Spring 2019 COMP 3511 Homework Assignment #3
Handout Date: March 30, 2019 Due Date: April 12, 2019

Name: _Huang Daoji_ ID: _20623420_E-Mail: _dhuangal@connect.ust.hk_

Please read the following instructions carefully before answering the questions:

- You should finish the homework assignment **individually**.
- There are total of **4** questions.
- When you write your answers, please try to be precise and concise.
- Fill in your name, student ID, email and Section number at the top of each page.
- Please fill in your answers in the space provided.

- **<u>Homework Collection</u>:** the homework is submitted to **assignment #3** on **CASS**

1. [20 points] Multiple choices

   1) The first readers-writers problem ____.

      A) requires that, once a writer is ready, that writer performs its write as soon as possible.
      B) is not used to test synchronization primitives.
      C) requires that no reader will be kept waiting unless a writer has already obtained permission to use the shared database.
      D) requires that no reader will be kept waiting unless a reader has already obtained permission to use the shared database.
      Answer: _C_

   2) Which of the following conditions must be satisfied to solve the critical section problem?

      ①Aging ②Mutual Exclusion ③Deadlock ④Progress ⑤Bounded Waiting
      A) ①②③⑤
      B) ②③④⑤
      C) ②④⑤
      D) ③④⑤
      Answer: _C_

   3) Assume an adaptive mutex is used for accessing shared data on a Solaris system with multiprocessing capabilities. Which of the following statements is not true?

      A) A waiting thread may spin while waiting for the lock to become available.
      B) A waiting thread may sleep while waiting for the lock to become available.
      C) The adaptive mutex is only used to protect short segments of code.
      D) Condition variables and semaphores are never used in place of an adaptive mutex.
      Answer: _D_

4) ____________ occurs when a higher-priority process needs to access a data structure that is currently being accessed by a lower-priority process.

A) Deadlock
B) Priority inversion
C) A race condition
D) A critical section
Answer: _B_

5) A deadlocked state occurs whenever ____.

A) a process is waiting for I/O to a device that does not exist
B) the system has no available free resources
C) every process in a set is waiting for an event that can only be caused by another process in the set
D) a process is unable to release its request for a resource after use
Answer: _C_

6) Which of the following condition is required for deadlock to be possible?

A) Mutual exclusion
B) A process may hold allocated resources while awaiting assignment of other resources.
C) No resource can be forcibly removed from a process holding it.
D) All of the mentioned.
Answer: _D_

7) Suppose that there are three processes and ten resources of the same type. The current resource allocation and the maximum need of each process is given below, which of the following correctly characterizes this state?

Process	Maximum Needs	Current Allocation
P0	10	4
P1	3	1
P2	6	4

A) It is not safe.
B) It is safe.
C) The state cannot be determined.
D) It is an impossible state.
Answer: _A_

8) Absolute code can be generated for ____.

A) compile-time binding
B) load-time binding
C) execution-time binding
D) interrupt binding
Answer: _A_

9) Which of the following is true of compaction?

A) It can be done at assembly, load, or execution time.
B) It is used to solve the problem of internal fragmentation.
C) It cannot shuffle memory contents.
D) It is possible only if relocation is dynamic and done at execution time.
Answer: _D_

10) _____ is the dynamic storage-allocation algorithm which results in the largest leftover hole in memory.

A) First fit
B) Best fit
C) Worst fit
D) None of the above
Answer: _C_

2. [30 points] Synchronization

1) Some semaphore implementations provide a function getValue() that returns the current value of a semaphore. This function may, for instance, be invoked prior to calling wait() so that a process will only call wait() if the value of the semaphore is > 0, thereby preventing blocking while waiting for the semaphore. For example:

```
if (getValue(&sem) > 0)
  wait(&sem);
```

What is the problem in this approach? (5 points)

i. The process may bypass the wait(&sem) call and continue executing code after that, which may be against the mutual exclusion constraint. If the semaphore sem's value is zero, there may be other process executing in their critical section, and other process should be waiting for the semaphore, not bypassing the waiting just for the semaphore being zero.

ii. If the sem semaphore is used as a counting semaphore, bypassing the waiting call may result in wrong value.

2) Briefly describe what a reader-writer lock is for and why it can be more efficient than semaphores in some cases. (5 points)

i. The reader-writer lock is for reader-writer problems. It allows multiple processes accessing the same data simutanously while write operation require exclusive access to the shared data

ii. It allows multiple readers accessing the shared data by nature, and result in more simple code, while a single semaphore does not allow multiple readers to read at the same time, and the solution to reader-writer problem is more complicated.

iii. In the case that there are multiple readers, a reader-writer lock will be more efficient.

3) Considering the first reader-writer solution below, please explain which semaphores that readers and writers are waiting on when there is a writer in inside the Critical Section updating shared data. (5 points)

Writer

```
do {
   wait(rw_mutex);
   ...
   /* writing is performed */
   ...
   signal(rw_mutex);
} while (true);
```

Reader

```
do {
  wait(mutex);
  read_count++;
  if (read_count == 1)
    wait(rw_mutex);
  signal(mutex)
  ...
  /* reading is performed */
  ...
  wait(mutex);
  read_count--;
  if (read_count == 0)
    signal(rw_mutex);
  signal(mutex);
} while (true);
```

i. All other writers are waiting for the rw_mutex semaphore.

ii. The first reader is waiting for rw_mutex and other readers are waiting for mutex. That is for the first reader first aquired the mutex semaphore and has been waiting for rw_mutex while holding the mutex, which other readers are waiting for.

4) Given a condition variable x, Consider the following implementation of x.signal() using semaphores. Please explain whether this a Hoare monitor or Mesa monitor, and why? (5 points)

```
if (x_count > 0) {
  next_count++;
  signal(x_sem);
  wait(next);
  next_count--;
}
```

i. It is a hoare monitor

ii. In a hoare monitor, a process has to wait for other process signal the semaphore once it signal the same semaphore

iii. In the implementation of x.signal(), the process has to wait for the next semaphore to return from the x.signal() call. This correcpondes to the behaviour of a hoare monitor.

5) You are asked to implement a different reader-writer solution. There are two classes of processes accessing shared data, *readers* and *writers*. Readers never modify data, thus multiple readers can access the shared data simultaneously. Writers modify shared data, so at most one writer can access data (no other writers or readers). This solution gives priority to writers in the following manner: when a reader tries to access shared data, if there is a writer accessing the data or if there are any writer(s) waiting to access shared data, the reader must wait. In another word, readers must wait for all writer(s) if any to update shared data -- a reader can access shared data only when there is no writer either accessing or waiting.

Variables:

```
State variables (protected by a lock called “lock”
condition okToRead = NIL; /* readers waiting queue */
condition okToWrite = NIL; /* writers waiting queue */
int R_count = 0; /* number of readers accessing data */
int W_count = 0; /* number of writer accessing data */
int WR_count = 0; /* number of readers waiting */
int WW_count = 0; /* number of writers waiting*/
```

The writer code is given below. Please design the Reader's code. (10 points)

```
Writer() {
  // Writer tries to enter
  lock.acquire();
  while ((R_count + W_count) > 0) {// Is it safe to write?
    WW_count++; // Update the counter of waiting writers
    okToWrite.wait(&lock); // Waiting on condition variable,
               atomically release the lock, regain the lock later
    WW_count--; // No longer waiting
  }
  W_count++; // Writer inside
  lock.release();

  // Perform actual read/write access

  // Writer finishes update
  lock.acquire();
  W_count--; // No longer active
  if (WW_count > 0){ // Give priority to writers
    okToWrite.signal(); // Wake up one writer
  } else if (WR_count > 0) {// Otherwise, wake up readers
    okToRead.broadcast(); // Wake up all waiting readers
  }
  lock.relesae();
}
```

Answer:

```
Reader(){
  // Reader tries to enter
  lock.acquire();
  while((W_count + WW_count) > 0){ // Is it safe to read?
    WR_count++; // Update to counter of waiting readers
    okToRead.wait(&lock); // Waiting on conditional variable
    WR_count--; // No longer waiting
  }
  R_count++; // Reader inside
  lock.release();

  // Perform actual read access

  // Reader finishes update
  lock.acquire();
  R_count--; // No longer active
  if(R_count == 0 && WW_count > 0){ // do not check WW_count
would also be fine, for foo.signal() does nothing then
    okToWrite.signal(); // Try to wake up one writer(if any)
  }
  lock.release();
```

3. [30 points] Deadlock

1) What does a deadlock prevention mechanism do? Use an example to illustrate why this can lead to low resource utilization. (5 points)

i. It provides a set of methods to ensure at least one of the necessary conditions (mutual exclusion, hold and wait, no preemption and circular wait)cannot hold.

ii. For example, if the OS try to meke sure 'hold and wait' cannot hold, it may require each process to be allocated all of its requested resources while it may not be able to use these much at the same time, or a process may only request resources when it has none, which also leads to low utilization of resources.

2) Please briefly explain the two methods of deadlock recovery. (5 points)

i. The two methods are process termination and resource preemption

ii. Process termination method will abort all processes that are in deadlock state or abort one chosen process at one time until deadlock is eliminated.

iii. Resource peremption method will preempt some resources form processes and give these resources to others until no deadlock happens. It may involve selscting a victim resource / process, and rollback to some safe state, and restart from that state.

3) Consider a system with three processes and twelve instances of one type of resource. The current resource allocation and the maximum need of each process is given below, please find a safe sequence. (5 points)

Process	Maximum Needs	Current Allocation
P0	10	5
P1	4	2
P2	9	2

i. Safe sequence may be <P1, P0, P2>

ii. work = 3 (<= 2) -> select P1, work = 5 (<= 5) -> select P0, work = 10 (<= 7), select P2, done.

4) Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	0 0 1 1	1 0 2 2	1 2 1 1
P_1	3 0 4 0	5 6 6 2	
P_2	1 0 2 0	6 4 3 1	
P_3	1 1 0 0	1 2 2 1	
P_4	0 4 1 1	2 4 3 3	

Answer the following questions using the banker's algorithm (15 points)

a) Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete. (5 points)

i. <P0, P3, P4, P1, P2>

ii. work: <1,2,1,1> -> <1,2,2,2> -> <2,3,2,2> -> <2,7,3,3> -> <5,7,7,3> -> <6,7,9,3>

b) If a request from process P4 arrives for (1, 0, 0, 0), can the request be granted immediately? (5 points)

i. No, if this request is granted, avail = <0,2,1,1>, then no process will be able to make progress(with regrad to resource request).

c) If a request from process P1 arrives for (0, 1, 0, 0), can the request be granted immediately? (5 points)

i. Yes, the safe sequence is still <P0, P3, P4, P1, P2>

ii. work: <1,1,1,1> -> <1,1,2,2> -> <2,2,2,2> -> <2,6,3,3> -> <5,7,7,3> -> <6,7,9,3>

4. [20 points] Memory management

1) Briefly describe internal and external fragmentation, and methods to mitigate the problems. (5 points)

   1. External fragmentation occurs when total memory space exists to satisfy a request, but it is not contiguous with scattered holes inside
   2. Internal fragmentation is when the memory allocated to a process is larger than requested, so part of the memory allocated is not being used.
   3. External fragmentation can be redused by compaction: shuffle memory contents to place the free memory together
   4. Another solution may be to allow the logical address space to be non-contiguous, include techniques such as segmentation and paging

2) Consider the following segment table:

Segment	Base	Length
0	0000010000000000	001011101110
1	0010000000100000	011110011110
2	0011000000000000	010100011110
3	0100000000000010	010110001100
4	1000000000001000	010100010110

Consider the following 16-bit logical addresses with 4-bit segment and 12-bit offset, what are the physical addresses of them? (15 points)

a) 0001001011110000

b) 0000100011101110

c) 0010010100010000

d) 0011010010000000

e) 0100000100010000

a. 0010000000100000+001011110000 = 0010001100010000

b. Addressing error, offset larger than length

c. 0011000000000000+010100010000 = 0011010100010000

d. 0100000000000010+010010000000 = 0100010010000010

e. 1000000000001000+000100010000 = 1000000100011000