Gebze Technical University

Computer Engineering Faculty

Homework 2 Report

Signals and Processes

Student: Muhammed Okumuş

No: 151044017

Contents

1 Program Design

Under this section, I will share function prototypes, their parts in the work flow of the whole program and their algorithms.

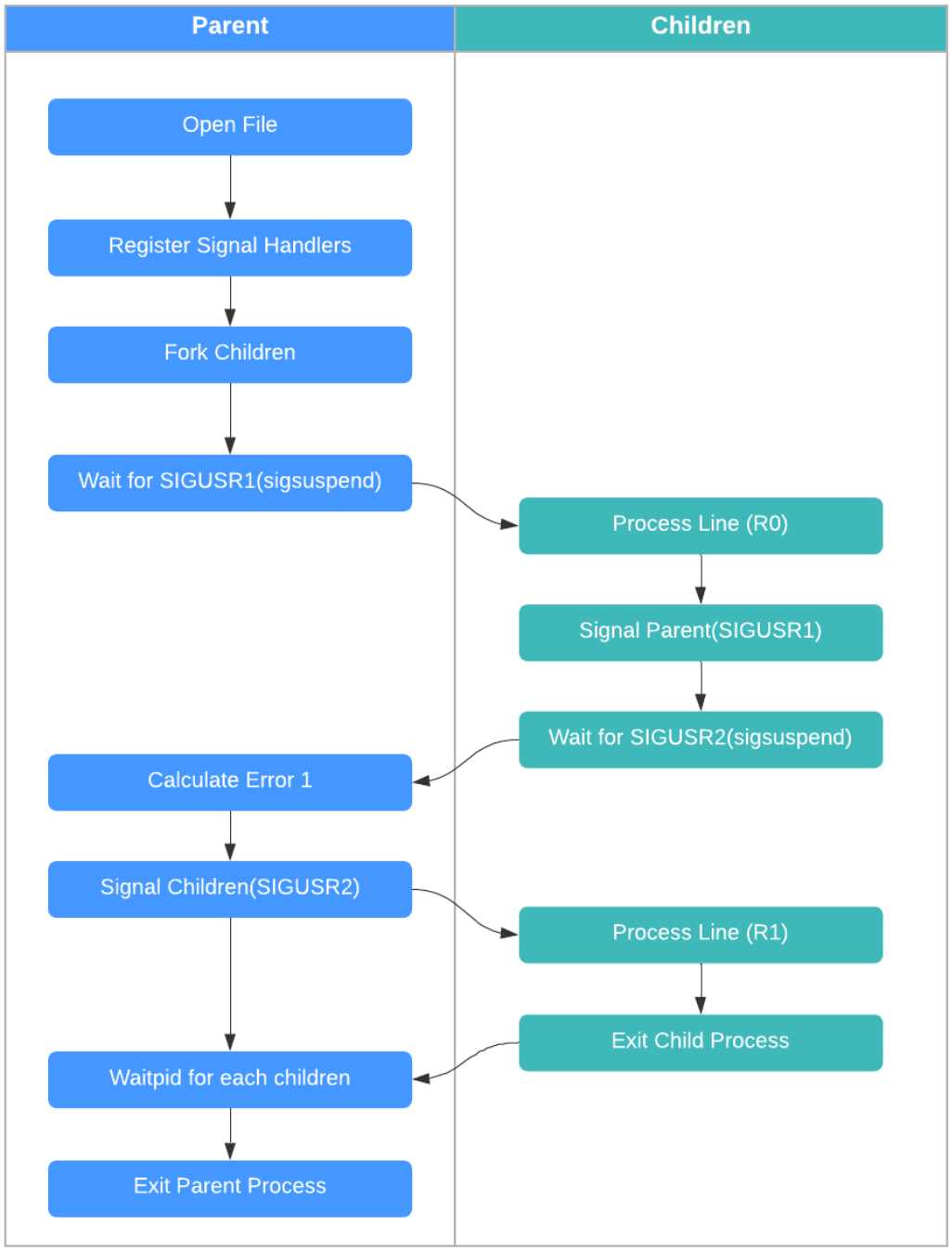

Figure 1: *Main flow of the parent and children process*

The program can be explained in two different sections: before fork and after fork. Before the fork system call main process initlizes the some shared variables with mmap, assigns signals to the signal handler, validates input and opens the input file.

After the fork system call parent immediately start waiting for SIGUSR1 using sigsuspend since children hasn't produced any input yet. At this point children processes call process_line function with the file descriptor acquired from the parent process and their child number(0 to 7) which decides which line to be processed by the function. process_line function uses **pread/pwrite** to append approximation calculated by the calculate function.

There are 2 buffers used for appending to a line in the file.Lagrange result buffer which hold the result of the approximation. And after buffer which hold the file content after appended offset. After the result is available for the line, file is rewritten after the changed offset by two pwrite calls. Also an **advisory write lock** is present, wrapping all the read and write operations.

After each process_line function call, shared variable(mmap'ed) i_child_done variable is incremented and if all the childs done processing a signal is send to the parent via the kill system call.Also each child starts waiting on SIGUSR2 with sigsuspend after the process_line function call.

At this point parent process was waiting on SIGUSR1 and just got it. It immediately calcutes error using the avarage_error function and prints error at degree 5 and start waiting with waitpid() for each children. After that SIGUSR2 is raised and send to each children. Children calculates the degree 6 approximation, prints coefficent(multiplied by the target value) and calls _exit.

After the all children conludes, parent calculates and prints out the error rate for the second time, but this time for degree 6 polynomial. Finally the file is closed and resources are freed.

2 Signal Handling

Signal handling is crucial in the program since all the syncronization is done via signals. There signals are registed to the signal handler: SIGINT, SIGUSR1 and SIGUSR2. Interrupt signal is used for handling the Ctrl+C exit via the user while SIGUSR1-2 are used for signalling between the parent and children processes. SIGUSR1 signal is raised only when a condition met which is when all childeren are done processing there respective lines in file. Only one childeren is lucky enough the raise this signal.

```
...
// Variable to be shared among children...
i_child_done = mmap(...);
*i_child_done = 0;
...
(*i_child_done)++;
if (*i_child_done == 8)
  kill(getppid(), SIGUSR1);
...

void sig_handler(int sig_no)
{
  if (sig_no == SIGUSR1)
    childs_done = 1;
  else if (sig_no == SIGUSR2)
    parent_done = 1;
  else
    exit_requested = sig_no;
}
```

SIGUSR2 in other hand is raised 8 times using the kill system call in the parent process after the first error calculation.

```
//SIGNAL CHILDEREN(SIGUSR2)
for (int i = 0; i < 8; i++)
{
  debug_printf("Parent signalling to C%d\n", i);
  kill(pid[i], SIGUSR2);
}
```

3 Debug Print

Here is example output of the function with the debug define is set to 1.

```
I'm C0 [pid: 12516, ppid: 12515]
I'm C1 [pid: 12517, ppid: 12515]
I'm C3 [pid: 12519, ppid: 12515]
I'm C4 [pid: 12520, ppid: 12515]
I'm C5 [pid: 12521, ppid: 12515]
I'm the father [pid: 12515, ppid: 2941]
Parent waiting: 0
I'm C6 [pid: 12522, ppid: 12515]
I'm C2 [pid: 12518, ppid: 12515]
I'm C7 [pid: 12523, ppid: 12515]
C0 is waiting
C1 is waiting
C3 is waiting
C4 is waiting
C5 is waiting
C6 is waiting
C2 is waiting
C7 is waiting
Parent's done waiting: 1
round 0 -- error1: 5.9
round 0 -- error2: 0.7
round 0 -- error3: 1.1
round 0 -- error4: 2.2
round 0 -- error5: 82.2
round 0 -- error6: 91.8
round 0 -- error7: 2.1
round 0 -- error8: 0.6
Error of polynomial of degree 5: 23.3
Parent signalling to C0
Parent signalling to C1
Parent signalling to C2
C0 is done waiting
Parent signalling to C3
C1 is done waiting
Parent signalling to C4
C2 is done waiting
Parent signalling to C5
C3 is done waiting
Parent signalling to C6
```

```
C4 is done waiting
C5 is done waiting
Parent signalling to C7
C6 is done waiting
C7 is done waiting
Polynomial 0: 0.7,0.0,0.5,-0.0,0.1,-0.2,-0.0,
waitpid0
Polynomial 1: 0.0,1.2,-0.0,-0.3,-0.0,0.0,0.1,
waitpid1
Polynomial 2: 0.1,1.2,-0.8,-0.6,0.3,-0.0,0.8,
waitpid2
Polynomial 3: 0.0,0.6,-0.2,1.0,-0.0,0.2,-0.6,
waitpid3
Polynomial 4: -0.1,2.0,6.0,-5.0,0.0,-5.8,3.9,
waitpid4
Polynomial 5: 5.8,-6.0,0.0,31.5,48.0,-61.4,-16.8,
waitpid5
Polynomial 7: -0.0,0.3,1.0,0.0,0.2,-0.0,-0.5,
Polynomial 6: 2.5,-0.8,2.5,0.1,-0.0,0.1,-3.3,
waitpid6
waitpid7
round 1 -- error1: 5.2
round 1 -- error2: 0.5
round 1 -- error3: 2.1
round 1 -- error4: 5.9
round 1 -- error5: 49.8
round 1 -- error6: 3.8
round 1 -- error7: 9.5
round 1 -- error8: 0.9
Error of polynomial of degree 6: 9.7
Parent: all childeren exited
```