Operating Systems – 1: CS3510

Programming Assignment 1 - Report

➢ Design of Program:

1) This program gives the collatz sequence of a number (taken as input from user) using parent and child processes.

2) The positivity check for input is done using if – else statement and error message is displayed if user inputs non positive integers and process is terminated immediately.

3) If the input given is positive then a variable pid is defined and the fork() system call is called which creates a child process, The necessary if else statements are written.

4) If fork fails program gets terminated giving a stderr output file which shows . If (pid == 0) case is the execution of child process and else case is execution of parent.

5) Assume input is n, in child process checks it whether n is even or odd using if – else statement, if n is even, it is halved else n is given by (3 x n + 1) and this if else statement runs in a loop until value of n becomes 1.

6) A print statement is written inside the loop which prints the collatz sequence of n and n is printed outside of loop.

7) The parent process (else case) has a wait() call , the value of n in parent process is printed with a message. The parent ids and child ids are printed at each process for the analyzing the process.

8) In parent process (else case) the print statements are written confirming the termination of child and parent.

➢ Prerequisites for Analysis:

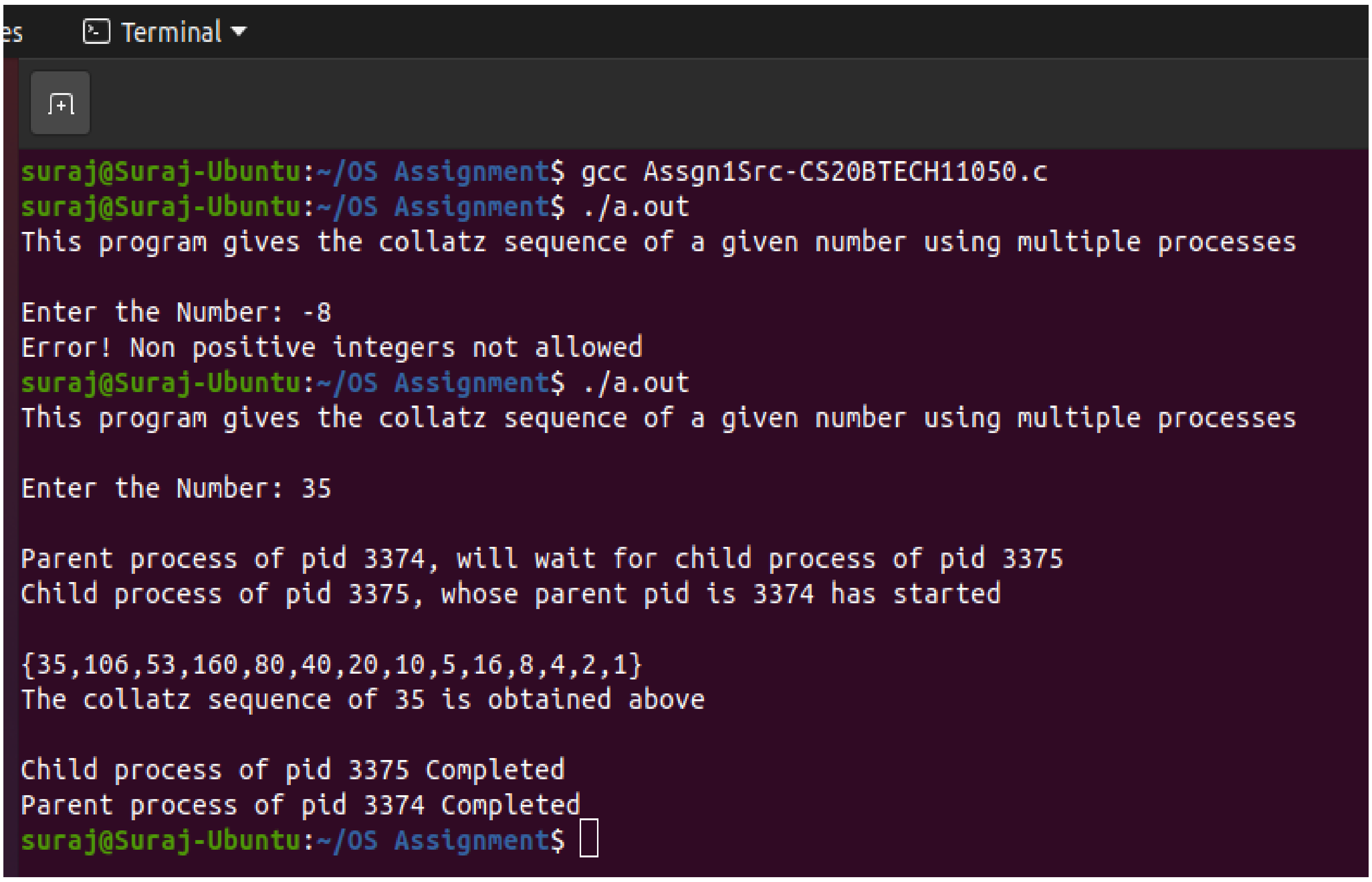
- **fork() system call**: pid_t fork(void); from #include <unistd.h>
  fork() creates a new process by duplicating the calling process (parent process). The new process is referred to as the *child* process. The child has a new pid and the parent process and child process run in different memory spaces

- **wait() system call:** pid_t wait(int **wstatus*); from #include <sys/wait.h>
  wait() is used to wait for state changes in child of a parent process. If a wait is not performed then the child remains as a zombie process.

- **getpid():** returns the process id of calling process.

- **getppid():** returns the process id of the parent of the calling process.

➢ Output of the program:



```
suraj@Suraj-Ubuntu:~/OS Assignment$ gcc Assgn1Src-CS20BTECH11050.c
suraj@Suraj-Ubuntu:~/OS Assignment$ ./a.out
This program gives the collatz sequence of a given number using multiple processes

Enter the Number: -8
Error! Non positive integers not allowed
suraj@Suraj-Ubuntu:~/OS Assignment$ ./a.out
This program gives the collatz sequence of a given number using multiple processes

Enter the Number: 35

Parent process of pid 3374, will wait for child process of pid 3375
Child process of pid 3375, whose parent pid is 3374 has started

{35,106,53,160,80,40,20,10,5,16,8,4,2,1}
The collatz sequence of 35 is obtained above

Child process of pid 3375 Completed
Parent process of pid 3374 Completed
suraj@Suraj-Ubuntu:~/OS Assignment$ 
```

➢ Analysis of Program:

1) Once the execution process starts after giving commands as per Assgn1Readme-CS20BTECH11050.txt file user has to give input to the program.

2) If the input given is negative the program gives an error message as above and terminates because collatz conjecture is not valid for negative integers or zero.

3) If the input is positive then the fork() system call is called from library unistd.h fork() duplicates the parent process i.e creates a child process with new pid as described above, a successful fork returns pid = 0 to the child process and pid = pid of child process to the parent.

4) The parent process starts execution and prints a the statement above then due to the wait() system call the parent waits for the child to terminate, the pid of parent and child has been printed for reference. Here getpid() function is used to print the process id of calling process i.e parent process and pid is used to print child process.

5) The child process starts its execution while parent waits the process id of child process and parent process are printed which shows that child has started making parent wait() due to wait function call. Here process id of parent process is printed using getppid() function.

6) After printing the input (n) the loop gets started which has the conditions of collatz sequence and loop runs until n = 1 and the child process terminates after printing the last number (n = 1), bracket and terminates. Here we can observe that the input value (n) has been reduced to 1 in child process.

7) The parent process prints message giving “The collatz sequence of n is obtained above” here we can observe that parent process prints the value of input which is same as original input given by user.

8) From the above print statement, we can conclude that both child and parent processes run in different memory spaces and altering a value in child does not effect the value in parent since child is just a duplicate of parent with different process ids and memory addresses.

9) Later the termination of child is confirmed by parent process and then it terminates giving a print statement.