

FACULTY OF ENGINEERING & TECHNOLOGY

BACHELOR OF TECHNOLOGY **OPERATING SYSTEMS LAB**

(303105252)

4th SEMESTER

Department of Artificial Intelligence

|  |
| --- |
| **LABORATORY MANUAL 2024-25** |

**CERTIFICATE**

This is to certified that Mr. **VIJAY KUMAR K** with enrolment No: **2303031240064** has successfully completed his/her laboratory experiments in the **OPERATING SYSTEM (303105252)** from the department of AI & AIDS during the academic year **2024-2025.**



Date of Submission: Staff in charge: **Dr. Sandeep Wadekar**

Head of Department: **Dr. Sanjay Agal**

**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **Title** | **Page no.** | **Performance date** | **Assessment date** | **Marks** | **Sign** |
| 1 | Study of Basic commands of Linux. |  |  |  |  |  |
| 2 | Study the basics of shell programming. |  |  |  |  |  |
| 3 | Write a Shell script to print given numbers sum of all digits. |  |  |  |  |  |
| 4 | Write a shell script to validate the entered date. (eg. Date format is: dd-mm-yyyy). |  |  |  |  |  |
| 5 | Write a shell script to check entered string is palindrome or not. |  |  |  |  |  |
| 6 | Write a Shell script to say Good morning/Afternoon/Evening as you log in to system. |  |  |  |  |  |
| 7 | Write a C program to create a child process. |  |  |  |  |  |
| 8 | Finding out biggest number from given three numbers supplied as command line arguments. |  |  |  |  |  |
| 9 | Printing the patterns using for loop. |  |  |  |  |  |
| 10 | Shell script to determine whether given file exist or not. |  |  |  |  |  |
| 11 | Write a program for process creation using C. (Use of gcc compiler). |  |  |  |  |  |
| 12 | Implementation of FCFS &Round Robin Algorithm. |  |  |  |  |  |
| 13 | Implementation of Banker's Algorithm. |  |  |  |  |  |

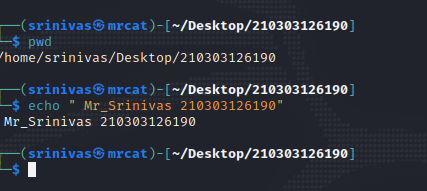
**PRACTICAL – 1**

**Aim**:Study of Basic commands of Linux.

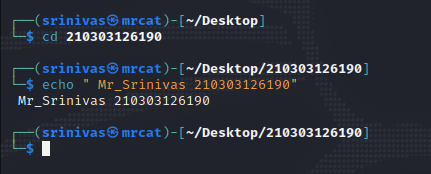
**Procedure:**

1. **PWD**

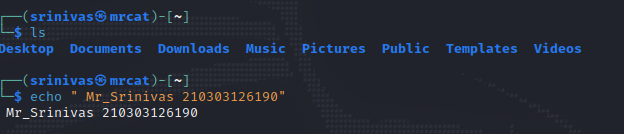
* Description: The pwd Linux command prints the current working directory path, starting from the root **(/)**. Use the pwd command to find your way in the Linux file system structure maze or to pass the working directory in a Bash script. In this tutorial, you will learn to use the pwd command.
* Syntax: pwd



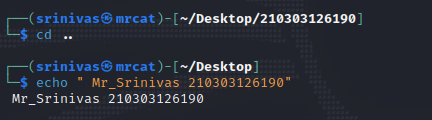
1. **CD**
   * + Description: The cd command is used to change the current directory in both Linux and other Unix-like systems.
     + Syntax: cd [directory]



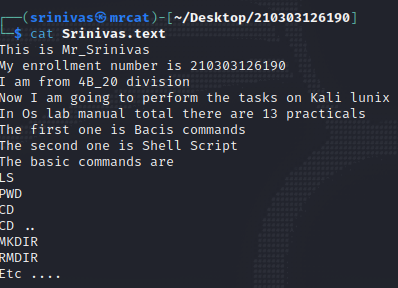
1. **LS**
   * + Description: we use ls command to list files and directories. This command will print all the file and directories in the current directory.
     + Syntax: ls [directory]



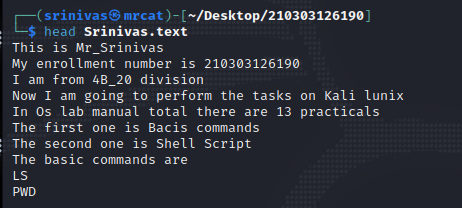
1. **CD ..**
   * + Description: This command is used to move to the parent directory of current directory, or the directory one level up from the current directory. “..” represents parent directory.
     + Syntax: cd ..



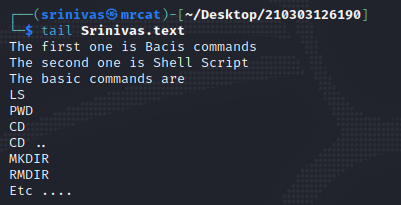
1. **CAT**
   * + Description: The cat command is a utility command in Linux. One of its most common usages is to print the content of a file onto the standard output stream. Other than that, the cat command also allows us to write some texts into a file.
     + Syntax: cat [file-name]



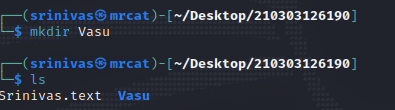
1. **HEAD**
   * + Description: The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.
     + Syntax: head [option] [file]



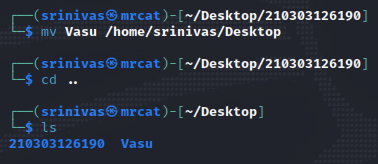
1. **TAIL**
   * + Description: Tail is a command which prints the last few numbers of lines (10 lines by default) of a certain file, then terminates. By default, “tail” prints the last 10 lines of a file, then exits. as you can see, this prints the last 10 lines of /var/log/messages.
     + Syntax: tail [option] [file]



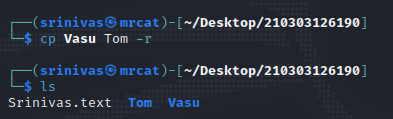
1. **MKDIR**
   * + Description: The mkdir command in Linux/Unix allows users to create or make new directories. mkdir stands for “make directory.” With mkdir , you can also set permissions, create multiple directories (folders) at once, and much more.
     + Syntax: mkdir [directory name]



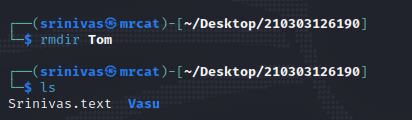
1. **MV**
   * + Description: The mv command termed as “Move”, which is a command-line utility to move files or directories from source to target. It supports the moving of a single file, multiple files, and directories.
     + Syntax: mv [option] source destination



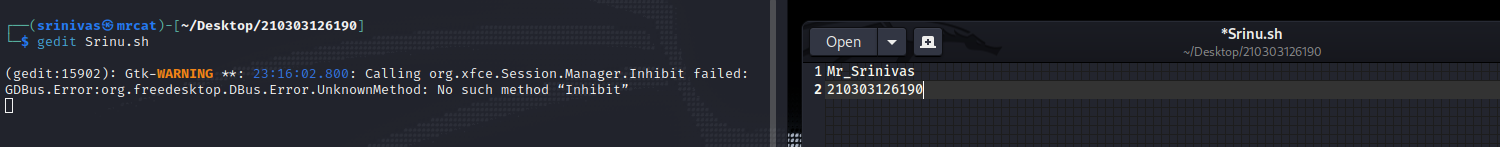
1. **CP**
   * + Description: cp command copies files (or, optionally, directories). The copy is completely independent of the original. You can either copy one file to another, or copy arbitrarily many files to a destination directory. In the first format, when two file names are given, cp command copies SOURCE file to DEST file.
     + Syntax: cp [option] source destination



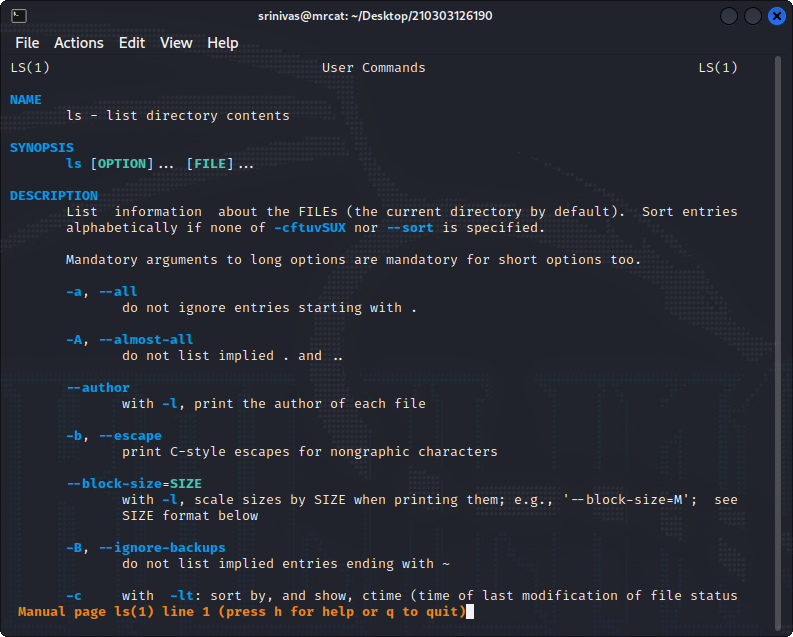
1. **RMDIR**
   * + Description: mdir command is used remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty. So if the specified directory has some directories or files in it then this cannot be removed by rmdir command.
     + Syntax: rmdir [directory name]



1. **GEDIT**
   * + Description: The gedit command is used to create and open a file
     + Syntax: gedit filename.txt



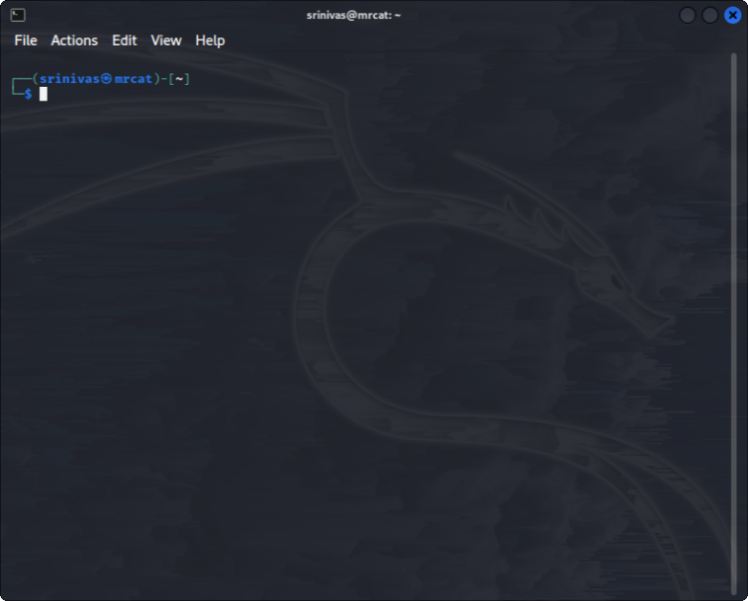
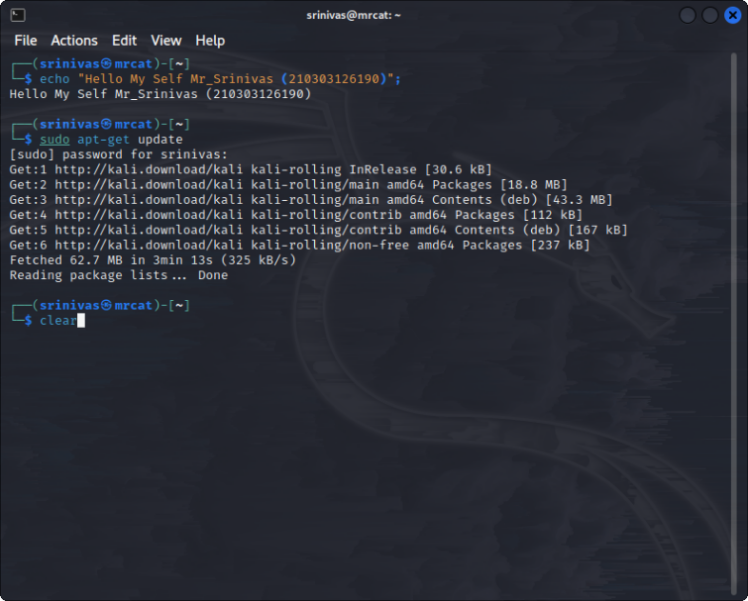
1. **MAN**
   * + Description: man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS
     + Syntax: man command



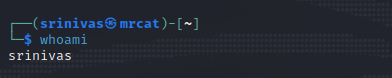
1. **ECHO**
   * + Description: Display text on the screen
     + Syntax: Display text on the screen



1. **CLEAR**
   * + Description: Used to clear the screen
     + Syntax: clear



1. **WHOAMI**
   * + Description: whoami prints the effective user ID. This command prints the username associated with the current effective user ID
     + Syntax: whoami [option]



1. **WC**
   * + Description: wc (word count) command, can return the number of lines, words, and characters in a file.
     + Syntax: wc [option]… [file]…

Example:

* Print the byte counts of file myfile.txt

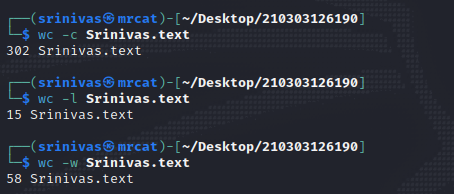
wc -c myfile.txt

* Print the line counts of file myfile.tx

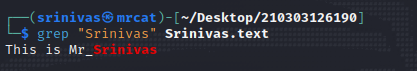
wc -l myfile.txt

* Print the word counts of file myfile.txt

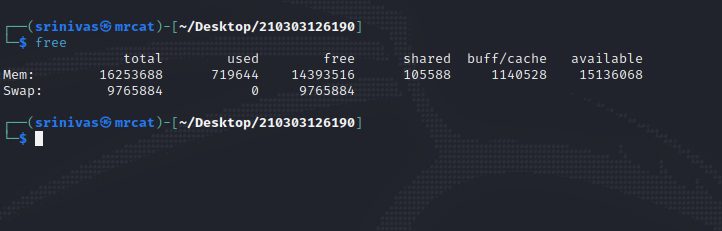
wc -w myfile.txt



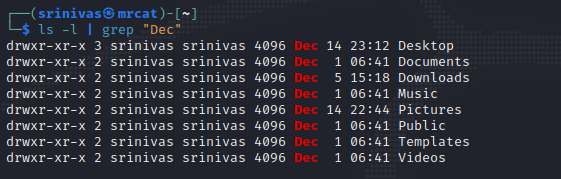
1. **GREP**
   * + Description: grep command uses a search term to look through a file
     + Syntax: grep [option]… Pattern [file]



1. **FREE**
   * + Description: To display the RAM details in Linux machine need to write following command.
     + Syntax: free



1. **PIPE (|)**
   * + Description: Pipe command is used to send output of one program as a input to another. Pipes “|” help combine 2 or more commands
     + Syntax: Command 1 | command 2



**PRACTICAL – 2**

**Aim**:Study the basics of Shell programming.

**What is a Shell?**

It is a list of commands in a computer program that is run by the Unix shell which is a command line interpreter. A shell script usually has comments that describe the steps.

An Operating is made of many components,

But its two prime components are –

* + - Kernel
    - ****Shell

A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one. A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it. When you run the terminal, the Shell issues a command prompt (usually $), where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal. The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name Shell.

**Types Of Shells:**

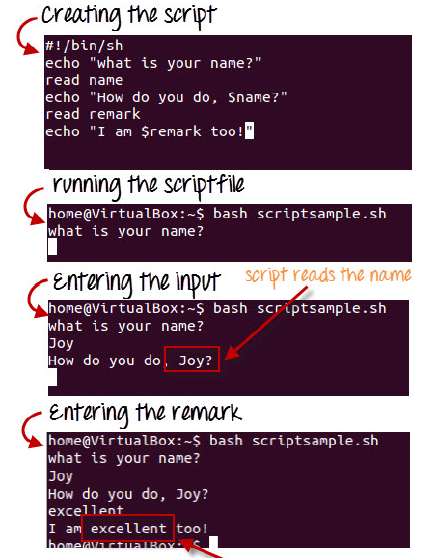
1. **Bournee shell:** This is default shell for version 7 unix. The character $ is the default prompt for the bourne shell.
2. **C shell:** This is a unix shell and a command processor that is run in a text window . The character % is the default prompt for the C shell. File commands can also be read easily by the C shell , which is known as a script.

**How to create file in linux:**

In Linux there are two commands which are used to create the files in Linux:

1. Gedit
2. nano

**What is Shell Scripting?**



Shell scripting is writing a series of command for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user. Let us understand the steps in creating a Shell Script

**1.** Create a file using a vi editor(or any other editor). Name script file with extension .sh

**2.** Start the script with #! /bin/sh

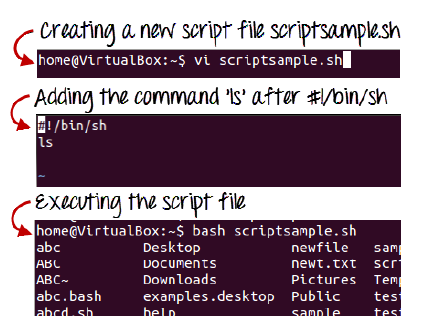
**3.** Write some code.

**4.** Save the script file as filename.sh

**5.** For executing the script type bash filename.sh

"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell. Let's create a small script -

Let's create a small script –



#!/bin/sh

ls

Let's see the steps to create it –

Command 'ls' is executed when we execute the scrip sample.sh file.

**Adding shell comments**

Commenting is important in any program. In Shell programming, the syntax to add a comment is

#comment

Let understand this with an example.

**What are Shell Variables?**

As discussed earlier, Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can by the shell only.

For example, the following creates a shell variable and then prints it:

variable ="Hello"

echo $variable

Below is a small script which will use a variable.

#!/bin/sh

echo "what is your name?"

read name

echo "How do you do, $name?"

read remark

echo "I am $remark too!"

Let's understand, the steps to create and execute the script



As you see, the program picked the value of the variable 'name' as Joy and 'remark' as excellent.

This is a simple script. You can develop advanced scripts which contain conditional statements, loops, and functions. Shell scripting will make your life easy and Linux administration a breeze.

**Summary:**

• Kernel is the nucleus of the operating systems, and it communicates between hardware and software

• Shell is a program which interprets user commands through CLI like Terminal

• The Bourne shell and the C shell are the most used shells in Linux

• Shell scripting is writing a series of command for the shell to execute

• Shell variables store the value of a string or a number for the shell to read

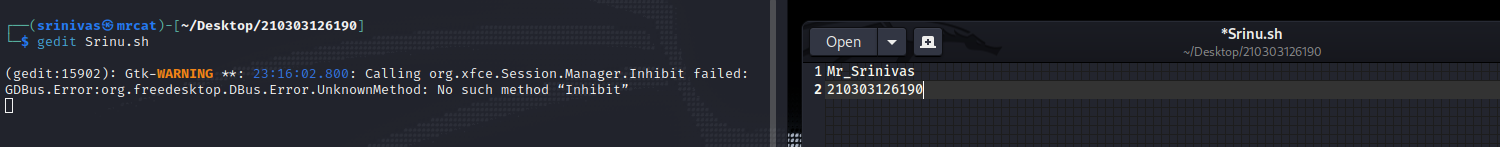
• Shell scripting can help you create complex programs containing conditional statements, loops, and functions .

These two commands are useful to create the files.

**1. Gedit:**

Syntax : gedit prac1.txt

Description: Gedit, the deafault GUI editor if you use Gnome ,also runs under KDE and other desktops . Most gNewsense and linux installations use gnome by default. To start Gedit open a terminal and type.

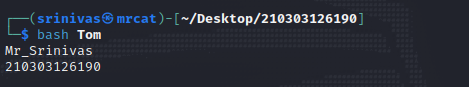


After Gedit command function a new window will open in that we have to give input. After giving input we have save the file and after that use command bash.

**2. Bash:**

Syntax : bash prac11.sh

Description : it is used to read the data in existing file in the linux .



**PRACTICAL –3**

**Aim**:Write a Shell script to print given numbers sum of all digits.

Sample Code:

echo "Hello, World!"# Hello World Program in Bash Shell

echo "enter number"

read number

reverse=0

original=$number

while [ $number -ne 0 ]

do

reminder=$(( $number % 10 ))

reverse=$(( $reverse \* 10 + $reminder ))

number=$(( $number / 10 ))

done

if [ $original -eq $reverse ]

then

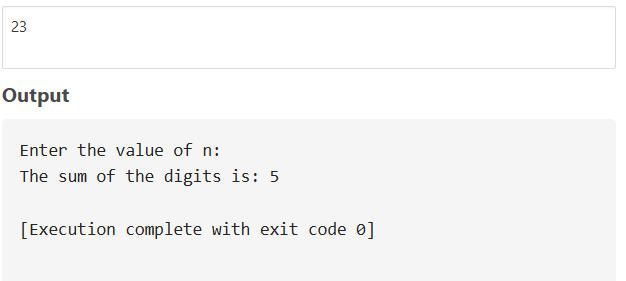
echo "$original is a palindrome"

else

echo "$orignal is a not palindrome"

fi

Output:



**PRACTICAL – 4**

**Aim**:Write a shell script to validate the entered date.

(eg. Date format is: dd-mm-yyyy).

**Sample Code:**

echo "date"

read d

echo "month"

read m

echo "year"

read y

n=`expr $y % 4`

echo $d"/"$m"/"$y

if [ $m -eq 4 ] || [ $m -eq 6 ] || [ $m -eq 9 ] || [ $m -eq 11 ]

then

if [ $d -gt 0 ] && [ $d -lt 31 ]

then

echo "valid"

else

echo "not valid"

fi

elif [ $m -eq 2 ]

then

if [ $n -eq 0 ] && [ $d -gt 0 ] && [ $d -lt 30 ]

then

echo "valid"

elif [ $n -gt 0 ] && [ $d -gt 0 ] && [ $d -lt 29 ]

then

echo "not valid"

else

echo "not valid"

fi

else

if [ $d -gt 0 ] && [ $d -lt 32 ]

then

echo "valid"

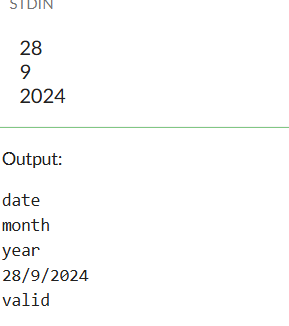
else

echo "not valid"

fi

fi

**Output:**



**PRACTICAL – 5**

**Aim:** Write a shell script to print whether the number is palindrome or not?

**Sample Code:**

echo " enter the number";

read a;

c=$a;

sum=0;

while [ $a -ne 0 ]

do

b=`expr $a % 10`;

echo "$a";

sum=`expr $sum \\* 10`;

sum=`expr $sum + $b`;

a=`expr $a \/ 10`;

done

echo "$sum";

if [ $c -eq $sum ]

then {

echo "it is palindrome $sum";

} fi;

OUT PUT :-

