Swapnil Acharya

StarId: ea4963aw Due Date: 2/19/2020

Purpose

Thread Synchronization and SMTP protocol

CSCI 610  
 Advanced concepts in operating systems

Homework 1

Problem A

# @author Swapnil Acharya

# @since 02/14/2020

# @file Makefile

all**:** hw1

hw1**:** hw1\_threadSyncFinal.cpp

g++ -std=c++0x -pthread -o hw1 hw1\_threadSyncFinal.cpp

clean**:**

rm -f hw1

/\*\*

\***@author** Swapnil Acharya

\***@since** 02/14/2020

\***@file** hw1\_threadSyncFinal.cpp

\*Description: demonstrates thread synchronization in a shared variable environment

\*\*/

#include <stdio.h>

#include <iostream>

#include <pthread.h>

#include <unistd.h>

#include <errno.h>

#include <string>

#include <stdbool.h>

**using** **namespace** std**;**

const int MAXPEERS **=** 8**;** //MAX NUMER OF PEERS

const int SLEEP\_DURATION **=** 1**;** //SLEEP\_DURATION for thread synchronization

//variables for programs flow control

int MAILING\_SYSTEM **=** 0**;**

int WAIT\_REMOVE\_COUNT **=** 0**;**

int WAIT\_SET\_COUNT **=** 0**;**

//Declarre mutexes

pthread\_mutex\_t Out **=** PTHREAD\_MUTEX\_INITIALIZER**;**

pthread\_mutex\_t Mutex1 **=** PTHREAD\_MUTEX\_INITIALIZER**;**

pthread\_mutex\_t Mutex2 **=** PTHREAD\_MUTEX\_INITIALIZER**;**

//declare conditional variables to be used within mutexes

pthread\_cond\_t Wait\_For\_Set\_BroadCast **=** PTHREAD\_COND\_INITIALIZER**;**

pthread\_cond\_t Wait\_For\_Remove\_BroadCast **=** PTHREAD\_COND\_INITIALIZER**;**

void println **(**string s**)** **{**

int \_err **=** pthread\_mutex\_lock**(&**Out**);**

**if(**\_err **==** **-**1**)**

**{**

printf**(**"PrintLn, Couldnot lock Mutex, errno =%i\n"**,**errno**);**

**}**

cout **<<** s **<<** endl**;**

\_err **=** pthread\_mutex\_unlock**(&**Out**);**

**if(**\_err **==** **-**1**)**

**{**

printf**(**"Println, Could not unlock Mutex, errno=%i\n"**,**errno**);**

**}**

**}**

class Mailbox**;**

class Letter **{**

private**:**

Mailbox**\*** recipientList**[**MAXPEERS**];**

int id**;**

int count**,** next**;**

public**:**

Letter **(**int identity**)** **{**

**for** **(**int i **=** 0**;** i **<** MAXPEERS**;** i**++)**

recipientList**[**i**]** **=** **(**Mailbox **\*)nullptr;**

id **=** identity**;**

next**=** 0**;**

count **=** 0**;**

**}**

int getId **()** **{**

**return** id**;**

**}**

void setRecipient **(**Mailbox**\*** mb**)** **{**

recipientList**[**next**++]** **=** mb**;**

**if** **(**next **>** MAXPEERS**)**

next **=** 0**;**

**}**

Mailbox**\*** getNext**(**Mailbox**\*** mb**)** **{**

int current **=** 0**;**

**while** **(**recipientList**[**current**]** **!=** mb**)**

current **=** **(**current **+** 1**)** **%** MAXPEERS**;**

next **=** **(**current **+** 1**)** **%** MAXPEERS**;**

**while** **(**recipientList**[**next**]** **==** **(**Mailbox **\*)nullptr)**

next **=** **(**next **+** 1**)** **%** MAXPEERS**;**

count**++;**

**return** **(**recipientList**[**next**]);**

**}**

void removeRecipient **(**Mailbox**\*** mb**)** **{**

int current **=** 0**;**

**while** **(**recipientList**[**current**]** **!=** mb**)**

current **=** **(**current **+** 1**)** **%** MAXPEERS**;**

recipientList**[**current**]** **=** **(**Mailbox **\*)nullptr;**

**}**

**};**

class Mailbox **{**

private**:**

Letter**\*** letter**;**

public**:**

Mailbox **()** **{**

letter **=** **(**Letter **\*)nullptr;**

**}**

void setLetter **(**Letter**\*** l**){**

letter **=** l**;**

**}**

Letter**\*** removeLetter **(**int id**){**

int \_err **=** pthread\_mutex\_lock**(&**Mutex1**);**

**if(**\_err **==** **-**1**){**

println**(**"RemoveLetter, Could not lock Mutex1"**);**

**}**

Letter**\*** myletter **=** letter**;**

**if(**MAILING\_SYSTEM **==** 0**){**

//let initial remove letter requests through without any locks

myletter **=** letter**;**

letter **=** **(**Letter **\*)nullptr;**

**}**

**else** **if(**MAILING\_SYSTEM **==** 1**){**

//let the second set of letter requests pass through

//conditional locks dependinf on program flow

**if(**WAIT\_REMOVE\_COUNT **<** MAXPEERS**-**1**){**

//In this case untill all the peers have set letters and

//now are ready to get next set of letter make them wait

//untill all the threads are waitting to get next set of letters

WAIT\_REMOVE\_COUNT **++;**

//wait for broadcast signal

pthread\_cond\_wait**(&**Wait\_For\_Remove\_BroadCast**,&**Mutex1**);**

myletter **=** letter**;**

letter **=** **(**Letter **\*)nullptr;**

**}**

**else** **if(**WAIT\_REMOVE\_COUNT **==** MAXPEERS**-**1**){**

//when MAXPEERS - 1, peers are waiting to removeletter,

//let only the Peer# Maxpeers removes the letter and

//then after than broacasr removeletter signal so All

//other waiting peers can remove letters

myletter **=** letter**;**

letter **=** **(**Letter **\*)nullptr;**

WAIT\_REMOVE\_COUNT **=** 0**;**

//send broadcast signal

pthread\_cond\_broadcast**(&**Wait\_For\_Remove\_BroadCast**);**

**}**

**}**

**if(**myletter **==** **(**Letter **\*)nullptr){**

//check to see if accessed letter in null or hasnt been set

println**(**"Peer " **+** to\_string**(**id**)** **+** " : Acesses Null"**);**

**}**

\_err **=** pthread\_mutex\_unlock**(&**Mutex1**);**

**if(**\_err **==** **-**1**){**

println**(**"RemoveLetter, Could not unlock Mutex1"**);**

**}**

**return** myletter**;**

**}**

**};**

class Peer **{**

private**:**

int id**;**

int count**;**

Letter**\*** letter**;**

Mailbox**\*** mb**;**

bool \_initialLockRemoval**;**

public**:**

Peer **(**int identifier**,** Mailbox**\*** mbx**){**

id **=** identifier**;**

mb **=** mbx**;**

count **=** 0**;**

**}**

int getId**(){**

**return** id**;**

**}**

void **operator** **()** **(){**

run**();**

**}**

void run **(** **)** **{**

**while** **(**count **!=** MAXPEERS**)** **{**

letter **=** mb**->**removeLetter**(**id**);**

Mailbox**\*** nextMB **=** letter**->**getNext**(**mb**);**

count**++;**

println **(**"Peer " **+** to\_string**(**id**)** **+** " : has letter " **+** to\_string**(**letter**->**getId**())** **+** ", count = " **+** to\_string**(**count**));**

int \_err **=** pthread\_mutex\_lock**(&**Mutex2**);**

**if(**\_err **==** **-**1**){**

println**(**"Run, Could not lock Mutex2"**);**

**}**

**if(**WAIT\_SET\_COUNT **<** MAXPEERS**-**1**){**

// make MAXPEERS -1, wait to set their letters untill

// all (MAXPEERS - 1), threads have arrived to set letter

WAIT\_SET\_COUNT**++;**

//wait for boradcast signal

pthread\_cond\_wait**(&**Wait\_For\_Set\_BroadCast**,&**Mutex2**);**

**if** **(**count **<** MAXPEERS**){**

nextMB **->** setLetter**(**letter**);**

**}**

**}**

**else** **if(**WAIT\_SET\_COUNT **==** MAXPEERS **-** 1**){**

//when the final peer#MAXPEERS arrives to set letter

//then let it set letter and then broadcast

//signal to let other waiting peers to set leters as well

**if** **(**count **<** MAXPEERS**){**

nextMB **->** setLetter**(**letter**);**

**}**

WAIT\_SET\_COUNT **=** 0**;**

//when first set of letter are fetched change the flow of program

MAILING\_SYSTEM **=** 1**;**

//send broadcast signal

pthread\_cond\_broadcast**(&**Wait\_For\_Set\_BroadCast**);**

**}**

\_err **=** pthread\_mutex\_unlock**(&**Mutex2**);**

**if(**\_err **==** **-**1**){**

println**(**"Run, Could not unlock Mutex2"**);**

**}**

**}**

**}**

**};**

/\*\*

\***@brief** this function is used to as start routine for threads

\***@pre** this function requires a Peer object

\***@post** this function start the targeted run function for a thread

\*/

void **\*** startPeerThreads**(**void **\*** \_peer**){**

//typcasting void \* type to Peer class type

Peer **\*** \_aPeer **=** **((**Peer **\*)**\_peer**);**

//stating the threads run functio

\_aPeer **->** run**();**

**}**

int main **(**int argc**,** const char **\*** argv**[])** **{**

//make array letters

int i **=** 0**;**

Letter**\*** letterList**[**MAXPEERS**];**

**for(**i **=** 0**;** i **<** MAXPEERS**;** i**++){**

//make new letter and assign each letter a id

letterList**[**i**]** **=** **new** Letter**(**100**+**i**);**

**}**

//make array of mailboxes

i **=** 0**;**

Mailbox**\*** mailBoxList**[**MAXPEERS**];**

**for(**i **=** 0**;** i**<** MAXPEERS**;** i**++){**

//make new mailbox

mailBoxList**[**i**]** **=** **new** Mailbox**();**

**}**

//set all mailboxes as recipients for all Letters

i **=** 0**;**

int j **=** 0**;**

**for(**i **=** 0**;** i **<** MAXPEERS**;** i**++){**

**for(** j **=** 0**;** j **<** MAXPEERS**;** j**++){**

letterList**[**i**]->**setRecipient**(**mailBoxList**[**j**]);**

**}**

**}**

//give each mailbox a letter

i **=** 0**;**

**for(**i **=** 0**;** i **<** MAXPEERS**;** i**++){**

mailBoxList**[**i**]->**setLetter**(**letterList**[**i**]);**

**}**

//create array of peers and assign each pair 1 mailBox

i **=** 0**;**

Peer**\*** peerList**[**MAXPEERS**];**

**for(** i **=**0**;** i **<** MAXPEERS**;** i**++){**

peerList**[**i**]** **=** **new** Peer**(**i**,**mailBoxList**[**i**]);**

**}**

//create array of threads

pthread\_t PeerWorkers**[**MAXPEERS**];**

i **=** 0**;**

**for(**i **=** 0**;** i **<** MAXPEERS**;** i**++){**

pthread\_create**(&**PeerWorkers**[**i**],NULL,**startPeerThreads**,**peerList**[**i**]);** //start threads

**}**

println **(**"All peers are active."**);**

//join the threads into main thread

i **=** 0**;**

**for(** i **=** 0**;** i **<** MAXPEERS**;** i**++){**

pthread\_join**(**PeerWorkers**[**i**],NULL);**

**}**

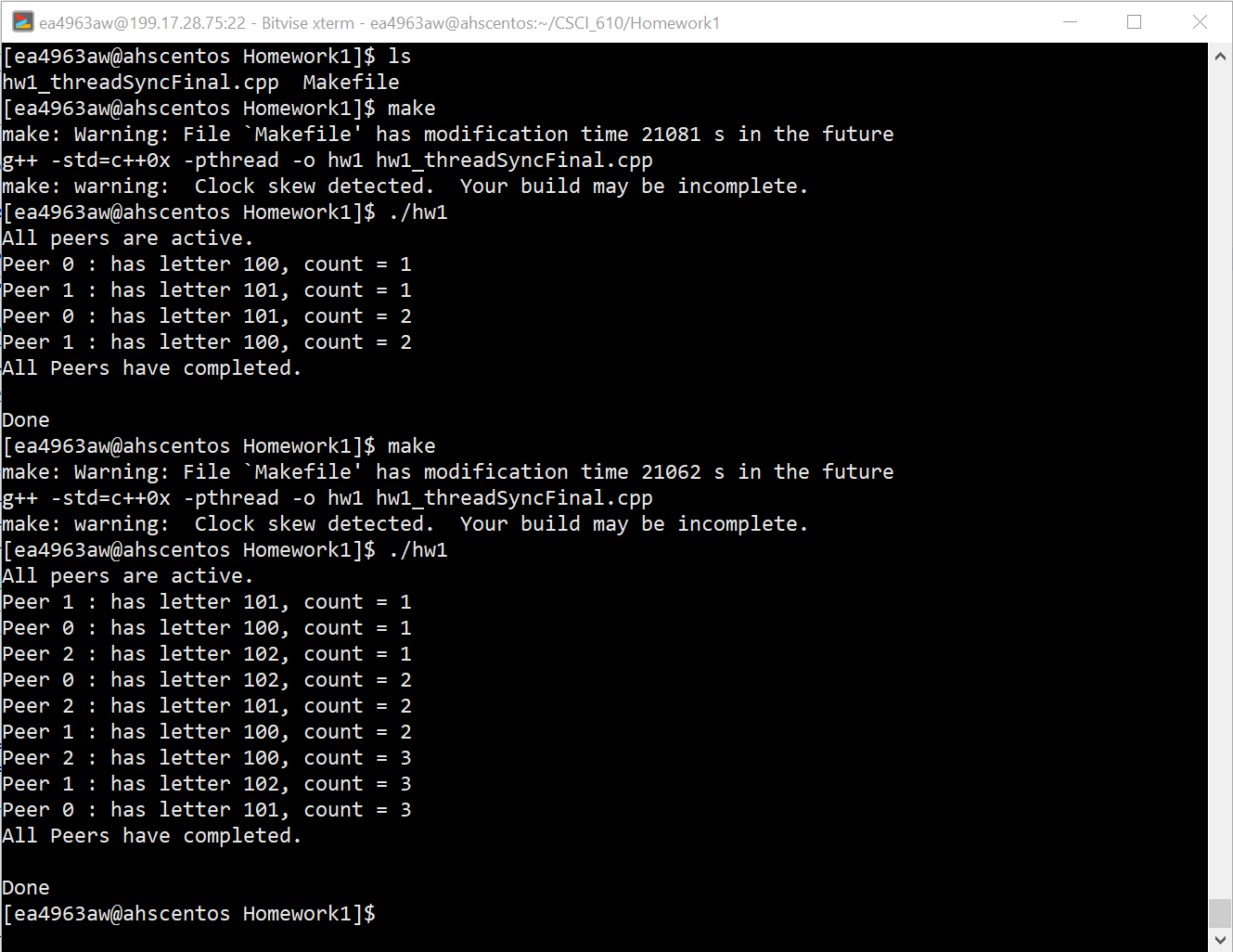
println **(**"All Peers have completed."**);**

printf**(**"\nDone\n"**);**

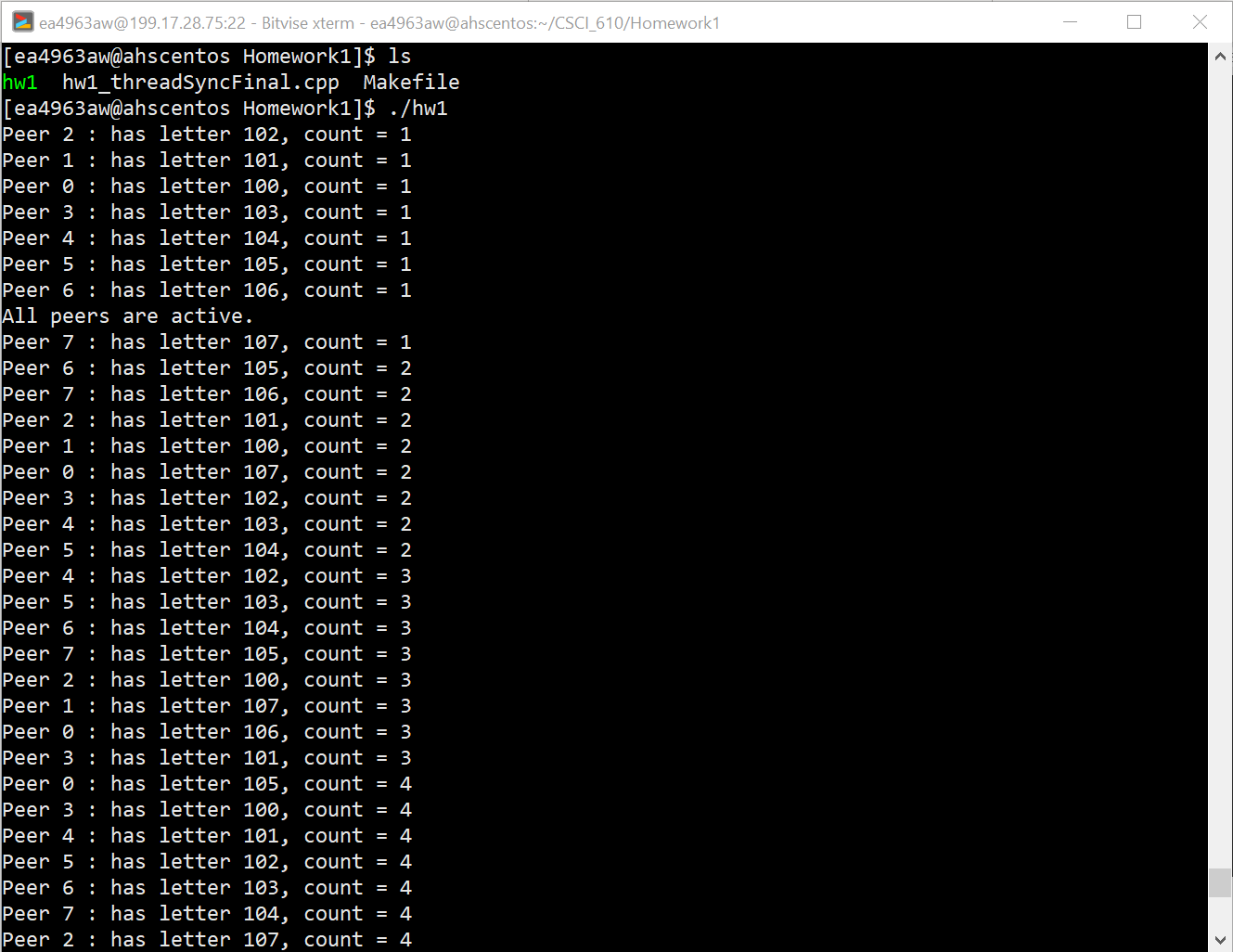
**return** 0**;**

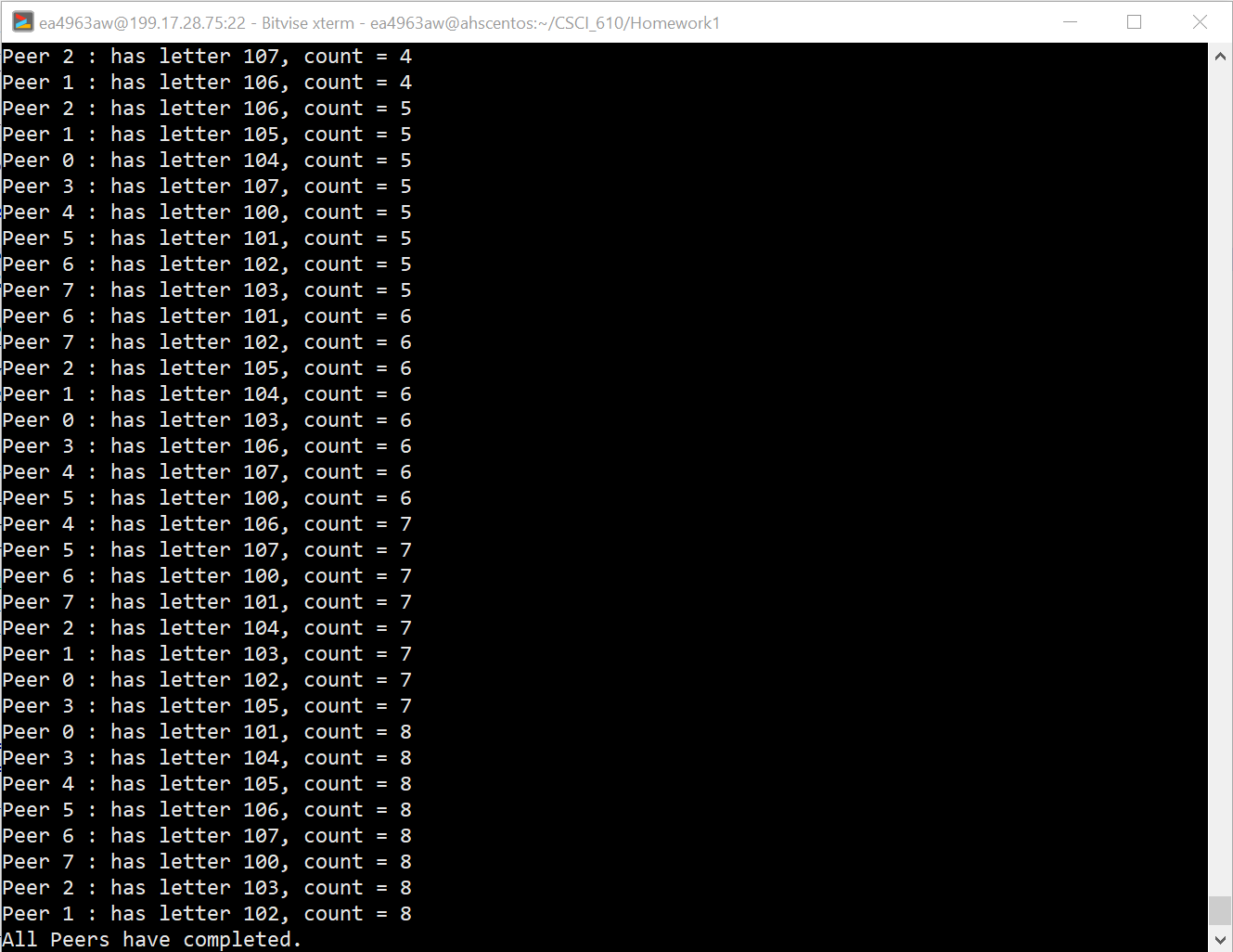
**}**

MAXPEERS = 2 and 3



MAXPEERS = 8





Problem B

1. What is the value of AAAA?

25

1. What is the value of BBBB?

EHLO

1. What is the value of CCCC?

MAIL FROM

1. What is the value of DDDD?  
   RCPT TO
2. What is the value of EEEE?  
   DATA