ASSIGNMENT 1.1 -PROCESS CREATION AND TERMINATION SYSTEM CALLS

The system calls used in this assignment are-

waitpid()- It is a sys call in which the current process is stopped from working till the system call waitpid() ensures that the child process is finished or stopped .

Header file- #include<sys/types.h>

#include<sys/wait.h>

The arguments passed in the waitpid() system call-

waitpid(child,&status,0)- Here I have used the status as NULL so that the waitpid() waits until the child process has nothing left in it.

The values of pid have different meanings-

<-1 - the wait is done for the child process whose process ID is same as the absolute value of the pid .

-1 - the wait is done for any child process.

0 – the wait is done for the child process whose process ID is same as that of calling process

>0 – the wait is done for the child process whose process ID is equal to that of pid.

write()-This system call writes data in bytes format as specified by the programmer. The bytes are provided in a buffer that was declared by the user in the program beginning.

Header file-#include<unistd.h>

write(int fildes, const void *buf,size_t bytes)- here

fildes - I have used the standard file descriptor 1 to write .

*buf - A pointer set to the buffer of at least n bytes bytes, that we will be writing to the file.

size_t bytes - The number of bytes that are to be written is specified here.

exit()- The system call is used to terminate the calling process immediately .

Header file- #include<stdlib.h>

#include<stdio.h>

exit(int status)- Here I have used status as 1 to get to exit with an error . And have 0 to make the exit without any error.

fork()- The system call is used to created new child processes. Once the child process is created the parent and child will now execute according to the fork() system call . A child process has the same program counter as of the parent process.

Header file-#include<sys/types.h>

#include<unistd.h>

fork (0)-Here I have used 0 to create a new child process

The different values of fork() syscall:-

1. <0 – Child process creation unsuccessful
2. 0-Returns the newly created child process
3. >0-Returns the parent call

The working of the program-

The make command is used to run the makefile . Now after the makefile is run successfully. The following commands are done-

```
q1: q1.o
	gcc q1.o -o q1
q1.o: q1.s
	gcc -c q1.s -o q1.o
q1.s: q1.i
	gcc -S q1.i -o q1.s
q1.i: q1.c
	gcc -E q1.c -o q1.i


clean:
	rm q1.i q1.s q1.o q1
```

This way we got our output file a.out.

The output file shows the Roll No , Section , Marks in all the 4 assignments and the average marks of the student. The process starts when a child process is created first with the help of fork() system call .

The child process gives the output of all the students who belong to section A . And in case the process interrupted in between the waitpid() system call is used to make the parent process wait for the completion of the child process.

Once the child process is completed the parent process runs and prints the details of the students of section B.

Error Handling-

1. A check is done to print an error message in case the .csv file is not found in the directory same as that of the program file. It will not be able to locate it and hence the file is not found error message is flashed on the screen.

2. A check is provided so that the child process gets completely executed before the parent process starts . If the child is not completed before the interruption of the parent process the system will give exit with status 1 message . Indicating the error that the child process was not yet finished when the parent begin .