# SWE4009 <u>LINUX Programming</u> L23+L24

# Practice Lab Session

Name:P.Tejapala
Reg No.: 17MIS1082

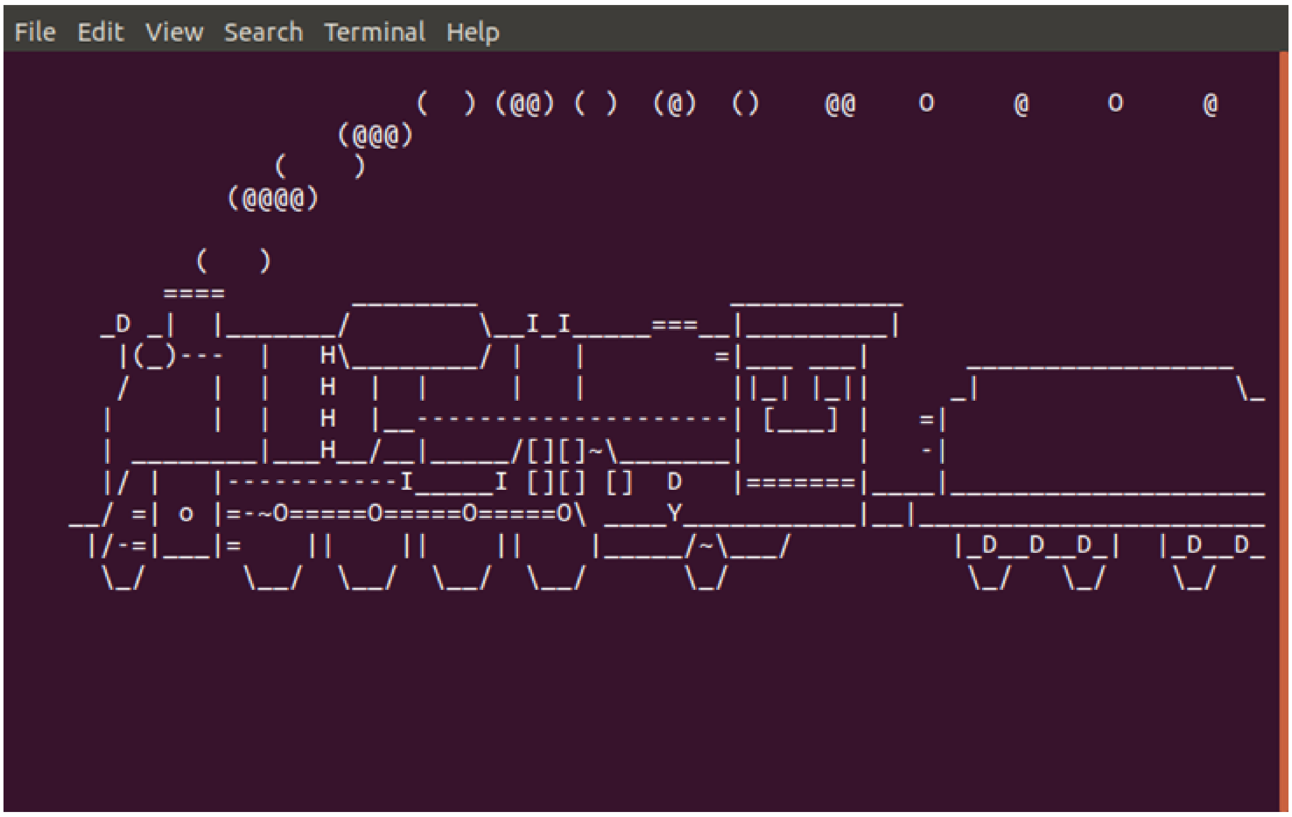
**Question 1:**

**To use SL command**

**Code:**

```
sudo apt-get install sl       #command to install
sl -v                         #to check version
sl                            #to run
```

**Output:**

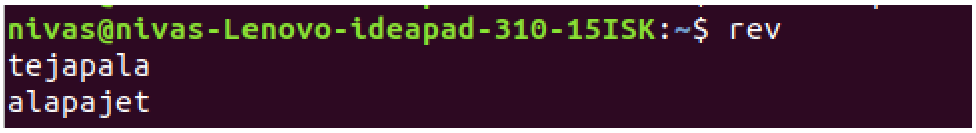


```
                        (  ) (@@) ( )  (@)  ()    @@    O     @     O     @
                   (@@@)
               (    )
            (@@@@)

          (   )
        ====        ________                ___________
    _D _|  |_______/        \__I_I_____===__|_________|
     |(_)---  |   H\________/ |   |        =|___ ___|      _________________
     /     |  |   H  |  |     |   |         ||_| |_||     _|                \_
    |      |  |   H  |__--------------------| [___] |   =|
    | ________|___H__/__|_____/[][]~\_______|       |   -|
    |/ |   |-----------I_____I [][] []  D   |=======|____|_____________________
  __/ =| o |=-~O=====O=====O=====O\ ____Y___________|__|_______________________
   |/-=|___|=    ||    ||    ||    |_____/~\___/          |_D__D__D_|  |_D__D_
    \_/      \__/  \__/  \__/  \__/      \_/               \_/   \_/    \_/
```

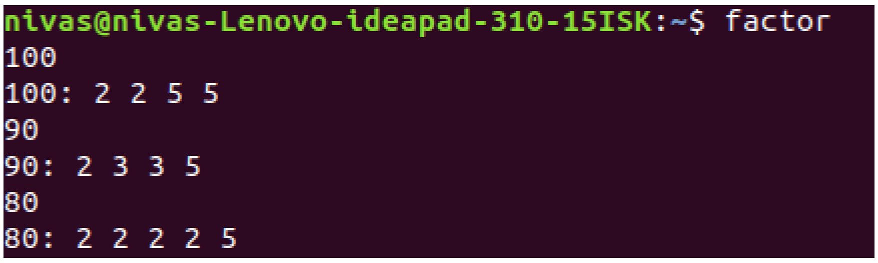
## Question 2:

### To reverse a string using rev command



```
nivas@nivas-Lenovo-ideapad-310-15ISK:~$ rev
tejapala
alapajet
```

### To use factor command



```
nivas@nivas-Lenovo-ideapad-310-15ISK:~$ factor
100
100: 2 2 5 5
90
90: 2 3 3 5
80
80: 2 2 2 2 5
```

## Additional Question :

### Write a C program to implement Simple reader-writer algorithm using shared memory segment with semaphore.

**Code:**

```
#include<stdio.h>
#include<unistd.h>

int main() {

   int pipefds1[2],  pipefds2[2];
   int returnstatus1,
   returnstatus2; int pid;

   char pipe1writemessage[20] = "Hi";
   char pipe2writemessage[20] =
   "Hello"; char readmessage[20];

   returnstatus1 = pipe(pipefds1);

   if (returnstatus1 == -1) {
      printf("Unable to create pipe 1
      \n"); return 1;

   }

   returnstatus2 = pipe(pipefds2);
   if (returnstatus2 == -1) {
      printf("Unable to create pipe 2
      \n"); return 1;
   }
   pid = fork();


   if (pid != 0) // Parent process {
      close(pipefds1[0]); // Close the unwanted pipe1 read
      side close(pipefds2[1]); // Close the unwanted pipe2
      write side printf("In Parent: Writing to pipe 1 -
      Message is %s\n",
pipe1writemessage);
      write(pipefds1[1], pipe1writemessage,
      sizeof(pipe1writemessage)); read(pipefds2[0], readmessage,
      sizeof(readmessage));
      printf("In Parent: Reading from pipe 2 - Message is
%s\n", readmessage);
   } else { //child process
      close(pipefds1[1]); // Close the unwanted pipe1
      write side close(pipefds2[0]); // Close the unwanted
      pipe2 read side read(pipefds1[0], readmessage,
      sizeof(readmessage));
      printf("In Child: Reading from pipe 1 - Message is %s\n",
      readmessage); printf("In Child: Writing to pipe 2 - Message is
      %s\n",
pipe2writemessage);
      write(pipefds2[1], pipe2writemessage, sizeof(pipe2writemessage));
   }
   return 0;
}
```

Explanation

Step 1 − Create pipe1 for the parent process to write and the child process to read.

Step 2 − Create pipe2 for the child process to write and the parent process to read.

Step 3 − Close the unwanted ends of the pipe from the parent and child side.

Step 4 − Parent process to write a message and child process to read and display on the screen.

Step 5 − Child process to write a message and parent process to read and display on the screen.

**HOT Question :**

**Write a bash shell script to monitor the health of your system. Let the details be stored and archived in any folder of your choice**

**Code:**

Health.sh

```
vmstat 1200 > vmstat1.data
filename= "/home/srihari/vmstat1.data"
tail -f $filename |
while read $line do
if [ (cat vmstat1.data | grep "swap")>0  ]
then
 echo "some rogue process has consumed massive amounts of memory">
swap.txt
fi
if [ (cat vmstat1.data | grep "r")>1  ]
then
 echo "some process are waiting to execute"> runqueue.txt
fi
if [ (cat vmstat1.data | grep "cpu")>1000  ]
then
 echo "cpu usage is more"> cpu.txt
fi
End
```

Explanation:

the vmstat 1200 – monitors every 24 hours and puts the data into the vmstat1.data

grep “swap”- the swap should always be zero if its not then some process has consumed massive memory. That will be monitored in this line

grep “r”- the running queue is constantly above process 1 it indicates the system is slow and some process is waiting to be executed. That will be monitored here.

Grep “cpu”- it indicates the cpu usage of the system. If the cpu usage is more it will be monitored and will alert in this line.