CIS 415 Operating Systems

Assignment <2> Report Collection

Submitted to:

Prof. Allen Malony

Author:

<*Luying Cai*>

Report

Introduction

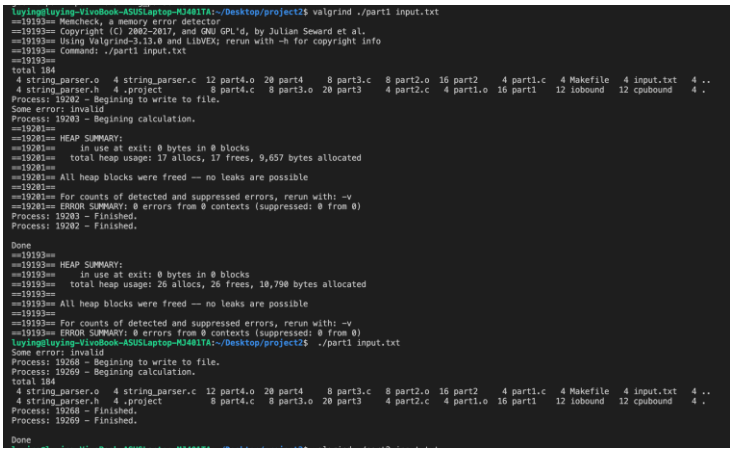
In this project, we will implement the MCP Ghost in the Shell (MCP for short) whose primary job is to launch a pool of sub-processes that will execute commands given in an input file. The MCP will read a list of commands (with arguments) to run from a file, it will then start up and run the process that will execute the command, and then schedule the processes to run concurrently in a time-sliced manner. In addition, the MCP will monitor the processes, keeping track of how the processes are using system resources. There are a total of 4 parts to the project, each building on the other.

Background

In this project, we need to use some system calls. For part1, we need to figure out how to read the commands and their arguments from the "workload" file and get them in the proper form to call exec. For part2, we need to implement a way for the MCP to stop all forked (MCC) child processes right before they call exec(), and implement the needed mechanism for the MCP to signal a running process to stop. For part3, we need to using signal processing to implement a more intelligent way of process scheduling. For part4, we need to see how the workload execution is proceeding by looking in the /proc directory for information on the workload processes.

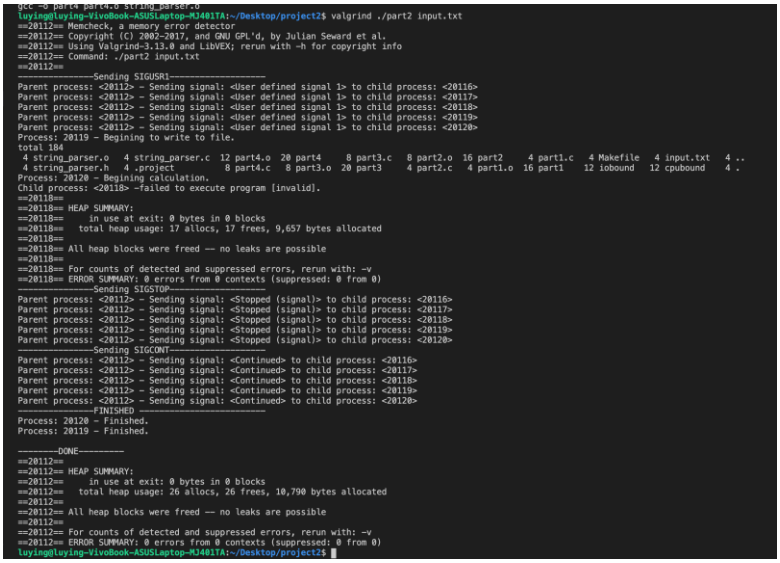
Implementation

For part 1:

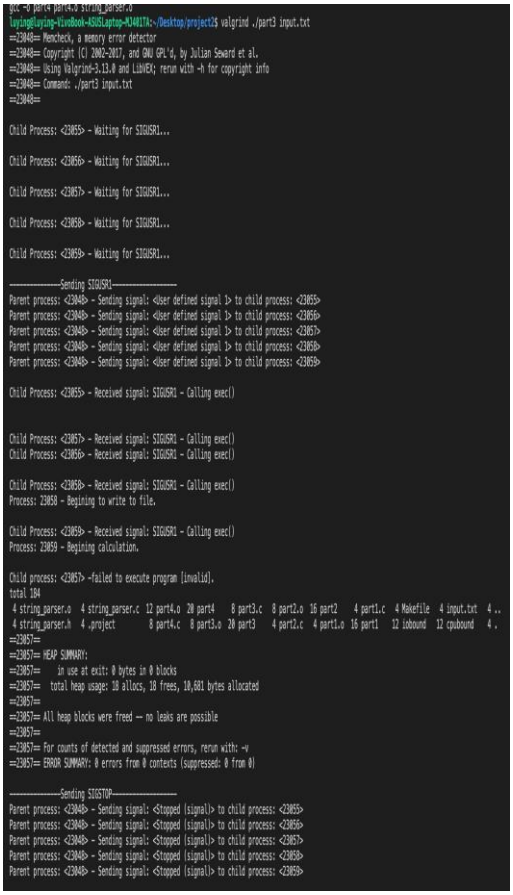


No leak problem and memory error shows on my laptop by using provided input file.

For part 2:



For part 3:



For part 4:

```
Alarm setting.....

<30973> count.....
PID = 30973
process ID = 30973
process state = (
parent pid = 32765
process group ID of the process = 2147479968
The controlling terminal of the process = 3
The number of minor faults the process has made which have not required loading a memory page from disk = 139881721734215
The number of major faults that the process's waited-for children have made = 1
 The nice value = 12884901888
Number of threads in this process = 140726723869904
Process: 30973 - Finished.

Alarm setting.....

<30974> count.....
PID = 30974
process ID = 30974
process state = (
parent pid = 32765
process group ID of the process = 2147479968
The controlling terminal of the process = 3
The number of minor faults the process has made which have not required loading a memory page from disk = 139881721734215
The number of major faults that the process's waited-for children have made = 1
 The nice value = 12884901888
Number of threads in this process = 140726723869904
Process: 30974 - Finished.

 Normal exit : <30973>

 Normal exit : <30974>

 --------CHILD PROCESS ALL DEAD! ! ! ! ------
```

For some reason, my only can print info which process didn't finish.

Performance Results and Discussion

for part3, if I didn't use [WUNTRACED], my process will exit by using WIFEXITED is true.

```
--------------------CHECK IS ANY CHILD STILL ALIVE-----------------------

 Normal exit : <31539>

 Normal exit : <31540>

 Normal exit : <31541>

 Normal exit : <31542>

 Normal exit : <31543>

 --------CHILD PROCESS ALL DEAD! ! ! ! ------
```

I didn't know why and for understand each flag, I use this website to study.

https://blog.csdn.net/Roland_Sun/article/details/32084825

for part 4, my child info can only print process who alive lively. Therefore, for understand all of process info, I use a loop to print each process before calling round robin helper function.

```
sleep(3);
for(int i=0; i<commandSize; i++){
    fakeTop(arrayPid[i]);
    sleep(2);
}
//to check which child process is still live. go to for loop to
printf("\n----------------------CHECK IS ANY CHILD STILL ALIVE--
roundRobin(set, sig, arrayPid, commandSize);
```

Conclusion

I think this project is difficult for me to determine which child process finished and after one child process finished, how to handle it never execute again.