Homework #5

[ECE30021/ITP30002] Operating Systems

Mission

- Implement producer-consumer synchronization using pthread, POSIX mutex locks, and unnamed semaphores
  - Develop on Ubuntu Linux (Google cloud or VirtualBox) using vim and gcc.

    Ex) gcc hw5.c -pthread

- Submission
  - Submit a hw5_<student_id>.c file

- Due date: PM 11:00, May 31th

Honor Code Guidelines

- “Assignment”
  - Assignments are an educational activity necessary to fully understand the lecture, and to apply the materials to practical problems. Students should complete all assignments with honesty and sincerity to develop the knowledge and skills intended in the assignment.
  - Submitted assignments are reflected in a grade evaluation. Therefore, student should never commit any kinds of dishonest behavior including acquisition, utilization, request, and providing of unauthorized information or assistance that can disrupt the fairness of evaluation.
  - It is allowed to get help from peers in order to understand the lecture materials, announcements, or the directions for homework. However, each student should complete the entire assignments on their own.
  - Submitting assignments or program codes written by others or acquired from the internet without explicit approval of the professor is regarded as cheating.
  - Showing or lending one’s own homework to other student is also considered cheating that disturbs fair evaluation and hinders the academic achievement of the other student.
  - It is regarded as cheating if two or more students conduct their homework together and submit it individually when the homework is not a group assignment.
  - One may receive help from other students if a problem in the assignment could not be solved even with the best effort; however, the assistance should never go beyond the correction of basic grammatical errors. It is cheating to get help about design, logic, and algorithm required in assignments from other person than the professor, TA, and tutor.
  - It can be suspected or regarded as cheating if the similarity between assignments submitted by different students is beyond an acceptable degree that can be considered as a coincidence or when the student is not able to explain in detail about their homework.

Honor Code Guidelines

- “과제”
  - 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
  - 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
  - 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
  - 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
  - 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
  - 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
  - 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
  - 서로 다른 학생이 제출한 제출물간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

Reading Assignment

- Search Internet for the documents about the following functions and read them carefully.
  - POSIX mutex lock functions
    - pthread_mutex_init()
    - pthread_mutex_lock()
    - pthread_mutex_unlock()
  - POSIX semaphore functions
    - sem_init()
    - sem_wait()
    - sem_post()
    - sem_getvalue()
    - sem_destroy()

Example Binary File

- Running hw5_example
  - Upload hw5_example to Ubuntu and modify permission

    chmod u+x hw5_example

  - Execution

    ./hw5_example [duration]

    - duration: specifies how long(in sec) the program runs. Default value is 30 sec.

An Example

```
$ ./hw5_example 10
Buffer contents:
        count = 0, in = 0, out = 0
[Producer] Created a message "You are genius!"
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "You are genius!" ==> buffer
Buffer contents:
        count = 1, in = 1, out = 0
        0] You are genius!
Producer is leaving critical section.
------------------------------------------------------------
---------------------- CONSUMER -------------------------
Consumer is entering critical section.
[Consumer] buffer ==> "You are genius!"
Buffer contents:
        count = 0, in = 1, out = 1
Consumer is leaving critical section.
------------------------------------------------------------
[Consumer] Consumed a message "You are genius!"
[Producer] Created a message "God loves you and has a wonderful plan for your life."
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "God loves you and has a wonderful plan for your life." ==> buffer
Buffer contents:
        count = 1, in = 2, out = 1
        1] God loves you and has a wonderful plan for your life.
Producer is leaving critical section.
------------------------------------------------------------
---------------------- CONSUMER -------------------------
Consumer is entering critical section.
[Consumer] buffer ==> "God loves you and has a wonderful plan for your life."
Buffer contents:
        count = 0, in = 2, out = 2
Consumer is leaving critical section.
------------------------------------------------------------
[Consumer] Consumed a message "God loves you and has a wonderful plan for your life."
[Producer] Created a message "I love you! God loves you!"
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "I love you! God loves you!" ==> buffer
Buffer contents:
        count = 1, in = 3, out = 2
        2] I love you! God loves you!
Producer is leaving critical section.
------------------------------------------------------------
---------------------- CONSUMER -------------------------
Consumer is entering critical section.
[Consumer] buffer ==> "I love you! God loves you!"
Buffer contents:
        count = 0, in = 3, out = 3
Consumer is leaving critical section.
------------------------------------------------------------
[Consumer] Consumed a message "I love you! God loves you!"
[Producer] Created a message "You are so precious!"
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "You are so precious!" ==> buffer
Buffer contents:
        count = 1, in = 4, out = 3
        3] You are so precious!
Producer is leaving critical section.
------------------------------------------------------------
---------------------- CONSUMER -------------------------
Consumer is entering critical section.
[Consumer] buffer ==> "You are so precious!"
Buffer contents:
        count = 0, in = 4, out = 4
Consumer is leaving critical section.
------------------------------------------------------------
[Consumer] Consumed a message "You are so precious!"
[Producer] Created a message "You are genius!"
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "You are genius!" ==> buffer
Buffer contents:
        count = 1, in = 0, out = 4
        4] You are genius!
Producer is leaving critical section.
------------------------------------------------------------
[Producer] Created a message "You are so precious!"
---------------------- PRODUCER -------------------------
Producer is entering critical section.
[Producer] "You are so precious!" ==> buffer
Buffer contents:
        count = 2, in = 1, out = 4
        4] You are genius!
        0] You are so precious!
Producer is leaving critical section.
------------------------------------------------------------

Bye!
```

Global Declarations

- Message buffer
  - #define BUFFER_SIZE 5
  - #define MAX_MESSAGE 64
  - char buffer[BUFFER_SIZE][MAX_MESSAGE];
  - int in = 0, out = 0, count = 0;
- Mutex and Semaphores
  - pthread_mutex_t mutex;
  - sem_t empty, full;

  Note. You need to include proper header files to use mutex locks and semaphores.
- Global variable to control threads
  - int repeat = 1; // Producer and Consumer threads terminates
    // when main() sets *repeat* to zero.

main()

- The behavior of main()
  - Read *duration* from the command line arguments.
  - Initialize
    - Random seed “srand(time(NULL));”
    - *mutex*, *full*, and *empty*.
  - Display the initial content of buffer.
    - Call DisplayBuffer()
  - Create threads for producer and consumer.
  - Wait for *duration* seconds.
  - Set *repeat* to zero to terminate producer and Consumer.
  - If the value of *full* is zero, call sem_post(&full);
  - If the value of *empty* is zero, call sem_post(&empty);
  - Wait for the producer and consumer.
  - Destroy *mutex*, *full*, and *empty*.
  - Print a good-bye message.

Producer

- The behavior of producer
  - Repeat until *repeat* is set to zero.
    - Randomly wait for 1~3 seconds. (sleep(rand() % 3 + 1);)
    - Generate a message by randomly choosing one of the following candidates.

```
int no_messages = 10;
char *messages[64] =  {
    "Nice to see you!",
    “Aal izz well!",
    "I love you! God loves you!",
    "God loves you and has a wonderful plan for your life.",
    "God bless you!",
    "You are genius!",
    "Cheer up!",
    "Everything is gonna be okay.",
    "You are so precious!",
    "Hakuna matata!"
 };
```

    - Print a message indicating the message was produced.

Producer (cont'd)

- Implement entry section using mutex lock and semaphores
- Print a message indicating the start of critical section.
- Add the message to the buffer and increase *counter*.
- Display the content of buffer.
  - Call DisplayBuffer()
- Print a message indicating the end of critical section.
- Implement exit section using mutex lock and semaphores

Consumer

- The behavior of consumer
  - Repeat until *repeat* is set to zero.
    - Implement entry section using mutex lock and semaphores
    - Print a message indicating the start of critical section.
    - Delete the message from the buffer and decrease *counter*.
    - Print a message indicating a message was retrieved from the buffer.
    - Display the content of buffer.
      - Call DisplayBuffer()
    - Print a message indicating the end of critical section.
    - Implement exit section using mutex lock and semaphores
    - Print a message indicating the message was consumed.
    - Randomly wait for 2~4 seconds. (sleep(rand() % 3 + 2);)

DisplayBuffer()

```
void DisplayBuffer()
{
    printf("Buffer contents:₩n");
    printf("₩tcount = %d, in = %d, out = %d₩n",
                                                count, in, out);

    int p = out;
    for(int i = 0; i < count; i++){
        printf("₩t%d] %s₩n", p, buffer[p]);
        p = (p + 1) % BUFFER_SIZE;
    }
}
```