Morgan Elder

CMPSCI 4760

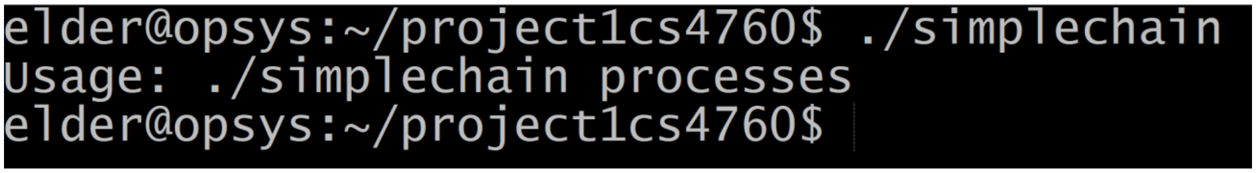
3 February 2022

Project 1

Observations

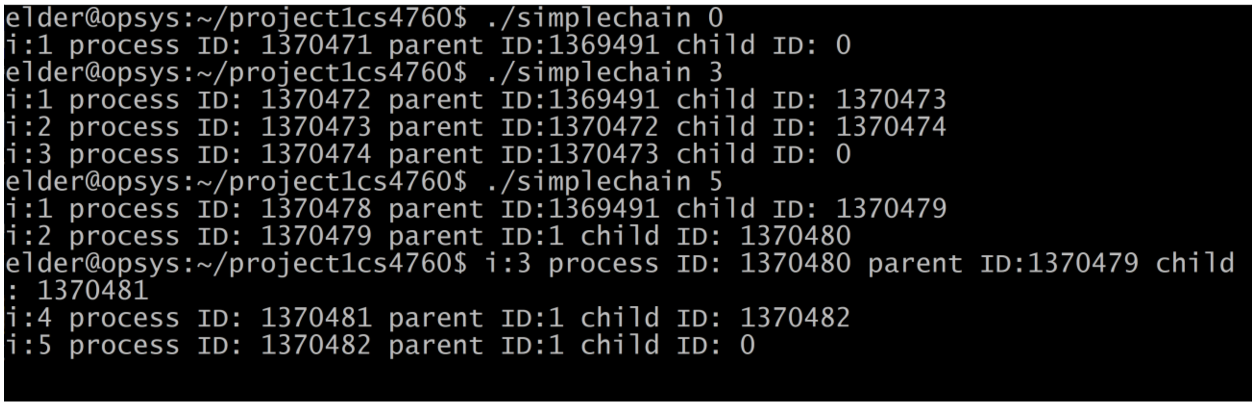
1. Run your program and observe the results for different number of processes.

The following screenshot shows when no arguments are passed to simplechain. The usage message is displayed.



```
elder@opsys:~/project1cs4760$ ./simplechain
Usage:  ./simplechain processes
elder@opsys:~/project1cs4760$
```

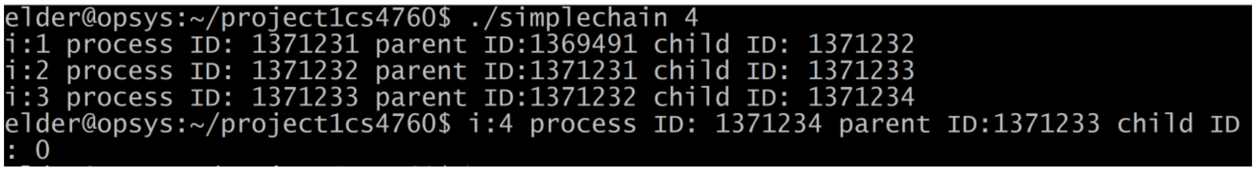
The following screenshot shows the results of passing the arguments of 0, 3, and 5 as the number processes to simplechain. If 0 processes are forked, then there is only a single process displayed. If more than more 1 process is forked (3 and 5), then the program prints the results in the order that the processes are forked. When the number of forked processes is about 5 or more, one of the parent processes tends to terminate before the child makes a call for the parent process ID. At that time, the child (orphaned) process is assigned the parent process of 1.



```
elder@opsys:~/project1cs4760$ ./simplechain 0
i:1 process ID: 1370471 parent ID:1369491 child ID: 0
elder@opsys:~/project1cs4760$ ./simplechain 3
i:1 process ID: 1370472 parent ID:1369491 child ID: 1370473
i:2 process ID: 1370473 parent ID:1370472 child ID: 1370474
i:3 process ID: 1370474 parent ID:1370473 child ID: 0
elder@opsys:~/project1cs4760$ ./simplechain 5
i:1 process ID: 1370478 parent ID:1369491 child ID: 1370479
i:2 process ID: 1370479 parent ID:1 child ID: 1370480
elder@opsys:~/project1cs4760$ i:3 process ID: 1370480 parent ID:1370479 child
: 1370481
i:4 process ID: 1370481 parent ID:1 child ID: 1370482
i:5 process ID: 1370482 parent ID:1 child ID: 0
```

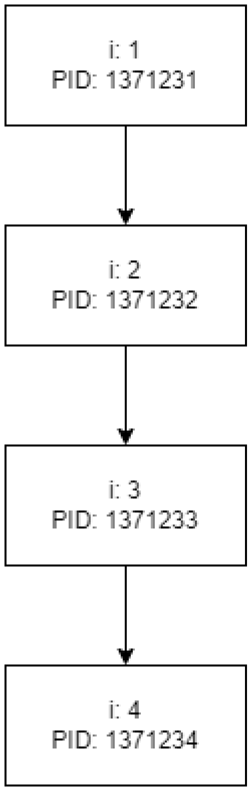
2. Redraw Figure 3.2 by filling in actual PIDs of the processes in the figure for a run with command-line argument value of 4

This screen shot is the result of chaining 4 processes.



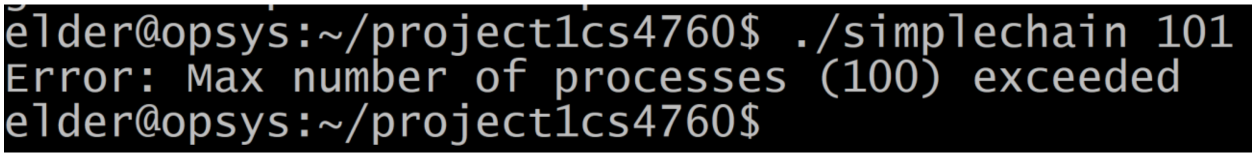
```
elder@opsys:~/project1cs4760$ ./simplechain 4
i:1 process ID: 1371231 parent ID:1369491 child ID: 1371232
i:2 process ID: 1371232 parent ID:1371231 child ID: 1371233
i:3 process ID: 1371233 parent ID:1371232 child ID: 1371234
elder@opsys:~/project1cs4760$ i:4 process ID: 1371234 parent ID:1371233 child ID
: 0
```

This is a redraw of figure 3-2 from the text.



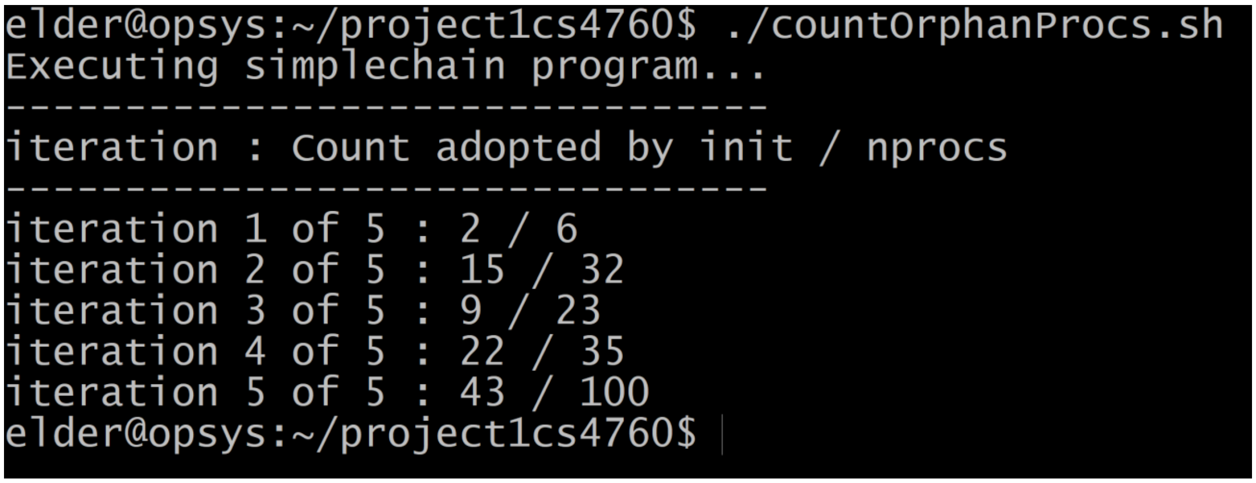
3. Experiment with different values for the number of processes (nprocs) up to a maximum of 100 processes. Observe the fraction that are adopted by init.

Here the max number of processes cannot be exceeded.



```
elder@opsys:~/project1cs4760$ ./simplechain 101
Error: Max number of processes (100) exceeded
elder@opsys:~/project1cs4760$
```

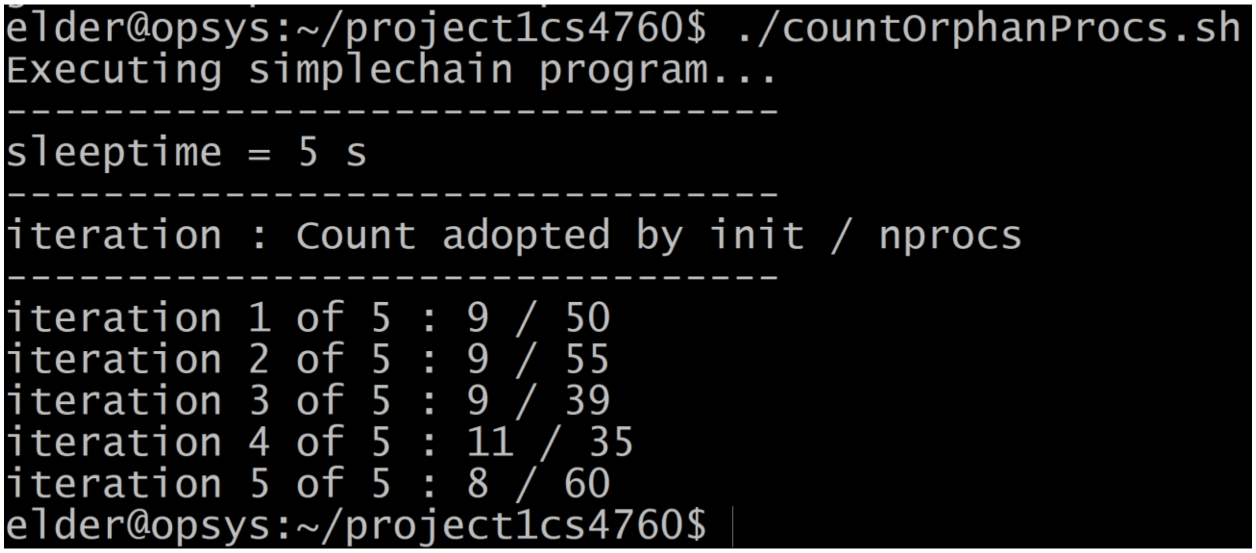
The script countOrphanProcs.sh executes the simple chain program with randomly generated number of processes. Then, the counts of processes adopted by init are displayed



```
elder@opsys:~/project1cs4760$ ./countOrphanProcs.sh
Executing simplechain program...
--------------------------------
iteration : Count adopted by init / nprocs
--------------------------------
iteration 1 of 5 : 2 / 6
iteration 2 of 5 : 15 / 32
iteration 3 of 5 : 9 / 23
iteration 4 of 5 : 22 / 35
iteration 5 of 5 : 43 / 100
elder@opsys:~/project1cs4760$
```

4. Place sleep(sleeptime); directly before the final fprintf statement in the code. Make sure to use the sleeptime parameter from command line or set it to a default value of 3. Make the previous observation again.

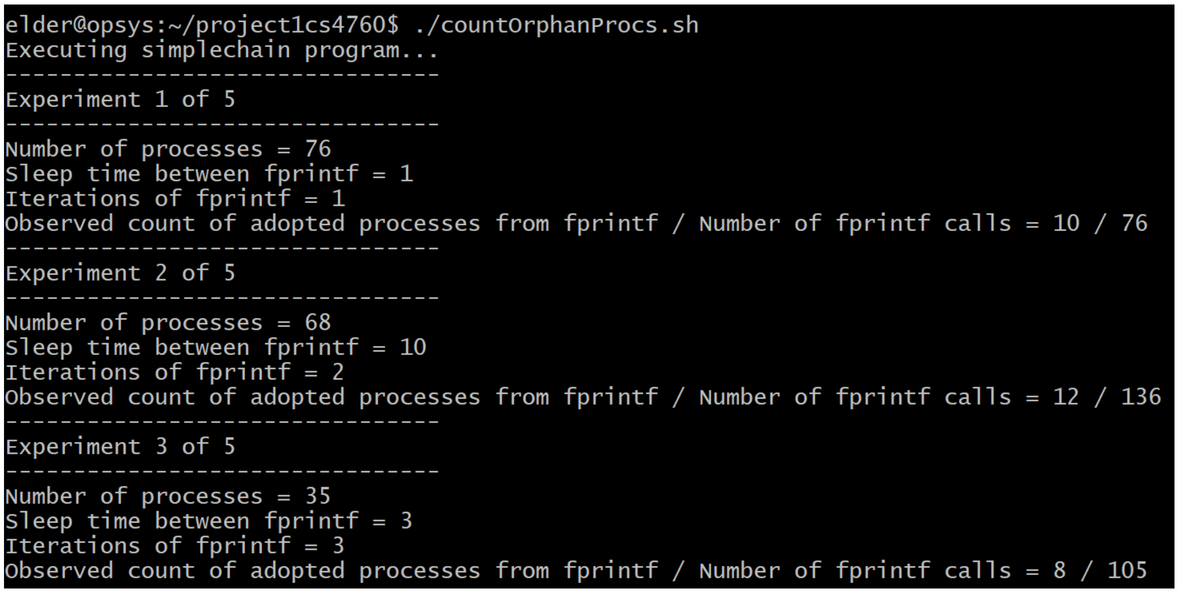
Here is the updated experiment that sleeps for 5 seconds before each fprintf. The script utilizes the command-line arguments for nprocs and sleeptime (options p and s). Again the number of processes for each iteration is randomly generated by the script.



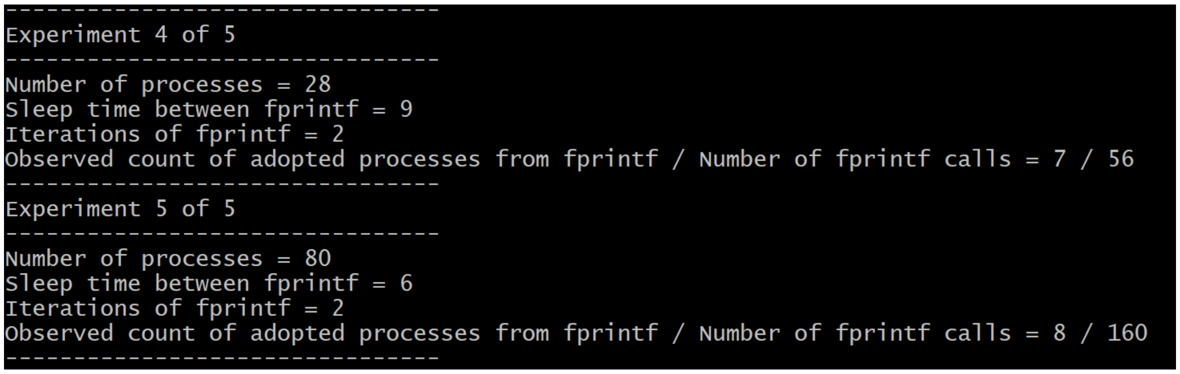
```
elder@opsys:~/project1cs4760$ ./countOrphanProcs.sh
Executing simplechain program...
--------------------------------
sleeptime = 5 s
--------------------------------
iteration : Count adopted by init / nprocs
--------------------------------
iteration 1 of 5 : 9 / 50
iteration 2 of 5 : 9 / 55
iteration 3 of 5 : 9 / 39
iteration 4 of 5 : 11 / 35
iteration 5 of 5 : 8 / 60
elder@opsys:~/project1cs4760$
```

5. Put a loop around the final fprintf in your code. Have the loop execute niters times. Put sleep(sleeptime); inside this loop just before the fprintf statement. Pass niters and sleeptime using command line options. Run the program for several values of nprocs, niters, and sleeptime. Observe the results.

The next two screenshots show the results of a script that executed the simplechain program 5 times. Each execution (or experiment) used randomly generated values for nprocs, sleeptime, and niter.



```
elder@opsys:~/project1cs4760$ ./countOrphanProcs.sh
Executing simplechain program...
--------------------------------
Experiment 1 of 5
--------------------------------
Number of processes = 76
Sleep time between fprintf = 1
Iterations of fprintf = 1
Observed count of adopted processes from fprintf / Number of fprintf calls = 10 / 76
--------------------------------
Experiment 2 of 5
--------------------------------
Number of processes = 68
Sleep time between fprintf = 10
Iterations of fprintf = 2
Observed count of adopted processes from fprintf / Number of fprintf calls = 12 / 136
--------------------------------
Experiment 3 of 5
--------------------------------
Number of processes = 35
Sleep time between fprintf = 3
Iterations of fprintf = 3
Observed count of adopted processes from fprintf / Number of fprintf calls = 8 / 105
```

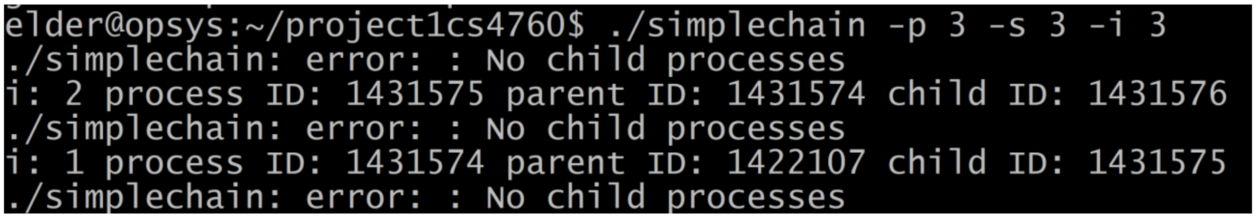


```
--------------------------------
Experiment 4 of 5
--------------------------------
Number of processes = 28
Sleep time between fprintf = 9
Iterations of fprintf = 2
Observed count of adopted processes from fprintf / Number of fprintf calls = 7 / 56
--------------------------------
Experiment 5 of 5
--------------------------------
Number of processes = 80
Sleep time between fprintf = 6
Iterations of fprintf = 2
Observed count of adopted processes from fprintf / Number of fprintf calls = 8 / 160
--------------------------------
```

6. Modify the code by adding the wait function call before the final fprintf statement. How does this affect the output of the program? Are you able to execute with a value of nprocs as 100?

In the simplechain program, the statement wait(NULL) was added before the final fprintf statement and before the sleep statement. The wait statement returns -1 if the current process is not waiting for a child process to finish. This error is reported through the perror message.

In the following screenshot, the program could not complete executing without returning an error. The third process is never printed because all parent processes are terminated from the perror. Also, the looping structure of fprintf does not iterate more than once.



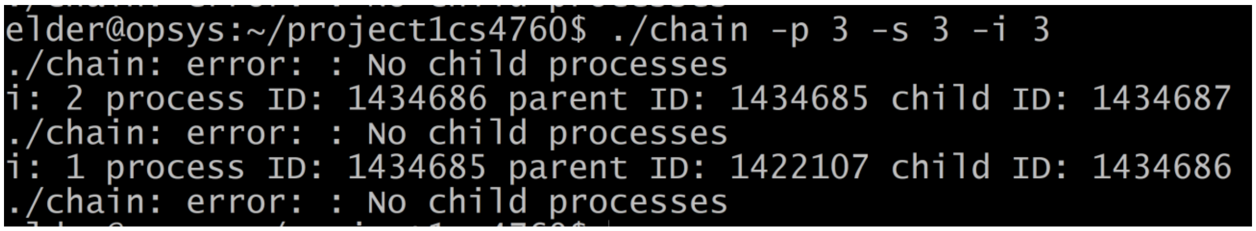
```
elder@opsys:~/project1cs4760$ ./simplechain -p 3 -s 3 -i 3
./simplechain: error: : No child processes
i: 2 process ID: 1431575 parent ID: 1431574 child ID: 1431576
./simplechain: error: : No child processes
i: 1 process ID: 1431574 parent ID: 1422107 child ID: 1431575
./simplechain: error: : No child processes
```

When the number of processes is set to 100, some looping does occur. However, some iterations do not complete 100 fprintf statements. Instead, the max number of processes for an iteration gets to around 22.

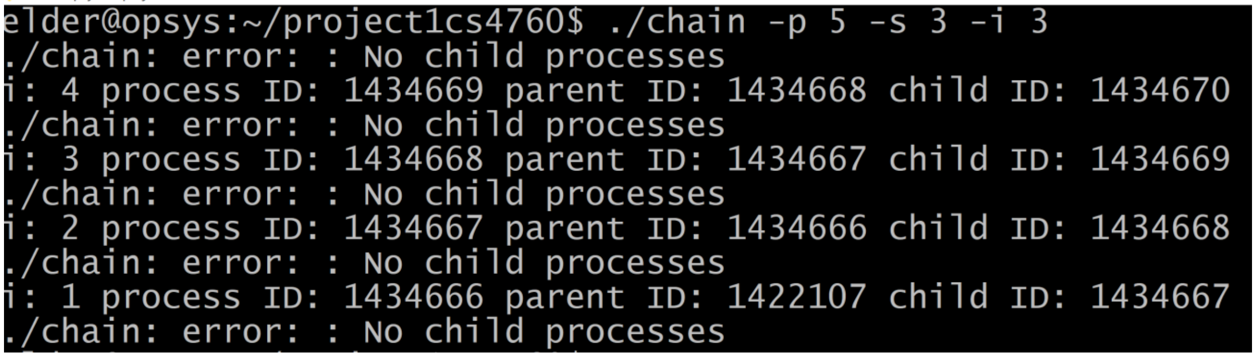
7. Modify your code by replacing the final fprintf statement with four fprintf statements, one each for the four integers displayed. Only the last one should output a newline. What happens when you run this program? Can you tell which process generated each part of the output? Run the program several times and see if there is a difference in the output.

Note: The program is correctly named in the following screenshots due a change from simplechain to chain.

After adding separate fprintf statements, the results are not different from the results of step 6. The wait/perror statements still prevent some iterations and processes from executing fprintf. Also, the sequence of printed statement is not affected from separate statements. The following



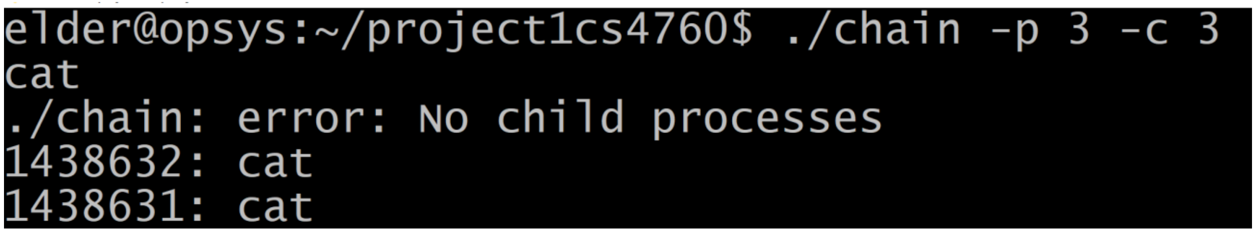
```
elder@opsys:~/project1cs4760$ ./chain -p 3 -s 3 -i 3
./chain: error: : No child processes
i: 2 process ID: 1434686 parent ID: 1434685 child ID: 1434687
./chain: error: : No child processes
i: 1 process ID: 1434685 parent ID: 1422107 child ID: 1434686
./chain: error: : No child processes
```



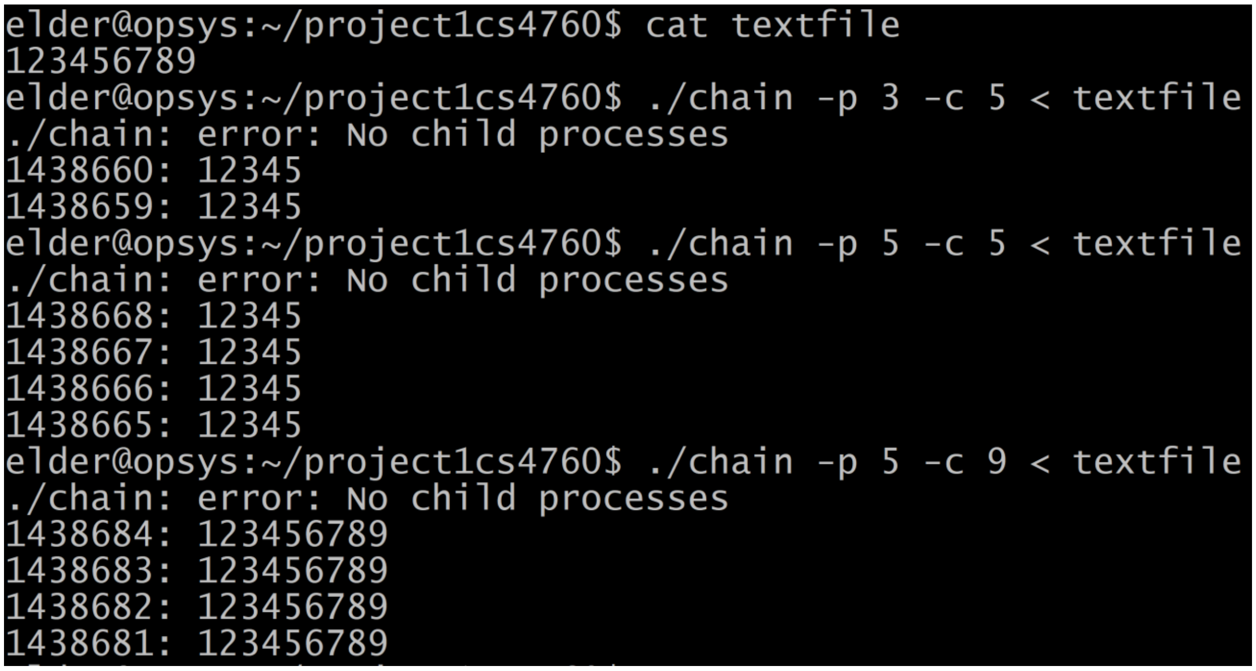
```
elder@opsys:~/project1cs4760$ ./chain -p 5 -s 3 -i 3
./chain: error: : No child processes
i: 4 process ID: 1434669 parent ID: 1434668 child ID: 1434670
./chain: error: : No child processes
i: 3 process ID: 1434668 parent ID: 1434667 child ID: 1434669
./chain: error: : No child processes
i: 2 process ID: 1434667 parent ID: 1434666 child ID: 1434668
./chain: error: : No child processes
i: 1 process ID: 1434666 parent ID: 1422107 child ID: 1434667
./chain: error: : No child processes
```

8. Modify your code by replacing the final fprintf statement with a loop that reads nchars characters from stdin one character at a time, and puts them in an array called mybuf. The values of nprocs and nchars should be passed as command line options. After the loop, put a ’\0’ character in index nchars of the array so that it contains a string. Output the PID of the process followed by a colon, a space, the string in mybuf, and a newline to stderr in a single fprintf. Run the program for several values of nprocs and nchars. Observe the results. Redirect the stdin from a file with some text that should be enough to make sure that all the processes terminate normally (more than nprocs × nchars characters).

In the next screenshot, the chain program is executed with arguments 3 and 3 for nprocs and nchars respectively. Entering the characters of “cat” through stdin prints the current process’ parent id and the char input. The wait/perror statements prevent the third process from printing.


```
elder@opsys:~/project1cs4760$ ./chain -p 3 -c 3
cat
./chain: error: No child processes
1438632: cat
1438631: cat
```

The following screenshot shows the output of a file containing the numerical characters of 1 through 9 in ascending order. The file is used as stdin for the chain program which reads the amount of characters specified in nchars.


```
elder@opsys:~/project1cs4760$ cat textfile
123456789
elder@opsys:~/project1cs4760$ ./chain -p 3 -c 5 < textfile
./chain: error: No child processes
1438660: 12345
1438659: 12345
elder@opsys:~/project1cs4760$ ./chain -p 5 -c 5 < textfile
./chain: error: No child processes
1438668: 12345
1438667: 12345
1438666: 12345
1438665: 12345
elder@opsys:~/project1cs4760$ ./chain -p 5 -c 9 < textfile
./chain: error: No child processes
1438684: 123456789
1438683: 123456789
1438682: 123456789
1438681: 123456789
```