James Peacemaker

Jan. 31, 2021

CPSC 405

FirstLab – Answer Document

Submissions

1. Submit your code on Canvas. Do not place code here.
2. Submit a run log of you executing the test cases provided. Copy/paste run log here.

I wasn’t sure if you wanted me to test the four files in the main FirstLab folder or do the test in the GradingTestScript subfolder, so I included both here.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstLab < test1\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstLab < test2\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Coletta, pid: 1029, ppid: 1028, priority: 1, state: EMBRYO

pname: Zac, pid: 1026, ppid: 1025, priority: 3, state: EMBRYO

pname: Gusty, pid: 1025, ppid: 1024, priority: 5, state: EMBRYO

pname: Jeremy, pid: 1028, ppid: 1025, priority: 7, state: EMBRYO

pname: Emily, pid: 1027, ppid: 1025, priority: 10, state: EMBRYO

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstLab < test3\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Coletta, pid: 1029, ppid: 1028, priority: 1, state: EMBRYO

pname: Jeremy, pid: 1028, ppid: 1025, priority: 7, state: EMBRYO

pname: Emily, pid: 1027, ppid: 1025, priority: 10, state: EMBRYO

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstLab < test4\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Coletta, pid: 1029, ppid: 1028, priority: 1, state: EMBRYO

pname: Jeremy, pid: 1028, ppid: 1025, priority: 7, state: EMBRYO

pname: Emily, pid: 1027, ppid: 1025, priority: 10, state: EMBRYO

procname: Emily, pid: 1027

procname: Emily, ppid: 1025

procname: Gusty not found

procname: Gusty not found

Grading Test Script output:

Test 1

# printprocs with only kernel proc

./firstLab < test1\_in.txt > testout.txt

diff ../test1\_out.txt testout.txt

test1 passed

Test 2

# fork 5 procs w/ diff prior and 3 parents

./firstLab < test2\_in.txt > testout.txt

diff ../test2\_out.txt testout.txt

test2 passed

Test 3

# fork 5 procs and kill 2

./firstLab < test3\_in.txt > testout.txt

diff ../test3\_out.txt testout.txt

test3 passed

Test 4

# fork procs and use printpid and printppid

./firstLab < test4\_in.txt > testout.txt

diff ../test4\_out.txt testout.txt

test4 passed

Test 5

# kill proc at end of list and kill nonexistent proc

./firstLab < test5\_in.txt > testout.txt

diff ../test5\_out.txt testout.txt

test5 passed

Test 6

# enqueue procs with same priority

./firstLab < test6\_in.txt > testout.txt

diff ../test6\_out.txt testout.txt

test6 passed

Test 7

# fork many procs

./firstLab < test7\_in.txt > testout.txt

diff ../test7\_out.txt testout.txt

test7 passed

Test 8

# Fork proc off a killed proc

./firstLab < test8\_in.txt > testout.txt

diff ../test8\_out.txt testout.txt

test8 passed

Test 9

# Attempt to kill kernel proc

./firstLab < test9\_in.txt > testout.txt

diff ../test9\_out.txt testout.txt

test9 passed

Test 10

# Fork procs with same name

./firstLab < test10\_in.txt > testout.txt

diff ../test10\_out.txt testout.txt

test10 passed

1. Create at least three other test cases and expected results. Think of some concepts you want to test. Describe each test case and its expected values. Simply adding another fork to an existing test case will not count as a test case. Copy/paste your test cases here.

Test case 5:

A test case where a process is added but the parent does not exist yet. The forks without parents should not be added.

fork Newton 12 James

fork Eden 9 Bettina

fork James 42 kernel\_process

fork Bettina 43 kernel\_process

fork Dolly 1 kernel\_process

printprocs

printpid Dolly

printppid Dolly

printpid Bettina

printppid Bettina

Test case 6:

A test case where a process is added more than once. The copies should be added in addition to the original.

fork James 42 kernel\_process

fork Bettina 43 kernel\_process

fork Newton 12 James

fork Eden 9 Bettina

fork Dolly 1 kernel\_process

fork Dolly 1 kernel\_process

printprocs

printpid Dolly

printppid Dolly

printpid Bettina

printppid Bettina

Test case 7:

A test to try and kill a process more than once, first it is added and killed twice, then just killed twice without being added again. It should not have a problem adding and killing. The last kill is ignored because the process is not there.

fork James 42 kernel\_process

fork Bettina 43 kernel\_process

fork Newton 12 James

fork Eden 9 Bettina

fork Dolly 1 kernel\_process

kill Dolly

fork Dolly 1 kernel\_process

kill Dolly

kill Dolly

printprocs

printpid Dolly

printppid Dolly

printpid Bettina

printppid Bettina

1. Submit a run log of you executing your test cases. Copy/paste run log here.

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstlab < test5\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Dolly, pid: 1029, ppid: 1024, priority: 1, state: EMBRYO

pname: James, pid: 1027, ppid: 1024, priority: 42, state: EMBRYO

pname: Bettina, pid: 1028, ppid: 1024, priority: 43, state: EMBRYO

procname: Dolly, pid: 1029

procname: Dolly, ppid: 1024

procname: Bettina, pid: 1028

procname: Bettina, ppid: 1024

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstlab < test6\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Dolly, pid: 1029, ppid: 1024, priority: 1, state: EMBRYO

pname: Dolly, pid: 1030, ppid: 1024, priority: 1, state: EMBRYO

pname: Eden, pid: 1028, ppid: 1026, priority: 9, state: EMBRYO

pname: Newton, pid: 1027, ppid: 1025, priority: 12, state: EMBRYO

pname: James, pid: 1025, ppid: 1024, priority: 42, state: EMBRYO

pname: Bettina, pid: 1026, ppid: 1024, priority: 43, state: EMBRYO

procname: Dolly, pid: 1029

procname: Dolly, ppid: 1024

procname: Bettina, pid: 1026

procname: Bettina, ppid: 1024

jpeacemakr@MyPrecious:/mnt/c/Users/Bettina/Desktop/cpsc405/Labs\_export/FirstLab$ ./firstlab < test7\_in.txt

procs in queue:

pname: kernel\_process, pid: 1024, ppid: 1024, priority: 0, state: RUNNABLE

pname: Eden, pid: 1028, ppid: 1026, priority: 9, state: EMBRYO

pname: Newton, pid: 1027, ppid: 1025, priority: 12, state: EMBRYO

pname: James, pid: 1025, ppid: 1024, priority: 42, state: EMBRYO

pname: Bettina, pid: 1026, ppid: 1024, priority: 43, state: EMBRYO

procname: Dolly not found

procname: Dolly not found

procname: Bettina, pid: 1026

procname: Bettina, ppid: 1024

1. Describe the algorithm used by the split module by answering the following questions.
   1. split returns a char \*\*. What is a char \*\*? Describe a char \*\*. Provide a concrete example of a char \*\*.

Split returns a char \*\*, which is a pointer to an array of strings. It is an array of arrays of char. In this case, it takes the line buffer and splits it up into chunks that can be used by different functions.

* 1. split has two loops. The first loop is a counting loop. What is the first loop counting?

It counts the number of spaces in the string to know how many times to split the string.

* 1. How does the second loop use the count computed in the fist

It uses the count to know how many strings it needs to create.

1. What relationships between files does the makefile define?

The makefile defines the relationship between main.c split.c split.h proc.c and proc.h

1. Can you solve the lab 1 problem with a singly linked list? Justify your answer.

A singly linked list could work but it would be cumbersome having to leave multiple placeholders as you traversed the list to keep track of the previous node.

1. Write a reflective report on your performance on lab 1. Your reflections must include the following.
   1. Whether you think the lab is easy/difficult.
   2. Why do you think the lab is easy/difficult?
   3. Do you think you could more easily solve this lab in another programming language? If so, why and what language?
   4. What aspects of C programming do you need to improve?
   5. What is your strategy for improving your C programming?
   6. Do you think it is important to improve your C programming?
   7. What part of lab 1 was most difficult for you?
   8. What part of lab 1 was easiest for you?

Place your reflective report here.

I thought the lab was hard for me personally because I had never programmed in C before. I had programmed in C++ about 20 years ago, which is probably also the last time I did a linked list. We used Ada95 and then C++ from 1996-2000 before I left in my senior year at what was then Mary Washington College. I returned in 2019 and have been taking one class a semester for the past few semesters to try and finally get my degree. I worked on brushing up on C syntax over winter break with online tutorials and the SoloLearn app, but I hadn’t done a real programming assignment in C until now.

I would have probably had an easier time using Java, because that was what I used in CPSC240. The past few programming classes I’ve had were all web programming and used Javascript, but I think Java would be better as a general purpose programming language. Java has a linked list class already built that I think would make it easier.

I need to improve on all aspects of C programming. I have a good feel for the logic involved, but the syntax is still a challenge. The way pointers and addresses and memory allocation are done was a bit foreign to me at first, but I am figuring it out.

I used the following sources as a reference for syntax but I did not copy code from them:

https://www.geeksforgeeks.org/linked-list-set-1-introduction/

https://www.techonthenet.com/c\_language/standard\_library\_functions/string\_h/strcpy.php

https://www.geeksforgeeks.org/enumeration-enum-c/