POSIX Thread Programming

Instructions for the template 1. Do not delete any of the words highlighted in yellow. Simply clear the highlighting formatting. 2. Delete all the instructions and tips – basically everything in this document will be erased or replaced by your own words unless it is highlighted in yellow.  
  
We have neither given nor received unauthorized assistance on this work.

**Team name: BestGroup**

Name of the VM: VirtualMachine (in mgibbon2/ & username: BestGroup)

Password: HaveAGreatDay24

Team members names:

Matthew Gibbons | Describe role in project results

Noah Storms | Describe role in project results

# Description of Project: 2 – 3 Sentences

Here write a few (2-3) sentences describing the purpose of the program. You may be more specific later in the pseudocode or conclusion section of this report. You should write in complete sentences to describe the purpose.

Pseudocode: This can be a picture, drawing, sketch, or digitally produced. For example, if you hand-wrote it on a sheet of paper, convert it into and image and paste it here. If you typed it in notepad, copy and paste it here.  
<START PART 1 PSEUDOCODE>  
n1 >= 2n2

(‘‘abcdab", ‘‘ab")

// don’t need validation checks because length of string 1 at least twice as long as string 2

NUM\_THREADS = 4

int currentlyAllocatedThreads = 0

int substringCount = 0

// charIndex = 0; 0 < 5 (6 – 2) + 1; 0 += 1  
// charIndex = 1; 1 < 5; 1 += 1

// charIndex = 2; 2 < 5; 2 += 1

// charIndex = 3; 3 < 5; 3 += 1

// charIndex = 4; 4 < 5; 4 += 1

// charIndex = 5; 5 !< 5  
for (int charIndex = 0; charIndex < n1 – n2 + 1; charIndex++)

// start thread 0, “abcdab”, “ab”, 6, 2, &(0 -> 1)

// thread 0 will also start at index 4 on next loop

// thread 0 “abcdab”, “ab”, 6, 2, &(1 -> 2)

// start thread 1, “abcdab”, “ab”, 6, 2, &substringCount

// start thread 2, “abcdab”, “ab”, 6, 2, &substringCount

// start thread 3, “abcdab”, “ab”, 6, 2, &substringCount

start thread (charIndex, s1, s2, n2, &substringCount)

}

// return 2

return substringCount

// thread 0, “abcdab”, “ab”, 2, &(0 -> 1)

// thread 1, “abcdab”, “ab”, 2, &(1)

// thread 2, “abcdab”, “ab”, 2, &(1)

// thread 3, “abcdab”, “ab”, 2, &(1)

// thread 4, “abcdab”, “ab”, 2, &(1 -> 2)

thread (int charIndex, string s1, string s2, int n1, int n2, int\* substringCountAddr) (

do {

// i = 0; 0 < 2; 0 += 1

// i = 1; 1 < 2; 1 += 1

// i = 2; 2 !< 2

for (int i = 0; i < n2; i++)

// THREAD 0

// s1[0 + 0] = ‘a’

// s2[0] = ‘a’

// ‘a’ = ‘a’

// s1[0 + 1] = ‘b’

// s2[1] = ‘b’

// ‘b’ = ‘b’

// completes for loop

// THREAD 1

// s1[1 + 0] = ‘b’

// s2[0] = ‘a’

// ‘b’ != ‘a’

// returns

// THREAD 2

// s1[2 + 0] = ‘c’

// s2[0] = ‘a’

// ‘c’ != ‘a’

// returns

// THREAD 3

// s1[3 + 0] = ‘c’

// s2[0] = ‘a’

// ‘c’ != ‘a’

// returns

// THREAD 0 second loop

// s1[4 + 0] = ‘a’

// s2[0] = ‘a’

// ‘a’ = ‘a’

// s1[0 + 1] = ‘b’

// s2[1] = ‘b’

// ‘b’ = ‘b’

// completes for loop

if s1[charIndex + i] doesn’t equal s2[i]

// thread 1 returns early

// thread 2 returns early

// thread 3 returns early

return

// thread 0 completed for loop

// thread 4 completed for loop

\*substringCountAddr++

charIndex += NUM\_THREAD

}

while (charIndex <= n2 – n1)

}  
  
other test cases:

(‘‘aaa", ‘‘a") = 3

(‘‘abac", ‘‘bc") = 0

# <END PART 1 PSEUDOCODE>

<START PART 2 PSEUDOCODE>

DEFINE BUFFER\_SIZE as 12

DEFINE buffer[BUFFER\_SIZE] // Circular buffer to store characters

DEFINE count = 0 // Number of items in the buffer

DEFINE in = 0 // Index for the producer to write

DEFINE out = 0 // Index for the consumer to read

DEFINE producer\_finished = 0 // Flag to indicate producer finished

DEFINE mutex // Mutex for thread synchronization

DEFINE cond\_producer // Condition variable for the producer

DEFINE cond\_consumer // Condition variable for the consumer

DEFINE producer function

OPEN file for reading

WHILE NOT end of file

LOCK mutex

WHILE count is equal to BUFFER\_SIZE

WAIT on cond\_producer using mutex

WRITE character from file to buffer[in]

in = (in + 1) MOD BUFFER\_SIZE

INCREMENT count

SIGNAL cond\_consumer

UNLOCK mutex

SET producer\_finished to 1

SIGNAL cond\_consumer

CLOSE file

DEFINE consumer function

WHILE TRUE

LOCK mutex

WHILE count is equal to 0 AND NOT producer\_finished

WAIT on cond\_consumer using mutex

IF producer\_finished AND count is equal to 0

UNLOCK mutex

BREAK

PRINT buffer[out]

out = (out + 1) MOD BUFFER\_SIZE

DECREMENT count

SIGNAL cond\_producer

UNLOCK mutex

CREATE producer thread

CREATE consumer thread

JOIN producer thread

JOIN consumer thread  
<END PART 2 PSEUDOCODE>

<START PART 3 PSEUDOCODE>

<END PART 3 PSEUDOCODE>  
<START PART 3 ANALYSIS>

list-forming.c

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K vs Num Threads | 2 | 4 | 8 | 16 | 32 | 64 |
| 200 | 420 | 1092 | 2443 | 4296 | 10154 | 15429 |
| 400 | 499 | 1391 | 3368 | 4968 | 14804 | 18098 |
| 800 | 1071 | 3904 | 4074 | 8327 | 18859 | 59812 |
| 1600 | 1666 | 4932 | 7631 | 21358 | 52955 | 92142 |

mylist-forming.c (local list of k nodes version)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K vs Num Threads | 2 | 4 | 8 | 16 | 32 | 64 |
| 200 | 418 | 569 | 2307 | 3138 | 6102 | 13359 |
| 400 | 451 | 585 | 1305 | 3480 | 6228 | 17219 |
| 800 | 424 | 775 | 1567 | 5138 | 9487 | 18665 |
| 1600 | 614 | 746 | 2704 | 5853 | 18108 | 46652 |

<END PART 3 ANALYSIS>

# Conclusion 1- 2 Paragraphs

Did the program behave the way you originally intended on the first try? If not, please identify and describe any challenges encountered, if there were several challenges, select just the most challenging part of the project. Examples could include coding and logical errors, difficulties encountered while writing the report, communication between team members, or other potential problems you now know to avoid not described here.

Describe in which ways the project could have improved? Do you feel as though you put forth your best efforts? Did your team function well as a group? If not, please share in which ways with the instructor, not in the conclusion of this report. Students who do not participate in group projects may submit a single submission that they created by themselves – it cannot be the same submission the group you are assigned to submitted if you did not participate in the group work.

# Lessons Learned

You are not confined to the following prompts, they are examples of the types of questions you might ponder while determining what your lessons learned were. Each team member can submit a separate “Lessons Learned” section to the dropbox in Canvas, or each student can paste a paragraph response in this document. Were you able to complete the project basically the first time through with little to no errors? Describe how you utilized the hints (if you needed to)? Describe which part of the project you learned the most from, what you learned, and how you feel it could benefit you someday in the future. Can you relate it to a potential career you may choose? How did completing this project contribute to your knowledge of how computing and OS (in general) behave? You may also add general take-aways about the assignment itself, if you would like. You may consider addressing: do you feel the assignment was about the appropriate breadth and depth you would expect from a higher level CS course?