110590450 歐佳昀 hw03

Chap 7:

7.8: The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

ANS:

Because acquiring a semaphore may put the process to sleep while it is waiting for the semaphore to become available.

Chap 8:

8.20: In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, and new resources are bought and added to the system.

If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

(a) Increase Available (new resources added).
(b) Decrease Available (resource permanently removed from system).
(c) Increase Max for one process (the process needs or wants more resources than allowed).
(d) Decrease Max for one process (the process decides that it does not need that many resources).
(e) Increase the number of processes.
(f) Decrease the number of processes.

ANS:

a. Increase Available (new resources added) –
   Safe, can changed without problems.
b. Decrease Available (resource permanently removed from system) –
   This could have effect on system, introduce the possibility of deadlock as the safety of the system assumed there were a certain number of available resources.
c. Increase Max for one process (the process needs more resources than allowed, it may want more) –

May have an effect on system, the possibility of deadlock happened.

d. Decrease Max for one process(the process decides it does not need that many resources) –
   Safely to change without any problems

e. Increase the number of processes –
   Can allowed assuming that resources were allocated to the new process such that the system does not enter an unsafe state.

f. Decrease the number of processes –
   Can safe to change without problems.

8.27: Consider the following snapshot of a system :

	Allocation A B C D	Max A B C D
P0	1 2 0 2	4 3 1 6
P1	0 1 1 2	2 4 2 4
P2	1 2 4 0	3 6 5 1
P3	1 2 0 1	2 6 2 3
P4	1 0 0 1	3 1 1 2

–

- Use the *banker's algorithm*, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.

(a) Available=(2,2,2,3) => ANS : P4,P0,P1, P2, P3

(b) Available=(4,4,1,1) => ANS : P2,P4,P1, P3, P0

(a)

	Allocation	Max	Avaliable	Need
P0	1202	4316	2223	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P4 -> Need(2111) <= Available(2223)，P4 release

Available(2223) + Allocation(1001) = Available(3224)

	Allocation	Max	Avaliable	Need
P0	1202	4316	3224	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P0 -> Need(2111) <= Available(3224)，P0 release

Available(3224) + Allocation(1202) = Available(4426)

	Allocation	Max	Avaliable	Need
P0	1202	4316	4426	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P1 -> Need(2312) <= Available(4426)，P1 release

Available(4426) + Allocation(0112) = Available(4538)

	Allocation	Max	Avaliable	Need
P0	1202	4316	4538	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P2 -> Need(2411) <= Available(4538)，P2 release

Available(4538) + Allocation(1240) = Available(5778)

	Allocation	Max	Avaliable	Need
P0	1202	4316	5778	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P3 -> Need(1422) <= Available(4538)，P3 release

Available(5778) + Allocation(1201) = Available(6979)

< P4,P0,P1, P2, P3>

- (b)

	Allocation	Max	Avaliable	Need
P0	1202	4316	4411	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P2 -> Need(2411) <= Available(4411)，P2 release

Available(4411) + Allocation(1240) = Available(5651)

	Allocation	Max	Avaliable	Need
P0	1202	4316	5651	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P4 -> Need(2111) <= Available(5651)，P4 release

Available(5651) + Allocation(1001) = Available(6652)

	Allocation	Max	Avaliable	Need
P0	1202	4316	6652	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P1 -> Need(3114) <= Available(6652)，P1 release

Available(6652) + Allocation(0112) = Available(6764)

	Allocation	Max	Avaliable	Need
P0	1202	4316	6764	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P3 -> Need(1422) <= Available(6764)，P3 release

Available(6764) + Allocation(1201) = Available(7965)

	Allocation	Max	Avaliable	Need
P0	1202	4316	7965	3114
P1	0112	2424		2312
P2	1240	3651		2411
P3	1201	2623		1422
P4	1001	3112		2111

P0 -> Need(3114) <= Available(7965)，P0 release

Available(7965)) + Allocation(1202) = Available(8 11 6 7)

<P2,P4,P1, P3, P0>

8.30: A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighbor town.
The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.)
Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock.

Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).

ANS:

```
semaphore bridge = 1;

void enterbridge() {
        bridge.wait();
}
void exitbridge() {
        bridge.signal();
}
```

Chap. 9:

9.15: Compare the memory organization schemes of contiguous memory allocation and paging with respect to the following issues:

- (a) external fragmentation
- (b) internal fragmentation
- (c) ability to share code across processes

ANS:

Contiguous memory allocation scheme:

Affected by external fragmentation, as address spaces are allocated contiguously, resulting in holes when old processes terminate and new processes start.

Does not allow processes to share code, as a process's virtual memory segment is not divided into non-contiguous fine-grained segments.

Pure segmentation:

Affected by external fragmentation, as a process's segment is laid out contiguously in physical memory, leading to fragmentation when segments of terminated processes are replaced by segments of new processes.

Segmentation allows processes to share code: two different processes can share a code segment but have distinct data segments.

Pure paging:

Not affected by external fragmentation, but affected by internal fragmentation: processes are allocated in page granularity, and if a page is not fully utilized, it results in internal fragmentation and corresponding space wastage.

Paging also enables processes to share code at the granularity of pages.

9.24: Consider a computer system with a 32-bit logical address and 8-KB page size. The system supports up to 1 GB of physical memory. How many entries are there in each of the following?

(a) A conventional, single-level page table

(b) An inverted page table

ANS :

Given computer system has 32-bit logical address

Logical Address Space size = 2^32 bytes

Physical Memory size = 1GB = 2^30 bytes

Page size = 8 KB = 2^3 * 2^10 bytes = 2^13 bytes

(a)Number of entries in single-level page table = Number of pages in the Logical Memory

= 2^32 bytes / 2^13 bytes = 2^19 entries

(b)

Number of entries in inverted page table = number of pages/frames in the Physical memory

= 2^30 / 2^13 = 2^17 entries