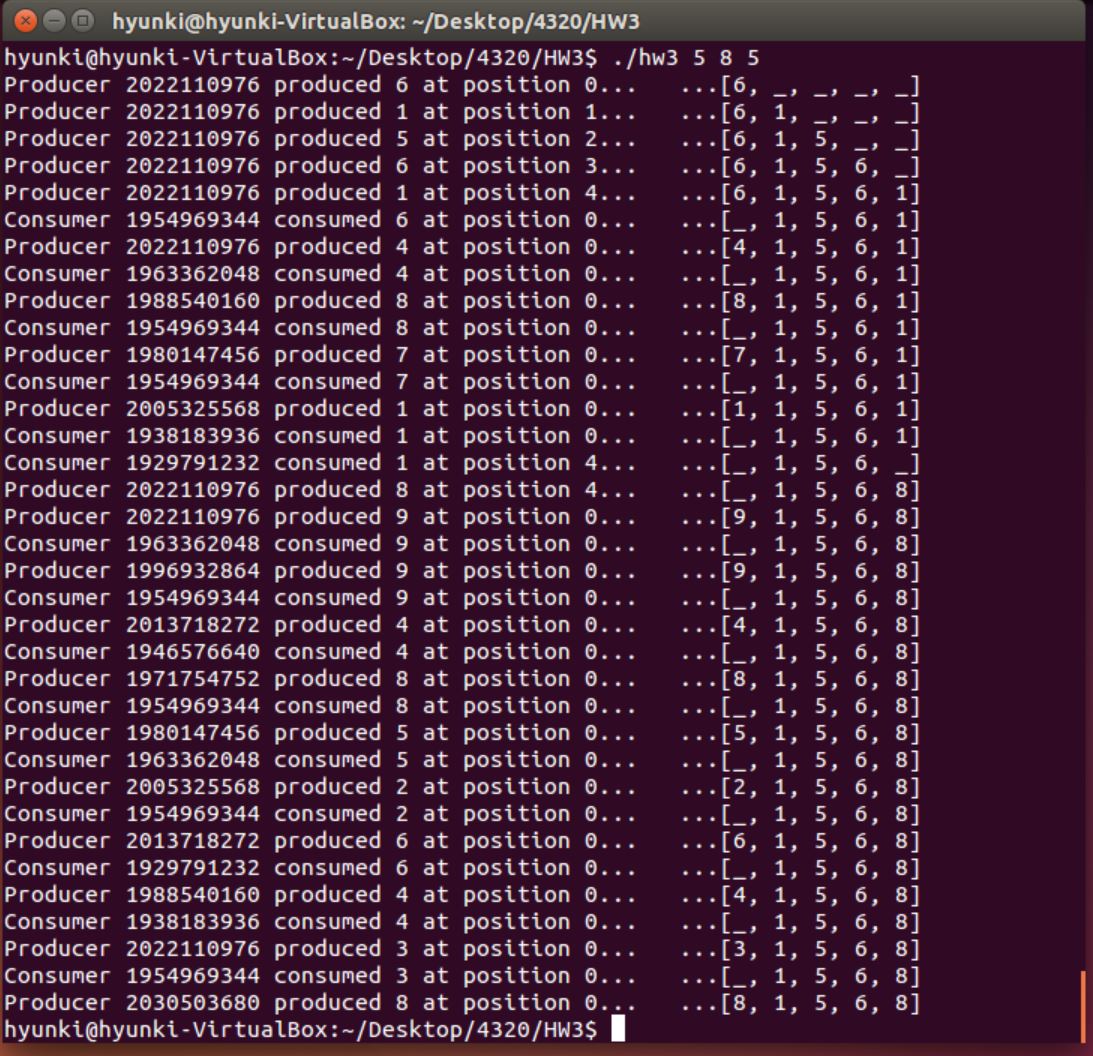
HomeWork3

CSC4320

#002-34-4677

Hyunki Lee

1. Screenshot of output



1. C source code

#include "buffer.h"

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <time.h>

buffer\_item buffer[BUFFER\_SIZE];

sem\_t empty;

sem\_t full;

pthread\_mutex\_t mutex;

int i = 0;

int leftPos = 0;

int rightPos = BUFFER\_SIZE-1;

int numOfItemInBuffer;

int pos = 0;

void \*producer(void \*param);

void \*consumer(void \*param);

int insert\_item(buffer\_item item)

{

int check = 0;

/\* Acquire Empty Semaphore \*/

sem\_wait(&empty);

/\* Acquire mutex lock to protect buffer \*/

pthread\_mutex\_lock(&mutex);

/\* Insert item into buffer \*/

if(numOfItemInBuffer == 0){

pos = 0;

leftPos = 0;

rightPos = BUFFER\_SIZE-1;

buffer[pos] = item;

numOfItemInBuffer++;

}

else if(numOfItemInBuffer == BUFFER\_SIZE){

check = -1;

}

else if(leftPos <= rightPos){

pos = BUFFER\_SIZE - rightPos;

rightPos--;

buffer[pos] = item;

numOfItemInBuffer++;

}

else{

pos = leftPos - 1;

leftPos--;

buffer[pos] = item;

numOfItemInBuffer++;

}

printf("Producer %u produced %d at position %d... ...[", (unsigned int)pthread\_self(), item, pos);

for( i =0; i < BUFFER\_SIZE; i++){

if(buffer[i] == 0){

printf("\_");

}else{

printf("%d", buffer[i]);

}

if(i != (BUFFER\_SIZE-1)){

printf(", ");

}

}

printf("]\n");

/\* Release mutex lock and full semaphore \*/

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

return check;

}

int remove\_item(buffer\_item \*item)

{

int check = 0;

/\* Acquire Full Semaphore \*/

sem\_wait(&full);

/\* Acquire mutex lock to protect buffer \*/

pthread\_mutex\_lock(&mutex);

/\* remove an object from buffer placing it in item \*/

if(numOfItemInBuffer == 0){

check = -1;

}

else if(leftPos <= rightPos){

pos = leftPos;

leftPos++;

\*item = buffer[pos];

buffer[pos] = 0;

numOfItemInBuffer--;

}

else{

pos = (BUFFER\_SIZE-rightPos)-1;

rightPos++;

\*item = buffer[pos];

buffer[pos] = 0;

numOfItemInBuffer--;

}

printf("Consumer %u consumed %d at position %d... ...[",(unsigned int)pthread\_self(), \*item, pos);

for( i = 0; i < BUFFER\_SIZE; i++){

if(buffer[i] == 0){

printf("\_");

}else{

printf("%d", buffer[i]);

}

if(i != (BUFFER\_SIZE-1)){

printf(", ");

}

}

printf("]\n");

/\* Release mutex lock and empty semaphore \*/

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

return check;

}

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

{

/\* Get command line arguments argv[1],argv[2],argv[3] \*/

if(argc !=4){

fprintf(stderr, "USAGE:./hw3 <sleeptime> <# of producer threads> <# of consumer threads>\n");

}

int sleepTime = atoi(argv[1]);

int num\_Pro\_Threads = atoi(argv[2]);

if(num\_Pro\_Threads < 1){

fprintf(stderr, "USAGE: <# of producer threads> should be larger than 1.\n");

}

int num\_Con\_Threads = atoi(argv[3]);

if(num\_Con\_Threads < 1){

fprintf(stderr, "USAGE: <# of consumer threads> should be larger than 1.\n");

}

/\* Initialize buffer related synchronization tools \*/

int j;

pthread\_mutex\_init(&mutex, NULL);

sem\_init(&empty, 0, BUFFER\_SIZE);

sem\_init(&full, 0, 0);

srand(time(NULL));

/\* Create producer threads based on the command line input \*/

pthread\_t pro[num\_Pro\_Threads];

for(j = 0; j < num\_Pro\_Threads; j++){

pthread\_create(&pro[j], NULL, producer, NULL);

}

/\* Create consumer threads based on the command line input \*/

pthread\_t con[num\_Con\_Threads];

for(j = 0; j < num\_Con\_Threads; j++){

pthread\_create(&con[j], NULL, consumer, NULL);

}

/\* Sleep for user specified time based on the command line input \*/

sleep(sleepTime\*3);

return 0;

}

void \*producer(void \*param)

{

/\* producer thread that calls insert\_item() \*/

int ranNum = rand() % 10;

while(1){

sleep(ranNum);

buffer\_item item = (rand() % 9)+1;

insert\_item(item);

}

}

void \*consumer(void \*param)

{

int ranNum = rand() % 10;

/\* consumer thread that calls remove\_item() \*/

while(1){

sleep(ranNum);

buffer\_item item;

remove\_item(&item);

}

}