ACEDMIC TASK-3

OPERATING SYSTEM(CSE-316)

STUDENT NAME: Abhinav Singh

STUDENT ID: 11802510

EMAIL ADDRESS: abhinavsingh080301@gmail.com

CODE-1: C PROGRAM

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        printf("\n\nWelcome to Hailstone, by Abhinav\n");
        int n=0;
        int k=0;

        pid_t pid;

                do
                {
                        printf("Please enter any number greater than 0
                        \n"); scanf("%d", &k);
                }while (k <= 0);

                pid = fork();

                if (pid == 0)
                {
```

```
                printf("Child is working...\n");
                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 on child process...\n");
                wait();
                printf("Parent process is done.\n");
        }
    return 0;
}
```

DISCRIPTION OF PROBLEM IN TERMS OF OPERATING SYSTEM:

In this question we need to create a process that in turn creates the child process and that child process will output some sequence. For ex. if 16 is passed as a parameter on the command line, the child process will output 16, 8, 4, 2, 1. In the concept of operating system the system call used for creating the child process is Fork(). This system call creates a child process and returns a integer value. Depending on that integer value we get the idea whether child process is created or not. If the process returns the value -1 indicates that child process has not been created. If the value returned is 0 then it indicates that child process is created.

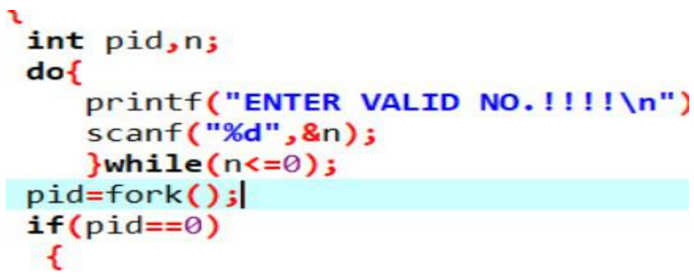
We need to prevent the parent process from completing before the child process because child process will become orphan process. Therefore to prevent this we need to use system call wait() so that the child process completes before the parent process.

ALGORITHM:

1- The program takes the input from the user.
2- In the next step the program creates the child process. And the value returned from the fork is compared with 0 to check if the child is created or not.
3- If the child process is created then the value is used inside a while loop and then sequence is printed.
4- While printing the sequence the parent process waits until the child process completes.

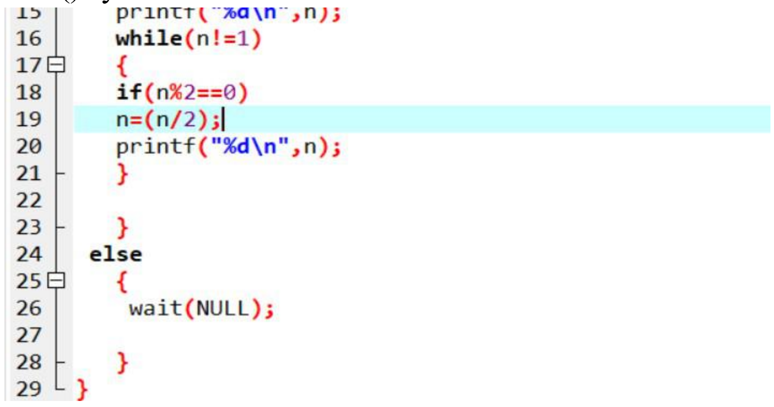
CONSTRAINTS IMPLEMENTED:

1- The first constraint is that the user should only input any positive integer. If the user provide any negative integer then the program checks and displays a message to enter a valid number.



```
{
 int pid,n;
 do{
     printf("ENTER VALID NO.!!!!\n")
     scanf("%d",&n);
     }while(n<=0);
 pid=fork();
 if(pid==0)
  {
```

2- The program should only terminate after the completion of child process. We use wait() system call.

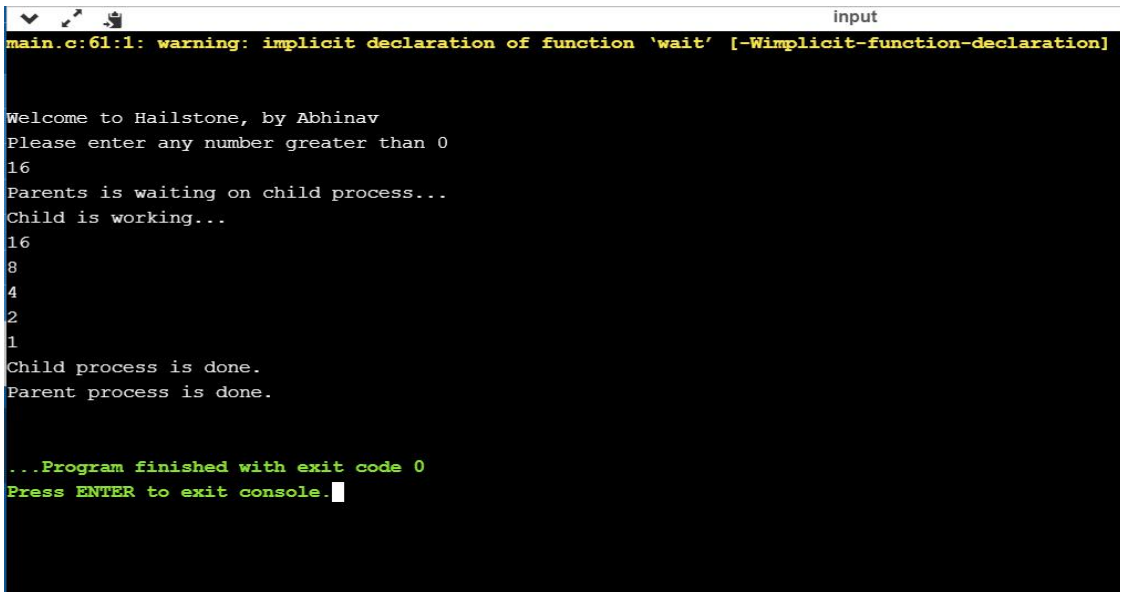


```
    printf("%d\n",n);
    while(n!=1)
    {
    if(n%2==0)
    n=(n/2);
    printf("%d\n",n);
    }

    }
  else
    {
     wait(NULL);

    }
}
```

TEST CASES APPLIED:

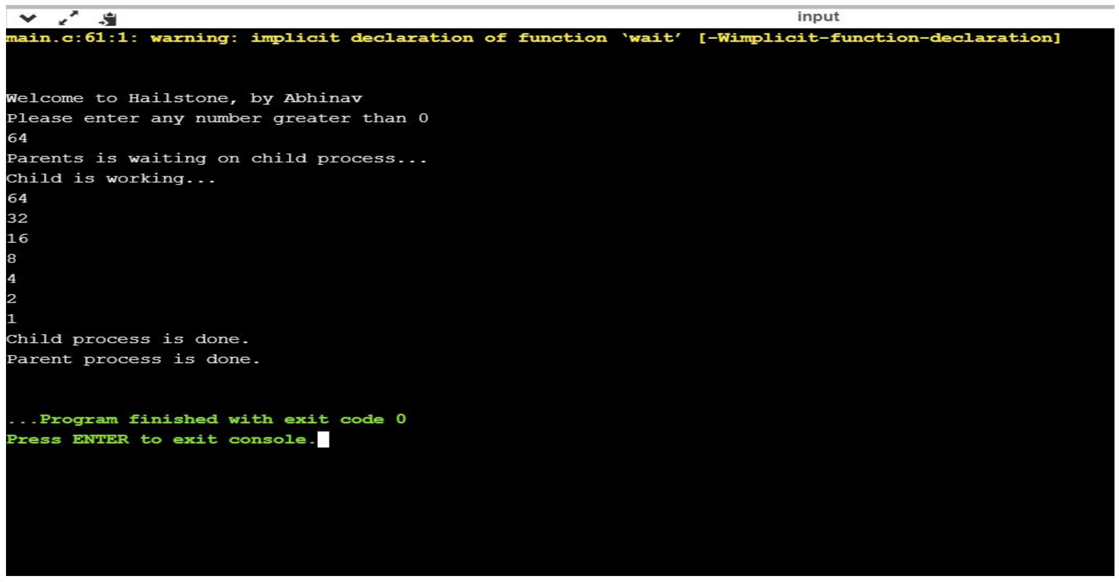


input

```
main.c:61:1: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]


Welcome to Hailstone, by Abhinav
Please enter any number greater than 0
16
Parents is waiting on child process...
Child is working...
16
8
4
2
1
Child process is done.
Parent process is done.


...Program finished with exit code 0
Press ENTER to exit console.
```



input

```
main.c:61:1: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]


Welcome to Hailstone, by Abhinav
Please enter any number greater than 0
64
Parents is waiting on child process...
Child is working...
64
32
16
8
4
2
1
Child process is done.
Parent process is done.


...Program finished with exit code 0
Press ENTER to exit console.
```

HAVE YOU MINIMUM 5 REVISIONS:

Yes I have made minimum 5 revisions.

CODE 2: C PROGRAM

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<errno.h>

int main(int argc, char *arg[])
{
int fd[2];
int filebytelen;
char buffer[100];
char childbuffer[100];

char *src=arg[1];
char *dest=arg[2];

if(pipe(fd)<0)
{
printf("error !!! %s\n",strerror(errno));
exit(1);
}

switch(fork())
{
case -1:
printf("error forking child process %s\n",strerror(errno));
exit(1);
case 0:
```

```
close(fd[1]);
ssize_t child_bytes= read(fd[0],childbuffer,sizeof(childbuffer));
close(fd[0]);

int target=open("dest",O_CREAT|O_WRONLY);
write(target,childbuffer,child_bytes);

default:
close(fd[0]);
int fileindes=open(src,O_RDONLY);
ssize_t num_bytes=read(fileindes,buffer,sizeof(buffer));
write(fd[1],buffer,num_bytes);
close(fd[1]);
}
return 0;
}
```

PROBLEM IN TERMS OF OPERATING SYSTEM:

In this question we have to make program named which implements the concept of file. The programs demands to be passed two parameters the name of the file having the content and name of the file in which the content needs to be copied. The concept behind using pipe is to perform the interprocess communication such that one process writes to the pipe and other process reads from that pipe. Pipe is an one way communication process.

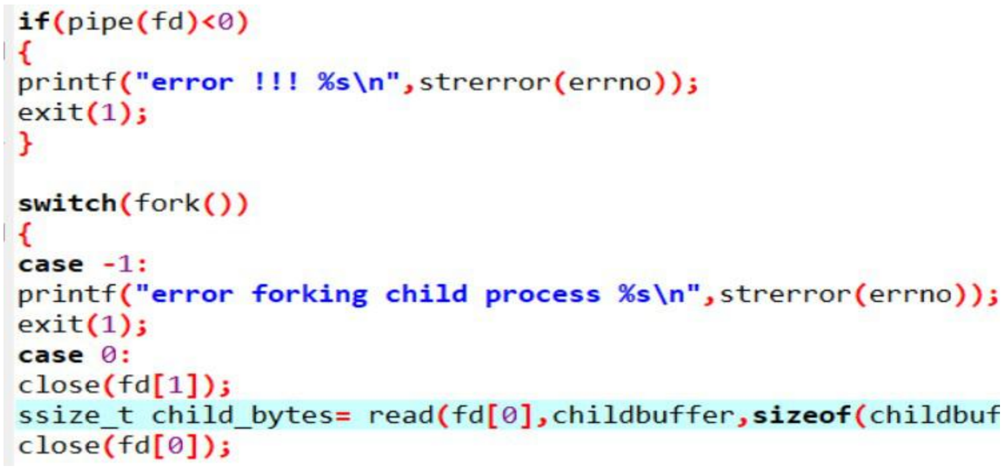
The program asks to perform communication in such a way that the parent process writes into the pipe and then child process will read from the pipe and copy it into the destination file using pipe.

ALGORITHEM:

1- The program declares an array of file descriptors.
2- Then the program creates child process and check for the return value. If the process is created then the content of the source file is copied into the pipe.
3- Then the copied data is pasted into the destination file. Then all the files are closed.

CONSTRAINTS IMPLEMENTED:

1- The program checks for all the errors which might be occur during the creation of the file.
2- It also checks for whether a child process is created or not.



```
if(pipe(fd)<0)
{
printf("error !!! %s\n",strerror(errno));
exit(1);
}

switch(fork())
{
case -1:
printf("error forking child process %s\n",strerror(errno));
exit(1);
case 0:
close(fd[1]);
ssize_t child_bytes= read(fd[0],childbuffer,sizeof(childbuf
close(fd[0]);
```

Output:-

Filecopy: filecopy.exe [target] [destination].

: Success

HAVE YOU MADE MINIMUM 5 REVISIONS :

YES I have made minimum 5 revisions.