Linux Programming for Robotics

Open Terminal: alt+ ctrl + t
Close Terminal ctrl+ d

Linux File System:

- How to navigate through a Linux file system
- How to interact with a Linux files system
- How to edit files using Shell (gedit editor)
- Manage access to files (permissions)
- Create simple Linux programs (Bash Scripts)
- Manage execution of Linux programs (processes)
- How to connect to the remote computer of a robot (ssh)
- Perception with ROS

Source: \url{https://www.hostinger.in/tutorials/linux-commands}

pwd
pwd command to find out the path of the current working directory (folder) you're in.

cd
cd command is used to navigate through the Linux files and directories. It requires either the full path or the name of the directory.
cd .. (with space and two dots) to move one directory up
cd (with tilde) to go straight to the home folder
cd- (with a hyphen) to move to your previous directory

ls
ls command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.
ls -R will list all the files in the sub-directories as well
ls -a will show the hidden files
ls -al will list the files and directories with detailed information like the permissions, size, owner, etc.

cat
cat command is used to display the contents of a file on the standard on screen.
cat > filename creates a new file

cat filename1 filename2>filename3 joins two files (1 and 2) and stores the output of them in a new file (3)

cp
cp <file/folder we want to copy> <name of the new file/folder>
cp -r my_scripts/ my_scripts_copy/
cp –help
cp command to copy files from the current directory to a different directory.

mv command
mv command is used to move files and rename files.
To rename files, the Linux command is mv oldname.ext newname.ext
mv <file/folder we want to move> <destination>

mkdir
mkdir command is used to make a new directory
There are extra mkdir commands as well:
mkdir dirName
use the p (parents) option to create a directory in between two existing directories. For example,
mkdir -p Music/2020/Newfile will create the new “2020” file.

rmdir
rmdir command is used to delete a directory. However, rmdir only allows you to delete empty directories.

rm
rm command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to rmdir — use rm -r.

touch
touch command allows you to create a blank new file through the Linux command line.
Example: touch /home/newfile.txt

locate
locate command is used to locate a file, just like the search command in Windows. Use -i for case insensitive.
locate -i word1*word2 to search for a file that contains two or more words.

find
find is used to search for files and directories. The difference is, you use the find command to locate files within a given directory.
As an example, find /home/ -name notes.txt command will search for a file called notes.txt within the home directory and its sub directories.
Other variations when using the find are:

To find files in the current directory use, find . -name notes.txt
To look for directories use, \/ -type d -name notes.txt

grep
grep is used to search through all the text in each file.
To illustrate, grep blue notepad.txt will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully.

sudo
Short for “SuperUser Do”, sudo command is used to perform tasks that require administrative or root permissions.

df
df command to get a report on the system’s disk space usage, shown in percentage and kilo bytes. If you want to see the report in megabytes, type df -m.

du
du (Disk Usage) command is to check how much space a file or a directory takes. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the -h argument to the command line.

head
head command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type head -n 5 filename.ext.

tail
tail command will display the last ten lines of a text file. For example, tail -n filename.ext

diff
diff command compares the contents of two files line by line. After analyzing the files, it will output the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.
The simplest form of this command is diff file1.ext file2.ext

tar
tar command is used to archive multiple files into zip format, with compression being optional.

chmod
chmod is used to change the read, write, and execute permissions of files and directories.

chmod octal file – change the permissions of file to octal, which can be found separately for user, group, and world by adding:
4 – read (r)
2 – write (w)
1 – execute (x)
Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
chmod +x read.py give permission to file read.py for execution.

chown
chown command enables you to change or transfer the ownership of a file to the specified username. For instance, chown linuxuser2 file.ext will make linuxuser2 as the owner of the file.ext.

jobs
jobs command will display all current jobs along with their statuses. A job is basically a process that is started by the shell.

kill
kill is used to terminate unresponsive program. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.
SIGTERM (15) — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.
SIGKILL (9) — forces programs to stop immediately. Unsaved progress will be lost.
Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to kill. If you don't know the PID, simply run the command ps ux.
After knowing what signal you want to use and the PID of the program, enter the following syntax:
kill [signal option] PID.

ping
ping command is used to check your connectivity status to a server. For example, by simply entering ping google.com, the command will check whether you're able to connect to Google and also measure the response time.

wget
wget command line is used to download files from the internet. Simply type wget followed by the download link.

uname
uname command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.

top
top is equivalent to Task Manager in Windows, the top command will display a list of running processes and how much CPU each process uses. It’s very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

history
When you have been using Linux for a certain period of time, you will quickly notice that you can run hundreds of commands every day. As such, running history command is particularly useful if you want to review the commands you have entered before.

man
man gives the detail of manual instruction of the command. Example man tail will show the manual instruction of the tail command.

echo
echo command is used to move some data into a file. For example, if you want to add the text, “Hello, my name is John” into a file called name.txt, you would type echo Hello, my name is John >> name.txt

zip, unzip
Use the zip command to compress your files into a zip archive, and use the unzip command to extract the zipped files from a zip archive.

hostname
If you want to know the name of your host/network simply type hostname. Adding a -i to the end will display the IP address of your network.

useradd, userdel
Since Linux is a multi-user system, this means more than one person can interact with the same system at the same time. useradd is used to create a new user, while passwd is adding a password to that user’s account. To add a new person named John type, useradd John and then to add his password type, passwd 123456789.
To remove a user is very similar to adding a new user. To delete the users account type, userdel UserName

Use the clear command to clean out the terminal if it is getting cluttered with too many past commands.
Try the TAB button to autofill what you are typing.
Ctrl+C and Ctrl+Z are used to stop any command that is currently working. Ctrl+C will stop and terminate the command, while Ctrl+Z will simply pause the command.
If you accidental freeze your terminal by using Ctrl+S, simply undo this with the unfreeze Ctrl+Q.
Ctrl+A moves you to the beginning of the line while Ctrl+E moves you to the end.

You can run multiple commands in one single command by using the “;” to separate them. For example Command1; Command2; Command3. Or use if you only want the next command to run when the first one is successful.

Tab Completion

$ls -l Do
$clear
ctrl + a -beginning of the line
ctrl + e -End of the line
ctrl + ←
ctrl + →
ctrl + u - delete cursor to start
ctrl + k - delete cursor to end
ctrl + shift + c - copy to clipboard
ctrl + shift + v - copy from clipboard
ctrl + r - search command history
ctrl + c - cancel the command
uparrow - previous command
downarrow - next command

Type and note down the operations

$ ls
$ ls -l
long list for more details
$ ls -lh
{command}[options][argument]
$ls -lah /opt
$ls -a - list all
$ls -F list.txt
$man ls
$ls --help
$ls -l --human-readable
q to quit

apropos list - lists all those commands
$file myfile.txt - determines file type
$stat myfile.txt - display ownership
$cd Document/
$pwd - print working directory
$cd .. - previous directory
$cd ~ - goes to home folder

```
$echo Hello
$echo "Hello"
$cal
$cal 2017
$cal 12 2017
$cal -A 1 12 2021 - after one month
$cal -B 1 12 2021 - before one month
$cal -A 1 -B 1 12 2021 - after and before one month
$cal -y
$date
$date -u - universal time
$date --universal (long names preceded by --)
$history
$!1 - executes first command
$!! - executes previous command
$history -c; history -w clears history
$exit
$echo $PATH  - gives the shell script path
$which cal
$which echo
$which which
$cat 1>output.txt - deletes the data written before
```

standard input - keyboard- 0
standard output - monitor - 1
error - 2

```
$cat >>output.txt - appends to the data written before
$cat 2>>error.txt - writes the error to error.txt
$cat -k xyz 2>error.txt - writes the error to error.txt
$cat input.txt
"Hello World!"
$cat 0<input.txt
$cat <input.txt
$cat <input.txt 1>output.txt
$tty - gives dev/pts/l
$date 1>date.txt
$cat 0<date.txt --delimiter =  " " --fields 1
$date | cut --delimiter = " " --fields 1
$date | cut --delimiter = " " --fields 1 > today.txt
$date | cut  >today.txt --delimiter =  " " --fields 1
$date | tee full.txt | cut --delimiter = " " --fields 1 > today.txt
```
tee stores in the file full.txt also pipe helps to store data in today.txt
```
$date | echo "hello"
```

$date | xargs echo "hello"
$date | cut --delimiter = " " --fields 1 > today.txt
$date | cut --delimiter = " " --fields 1 | xargs echo

$rm file.txt
$cat file.txt | xargs rm
Aliases - nicknames for pipelines
$cd \texttildelow
$gedit .bash_aliases - shows hidden files

alias getdate = 'date | tee */home/asha/text.txt* | cut --delimiter = " " --fields 1 | tee /home/asha/date.txt | xargs echo hello'
save - close - open terminal
$getdates

open .bashrc_aliases
add the following lines
alias calmagic = "cal -A 1 -B 1 2021 > */home/asha/thing.txt"*
save and close

ctrl+alt+t
$echo 12 2017 | calmagic

wild cards *

ls *
ls D*
ls Do*
ls *.txt - all that ends with .txt
ls ?.txt - ? single letter.txt
ls ??.txt
ls file[1234567890].txt
ls file*.txt
ls file[0-9]/txt
ls file[A-Z].txt
ls file[0-9][0-9].txt
ls file[0-9][A-Z][a-z].txt
ls file[0-9ABC].txt
cd Desktop/
touch file.txt
touch *~/Documents/asha.txt*
echo "Hello" > hello.txt
mkdir fodername
mkdir ~/Pictures/myPics
mkdir -p myFolder

```
cd ~/Desktop/
mkdir \{jan, feb, mar\}_\{2020, 2021\}
mkdir \{jan, feb, mar\}_\{2020 .. 2025\}
touch \{jan, feb, mar\}_{2020 .. 2025\}/file\{1 ..10\}.txt
ls \{jan, feb, mar\}_\{2020 .. 2025\}/file\{1 ..10\} > output.txt
touch file\{A,B,C\}.txt
touch file\{A .. C\}.txt
```

Delete files and Folders

```
rm
rm asha.txt
rm Documents/asha.txt
touch file\{1 ..3\}.txt
rm *.txt - removes all those ends with .txt
rm file*
rm *2*
rm *.jpg
rm *\[2,3\]*
rm -r folderName
mkdir -p folder/file\{1,2,3\}
touch folder/file\{1,2,3\}/file\{1,2,3\}.txt
rm -r folder
rm -ri folder
```

Copy files and Folders

```
cp file1.txt file2.txt
rm file*
. - this folder
.. - previous folder
```

Rename files and folders

```
cd Desktop/
mv oldfile.txt newfile.txt
mv oldfolder newfolder
mv file* newfolder
mv newfolder/ ~ /Documents/
```

```
sudo apt install mlocate
locate *.conf
locate *.CONF
locate -i *.CONF
locate -i --limit 3 *.CONF
locate -S
locate -S >~/Desktop/data.txt
locate -i --limit 3 *.CONF>~/Desktop/list.txt
locate -e .com
locate -e .conf
locate --existing .conf
locate --follow .conf
locate --follow *.conf
locate --existing --follow -i --limit 5 *.conf
touch findme.txt
locate findme.txt
updatedb
man updatedb
sudo updatedb
locate findme.txt
cd ..
locate findme.txt
cd Desktop/
locate -S
locate -S > data_after.txt
locate --existing --follow findme.txt
find
find /home
find /etc
find /home/asha
cd Desktop/
mkdir asha.txt
mkdir asha
cd asha
touch asha.txt
cd ..
cd asha2
mkdir asha2
cd asha2
find
cd ..
find
cd ~/Documents/
mkdir level1
cd level1/
mkdir level2
cd level2
mkdir level3
cd level3
```

```
touch level.txt
cd ~/Documents/
find
find . -maxdepth 1
find . -maxdepth 2
find . -maxdepth 3
find . -maxdepth 4
find .
find . -type f
find . -type d
find . -type d -maxdepth 1
find . -maxdepth 1 -type d
find . -maxdepth 1 -type f
find . -name "5.txt"
find . -name "level3.txt"
find . -name "level4.txt"
find . -name "*.txt"
find . --maxdepth 3 -name "*.txt"
find . -maxdepth 3 -name "*.txt"
find . -maxdepth 2 -name "*.txt"
find . -maxdepth 2 -name "?.txt"
find . -maxdepth 2 -iname "*.TXT"
find /-type f -size +1000k
find /-type f -size +100k
find / -type f -size +100k
find / -type f -size +1000k
sudo find / -type f -size +1000k
sudo find / -type f -size +1000k |wc -l
sudo find / -type f -size +1000k | wc -l
sudo find / -type f -size +100k | wc -l
find / -type f -size +100k | wc -l
find / -type f -size +100k -size -5M
find / -type f -size -100k -size +5M
find / -type f -size -100k -o -size +5M
find / -type f -size -100k -o -size +5M | wc -l
cd Desktop/
mkdir copy
sudo find / -type  f -size +100k ssize -5M
sudo find / -type  f -size +100k size -5M
sudo find / -type  f -size +100k -size -5M
sudo find / -type  f -size +100k -size -5M |wc -l
sudo find / -type  f -size +100k -size -5M | wc -l
sudo find / -type  f -size +100k -size -5M -exec cp WHT WHERE
sudo find / -type  f -size +100k -size -5M -exec cp {} ~/Desktop/copy
sudo find / -type  f -size +100k -size -5M -exec cp {} ~/Desktop/copy  \;
sudo find / -maxdepth 3 -type  f -size +100k -size -5M -exec cp {} ~/Desktop/copy  \;
cd copy/
ls
cd ..
sudo find / -maxdepth 3 -type  f -size +100k -size -5M -ok cp {} ~/Desktop/copy  \;
touch  haystack/folder{1..500}/files{1..500}.txt
touch haystack/folder{1..500}/files{1..500}.txt
touch haystack/folder${shuf -i 1-500 -n l}/needle.txt
find haystack/ -type f -name "needle.txt"
find haystack/ -type f -name "needle.txt" -exec mv {} ~/Desktop \;


echo "Hello" >file1.txt
echo "there" >file2.txt
echo "Welcome" >file3.txt
cat file1.txt file2.txt file3.txt
cat file1.txt file2.txt file3.txt > file.txt
cat file.txt
cat file[1-3].txt > file.txt
ls
echo "abc" >>new.txt
echo "def" >>new.txt
cat new.txt
tac new.txt
cat file[1-3].txt | tac
cat file[1-3].txt | tac > reverse.txt
cat file[1-3].txt | rev
cat file[1-3].txt | rev | tac
cat file.txt | head
head file.txt | head -n 2
find | head -n 5
find | head -n 5 | tac
cat /etc/cups/cups-browsed.conf
cat /etc/cups/cups-browsed.conf | wc -l
head -n 20 /etc/cups/cups-browsed.conf | wc -l
head -n 20  /etc/cups/cups-browsed.conf
head -n 20 | cat /etc/cups/cups-browsed.conf
head file.txt | tail -n 2
tail -n 20 cat /etc/cups/cups-browsed.conf | wc -l
tail -n 20 /etc/cups/cups-browsed.conf | wc -l
tail -n 20 /etc/cups/cups-browsed.conf
head -n 1 file.txt | tail -n 1
find  | tail -n 3
find  | tail -n 3 > export.txt
cat export.txt
```

```
echo "asha" >> file.txt
echo "vihan" >> file.txt
echo "laxmi" >> file.txt
sort file.txt > sorted.txt
cat sorted.txt
sort file.txt | tac rev_sorted.txt
sort file.txt | tac > rev_sorted.txt
cat rev_sorted.txt
sort -r file.txt | less (sort the text)
sort -r numbers.txt | less (wrong output )
sort -n numbers.txt | less (n- number)
sort -nr numbers.txt | less (nr- number and reverse)
sort number.txt
sort -u number.txt | less (u- unique_
ls -l /etc/
ls -l /etc | head -n 20
ls -l /etc | head -n 20 | sort -k 5n
ls -l /etc | head -n 20 | sort -k 5nr (n number)
ls -l /etc | head -n 20 | sort -k 6hr (h human readable)
ls -l /etc | head -n 20 | sort -k 5hr
ls -lh /etc | head -n 20 | sort -k 5hr
ls -lh /etc | head -n 20 | sort -k 6M (M-month)
ls -lh /etc | head -n 20 | sort -k 6Mr (Mr-Month reverse)
ls -lh /etc | head -n 20 | sort -k 7r
ls -lh /etc | head -n 20 | sort -k 7nr
ls -lh /etc | head -n 20 | sort -k 2n
ls -lh /etc | head -n 20 | sort -k 2nr
```

```
echo "hello world" > file.txt
cat file.txt
grep e file.txt (search e)
grep -c e file.txt
wc -l file.txt
grep -i hello file.txt (i case insensitive)
grep -i a file.txt
grep -ic a file.txt
grep -i "hello world" file.txt
grep -iv a file.txt (v search without a)
grep -iv o file.txt
grep -iv h file.txt
grep -cv h file.txt
cat file.txt
grep -i "e" file1.txt file.txt
grep -ci "e" file1.txt file.txt
mkdir file
mv hello.txt hello
mv file.txt file
ls file/ | grep file.txt
ls -F
ls -F /etc | grep -v
ls -F /etc | grep -v / >files.txt
ls -F /etc | grep -v / sort -r > files.txt
ls -F /etc
ls -F /etc | grep  magic
ls -F /etc | grep -v magic
ls -F /etc | grep -v magic >files.txt
ls -F /etc | grep -v magic| sort -r >files.txt
ls -F /etc | grep -v/ | sort -r >files.txt
ls -F /etc | grep -v / | sort -r >files.txt
ls -F /etc | grep -v /
man -k print | grep files
```

Archive and Compressing

```
ls
ls -lh
tar -cvf archive.tar file1.txt files.txt (c for create)
ls
ls -l | grep .tar
file archive.tar
mv archive.tar new.odt
file new.odt
mv new.odt new.tar
ls
tar -tf new.tar
rm file?.txt
ls
tar -xvf new.tar (x for extract)
ls
gzip new.tar
ls
ls -lh
gunzip new.tar.gz
```

```
ls
ls -lh
bzip2 new.tar
ls
ls -lh
bunzip2 new.tar.bz2
ls
zip other.zip file1.txt files.txt
ls
unzip other.zip
rm other.zip new.tar
ls
tar -cvf new.tar *.txt
ls
tar -cvzf new.tar.gz *.txt
ls
file new.tar.gz
tar -cvjf new.tar.bz2 *.txt
ls
rm *.txt | tar -xvf new.tar
rm *.txt
ls
tar -xvzf new.tar.gz
tar -xvjf new.tar.bz2
```

Introduction to Editors

Bash Scripting

```
nano our_script.sh
file our_script.sh
nano our_script.sh
which bash
nano our_script.sh
```

```
#!/bin/bash

echo "Welcome to bash scripting!!"

mkdir ~/Desktop/files
cd ~/Desktop/files
touch file{1..100}.txt
ls -lh ~/Desktop/files > ~/Desktop/file.log
```

```
file our_script.sh
```

```
nano our_script.sh
which python3
nano our_script.sh
bash our_script.sh
nano our_script.sh

nano backup.sh
which bash
nano backup.sh

#!/bin/bash
tar -cvzf backup.tar.gz ~/{Pictures} 2>/dev/null

cd ~
mkdir bin
mv ~/Desktop/backup.sh /bin
mv ~/Desktop/backup.sh ~/bin/
cd ~/bin/
ls
chmod +x backup.sh
nano backup.sh
ls
bash backup.sh
ls
cd ..
nano .bashrc
echo $PATH
backup
nano .bashrc
Add the foolowing line at the end
PATH="$PATH:$HOME/bin"

crontab -e

# 20 11 1 1(JUN) 0(SUN)
  * *  *   *   *     echo "Hello World" >> ~/Desktop/hello.txt

cd Desktop/
ls
cat hello.txt

crontab -e

# m h  dom mon dow   command
# 20 11 1 1(JUN) 0(SUN)
#  * *  *   *   *     echo "Hello World" >> ~/Desktop/hello.txt


#   0,15,30,45 * * * * echo "Hello World" >> ~/Desktop/hello.txt

    */15 * */3 * * echo "Hello World"
    59   23  * JAN,DEC SUN echo "we can"


cd bin
nano backup.sh

#!/bin/bash

tar -cvzf ~/backup/backup.tar.gz ~/{Desktop,Pictures,Videos} 2>/dev/null

date >> ~/backup/backup.log


ls backup/
backup
bash backup.sh

crontab -e

#    */15 * */3 * * echo "Hello World"
#    59   23  * JAN,DEC SUN echo "we can"

* * * * * bash ~/bin/backup.sh

cat backup.log
uname -o
www.GNU.org
```

Download, install, compile from GNU.org

https://ftp.gnu.org/gnu/coreutils/

```
cd /Documents
cd ~/Documents
ls
file coreutils-9.0.tar.xz
tar -xJf coreutils-9.0.tar.xz
```

```
ls
cd coreutils-9.0/
ls
cd src
ls
nano ls.c

add after main() {
printf("Hello\n");
ls | less
sudo apt-get install gcc
cd ..
bash configure
ls
sudo apt-get install make
ls
make
sudo make install

reverting back
cd ~/Documents/coreutils-9.0/src
nano ls.c
cd ..
make && sudo make install
www.ubuntu.com
lsb_release -a
uname -m
https://packages.ubuntu.com/focal/
apt-cache search docx
apt-cache search docx | grep text
apt-cache show docx2txt
apt-cache search "web server" | less
apt-cache show apache2
apt-cache search textt
apt-cache show apache2
cd /var/lib/apt/lists
ls
ls | less
ls | less -N
cd ~
apt-cache show gcc
cd /var/lib/apt/lists/
ls | less -N
nano archive.ubuntu.com_ubuntu_dists_focal_main_binary-amd64_Packages
```

Visual Studio

Open terminal
Install Visual Studio using the command

```
$sudo snap install --classic code
```

Refer \url{https://linuxhint.com/install_use_vs_code_ubuntu/} for more details.

Install extensions
1. Python
2. XML
3. Terminal