Assignment Two Ryan Lebeau 104535367

1. **Batch** - Batch systems do not interact with the computer directly, but are used by the operator only. These are done through systems like punch cards, and similar operations are frequently put together in batches. They should not be used in systems that need high interaction between the user and job.
   Interactive - The Interactive operating system allows users to directly use the os while processes are being done, either through the command line or a GUI.
   Time-Slicing - This operating system shares a single computer system that can be accessed by many users in different areas at specific terminals. The main difference between this system is that it focuses mainly on minimizing response time. Although this system has quick response times by switching between jobs frequently, but this causes a lack of reliability.
   Real-Time - These operating systems act within defined time constraints and control the entire system based off of their processing time, called response time. Since all systems are local it has exponentially faster response time than online systems.
   Distributed - This operating system is a wide scale Real-Time system that spreads out the processing over multiple processors and users. These system's communication is considered a loosely coupled system. The distributed nature of this system provides better service to users and less delays in data processing.

2. A program is something in normal written language like Java or C (.java or .c format). Once this language is compiled into useable code it is an executable file (.exe format). When the executable file is ran, the operating system now takes this as a process to complete it.

3. The sleep call will suspend the process within the CPU and allow it to process other processes while waiting. The sleep call will always have a parameter for an amount of time to wait, creating an interval timer that will expire once the time runs out.

4. During a context switch, the kernel changes control form the CPU's current process to another process that is ready to execute. Without context switching, modern multithreading would not be possible (switches occur at a rate of hundreds per second). The faster the switching can happen reliably the better a system can complete multiprocessing.

5. It is possible for a process to wait until a message is received from a mailbox using a pause() loop until the mail arrives.
   mailbox:send(ToID, msg)
   mailbox:receive(&msg)
   Part 2:
   mailbox:send(ToID, msg)
   While:true
       If: mailbox:get(FromID)
           mailbox:receive(FromID,msg)
           Break while

6. **a)** Using the many to many model, if the number of kernel threads were less than the number of processors the performance should be very fast and each thread should be processed as fast as possible.
   b) Again, with the many to many model, if the number of kernel threads were equal to the number of processors the performance will be as fast as possible as well just without the null process that are unused in part a.
   c) With the many to many model, if the number of kernel threads are greater than the number of processors but still less than the number of user-level threads the performance will be much slower than that of the previous parts. Although many different ways to handle this overflow of threads (like multithreading) are available, it still can not be as fast as the other parts (a or b).