# Homework 2

[Process, Thread, Scheduling]

1. When scheduling takes place only under circumstances **when a process switches from running state to the waiting state(for example, I/O Request , or invocation of waif wait for the termination of one of the child processes)** and **when a process terminates**, we say that the scheduling scheme is **non-preemptive** [**不可抢占式**] or **cooperative**; otherwise, it is **preemptive [可抢占式]**. Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
2. A solution to the problem of indefinite blockage of low-priority processes is **Priority Scheduling**, which is a technique of gradually increasing the priority of processes that wait in the system for a long time. For example, if priorities range from 127 (low) to 0 (high), we could increase the priority of a waiting process by 1 every 15 minutes. Eventually, even a process with an initial priority of 127 would have the highest priority in the system and would be executed.   
   [**This will be a good advice which could be used for bank service so as to increase the satisfaction of customers**]
3. Often, in a batch system, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. The **Short-term Scheduler**, or job scheduler, selects processes from this pool and loads them into memory for execution. The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them. It controls the degree of multiprogramming (the number of processes in memory). Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling, called **Multy-Level Queue Scheduling**. The key idea behind it is that sometimes it can be advantageous to remove processes from memory (and from active contention for the CPU) and thus reduce the degree of multiprogramming. Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called **Round Robin Scheduling**. The process is swapped out, and is later swapped in, by the medium-term scheduler.
4. Cooperating processes require an inter-process communication (**IPC**) mechanism that will allow them to exchange data and information. There are two fundamental models of inter-process communication: (1) **Message Parsing** and (2) **Shared Memory**. In the former model, a region of memory that is shared by cooperating processes is established. Processes can then exchange information by reading and writing data to the shared region. In the later model, communication takes place by means of messages exchanged between the cooperating processes.
5. A thread is a basic unit of **Process**; it comprises a thread ID, a program counter, a register set, and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals. A traditional (or heavyweight) process has a single thread of control. If a process has multiple threads of control, it can perform more than one task at a time.
6. According to ①**Von Meumann**, the basis of modern computers, ②**Main Memory** is the place to store the instructions and data of a program when trying to run that program. Once the program has been copied into ②, ③**The CPU** is the component to execute the instructions of the program one after another. This means some resources should be prepared to execute a program besides the program itself, and researchers use the concept of ④**PCB** to describe the entity of program and all other necessary resources needed to run that program. To manage ④ inside OS, ⑤**Data Structure** is proposed, which records the location information, identification information, control information of ④. To improve the efficiency of using CPU when CPU is assigned to a ④, ⑥**Thread** is further proposed to represent execution units in ④. With ⑥, ④ could provide several services by switching CPU among ⑥ of ④ now.   
   ①②③④⑤⑥： thread descriptor, Control unit, thread, FCB, process descriptor, computing machine, CPU, ENIAC, procedure, hard disk, ALU, computer architecture, Turing machine, system partition, TCB, main memory, von Neumann architecture, process, PCB, adapter, file system, controller
7. A **Librarie** provides the programmer with an **API** for creating and managing threads. There are two primary ways of implementing it. The first approach is to provide a library entirely in user space with no kernel support. All code and data structures for the library exist in user space. This means that invoking a function in the library results in a local function call in user space and not a system call. The second approach is to implement a kernel-level library supported directly by the operating system. In this case, code and data structures for the library exist in kernel space. Invoking a function in the API for the library typically results in a system call to the kernel. Three main thread libraries are in use today: (1) POSIX Pthreads, (2) Win32, and (3) Java.
8. In the follows algorithm, the fairness schedule is ⑥**First Job First Serve**, the raising throughput schedule is ⑦ **Shortest Job First**, the balance waiting time and running time schedule is ⑧**Round Robin**.  
   ⑥⑦⑧: first come first serve; shortest job first; round robin; high response rate first; priority first.
9. PCB is a part of the process entity, which one is not belong to the PCB in follows? ( **D** )

A）~~Process ID~~ B）~~State of CPU~~

C）~~Stack pointer~~  D）Global variables

1. To some extent, all other CPU scheduling algorithms could be seen as special instance of **Priority** algorithm.

A. lottery tickets B. Round-Robin C. priority D. privilege

1. Describe the differences among short-term, medium-term, and long-term scheduling?

**The Short-term Scheduling select which should be executed and allocate CPU.**

**The Medium-term Scheduling select which jobs or process should be swapped out.**

**The Long-term Scheduling select which programs or processes should be put inside the ready queue.**

1. Describe the actions taken by a kernel to context-switch between processes?

**The Kernel should record states of Processes when it switch to continue where it left off the computation.**

1. CPU-scheduling decisions may take place under the following four circumstances:
2. When a process switches from the waiting state to the ready state (for example, at completion of I/O)
3. When a process terminates
4. When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait for the termination of one of the child processes)
5. When a process switches from the running state to the ready state (for example, when an interrupt occurs)
6. A multiprogramming system has one CPU and two IO devices – IO1 and IO2. The competition for CPU follows the preemptive priority scheme, which means the job with higher priority could interrupt the running job with lower priority and get the CPU. While, the competition for IO devices is non-preemptive (Why?  ). Now there are three jobs – J1, J2 and J3 – stored in memory, and they arrive at same time. J1 has the highest priority and J3 has the lowest priority. Their ordered requests for CPU and IO devices are listed as follows:

**J1: IO2(30ms), CPU(10ms), IO1(30ms), CPU(10ms).**

**J2: IO1(20ms), CPU(20ms), IO2(40ms).**

**J3: CPU(30ms), IO1(20ms).**Please draw the **Gantt diagram** and answer the following questions:

1. Compute the turnaround time of those three jobs;
2. Compute the CPU utilization rate after all three jobs are finished;
3. Compute the utilization rate of IO1 after all three jobs are finished.

Processes

J1

J2

J3

10 20 30 40 50 60 70 80 90 CPU time(ms)

CPU Time

IO1 Time

IO2 Time

Wait