--------------------------KERNEL-------------------------------------------------

1. Write a program to demonstrates command line argument passing to a Kernel Module.

--------------------------------SHELL---------------------------------------------------

1.Modify /etc/profile so that you get a special greeting message when you connect to your system as root.

2.Write a script called whichdaemon.sh that checks if the **httpd** and **init** daemons are running on your system. If an **httpd** is running, the script should print a message like, "This machine is running a web server." Use **ps** to check on processes.

3.Create a script that will take a (recursive) copy of files in /etc so that a beginning system administrator can edit files without fear.

4.Write a script that takes exactly one argument, a directory name. If the number of arguments is more or less than one, print a usage message. If the argument is not a directory, print another message. For the given directory, print the five biggest files and the five files that were most recently modified.

5..Write a script that does the following:

Display the name of the script being executed.

Display the first, third and tenth argument given to the script.

Display the total number of arguments passed to the script.

If there were more than three positional parameters, use **shift** to move all the values 3 places to the left.

Print all the values of the remaining arguments.

Print the number of arguments.

6.Write a script that automates the installation of a third-party package of your choice. The package must be downloaded from the Internet. It must be decompressed, unarchived and compiled if these actions are appropriate. Only the actual installation of the package should be uninterruptable.

-----------------------------------------SYSTEM CALL--------------------------------------

1. Implement in C the following Linux commands using system calls: (a) cat (b) ls (c) mv (d) cd (e) mkdir (f) cp (f) rm (g) grep

2: Write a C program that takes one or more file/directory names as command line input and reports the following information on the file:

(a) File type (b) Number of links (c) Time of last access (d) Read, write and execute permissions

3: Write a C program to emulate the Unix 'ls -l' command.

4: Write a C program to list every file in a directory, its inode number and file name.

5: Write a C program that illustrates how to execute two commands concurrently with a command pipe. Ex: ls -l | sort

6: Write a C program that illustrates suspending and resuming processes using signals.

7: Write a C program that implements a producer-consumer system with two processes (using semaphores).