name option to find along with the name of the file you are looking for.
  2. **find ./foo -name foo.txt –**delete: To find and delete a file pass the --delete option to find. This will delete the file with no undo so be careful.
  3. To be prompted to confirm deletion combine -exec with rm -I : **find ./foo -name foo.txt -exec rm -i {} \;**
  4. **find ./foo -type d -name bar :** To find a directory specify the option -type d with find.
  5. **find ./foo -mtime -1**

**find ./foo -mtime +1**

To find files by modification time use the -mtime option followed by the number of days to look for. The number can be a positive or negative value.

* 1. **find ./foo -perm 777**: To find files by permission use the -perm option and pass the value you want to search for.
  2. **find ./foo -type f -name bar -exec chmod 777 {} \;** :To find and operate on file use the -exec option. This allows a command to be executed on files that are found.

**find . -name "rc.conf" -exec chmod o+r '{}' \;**

* 1. **find ./ -type f -name "\*.md" -exec grep 'foo' {} \;** :Another use of combining find with exec is to search for text within multiple files.
  2. **find . -type f -print | xargs grep "example:** Gives similar output as above.
  3. **find . -type f -empty:** Find an empty file within the current directory.
  4. **find /home -user exampleuser -mtime 7 -iname ".db":** Find all .db files (ignoring text case) modified in the last 7 days by a user named exampleuser.
  5. **find . -name "\*.bak" -delete:** find locates all files ending with .bak in the hierarchy starting at the current directory and fully recursing into the directory tree.

1. **Sed Command:**

**Sed** is a Stream Editor used for modifying the files in UNIX (or linux). Whenever you want to make changes to the file automatically, **sed** comes in handy to do this.

We can do many things apart from replacing text with sed. Here are the features of **sed** with examples.

1. **>sed ‘s/unix/linux/’ File.txt:** Sed command is mostly used to replace the text in a file. Here the "s" specifies the substitution operation. The "/" are delimiters. The "unix" is the search pattern and the "linux" is the replacement string.  
   By default, the sed command replaces the first occurrence of the pattern in each line and it won't replace the second, third...occurrence in the line.
2. **>sed 's/unix/linux/n' file.txt:** Replacing the nth occurrence of a pattern in a line.
3. **>sed 's/unix/linux/g' file.txt:** Use **g** (global) to replace all the occurrence of the pattern in a line.
4. **>sed 's/unix/linux/ng' file.txt:** Replacing from nth occurrence to all occurrences in a line.
5. **>sed -e 's/unix/linux/' -e 's/os/system/' file.txt:** Sed provides -e option to run multiple sed commands in a single sed command. We can run multiple sed commands by piping the output of one sed command as input to another sed command.
   1. File.txt has following contents: **unix** is great **os**. unix is opensource. unix is free os.
   2. Output comes out to be: **linux** is great **system**. unix is opensource. unix is free os.
6. **>sed '3 s/unix/linux/' file.txt:** We can restrict the sed command to replace the string on a specific line number. Here sed command replaces the string only on the third line. We can specify a range of line numbers to the sed command for replacing a string.
7. **>sed '1,3 s/unix/linux/' file.txt** ---replaces the lines with range from 1 to 3. **>sed '2,$ s/unix/linux/' file.txt** ---Here $ indicates the last line in the file. **>sed '/linux/ s/unix/centos/' file.txt ---** Here the sed command first looks for the lines which has the pattern "linux" and then replaces the word "unix" with "centos". We can specify a pattern to the sed command to match in a line. If the pattern match occurs, then only the sed command looks for the string to be replaced and if it finds, then the sed command replaces the string.
8. **>sed '2 d' file.txt**

**>sed '5,$ d' file.txt:** We can delete the lines a file by specifying the line number or a range or numbers.

1. **>sed 'p' file.txt:** We can make the sed command to print each line of a file two times.
2. **>sed -n '/pattern/p' file:** To print only the line containing the pattern.
3. **>sed '/pattern/d' file:** To delete the line containing the pattern.
4. **>sed '/Yahoo/q' file:** Print lines till you encounter a specific pattern.
5. **>sed -n '1,3p' file OR sed '3q' file:** To print Range of Lines, first 3 lines here.
6. **>sed -n '/Yahoo/,/Live/p' file:** Similar to give line number ranges, sed can also work on pattern ranges. Say, to print from lines between patterns "Yahoo" and "Live".
7. **>sed -n '/Redif/,$p' file:** To print the lines from pattern "Redif" till the end of the file
8. **>sed -n '1,/Inbox/p' file :** To print contents from the beginning of the file till the pattern "Inbox".
9. **Awk Command:**

AWK is an interpreted programming language used for text processing, Producing formatted text reports, Performing arithmetic operations, Performing string operations, and many more.

**The syntax of awk is**: *awk 'pattern{action}' file*

1. **awk '{print $3 "\t" $4}' marks.txt**: To print specific columns from the input field)
2. **awk 'NR!=1{print $1}' file1:** To omit the headers & print only data
3. **awk '{print $0}' file1** -To **print the entire file contents**
4. **awk -F"," '{print $1}' file1** - awk has a command line option "-F' with which we can specify the delimiter. Once the delimiter is specified, awk splits the file on the basis of the delimiter specified.
5. **awk '{print $1,$3}' FS=","** file1 To read delimited file
6. **Chmod Command:**

**Syntax:** chmod [who][+,-,=][permissions] filename

1. To change access permissions: chmod 777 filename
2. To change permissions recursively: chmod -R filename

## Multifile Commands:

Working with multifiles will be similar to regular files, by adding a prefix of m\_<command> most of the regular operations can be carried out. Following are the frequently used commands used with the multifiles.

|  |  |  |
| --- | --- | --- |
| **Command** | **Syntax** | **Purpose** |
| m\_cat | m\_cat filename | Similar to unix cat command |
| m\_mv | m\_mv *source\_path1.. dest\_path* | Similar to unix move command to move files  from one location to other |
| m\_rmfs | m\_rmfs *path* | To remove the multifile system |
| m\_rm | m\_rm  *url* [*url* ...] | Similar to Unix rm command to removes  multi files or files. |
| m\_chmod | m\_chmod *mode url* [*url* ...] | Similar to Unix chmod to provide permissions to  multi files |
| m\_touch | m\_touch *url* [*url* ...] | Similar to Unix touch command to create a empty  multi files |
| m\_cp | m\_cp *source\_file\_url1..* *dest\_url* | Similar to Unix Copy command to create copies of multi files |
| m\_wc | m\_wc myrecfmt.dml mfs/mydata.dat | This command is used to count number of records from one or more data files/multi files. |
| m\_kill | m\_kill [ signal] {jobname.rec | jobname }  signal – can have three values –TERM –KILL and –QUIT | This command is used to kill a running job. The command blocks until it can verify that the job is killed or after AB\_TIMEOUT seconds, whichever comes first. This command will be useful incases when jobs were started accidentally |
| m\_rollback | m\_rollback [-d] [-i] [-h] [-kill] recoveryfile  Options:  -d  Delete the job along with its recovery file and any log files it created.  -i  Display the state of the job and prompt the user whether the job should be deleted.  -kill causes m\_rollback to attempt to kill the job | This command is used perform a manual rollback incase of any Abinito job failure. It may not always be possible for the Co>Operating System to restore the system to an earlier state. |

1. **M\_dump Command:** To readthe data file (.dat) using a dml in Unix.

**Syntax**: m\_dump dml\_path file\_tobe\_read

Various versions of m\_dump:

1. **You can also convert multi file to serial file:**  
   m\_dump <dml\_name>  <mfile:mfs4/file\_name>    >   serial\_file\_name
2. **This command displays records starting from 5th row to 9th row in xyz.dat file:**

$ m\_dump xyz.dml xyz.dat -start 5 -end 9

1. **Exporting data from file to excel:**

m\_dump dml\_path file\_tobe\_read -show -html >> complete\_file\_path\_ofexcel

1. **Selecting specific data from the file using select clause**:

* m\_dump arr\_loader\_org\_entity\_full\_table.dml arr\_org\_entity\_lkp.dat -select 'n\_org\_refn=="Party\_1008815491"'
* m\_dump rec.dml file.dat -select 'salary > 50000'
* m\_dump <DML> <DATA> -select ""(date(""MMM"")) SNAP\_DT == 'AUG'""

## m\_db:

This command is used for performing database operation from the command prompt. Following are the various m\_db commands that are used frequently.

|  |  |  |
| --- | --- | --- |
| **Command** | **Syntax** | **Purpose** |
| m\_db unload | m\_db unload dbc\_file -table tablename | To Unloads data from database table, select or expression to stdout.  *Eg: m\_db unload mydb.dbc -table 'fred.mytable'* |
| m\_db truncate | m\_db truncate dbc\_file -table tablename | To truncate a table. |
| m\_db test | m\_db test dbc\_file | Runs diagnostic tests against your database for the dbc file specified. This command can be used to check db connections from wrapper scripts |
| m\_db load | m\_db load dbc\_file -dml 'metadata-string'  -table tablename | Loads data to a database table or insert statement from std in |

References:

<https://www.linode.com/docs/tools-reference/tools/>

<https://www.cyberciti.biz/faq/linux-unix-appleosx-bsd-cat-command-examples/>