AUBURN UNIVERSITY

COMP 3500 Lab 2 Assignment

Instructions This lab assignment explores the data shared problem and process synchronization using Peterson's solution.

Objectives of this assignment:

- to work on a Unix based system
- to "*dust off*" your programming skills in C
- to understand the fork() function to create a"child" process
- to understand the relationship (or lack of) between parent and child process
- to experience the ***data shared*** problem
- to deploy the **Peterson's solution** to address the data shared problem

IMPORTANT:

1) *Your code will be tested and graded* ***REMOTELY*** *on the Engineering Unix (Tux) machines. If the code does not work on those machines, you will not get any credit even if your code works on any other machine.*
2) *A late submission will get a 50% penalty if submitted right after the deadline. The next day, you cannot submit the lab.*
3) *One submission per group.*
4) *Writing and presentation of your report are considered to grade your lab (30%). Your conclusions* ***must be supported*** *by the data/measurements you collect.*
5) *The quality of your code will be evaluated (****80%****).*
6) ***Questions about this lab must be posted on Piazza if you need a timely answer benefiting all students****.*

Use this file to answer the questions. Highlight your answers and do NOT remove anything from this file. Just Insert your answers.

Part I: Programming on Tux machines

(10 points) Program Exercise 1:

Exercise 1: **Download** the program ***lab2-1.c***. Compile it and execute it. Observe the code and observe the output. This program has a parent and child processes *sharing* a variable. This program is ***intended*** to increment the **shared (common)** variable counter **countptr*. The parent process is ***supposed*** to increment **countptr* by increments of **20** while the child increments by **2s**. A satisfactory execution of this program may be: the child increments the counter **countptr* twice (reaching 4), then the parent increments the counter **countptr* thrice to reach finally 64. Answer the following questions:

1) Does the program really execute as supposed (or intended)? Justify/Explain

2) Is the variable *****countptr*** really a shared (common) variable? In other words, are the changes made to **countptr* by the child visible by the parent, and *vice versa*? Explain.

(90 points) Program Exercise 2:

The program ***lab2*-2.*c*** creates a genuine **shared** variable **countptr*. Download, compile, and execute this program.

1) Based on the execution, show that **countptr* is now a genuine shared variable (*countptr* points to a zone shared by the parent and the child). Now, are the changes to **countptr* made by the child visible by the parent?

2) Does the program really execute as supposed (or intended), i.e, the counter increases exclusively in increments of 2 or 20? Explain what is happening.

3) **Without modifying** the routine *add_n()*, use the *Peterson's solution* to correct the program ***lab2-2.c***. to execute as intended: the variable should increase by 2's or twenty's

Hint: Besides the pointer ***countptr*** used to point to the shared memory zone, you need to map three other integers Interested[2] and Turn (Peterson's variables); These variables may be shared exactly the way that the zone pointed by *countptr* is shared.

What to turn in?
Electronic copy

Turn in separate files:

1) THIS file with INSERTED answers
2) Program ***lab2-2.***.c (corrected)

A penalty of 10 points will be applied if these instructions are not followed.

1) Your report must:
   a. state whether your code works. If is does work, state any issues you are aware of.
   b. Good writing and presentation are expected.