1.What does a modern operating system do?

答：

Provides abstractions

Provides standard interface

Mediates(调解) resources usage

Consumes resources

2.What is a context switch? How is context switch implemented in OS?

答：

（1）The stopping of one process and starting (or restarting) of another process is called a context switch

（2）save and restore hardware state on a context switch. Save the state in Process Control Block

1. 3. Suppose we have the following simple bank account deposit and withdraw program, rewrite it so that it is safe for multiple thread concurrent execution, and that withdraw would wait until sufficient fund becomes available in the account:

unsignedint account = 0;

void deposit (unsigned int amount){

l->Acquire();

account += amount;

l->Release();

}

void withdraw (unsigned int amount){

l->Acquire();

while(account <= amount)

c->Wait(l);

account -= amount;

l->Release();

}

4.What are the four necessary conditions for dead lock?

Mutual Exclusion(互斥)

Hold and wait

No preemption(无抢占)

Circular wait(循环等待)

5.What are the three running states of a process? When a running state does an I/O operation, what is the new state of the process?

答：

(1) Running, ready, waiting

(2)Waiting

6. Suppose we have the following four processes to schedule:

What are the scheduling of both preemptive and non-preemptive Shortest Job First scheduling? And what are their average waiting time respectively?

Process Arriving Time CPU Burst

P1 0 8

P2 1 6

P3 2 4

P4 3 6

Both preemptive: AWT = 29

Non preemptive: AWT = 8+

1. What is the driven factor of operating system development?

* Much of operating system history driven by relative cost factors of hardware and people. Hardware started out fantastically expensive relative to people and the relative cost has been decreasing ever since. Relative costs drive the goals of the operating system.
* In the beginning: Expensive Hardware, Cheap People Goal: maximize hardware utilization.
* Now: Cheap Hardware, Expensive People Goal: make it easy for people to use computer.

1. Name three abstractions that are provided in process management by modern operating systems?

Processes, Unbounded Memory, Files, Synchronization and Communication Mechanisms.

1. What is multiprogramming?

Multiple processes at a time. Typical of Unix plus all currently envisioned new operating systems. Allows system to separate out activities cleanly.

1. What is the key difference between a process and a thread?

A thread is again an execution stream in the context of a thread state.

Key difference between processes and threads is that multiple threads share parts of their state. Typically, allow multiple threads to read and write same memory. (Recall that no processes could directly access memory of another process). But, each thread still has its own registers. Also has its own stack, but other threads can read and write the stack memory.

1. How do threads execute conceptually

Conceptually, threads execute concurrently. This is the best way to reason about the behavior of threads. But in practice, the OS only has a finite number of processors, and it can't run all of the runnable threads at once. So, must multiplex the runnable threads on the finite number of processors.

1. Why may an execution of a multi-thread program produce incorrect results?

* When execute concurrently, the result depends on how the instructions interleave.
* The results are nondeterministic - you may get different results when you run the program more than once.
* So, it can be very difficult to reproduce bugs.
* Nondeterministic execution is one of the things that make writing parallel programs much more difficult than writing serial programs.

Chances are, the programmer is not happy with all of the possible results listed above. Probably wanted the value of a to be 2 after both threads finish. To achieve this, must make the increment operation atomic. That is, must prevent the interleaving of the instructions in a way that would interfere with the additions.

1. What is an atomic operation?

* An atomic operation is one that executes without any interference from other operations. In other words, it executes as one unit.
* Typically build complex atomic operations up out of sequences of primitive operations. In our case the primitive operations are the individual machine instructions.
* More formally, if several atomic operations execute, the final result is guaranteed to be the same as if the operations executed in some serial order.

1. What is a semaphore?

* Semaphore is our first synchronization abstraction, which is, conceptually, a counter that supports two atomic operations, P and V
* P atomically waits until the counter is greater than 0, then decrements the counter and returns.
* V atomically increments the counter.

1. Suppose a student implements semaphore class using lock and condition variables as follows:

Class Semaphore{

Int count;

Lock\* l;

Condition\_Variable\* c;

Public:

Semaphore(int n){

Count = n;

L = new Lock;;

C = new Condition\_Variable;

}

Void P(){

L->Acquire();

while(count == 0) c->Wait();

Count--;

L->Release();

}

Void V()

{

L->Acquire();

Count++;

C->Signal();

L->Release();

}}