CSCI 3431.1 Fall, 2017 – Assignment 3 Operating Systems

0.1 Practice Exercises

1. [2 points] A distributed system using mailboxes has two IPC primitives, send and receive. The receive primitive specifies a process to receive from and blocks if no message from that process is available, even though messages may be waiting from other processes. There are no shared resources, but processes need to communicate frequently about other matters. Is deadlock possible? Discuss you answer.

Solutions:

Since there is not shared resources, but processes need to communicate frequently about other matters, so the deadlock is possible, because mutual exclusion, Hold and wait, No preemption and Circular wait, these four conditions can been satisfied.

For example, the initial condition is that all of the mailboxes is empty. Now, A send message to B and waiting response from B, B sends message to C and waiting response from C, if C sends messages to A and waiting the response from A. in this case the deadlock will occur.

Another example, receive process is receiving an email and writing it to hard disk, after that, due to some reasons, this process has been suspended. As a result, it is blocked, but this receive process owned the IO file handler. When it became active again, maybe it need to wait from another IO process, unfortunately, this second IO process need to wait the third process: another IO process which is waiting for the first receive process. In this case, deadlock occurs.

2. [3 points] Consider the following state of a system with five processes, P0, P1, P2, P3, and P4, and four types of resources A-D.

a. What is the content of the matrix Need?

Solutions:

Since the Need[i,j]=Max[i,j]-Allocation[i,j], so the Need matrix is as follows:

P0: 0 0 0 0

P1: 0 7 5 0

P2: 1 0 0 2

P3: 0 0 2 0

P4: 0 6 4 2

b. Is the process in a safe state?

Solutions:

YES, the process in a safe state when applying banker algorithm.

Details is as follows:

Initially, the Available is 1 5 2 0, so Work is 1 5 2 0.

P0 with Need (0 0 0 0), so firstly, schedule P0

Processes	Work A B C D	Need A B C D	Allocation A B C D	Work + Allocation A B C D	Finish
P0	1 5 2 0	0 0 0 0	0 0 1 2	1 5 3 2	True
P3	1 5 3 2	0 0 2 0	0 6 3 2		
P1		0 7 5 0	1 0 0 0		
P2		1 0 0 2	1 3 5 4		
P4		0 6 4 2	0 0 1 4		

Secondly, after P0 release the resources, current Work is 1 5 3 2, among P1 to P4, finding P3 needs 0 0 2 0, so, P3 can be scheduled.

Processes	Work A B C D	Need A B C D	Allocation A B C D	Work + Allocation A B C D	Finish
P0	1 5 2 0	0 0 0 0	0 0 1 2	1 5 3 2	True
P3	1 5 3 2	0 0 2 0	0 6 3 2	1 11 6 4	True
P1	1 11 6 4	0 7 5 0	1 0 0 0		
P2		1 0 0 2	1 3 5 4		
P4		0 6 4 2	0 0 1 4		

Thirdly, after P3 release the resources, current Work is 1 11 6 4, among P1, P2,P4, all of them can be scheduled. For example, selecting P1.

Processes	Work A B C D	Need A B C D	Allocation A B C D	Work + Allocation A B C D	Finish
P0	1 5 2 0	0 0 0 0	0 0 1 2	1 5 3 2	True
P3	1 5 3 2	0 0 2 0	0 6 3 2	1 11 6 4	True
P1	1 11 6 4	0 7 5 0	1 0 0 0	2 11 6 4	True
P2	2 11 6 4	1 0 0 2	1 3 5 4	3 14 11 8	True
P4		0 6 4 2	0 0 1 4		

Then, after P1 release the resources, current Work is 2 11 6 4, among P2,P4, all of them can be scheduled. For example, selecting P2.

Processes	Work A B C D	Need A B C D	Allocation A B C D	Work + Allocation A B C D	Finish
P0	1 5 2 0	0 0 0 0	0 0 1 2	1 5 3 2	True
P3	1 5 3 2	0 0 2 0	0 6 3 2	1 11 6 4	True
P1	1 11 6 4	0 7 5 0	1 0 0 0	2 11 6 4	True
P2	2 11 6 4	1 0 0 2	1 3 5 4	3 14 11 8	True
P4	3 14 11 8	0 6 4 2	0 0 1 4	3 14 12 12	true

Finally, P4 can get the sufficient resource to run.

In conclusion, in this time point, all of the process can be scheduled with an order without deadlock, so this is in a safe status.

c. If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately?

Solutions:

YES

Available (1 5 2 0)

P1 Need(0 4 2 0), so Available becomes (1 1 0 0), according to check the safe state using the above method, we can find a sequence P0,P2,P3, P4 in a safe state.

3. [3 points] Explain the difference between deadlock, livelock, and starvation.

Deadlock means that there are two or more threads, they are blocked because of waiting for each other forever.

Livelock means that there are two or more threads, but they are response to each other forever.

The difference between deadlock and livelock is if the thread itself is blocked. In deadlock, the threads are blocked in order to get resource but need to wait from each others. But in livelock, the threads are not blocked, they just could not forward because there is no responses from each other's.

Starvation means a situation where a thread is unable to get access to resources and is unable to make forward due to some scheduler algorithms.

4. [4 points] The four conditions (mutual exclusion, hold and wait, no preemption and circular wait) are necessary for a resource deadlock to occur.

a. [3 points] Give an example to show that these conditions are not sufficient for a resource deadlock to occur.

Solutions:

For example, process A need process B or process K for one IO device. process B needs process C, but process C needs process A. in this situation, if process K release the resource, process A can run without deadlock. This means that even the above situation has the four conditions, IO devices are the mutual exclusion. Process A hold and wait for process B or process K. process C hold the IO device as well, and could not be pre-emption. So process A, B and C are in a circular wait. But if the process K release the IO device, process A can run without deadlock.

Another example:

Assume that there are 3 process A, B and C. there are two types of resource R1 and R2. There are one R1 but two R2. The scenario is: A request R1 and hold it; B request R2 and hold it; C request the second R2 and hold it. Then, B request R1 but blocked, A request R2 but blocked. In this case, all of the four deadlock conditions are satisfied, but no deadlock. If C relase R2, then R2 can be assigned to A, and when A release R1, R1 can be assigned to B.

b. [1 points] When are these conditions sufficient for a resource deadlock to occur?

Solutions:

Also the above example, if there is not process K, process A must wait for process B, there will be a deadlock.

In other words, if each resource has only one, the deadlock will occur, otherwise, these four conditions are not sufficient for deadlock.

5. [3 points] In order to control network traffic, a router A, periodically sends a message to its neighbor, B, telling it to adjust the number of packets that it can handle. At some point, Router A is flooded with traffic and sends B a message asking to stop sending packets. It does this by specifying that the number of bytes B may sent to A is 0 (A's window size = 0). As traffic surges decrease, A sends a new message, telling B to restart transmission. It does this by increasing the window size from 0 to a positive number. That message is lost. As described, neither side will ever transmit.

a. [1 point] Can you consider the situation as a deadlock?

Solutions:

NO, this could not be considered as a deadlock. Because there are four conditions for deadlock.

mutual exclusion

Hold and wait

No preemption

Circular wait

In this case, there are not these four conditions.

b. [2 points] If yes, what type of deadlock is this? If no, please justify how network traffic resumes between A and B.

Solutions:

The Router A and B is in a circular wait scenario. It means that router A think the router B has received the signal for resending data. But the router B do not, it is waiting for the resending signals.

In order to solve this issue, there are two ways as follows:

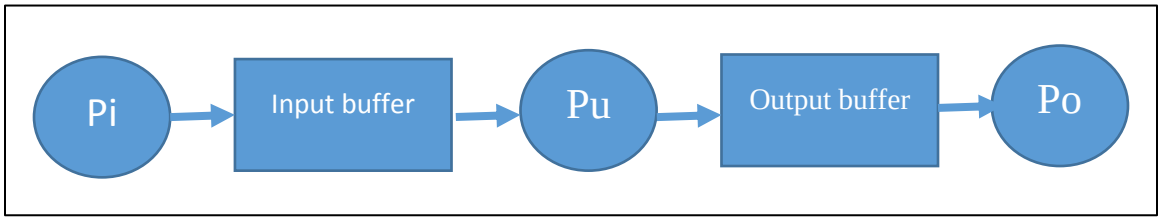
Router A has a time-out detection mechanism, if more than one specific time without the response from router B after sending a signal, router A will resend this signal.

Another method comes from router B, even receiving the stop sending signal from router A, router B will ask if the transaction can be resumed by router A regularly.

6. [8 points] A spooling system consists of an input process I, a user process P, and an output process O connected by two buffers. The processes exchange data in blocks of equal size. These blocks are buffered on a disk using floating boundary between the input and the output buffers, depending on the speed of the processes. Let max denote the maximum number of blocks on disk, i denotes the number of input blocks on disk, and o denotes the number of output blocks on disk. The communication primitives used ensure that the following resource constraint is satisfied: i + o <= max.

a) [4 points] Show that this system can become deadlocked.

Solutions:



For example, when the buffer in disk is full with input data but without output data. This is possible because the Process output is fast than Process input. In this case, i=max but o=0. When Pi try to write data into input buffer, Pi lock the input buffer but find the buffer is full, so

Pi is blocked and wait to resume. At the same time, Pu want to transfer data from input buffer to output buffer, when Pu try to lock the input buffer, it find that Pi has locked the input buffer. So Pu has to wait from Pi. But Pi is waiting for resume. So deadlock occurs.

The above sample is an assumption that the Pi will lock the whole input buffer. If the data is stored in input buffer by blocks. It`s also possible for deadlock. For example: assume that the input process Pi is slow than output process Po, so there will be a situation that i=max-2, and o is zero. It means that there are two blocks available for input but no blocks in output buffer. In this situation, the process output Po will be blocked. At this time, a new input request comes with 2 blocks, it means the input process needs two blocks to write, when Pi hold the first block and try to hold the second block. But at the same time, the user process Pu hold the second block in order to transfer the data from input buffer to output buffer. In this case, the deadlock appears as well. The Pi wait for Pu to release the second block, but the Pu wait for Pi to release the first block.

b) [4 points] Suggest additional resource constraint that will prevent the deadlock, but still permit the boundary between input and output buffers to vary in accordance with the present needs of the processes.

Solutions:

Allows Po and Pu to read data at the same time;

Deny Pi and Pu to write data at the same time;

When writing data to input buffer or output buffer, it should wait for all of readers release the resources.

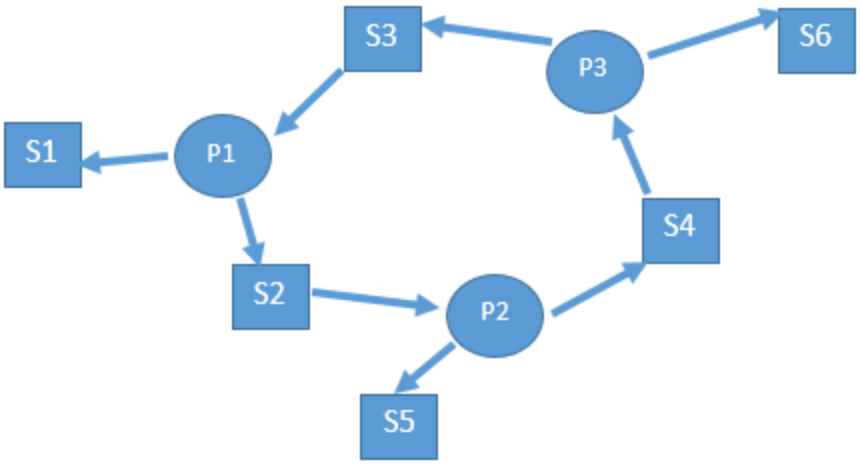
Through the above additional resource constraint, the deadlock will be avoided.

7. [7 points] Assume the following code snippets for three processes P1, P2 and P3. All processes are competing for 6 resources labeled S1…S6

a. [3 points] Using a resource allocation graph (see Fig. 7.1-7.3), show the possibility of a deadlock in this implementation.

Solutions:

The follows indicate that there is a cycle in an RAG, and there is probably a deadlock in this implementations.



b. [4 points] Modify the order of some of the get requests to prevent the possibility of any deadlock. You cannot move requests across procedures, only change the order inside each procedure. Use a resource allocation graph to justify your answer.

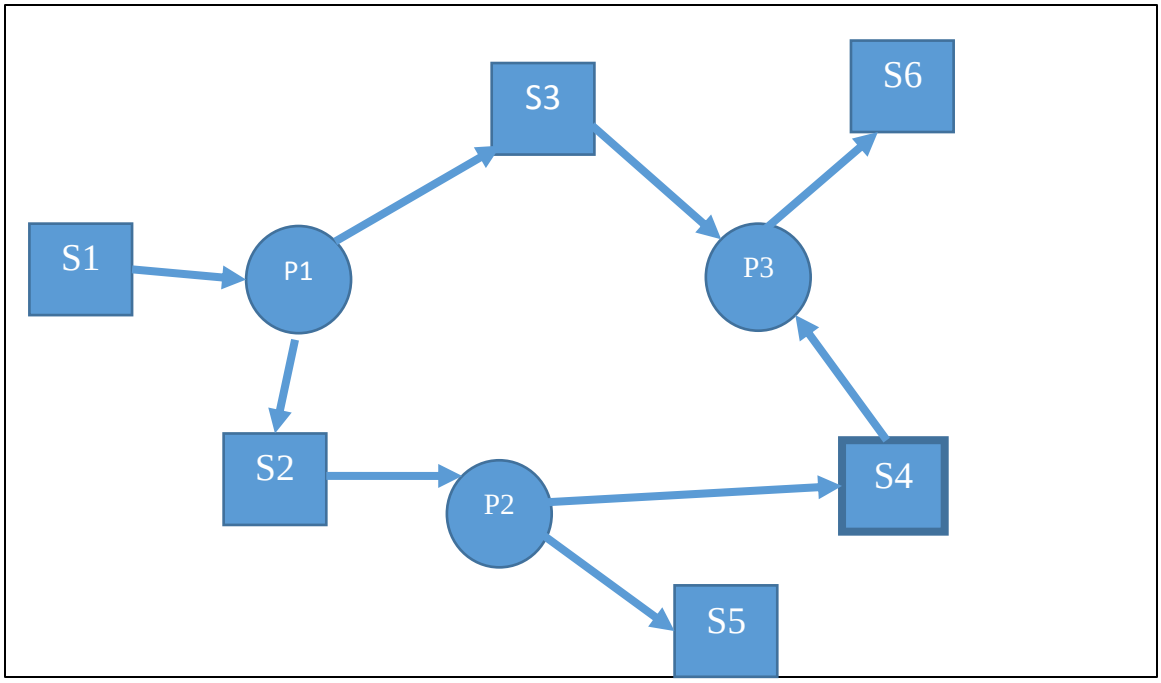
Solutions:

```
void P1(){
        while (1){
                get(S3);
                get(S2);
                get(S1);
/* Critical Section:  Use S1, S2, S3 */
                release(S3);
                release(S2);
                release(S1);
}
}
```

```
void P2(){
```

```
        while (1){
                get(S5);
                get(S4);
                get(S2);
                /* Critical Section:
                Use S4, S5, S2 */
                release(S5);
                release(S4);
                release(S2);
        }
}

void P3(){
        while (1){
                get(S6);
                get(S4);
                get(S3);
                /* Critical Section:
                Use S3, S6, S4 */
                release(S6);
                release(S4);
                release(S3);
        }
}
```



After changing the order of requests of P1, P2 and P3, we can prevent the possibility of any deadlock. For example, according to the resource allocation graph above, there is not cycle, so deadlock has been prevented.

Self-evaluation Please answer the following questions:

1. [3 points] Is your solution to the dining philosophers' problem deadlock free? Justify.

YES. In order to avoid the deadlock, I choose an algorithm as follows:

Only if the left and right philosophers of one philosopher do not eat, the philosopher can enter eating mode. The idea bases on:

1. The left and right do not eat, means that the left fork and right fork are available.
2. Using Java lock and condition to solve the issue.

2. [1 points] Were you able to complete this assignment? What grade are you expecting? Justify.

Yes, I did this assignment by myself on the time, and I feel my solution has reached the expectation from the teacher.

3. [2 points] Describe 2-3 challenges you faced while completing this assignment. How did you tackle those challenges?

Firstly, I have to learn how to draw a resource allocation graph and how to identify if there is a deadlock according to one resource allocation graph, I can answer question 7 in this assignment.

Secondly, I need to investigate the implement and algorithm about the dining piliosophers' problem and try to implement my idea with Java. There are a couple of new API I need to learn about how to avoid deadlock in Java.

4. [2 points] Provide a break down for the activities/milestones for this assignment. Give an

Date	Activities	Hours	Outcome
4/11	Revie the lecture handouts	3	Understanding concurrency, synchronization and others
5/11	0.1 Practice Exercises 1-3	2	Solutions
6/11	0.1 Practice Exercises 4	2	Solutions
7/11	0.1 Practice Exercises 5-7	3	Solutions
8/11	0.2 Programming Exercise: Java lock support, focus on DiningServerImpl	2	Source code
9/11	0.2 Programming Exercise: other classes	2	Source code

Lingda Cai A00372181