**ASSIGNMENT 1**

**Q1:**

**W**rite a shell script which accepts length and breadth of a rectangle and calculates the area and perimeter of the reactangle

**Code**

read -p "Enter length : " len

read -p "Enter breadth : " br

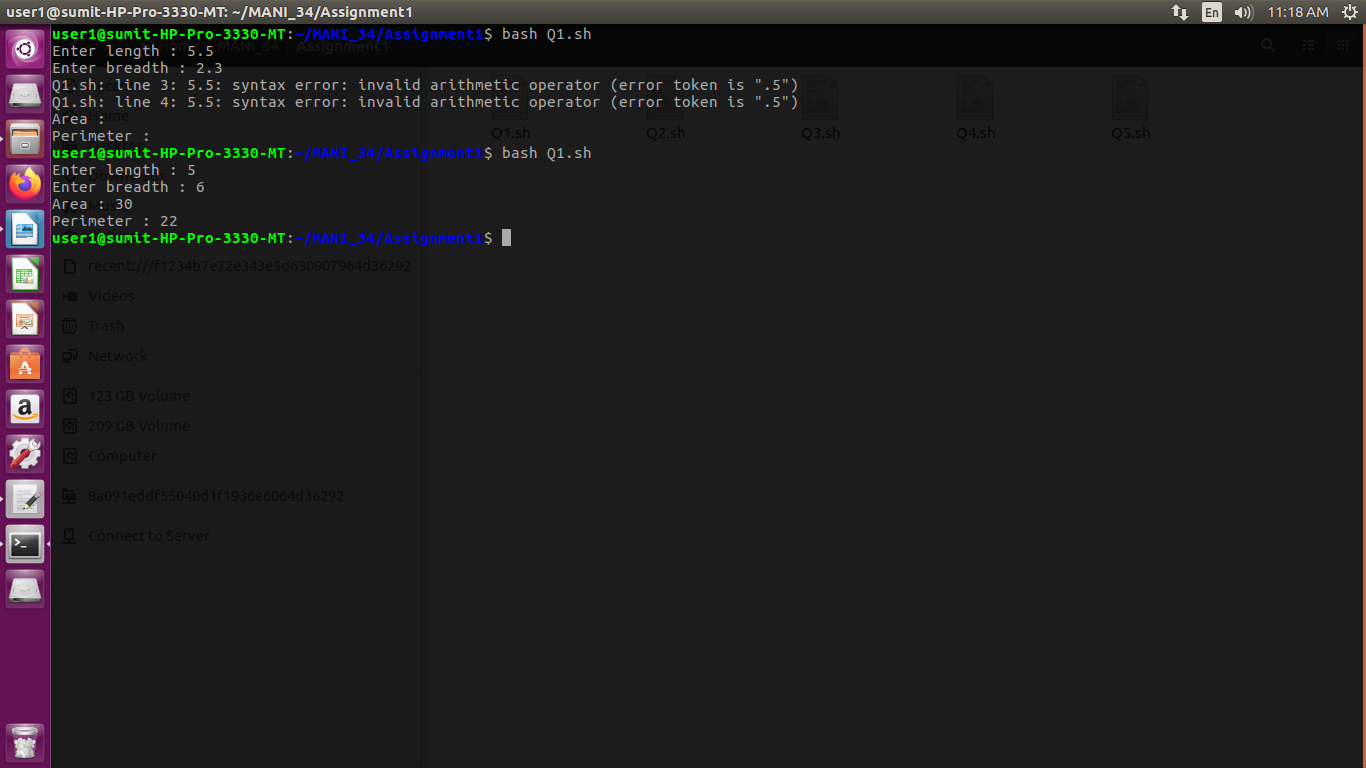
area=$((len\*br))

per=$((2\*len+2\*br))

echo "Area : $area"

echo "Perimeter : $per"

**Input/ Output**

****

**Q2:**

Write a shell Script which accepts basic salary of an employee and calculates net salary and display the salary.

**Code**

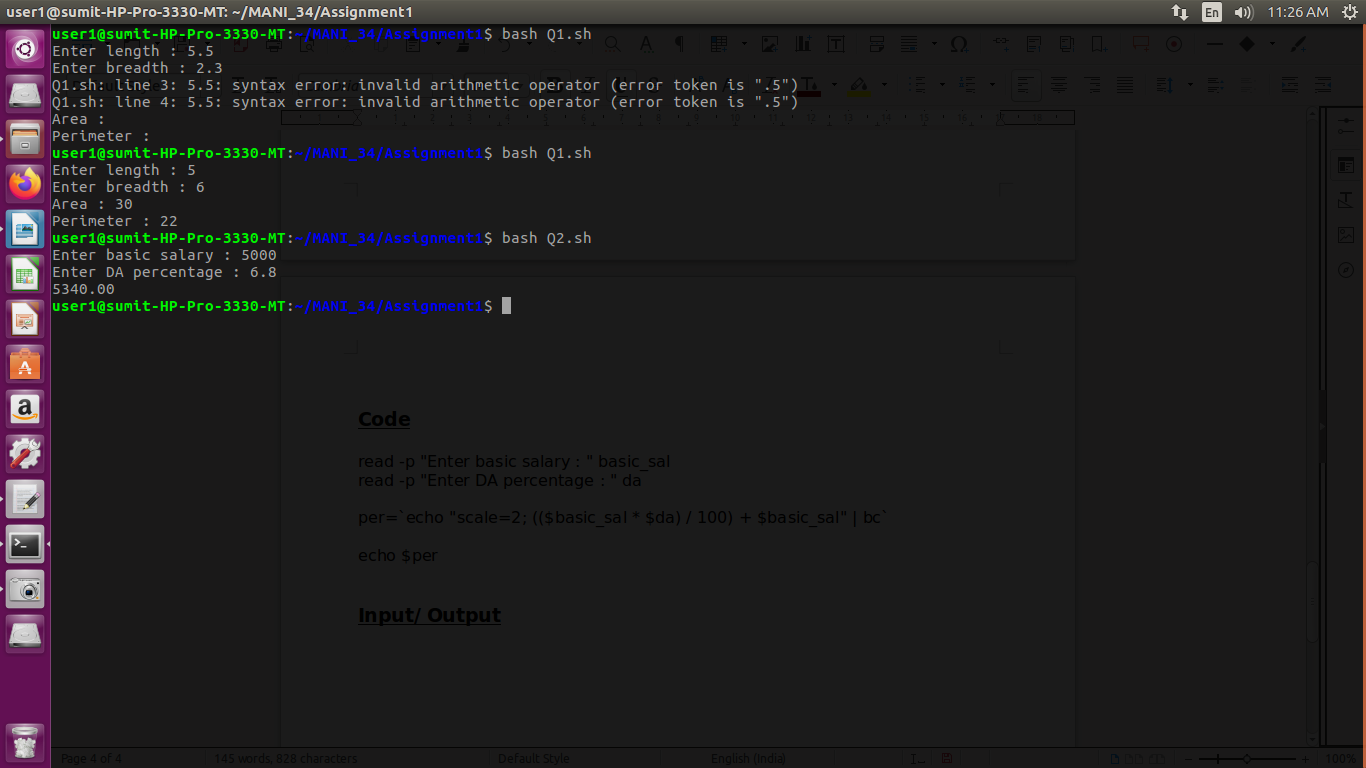
read -p "Enter basic salary : " basic\_sal

read -p "Enter DA percentage : " da

per=`echo "scale=2; (($basic\_sal \* $da) / 100) + $basic\_sal" | bc`

echo $per

**Input/ Output**

****

**Q3:**

Write a shell script which accepts a five digit number and prints the sum of its digits.

***Code***

*read -p "Enter 5 digit number : " num*

*sum=0*

*while [ $num -gt 0 ]*

*do*

*rem=$num%10*

*num=$((num/10))*

*sum=$((sum+rem))*

*done*

*echo "Sum of digits : $sum"*

***Input/ Output***

******

**Q4:**

Write a shell script which accepts a five digit number and reverse number.

***Code***

*read -p "Enter a number : " num*

*cpy=$num*

*nod=0*

*while [ $cpy -gt 0 ]*

*do*

*nod=$((nod+1))*

*cpy=$((cpy/10))*

*done*

*cpy=$num*

*echo $nod*

*nod=$((nod-1))*

*rev=0*

*while [ $cpy -gt 0 ]*

*do*

*rem=$cpy%10*

*rem=$((rem\*10\*\*nod))*

*rev=$((rev+rem))*

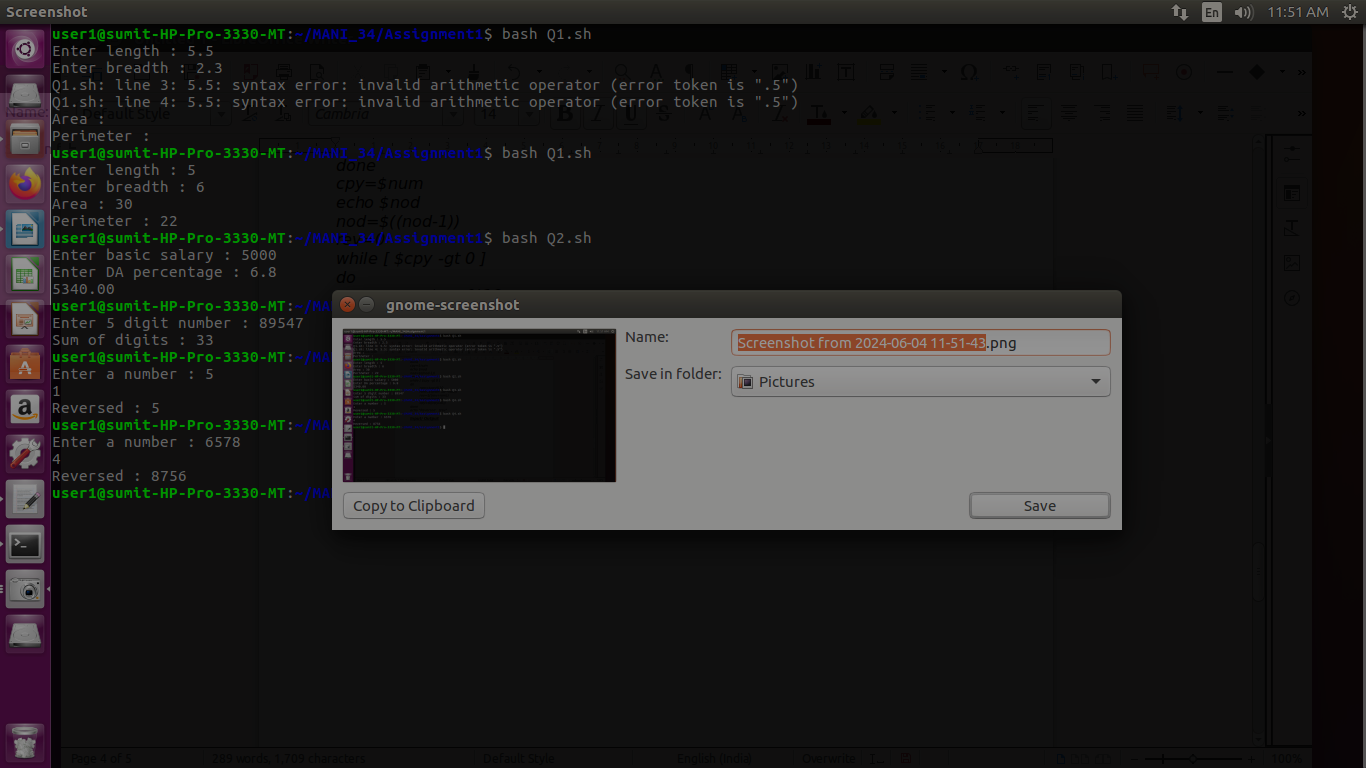
*nod=$((nod-1))*

*cpy=$((cpy/10))*

*done*

*echo "Reversed : $rev"*

***Input/ Output***

******

**Q5:**

*The* /etc/passwd file stores user account information it contains one entry per line for each user(user account) of the system. Each line contains seven fields which are separated by a colon(:) symbol. The fields are:

(i)Username

(ii) Password

(iii) User Id

(iv) Group Id

(v) User Id info

(vi) Home Directory

(vii) Login Shell

Write a shell script which accepts a user login name and diplays detail information about the users as available from the file/etc/passwd.

**Code**

`cd`

read -p "Enter username : " user

x=`cat /etc/passwd | grep $user`

g=`echo $x > file1`

echo "File created"

username=`cut -d ":" -f 1 file1`

echo "Username : $username"

echo "Password : `cut -d ":" -f 2 file1`"

echo "User ID : `cut -d ":" -f 3 file1`"

echo "Group ID : `cut -d ":" -f 4 file1`"

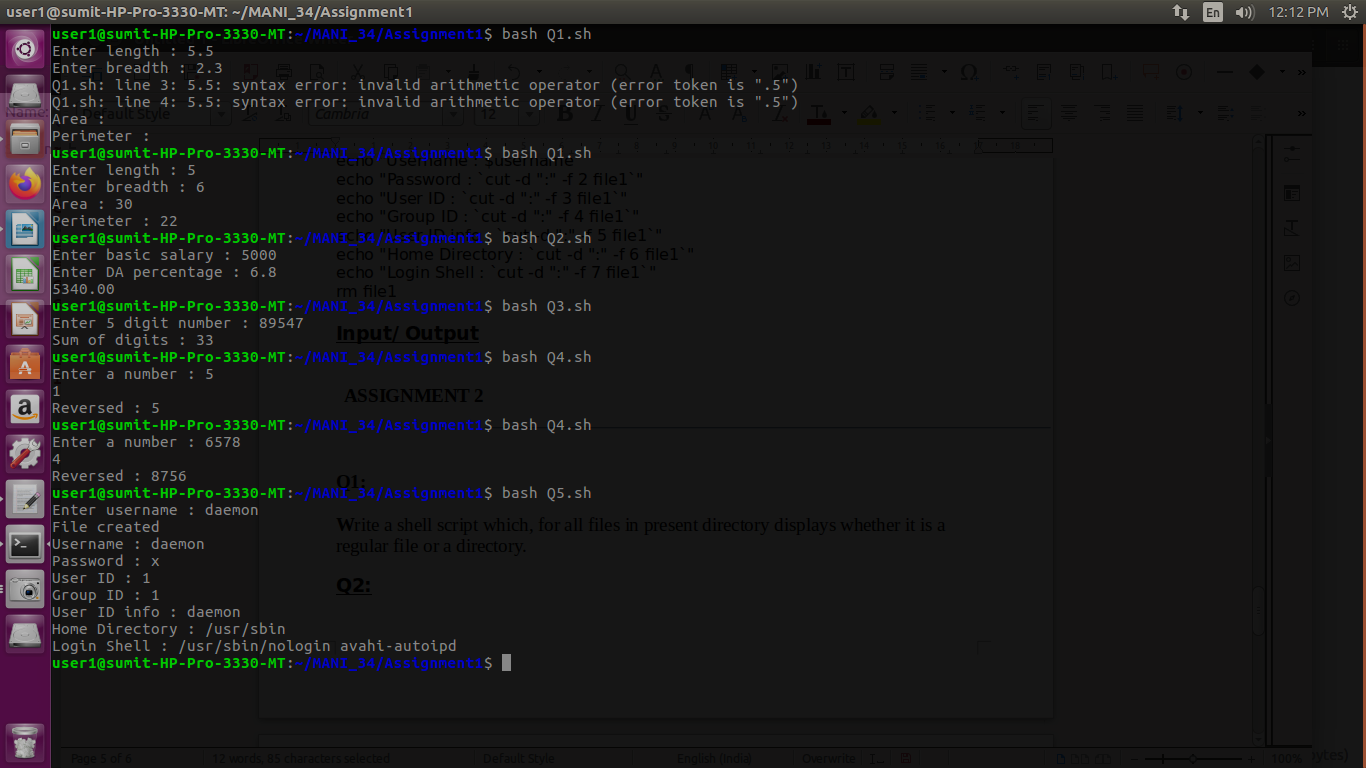
echo "User ID info : `cut -d ":" -f 5 file1`"

echo "Home Directory : `cut -d ":" -f 6 file1`"

echo "Login Shell : `cut -d ":" -f 7 file1`"

rm file1

**Input/ Output**

****

**ASSIGNMENT 2**

**Q1:**

**W**rite a shell script which, for all files in present directory displays whether it is a regular file or a directory.

**Code**

*lines=`ls -l | wc -l`*

*for (( i=2; i<=lines; i++ ))*

*do*

*temp=`ls -l | head -$i | tail -1`*

*line=`echo $temp > file`*

*echo -n `cut -d " " -f 9 file`*

*echo -n " : "*

*test=`cut -c 1 file`*

*if [ $test = "d" ]*

*then*

*echo "directory"*

*else*

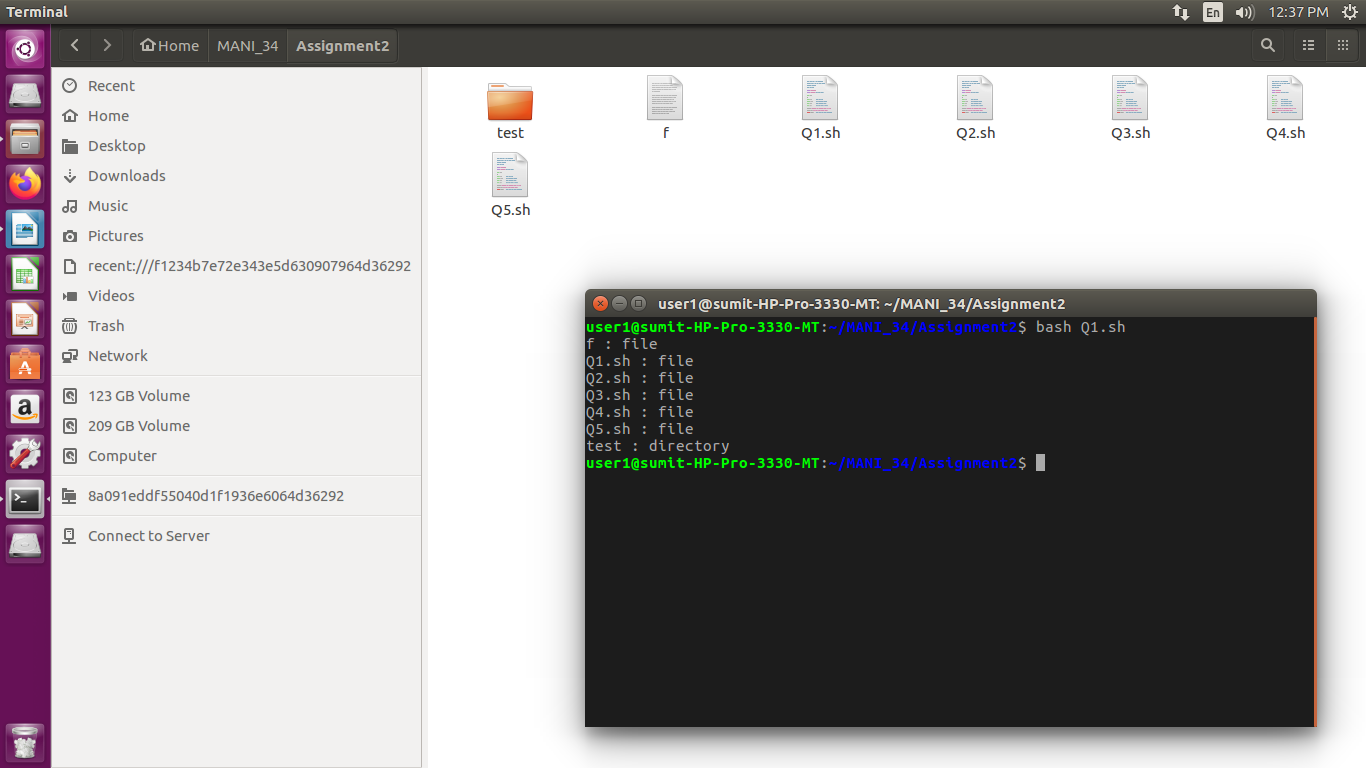
*echo "file"*

*fi*

*`rm file`*

*done*

***Input/ Output***

******

**Q2:**

The PATH variable is an environment variable that contains an ordered list of paths that Linux will search for executables when running a command. Write a shell script to display all the directories in the PATH variable in a simple way i.e., one line per directory. In addition, display information about each directory, such as the permissions and the modification times.

**Code**

*IFS=:*

*for path in $PATH;*

*do*

*echo "Directory : $path"*

*x=`ls -ld $path | cut -d " " -f 1`*

*echo "Permissions : $x"*

*y=`ls -ld $path | cut -d " " -f 8`*

*echo "Modification time : $y"*

*done*

***Input/ Output***

*user1@sumit-HP-Pro-3330-MT:~/MANI\_34/Assignment2$ bash Q2.sh*

*Directory : /home/user1/bin*

*ls: cannot access '/home/user1/bin': No such file or directory*

*Permissions :*

*ls: cannot access '/home/user1/bin': No such file or directory*

*Modification time :*

*Directory : /home/user1/.local/bin*

*ls: cannot access '/home/user1/.local/bin': No such file or directory*

*Permissions :*

*ls: cannot access '/home/user1/.local/bin': No such file or directory*

*Modification time :*

*Directory : /usr/local/sbin*

*Permissions : drwxr-xr-x*

*Modification time : 7*

*Directory : /usr/local/bin*

*Permissions : drwxr-xr-x*

*Modification time : 7*

*Directory : /usr/sbin*

*Permissions : drwxr-xr-x*

*Modification time : 9*

*Directory : /usr/bin*

*Permissions : drwxr-xr-x*

*Modification time : 9*

*Directory : /sbin*

*Permissions : drwxr-xr-x*

*Modification time : 2*

*Directory : /bin*

*Permissions : drwxr-xr-x*

*Modification time : 7*

*Directory : /usr/games*

*Permissions : drwxr-xr-x*

*Modification time : 7*

*Directory : /usr/local/games*

*Permissions : drwxr-xr-x*

*Modification time : 7*

*Directory : /snap/bin*

**Q3:**

Write a shell script which dispalys vendor\_id, model name, cpu Mhz, cache size information about the processor present in your computer. Hint : most of this information can be obtained by reading the file /proc/cpuinfo.

**Code**

*`cd`*

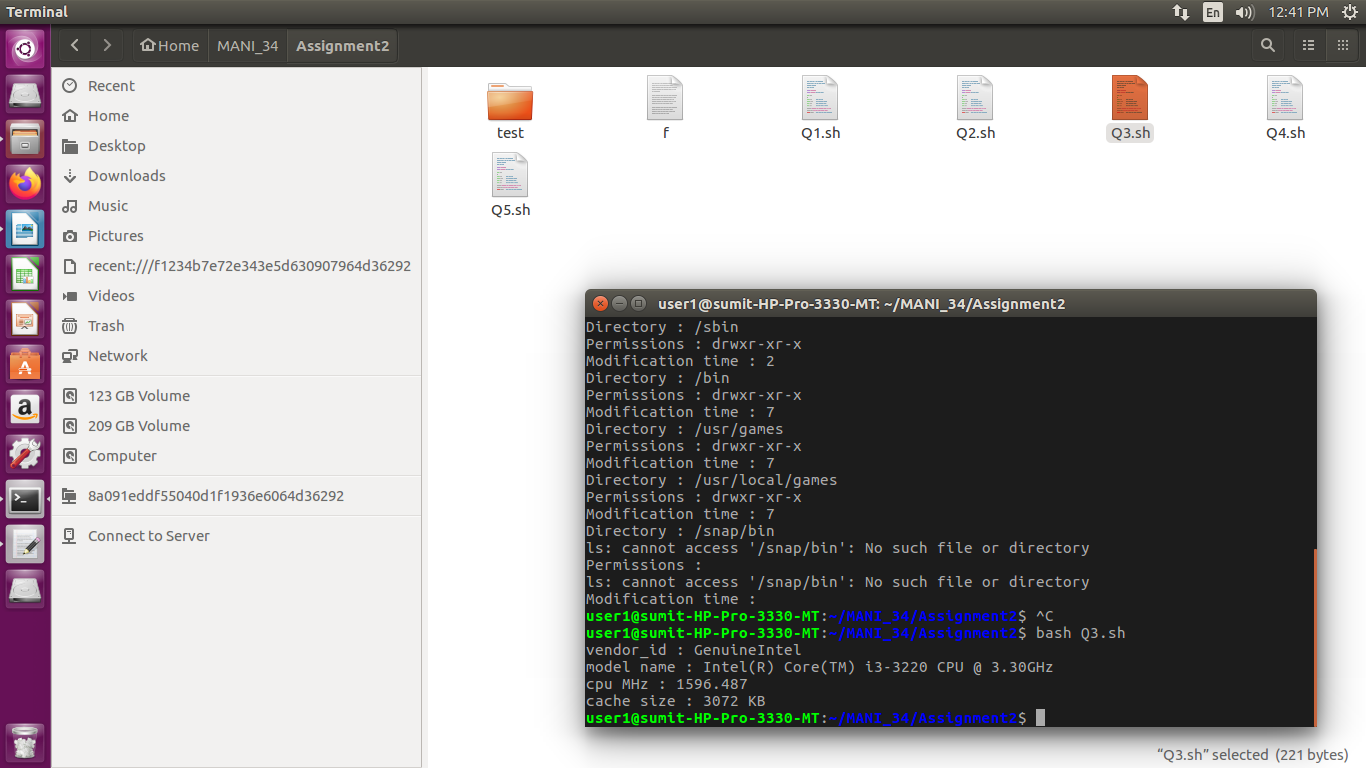
*echo `cat /proc/cpuinfo | grep "vendor\_id" | head -1`*

*echo `cat /proc/cpuinfo | grep "model name" | head -1`*

*echo `cat /proc/cpuinfo | grep "cpu MHz" | head -1`*

*echo `cat /proc/cpuinfo | grep "cache size" | head -1`*

***Input/ Output***

******

**Q4:**

Write a shell script to show your home directory, Operating System type, version, release number, kernel version and current path setting. Hint : use uname command or use content of /proc/sys/kernel/osrelease file.

**Code**

*echo "Home directory : `pwd`"*

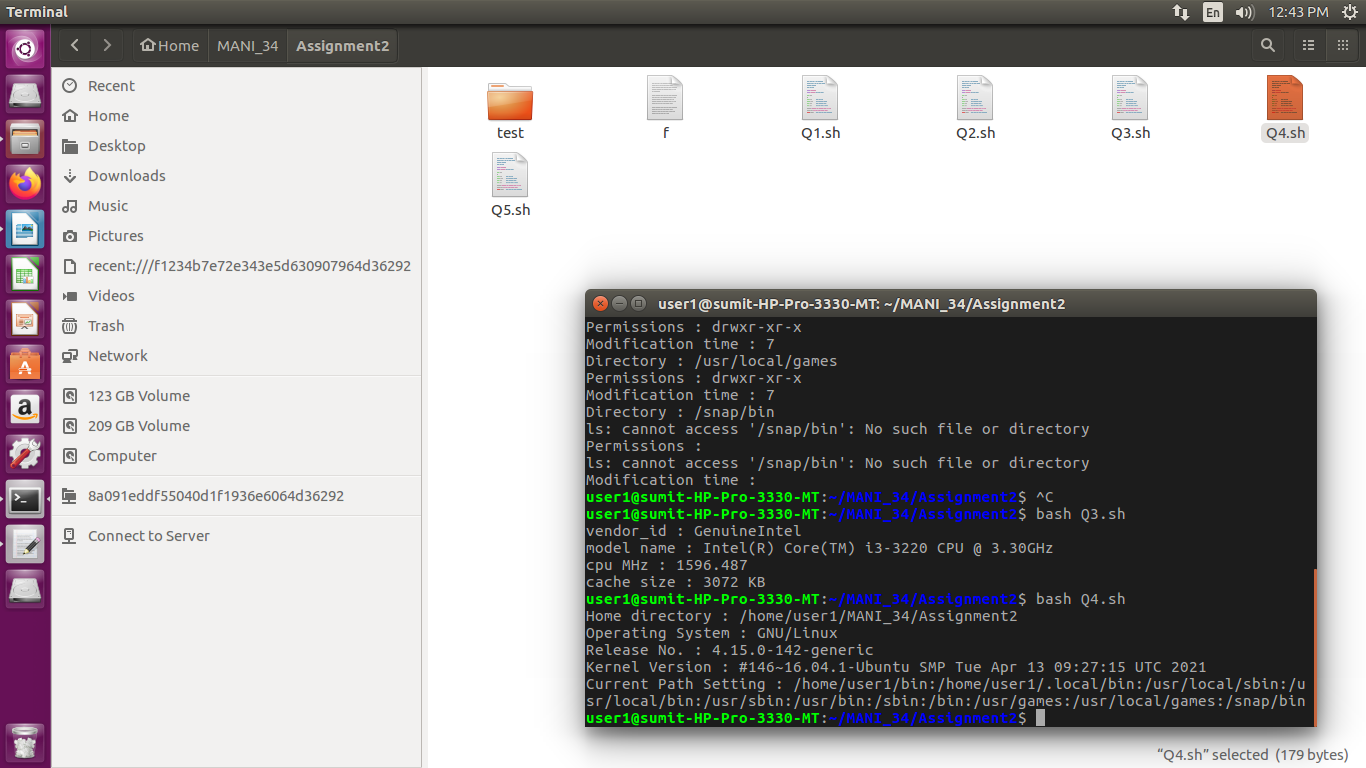
*echo "Operating System : `uname -o`"*

*echo "Release No. : `uname -r`"*

*echo "Kernel Version : `uname -v`"*

*echo "Current Path Setting : `echo $PATH`"*

***Input/ Output***

******

**Q5:**

Write a shell script to display a summary of the disk space usage for each directory argument(and any subdirectories), both in terms of bytes and kilobytes or megabytes(whichever is appropriate). [du -b]

**Code**

*lines=`du -b | wc -l`*

*for (( i=1; i<=lines; i++ ))*

*do*

*line=`du -b | head -$i | tail -1`*

*`echo $line > temp`*

*byte=`cut -d " " -f 1 temp`*

*dire=`cut -d " " -f 2 temp`*

*if [ $dire == "." ]*

*then*

*dire="Current Directory"*

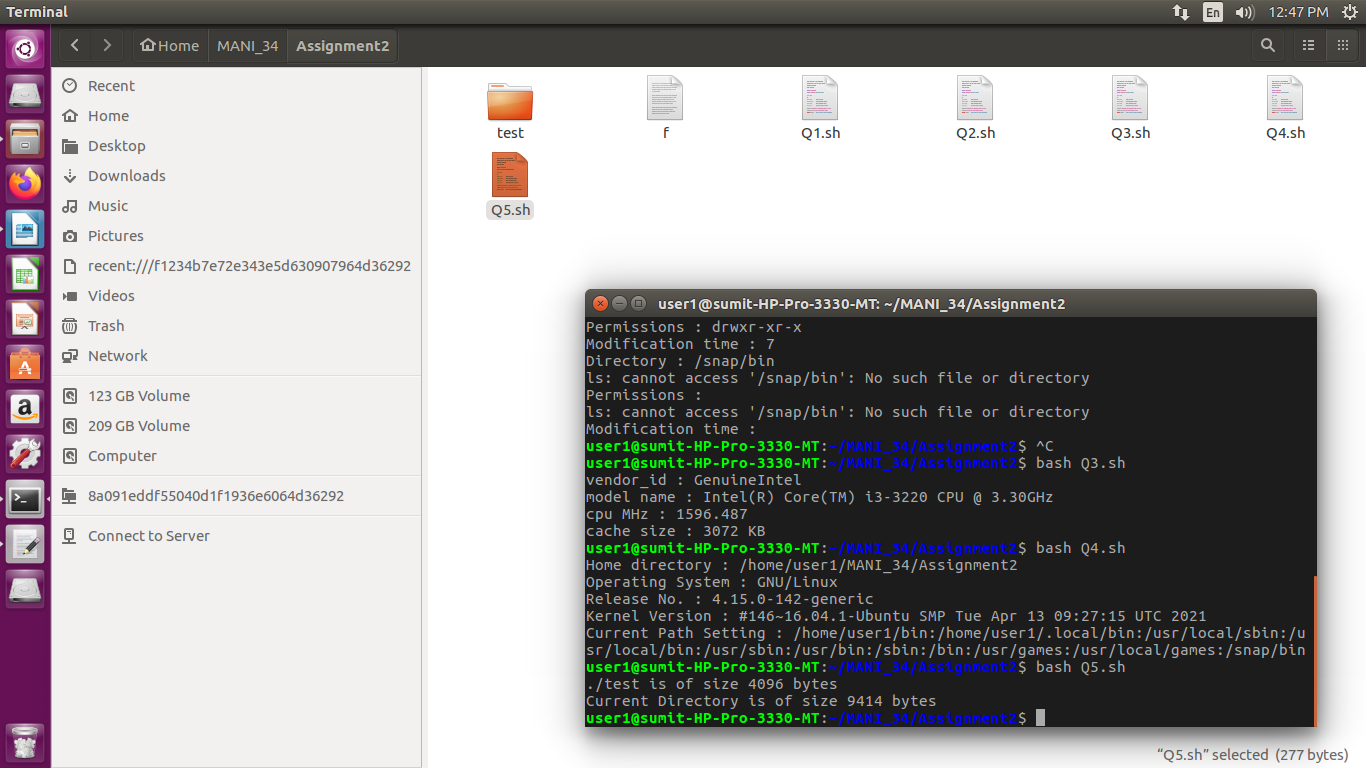
*fi*

*echo "$dire is of size $byte bytes"*

*rm temp*

*done*

***Input/ Output***

******

**ASSIGNMENT 3**

**Q1:**

**W**rite a shell script that reads a input file that contains three integers in each line. The script should display the sum of all integers in each line.

**Code**

*read -p "Enter file name : " file*

*sum=0*

*while read -r lines*

*do*

*sum=$((sum+lines))*

*done < $file*

*echo "Sum of numbers of file is : $sum"*

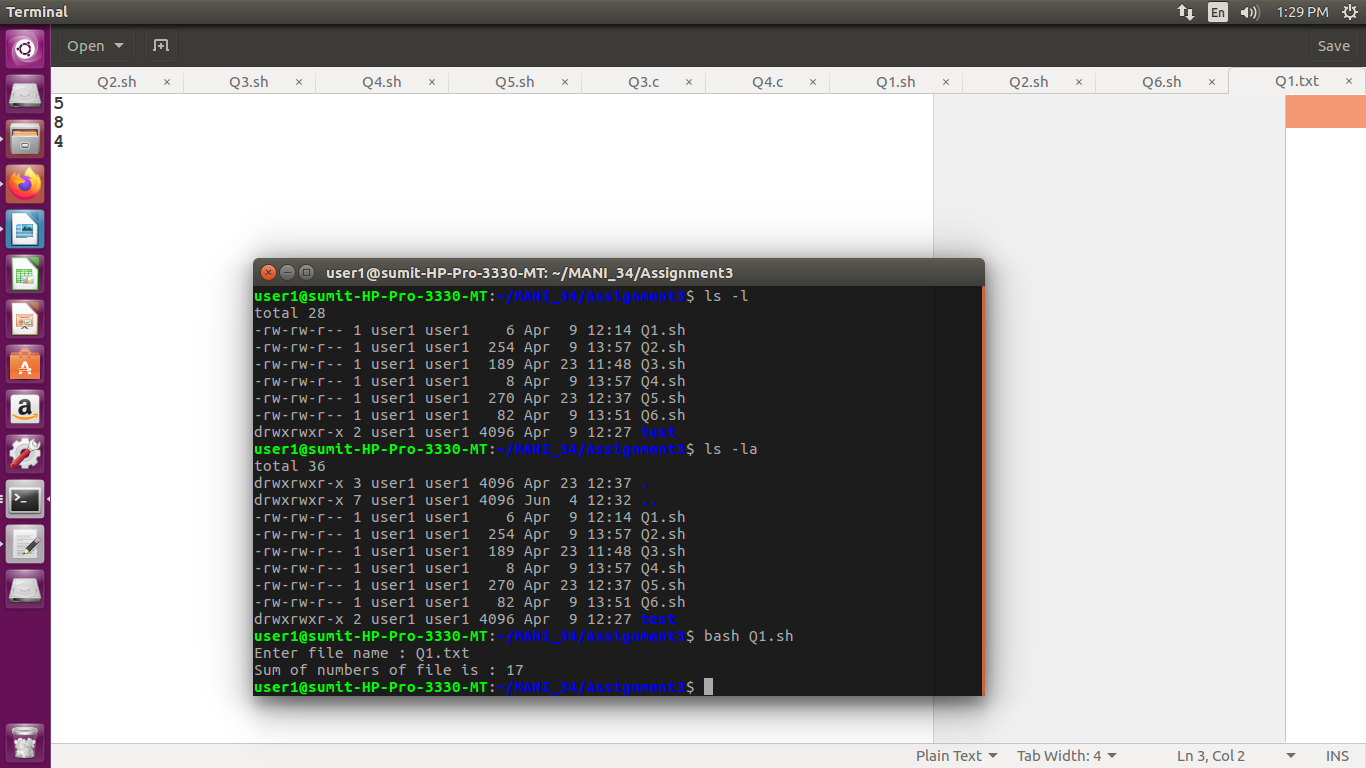
***Content of File -> Q1.txt***

*5*

*8*

*4*

***Input/ Output***

******

**Q2:**

**W**rite a shell script that find out how many file and directory are there in the current directory. Also list the file directory names separately.

**Code**

*echo "No of files : `ls -l | egrep -c '^-'`"*

*echo "\*\*\*\*Files\*\*\*\*"*

*for i in \**

*do*

*if [ -f $i ]*

*then*

*echo "$i"*

*fi*

*done*

*echo "No of directories : `ls -l | egrep -c '^d'`"*

*echo "\*\*\*\*Directories\*\*\*\*"*

*for i in \**

*do*

*if [ -d $i ]*

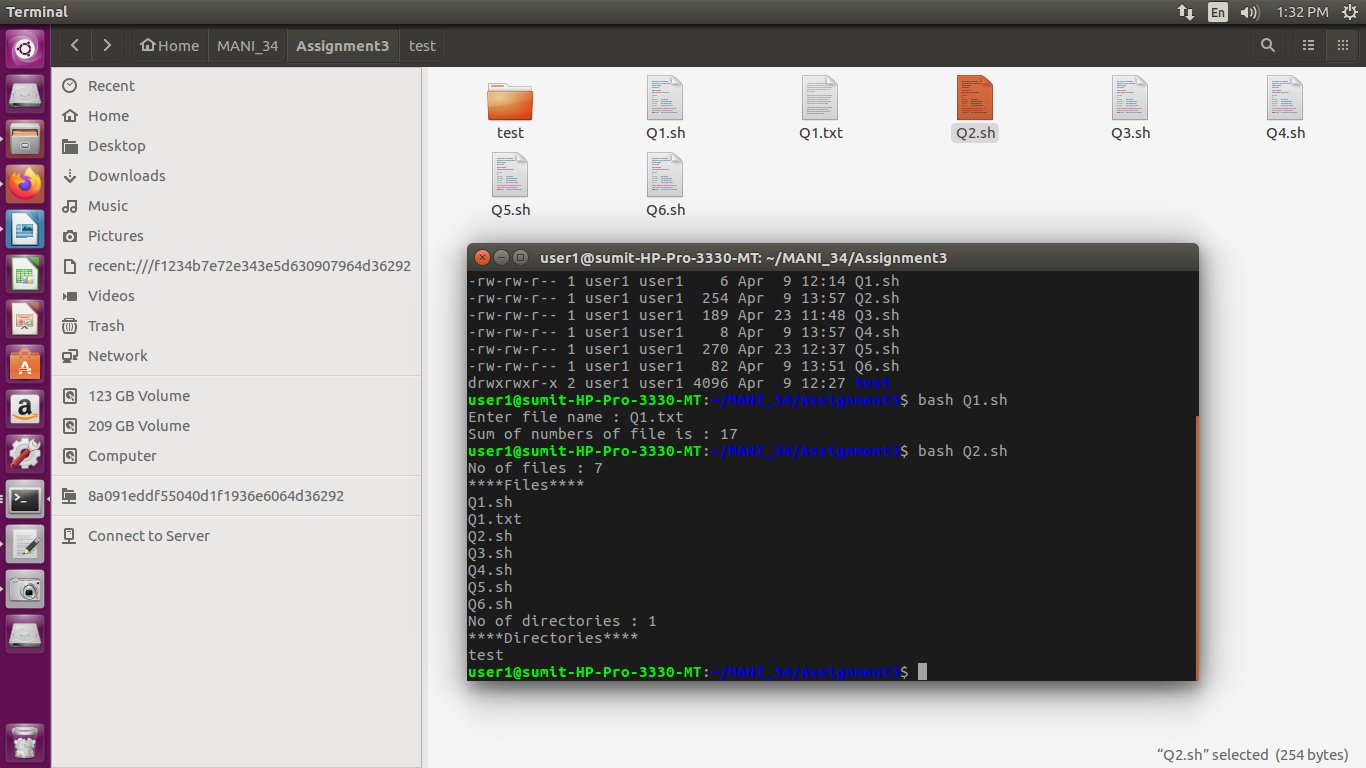
*then*

*echo "$i"*

*fi*

*done*

***Input/ Output***

******

**Q3:**

**W**rite a shell script that adds up the sizes reported by the ls command for the files in the current directory. The script should print out only the total number of bytes used.

**Code**

*size=0*

*for i in \**

*do*

*if [ -f $i ]*

*then*

*field=`ls -l $i | cut -d " " -f 5`*

*echo "$i : $field"*

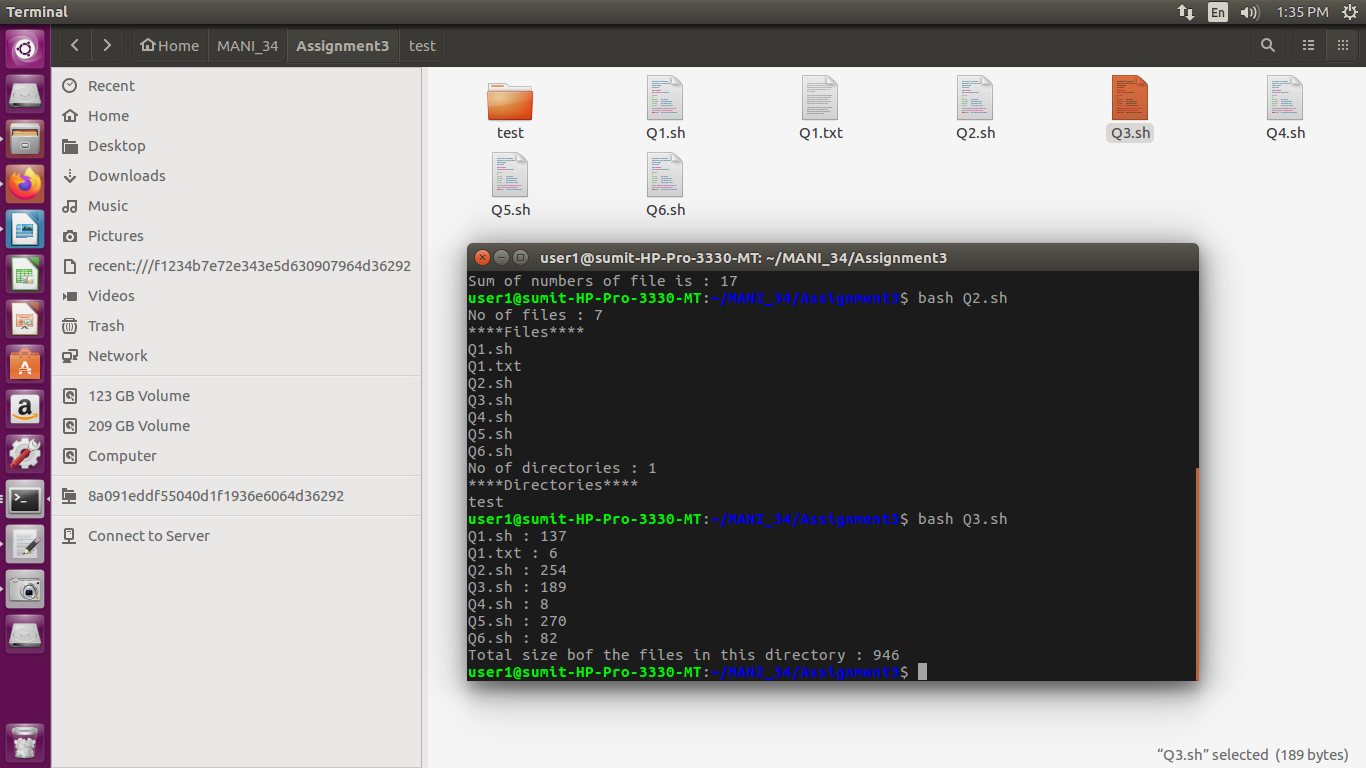
*size=$((size+field))*

*fi*

*done*

*echo "Total size bof the files in this directory : $size"*

***Input/ Output***

******

**Q4:**

**W**rite a shell script that deletes all temporary files(end with ~) in the current directory.

**Code**

*`rm \*~`*

**Q5:**

**W**rite a shell script to rename file having extension .sh to .exe.

**Code**

*files=`ls \*.sh`*

*for file in $files*

*do*

*filename=`echo $file | cut -d "." -f 1`*

*filename+=".exe"*

*`mv $file $filename`*

*done*

*echo `ls`*

*for file in \**

*do*

*if [ -f $file ]*

*then*

*filename=`echo $file | cut -d "." -f 1`*

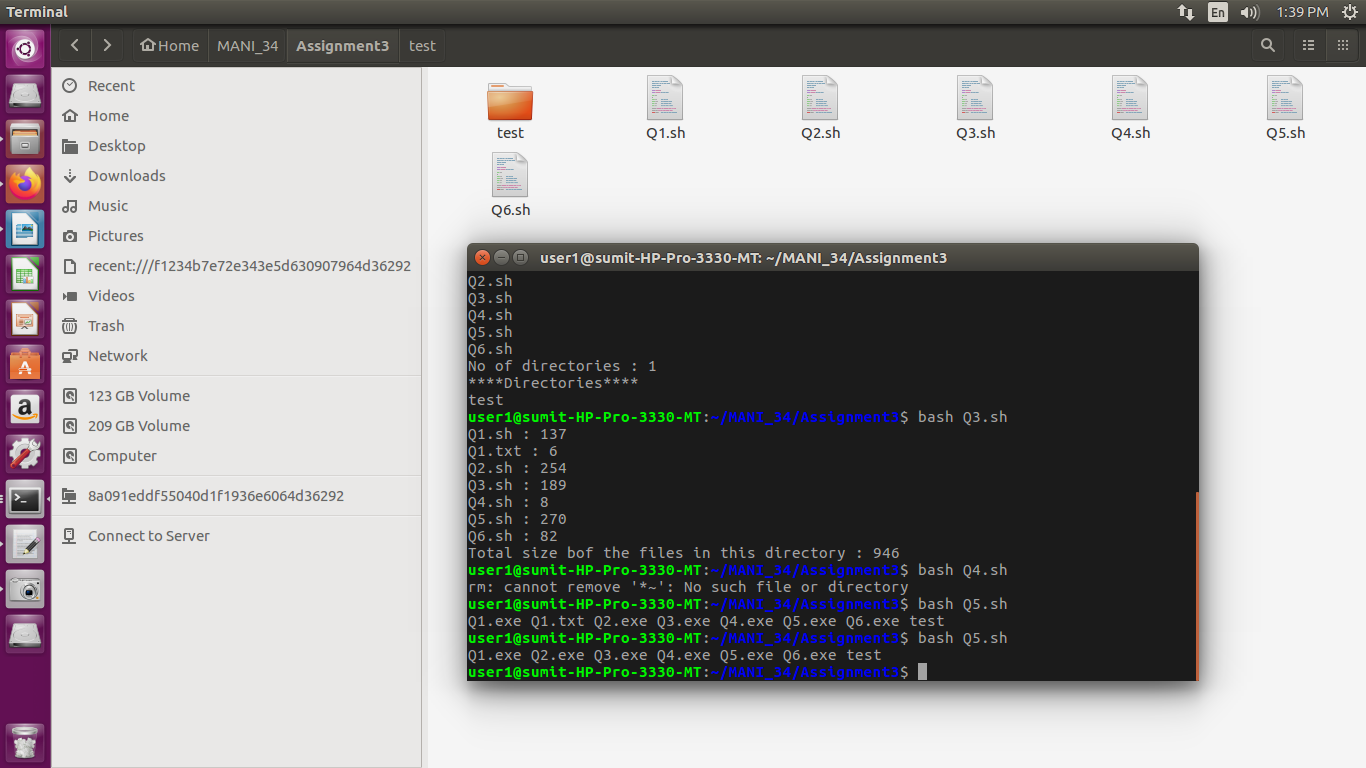
*filename+=".sh"*

*`mv $file $filename`*

*fi*

*done*

***Input/ Output***

******

**Q6:**

**W**rite a shell script to count number of shell scripts with .sh extension present in the current directory.

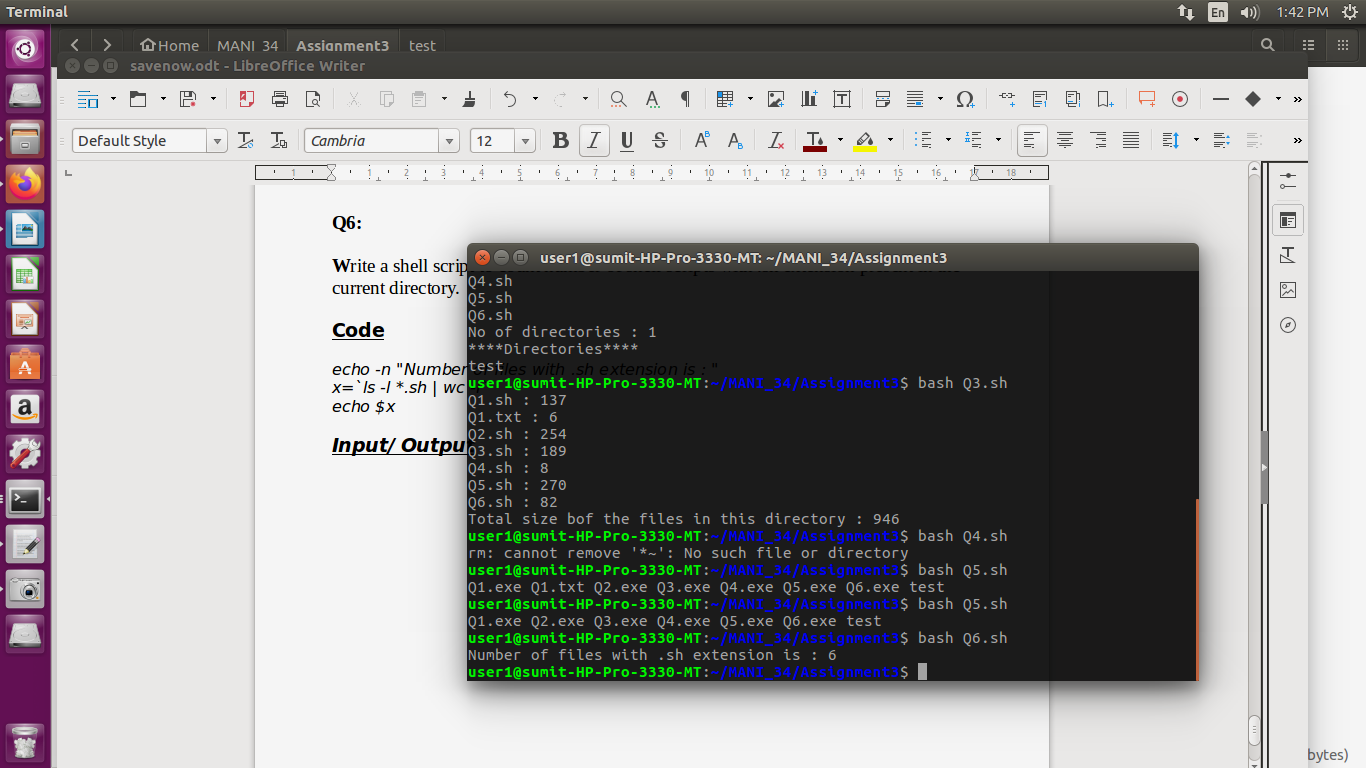
**Code**

*echo -n "Number of files with .sh extension is : "*

*x=`ls -l \*.sh | wc -l`*

*echo $x*

***Input/ Output***

******

**ASSIGNMENT 4**

**Q1:**

**W**rite a C program to create a child process. The parent process must wait until the child finishes. Both the processes must print their own pid and parent pid. Additionally, the parent process should print the exit status of the child.

**Code**

*#include<stdio.h>*

*#include<unistd.h>*

*#include<stdlib.h>*

*int main()*

*{*

*int cstatus;*

*printf("The pid of parent is : %d(before creating a child process)\n",getpid());*

*int x = fork(); //x will be 0 if it is a children process else it will be >0*

*if(x==0)*

*{*

*printf("Children Process : %d, my parent id : %d\n",getpid(),getppid());*

*}*

*else*

*{*

*int pid = wait(&cstatus); //this will wait untill the children process gets executed*

*if(pid==x)*

*{*

*printf("Child %d finished with status %d\n",x,cstatus);*

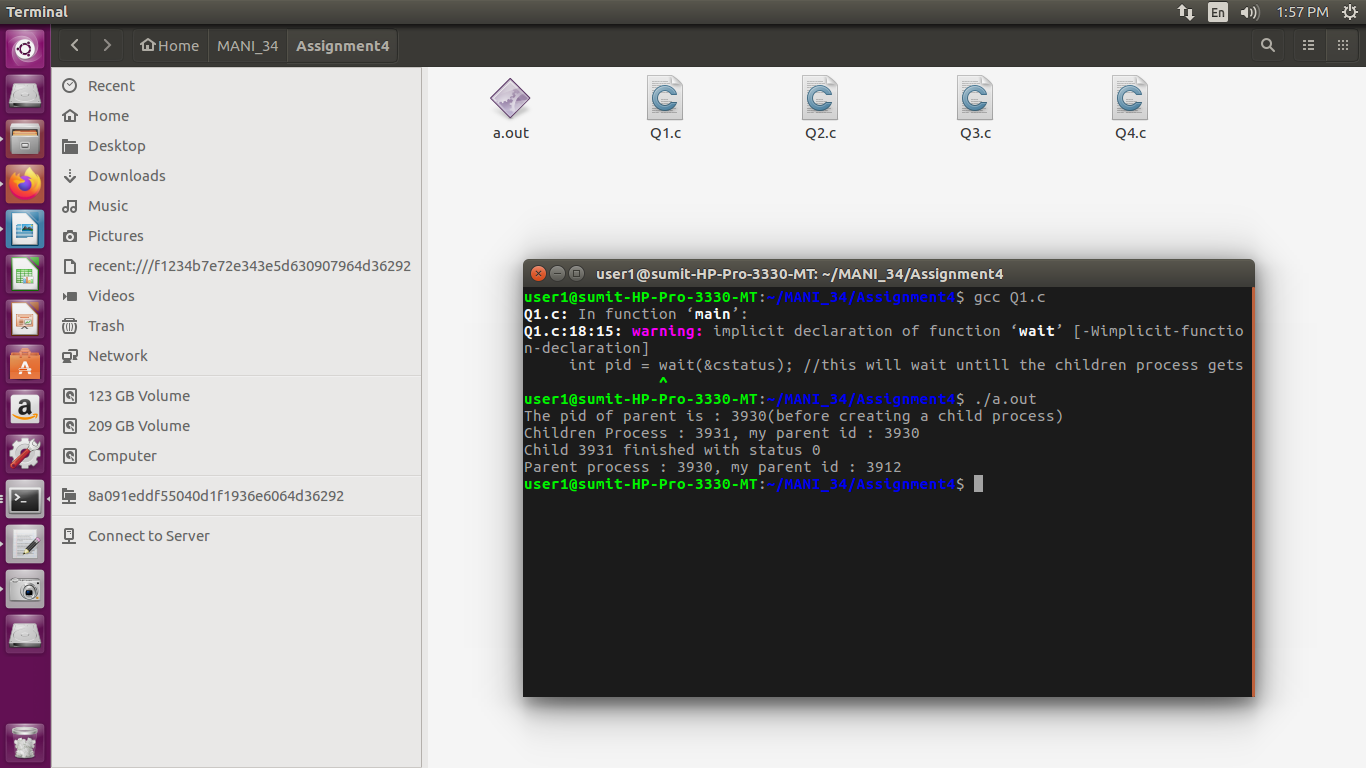
*printf("Parent process : %d, my parent id : %d\n",getpid(),getppid());*

*}*

*}*

*}*

***Input/ Output***

******

**Q2:**

**W**rite a C program which prints prime numbers between the range 1 to 1000000 by creating ten child processes and subdividing the task equally among all child processes i.e., the first child should print prime numbers in the range 1 to 100000, them second child in the range 100001 – 200000. The child processes must run in parallel and the parent process must wait until all the child processes finish.

**Code**

#include<stdio.h>

#include<unistd.h>

#include<stddef.h>

#include<stdlib.h>

int prime(int num)

{

if(num<=1) return 0;

for(int i=2;i<=num/2;i++)

{

if(num%i==0)

return 0;

}

return 1;

}

void range(int start, int end)

{

for(int i=start;i<=end;i++)

{

if(prime(i))

printf("%d ",i);

}

}

int main()

{

int cstatus,start=1,brk=100;

pid\_t pid,cpid;

for(int i=0;i<10;i++)

{

pid=fork();

if(pid==0)

{

printf("Child %d\n",i+1);

range(start,brk);

printf("\n");

exit(0);

}

else if(pid<0)

{

perror("fork");

exit(1);

}

else{

start=brk+1;

brk=brk+100;

}

}

for(int j=0;j<10;j++)

wait(NULL);

return 0;

}

**Output**

Child 1

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Child 2

101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Child 3

211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293

Child 4

307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397

Child 5

401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499

Child 6

503 509 521 523 541 547 557 563 569 571 577 587 593 599

Child 7

601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691

Child 8

701 709 719 727 733 739 743 751 757 761 769 773 787 797

Child 9

809 811 821 823 827 829 839 853 857 859 863 877 881 883 887

Child 10

907 911 919 929 937 941 947 953 967 971 977 983 991 997