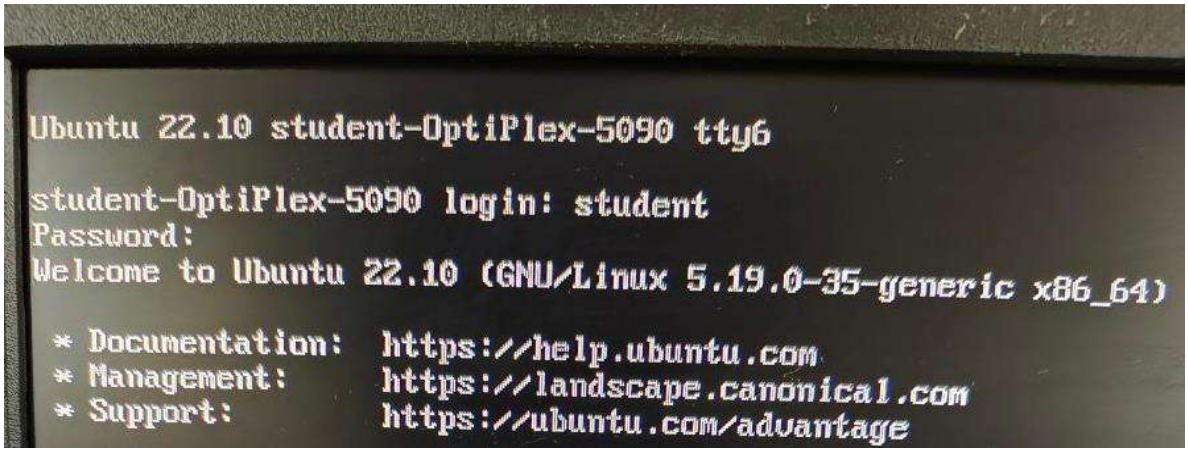
Q1 Connect to the Linux Server and understand the basic Directory Structure of Linux.

- **/ (Root directory):** The top-level directory, everything stems from here.
- **/bin:** Contains essential binary/executable files.
- **/sbin:** Similar to /bin, but holds binaries used by the system administrator.
- **/etc:** Configuration files for the system and installed programs.
- **/home:** Home directories for users.
- **/var:** Variable files like logs, spool files, etc.
- **/tmp:** Temporary files.
- **/usr:** User-related programs and files.
- **/lib and /lib64:** Essential shared libraries.
- **/opt:** Optional software packages.
- **/mnt and /media:** Mount points for removable media.
- **/dev:** Device files.
- **/proc and /sys:** Virtual filesystems that provide information about processes, hardware, etc.

To connect to a Linux server, you'd typically use an SSH client like PuTTY (on Windows) or the ssh command on a Unix-based system.



Q2 To understand help commands like:-man, info,help, whatis, apropos

- **man:** Manual pages command that displays the manual of a specific command or a topic. Example: `man ls` shows the manual for the `ls` command.

```
LS(1)                    User Commands                    LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current direc-
       tory  by  default).   Sort entries alphabetically if
       none of -cftuvSUX nor --sort is specified.
 Manual page ls(1) line 1 (press h for help or q to quit)
```

- **info:** Another documentation system often more detailed than `man` pages, with hyperlinks and structured sections. Example: `info ls` for information about `ls`.

```
Next: dir invocation,  Up: Directory listing

10.1 'ls': List directory contents
==================================

The 'ls' program lists information about files (of any type,\
 including
directories).  Options and file arguments can be intermixed \
arbitrarily,
as usual.  Later options override earlier options that are i\
ncompatible.

-----Info: (coreutils)ls invocation, 80 lines --Top----------
Welcome to Info version 7.0.3.  Type H for help, h for tutor
```

- **help:** Shell built-in command providing information on shell commands. For instance, `help cd` provides details about the `cd` command within the shell.

```
lynx@lynx:~/test$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the v
alue of the
    HOME shell variable.

    The variable CDPATH defines the search path for the directory
containing
    DIR.  Alternative directory names in CDPATH are separated by a
 colon (:).
```

- **whatis:** Provides a brief description of a command. Example: `whatis ls` gives a short description of the `ls` command.

```
lynx@lynx:~/test$ whatis ls
ls (1)               - list directory contents
lynx@lynx:~/test$
```

- **apropos:** Searches the manual page names and descriptions for the given keyword. For example, `apropos network` searches for commands related to "network."

```
lynx@lynx:~/test$ apropos network
nmap (1)             - Network exploration tool and security / ...
BIO_ctrl_dgram_connect (3ssl) - Network BIO with datagram seman...
BIO_ctrl_set_connected (3ssl) - Network BIO with datagram seman...
BIO_dgram_get_peer (3ssl) - Network BIO with datagram semantics
BIO_dgram_recv_timedout (3ssl) - Network BIO with datagram sema...
BIO_dgram_send_timedout (3ssl) - Network BIO with datagram sema...
BIO_dgram_set_peer (3ssl) - Network BIO with datagram semantics
BIO_new_dgram (3ssl) - Network BIO with datagram semantics
BIO_s_datagram (3ssl) - Network BIO with datagram semantics
```

Q3 To understand basic directory navigation commands like cat,ed, mv, cp, rm, mkdir,rmdir file, pwd command.

- **cd:** Change directory. Use `cd` followed by a directory name to navigate. Example: `cd Documents` moves to the "Documents" directory.
- **ls:** List directory contents. Use `ls` to display files and directories in the current directory. Example: `ls -l` shows detailed information.
- **pwd:** Print working directory. Displays the current directory path.
- **cat:** Concatenate and display file content. Example: `cat file.txt` shows the content of "file.txt".
- **ed:** Line-oriented text editor. Used in the terminal for editing files.
- **mv:** Move or rename files/directories. Example: `mv file1.txt file2.txt` renames "file1.txt" to "file2.txt".
- **cp:** Copy files/directories. Example: `cp file.txt /destination` copies "file.txt" to the `/destination` directory.
- **rm:** Remove files/directories. Be cautious, as it deletes files without confirmation. Example: `rm file.txt` deletes "file.txt".
- **mkdir:** Create directories. Example: `mkdir new_directory` creates a directory named "new_directory".
- **rmdir:** Remove directories. It only works if the directory is empty. Example: `rmdir empty_directory` removes the "empty_directory".

```
lynx@lynx:~/test/exercises$ cd ..
lynx@lynx:~/test$ ls
EOFwater,  exercises  file.txt  non-empty-dir
lynx@lynx:~/test$ pwd
/home/lynx/test
lynx@lynx:~/test$ cat exercises/example1
water, water everywhere and all the boards did shrink;
water, water everywhere, no drop to drink.
lynx@lynx:~/test$ rm file.txt
rm: remove write-protected regular empty file 'file.txt'? y
lynx@lynx:~/test$ mkdir new-test
lynx@lynx:~/test$ ls
EOFwater,  exercises  new-test  non-empty-dir
lynx@lynx:~/test$ rmdir new-test/
lynx@lynx:~/test$
```

Q4 To understand basic commands like:- date,cal, echo,bc,ls, who, whoami, hostname, uname, tty,aliase

- **date:** Displays the current date and time. Example: `date`
- **cal:** Shows a calendar for the current month or a specified month/year. Example: `cal` or `cal 12 2023` for December 2023.
- **echo:** Prints text or variables to the terminal. Example: `echo Hello, World!`
- **bc:** Basic calculator. Launches an interactive calculator. Example: `bc`
- **ls:** Lists files and directories in the current directory. Example: `ls`
- **who:** Displays information about currently logged-in users. Example: `who`
- **whoami:** Prints the username of the current user. Example: `whoami`
- **hostname:** Displays the system's hostname. Example: `hostname`
- **uname:** Shows system information. Example: `uname -a` displays all system information.
- **tty:** Prints the file name of the terminal connected to the standard input. Example: `tty`
- **alias:** Creates an alias for a command. Example: `alias ll='ls -l'` creates an alias 'll' for the `ls -l` command

```
lynx@lynx:~/test$ date
Sat Dec  2 17:54:38 IST 2023
lynx@lynx:~/test$ cal
    December 2023
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
lynx@lynx:~/test$ echo Hello, World!
Hello, World!
lynx@lynx:~/test$ who
lynx     tty2         2023-12-02 11:27 (:0)
lynx@lynx:~/test$ whoami
lynx
lynx@lynx:~/test$ hostname
lynx
lynx@lynx:~/test$ uname -a
Linux lynx 6.1.56-1-lts #1 SMP PREEMPT_DYNAMIC Fri, 06 Oct 2023 14:13:03 +0000 x86_64 GNU/Linux
lynx@lynx:~/test$ tty
/dev/pts/1
lynx@lynx:~/test$
```

Q5 To understand vi basics, Three modes of vi Editor, how to write, save, execute a shell script in vi editor.

1. **Command Mode:** The default mode when you open Vi. Used for navigating, deleting, copying, and pasting text. To enter Command Mode from another mode, press `Esc`.
2. **Insert Mode:** Used for typing and inserting text. To enter Insert Mode from Command Mode, press `i` for insert before the cursor, or `a` for insert after the cursor.
3. **Visual Mode:** Used for selecting and manipulating blocks of text. To enter Visual Mode from Command Mode, press `v`.

To write, save, and execute a shell script in Vi:

1. **Open Vi:** Type `vi scriptname.sh` in the terminal to open Vi with a new or existing script named `scriptname.sh`.
2. **Enter Insert Mode:** Press `i` to start typing your script.
3. **Write Your Script:** Enter the script commands.
4. **Save and Exit:**
   - **Save:** Press `Esc` to enter Command Mode, then type `:w` and press `Enter` to save.
   - **Save and Exit:** To save and exit, enter Command Mode (`Esc`), then type `:wq` and press `Enter`.

Q6 To understand process related commands like: - ps, top, pstree, nice, renice in Linux.

- **ps:** Process Status command used to display information about active processes. Commonly used options include `ps aux` to display all processes running on the system and `ps -ef` to show detailed information about all processes.

```
lynx@lynx:~/test$  ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.0  22184 14208 ?        Ss   11:27   0:26 /sbin/init
root           2  0.0  0.0      0     0 ?        S    11:27   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   11:27   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   11:27   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   11:27   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   11:27   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   11:27   0:00 [kworker/0:0H-events_highpri]
root          10  0.0  0.0      0     0 ?        I<   11:27   0:00 [mm_percpu_wq]
root          12  0.0  0.0      0     0 ?        I    11:27   0:00 [rcu_tasks_kthread]
root          13  0.0  0.0      0     0 ?        I    11:27   0:00 [rcu_tasks_rude_kthread]
root          14  0.0  0.0      0     0 ?        I    11:27   0:00 [rcu_tasks_trace_kthread]
root          15  0.0  0.0      0     0 ?        S    11:27   0:00 [ksoftirqd/0]
```

- **top:** Interactive real-time process viewer. It displays system summary information as well as a list of processes currently being managed by the Linux kernel. `top` provides a dynamic view of system processes, their resource usage, and system performance.

```
top - 17:57:57 up  6:30,  1 user,  load average: 2.65, 1.95, 1.57
Tasks: 316 total,   2 running, 314 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.5 us,  0.6 sy,  0.0 ni, 92.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15409.9 total,   6947.4 free,   5567.2 used,   3379.6 buff/cache
MiB Swap:   4931.0 total,   4931.0 free,      0.0 used.   9842.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 189601 lynx      20   0 2830776 432712 102172 R 100.0   2.7  32:34.70 Isolated Web Co
 222662 lynx      20   0 3091908 339552 125260 S  10.0   2.2   0:15.33 Isolated Web Co
 226168 lynx      20   0    9852   5728   3404 R  10.0   0.0   0:00.01 top
      1 root      20   0   22184  14208  10500 S   0.0   0.1   0:26.72 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.43 ksoftirqd/0
     16 root      20   0       0      0      0 I   0.0   0.0   0:06.23 rcu_preempt
     17 root      rt   0       0      0      0 S   0.0   0.0   0:00.07 migration/0
```

- **pstree:** Displays a tree diagram of processes, showing their hierarchical structure. It illustrates the parent-child relationship between processes.

```
lynx@lynx:~/test$ pstree
systemd─┬─NetworkManager─┬─dnsmasq
        │                └─3*[{NetworkManager}]
        ├─bluetoothd
        ├─dbus-daemon
        ├─dnsmasq───dnsmasq
        ├─firefox─┬─Isolated Servic───21*[{Isolated Servic}]
                  ├─Isolated Servic───24*[{Isolated Servic}]
                  ├─Isolated Web Co───38*[{Isolated Web Co}]
                  ├─6*[Isolated Web Co───27*[{Isolated Web Co}]]
                  ├─Isolated Web Co───35*[{Isolated Web Co}]
                  ├─Isolated Web Co───37*[{Isolated Web Co}]
                  ├─Privileged Cont───27*[{Privileged Cont}]
                  ├─RDD Process───3*[{RDD Process}]
```

- **nice:** A command used to execute processes with a specified priority. By default, processes are executed with a certain priority level (usually 0). nice allows setting the priority level of a process, with lower values indicating higher priority (a higher priority process gets more CPU time).

```
lynx@lynx:~/test$ nice -n 5 firefox
lynx@lynx:~/test$
```

- **renice:** Command to change the priority of already running processes. It alters the scheduling priority of one or more running processes. For instance, renice +10 PID increases the priority of a process with ID PID by 10 (lowering its priority), and renice -5 PID decreases the priority by 5 (raising its priority).

```
lynx@lynx:~/test$ renice -5 196447
renice: failed to set priority for 196447 (process ID): Permission denied
lynx@lynx:~/test$ sudo !!
sudo renice -5 196447
196447 (process ID) old priority 0, new priority -5
lynx@lynx:~/test$
```

Q7 To understand how to examine and change File permissions.

Examining File Permissions:

- **ls -l Command:** Use this command to view permissions. It displays file/directory information, including permissions, owners, groups, and more.
  For example:

```
lynx@lynx:~/test$ ls -l
total 0
-r--r--r-- 1 lynx lynx 0 Dec  2 17:13 file.txt
```

- Here, -r--r--r-- indicates permissions for the owner, group, and others.

Changing File Permissions:

- **chmod Command:** Used to change file permissions.
  - **Symbolic Mode:** Modifies permissions symbolically using letters (u, g, o for user, group, others; +, -, = for add, remove, set).
    - Example: chmod u+x filename adds execute permission for the file's owner.
  - **Numeric Mode:** Assigns permissions using numeric values (each digit represents user, group, others).
    - Example: chmod 755 filename gives read, write, and execute permissions to the owner, and read/execute to group and others.

Changing Ownership:

- **chown Command:** Changes file ownership.
  - Example: chown user:group filename changes the owner and group of the file.

Changing Group Ownership:

- **chgrp Command:** Changes group ownership of a file.
  - Example: chgrp newgroup filename changes the group of the file to newgroup.

Q8 Set a file to be read-only with the chmod command. Interpret the file permissions displayed by the ls -l command.

```
lynx@lynx:~/test$ ls -l
total 0
-rw-r--r-- 1 lynx lynx 0 Dec  2 17:13 file.txt
lynx@lynx:~/test$ chmod a-w file.txt
lynx@lynx:~/test$ ls -l
total 0
-r--r--r-- 1 lynx lynx 0 Dec  2 17:13 file.txt
lynx@lynx:~/test$
```

Q9 Delete one or more directories with the rmdir command. See what happens if the directory is not empty. Experiment (carefully!) with the rm -r command to delete a directory and its content.

```
lynx@lynx:~/test$ ls
empty-dir  file.txt  non-empty-dir
lynx@lynx:~/test$ rmdir non-empty-dir/
rmdir: failed to remove 'non-empty-dir/': Directory not empty
lynx@lynx:~/test$ rmdir empty-dir/
lynx@lynx:~/test$
```

Q10 Change your directory to the directory exercises. Create a file in that directory, named the file as example using the cat command containing the following text: water, water everywhere and all the boards did shrink; water, water everywhere, no drop to drink.

```
lynx@lynx:~/test$ mkdir exercises
lynx@lynx:~/test$ cat > exercises/example1 << EOF
water, water everywhere and all the boards did shrink;
water, water everywhere, no drop to drink.
EOF
lynx@lynx:~/test$ cat exercises/example1
water, water everywhere and all the boards did shrink;
water, water everywhere, no drop to drink.
lynx@lynx:~/test$
```

Q11 Write basic shell script to display the table of a number.

```
#!/bin/bash

# Get user input for the number
echo "Enter a number to display its table:"
read num

# Loop to display the table
echo "Table of $num:"
for (( i=1; i<=10; i++ ))
do
    echo "$num x $i = $((num*i))"
done
```

```
lynx@lynx:~/test$ ./test.sh
Enter a number to display its table:
6
Table of 6:
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

Q12 Write basic shell script to input a character from user and then check whether it is uppercase, lowercase or digit.

```
#!/bin/bash

# Get user input for a character
echo "Enter a character:"
read char

# Check if the input is uppercase, lowercase, or a digit
if [[ $char =~ [[:upper:]] ]]; then
    echo "The character is uppercase."
elif [[ $char =~ [[:lower:]] ]]; then
    echo "The character is lowercase."
elif [[ $char =~ [[:digit:]] ]]; then
    echo "The character is a digit."
else
    echo "The character is neither uppercase, lowercase, nor a digit."
fi
```

```
lynx@lynx:~/test$ ./test.sh
Enter a character:
hello
The character is lowercase.
lynx@lynx:~/test$
```

Q13 Write basic shell script to calculate factorial of a number.

```
#!/bin/bash

# Function to calculate factorial
factorial() {
    if [ $1 -eq 0 ] || [ $1 -eq 1 ]; then
        echo 1
    else
        fact=1
        for (( i=1; i<=$1; i++ ))
        do
            fact=$((fact * i))
        done
        echo $fact
    fi
}

# Get user input for the number
echo "Enter a number to calculate its factorial:"
read num

# Call the factorial function and display the result
result=$(factorial $num)
echo "The factorial of $num is: $result"
```

```
lynx@lynx:~/test$ ./test.sh
Enter a number to calculate its factorial:
20
The factorial of 20 is: 2432902008176640000
lynx@lynx:~/test$
```

Q14 Write basic shell script to input the month number and generate corresponding calendar.

```
#!/bin/bash

# Get user input for the month number
echo "Enter a month number (1-12) to generate the calendar:"
read month

# Check if the input is a valid month number
if [ $month -ge 1 ] && [ $month -le 12 ]; then
    cal $month $(date +"%Y")
else
    echo "Invalid month number. Please enter a number between 1 and
12."
fi
```

```
lynx@lynx:~/test$ ./test.sh
Enter a month number (1-12) to generate the calendar:
3
     March 2023
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Q15 Write basic shell script to list all directories.

```
#!/bin/bash

# List all directories in the current directory
echo "List of directories:"
for dir in */; do
    if [ -d "$dir" ]; then
        echo "$dir"
    fi
done
```

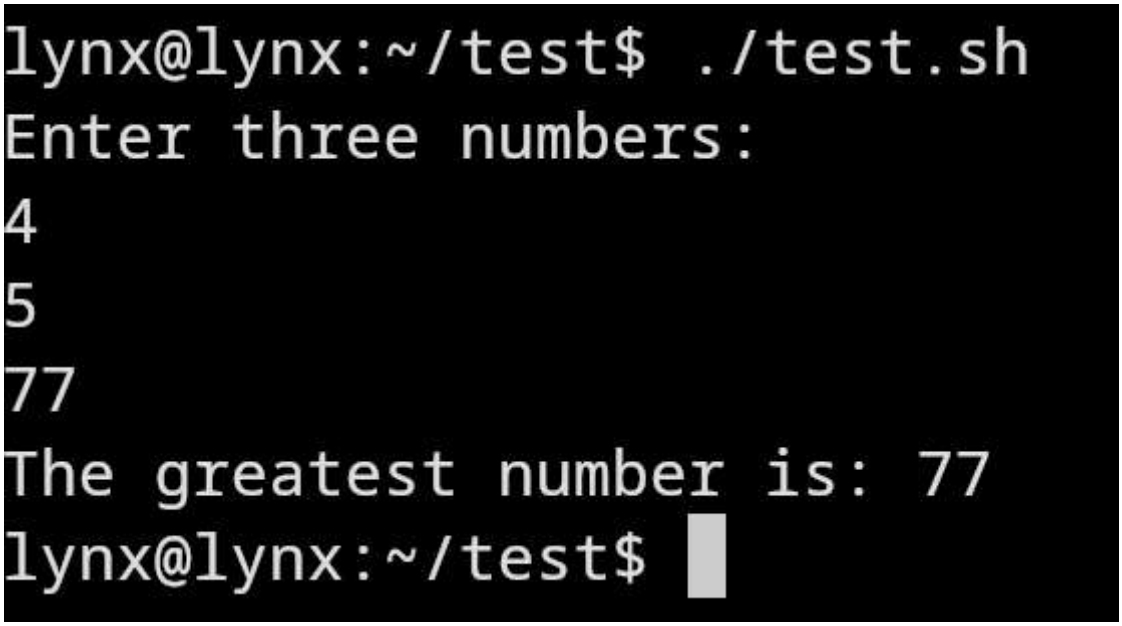
```
lynx@lynx:~/test$ ./test.sh
List of directories:
exercises/
non-empty-dir/
lynx@lynx:~/test$
```

Q16 Write basic shell script to display greatest of three numbers.

```
#!/bin/bash

# Get user input for three numbers
echo "Enter three numbers:"
read num1
read num2
read num3

# Check which number is the greatest
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
    echo "The greatest number is: $num1"
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
    echo "The greatest number is: $num2"
else
    echo "The greatest number is: $num3"
fi
```



Q17 Write basic shell script to check whether the number entered by user is prime or not.

```
#!/bin/bash

# Function to check if a number is prime
is_prime() {
    num=$1
    if [ $num -le 1 ]; then
        echo "Not a prime number."
    elif [ $num -eq 2 ] || [ $num -eq 3 ]; then
        echo "Prime number."
    else
        for (( i=2; i<=($num/2); i++ ))
        do
            if [ $((num%i)) -eq 0 ]; then
                echo "Not a prime number."
                exit
            fi
        done
        echo "Prime number."
    fi
}

# Get user input for the number
echo "Enter a number to check if it's prime:"
read number

# Call the function to check if the number is prime
is_prime $number
```

```
lynx@lynx:~/test$ ./test.sh
Enter a number to check if it's prime:
7
Prime number.
lynx@lynx:~/test$
```