RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

HARDWORK DISCIPLINE

REC

RAJALAKSHMI
ENGINEERING COLLEGE

Laboratory Record Note Book

Name: . . . .

Year / Branch / Section: . . . .

University Register No. : . . . .

College Roll No. : . . . .

Semester: . . . .

Academic Year: . . . .

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name:____

Academic Year:_________________ Semester :_______ Branch :______

Register No:

Certified that this is the bonafide record of work done by the above student

in the CS19441 – Operating Systems *Laboratory during the year 2023-2024*

Signature of Faculty in-charge

Submitted for the Practical Examination held on ....

Internal Examiner   External Examiner

INDEX

Roll.No: Name:

Year : Branch : Section :

Ex No: 1 a

Date:

INSTALLATION AND CONFIGURATION OF LINUX

Aim:

To install and configure Linux operating system in a Virtual Machine.

Installation/Configuration Steps:

1. Install the required packages for virtualization

   ```
   dnf install xen virt-manager qemu libvirt
   ```

2. Configure xend to start up on boot

   ```
   systemctl enable virt-manager.service
   ```

3. Reboot the machine

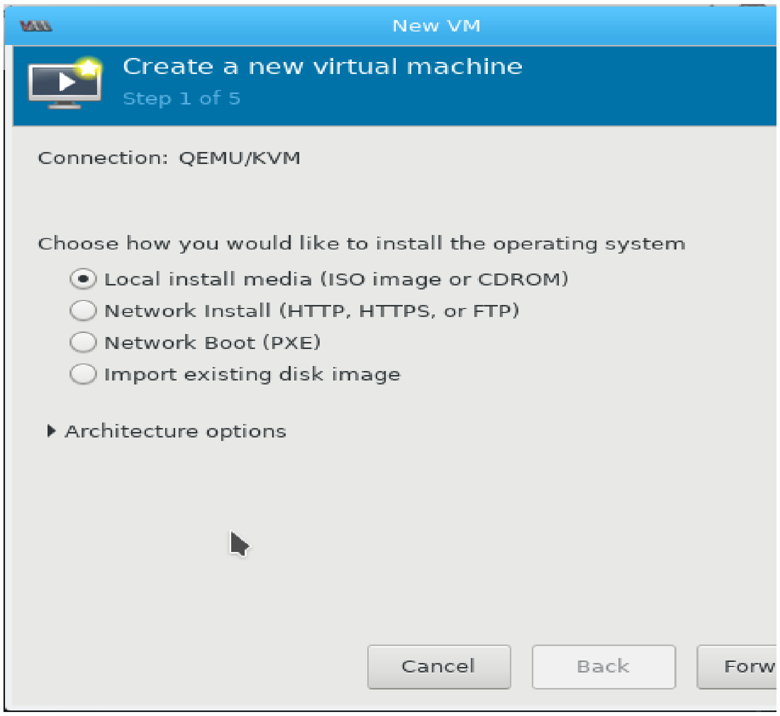
   ```
   Reboot
   ```

4. Create Virtual machine by first running virt-manager

   ```
   virt-manager &
   ```

5. Click on File and then click to connect to localhost
6. In the base menu, right click on the localhost(QEMU) to create a new VM
7. Select Linux ISO image
8. Choose puppy-linux.iso then kernel version
9. Select CPU and RAM limits
10. Create default disk image to 8 GB
11. Click finish for creating the new VM with PuppyLinux

Output:



New VM

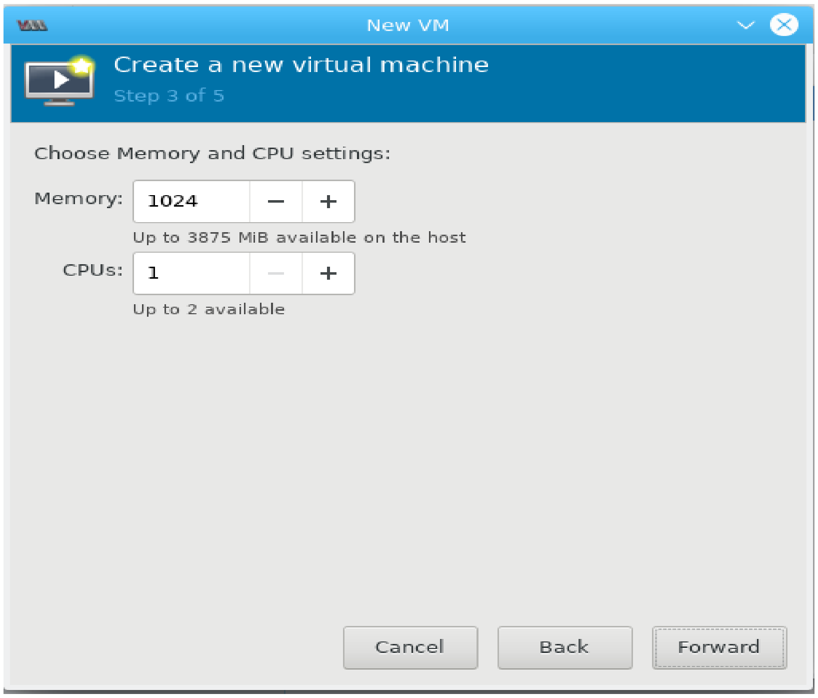
Create a new virtual machine

Step 1 of 5

Connection: QEMU/KVM

Choose how you would like to install the operating system

- ● Local install media (ISO image or CDROM)
- ○ Network Install (HTTP, HTTPS, or FTP)
- ○ Network Boot (PXE)
- ○ Import existing disk image



Create a new virtual machine

Step 3 of 5

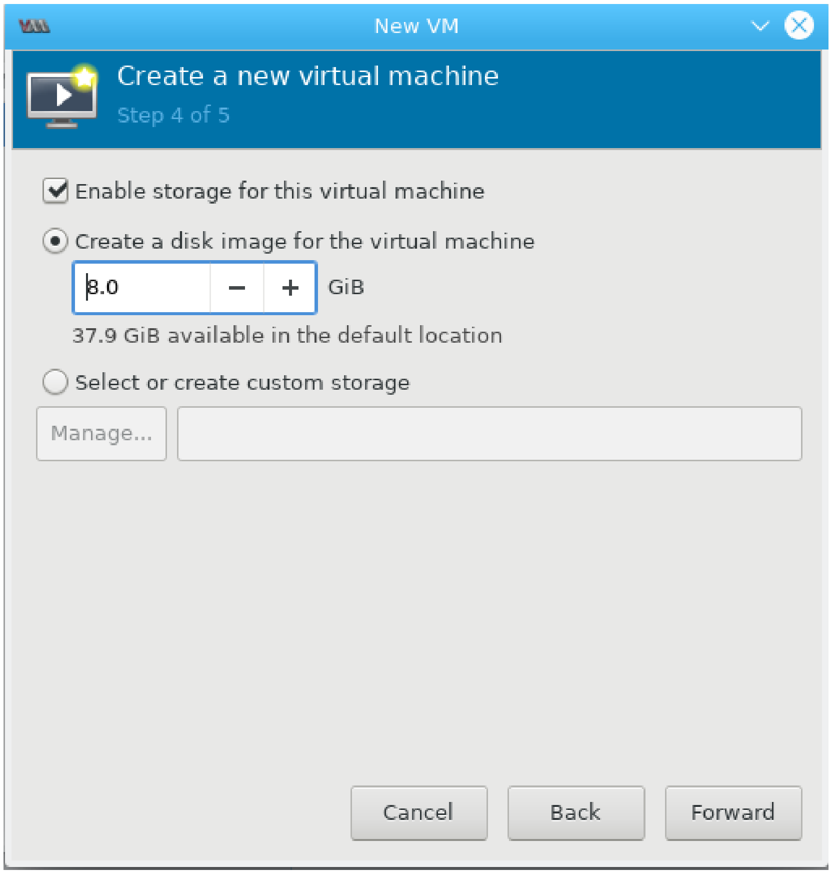
Choose Memory and CPU settings:

Memory: 1024

Up to 3875 MiB available on the host

CPUs: 1

Up to 2 available



Create a new virtual machine

Step 4 of 5

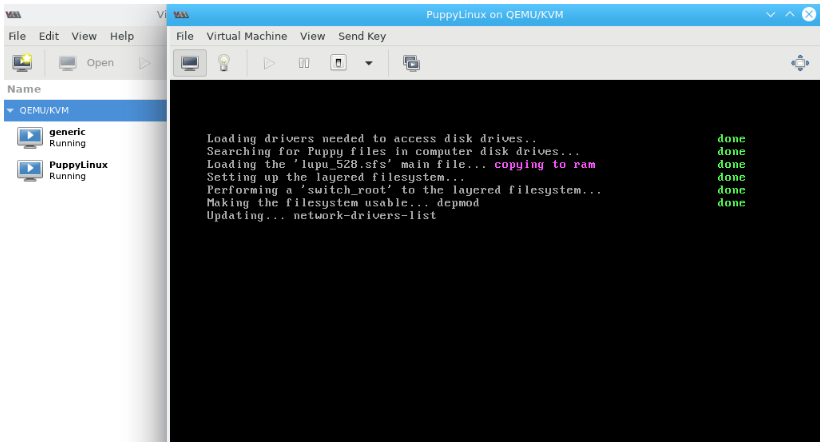
☑ Enable storage for this virtual machine

● Create a disk image for the virtual machine

8.0 GiB

37.9 GiB available in the default location

○ Select or create custom storage



```
Loading drivers needed to access disk drives..                            done
Searching for Puppy files in computer disk drives...                      done
Loading the 'lupu_528.sfs' main file... copying to ram                    done
Setting up the layered filesystem...                                      done
Performing a 'switch_root' to the layered filesystem...                   done
Making the filesystem usable... depmod                                    done
Updating... network-drivers-list
```

RESULT

Ex No: 1 b

Date:

BASIC LINUX COMMANDS

1.1 GENERAL PURPOSE COMMANDS

1. The ‘date’ command:

The date command display the current date with day of week, month, day, time (24 hours clock) and the year.

SYNTAX: $ date

The date command can also be used with following format.

Format	Purpose	Example
+ %m	To display only month	$ date + %m
+ %h	To display month name	$ date + %h
+ %d	To display day of month	$ date + %d
+ %y	To display last two digits of the year	$ date + %y
+ %H	To display Hours	$ date + %H
+ %M	To display Minutes	$ date + %M
+ %S	To display Seconds	$ date + %S

OUTPUT

2. The echo’command:

The echo command is used to print the message on the screen.

SYNTAX: $ echo

EXAMPLE: $ echo “God is Great”

OUTPUT

3. The ‘cal’ command:

The cal command displays the specified month or year calendar.

SYNTAX: $ cal [month] [year]

EXAMPLE: $ cal Jan 2012

OUTPUT

4. The ‘bc’ command:

Unix offers an online calculator and can be invoked by the command bc.

SYNTAX: $ bc

EXAMPLE: bc –l

16/4

5/2

OUTPUT

5. The ‘who’ command

The who command is used to display the data about all the users who are currently logged into the system.

SYNTAX: $ who

OUTPUT

6. The ‘who am i’ command

The who am i command displays data about login details of the user.

SYNTAX: $ who am i

OUTPUT

7. The ‘id’ command

The id command displays the numerical value corresponding to your login.

SYNTAX: $ id

OUTPUT

8. The ‘tty’ command

The tty (teletype) command is used to know the terminal name that we are using.

SYNTAX: $ tty

OUTPUT

9. The ‘clear’ command

The clear command is used to clear the screen of your terminal.

SYNTAX: $ clear

OUTPUT

10. The ‘man’ command

The man command gives you complete access to the Unix commands.

SYNTAX: $ man [command]

OUTPUT

11. The ‘ps’ command

The ps command is used to the process currently alive in the machine with the 'ps' (process status) command, which displays information about process that are alive when you run the command. 'ps;' produces a snapshot of machine activity.

SYNTAX: $ ps

EXAMPLE:

```
$ ps
$ ps –e
$ps  –aux
```

OUTPUT

12. The ‘uname’ command

The uname command is used to display relevant details about the operating system on the standard output.

-m -> Displays the machine id (i.e., name of the system hardware)

-n -> Displays the name of the network node. (host name)

-r -> Displays the release number of the operating system.

-s -> Displays the name of the operating system (i.e.. system name)

-v -> Displays the version of the operating system.

-a -> Displays the details of all the above five options.

SYNTAX: $ uname [option]

EXAMPLE: $ uname –a

OUTPUT

1.2 DIRECTORY COMMANDS

1. The ‘pwd’ command:

The pwd (print working directory) command displays the current working directory.

SYNTAX: $ pwd

OUTPUT

2. The ‘mkdir’ command:

The mkdir is used to create an empty directory in a disk.

SYNTAX: $ mkdir dirname

EXAMPLE: $ mkdir receee

OUTPUT

3. The ‘rmdir’ command:

The rmdir is used to remove a directory from the disk. Before removing a directory, the directory must be empty (no files and directories).

SYNTAX: $ rmdir dirname

EXAMPLE: $ rmdir receee

OUTPUT

4. The ‘cd’ command:

The cd command is used to move from one directory to another.

SYNTAX: $ cd dirname

EXAMPLE: $ cd receee

OUTPUT

5. The ‘ls’ command:

The ls command displays the list of files in the current working directory.

SYNTAX: $ ls

EXAMPLE: $ ls

$ ls –l

$ ls –a

OUTPUT

1.3 FILE HANDLING COMMANDS

1. The ‘cat’ command:

The cat command is used to create a file.

SYNTAX: $ cat > filename

EXAMPLE: $ cat > rec

OUTPUT

2. The ‘Display contents of a file’ command:

The cat command is also used to view the contents of a specified file.

SYNTAX: $ cat filename

OUTPUT

3. The ‘cp’ command:

The cp command is used to copy the contents of one file to another and copies the file from one place to another.

SYNTAX: $ cp oldfile newfile

EXAMPLE: $ cp cse ece

OUTPUT

4. The ‘rm’ command:

The rm command is used to remove or erase an existing file

SYNTAX: $ rm filename

```
EXAMPLE:    $ rm rec
            $ rm –f  rec
```

Use option –fr to delete recursively the contents of the directory and its subdirectories.

OUTPUT

5. The ‘mv’ command:

The mv command is used to move a file from one place to another. It removes a specified file from its original location and places it in specified location.

SYNTAX: $ mv oldfile newfile

EXAMPLE: $ mv cse eee

OUTPUT

6. The ‘file’ command:

The file command is used to determine the type of file.

SYNTAX: $ file filename

EXAMPLE: $ file receee

OUTPUT

7. The ‘wc’ command:

The wc command is used to count the number of words, lines and characters in a file.

SYNTAX: $ wc filename

EXAMPLE: $ wc receee

OUTPUT

8. The ‘Directing output to a file’ command:

The ls command lists the files on the terminal (screen). Using the redirection operator ‘>’ we can send the output to file instead of showing it on the screen.

SYNTAX: $ ls > filename

EXAMPLE: $ ls > cseeee

OUTPUT

9. The ‘pipes’ command:

The Unix allows us to connect two commands together using these pipes. A pipe (|) is an mechanism by which the output of one command can be channeled into the input of another command.

SYNTAX: $ command1 | command2

EXAMPLE: $ who | wc –l

10. The ‘tee’ command:

While using pipes, we have not seen any output from a command that gets piped into another command. To save the output, which is produced in the middle of a pipe, the tee command is very useful.

SYNTAX: $ command | tee filename

EXAMPLE: $ who | tee sample | wc -l

The ‘Metacharacters of unix’ command:

Metacharacters are special characters that are at higher and abstract level compared to most of other characters in Unix. The shell understands and interprets these metacharacters in a special way.

* - Specifies number of characters

?- Specifies a single character

[ ]- used to match a whole set of file names at a command line.

! – Used to Specify

EXAMPLE:

$ ls r** - Displays all the files whose name begins with ‘r’

$ ls ?kkk - Displays the files which are having ‘kkk’, from the second characters irrespective of the first character.

$ ls [a-m] – Lists the files whose names begins alphabets from ‘a’ to ‘m’

$ ls [!a-m] – Lists all files other than files whose names begins alphabets from ‘a’ to ‘m’

11. The ‘File permissions’ command:

File permission is the way of controlling the accessibility of file for each of three users namely Users, Groups and Others.

There are three types of file permissions are available, they are

r-read
w-write
x-execute

The permissions for each file can be divided into three parts of three bits each.

First three bits	Owner of the file
Next three bits	Group to which owner of the file belongs
Last three bits	Others

EXAMPLE: $ ls college

```
-rwxr-xr--     1      Lak    std     1525   jan10  12:10  college
```

Where,

-rwx The file is readable, writable and executable by the owner of the file.

Lak Specifies Owner of the file.

r-x Indicates the absence of the write permission by the Group owner of the file.

Std Is the Group Owner of the file.

r-- Indicates read permissions for others.

12. The ‘chmod’ command:

The chmod command is used to set the read, write and execute permissions for all categories of users for file.

SYNTAX: $ chmod category operation permission file

Category	Operation	permission
u-users	+ assign	r-read
g-group	-Remove	w-write
o-others	= assign absolutely	x-execute
a-all		

EXAMPLE:

$ chmod u –wx college

Removes write & execute permission for users for ‘college’ file.

$ chmod u +rw, g+rw college

Assigns read & write permission for users and groups for ‘college’ file.

$ chmod g=wx college

Assigns absolute permission for groups of all read, write and execute permissions for ‘college’ file.

13. The ‘Octal Notations’ command:

The file permissions can be changed using octal notations also. The octal notations for file permission are

Read permission	4
Write permission	2
Execute permission	1

EXAMPLE:

$ chmod 761 college

Assigns all permission to the owner, read and write permissions to the group and only executable permission to the others for ‘college’ file.

1.4 GROUPING COMMANDS

1. The ‘semicolon’ command:

The semicolon(;) command is used to separate multiple commands at the command line.

SYNTAX: $ command1;command2;command3… .... ;commandn

EXAMPLE: $ who;date

2. The ‘&&’ operator:

The ‘&&’ operator signifies the logical AND operation in between two or more valid Unix commands.It means that only if the first command is successfully executed, then the next command will executed.

SYNTAX: $ command1 && command && command3…..................&&commandn

EXAMPLE: $ who && date

3. The ‘||’ operator:

The ‘||’ operator signifies the logical OR operation in between two or more valid Unix commands.It means, that only if the first command will happen to be un successfully,it will continue to execute next commands.

SYNTAX: $ command1 || command || command3................... ||commandn

EXAMPLE: $ who || date

1.5 FILTERS

1. The head filter

It displays the first ten lines of a file.

SYNTAX: $ head filename

EXAMPLE: $ head college	Display the top ten lines.
$ head -5 college	Display the top five lines.

2. The tail filter

It displays ten lines of a file from the end of the file.

SYNTAX: $ tail filename

EXAMPLE: $ tail college	Display the last ten lines.
$tail -5 college	Display the last five lines.

3. The more filter:

The pg command shows the file page by page.

SYNTAX: $ ls –l | more

4. The ‘grep’ command:

This command is used to search for a particular pattern from a file or from the standard input and display those lines on the standard output. “Grep” stands for “global search for regular expression.”

SYNTAX: $ grep [pattern] [file_name]

EXAMPLE: $ cat > student

```
Arun cse
Ram ece
Kani cse
$ grep “cse” student
 Arun cse
Kani cse
```

5. The ‘sort’ command:

The sort command is used to sort the contents of a file. The sort command reports only to the screen, the actual file remains unchanged.

SYNTAX: $ sort filename

EXAMPLE: $ sort college

OPTIONS:

Command	Purpose
Sort –r college	Sorts and displays the file contents in reverse order
Sort –c college	Check if the file is sorted
Sort –n college	Sorts numerically
Sort –m college	Sorts numerically in reverse order
Sort –u college	Remove duplicate records
Sort –l college	Skip the column with +1 (one) option.Sorts according to second column

6. The ‘nl’ command:

The nl filter adds lines numbers to a file and it displays the file and not provides access to edit but simply displays the contents on the screen.

SYNTAX: $ nl filename

EXAMPLE: $ nl college

7. The ‘cut’ command:

We can select specified fields from a line of text using cut command.

SYNTAX: $ cut -c filename

EXAMPLE: $ cut -c college

OPTION:

-c – Option cut on the specified character position from each line.

1.5 OTHER ESSENTIAL COMMANDS

1. **free**

Display amount of free and used physical and swapped memory system.

synopsis-   free [options]

<u>example</u>

```
[root@localhost ~]# free -t

            total       used        free       shared    buff/cache    available

Mem:       4044380     605464     2045080     148820     1393836      3226708

Swap:      2621436        0       2621436

Total:     6665816     605464     4666516
```

2. **top**

It provides a dynamic real-time view of processes in the system.

synopsis-   top [options]

<u>example</u>

```
[root@localhost ~]# top

top - 08:07:28 up 24 min,  2 users,  load average: 0.01, 0.06, 0.23

Tasks: 211 total,   1 running, 210 sleeping,   0 stopped,   0 zombie

%Cpu(s):  0.8 us,  0.3 sy,  0.0 ni, 98.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st

KiB Mem :  4044380 total,  2052960 free,   600452 used,  1390968 buff/cache

KiB Swap:  2621436 total,  2621436 free,        0 used.  3234820 avail Mem

      PID USER  PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND

      1105 root      20   0  175008  75700  51264 S   1.7                      1.9   0:20.46   Xorg

      2529 root      20   0   80444   32640  24796 S   1.0                      0.8   0:02.47   gnome-term
```

3. **ps**

It reports the snapshot of current processes

synopsis-   ps [options]

<u>example</u>

```
[root@localhost ~]# ps -e

      PID      TTY         TIME           CMD

      1          ?        00:00:03            systemd

      2          ?        00:00:00            kthreadd

      3          ?        00:00:00            ksoftirqd/0
```

4. **vmstat**

It reports virtual memory statistics

synopsis-   vmstat [options]

<u>example</u>

```
[root@localhost ~]# vmstat

procs ------------memory----------- – –swap-  -     ·io----- -system-  -------cpu------
```

```
r  b   swpd     free        buff  cache        si   so     bi    bo   in   cs  us sy id wa st

0  0      0  1879368   1604 1487116     0    0    64     7   72  140  1  0 97  1  0
```

5. **df**

It displays the amount of disk space available in file-system.

Synopsis-   df [options]

<u>example</u>

```
[root@localhost ~]# df

Filesystem              1K-blocks     Used            Available          Use%               Mounted on

devtmpfs                  2010800        0            2010800            0%                 /dev
tmpfs                     2022188      148            2022040            1%                 /dev/shm
tmpfs                     2022188     1404            2020784            1%                 /run
/dev/sda6                 487652     168276           289680           37%                 /boot
```

6. ping

It is used verify that a device can communicate with another on network. PING stands for Packet Internet Groper.

synopsis-   ping [options]

```
[root@localhost ~]# ping 172.16.4.1

PING 172.16.4.1 (172.16.4.1) 56(84) bytes of data.
64 bytes from 172.16.4.1: icmp_seq=1 ttl=64 time=0.328 ms
64 bytes from 172.16.4.1: icmp_seq=2 ttl=64 time=0.228 ms
64 bytes from 172.16.4.1: icmp_seq=3 ttl=64 time=0.264 ms
64 bytes from 172.16.4.1: icmp_seq=4 ttl=64 time=0.312 ms
^C
--- 172.16.4.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.228/0.283/0.328/0.039 ms
```

7. ifconfig

It is used configure network interface.

synopsis-   ifconfig [options]

example

```
[root@localhost ~]# ifconfig

enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.6.102  netmask 255.255.252.0  broadcast 172.16.7.255
        inet6 fe80::4a0f:cfff:fe6d:6057  prefixlen 64  scopeid 0x20<link>
        ether 48:0f:cf:6d:60:57  txqueuelen 1000  (Ethernet)
        RX packets 23216  bytes 2483338 (2.3 MiB)
        RX errors 0  dropped 5  overruns 0  frame 0
        TX packets 1077  bytes 107740 (105.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

8. traceroute

It tracks the route the packet takes to reach the destination.

synopsis-   traceroute [options]

example

```
[root@localhost ~]# traceroute www.rajalakshmi.org
traceroute to www.rajalakshmi.org (220.227.30.51), 30 hops max, 60 byte packets
1  gateway (172.16.4.1)  0.299 ms  0.297 ms  0.327 ms
2  220.225.219.38 (220.225.219.38)  6.185 ms  6.203 ms  6.189 ms
```

RESULT

Ex no: 2

Date:

System Monitoring Using Shell Script

Aim:

To write a Shellscript to to display system information.

Program:

```
#!/bin/bash

# Sample script written for Part 4 of the RHCE series

# This script will return the following set of system

information: # -Hostname information:

echo -e "\e[31;43m***** HOSTNAME INFORMATION *****\e[0m"

hostnamectl

echo ""

# -File system disk space usage:

echo -e "\e[31;43m***** FILE SYSTEM DISK SPACE USAGE

*****\e[0m" df -h

echo ""

# -Free and used memory in the system:

echo -e "\e[31;43m ***** FREE AND USED MEMORY

*****\e[0m" free

echo ""

# -System uptime and load:

echo -e "\e[31;43m***** SYSTEM UPTIME AND LOAD *****\e[0m"

uptime

echo ""
```

```
# -Logged-in users:

echo -e "\e[31;43m***** CURRENTLY LOGGED-IN USERS

*****\e[0m" who

echo ""

# -Top 5 processes as far as memory usage is concerned

echo -e "\e[31;43m***** TOP 5 MEMORY-CONSUMING PROCESSES

*****\e[0m" ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6

echo ""

echo -e "\e[1;32mDone.\e[0m"
```

Output:

Ex. No.: 3 a

Date:

EMPLOYEE AVERAGE PAY

Aim:

To find out the average pay of all employees whose salary is more than 6000 and no. of days worked is more than 4.

Algorithm:

1. Create a flat file emp.dat for employees with their name, salary per day and number of days worked and save it.
2. Create an awk script emp.awk
3. For each employee record do
   a. If Salary is greater than 6000 and number of days worked is more than 4then print name and salary earned
   b. Compute total pay of employee
4. Print the total number of employees satisfying the criteria and their average pay.

Program Code:

Output:

Ex. No.: 3 b

Date:

RESULTS OF EXAMINATION

Aim:

To print the pass/fail status of a student in a class.

Algorithm:

```
1. Read the data from file
2. Get a data from each column
3. Compare the all subject marks column
    a. If marks less than 45 then print Fail
    b. else print Pass
```

Program Code:

OUTPUT

Ex. No.: 4
Date:

SIGNAL CATCHING

Aim:

To write a C program to catch signals used in Linux.

Algorithm:

1. The program is initialized for catching interrupt signal(SIGINT).
2. If Cntrl+C is pressed within 3 seconds then my_handler is called
3. my_handler routine displays the signal that was caught.
4. If no interrupt received then PART-II is executed.
5. In PART-II,Cntrl+C is ignored till 3 seconds then it goes to PART-III.
6. In PART-III, the default action takes place.

Program Code:

```
// signals.c
#include <signal.h>
#include <stdio.h>
void my_handler (int sig);      /* function prototype */

int main()
{
        struct sigaction my_action;

        /* Part I: Catch SIGINT */
        my_action.sa_handler = my_handler;
        my_action.sa_flags = SA_RESTART;
        sigaction (SIGINT, &my_action, NULL);
        printf ("Catching SIGINT\n");
        sleep (3);
        printf (" No SIGINT within 3 seconds\n");

        /* Part II: Ignore SIGINT */
        my_action.sa_handler = SIG_IGN;
        my_action.sa_flags = SA_RESTART;
        sigaction (SIGINT, &my_action, NULL);
        printf ("Ignoring SIGINT\n");
        sleep (3);
        printf (" Sleep is over\n");


        /* Part III: Default action for SIGINT */
        my_action.sa_handler = SIG_DFL;
        my_action.sa_flags = SA_RESTART;
        sigaction (SIGINT, &my_action, NULL);
        sleep (3);
        printf ("No SIGINT within 3 seconds\n");
}

void my_handler (int sig)
{
  printf (" \t I got SIGINT, number %d\n", sig);
  exit(0);
}
```

Output:

Ex no: 5

Date:

SYSTEM CALL TRACING

Aim:

To write a C program and trace system calls used and print the same in ascending order using shell script.

Algorithm:

1. Create a C program with an output statement helloworld.
2. Compile and trace system calls while executing the executable file.
3. The output of the system calls trace is put in hellotrace file.
4. Shellscript to read the contents of hellotrace file and print only system call name as output.

Program Code:

Output:

Ex. No.: 6

Date:

IPC USING SHARED MEMORY

Aim:

To write a C program to do Inter Process Communication (IPC) using shared memory between sender process and receiver process.

Algorithm:

SENDER

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Write a string to the shared memory segment using sprintf
5. Set delay using sleep
6. Detach shared memory segment using shmdt

RECEIVER

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Print the shared memory contents sent by the sender process.
5. Detach shared memory segment using shmdt

Program Code:

Output:

Ex. No.: 7 a
Date:

FIRST COME FIRST SERVE

Aim:

To implement First-come First- serve(FCFS) scheduling technique

Algorithm:

1. Get the number of processes from the user.
2. Read the process name and burst time.
3. Calculate the total process time.
4. Calculate the total waiting time and total turnaround time for each process
5. Display the process name & burst time for each process.
6. Display the total waiting time, average waiting time, turnaround time

Program Code:

Output:

Ex. No.: 7b
Date:

SHORTEST JOB FIRST

Aim:

To implement the Shortest Job First(SJF) scheduling technique

Algorithm:

1. Declare the structure and its elements.
2. Get number of processes as input from the user.
3. Read the process name, arrival time and burst time
4. Initialize waiting time, turnaround time & flag of read processes to zero.
5. Sort based on burst time of all processes in ascending order
6. Calculate the waiting time and turnaround time for each process.
7. Calculate the average waiting time and average turnaround time.
8. Display the results.

Program Code:

Output:

Ex. No.: 7 c
Date:

PRIORITY SCHEDULING

Aim:

To implement priority scheduling technique

Algorithm:

1. Get the number of processes from the user.
2. Read the process name, burst time and priority of process.
3. Sort based on burst time of all processes in ascending order based priority
4. Calculate the total waiting time and total turnaround time for each process
5. Display the process name & burst time for each process.
6. Display the total waiting time, average waiting time, turnaround time

Program Code:

Output:

Ex. No.: 7d
Date

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

```
1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array rem_bt[] to keep track of remaining burst time of processes which is initially
   copy of bt[] (burst times array)
5. Create another array wt[] to store waiting times of processes. Initialize this array as 0.
6. Initialize time : t = 0
7. Keep traversing the all processes while all processes are not done. Do following for i'th
process if it is    not done yet.
        a- If rem_bt[i] > quantum
                (i)  t = t + quantum
                (ii) bt_rem[i] -= quantum;
       b- Else // Last cycle for this process
               (i)  t = t + bt_rem[i];
               (ii) wt[i] = t - bt[i]
               (iii) bt_rem[i] = 0; // This process is over
8. Calculate the waiting time and turnaround time for each process.
9. Calculate the average waiting time and average turnaround time.
10. Display the results.
```

Program Code:

Output:

Ex. No.: 8
Date:

PRODUCER CONSUMER USING SEMAPHORES

Aim:

To write a program to implement solution to producer consumer problem using semaphores.

Algorithm:

1. Initialize semaphore empty, full and mutex.
2. Create two threads- producer thread and consumer thread.
3. Wait for target thread termination.
4. Call sem_wait on empty semaphore followed by mutex semaphore before entry into critical section.
5. Produce/Consumer the item in critical section.
6. Call sem_post on mutex semaphore followed by full semaphore before exiting critical section.
7. Allow the other thread to enter its critical section.
8. Terminate after looping ten times in producer and consumer threads each.

Program Code:

Output:

Ex. No.: 9
Date:

DEADLOCK AVOIDANCE

Aim:

To find out a safe sequence using Banker's algorithm for deadlock avoidance.

Algorithm:

```
1. Initialize work=available and finish[i]=false for all values of i
2. Find an i such that both:
   finish[i]=false and $Need_i$ <= work
3. If no such i exists go to step 6
4. Compute work=work+$allocation_i$
5. Assign finish[i] to true and go to step 2
6. If finish[i]==true for all i, then print safe sequence
7. Else print there is no safe sequence
```

Program Code:

Output:

Ex. No.: 10 a
Date:

BEST FIT

Aim:

To implement Best Fit memory allocation technique using Python.

Algorithm:

1. Input memory blocks and processes with sizes
2. Initialize all memory blocks as free.
3. Start by picking each process and find the minimum block size that can be assigned to current process
4. If found then assign it to the current process.
5. If not found then leave that process and keep checking the further processes.

Program Code:

Output:

Ex. No.: 10 b
Date:

FIRST FIT

Aim:

To write a C program for implementation memory allocation methods for fixed partition using first fit.

Algorithm:

1. Define the max as 25.
2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max].
3: Get the number of blocks,files,size of the blocks using for loop.
4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
5: Check highest

Program Code:

Output:

Ex. No.: 11 a
Date:

FIFO PAGE REPLACEMENT

Aim:

To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

Algorithm:

1. Declare the size with respect to page length
2. Check the need of replacement from the page to memory
3. Check the need of replacement from old page to new page in memory
4. Form a queue to hold all pages
5. Insert the page require memory into the queue
6. Check for bad replacement and page fault
7. Get the number of processes to be inserted
8. Display the values

Program Code:

Output:

Ex. No.: 11 b
Date:

LRU

Aim:

To write a c program to implement LRU page replacement algorithm.

Algorithm:

1: Start the process
2: Declare the size
3: Get the number of pages to be inserted
4: Get the value
5: Declare counter and stack
6: Select the least recently used page by counter value
7: Stack them according the selection.
8: Display the values
9: Stop the process

Program Code:

Output

Ex. No.: 12
Date:

CUSTOMIZATION OF LINUX KERNEL

Aim:

To download the vanilla Linux kernel from repository and customize to our requirements.

Customization Steps:

1. Download the vanilla kernel from www.kernel.org

2. Switch to root user using the command

```
[root@localhost os]#su
```

3. Use dnf to install kernel-devel package

```
[root@localhost os]#dnf install kernel-devel
```

4. Install gcc development tools

```
[root@localhost os]#dnf group install "Development Tools"
```

5. Install additional software packages

```
[root@localhost os]#dnf install ncurses-devel bison flex
                          elfutils-libelf-devel  openssl-devel
```

6. Copy the downloaded kernel source to /usr/src/kernels

```
[root@localhost os]#cp linux-5.0.0.tar.xz /usr/src/kernels
```

7. Go to kernel source directory

```
[root@localhost os]#cd /usr/src/kernels
```

8. Extract the downloaded vanilla kernel

```
[root@localhost os]#unxz linux-5.0.2.tar.xz
[root@localhost os]#tar xvf linux-5.0.2.tar
```

9. Remove all old configuration files

```
[root@localhost os]#make mrproper
```

10. Configure the Kernel

```
[root@localhost os]#make menuconfig
```

11. Build the Kernel(For faster build use –j 2 option)

```
[root@localhost os]#make all
```

12. Remove all temporary files

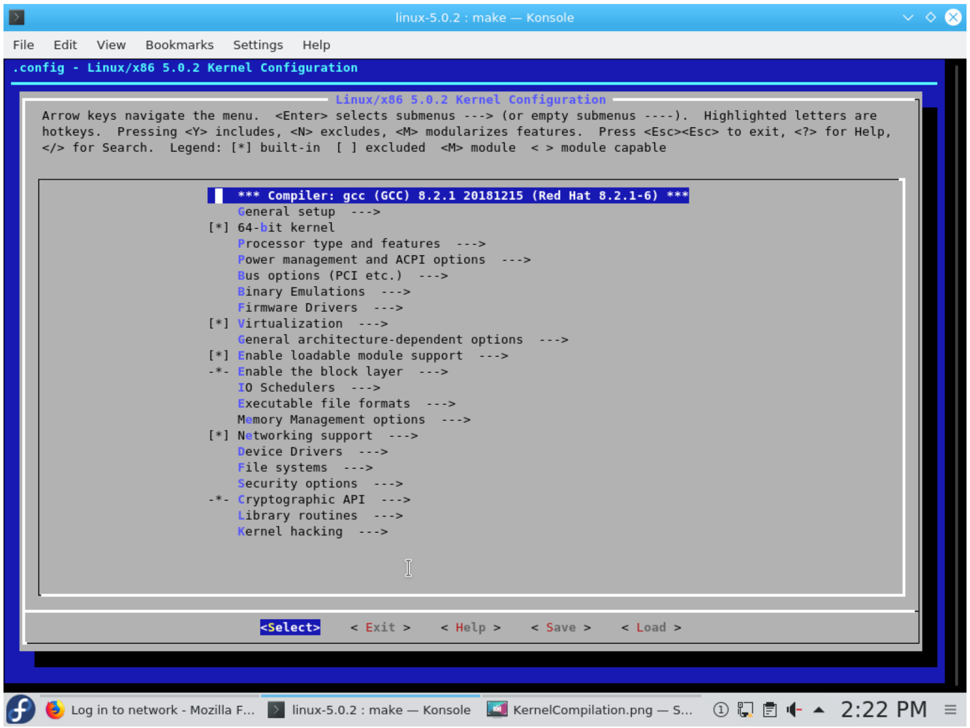
```
[root@localhost os]#make clean
```

13. Install Kernel and its modules

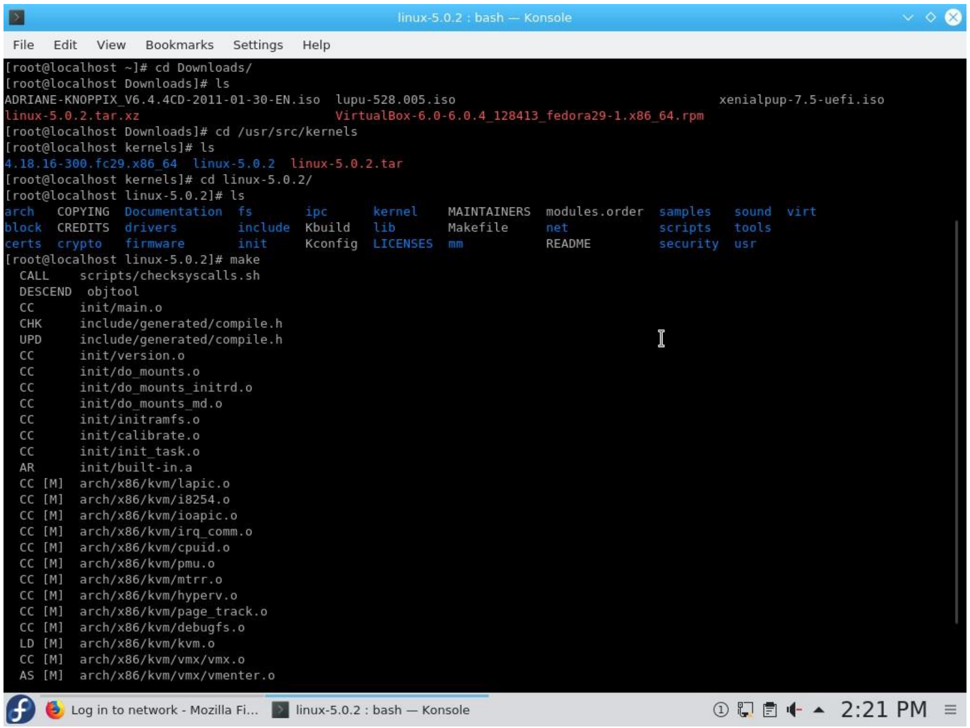
```
[root@localhost os]#make modules_install
[root@localhost os]#make install
```

14. Reboot the system

Output:


linux-5.0.2 : make — Konsole
File
Edit
View
Bookmarks
Settings
Help
Log in to network - Mozilla F...
linux-5.0.2 : make — Konsole
KernelCompilation.png — S...
2:22 PM

```
.config - Linux/x86 5.0.2 Kernel Configuration
                              Linux/x86 5.0.2 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                      *** Compiler: gcc (GCC) 8.2.1 20181215 (Red Hat 8.2.1-6) ***
                          General setup  --->
                      [*] 64-bit kernel
                          Processor type and features  --->
                          Power management and ACPI options  --->
                          Bus options (PCI etc.)  --->
                          Binary Emulations  --->
                          Firmware Drivers  --->
                      [*] Virtualization  --->
                          General architecture-dependent options  --->
                      [*] Enable loadable module support  --->
                      -*- Enable the block layer  --->
                          IO Schedulers  --->
                          Executable file formats  --->
                          Memory Management options  --->
                      [*] Networking support  --->
                          Device Drivers  --->
                          File systems  --->
                          Security options  --->
                      -*- Cryptographic API  --->
                          Library routines  --->
                          Kernel hacking  --->

                      <Select>    < Exit >    < Help >    < Save >    < Load >
```



linux-5.0.2 : bash — Konsole
File Edit View Bookmarks Settings Help
Log in to network - Mozilla Fi...
linux-5.0.2 : bash — Konsole
2:21 PM

```
[root@localhost ~]# cd Downloads/
[root@localhost Downloads]# ls
ADRIANE-KNOPPIX_V6.4.4CD-2011-01-30-EN.iso  lupu-528.005.iso                                  xenialpup-7.5-uefi.iso
linux-5.0.2.tar.xz                          VirtualBox-6.0-6.0.4_128413_fedora29-1.x86_64.rpm
[root@localhost Downloads]# cd /usr/src/kernels
[root@localhost kernels]# ls
4.18.16-300.fc29.x86_64  linux-5.0.2  linux-5.0.2.tar
[root@localhost kernels]# cd linux-5.0.2/
[root@localhost linux-5.0.2]# ls
arch   COPYING  Documentation  fs       ipc      kernel    MAINTAINERS  modules.order  samples   sound  virt
block  CREDITS  drivers        include  Kbuild   lib       Makefile     net            scripts   tools
certs  crypto   firmware       init     Kconfig  LICENSES  mm           README         security  usr
[root@localhost linux-5.0.2]# make
  CALL    scripts/checksyscalls.sh
  DESCEND  objtool
  CC      init/main.o
  CHK     include/generated/compile.h
  UPD     include/generated/compile.h
  CC      init/version.o
  CC      init/do_mounts.o
  CC      init/do_mounts_initrd.o
  CC      init/do_mounts_md.o
  CC      init/initramfs.o
  CC      init/calibrate.o
  CC      init/init_task.o
  AR      init/built-in.a
  CC [M]  arch/x86/kvm/lapic.o
  CC [M]  arch/x86/kvm/i8254.o
  CC [M]  arch/x86/kvm/ioapic.o
  CC [M]  arch/x86/kvm/irq_comm.o
  CC [M]  arch/x86/kvm/cpuid.o
  CC [M]  arch/x86/kvm/pmu.o
  CC [M]  arch/x86/kvm/mtrr.o
  CC [M]  arch/x86/kvm/hyperv.o
  CC [M]  arch/x86/kvm/page_track.o
  CC [M]  arch/x86/kvm/debugfs.o
  LD [M]  arch/x86/kvm/kvm.o
  CC [M]  arch/x86/kvm/vmx/vmx.o
  AS [M]  arch/x86/kvm/vmx/vmenter.o
```

RESULT

Ex. No: 13
Date:

DEVELOP A SIMPLE KLM

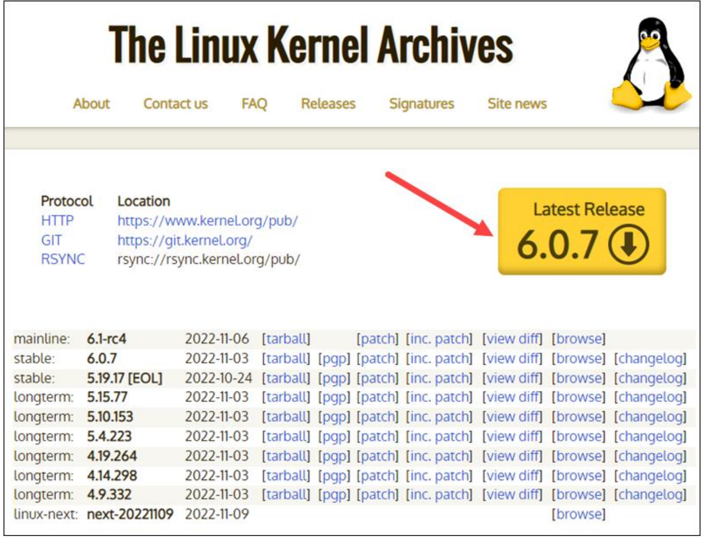
Aim:

To build a Linux Kernel from Scratch

Steps:

Step 1: Download the Source Code

1. Visit the official kernel website and download the latest kernel version. The downloaded file contains a compressed source code.



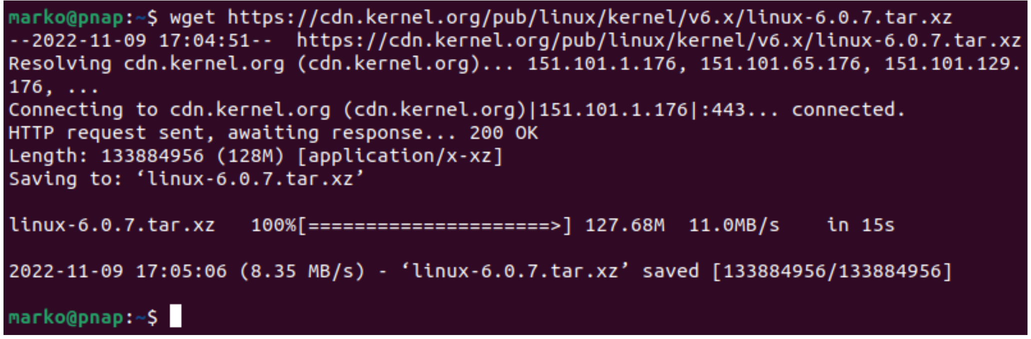
Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

mainline:	6.1-rc4	2022-11-06	[tarball]		[patch]	[inc. patch]	[view diff]	[browse]	
stable:	6.0.7	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable:	5.19.17 [EOL]	2022-10-24	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.15.77	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.10.153	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	5.4.223	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.19.264	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.298	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.9.332	2022-11-03	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20221109	2022-11-09						[browse]	

2. Open the terminal and use the wget command to download the Linux kernel source code:

   wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz

   The output shows the “saved” message when the download completes.



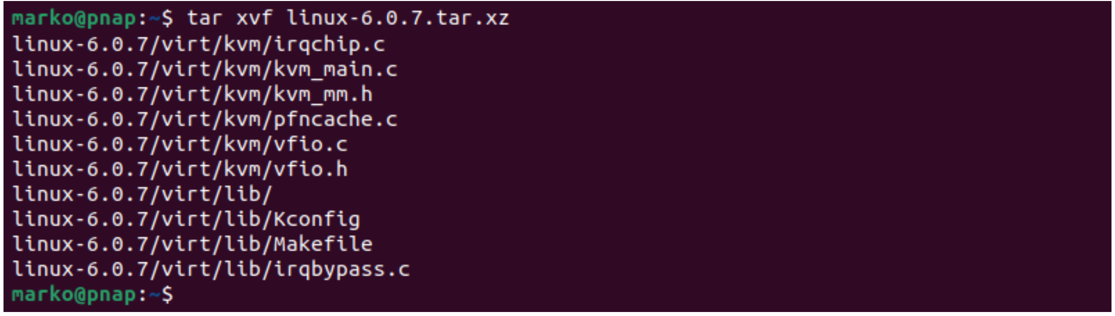
```
marko@pnap:~$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
--2022-11-09 17:04:51--  https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151.101.129.
176, ...
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.1.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 133884956 (128M) [application/x-xz]
Saving to: ‘linux-6.0.7.tar.xz’

linux-6.0.7.tar.xz   100%[====================>] 127.68M  11.0MB/s    in 15s

2022-11-09 17:05:06 (8.35 MB/s) - ‘linux-6.0.7.tar.xz’ saved [133884956/133884956]

marko@pnap:~$ █
```

Step 2: Extract the Source Code

When the file is ready, run the tar command to extract the source code:

tar xvf linux-6.0.7.tar.xz

The output displays the extracted kernel source code:



```
marko@pnap:~$ tar xvf linux-6.0.7.tar.xz
linux-6.0.7/virt/kvm/irqchip.c
linux-6.0.7/virt/kvm/kvm_main.c
linux-6.0.7/virt/kvm/kvm_mm.h
linux-6.0.7/virt/kvm/pfncache.c
linux-6.0.7/virt/kvm/vfio.c
linux-6.0.7/virt/kvm/vfio.h
linux-6.0.7/virt/lib/
linux-6.0.7/virt/lib/Kconfig
linux-6.0.7/virt/lib/Makefile
linux-6.0.7/virt/lib/irqbypass.c
marko@pnap:~$
```

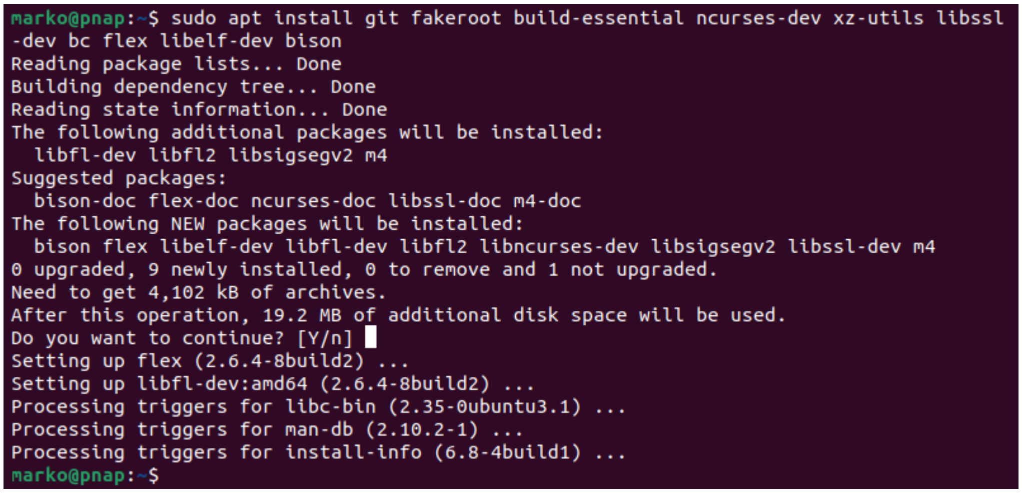
Step 3: Install Required Packages

Install additional packages before building a kernel. To do so, run this command:

sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison

The command we used above installs the following packages:

Package	**Package description**
git	Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes.
fakeroot	Creates the fake root environment.
build-essential	Installs development tools such as C, C++, gcc, and g++.
ncurses-dev	Provides API for the text-based terminals.
xz-utils	Provides fast file compression and file decompression.
libssl-dev	Supports SSL and TSL that encrypt data and make the internet connection secure.
bc (Basic Calculator)	Supports the interactive execution of statements.
flex (Fast Lexical Analyzer Generator)	Generates lexical analyzers that convert characters into tokens.
libelf-dev	Issues a shared library for managing ELF files (executable files, core dumps and object code)
bison	Converts grammar description to a C program.



```
marko@pnap:~$ sudo apt install git fakeroot build-essential ncurses-dev xz-utils libssl
-dev bc flex libelf-dev bison
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2 libsigsegv2 m4
Suggested packages:
  bison-doc flex-doc ncurses-doc libssl-doc m4-doc
The following NEW packages will be installed:
  bison flex libelf-dev libfl-dev libfl2 libncurses-dev libsigsegv2 libssl-dev m4
0 upgraded, 9 newly installed, 0 to remove and 1 not upgraded.
Need to get 4,102 kB of archives.
After this operation, 19.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] 
Setting up flex (2.6.4-8build2) ...
Setting up libfl-dev:amd64 (2.6.4-8build2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
marko@pnap:~$
```

Step 4: Configure Kernel

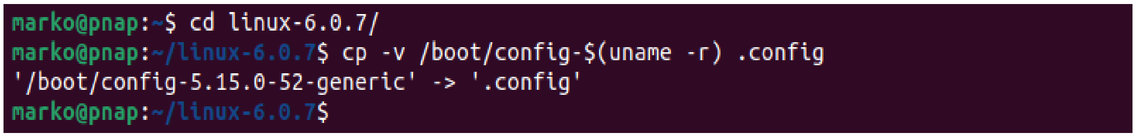
The Linux kernel source code comes with the default configuration. However, you can adjust it to your needs. To do so, follow the steps below:

1. Navigate to the linux-6.0.7 directory using the cd command:

cd linux-6.0.7

2. Copy the existing Linux config file using the cp command:

cp -v /boot/config-$(uname -r) .config

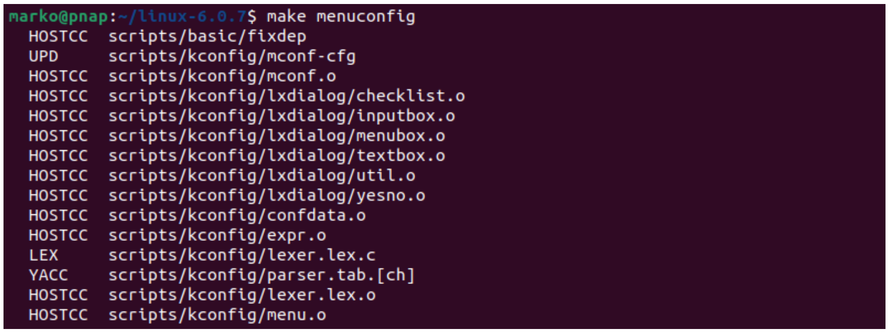


```
marko@pnap:~$ cd linux-6.0.7/
marko@pnap:~/linux-6.0.7$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-52-generic' -> '.config'
marko@pnap:~/linux-6.0.7$
```

3. To make changes to the configuration file, run the make command:

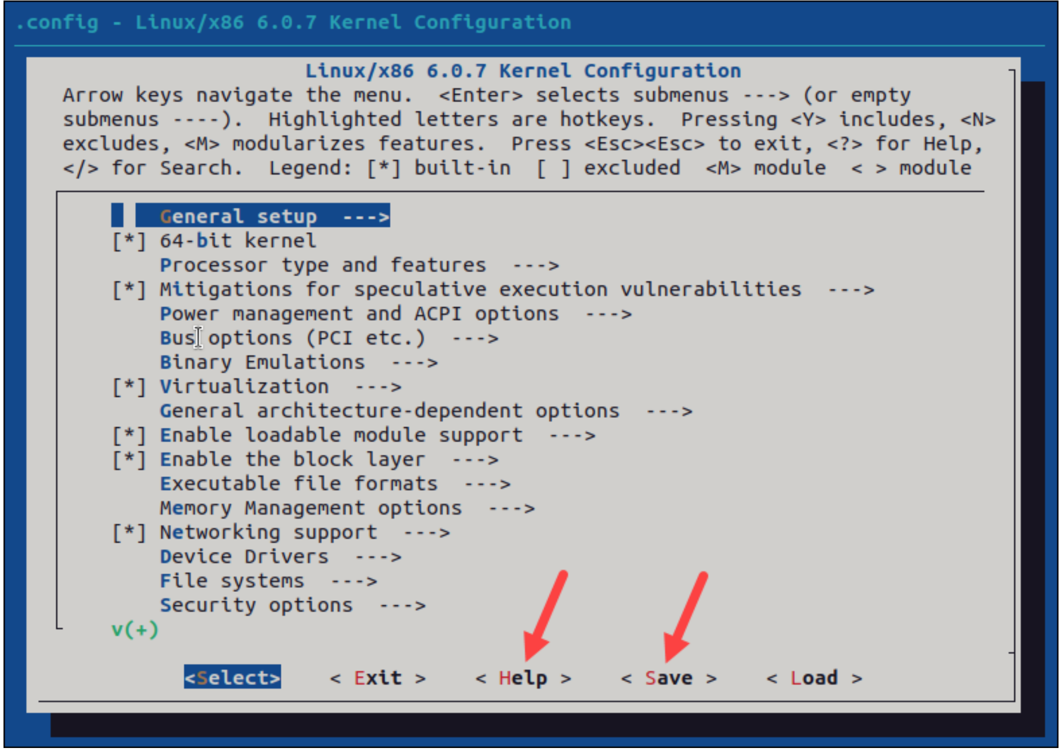
make menuconfig

The command launches several scripts that open the configuration menu:



```
marko@pnap:~/linux-6.0.7$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
```

The configuration menu includes options such as firmware, file system, network, and memory settings. Use the arrows to make a selection or choose Help to learn more about the options. When you finish making the changes, select Save, and then exit the menu.



```
.config - Linux/x86 6.0.7 Kernel Configuration

                    Linux/x86 6.0.7 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N>
  excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module

          General setup  --->
      [*] 64-bit kernel
          Processor type and features  --->
      [*] Mitigations for speculative execution vulnerabilities  --->
          Power management and ACPI options  --->
          Bus options (PCI etc.)  --->
          Binary Emulations  --->
      [*] Virtualization  --->
          General architecture-dependent options  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          Executable file formats  --->
          Memory Management options  --->
      [*] Networking support  --->
          Device Drivers  --->
          File systems  --->
          Security options  --->
      v(+)

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

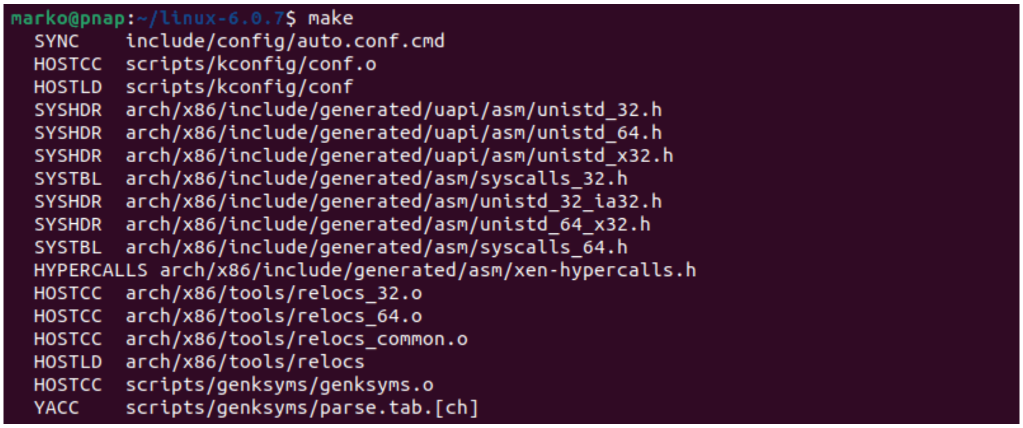
Step 5: Build the Kernel

1. Start building the kernel by running the following command:

make

The process of building and compiling the Linux kernel takes some time to complete.

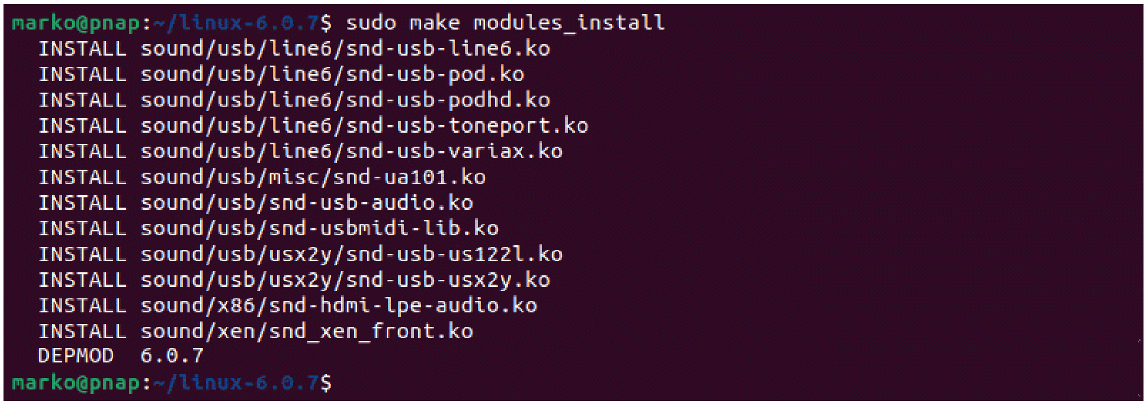
The terminal lists all Linux kernel components: memory management, hardware device drivers, filesystem drivers, network drivers, and process management.



```
marko@pnap:~/linux-6.0.7$ make
  SYNC    include/config/auto.conf.cmd
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
  HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
  HOSTCC  arch/x86/tools/relocs_32.o
  HOSTCC  arch/x86/tools/relocs_64.o
  HOSTCC  arch/x86/tools/relocs_common.o
  HOSTLD  arch/x86/tools/relocs
  HOSTCC  scripts/genksyms/genksyms.o
  YACC    scripts/genksyms/parse.tab.[ch]
```

2. Install the required modules with this command:

sudo make modules_install

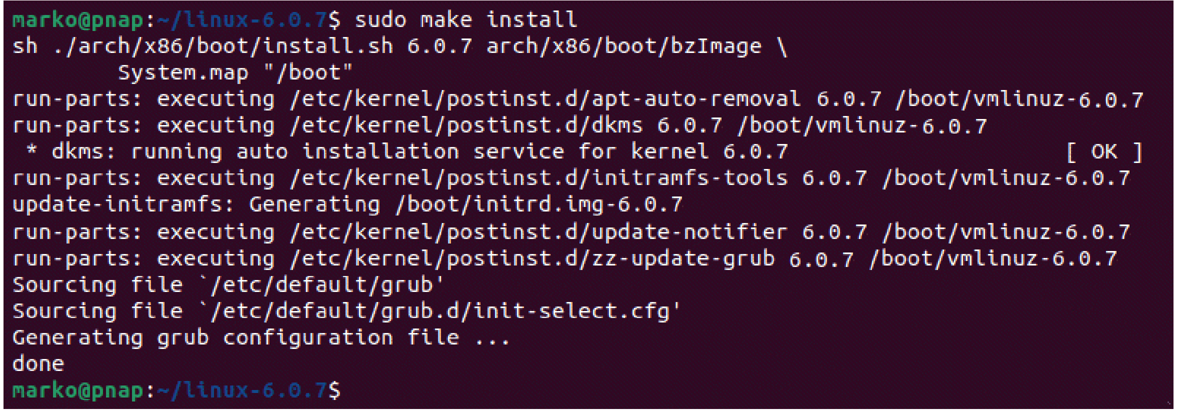


```
marko@pnap:~/linux-6.0.7$ sudo make modules_install
  INSTALL sound/usb/line6/snd-usb-line6.ko
  INSTALL sound/usb/line6/snd-usb-pod.ko
  INSTALL sound/usb/line6/snd-usb-podhd.ko
  INSTALL sound/usb/line6/snd-usb-toneport.ko
  INSTALL sound/usb/line6/snd-usb-variax.ko
  INSTALL sound/usb/misc/snd-ua101.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  6.0.7
marko@pnap:~/linux-6.0.7$
```

3. Finally, install the kernel by typing:

sudo make install

The output shows done when finished:



```
marko@pnap:~/linux-6.0.7$ sudo make install
sh ./arch/x86/boot/install.sh 6.0.7 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/dkms 6.0.7 /boot/vmlinuz-6.0.7
 * dkms: running auto installation service for kernel 6.0.7                     [ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.7 /boot/vmlinuz-6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.7 /boot/vmlinuz-6.0.7
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
done
marko@pnap:~/linux-6.0.7$
```

Step 6: Update the Bootloader (Optional)

The GRUB bootloader is the first program that runs when the system powers on.

The make install command performs this process automatically, but you can also do it manually.

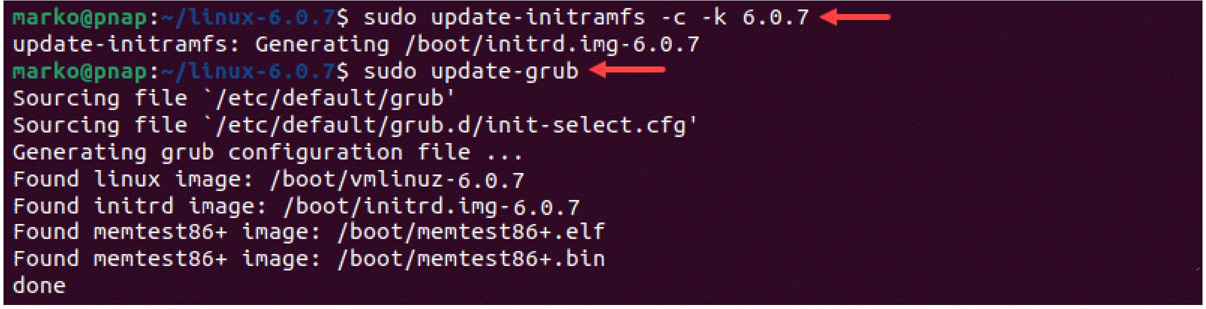
1. Update the initramfs to the installed kernel version:

sudo update-initramfs -c -k 6.0.7

2. Update the GRUB bootloader with this command:

sudo update-grub

The terminal prints out the process and confirmation message:



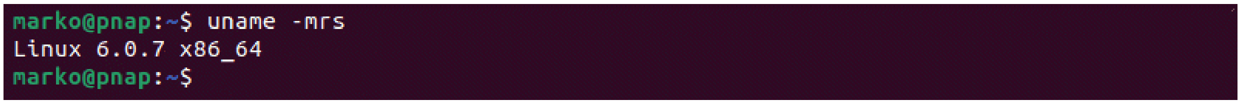
```
marko@pnap:~/linux-6.0.7$ sudo update-initramfs -c -k 6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
marko@pnap:~/linux-6.0.7$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.0.7
Found initrd image: /boot/initrd.img-6.0.7
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

Step 7: Reboot and Verify Kernel Version

When you complete the steps above, reboot the machine.
When the system boots up, verify the kernel version using the uname command:

uname -mrs

The terminal prints out the current Linux kernel version.



```
marko@pnap:~$ uname -mrs
Linux 6.0.7 x86_64
marko@pnap:~$
```

RESULT: