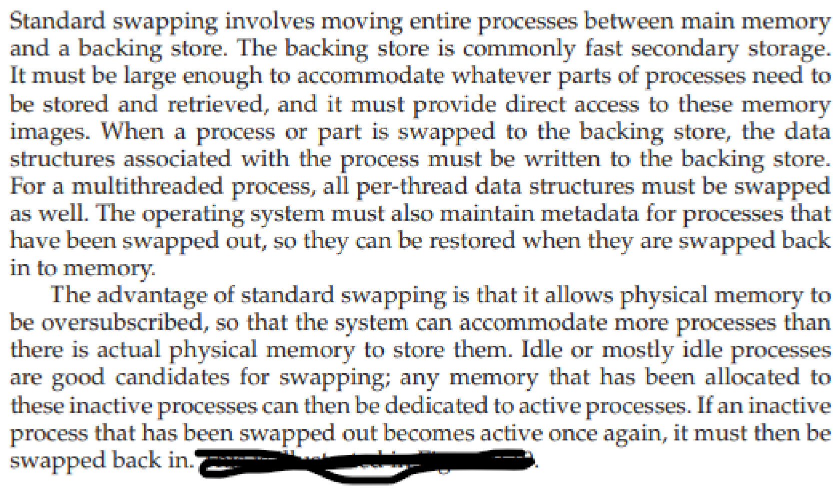
Time: 30 Minutes | CS-2005 Operating Systems | Marks: 20
Roll Number: | Quiz-3 | section:

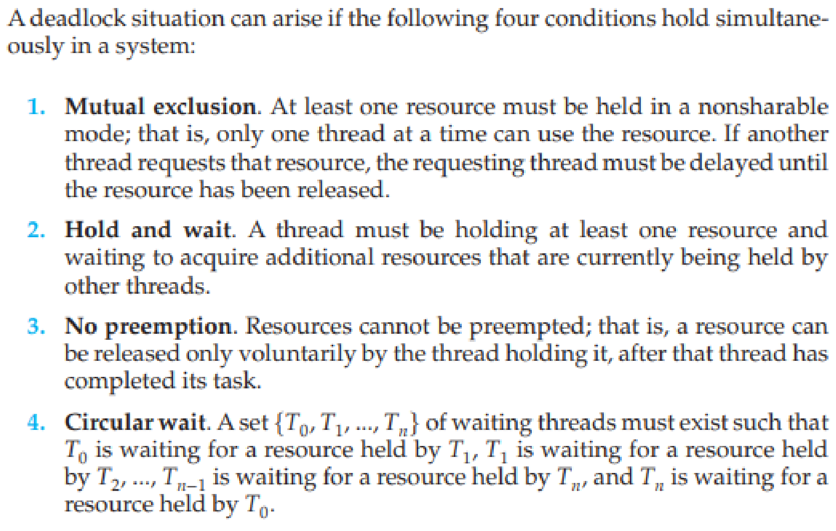
1. Describe standard swapping and its advantages.



Standard swapping involves moving entire processes between main memory and a backing store. The backing store is commonly fast secondary storage. It must be large enough to accommodate whatever parts of processes need to be stored and retrieved, and it must provide direct access to these memory images. When a process or part is swapped to the backing store, the data structures associated with the process must be written to the backing store. For a multithreaded process, all per-thread data structures must be swapped as well. The operating system must also maintain metadata for processes that have been swapped out, so they can be restored when they are swapped back in to memory.

The advantage of standard swapping is that it allows physical memory to be oversubscribed, so that the system can accommodate more processes than there is actual physical memory to store them. Idle or mostly idle processes are good candidates for swapping; any memory that has been allocated to these inactive processes can then be dedicated to active processes. If an inactive process that has been swapped out becomes active once again, it must then be swapped back in. [redacted].

2. Describe necessary conditions which can rise deadlock situation in system.

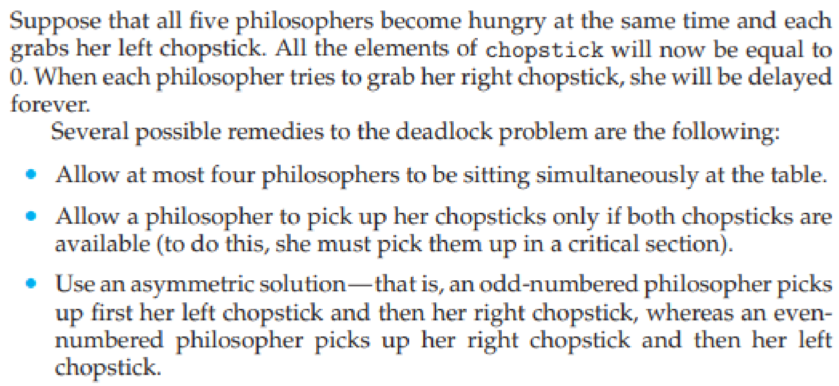


A deadlock situation can arise if the following four conditions hold simultaneously in a system:

1. **Mutual exclusion**. At least one resource must be held in a nonsharable mode; that is, only one thread at a time can use the resource. If another thread requests that resource, the requesting thread must be delayed until the resource has been released.
2. **Hold and wait**. A thread must be holding at least one resource and waiting to acquire additional resources that are currently being held by other threads.
3. **No preemption**. Resources cannot be preempted; that is, a resource can be released only voluntarily by the thread holding it, after that thread has completed its task.
4. **Circular wait**. A set $\{T_0, T_1, ..., T_n\}$ of waiting threads must exist such that T_0 is waiting for a resource held by T_1, T_1 is waiting for a resource held by T_2, ..., T_{n-1} is waiting for a resource held by T_n, and T_n is waiting for a resource held by T_0.

3. You are aware of dinning philosopher's problem. Suppose we have
semaphore chopstick [5];
and all five philosophers become hungry at the same time and each grabs her left chopstick.
   a. What is the value of chopstick now?
   b. What problem is going to take place when all philosophers would go to grab her right chopstick.
   c. If any problem, take place then give all possible remedies to avoid the problem.

Time: 30 Minutes CS-2005 Operating Systems Marks: 20
Roll Number: Quiz-3 section:


Suppose that all five philosophers become hungry at the same time and each grabs her left chopstick. All the elements of `chopstick` will now be equal to 0. When each philosopher tries to grab her right chopstick, she will be delayed forever.

Several possible remedies to the deadlock problem are the following:

- Allow at most four philosophers to be sitting simultaneously at the table.
- Allow a philosopher to pick up her chopsticks only if both chopsticks are available (to do this, she must pick them up in a critical section).
- Use an asymmetric solution—that is, an odd-numbered philosopher picks up first her left chopstick and then her right chopstick, whereas an even-numbered philosopher picks up her right chopstick and then her left chopstick.

Separate Quiz Paper:

1. Write a code for the following problem.
   A student majoring in anthropology and minoring in computer science has embarked on a research project to see if African chimpanzees(chimps) can be taught about deadlocks. He locates a deep canyon and fastens a rope across it, so the chimpanzees can cross hand-over-hand. Several chimps can cross at the same time, provided that they are all going in the same direction. If eastward moving and westward moving chimps ever get onto the rope at the same time, a deadlock will result (the chimps will get stuck in the middle) because it is impossible for one chimps to climb over another one while suspended over the canyon. If a chimps wants to cross the canyon, it must check to see that no other chimp is currently crossing in the opposite direction. Write a program using semaphores that avoids deadlock. Do not worry about a series of eastward moving chimps holding up the westward moving chimps indefinitely.

Solution:

Time: 30 Minutes CS-2005 Operating Systems Marks: 20
Roll Number: Quiz-3 section:

```
enum direction { WEST=0, EAST=1 };
Semaphore mutex; // initially 1, protect critical
sections
Semaphore blocked[2];
// one for each direction. Each initially 0, so always
sleep on first P
int blockedCnt[2]; // number of chimps waiting on
each side
int travelers[2]; // number of chimps on the rope
heading each direction
//(at least one is zero at any time)
Chimp(direction dir)
{
int revdir = !dir; // the reverse direction
mutex->P();
while (travellers[revdir])
{
blockedCnt[dir]++; // announce our intention to
block
mutex->V(); // trade mutex for block
blocked[dir]->P();
mutex->P();
}
travellers[dir]++; // we’re free to cross
mutex->V(); // cross bridge.
mutex->P();
travellers[dir]--;
if (!travellers[dir]) { // if we’re the last one heading
this way,
// wakeup chimps waiting for us to finish.
while(blockedCnt[revdir]--)
blocked[revdir]->V();
}
mutex->V(); }
```