|  |  |
| --- | --- |
| **Exp. No: 2a** | **SHELL SCRIPTING** |
|  | **ARITHMETIC OPERATORS** |

**Aim:**

To write a UNIX shell program to find the arithmetic operations for the given numbers.

**Algorithm:**

Step 1: Start the program.

Step 2: Read two values.

Step 3: Do the arithmetic operation (i.e. addition, subtraction, multiplication, division) for the

given values.

Step 4: Print the arithmetic operation values.

Step 5: Stop the program.

**Program:**

echo "Enter x value:"

read x

echo "Enter y value:"

read y

echo "Addition"

let "z = $(( x + y ))"

echo "z= $z"

echo "Subtraction"

let "z = $((x - y ))"

echo "z= $z"

echo "Multiplication"

let "z = $(( x \* y ))"

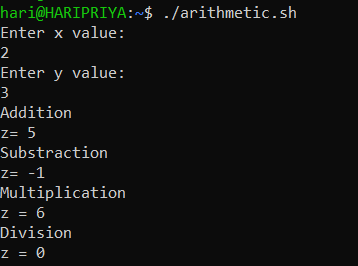
echo "z = $z"

echo "Division"

let "z = $(( x / y ))"

echo "z = $z"

**OUTPUT:**



|  |  |
| --- | --- |
| **Exp. No: 2b** | **SHELL SCRIPTING** |
|  | **GREATEST OF TWO NUMBERS** |

**Aim:**

To write a UNIX shell program to find the greatest of two numbers.

**Algorithm:**

Step 1: Start the program.

Step 2: Read two values.

Step 3: check the two values which is greater.

Step 4: Print the greater value.

Step 5: Stop the program.

**Program:**

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

if [ $num1 -gt $num2 ]

then

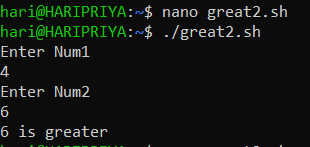
echo "$num1 is greater"

else

echo "$num2 is greater"

fi

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2c** | **SHELL SCRIPTING** |
|  | **FINDING ODD OR EVEN** |

**Aim:**

To write a UNIX shell program to find the sum of two numbers.

**Algorithm:**

Step 1: Start the program.

Step 2: Read the values.

Step 3: Check whether the number is odd or even.

Step 4: Print the value.

Step 5: Stop the program.

**Program:**

echo -n "Enter a number:"

read n

echo -n "RESULT: "

if [ `expr $n % 2` == 0 ]

then

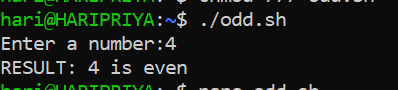
echo "$n is even"

else

echo "$n is Odd"

fi

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2d** | **SHELL SCRIPTING** |
|  | **GREATEST OF 3 NUMBERS** |

**Aim:**

To write a UNIX shell program to find the greatest of three numbers.

**Algorithm:**

Step 1: Start the program.

Step 2: Read two values.

Step 3: check the three values which is greater.

Step 4: Print the greater value.

Step 5: Stop the program

**Program:**

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo "$num1 is greater"

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

then

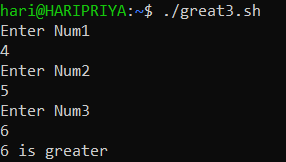
echo "$num2 is greater"

else

echo "$num3 is greater"

fi

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2e** | **SHELL SCRIPTING** |
|  | **SIMPLE CALCULATOR** |

**Aim:**

To write a UNIX shell program for a simple calculator.

**Algorithm:**

Step 1: Start the program.

Step 2: Read two values.

Step 3: Do the arithmetic operation (i.e. addition, subtraction, multiplication, division) for the

given values.

Step 4: Print the values obtained by the program.

Step 5: Stop the program

**Program:**

echo "Enter Two numbers : "

read a

read b

echo "Enter Choice :"

echo "1. Addition"

echo "2. Subtraction"

echo "3. Multiplication"

echo "4. Division"

read ch

case $ch in

1)res=`echo $a + $b | bc`

;;

2)res=`echo $a - $b | bc`

;;

3)res=`echo $a \\* $b | bc`

;;

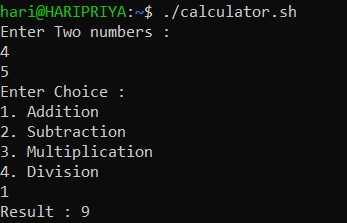
4)res=`echo "scale=2; $a / $b" | bc`

;;

esac

echo "Result : $res"

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2f** | **SHELL SCRIPTING** |
|  | **VOWEL OR CONSTANT** |

**Aim:**

To write a UNIX shell program to find the Given character is vowel or constant.

**Algorithm:**

Step 1: Start the program.

Step 2: Read the values.

Step 3: check the character is constant or vowel.

Step 4: Print the value.

Step 5: Stop the program.

**Program:**

read -p "Enter something: " char

if [[ "$char" == \*[AEIOUaeiou]\* ]]; then

echo "vowel"

else

echo "consonant"

fi

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2g** | **SHELL SCRIPTING** |
|  | **SUM OF N NATURAL NUMBERS** |

**Aim:**

To write a UNIX shell program to find the sum of N natural numbers.

**Algorithm:**

Step 1: Start the program.

Step 2: Read the values.

Step 3: Add the n number of values.

Step 4: Print the value.

Step 5: Stop the program.

**Program:**

echo -n "Enter nth number's value:"

read digit

t=1

total=0

while test $t -le $digit

do

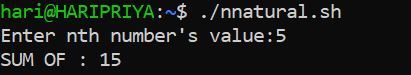
total=`expr $total + $t`

t=`expr $t + 1`

done

echo "SUM OF $DIGIT: $total "

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 2h** | **SHELL SCRIPTING** |
|  | **FACTORIAL OF A GIVEN NUMBER** |

**Aim:**

To write a UNIX shell program to find the factorial of a given number.

**Algorithm:**

Step 1: Start the program.

Step 2: Read the values.

Step 3: Do the factorial operation of the given number .

Step 4: Print the value.

Step 5: Stop the program.

**Program:**

echo "Enter a number"

read num

fact=1

while [ $num -gt 1 ]

do

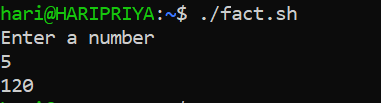
fact=$((fact \* num)) #fact = fact \* num

num=$((num - 1)) #num = num - 1

done

echo $fact

**Output:**



|  |  |
| --- | --- |
| **Exp. No: 2i** | **SHELL SCRIPTING** |
|  | **FIBONACCI SERIES** |

**Aim:**

To write a UNIX shell program to find the Fibonacci series of a given number.

**Algorithm:**

Step 1: Start the program.

Step 2: Read the values.

Step 3: Do the Fibonacci operation of the given number .

Step 4: Print the value.

Step 5: Stop the program.

**Program:**

echo "Enter a number:"

read N

a=0

b=1

echo "The Fibonacci series is : "

for (( i=0; i<N; i++ ))

do

echo -n "$a "

fn=$((a + b))

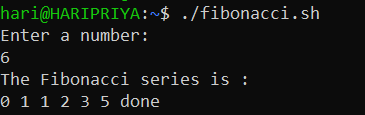
a=$b

b=$fn

done

echo "done"

**Output:**

****

|  |  |
| --- | --- |
| **Exp. No: 3a** | **UNIX SYSTEM CALLS** |
|  |

**Aim:**

To write a C program to implement the basic UNIX System Calls.

**Algorithm:**

Step1: start the program.

Step2: Execute the fork () system call and then stop.

Step3: Execute the exec () system call and then stop.

Step4: Execute the wait () and sleep () system calls and then stop.

Step5: Execute the stat () system call.

Step6: stop the program.

**Program 1:**

**fork() System Call**

#include<stdio.h>

#include<unistd.h>

int main(){

printf("Unix system call");

fork();

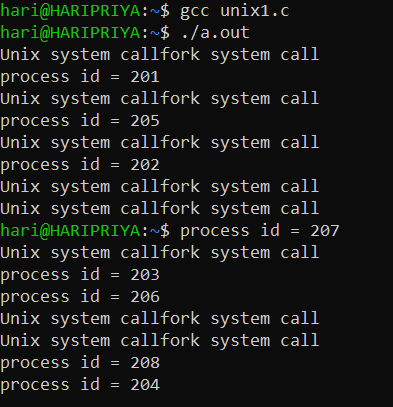
fork();

fork();

printf("fork system call\n");

printf("process id = %d\n",getpid());}

**Output:**

****

**2)fork() System Call**

#include<stdio.h>

#include<unistd.h>

int main()

{

pid\_t p;

p=fork();

if(p<0)

printf("Error in creating process\n.");

else if(p==0)

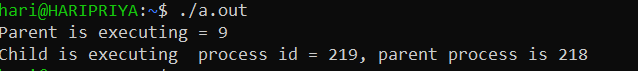
printf("Child is executing process id = %d, parent process is %d\n", getpid(),getppid());

else

printf("Parent is executing = %d\n",getppid());

}

**Output:**

****

**Exec() system call**

**Program:**

**Add.c**

#include<stdio.h>

#include<unistd.h>

int main()

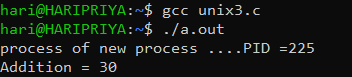
{

printf("process of new process ....PID =%d\n",getpid());

printf("Addition = %d\n",20+10);

}

**Output:**



**Exec.c**

#include<stdio.h>

#include<unistd.h>

int main()

{

printf("Current program in execution ... PID = %d\n",getpid());

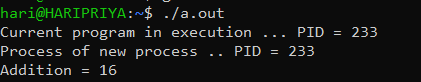
char \*a[]={"./add",NULL};

execv(a[0],a);

printf("Back to current process");

}

**Output:**

****

Failure.c

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc,char \* argv[])

{

execvp(argv[1],&argv[1]);

perror("exec failure");

exit(1);

}

**Output:**

****

**Wait () system call**

**Program:**

#include<stdio.h>

#include<unistd.h>

int main(){

pid\_t p;

int a,b;

p=fork();

if(p<0)

printf("ERROR");

else if (p==0)

{

printf("child process PID = %d\n",getpid());

printf("Enter a & b value for addition \n");

scanf("%d %d ",&a,&b);

printf("call by child = %d\n",a+b);

}

else

{

printf("Enter a & b for mul \n");

scanf("%d %d",&a,&b);

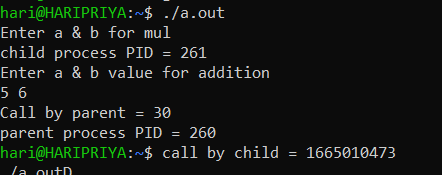
printf("Call by parent = %d\n",a\*b);

printf("parent process PID = %d\n",getpid());

}

**}**

**Output:**

****

**Sleep() system call**

**Program:**

#include<stdio.h>

#include<unistd.h>

int main()

{

pid\_t p;

int a,b;

p=fork();

if(p<0)

printf("ERROR");

else if (p==0)

{

printf("child process PID = %d\n",getpid());

printf("Enter a & b value for addition \n");

scanf("%d %d",&a,&b);

printf("call by child = %d\n",a+b);

}

else

{

wait();

printf("Enter a & b for mul \n");

scanf("%d %d",&a,&b);

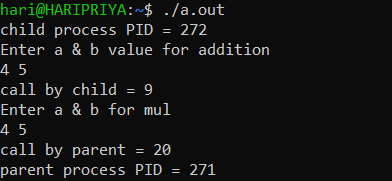
printf("call by parent = %d\n",a\*b);

printf("parent process PID = %d\n",getpid());

}

}

**Output:**



**Stat() system call**

**Program:**

#include<stdio.h>

#include<unistd.h>

#include<sys/stat.h>

#include<sys/types.h>

int main()

{

struct stat buf;

stat("unix8.c",&buf);

printf("FILE MODE = %o\n",buf.st\_mode);

printf("FILE SIZE = %ld\n",buf.st\_size);

printf("FILE BLOCK SIZE = %ld\n",buf.st\_blksize);

printf("PROCESS ID =%d\n",buf.st\_gid);

printf("NO OF BLOCKS ALLOCATED = %ld\n",buf.st\_blocks);

printf("NO OF HARD LINK = %u\n",(unsigned int)buf.st\_nlink);

printf("File permissions User\n");

printf((buf.st\_mode & S\_IRUSR)?"r":"-");

printf((buf.st\_mode & S\_IWUSR)?"w":"-");

printf((buf.st\_mode & S\_IXUSR)?"x":"-");

printf("\nFile permissions Group\n");

printf((buf.st\_mode & S\_IRGRP)?"r":"-");

printf((buf.st\_mode & S\_IWGRP)?"x":"-");

printf((buf.st\_mode & S\_IXGRP)?"x":"-");

printf("\nFile Permissions Other\n");

printf((buf.st\_mode & S\_IROTH)?"r":"-");

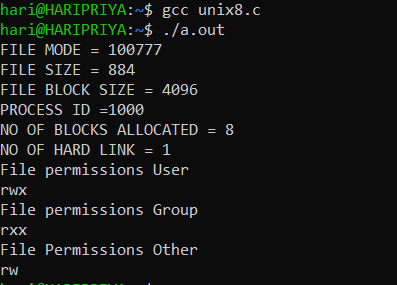
printf((buf.st\_mode & S\_IWOTH)?"w":"-");

printf("\n");

return 0;

}

**Output:**

****