CSE2005-Operating Systems Lab

Assessment-1 Questions

1. **Basic Linux Commands**
2. **Shell Programming**
    a. **Handling the command line arguments**
    b. **String reversal**
    c. **If-Else, Nested If Else, Switch cases in shell**
3. **Parent child process creation using fork( ) and exec() system call**
    A) Checking the Process Identifier
    B) Assigning new task to child
    C) Providing the path name and program name to exec()
    D) Synchronizing Parent and child process using wait()
4. **The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm:**
    n = n/2, if n is even n = 3 × n + 1, if n is odd
    The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1.
    For example, if n = 35, the sequence is 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1.
    Write a C program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the command line.
    For example, if 8 is passed as a parameter on the Command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence.
    Have the parent invoke the wait() call to wait for the child process to complete before exiting the program (High).

Go To Top

Answer 1) **Basic Linux Commands**

Command	Function
pwd	pwd command is used to find out the path of the current working directory (folder). The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash (/)
cd	cd is used to navigate through the Linux files and directories. cd (directoryName) to move down the immediate directory cd .. to move one directory up cd to go straight to the home folder cd- to move to your previous directory
ls	The ls command is used to view the contents of a directory. By default, this command will display the contents of your current working directory. ls -l will list all the files and directory in the directory along with their modification permissions. ls -R will list all the files in the sub-directories as well. ls -a will show the hidden files. ls -al will list the files and directories with detailed information like the permissions, size, owner, etc.
cat	It is used to list the contents of a file on the standard output (sdout). To run this command, type cat followed by the file's name and its extension. For instance: cat file.txt.
vi	vi command is to create new files and open and edit existing files.
mkdir	mkdir command to make a new directory
rmdir	rmdir command is used to delete a directory. However, rmdir only allows you to delete empty directories.
grep	grep lets us search through all the text in a given file.
sudo	Short for "SuperUser Do", this command enables you to perform tasks that require administrative or root permissions.
chmod	chmod is used to change the read, write, and execute permissions of files and directories.
ping	ping command is used to check our connectivity status to a server.
man	Shows a documentation on another linux commands for us to learn how to use those commands
echo	This command is used to move some data into a file.
rm	The rm command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to rmdir — use rm -r

Go To Top

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet1
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ cd
gaurav1020@DESKTOP-R0RPIEK:~$ pwd
/home/gaurav1020
gaurav1020@DESKTOP-R0RPIEK:~$ cd cyclesheet1
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls
Fibonacci.sh              Multiplication_Table.sh  Sample.txt   reverse_and_sum.sh
Login_details_isolate.sh  Read_only_permission.sh  Sample2.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ cat Sample.txt
Hello World!
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ mkdir Test
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls
Fibonacci.sh              Multiplication_Table.sh  Sample.txt   Test
Login_details_isolate.sh  Read_only_permission.sh  Sample2.txt  reverse_and_sum.sh
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ rmdir Test
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls
Fibonacci.sh              Multiplication_Table.sh  Sample.txt   reverse_and_sum.sh
Login_details_isolate.sh  Read_only_permission.sh  Sample2.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ rm Sample2.txt
rm: remove write-protected regular file 'Sample2.txt'? y
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls
Fibonacci.sh              Multiplication_Table.sh  Sample.txt
Login_details_isolate.sh  Read_only_permission.sh  reverse_and_sum.sh
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ grep Hello Sample.txt
Hello World!
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls -l Sample.txt
-r--r--r-- 1 gaurav1020 gaurav1020 13 Feb 23 16:12 Sample.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ chmod 744 Sample.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ls -l Sample.txt
-rwxr--r-- 1 gaurav1020 gaurav1020 13 Feb 23 16:12 Sample.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ echo Hi,Gaurav >> Sample.txt
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ cat Sample.txt
Hello World!
Hi,Gaurav
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ping -c 3 google.com
PING google.com (216.58.196.206) 56(84) bytes of data.
64 bytes from del03s06-in-f14.1e100.net (216.58.196.206): icmp_seq=1 ttl=116 time=210 ms
64 bytes from del03s06-in-f14.1e100.net (216.58.196.206): icmp_seq=2 ttl=116 time=64.4 ms
64 bytes from del03s06-in-f14.1e100.net (216.58.196.206): icmp_seq=3 ttl=116 time=43.3 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 43.278/105.819/209.742/73.990 ms
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet1
Hello World!
Hi,Gaurav
~
~
~
~
~
~
~
```

Go To Top

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet1
VIM(1)                          General Commands Manual                          VIM(1)

NAME
       vim - Vi IMproved, a programmer's text editor

SYNOPSIS
       vim [options] [file ..]
       vim [options] -
       vim [options] -t tag
       vim [options] -q [errorfile]

       ex
       view
       gvim gview evim eview
       rvim rview rgvim rgview

DESCRIPTION
       Vim  is  a  text editor that is upwards compatible to Vi.  It can be used to edit all kinds of
       plain text.  It is especially useful for editing programs.

       There are a lot of enhancements above Vi: multi level undo, multi windows and buffers,  syntax
       highlighting, command line editing, filename completion, on-line help, visual selection, etc..
       See ":help vi_diff.txt" for a summary of the differences between Vim and Vi.

       While running Vim a lot of help can be obtained from the on-line help system, with the ":help"
       command.  See the ON-LINE HELP section below.

       Most often Vim is started to edit a single file with the command

            vim file

       More generally Vim is started with:

            vim [options] [filelist]

       If the filelist is missing, the editor will start with an empty buffer.  Otherwise exactly one
       out of the following four may be used to choose one or more files to be edited.

       file ..     A list of filenames.  The first one will be the current file  and  read  into  the
                   buffer.   The  cursor will be positioned on the first line of the buffer.  You can
                   get to the other files with the ":next" command.  To edit a file that starts  with
                   a dash, precede the filelist with "--".

       -           The  file to edit is read from stdin.  Commands are read from stderr, which should
                   be a tty.

       -t {tag}    The file to edit and the initial cursor position depends on a  "tag",  a  sort  of
                   goto  label.  {tag} is looked up in the tags file, the associated file becomes the
 Manual page vi(1) line 1 (press h for help or q to quit)
```

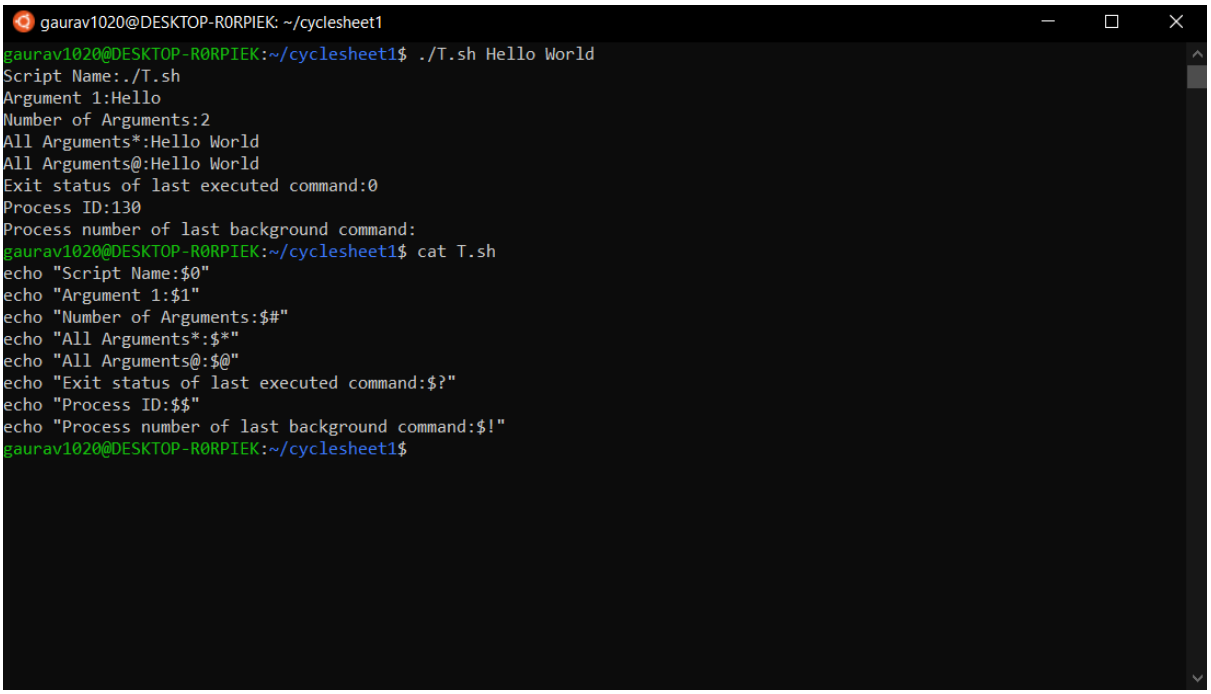
[Go To Top](#)

Answer 2)

Shell Programming

a) Handling Command Line Arguments

S. No.	Parameter	Description
1	$0	Returns filename of the script
2	$n	n is positive integer. Returns the nth argument given to the script when the script was invoked
3	$#	Returns the number of arguments given to the script when it was invoked
4	$*	Returns all arguments given to the script when it was invoked
5	$@	Returns all arguments given to the script when it was invoked
6	$?	Returns the exit status of last command executed
7	$$	Returns process number of the current shell i.e. process id under which the script is executing
8	$!	The process number of last background command



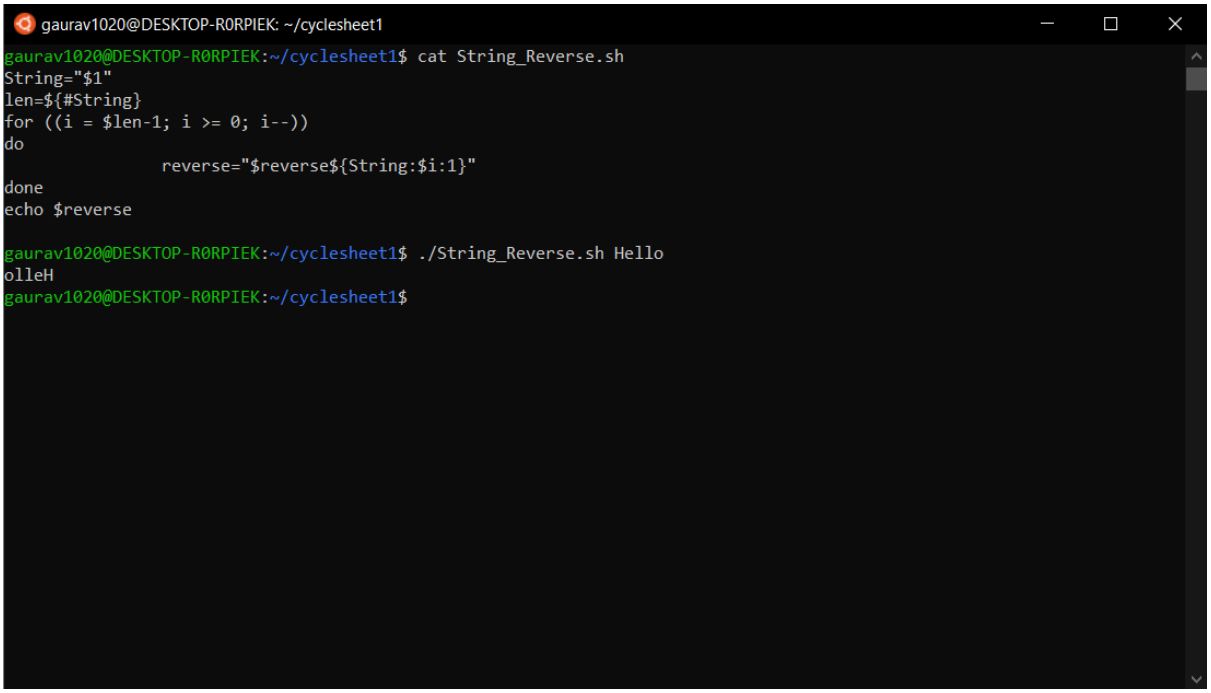
Go To Top

b) String Reversal

Code:

```
String="$1"
len=${#String}
for ((i = $len-1; i >= 0; i--))
do
        reverse="$reverse${String:$i:1}"
done
echo $reverse
```

Output:



Go To Top

c) If-Else, Nested If Else, Switch cases in shell

Code:

```
echo "IF ELSE STATEMENT"
if (( $1 == $2 ))
then
	echo "$1 is Equal to $2"
else
	echo "$1 is NOT Equal to $2"
fi
echo "NESTED IF ELSE STATEMENT"
if (( $1 >= 0 ))
then
		echo "$1 is Non Negative Integer"
		if (( $1 == 0 ))
		then
			echo "$1 is Equal to Zero"
		else
			echo "$1 is NOT Equal to Zero"
		fi
else
		echo "$1 is Negative Integer"
fi
echo "SWITCH CASE STATEMENT"
case $3 in
	1) echo "Case 1 invoked";;
	2) echo "Case 2 invoked";;
	3) echo "Case 3 invoked";;
esac
```

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet1
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ cat Ifelse_NestedIf_SwitchCase.sh
echo "IF ELSE STATEMENT"
if (( $1 == $2 ))
then
        echo "$1 is Equal to $2"
else
        echo "$1 is NOT Equal to $2"
fi
echo "NESTED IF ELSE STATEMENT"
if (( $1 >= 0 ))
then
                echo "$1 is Non Negative Integer"
                if (( $1 == 0 ))
                then
                        echo "$1 is Equal to Zero"
                else
                        echo "$1 is NOT Equal to Zero"
                fi
else
                echo "$1 is Negative Integer"
fi
echo "SWITCH CASE STATEMENT"
case $3 in
        1) echo "Case 1 invoked";;
        2) echo "Case 2 invoked";;
        3) echo "Case 3 invoked";;
esac

gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$ ./Ifelse_NestedIf_SwitchCase.sh -20 20 3
IF ELSE STATEMENT
-20 is NOT Equal to 20
NESTED IF ELSE STATEMENT
-20 is Negative Integer
SWITCH CASE STATEMENT
Case 3 invoked
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet1$
```

Go To Top

Answer 3)

A) Checking the Process Identifier

Code:

```
#include  <stdio.h>

#include  <sys/types.h>

void  main(void)

{

        pid_t pid;

        pid = getpid();

        printf("The Process ID is: ");

        printf("%d",pid);

        printf("\n");

}
```

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA1/Fork
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ls
PID_Check.c
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ cat PID_Check.c
#include  <stdio.h>
#include  <sys/types.h>

void  main(void)
{
        pid_t pid;
        pid = getpid();
        printf("The Process ID is: ");
        printf("%d",pid);
        printf("\n");
}
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ gcc PID_Check.c -o PID_Check
PID_Check.c: In function ‘main’:
PID_Check.c:7:8: warning: implicit declaration of function ‘getpid’ [-Wimplicit-function-declaration]
    7 |  pid = getpid();
      |        ^~~~~~
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ls
PID_Check  PID_Check.c
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ./PID_Check
The Process ID is: 6115
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$
```

Go To Top

B) Assigning new task to child

Code:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main() {
	pid_t pid;
	pid=fork();
	if(pid<0) {
		printf("Child process not created");
	}
	else if(pid==0) {
		printf("This is child process with Process ID : %d",getpid());
		printf("\n");
	}
	else {
		printf("This is Parent Process with Process ID : %d",getpid());
		printf("\n");
	}
}
```

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA1/Fork
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ls
New_Task_Child.c  PID_Check  PID_Check.c
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ cat New_Task_Child.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
        pid_t pid;
        pid=fork();
        if(pid<0) {
                printf("Child process not created");
        }
        else if(pid==0) {
                printf("This is child process with Process ID : %d",getpid());
                printf("\n");
        }
        else {
                printf("This is Parent Process with Process ID : %d",getpid());
                printf("\n");
        }
}

gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ gcc New_Task_Child.c -o New_Task_Child
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ./New_Task_Child
This is Parent Process with Process ID : 6161
This is child process with Process ID : 6162
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$
```

Go To Top

C) Providing the path name and program name to exec()

Code:

ex1.c

```
#include  <stdio.h>
#include  <unistd.h>
#include  <sys/types.h>

int main() {
	printf("This is in program ex1.c with Process ID : %d",getpid());
	printf("\n");
	char * arg[] = {"19","BCE","2119",NULL};
	execv("./ex2", arg);
	return 0;
}
```

ex2.c

```
#include  <stdio.h>
#include  <unistd.h>
#include  <sys/types.h>

int main() {
		printf("This is in program ex2.c with Process ID : %d",getpid());
		printf("\n");
		return 0;
}
```

Go To Top

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA1/Exec/Path_Program_Name_Exec
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ ls
ex1.c  ex2.c
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ cat ex1.c
#include  <stdio.h>
#include  <unistd.h>
#include  <sys/types.h>

int main() {
        printf("This is in program ex1.c with Process ID : %d",getpid());
        printf("\n");
        char * arg[] = {"19","BCE","2119",NULL};
        execv("./ex2", arg);
        return 0;
}
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ cat ex2.c
#include  <stdio.h>
#include  <unistd.h>
#include  <sys/types.h>

int main() {
                printf("This is in program ex2.c with Process ID : %d",getpid());
                printf("\n");
                return 0;
}
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ gcc ex1.c -o ex1
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ gcc ex2.c -o ex2
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$ ./ex1
This is in program ex1.c with Process ID : 6236
This is in program ex2.c with Process ID : 6236
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Exec/Path_Program_Name_Exec$
```

D) Synchronizing Parent and child process using wait()

Code:

```
#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main() {

    pid_t pid;

    pid=fork();

    if(pid<0) {

        printf("Child process not created");

    }

    else if(pid==0) {

        printf("This is child process with Process ID : %d",getpid());

        printf("\nAdding Two Numbers: \n");
```

Go To Top

```
        int a, b;
        scanf("%d %d",&a,&b);
        printf("Sum is: %d",a+b);
    }
    else {
        wait();
        printf("\nThis is Parent Process with Process ID : %d",getpid());
        printf("\n");
    }
}
```

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA1/Fork
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ls
New_Task_Child  New_Task_Child.c  PID_Check  PID_Check.c  Parent_Child_Synchronize.c
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ cat Parent_Child_Synchronize.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main() {
        pid_t pid;
        pid=fork();
        if(pid<0) {
                printf("Child process not created");
        }
        else if(pid==0) {
                printf("This is child process with Process ID : %d",getpid());
                printf("\nAdding Two Numbers: \n");
                int a, b;
                scanf("%d %d",&a,&b);
                printf("Sum is: %d",a+b);
        }

        else {
                wait();
                printf("\nThis is Parent Process with Process ID : %d",getpid());
                printf("\n");
        }
}

gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ gcc Parent_Child_Synchronize.c -o Parent_Child_Synchronize
Parent_Child_Synchronize.c: In function ‘main’:
Parent_Child_Synchronize.c:19:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-functio
n-declaration]
   19 |   wait();
      |   ^~~~
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ./Parent_Child_Synchronize
This is child process with Process ID : 6322
Adding Two Numbers:
19 21
Sum is: 40
This is Parent Process with Process ID : 6321
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$
```

Go To Top

Answer 4)

Code:

```
#include <stdio.h>
 #include <sys/types.h>
 #include <unistd.h>
 int main() {
        int k = 0;
        pid_t pid;
        do {
              printf("Enter a valid number to run Collatz Conjecture on:.\n");
              scanf("%d", & k);
        } while (k < 0);
        pid = fork();
        if (pid == 0) {
              printf("Child Process is Running having Process ID: %d\n",getpid());
              printf("%d\n", k);
              while (k != 1) {
                    if (k % 2 == 0) {
                          k = k / 2;
                    } else if (k % 2 == 1) {
                          k = 3 * (k) + 1;
                    }
                    printf("%d\n", k);
              }
              printf("Child process is done.\n");
        }
        else {
               printf("Parents is waiting for the completion of child process.\n");
               wait();
               printf("Parent process is done having Process ID:%d\n",getpid());
        }
    return 0;}
```

Go To Top

Output:

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA1/Fork
int main() {
        int k = 0;
        pid_t pid;
        do {
                printf("Enter a valid number to run Collatz Conjecture on:.\n");
                scanf("%d", & k);
        } while (k < 0);
        pid = fork();
        if (pid == 0) {
                printf("Child Process is Running having Process ID: %d\n",getpid());
                printf("%d\n", k);
                while (k != 1) {
                        if (k % 2 == 0) {
                                k = k / 2;
                        } else if (k % 2 == 1) {
                                k = 3 * (k) + 1;
                        }
                        printf("%d\n", k);
                }
                printf("Child process is done.\n");
        }
        else {
                 printf("Parents is waiting for the completion of child process.\n");
                 wait();
                 printf("Parent process is done having Process ID:%d\n",getpid());
        }
    return 0;
}
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ gcc Collatz_Conjecture.c -o Collatz_Conjecture
Collatz_Conjecture.c: In function ‘main’:
Collatz_Conjecture.c:30:19: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   30 |                 wait();
      |                 ^~~~
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$ ./Collatz_Conjecture
Enter a valid number to run Collatz Conjecture on:.
5
Parents is waiting for the completion of child process.
Child Process is Running having Process ID: 6354
5
16
8
4
2
1
Child process is done.
Parent process is done having Process ID:6353
gaurav1020@DESKTOP-R0RPIEK:~/DA1/Fork$
```

Go To Top