**Homework01 Questions:**

1. What are the sections in memory that are typical for most processes?

Stack, Heap, Data, Text

1. Assume some process has started and has since finished executing. During the process' execution it first encountered an I/O event, and then later was interrupted before it was finished. What is the sequence of Process States for this process?

**New🡪Ready🡪Running🡪Waiting🡪Ready🡪Running🡪Ready🡪Rinning🡪Terminated**

1. Detail the pieces of information contained in a PCB.

Process state, Program counter, CPU registers, CPU scheduling information, Memory-management information, Accounting information, I/O status information

1. Describe Context Switching, and its greatest drawback.

When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch

Context of a process represented in the PCB

Context-switch time is pure overhead; the system does no useful work while switching

The more complex the OS and the PCB **🡪** the longer the context switch

Time dependent on hardware support

Some hardware provides multiple sets of registers per CPU **🡪** multiple contexts loaded at once

1. What are the differences between shared memory and memory passing related to IPC?

Message passing is a time consuming process because it is implemented through kernel (system calls). In shared memory make sure that the processes are not writing to the same location simultaneously

1. Define a thread and what composes it.

A thread is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers, and a thread ID.

1. What are the four major benefits of multithreaded programming? Describe each.

**Responsiveness –** may allow continued execution if part of process is blocked, especially important for user interfaces

**Resource Sharing –** threads share resources of process, easier than shared memory or message passing

**Economy –** cheaper than process creation, thread switching lower overhead than context switching

**Scalability –** process can take advantage of multicore architectures

1. Describe and detail some of the challenges of writing multithreaded code.

Dividing activities

Balance

Data splitting

Data dependency

Testing and debugging

Parallelism implies a system can perform more than one task simultaneously

Concurrency supports more than one task making progress

Single processor / core, scheduler providing concurrency

1. There are 3 major multithreaded models, detail each of their benefits and drawbacks.

Many-to-One:

* Many user-level threads mapped to single kernel thread
* One thread blocking causes all to block
* Multiple threads may not run in parallel on muticore system because only one may be in kernel at a time
* Few systems currently use this model

One-to-One:

* Each user-level thread maps to kernel thread
* Creating a user-level thread creates a kernel thread
* More concurrency than many-to-one
* Number of threads per process sometimes restricted due to overhead

Many-to-Many:

* Allows many user level threads to be mapped to many kernel threads
* Allows the operating system to create a sufficient number of kernel threads
* Windows with the *ThreadFiber* package
* Otherwise not very common

1. Given an application where 52% can be ran in parallel; describe the speed up gained from going from 1 processing core to 5 processing cores.

1: same

2:100/(48+26)=100/74=135%

3:153%

4:164%

5:171%