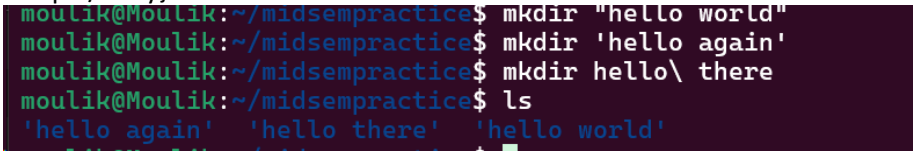


## Important commands

- Ls
  - -a include hidden
  - -l show details
  - -R show recursively
  - -t sort by time newest to oldest
  - -X sort by extension alphabetically
  - -S Biggest to smallest
  - -r reverse
- .. Parent directory
- . Own directory
- Mkdir
  - “-p” option: creates the directory only if it doesn’t exist
  - mkdir -p  
Music/{Jazz/Blues,Folk,Disco,Rock/{Gothic,Punk,Progressive},Classical/B  
aroque/Early}
  - 

```
moulik@moulik: ~/midsempractice$ mkdir "hello world"
moulik@moulik: ~/midsempractice$ mkdir 'hello again'
moulik@moulik: ~/midsempractice$ mkdir hello\ there
moulik@moulik: ~/midsempractice$ ls
'hello again' 'hello there' 'hello world'
```
- Cat
  - -n display line numbers
  - -s omit repeated output lines
- Mv
  - -i interactive prompt before overwrite
  - -f force overwrite
  - -n no overwrite
  - \*in case of multiple of f i n last is followed
- cp
  - -R -r recursive
  - -p preserve attributes
  - -f -i -n same as mv
- rm
  - -i interactive
  - -r recursively remove
  - -d same as rmdir(empty directories)
- head tail
  - -n specify number of lines from top or bottom
  - -2 -3 etc. also work similarly
- man
- clear
- which - identifies location of commands
- echo
  - -e identify stuff like \n and \t

## simple editors

- vi
- nano
- gedit

## important questions

Lab 01 UNIX  
If you have to create a folder with name "my folder 123", what all will work?

- ☐ mkdir "my folder 123"
- ☐ mkdir my\folder\123
- ☐ mkdir my\folder"123"
- ☐ mkdir "my\folder\123"

Submit

**Correct Answer**  
Right: mkdir my\folder"123"; mkdir my\folder\123, mkdir "my folder 123"  
Answer(s):

**Answer Description**  
mkdir "my\folder\123" will just put \ in the names and hence will not match what was expected. If you want to preserve space, you need to put it in "".

## Advanced UNIX commands

- REGEX regexone.com

- grep

- -E extended REGEX
- `$'\t'` for finding tab character

More metacharacters: Suppose you want to use "or" operator i.e. "|". The first command will fail, since it tries to match the same exact pattern. The second works, since by using \, you are escaping the | i.e. you will interpret it as the "or" operator. But this tends to be not clear when reading regular expressions. The best way around is the use of the third command, use of -E (has to be capital) in which case you don't need to escape special characters. Use the man page of grep to know more.

- a. `grep "cried|could" bigfile`
- b. `grep "cried\|could" bigfile`
- c. `grep -E "cried|could" bigfile`

- find

- -name name
- -type file type
- -size file size
- -mtime modify time
- -perm permissions
- -delete ???
- syntax `find (directory name)(tags to match by)`
- `find input/ -regex ".*[(\ .txt)|(\ .docx)|(\ .pdf)]"`

- wc

- "-l" number of lines.
- "-w" number of words.
- "-m" number of characters.
- "-c" number of byte

- cut: helps cut parts of a line by delimiter, byte position

- "-f" specifies field(s)
- "-b" specifies bytes(s)
- "-d" specify a delimiter that will be used instead of the default "TAB"
- --complement - Complement the selection. When using this option cut displays all bytes, characters, or fields except the selected.
- --output-delimiter - The default behavior of cut is to use the input delimiter as the output delimiter. This option allows you to specify a different output delimiter string
- eg. `cut -d " " -f 1,3 "hello abc asd qwe"` will give error
- eg. `echo "hell a b c" | cut -d " " -f 1,3 ;` will give hell b
- eg. `echo "hell a b c" | cut --output-delimiter "!" -d " " -f 1,3 ;` will give hell!b
- default input delimiter is tab and output delimiter is space

- paste: helps merge lines of files horizontally

```

moulik@moulik:~$ echo -e "abcd\nabcd\nabcd">file
moulik@moulik:~$ echo -e "abc\nabc\nabc">file1
moulik@moulik:~$ cat file1
abc
abc
abc
moulik@moulik:~$ cat file2
abcd
abcd
abcd
moulik@moulik:~$ paste file1 file2
abc      abcd
abc      abcd
abc      abcd
moulik@moulik:~$ paste -s file1 file2
abc      abc      abc
abcd     abcd     abcd
moulik@moulik:~$ paste -d "!" file1 file2
abc!abcd
abc!abcd
abc!abcd
moulik@moulik:~$

```

- sort
  - -r reverse
  - -n numerically
  - -k taking column number default delimiter is whitespaces
  - -t set delimiter
- zip : zip foo.zip foo
  - -r recursive
  - -e encrypt with password given on prompt
  - -s set splitsize if archive exceed specified size then it is split into archive1.zip archive2.zip etc.
- unzip : unzip foo.zip
  - -l list details
  - -d choose directory where you want to unzip default is .
- tar <output file> <files to compress>
  - -c: Create a new archive.
  - -v: Verbose mode
  - -f: Specify the name of the archive file
  - -x: Extract files from an archive.
  - -z: Specifies that the archive is compressed with gzip
  - -t: Lists the contents of the archive

JS regex

```

let patt=/^[a-z0-9]+@[a-z]+\.[a-z]{3}$/
if(!patt.test(x)){
    throw "Error: Invalid Email Address."
}

```