**EXPERIMENT 6**

**AIM**

To study shell scripting.

**THEORY**

**What Is Shell Script**

->A shell script is a text [file](http://searchexchange.techtarget.com/definition/file) that contains a sequence of commands for a [UNIX](http://searchdatacenter.techtarget.com/definition/Unix)-based [operating system](http://whatis.techtarget.com/definition/operating-system-OS). It's called a shell script because it combines into a "script" in a single file a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time. The [shell](http://searchdatacenter.techtarget.com/definition/shell) is the operating system's command interpreter and the set of commands you use to communicate with the system. A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.

**Declaration of variable and execution**

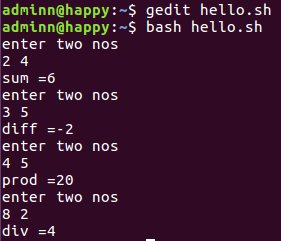
-> Declaration in Shell Script is very easy. Just write the variable name and assign the value to it. When accessing value we write variable name with ‘$’ sign. Script will give error if we try to use to different data type.

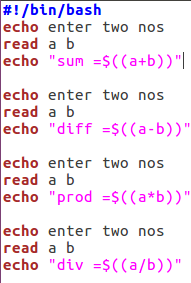
To execute the shell script file we write that following command in terminal.

**Sh filename.sh**

**Command Line argument.**

->We pass the command line argument directly after .sh extension with space. It is automatically stored in $1,$2… and so on.

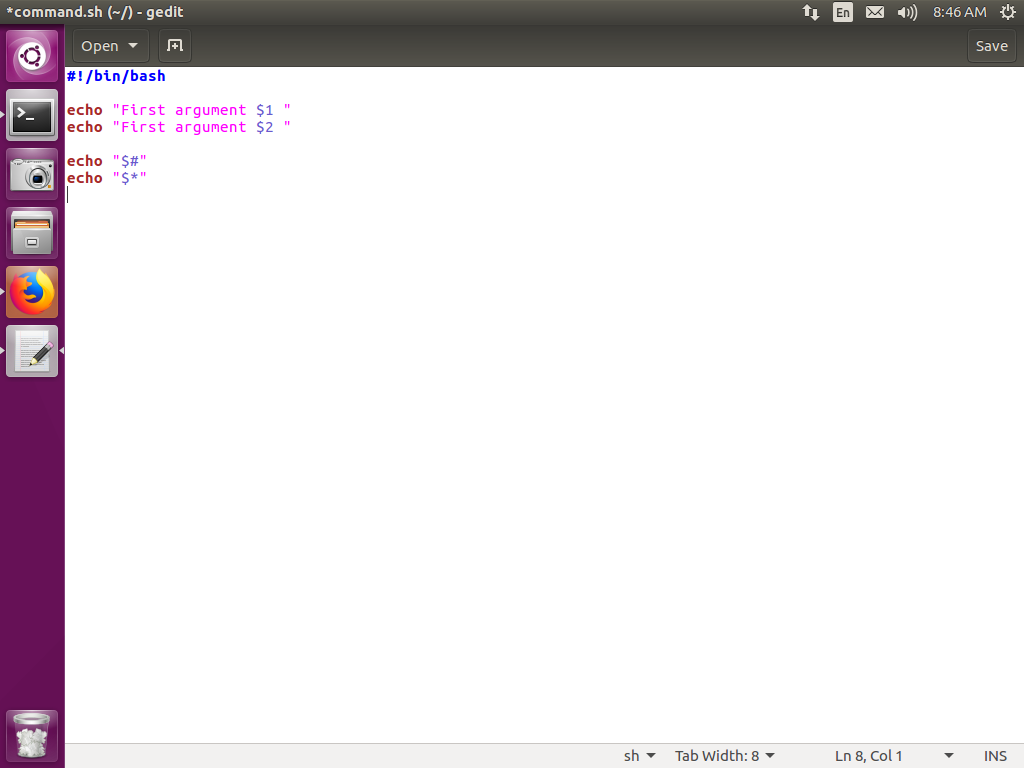


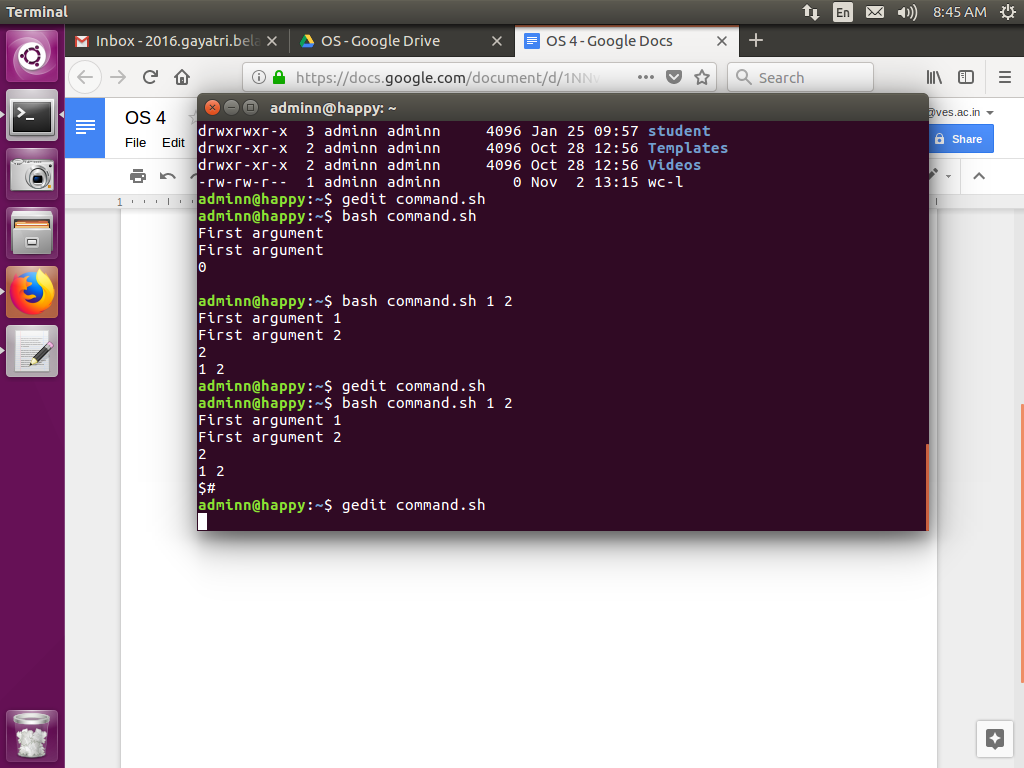


Command line arguments

$#-number of arguments

$\*-displays all the arguments



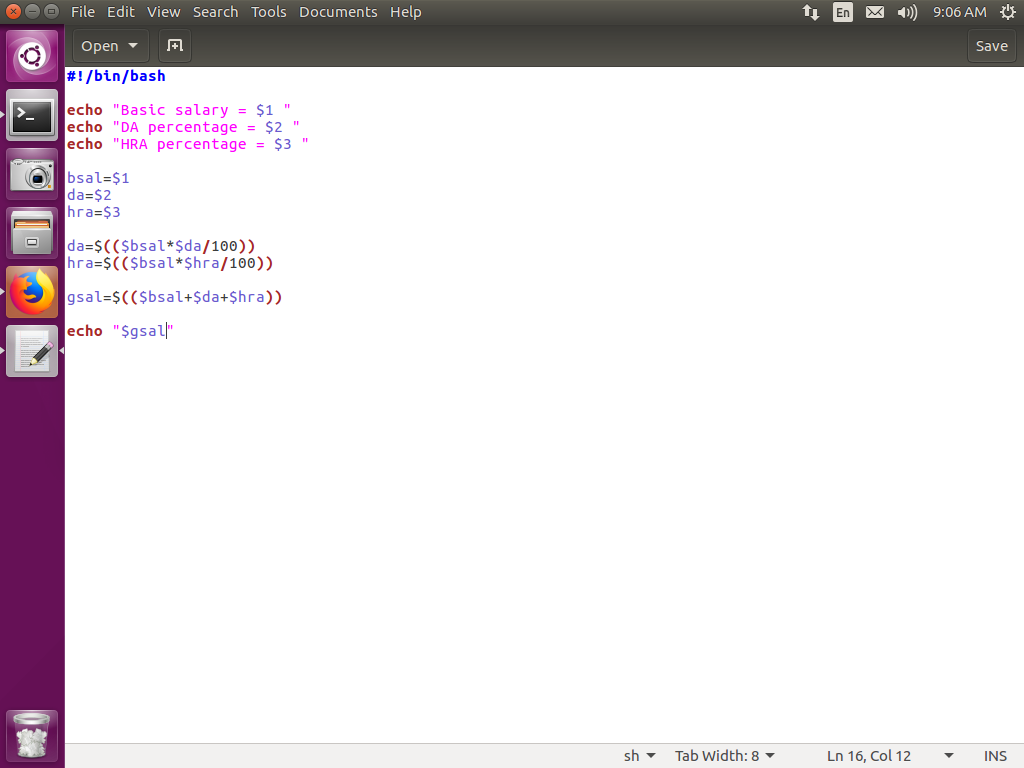


Salary calculation

Gross\_sal=bsal+da+hra

da=50%bsal

hra=30%bsal



CONTROL STRUCTURES

Relational operators supported by shell scripts :

-gt greater than

-lt less than

-eq equal

-ne not equal to

-ge greater than or equal to

-le less than or equal to

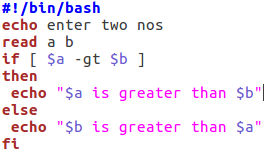
1. If control structure

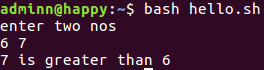
If [ condition ]

then

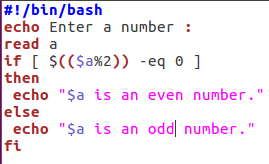
Block of code

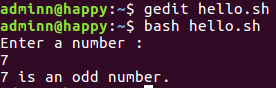
Fi



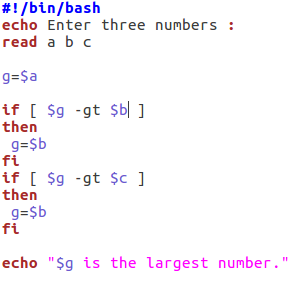


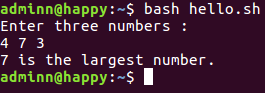
Program to determine entered number is even or odd

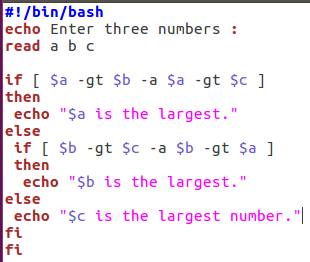


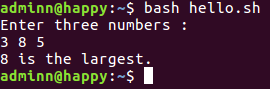


Program to find largest among three numbers

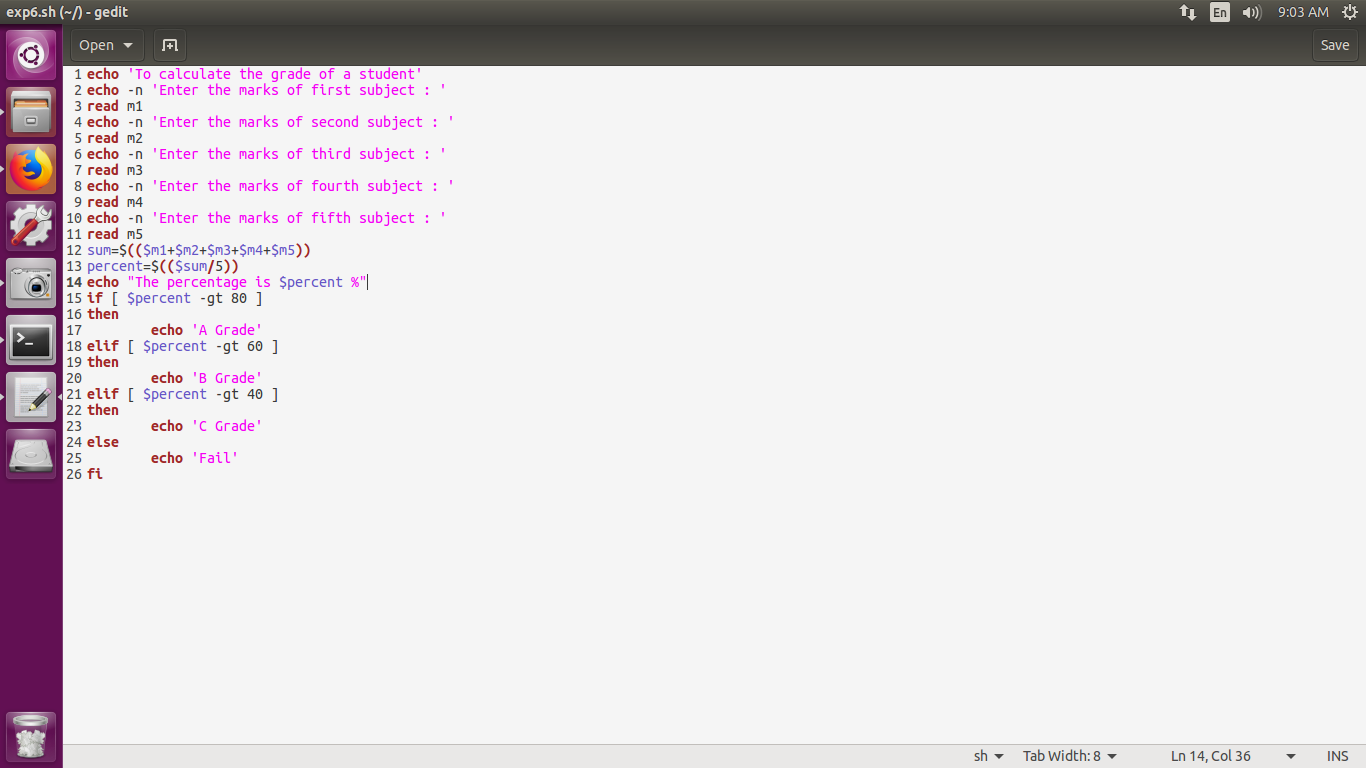


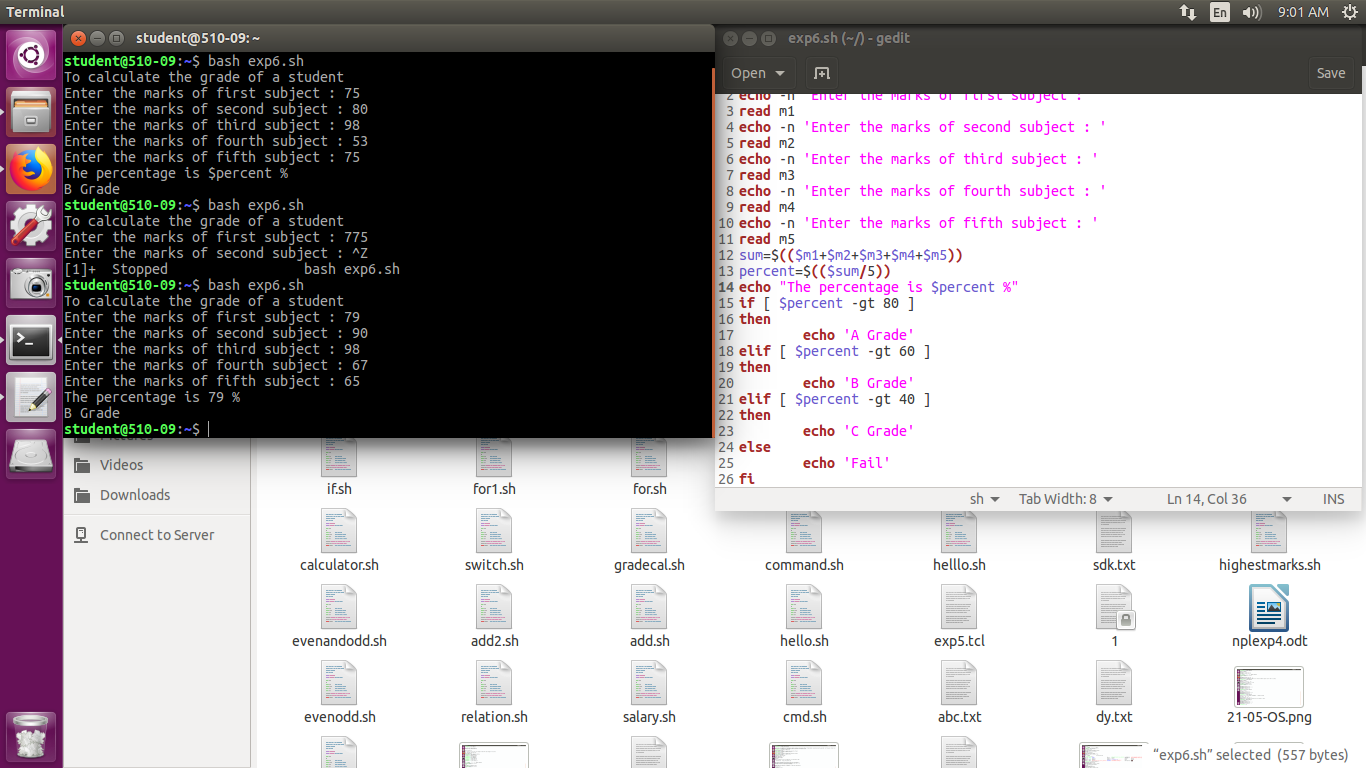






To calculate the grade





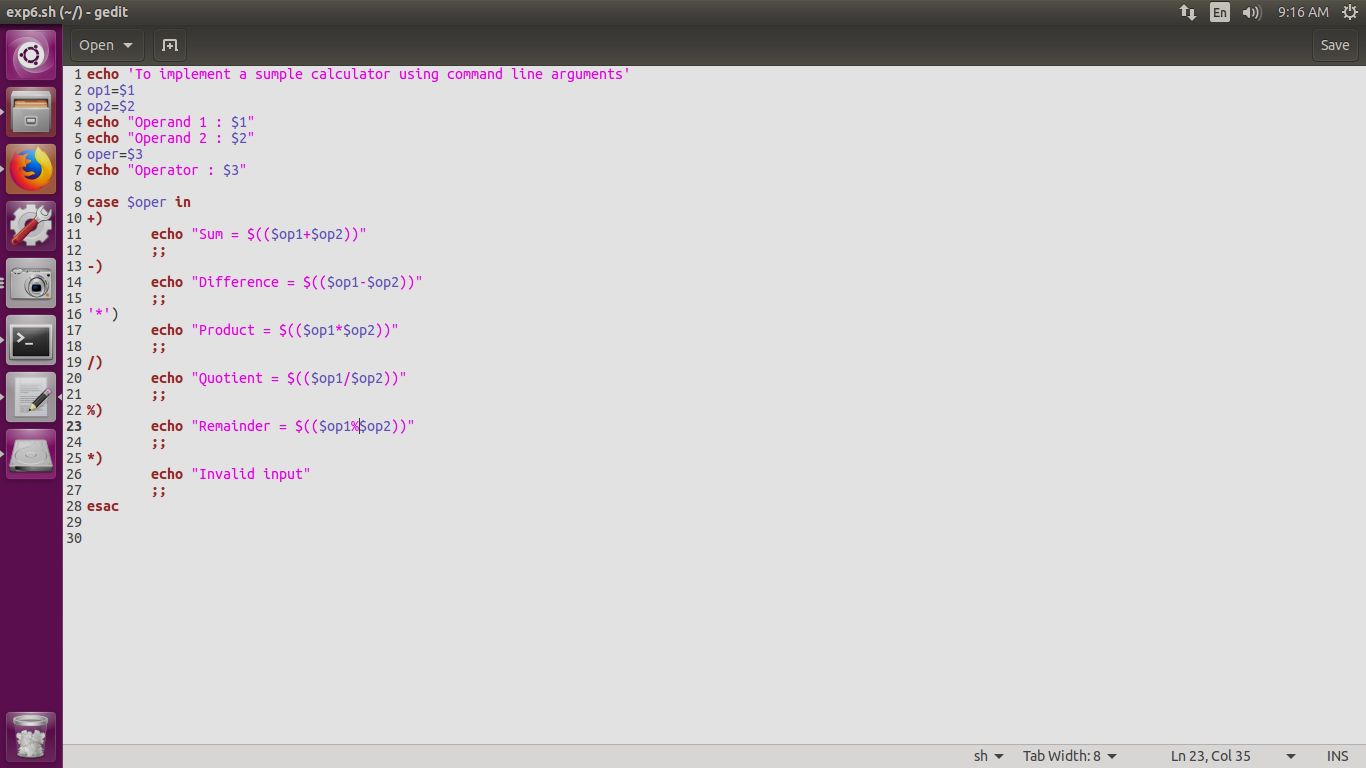
SWITCH CASE

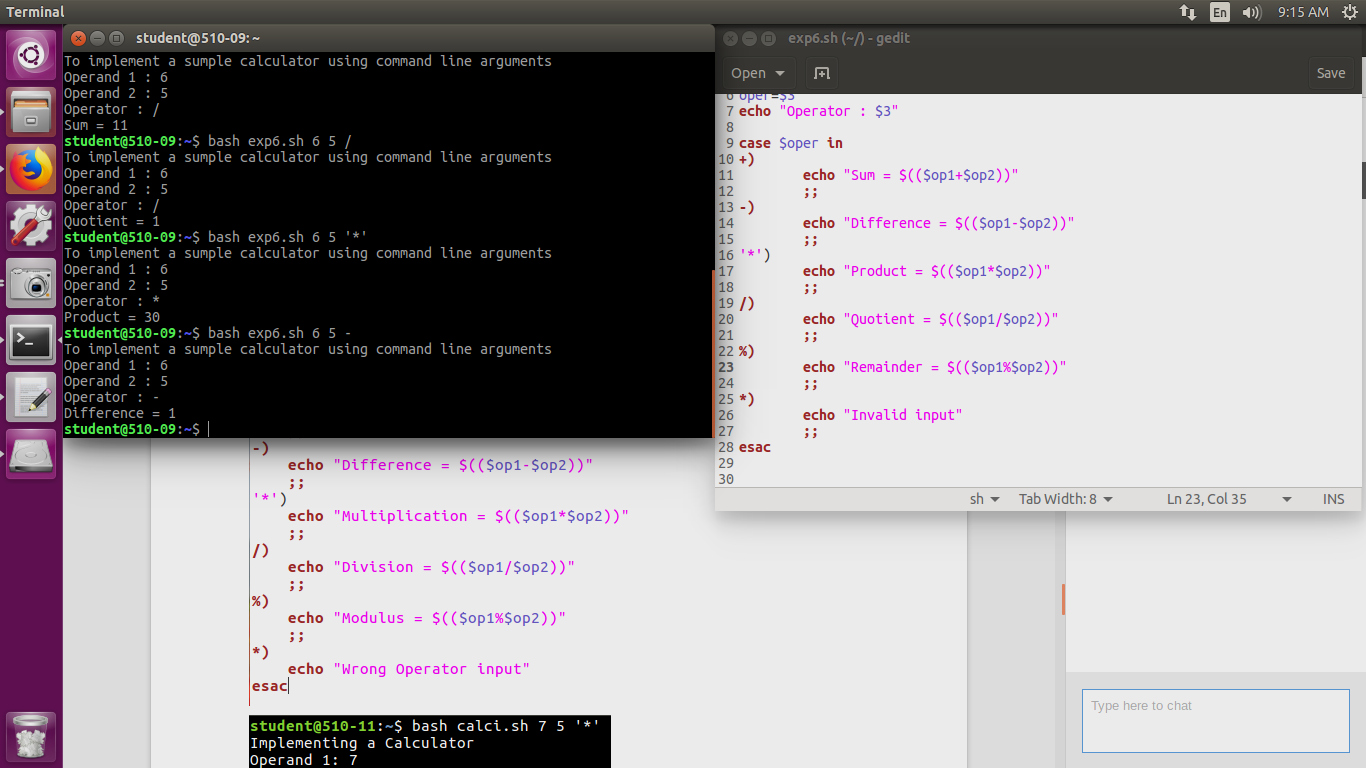
-o or

-a and

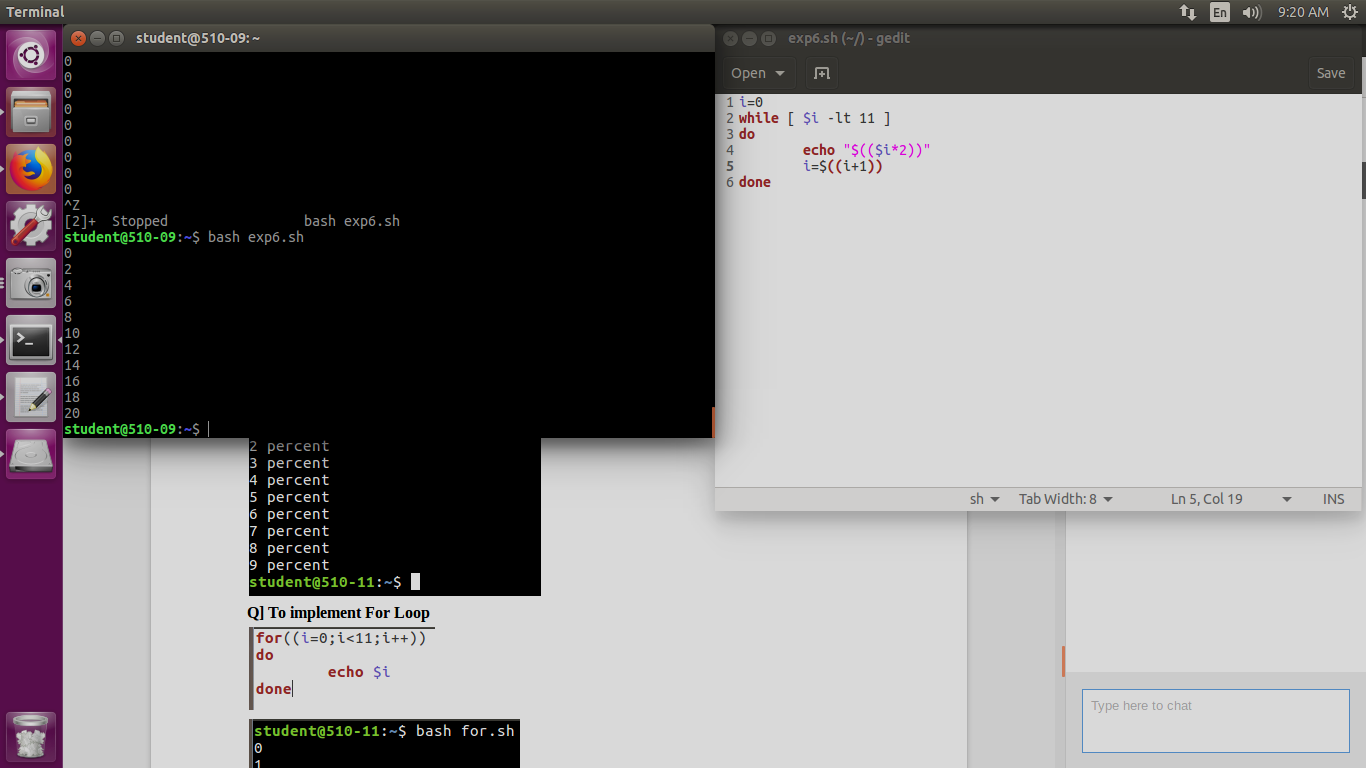
! not

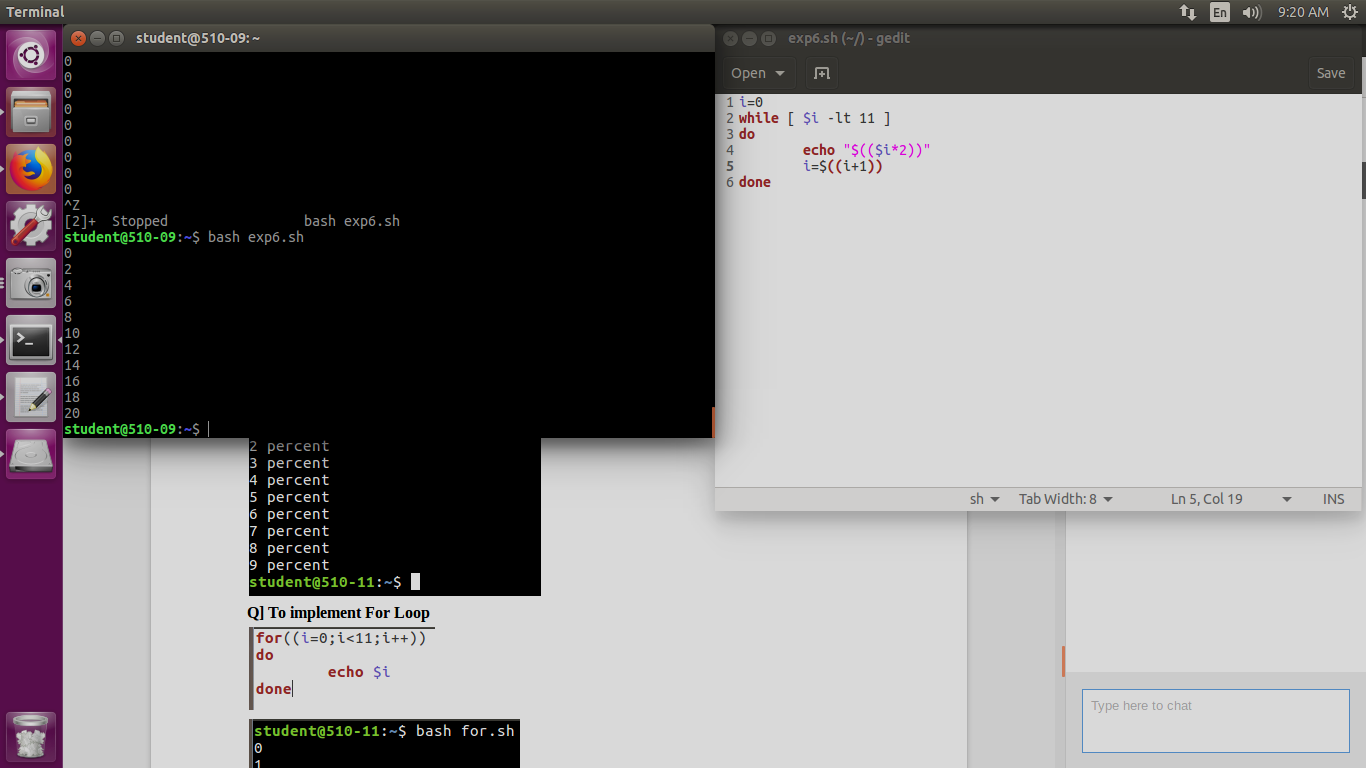
Simple calculator using switch case



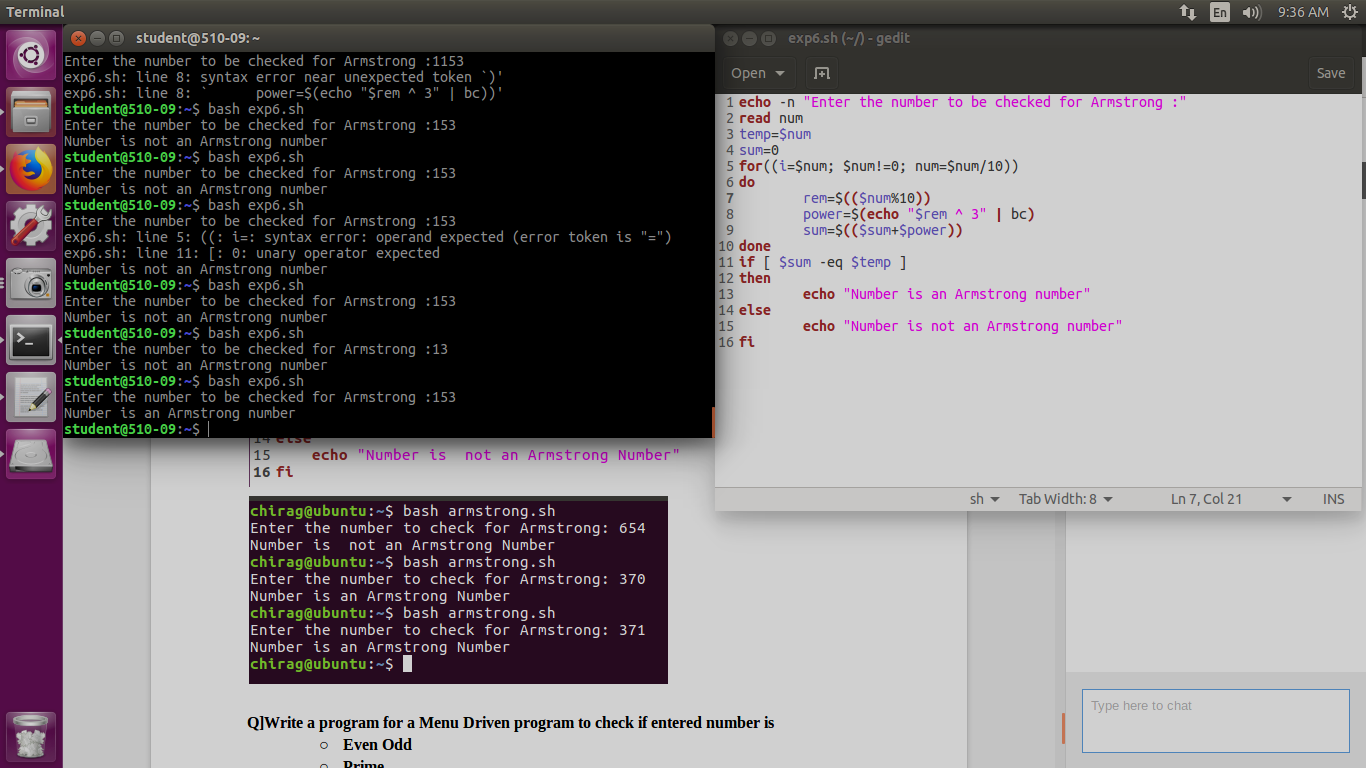


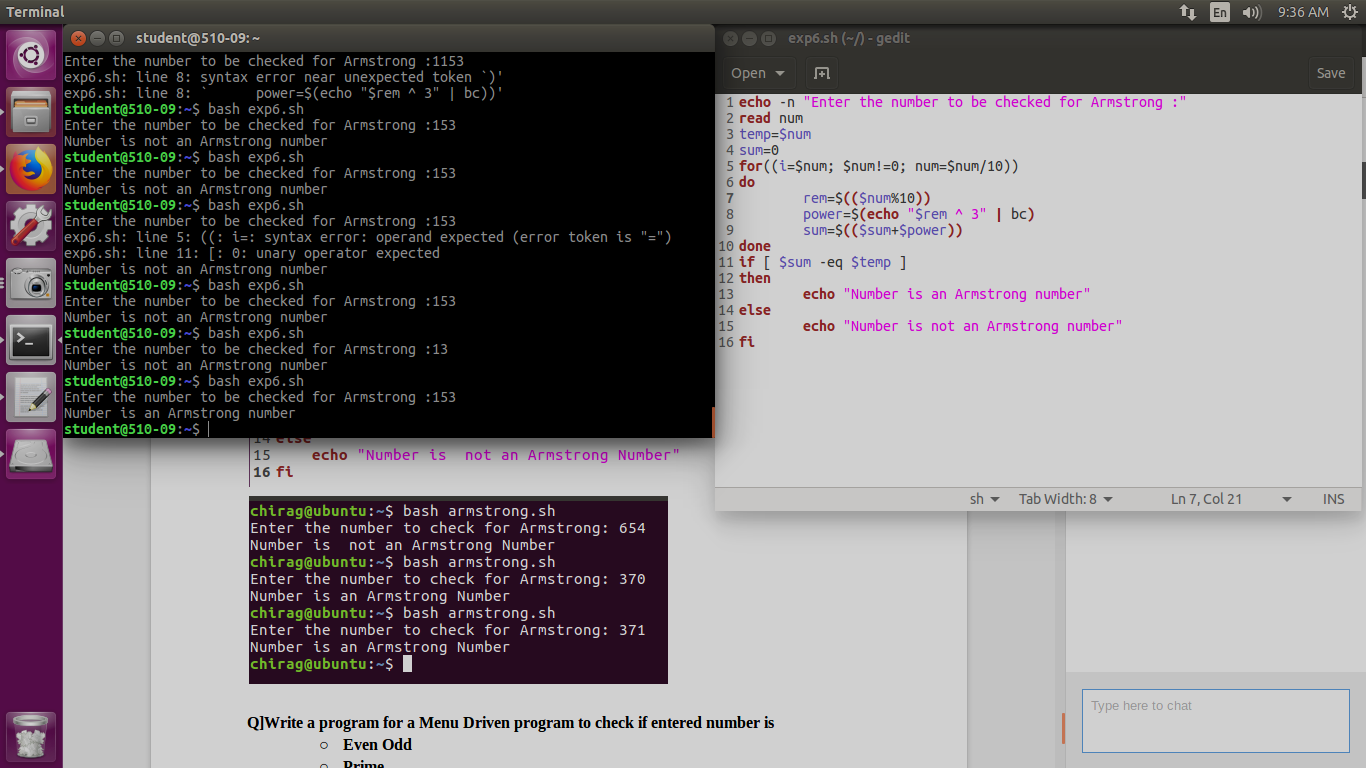
While loop





Checking Armstrong number using for





**Q]Write a program for a Menu Driven program to check if entered number is**

* + **Even Odd**
  + **Prime**
  + **Palindrome**

#Menu Driven Program

ch=1

while [ $ch -eq 1 ]

do

echo "\*\*\*\*\* MENU \*\*\*\*\*"

echo "1. Check Even Odd"

echo "2. Check Palindrome"

echo "3. Check Prime"

echo "4. Exit"

echo -n "Enter your choice: "

read choice

case $choice in

1)

#Even Odd

echo -n "Enter the Number to check Even/Odd: "

read num

if [ $(($num%2)) -eq 0 ]

then

echo "Even Number"

else

echo "Odd Number"

fi

;;

2)

#Palindrome

echo -n "Enter the Number to check Palindrome: "

read num

number=$num

reverse=0

while [ $num -gt 0 ]

do

a=$(( $num % 10 ))

num=$(( $num / 10 ))

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

done

echo "Reverse is $reverse"

if [ $number -eq $reverse ]

then

echo "Number is palindrome"

else

echo "Number is not palindrome"

fi

;;

3)

#Prime Number

echo -n "Enter a number: "

read num

i=2

if [ $num -lt 2 ]

then

echo "$num is not prime"

exit 0

fi

while [ $i -le $(($num/2)) -a $rem!=0 ]

do

rem=$(($num%$i))

i=$(($i+1))

done

if [ $rem -eq 0 ]

then

echo "$num is not prime"

else

echo "$num is prime"

fi

;;

4)

echo "Thank You!"

exit

;;

\*)

echo "Wrong Input"

esac

echo -n "Do you wish to continue? (1/0): "

read ch

if [ $ch -eq 0 ]

then

exit

fi

done

#Prime Number

echo -n "Enter a number: "

read num

i=2

while [ $i -lt $num ]

do

if [ $(($num%$i)) -eq 0 ]

then

echo "$num is not a prime number"

exit

fi

i=$(($i+1))

done

