Operating Systems 1

Tanmay Garg CS20BTECH11063

Programming Assignment 1

Program Design:

- We first take input from the user from command line which should be a positive integer
- Then we call the fork() system call
- If it is not able to fork() then it gives pid the value less than 0 and throws an error
- If it is able to fork() then it makes pid as 0 and now the child process is working
    - When the child process is started it runs a while loop to begin with the Collatz Conjecture
    - If n is even, $n \leftarrow n/2$
    - If n is odd, then n becomes $n \leftarrow 3n + 1$
    - The process continues until $n = 1$
- In all other situations it calls the wait() system call to wait for the child process to get over
- The child process outputs the list of numbers generated by the Collatz Conjecture

Assgn1Src_CS20BTECH11063.c > main(int, char * [])

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
    //If number of arguments is not 2 then throws an error and terminates the program
    if(argc != 2)
    {
        printf("Incorrect number of inputs\n");
        printf("Input: final_exec n (where n is the number)\n");
        return EXIT_FAILURE;
    }
    //Our number n on which Colatz conjecture will be tested
    int n = atoi(argv[1]);
    //If n is less than zero then ask for input again
    while(n <= 0)
    {
        printf("Please enter number greater than 0: ");
        scanf("%d", &n);
    }

    //forking a child process
    pid_t pid = fork();
    if (pid < 0)
    {
        //If error occurs
        fprintf(stderr, "Fork Failed\n");
        return EXIT_FAILURE;
    }
    else if (pid == 0)
    {
        //Child Process
        printf("Child Process is working: ");
        printf("%d ", n);
        while (n != 1)
        {
            //If n is even it is divided by 2
            if (n % 2 == 0)
            {
                n = n / 2;
            }
            //else n is 3n+1
            else
            {
                n = 3 * n + 1;
            }
            printf("%d ", n);
        }
        printf("\nChild Process is completed\n");
    }
    else
    {
        //Parent Process
        //Parent waits for child process to be completed
        printf("Parent Process is waiting for Child Process\n");
        wait(NULL);
        printf("Parent Process is completed\n");
    }
    return EXIT_SUCCESS;
}
```

Program Analysis:

- When the program performs fork() it assigns a Process ID to pid
- If the Process ID is 0 then it proceeds to the Child Process
- The Loop performs Collatz Conjecture on the number till it reaches the value 1
- The Program has been tested for different values such as 65, 40, 101 and all of them converge into the series 4, 2, 1
- If the Process ID is greater than 0 then it waits for the child process to be completed
- If the Process ID is less than 0 then it shows an error as the process could not be created and hence returns EXIT_FAILURE
- If the entire program works and gives a proper output, then it simply returns EXIT_SUCCESS
- In the situation when the number of input arguments in command line is not 2 then it throws an error and exits the program such as shown in the demo below
- In the situation when the input number is less than 0 then it throws an error and asks user to input positive number