Midterm Exam, Operating Systems, Spring 2022, Department of CSIE, NTUT, May 2, 2022

Note: There are 8 questions worth a total of 110 points. You have 110 minutes. The exam is ***open book***. **Any form of discussions are *strictly prohibited* during the exam**.

1. (**10pt**) Modern computers are designed based on *von Neumann architecture*.
   (1) Please list the major components of a computer system, and describe how they are connected. (5pt)
   (2) Please give the detailed steps of actions taken by these components during an I/O operation (e.g. reading a file). (5pt)

2. (**10pt**) What's the purpose of *interrupts*? How do we usually handle interrupts in OS kernel? Why are modern operating systems usually interrupt-driven? What are the benefits of this design?

3. (**20pt**) Regarding the following questions about *process and thread management*, please indicate whether each statement is *true* or *false*. **If a statement is incorrect, please explain the reasons to get the full score**. (*not* just correcting the error)
   (1) Kernel-level threads are directly supported and managed by OS kernel. (4pt)
   (2) Multiple user-level threads, as managed by programmers, can always be run in parallel on multicore systems. (4pt)
   (3) The dominant factor of performance gains from adding additional CPU cores to an application is the number of cores. (4pt)
   (4) Multithreaded programs can always provide better performance than a single-threaded solution. (4pt)
   (5) A multithreaded program using multiple user-level threads achieves better performance on a multiprocessor system than on a single-processor system. (4pt)

4. (**10pt**) Answer the following questions regarding *deadlocks*:
   (1) Please explain the necessary conditions for deadlocks. (5pt)
   (2) Among the different methods for handling deadlocks, what are the differences between *deadlock prevention* and *deadlock avoidance*? (5pt)

5. (**10pt**) Consider the following snapshot of a system with five processes P_0 through P_4 and four resource types A, B, C, and D:

	Allocation	*Max*	*Available*
	A B C D	A B C D	A B C D
P_0	3 0 1 3	5 1 1 8	1 0 0 2
P_1	2 2 4 0	3 2 4 1	
P_2	3 1 2 1	3 4 2 1	
P_3	1 4 0 1	4 6 3 3	
P_4	2 2 1 2	6 3 2 5	

   Answer the following questions using the *banker's algorithm*:
   (1) Determine whether or not the system is safe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe. (5pt)
   (2) If a request from process P4 arrives for (0, 0, 0, 2), can the request be granted immediately? Please justify your answer. (5pt)

6. **(15pt)** Answer the following questions regarding memory management.
   (1) What are the differences between *pure paging* and *pure segmentation* in terms of the following aspects: fragmentation, address translation structures, ability to share code across processes, and memory protection? (10pt)
   (2) Consider a paging system with the page table stored in memory. Assume that a memory reference takes 80 nanoseconds. If we add TLBs, and we want the effective memory access time to be less than 10% slowdown, what is the minimum percentage of cache hit in the TLBs? (Assume that finding a page-table entry in the TLBs takes 4 nanoseconds, if the entry is present.) (5pt)

7. (**15pt**) Answer the following questions regarding synchronization. The simplest software tool to the solve the critical-section problem is the *mutex lock*. Instead of implementing mutex locks using atomic hardware instructions, we can simply implement two operations for acquiring and releasing the lock as follows:

```
acquire()
{
   while (!available)
      ;
   available = false;
}
release()
{
   available = true;
}
```

(1) What's the problem with this implementation? Please compare the effects of such implementation on uniprocessor and multiprocessor systems, respectively. (5pt)

(2) What changes would be necessary so that a process waiting to acquire a mutex lock would be blocked and placed into a waiting queue until the lock became available? (5pt)

(3) What is the idea of *semaphore*? Please compare its difference in usage with mutex locks. (5pt)

8. (**20pt**) Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Priority	Burst Time
P_1	2	3
P_2	1	2
P_3	4	10
P_4	2	6
P_5	3	4

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0. Note that a *small* priority number implies a *high* priority. Answer the questions using the following scheduling algorithms: **SJF**, **non-preemptive priority**, and **RR** (time quantum=2).

(1) What is the *turnaround time* of each process for each of the scheduling algorithms? (10pt)
(2) What is the *waiting time* of each process for each of the scheduling algorithms? Which of the algorithms results in the minimum average waiting time (over all processes)? (10pt)