Homework 08

CS307-Operating System (D), Chentao Wu, Spring 2020.

Name: 方泓杰(Hongjie Fang) Student ID: 518030910150 Email: galaxies@sjtu.edu.cn

- (8.3) Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix `Need`?

b. Is the system in a safe state?

c. If a request from thread T_1 arrives for $(0, 4, 2, 0)$, can the request be granted immediately?

Solution. Here are the answers to the sub-questions.

a. According to the definition of `Need`, we have the following formula.

$$\texttt{Need} = \texttt{Max} - \texttt{Allocation}$$

Therefore, the `Need` matrix for threads T_0, T_1, T_2, T_3 and T_4 are $(0, 0, 0, 0)$, $(0, 7, 5, 0)$, $(1, 0, 0, 2)$, $(0, 0, 2, 0)$ and $(0, 6, 4, 2)$ respectively.

b. The system is <u>in a safe state</u> according to the banker's algorithm. Either T_0 or T_3 can run with the help of `Available` resources. Once T_3 finishes running, it will release its allocation resources, which allow every other thread to run. The order $(T_0, T_3, T_1, T_2, T_4)$ is a feasible order to arrange the threads to run according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_0	$(1, 5, 2, 0)$	$(0, 0, 0, 0)$	$(0, 0, 1, 2)$	$(1, 5, 3, 2)$
T_3	$(1, 5, 3, 2)$	$(0, 0, 2, 0)$	$(0, 6, 3, 2)$	$(1, 11, 6, 4)$
T_1	$(1, 11, 6, 4)$	$(0, 7, 5, 0)$	$(1, 0, 0, 0)$	$(2, 11, 6, 4)$
T_2	$(2, 11, 6, 4)$	$(1, 0, 0, 2)$	$(1, 3, 5, 4)$	$(3, 14, 11, 8)$
T_4	$(3, 14, 11, 8)$	$(0, 6, 4, 2)$	$(0, 0, 1, 4)$	$(3, 14, 12, 12)$

Therefore, the system is in a safe state.

c. The request <u>can be granted immediately</u>. If we grant the request immediately, then:

- The new `Available` vector is $(1, 1, 0, 0)$;
- The new `Allocation` vector for thread T_1 is $(1, 4, 2, 0)$;
- The new `Need` vector for thread T_1 is $(0, 3, 3, 0)$.

The order $(T_0, T_2, T_1, T_3, T_4)$ is a feasible order to arrange the threads to run according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_0	$(1, 1, 0, 0)$	$(0, 0, 0, 0)$	$(0, 0, 1, 2)$	$(1, 1, 1, 2)$
T_2	$(1, 1, 1, 2)$	$(1, 0, 0, 2)$	$(1, 3, 5, 4)$	$(2, 4, 6, 6)$
T_1	$(2, 4, 6, 6)$	$(0, 3, 3, 0)$	$(1, 4, 2, 0)$	$(3, 8, 8, 6)$
T_3	$(3, 8, 8, 6)$	$(0, 0, 2, 0)$	$(0, 6, 3, 2)$	$(3, 14, 11, 8)$
T_4	$(3, 14, 11, 8)$	$(0, 6, 4, 2)$	$(0, 0, 1, 4)$	$(3, 14, 12, 12)$

Therefore, the request can be granted immediately.

$\square$

- (8.9) Consider the following snapshot of a system: Using the banker's algorithm, determine

	Allocation	***Max***
	A B C D	*A B C D*
T_0	3 0 1 4	5 1 1 7
T_1	2 2 1 0	3 2 1 1
T_2	3 1 2 1	3 3 2 1
T_3	0 5 1 0	4 6 1 2
T_4	4 2 1 2	6 3 2 5

whether or not each of the following state is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. `Available` $= (0, 3, 0, 1)$

b. `Available` $= (1, 0, 0, 2)$

Solution. According to the definition of `Need`, we have the following formula.

$$\texttt{Need} = \texttt{Max} - \texttt{Allocation}$$

Therefore, the `Need` matrix for threads T_0, T_1, T_2, T_3 and T_4 are $(2, 1, 0, 3)$, $(1, 0, 0, 1)$, $(0, 2, 0, 0)$, $(4, 1, 0, 2)$ and $(2, 1, 1, 3)$ respectively.

a. The state is <u>unsafe</u>. After finishing running T_2, T_1 and T_3, the available resources is unable to support either T_0 and T_4 to run because the available instances in resource D is only 2 but both thread T_0 and T_3 need 3. The detailed information is in the table below.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_2	$(0, 3, 0, 1)$	$(0, 2, 0, 0)$	$(3, 1, 2, 1)$	$(3, 4, 2, 2)$
T_1	$(3, 4, 2, 2)$	$(1, 0, 0, 1)$	$(2, 2, 1, 0)$	$(5, 6, 3, 2)$
T_3	$(5, 6, 3, 2)$	$(4, 1, 0, 2)$	$(0, 5, 1, 0)$	$(5, 11, 4, 2)$
Deadlock	$(5, 11, 4, 2)$	$\cdots$	$\cdots$	$\cdots$

Therefore, the state is unsafe.

b. The state is <u>safe</u>. The order $(T_1, T_2, T_0, T_3, T_4)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_1	$(1, 0, 0, 2)$	$(1, 0, 0, 1)$	$(2, 2, 1, 0)$	$(3, 2, 1, 2)$
T_2	$(3, 2, 1, 2)$	$(0, 2, 0, 0)$	$(3, 1, 2, 1)$	$(6, 3, 3, 3)$
T_0	$(6, 3, 3, 3)$	$(2, 1, 0, 3)$	$(3, 0, 1, 4)$	$(9, 3, 4, 7)$
T_3	$(9, 3, 4, 7)$	$(4, 1, 0, 2)$	$(0, 5, 1, 0)$	$(9, 8, 5, 7)$
T_4	$(9, 8, 5, 7)$	$(2, 1, 1, 3)$	$(4, 2, 1, 2)$	$(13, 10, 6, 9)$

Therefore, the state is safe.

$\square$

- (8.18) Which of the six resource-allocation graphs shown in Fig. 1 illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution.

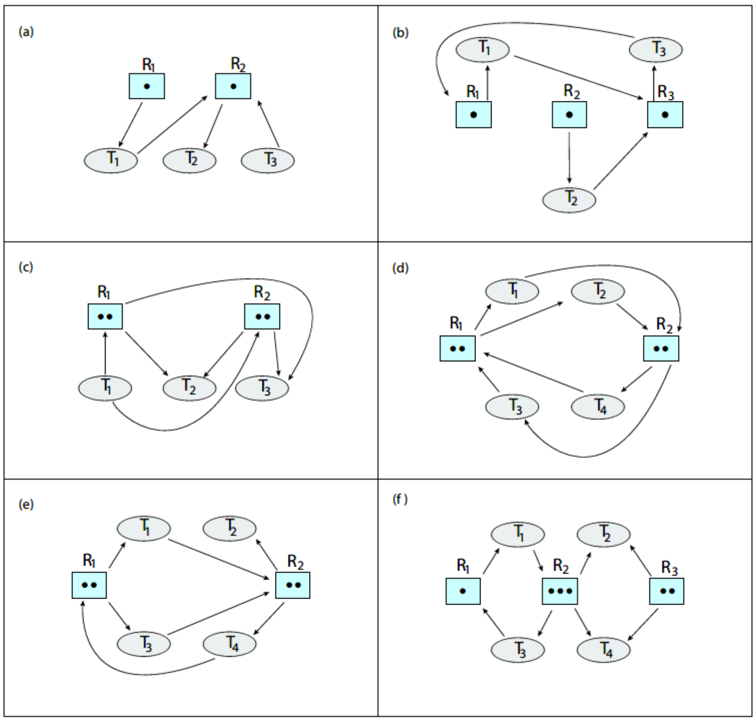

Figure 1: Resource-allocation graphs

Solution. Here are the answers to the sub-questions.

a. No deadlock. A feasible order of execution is (T_2, T_3, T_1).

b. Deadlock. The cycle of threads and resources is $R_1 \to T_1 \to R_3 \to T_3 \to R_1$, which forms a deadlock.

c. No deadlock. A feasible order of execution is (T_2, T_3, T_1).

d. Deadlock. The cycles of threads and resources are $R_1 \to T_1 \to R_2 \to T_3 \to R_1$ and $R_1 \to T_2 \to R_2 \to T_4 \to R_1$, which form a deadlock.

e. No deadlock. A feasible order of execution is (T_2, T_1, T_3, T_4).

f. No deadlock. A feasible order of execution is (T_2, T_4, T_1, T_3).

□

- (8.27) Consider the following snapshot of a system:

	Allocation	***Max***
	A B C D	*A B C D*
T_0	1 2 0 2	4 3 1 6
T_1	0 1 1 2	2 4 2 4
T_2	1 2 4 0	3 6 5 1
T_3	1 2 0 1	2 6 2 3
T_4	1 0 0 1	3 1 1 2

Using the banker's algorithm, determine whether or not each of the following state is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. `Available` $= (2, 2, 2, 3)$

b. `Available` $= (4, 4, 1, 1)$

c. `Available` $= (3, 0, 1, 4)$

d. `Available` $= (1, 5, 2, 2)$

Solution. According to the definition of `Need`, we have the following formula.

$$\texttt{Need} = \texttt{Max} - \texttt{Allocation}$$

Therefore, the `Need` matrix for threads T_0, T_1, T_2, T_3 and T_4 are $(3, 1, 1, 4)$, $(2, 3, 1, 2)$, $(2, 4, 1, 1)$, $(1, 4, 2, 2)$ and $(2, 1, 1, 1)$ respectively.

a. The state is safe. The order $(T_4, T_0, T_1, T_2, T_3)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_4	$(2, 2, 2, 3)$	$(2, 1, 1, 1)$	$(1, 0, 0, 1)$	$(3, 2, 2, 4)$
T_0	$(3, 2, 2, 4)$	$(3, 1, 1, 4)$	$(1, 2, 0, 2)$	$(4, 4, 2, 6)$
T_1	$(4, 4, 2, 6)$	$(2, 3, 1, 2)$	$(0, 1, 1, 2)$	$(4, 5, 3, 8)$
T_2	$(4, 5, 3, 8)$	$(2, 4, 1, 1)$	$(1, 2, 4, 0)$	$(5, 7, 7, 8)$
T_3	$(5, 7, 7, 8)$	$(1, 4, 2, 2)$	$(1, 2, 0, 1)$	$(6, 9, 7, 9)$

Therefore, the state is safe.

b. The state is safe. The order $(T_4, T_2, T_1, T_0, T_3)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_4	$(4, 4, 1, 1)$	$(2, 1, 1, 1)$	$(1, 0, 0, 1)$	$(5, 4, 1, 2)$
T_2	$(5, 4, 1, 2)$	$(2, 4, 1, 1)$	$(1, 2, 4, 0)$	$(6, 6, 5, 2)$
T_1	$(6, 6, 5, 2)$	$(2, 3, 1, 2)$	$(0, 1, 1, 2)$	$(6, 7, 6, 4)$
T_0	$(6, 7, 6, 4)$	$(3, 1, 1, 4)$	$(1, 2, 0, 2)$	$(7, 9, 6, 6)$
T_3	$(7, 9, 6, 6)$	$(1, 4, 2, 2)$	$(1, 2, 0, 1)$	$(8, 11, 6, 7)$

Therefore, the state is safe.

c. The state is unsafe. We cannot finish any thread only using the current available resources, since there is no instance of resource B left and each thread needs at least one instance of resource B to run. Therefore, the state is unsafe.

d. The state is safe. The order $(T_3, T_4, T_2, T_1, T_0)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_3	$(1, 5, 2, 2)$	$(1, 4, 2, 2)$	$(1, 2, 0, 1)$	$(2, 7, 2, 3)$
T_4	$(2, 7, 2, 3)$	$(2, 1, 1, 1)$	$(1, 0, 0, 1)$	$(3, 7, 2, 4)$
T_2	$(3, 7, 2, 4)$	$(2, 4, 1, 1)$	$(1, 2, 4, 0)$	$(4, 9, 6, 4)$
T_1	$(4, 9, 6, 4)$	$(2, 3, 1, 2)$	$(0, 1, 1, 2)$	$(4, 10, 7, 6)$
T_0	$(4, 10, 7, 6)$	$(3, 1, 1, 4)$	$(1, 2, 0, 2)$	$(5, 12, 7, 8)$

$\square$

- (8.28) Consider the following snapshot of a system:

	Allocation	***Max***	***Available***
	A B C D	*A B C D*	*A B C D*
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Answer the following questions using the banker's algorithm:

1. Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.
2. If a request from thread T_4 arrives for $(2, 2, 2, 4)$, can the request be granted immediately?
3. If a request from thread T_2 arrives for $(0, 1, 1, 0)$, can the request be granted immediately?
4. If a request from thread T_3 arrives for $(2, 2, 1, 2)$, can the request be granted immediately?

Solution. According to the definition of `Need`, we have the following formula.

$$\texttt{Need} = \texttt{Max} - \texttt{Allocation}$$

Therefore, the `Need` matrix for threads T_0, T_1, T_2, T_3 and T_4 are $(3, 3, 3, 2)$, $(2, 1, 3, 0)$, $(0, 1, 2, 0)$, $(2, 2, 2, 2)$ and $(3, 4, 5, 4)$ respectively.

a. The system is in a safe state. The order $(T_2, T_0, T_1, T_3, T_4)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_2	$(2, 2, 2, 4)$	$(0, 1, 2, 0)$	$(2, 4, 1, 3)$	$(4, 6, 3, 7)$
T_0	$(4, 6, 3, 7)$	$(3, 3, 3, 2)$	$(3, 1, 4, 1)$	$(7, 7, 7, 8)$
T_1	$(7, 7, 7, 8)$	$(2, 1, 3, 0)$	$(2, 1, 0, 2)$	$(9, 8, 7, 10)$
T_3	$(9, 8, 7, 10)$	$(2, 2, 2, 2)$	$(4, 1, 1, 0)$	$(13, 9, 8, 10)$
T_4	$(13, 9, 8, 10)$	$(3, 4, 5, 4)$	$(2, 2, 2, 1)$	$(15, 11, 10, 11)$

Therefore, the system is in a safe state.

b. The request can not be granted immediately. If we grant the request, the `Available` vector will become $(0, 0, 0, 0)$, which means there is no available resources. What's more, thread T_4 still can not finish running after receiving the resources. So there is no thread that can be executed currently, which means the system is in an unsafe state. Therefore, The request can not be granted immediately.

c. The request can be granted immediately. If we grant the request immediately, then:
   - The new `Available` vector is $(2, 1, 1, 4)$;
   - The new `Allocation` vector for thread T_2 is $(2, 5, 2, 3)$;
   - The new `Need` vector for thread T_2 is $(0, 0, 1, 0)$.

   The order $(T_2, T_0, T_1, T_3, T_4)$ is still a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_2	$(2, 1, 1, 4)$	$(0, 0, 1, 0)$	$(2, 5, 2, 3)$	$(4, 6, 3, 7)$
T_0	$(4, 6, 3, 7)$	$(3, 3, 3, 2)$	$(3, 1, 4, 1)$	$(7, 7, 7, 8)$
T_1	$(7, 7, 7, 8)$	$(2, 1, 3, 0)$	$(2, 1, 0, 2)$	$(9, 8, 7, 10)$
T_3	$(9, 8, 7, 10)$	$(2, 2, 2, 2)$	$(4, 1, 1, 0)$	$(13, 9, 8, 10)$
T_4	$(13, 9, 8, 10)$	$(3, 4, 5, 4)$	$(2, 2, 2, 1)$	$(15, 11, 10, 11)$

Therefore, the request can be granted immediately.

d. The request can be granted immediately. If we grant the request immediately, then:

- The new `Available` vector is $(0, 0, 1, 2)$;
- The new `Allocation` vector for thread T_2 is $(6, 3, 2, 2)$;
- The new `Need` vector for thread T_2 is $(0, 0, 1, 0)$.

The order $(T_3, T_0, T_1, T_2, T_4)$ is a feasible order according to the following analysis.

Thread Order	`Available` (before)	`Need`	`Allocation`	`Available` (after)
T_3	$(0, 0, 1, 2)$	$(0, 0, 1, 0)$	$(6, 3, 2, 2)$	$(6, 3, 3, 4)$
T_0	$(6, 3, 3, 4)$	$(3, 3, 3, 2)$	$(3, 1, 4, 1)$	$(9, 4, 7, 5)$
T_1	$(9, 4, 7, 5)$	$(2, 1, 3, 0)$	$(2, 1, 0, 2)$	$(11, 5, 7, 7)$
T_2	$(11, 5, 7, 7)$	$(0, 1, 2, 0)$	$(2, 4, 1, 3)$	$(13, 9, 8, 10)$
T_4	$(13, 9, 8, 10)$	$(3, 4, 5, 4)$	$(2, 2, 2, 1)$	$(15, 11, 10, 11)$

Therefore, the request can be granted immediately.

$\square$