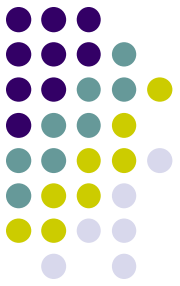


Basics of Linux / Unix

Santosh Kyadari
(Computer Centre)

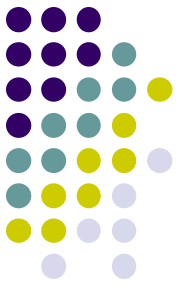
Date: 7 -11 -2019

Introduction

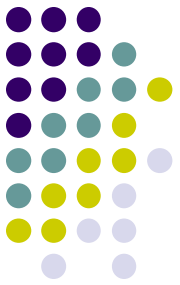


- Linus Torvalds – Creator of Linux
- Open Source Operating System
- Free Software
- Source Code Available
- Kernel can be customized to user's needs

File structure



- /root , /home/users → Home directories
- /boot → Kernel , boot loaders
- /bin , /usr/bin , /usr/local/bin → **user executables**
- /sbin, /usr/sbin → **System/Administration executables**
- /etc → **configuration files**
- /tmp → **Temporary files**
- /lib, /lib64, /usr/lib, /usr/local/lib → **shared libraries**
- /usr → **distribution packages**
- /usr/local → **Local packages**
- /var, /srv → **Variable data, server data**
- /proc , /sys → **system information**
- /media , /mnt → **mount points**
- **man hier**
- More info: http://www.comptechdoc.org/os/linux/commands/linux_crfilest.html



ls command

ls - list directory contents

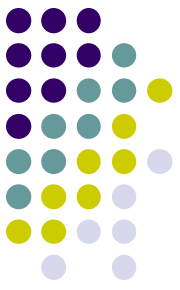
Usage : **ls** [OPTIONS] [FILE]

OPTIONS


- -l Use a long listing format
- -a Do not ignore entries starting with . (hidden files)
- -h Print sizes in human readable format (e.g., 1K 234M 2G)
- -d List directory entries instead of contents
- -R List subdirectories recursively
- -r Reverse order while sorting
- -s Print the allocated size of each file, in blocks
- -S Sort by file size
- -t Sort by modification time
- -1 List one file per line

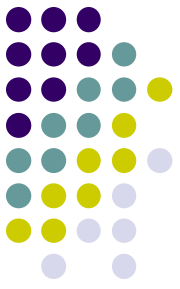
Mostly used options in ls

ls -l, ls -la, ls -1, ls -lh, ls -ltr, ls -lS



File system commands

- **pwd** - report your current directory
 - **cd** *<to where>* - change your current directory
 - **ls** *<directory>* -list contents of directory
 - **cp** *<old file>* *<new file>* - copy
 - **mv** *<old file>* *<new file>* - move (or rename)
 -  **rm** *<file>* -delete a file
 - **mkdir** *<new directory name>* -make a directory
 - **mkdir -p** /work/junk/{one,two,three,four}
 - **rmdir** *<directory>* -remove an empty directory
 - **man** *<command name>*
 - **man -k** mail
- \$ man** *command* gives you help on that command.



man command

- To display the manual / help of any command

man [OPTION] [COMMAND NAME]

man find

Some options

-k searches the pattern in all the manuals

-w displays the location of the manual page

man -w find

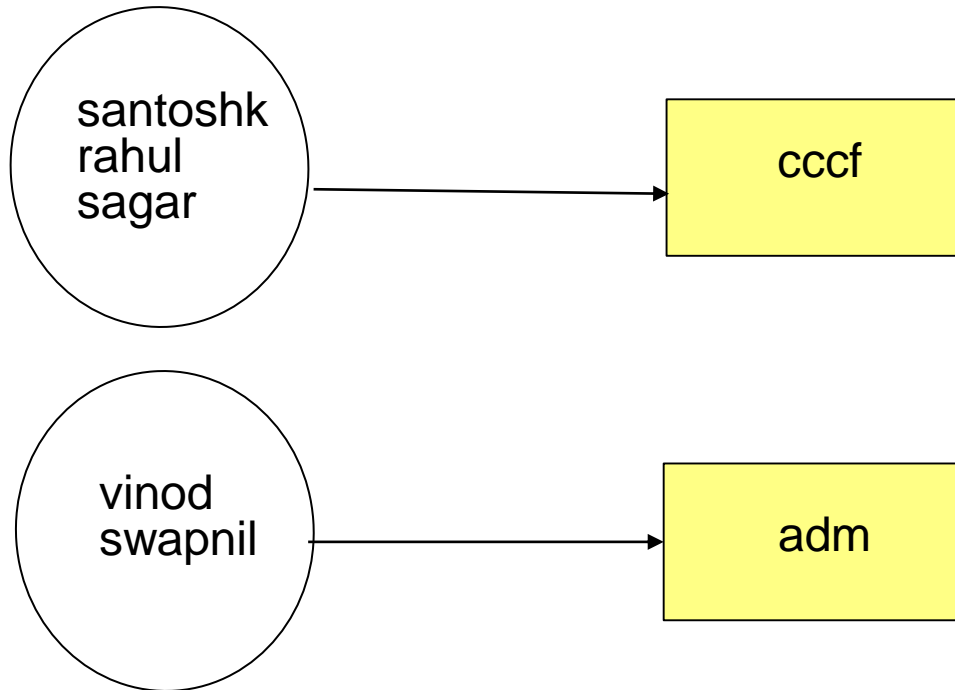


Ownership in Linux.

There are 3 kinds of users in linux : you (user i.e owner), your friends (group) and everyone else (others).

Users

Groups



\$ ls -l

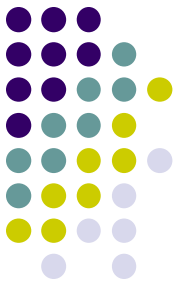
-rwxrwx-r--

1 santoshk cccf 224 Oct 14 17:57 display_time.sh

drwxrwxr-x

2 santoshk cccf 4096 Oct 14 19:19 test_dir

File permissions.



- 3 types of permissions

r - Read permissions

w - Write permissions

x - execute permissions

d - Directory

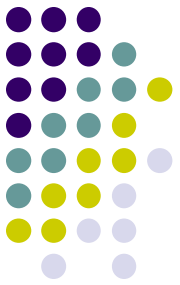
- File

`$ ls -l`

```
-rwxrw-r-- 1 santoshk cccf 224 Oct 14 17:57 display_time.sh
```

```
drwxrwxr-x 2 santoshk cccf 4096 Oct 14 19:19 test_dir
```

- For a file if x is set that user can execute the file
- For a directory if x is set that user can that user can enter in that directory.



Changing File Permissions and Ownership

- Make a file readable to your friends:

```
$ chmod 765 <filename>
```

```
7 -> 111 -> rwx
```

```
6 -> 110 -> rw-
```

```
5 -> 101 -> r-x
```

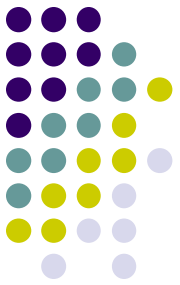
```
-rwx rw- r-x 1 santoshk cccf 224 Oct 14 17:57 <filename>
```

- Change who owns a file:

```
$ chown <user> <filename>
```

- Change to which group the file belongs:

```
$ chgrp <group> <filename>
```



cat command

- cat command allows to create files, view file, concatenate files

```
cat test1
```

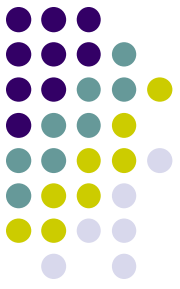
```
cat test1 test2
```

```
cat >test2 # press CTRL + D
```

```
cat test; cat test1; cat test2
```

```
cat test test2 > test1
```

```
cat test >> test1
```



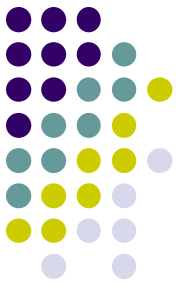
PATH: shell variable

```
$ echo $PATH
```

```
/usr/lib/qt-s.3/bin :/usr/kerberos/bin :/usr/local/bin: /bin:/usr/bin  
:/home/webteam/santoshk/bin
```

- If a program (like `ls`) is in one directory found in your path, then typing it (`~>ls <enter>`) will execute it.
- Otherwise you can type the full absolute address to execute a program (`~>/usr/bin/ls <enter>`)

Finding things in your PATH.



- Type “which *<command>*” to find the location of the program which would run when you type *<command>*.

```
$ which grep
```

```
/bin/grep
```

- If you don't remember a command name if it was grep or grepdiff, type “gre<TAB>” to get a list of commands that starts with gre.

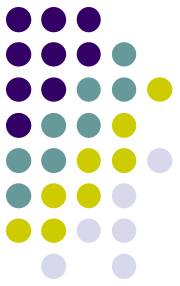
```
grefer      grep-changelog  grepjar
```

```
grep       grepdiff
```

- when all else fails, use “find” to find a file.

```
$ find <start dir> -name “*.txt”
```

Other useful pre-defined shell variables



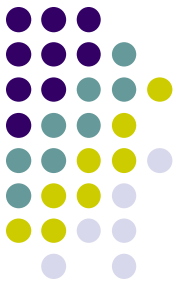
- **HOSTNAME** Name of the computer
- **HOME** Home directory of the user
- **USER** your user login
- **PWD** current directory
- **PATH** defines list of directories to search through when looking for a command to execute.

```
$ echo $HOSTNAME
```

```
cc1.tifr.res.in
```

Commands to see all the variables: **env**, **set**

touch



- Look at the full listing again:

```
⌘ ls -l .forward
```

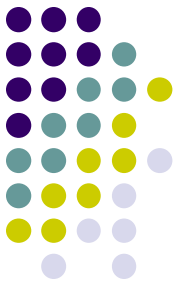
```
-rw-r--r--  1 darin  csua  23 Jan 23  2009  
.forward
```

- Each file has a date stamp of when it was modified.
- Use touch to set the timestamp to the current clock.

```
⌘ touch <filename>
```

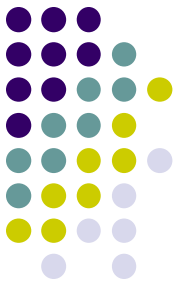
- Touch creates the file if it didn't exist.
- You can only touch a file to which you can write.

Working on multiple files



- some commands can work on many files at once:
\$ **ls file1 file2 file27**
- Use * to match any number of unknown characters
\$ **ls file***
- Use ? to match one unknown character.
\$ **ls file?**
\$ **ls file[1-2]**
\$ **ls file[13]**

Getting Recursive



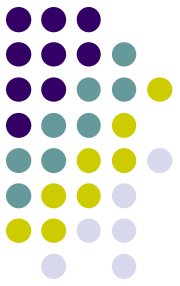
- remove a directory and its contents:

```
$ rm -r <directory> ☠
```

- copy a directory and its contents:

```
$ cp -r <directory>
```


(un)aliasing



- create shortcuts for yourself

```
$ alias ll='ls -la'
```

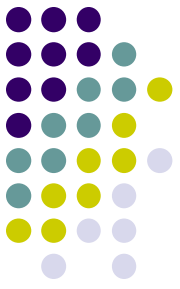
- Use alias with no arguments to discover current aliases

```
$ alias
```

```
alias rm='rm -i'
```

```
alias ll='ls -l --color=tty'
```

Type “**unalias rm**” to remove alias.



Symbolic Links

- Reference to another file or directory
- use `ln -s <old file> <second name>` to create a symbolic link to a file.

```
$ ln -s nfs.txt link.txt
```

```
$ ls -l
```

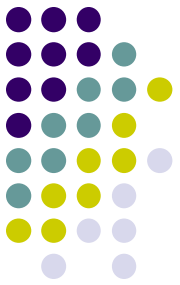
```
-rw-rw-r-- 1 santoshk santoshk 26823 Oct 14 19:01 nfs.txt
```

```
lrwxrwxrwx 1 santoshk santoshk    7 Oct 14 19:54 link.txt -> nfs.txt
```

- The first “l” tells you that it’s a symbolic link.
- Symbolic links can be used as if it were its target.

Redirecting output to a file with >

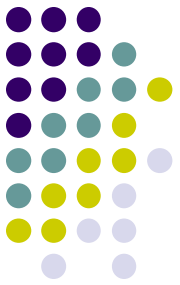
Redirecting input from a file with <



Redirection Symbols

- >file Make file the standard output
- <file Make file the standard Input
- >>file Make file the standard output, appending to it if already exists
- n>file Make file the output for file descriptor n

File Descriptor	Name	Common Abbreviation	Usual Default
0	Standard input	stdin	Keyboard
1	Standard output	stdout	Terminal
2	Standard error	stderr	Terminal



Redirecting examples

ls -l > abcd.txt Redirects output to **abcd.txt**

sort < account.txt Accepts the input from **account.txt**

ls -l santosh.txt 2> error.txt Redirects error to **error.txt**

ls -l santosh.txt 2>&1 **error.txt** Redirect output and error to **error.txt**

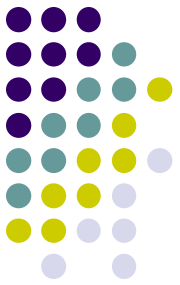
ls -l 2>&1 | tee -a log.txt

ls -l &> file

ls -l &>> test

ls -l >>log.txt 2>&1

Piping |



- Pipes take the output of the first program and feed that output into the input of the next program.
- The output of a command can be piped to another command for further processing
- Also sometimes known as “filters”.

Examples:

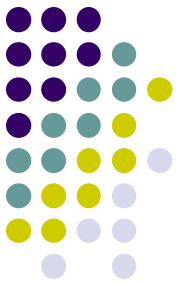
```
ls -l | wc -l
```

```
cat nfs.txt | more
```

```
last | grep “^root” | less
```

```
last | grep “^root” | cut -d -f 2 | less
```

```
grep “error” something.out | tail -1
```



Date and Time : date

- **date** command prints or sets the system date and time

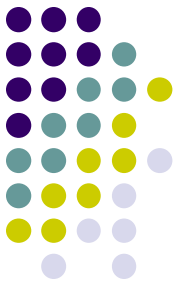
\$ date

Wed Oct 13 17:23:56 IST 2010

\$ date '+%d/%b/%Y %H:%M:%S'

13/Oct/2010 17:22:01

Pattern extraction : grep



- grep is global / regular expression / print

```
$ grep <pattern> <filename>
```

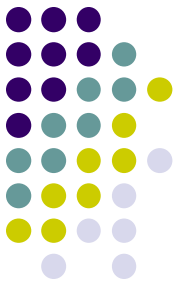
```
$ grep apple fruitlist.txt
```

```
$ grep -i apple fruitlist.txt
```

-i Ignore case

-v Invert the sense of matching

Cutting the fields in a text file



- Cut out selected fields of each line of a file

`cut [options] filename`

- Options

- `-d` Delimiter default is space “ “
- `-f` Column/ field list
- `-c` Character position list

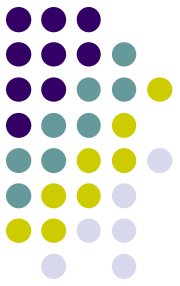
Example

`cut -f 2 -d ",“ filename` # displays second column

`cut -f 1,5 -d “:“ passwd` # displays user Id and
Full name of user in passwd file

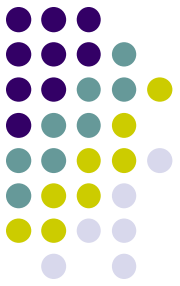
`cut -c5,15 abcd.txt` # displays characters from 1-15

shell and shell scripts.



- shell :- A shell is a piece of software that provides an interface for users of an operating system which provides access to the services of a kernel.
 - To see current shell `$ echo $SHELL`
 - To change or use different shell `$ /bin/sh` or `/bin/bash`
- shell script :- Bunch of commands you'd like to automate. You can put them on separate lines of a file. Then type "`shell_name <filename>`" to run the script.
 - `$ sh myscript.sh`
- To make a script executable without giving shell name, the script should have executable file permissions and first line of script should be `#!/<path/shell name>`
 - `$./myscript.sh` or `$ path/myscript.sh`

Simple shell script



```
#!/bin/sh
```

```
#Script to display date and time after every one second
```

```
#alias DSTAMP='date "\"+%d/%b/%Y %H:%M:%S\"'
```

```
alias DSTAMP='date'
```

```
for N in `seq 1 8`
```

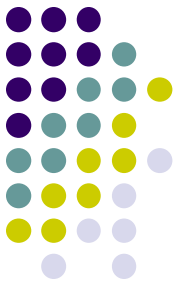
```
do
```

```
    echo "Count $N: Now Date and Time is $(DSTAMP)"
```

```
    sleep 1
```

```
done
```

Login using ssh

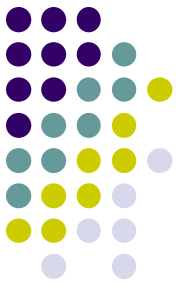


- ssh – remote login program

```
$ ssh -l santoshk cc1.tifr.res.in
```

ssh client in windows is putty. Download from
<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

Copy to remote machine : scp



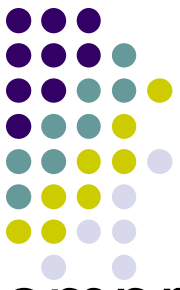
- copy local to remote

```
⌘ scp <source file> user@machine:<path>
```

- copy remote to local

```
⌘ scp user@machine:<path> <source file>
```

- p Preserves mode, time stamps
- r Recursively copy entire directories.
- v Verbose mode.



Backup using tar

-c for creation / backup & -v verbose & -z zip/compre

-x for extract / restore & -f file name

- `tar -cvf backup.tar ../workshop`
- `tar -cvzf backup.tar.gz ../workshop`

Restore using tar

- `tar -xvf backup.tar`
- `tar -xvzf backup.tar.gz`

Compiling with gcc in Linux



-o Output file name

```
gcc -o hello hello.c
```

```
ls -l
```

- -rwxr-xr-x 1 santoshk CCCF 6443 Apr 16 16:43 hello
- -rw-r--r-- 1 santoshk CCCF 75 Apr 15 14:52 hello.c

Installing packages in Linux from tar



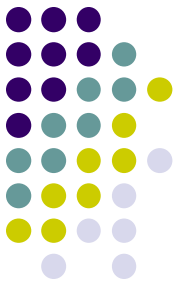
Download and extract tar file for e.g netcat.tar.gz

```
./configure --prefix=/home/santoshk/netcat
```

```
make
```

```
make install
```

Installing packages in Ubuntu and Centos



Ubuntu

apt-get install denyhosts

OR

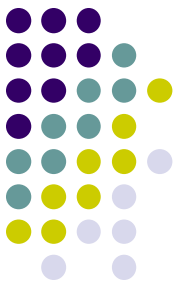
dpkg -i denyhosts_2.10-2_all.deb

Centos/Fedora/RHEL

yum install denyhosts

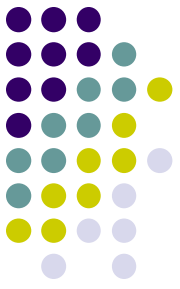
OR

rpm -ivh denyhosts-2.6-5.el6.rf.rpm



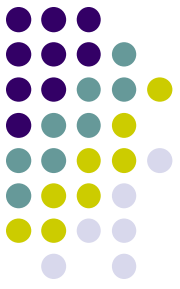
More commands

- **sort** <filename> - sort lines of text files
 - **sort -nr +0 -1** <filename> # sorts according to first field
- **uniq** <filename> - report uniq lines
 - **uniq -c** <filename> # display the uniq entries with count
- **tee** - read from standard input and write to standard output and files
 - **find / "abc*.*" 2>&1 | tee -a log.txt**
 - #finds files and displays output and error and tees to log.txt
- **tar** – backup / archiving utility
 - **tar -cvf abcd.tar /usr** #create a tar file of /usr directory
- **head** - output the first part of files
 - **head -10 abcd.txt** #displays top 10 lines of abcd.txt



More commands

- **tail** - output the last part of files
 - **tail -5** abcd.txt # displays last 5 lines of abcd.txt
 - **tail -f** maillog.log # displays continuously the new appending data.
- **cat** - concatenate files and print on the standard output
 - **cat** a.txt b.txt >>z.txt #appends a.txt and b.txt to z.txt
- **more** – view the contents of a text file one screen at a time
- **echo** - display a line of text\
- **tr** - translate or delete characters
 - **echo** “Hello world” | **tr** '[a-z]' '[A-Z]' # will display HELLO WORLD
- **expr** - Evaluate an expression
 - **expr** 5 * 2 # multiplies 5 and 2



Advance Commands

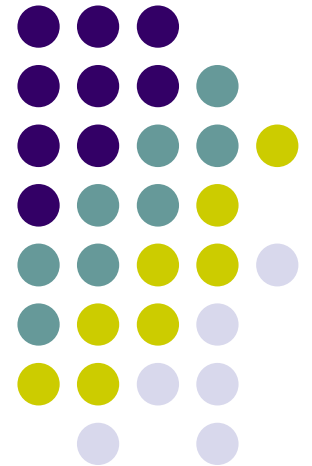
- Some of system related commands

exec, time, top, ps, su, rpm, yum, dd, find, stat, lsof, ps, xargs, chattr

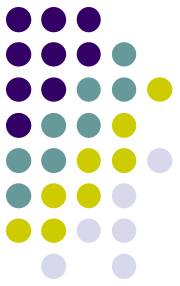
- Some of Network related commands

ping, netstat, ifconfig, ifup, ifdown, dig, nslookup, host, rsync, ftp, ssh, telnet, wget, lynx, ntpdate, whois, tcptrack

vi editor

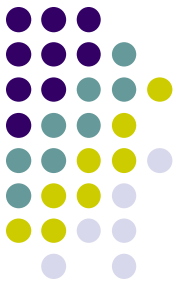


Introduction

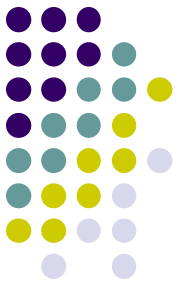


- Some of text editors are vi, nano, pico
- Original vi program was written by Bill Joy in 1976
- Use vi editor to:
 - create text files
 - edit text files
- The vi editor is not a text formatter like MS Word
- The current iteration of **vi** for Linux is called **vim**
Vi Improved

Starting vi

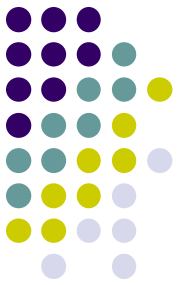


- Type **vi** `<filename>` at the shell prompt
- After pressing enter the command prompt disappears and you see tilde(**~**) characters on all the lines
- These tilde characters indicate that the line is blank



Vi modes

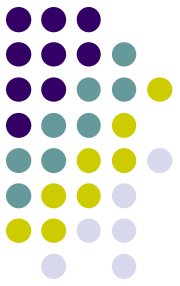
- There are two modes in vi
 - Command mode
 - Input mode
- When you start vi by default it is in command mode
- You enter the input mode through various commands
- You exit the input mode by pressing the Esc key to get back to the command mode



How to exit from vi

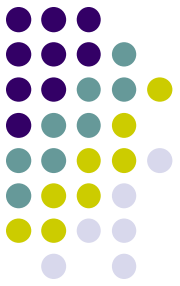
- First go to command mode
 - press **Esc** There is no harm in pressing **Esc** even if you are in command mode. Your terminal will just beep and/or or flash if you press **Esc** in command mode
- There are different ways to exit when you are in the command mode

How to exit from vi (command mode)



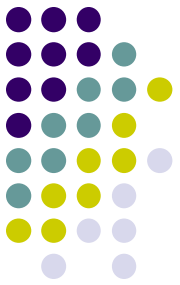
- **:q** <enter> is to exit, if you have not made any changes to the file
- **:q!** <enter> is the forced quit, it will discard the changes and quit
- **:wq** <enter> is for save and Exit
- **:x** <enter> is same as above command
- The **!** Character forces over writes, etc.
:wq!

Moving Around



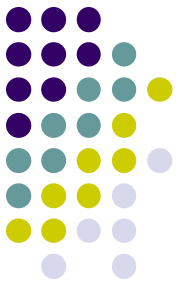
- You can move around only when you are in the command mode
- Arrow keys usually works (but may not)
- The standard keys for moving cursor are:
 - **h** - for left
 - **l** - for right
 - **j** - for down
 - **k** - for up

Moving Around



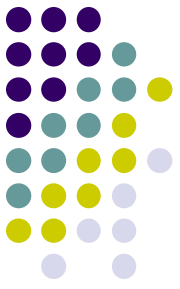
- **w** - to move one word forward
- **b** - to move one word backward
- **\$** - takes you to the end of line
- **<enter>** takes the cursor to the beginning of next line

Moving Around



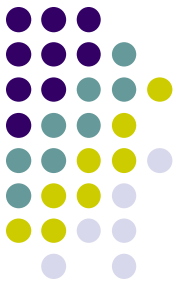
- - - (minus) moves the cursor to the first character in the current line
- **H** - takes the cursor to the beginning of the current screen(Home position)
- **L** - moves to the Lower last line
- **M** - moves to the middle line on the current screen

Moving Around



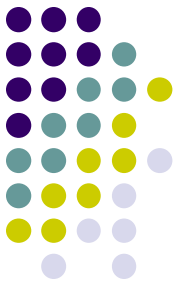
- **f** - (find) is used to move cursor to a particular character on the current line
 - For example, **fa** moves the cursor from the current position to next occurrence of 'a'
- **F** - finds in the reverse direction

Moving Around

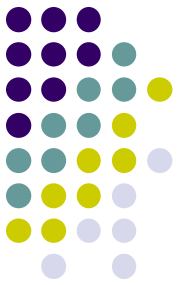


- **)** - moves cursor to the next sentence
- **}** - move the cursor to the beginning of next paragraph
- **(** - moves the cursor backward to the beginning of the current sentence
- **{** - moves the cursor backward to the beginning of the current paragraph

Moving Around



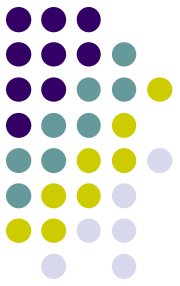
- **Control-d** scrolls the screen down (half screen)
- **Control-u** scrolls the screen up (half screen)
- **Control-f** scrolls the screen forward (full screen)
- **Control-b** scrolls the screen backward (full screen).
- **xG-** to go at x line
- **G-** takes you to bottom line of file
- **gg-** takes you to first line



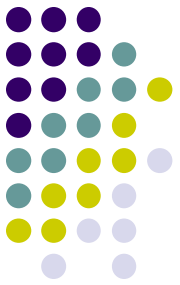
Entering text

- To enter the text in vi you should first switch to **input mode**
 - To switch to input mode there are several different commands
 - **a** - Append mode places the insertion point after the current character
 - **i** - Insert mode places the insertion point before the current character

Entering text



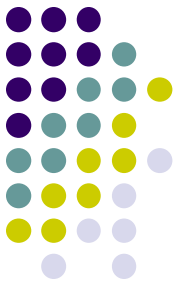
- **I** - places the insertion point at the beginning of current line
- **o** - is for open mode and places the insertion point after the current line
- **O** - places the insertion point before the current line
- **R** - starts the replace (overwrite) mode



Editing text

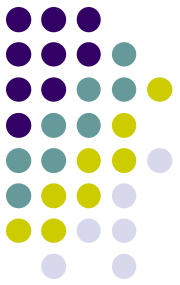
- **x** - deletes the current character
- **d** - is the delete command but pressing only d will not delete anything you need to press a second key
 - **dw** - deletes to end of word
 - **dd** - deletes the current line
 - **d0** - deletes to beginning of line

The change command



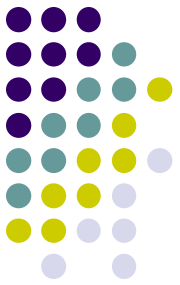
- **c** - this command deletes the text specified and changes the vi to input mode. Once finished typing you should press **<Esc>** to go back to command mode
- **cw** - Change to end of word
- **cc** - Change the current line
- There are many more options

Structure of vi command

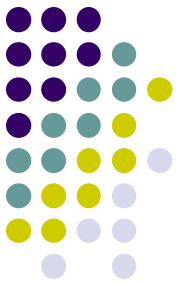


- The vi commands can be used followed by a number such as **n<command key(s)>**
 - For example **dd** deletes a line **5dd** will delete five lines.
- This applies to almost all vi commands
- This how you can accidentally insert a number of characters into your document

Undo and repeat command



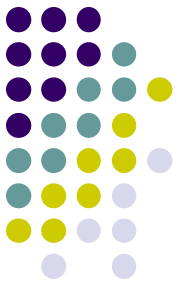
- **u** - undo the changes made by editing commands
- **.** (dot or period) repeats the last edit command



Copy, cut and paste

- **yy** - (yank) copy current line to buffer
- **nyy** - Where **n** is number of lines
- **p** - Paste the yanked lines from buffer to the line below
- **P** - Paste the yanked lines from buffer to the line above

(the paste commands will also work after the **dd** or **ndd** command)



vi Tricks

- Indent four lines: `4>>`
- Will delete the character under the cursor, and put it afterwards. In other words, it swaps the location of two characters: `xp`
- Similar to `xp`, but swapping lines: `ddp`

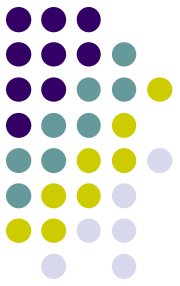
Creating a shell script using vi



- Create a directory **class**
- Change into **class**
- **vi myscript.sh**
- inside the file enter following commands

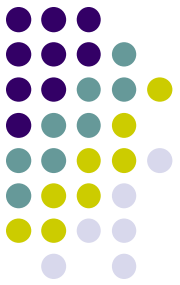
```
clear
echo "======"
echo "Hello World"
echo "======"
sleep 3
clear
echo Host is $HOSTNAME
echo User is $USER
```


Creating a shell script using vi

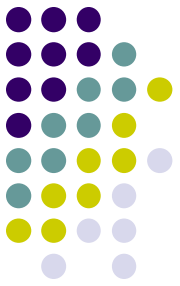


- Save the file
- Change the permissions on myscript.sh
chmod 700 myscript.sh <enter>
- Now execute myscript.sh
myscript.sh <enter>
- Did the script run?
- Why not?
 - Hint, think about absolute vs relative path
 - Type **echo \$PATH** to see your PATH variable
 - Try this **./myscript.sh** <enter>
 - The **./** mean right here in this directory!

References



- Unix shell programming -by Yashwant Kanetkar
- Unix Concepts and Applications –by Sumitabha Das
- <http://www.grymoire.com/Unix/Sed.html>
- <http://www.grymoire.com/Unix/Awk.html>
- <http://www.grymoire.com/Unix/Quote.html>
- <http://www.grymoire.com/Unix/Find.html>



Thanks