CS 4/53201 # Homework #3 **Operating Systems**

Due in class on Monday 9 November 1998

---

1. (Exercise 7.6 from OSC) 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, new resources are brought 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 possibility of deadlock), and under what circumstances?

a. Increase Available (new resources added)

If the system was in a safe state (as it should be), adding additional resources will only make the banker's job easier; it can not put the system into an unsafe state.

b. Decrease Available (resource permanently removed from system)

Removing resources may put the system into an unsafe state, leaving it over-committed and in an unsafe state with respect to the new number of available resources. This removal can only be done when the resulting state is approved by the safety algorithm.

c. Increase Max for one process (the process needs more resources than allowed, it may want more)

If the new Max is greater than the number of available resources, this process may not run to completion, which can lead to deadlock. However, even if the new Max is less than the number of available resources, increasing Max could still put the system into an unsafe state, leaving it over-committed and in an unsafe state with respect to the new Max. As in (b), this change can only be done when the resulting state is approved by the safety algorithm.

d. Decrease Max for one process (the process decides it does not need that many resources)

If the system was in a safe state (as it should be), decreasing Max for one process will only make the banker's job easier; it can not put the system into an unsafe state.

e. Increase the number of processes

Increasing the number of processes puts more demand on the system, and may put the system into an unsafe state, leaving it over-committed and in an unsafe state with respect to the new number of processes. This increase can only be done when the resulting state is approved by the safety algorithm.

f. Decrease the number of processes

If the system was in a safe state (as it should be), decreasing the number of processes will only make the banker's job easier; it can not put the system into an unsafe state.

2. (7.13 from OSC (7.11 in the 4th edition), modified as follows) Consider the following snapshot of a system:

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	1	6	5	2				
P4	0	0	1	4	1	6	5	6				

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix Need?

	Need			
	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	1	0	2	0
P4	1	6	4	2

b. Is the system in a safe state?

	After P0 runs (Need stays same)				P1 can't run After P2 runs				After P1 runs				After P3 runs			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	7	5	0	0	7	5	0	0	0	0	0	0	0	0	0
P2	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
P3	1	0	2	0	1	0	2	0	1	0	2	0	0	0	0	0
P4	1	6	4	2	1	6	4	2	1	6	4	2	1	6	4	2
Avail	1	5	3	2	2	8	8	6	3	8	8	6	3	14	11	8

Finally, after P4 runs, avail ends up at 3 14 12 12

c. If a request from process P4 arrives for (1, 4, 1, 0) can the request be granted immediately?

	If request is granted (new Need)				After P0 runs			
	A	B	C	D	A	B	C	D
P0	0	0	0	0	0	0	0	0
P1	0	7	5	0	0	7	5	0
P2	1	0	0	2	1	0	0	2
P3	1	0	2	0	1	0	2	0
P4	0	2	3	2	0	2	3	2
Avail	0	1	1	0	0	1	2	2

After the request is granted, process P0 is the only process with a Need <= Avail, so it's the only process that can run. But—after it runs, there aren't enough resources for anyone else to run. As a result, the request from process P4 should not be granted.

3. (Exercise 8.4 from OSC (8.3 in the 4th edition)) When a process is rolled out of memory, it loses its ability to use the CPU (at least for a while). Describe another situation where a process loses its ability to use the CPU, but where the process does not get rolled out.

If a process times out, or gets blocked while waiting on an I/O operation, it usually is moved off the CPU. However, it may not necessarily get swapped out of memory in that case.

4. (Exercise 8.7 from OSC) Why are page sizes always powers of 2?

This makes address translation much easier, as a certain number of bits in the virtual address can directly represent the page number and the offset within the page. If N bits are used to represent the offset within the page, then the page size is always 2^N.

5. (Exercise 8.11 from OSC) What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What would the effect of updating some byte in the one page be on the other page?

This would mean that the two processes are sharing the same code or data (whatever is in that one page). Thus one process could transfer data to the other by placing it within the "shared" page, and the other could simply access that data directly (instead of having to copy it from some message buffer into its own address space).

However, there is also a potential problem with consistency between the two pages. If one process modifies the page, there must be some method to make sure that the change is reflected in the other process. For example, if the other process doesn't know that page is in memory, it must not load a fresh copy from the disk, overwriting the change made by the first process.