東吳大學　九十四 學年度碩士班研究生招生考試試題

系級	資訊科學系碩士班A組	考試時間	100 分鐘
科目	作業系統	本科總分	100 分

1. True / False (Briefly state the reason, if false; **15%**)

(1) Multiprogramming and multiprocessing both require the use of two or more CPUs.
(2) Smaller time slice (or time quantum) is better for Round Robin (RR) scheduling.
(3) Multitasking permits multiple users to share a computer system resource on an operating system.
(4) In a virtual memory management, fixed frame allocation and local replacement are possible to combination.
(5) To minimize the number of page faults, need larger pages.

2. Why must the bit map (or bit vector) for file allocation be kept on mass storage, rather than in main memory? **(10%)**

3. Explain the reason why Java cannot make a system call directly, and need to provide *native* methods that are written in, say, C or C++ for its system call? **(10%)**

4. List and compare (give at least 3 items) the MAC (Medium Access Control) techniques for solving the link contention of token passing network and Ethernet network. **(10%)**

5. Consider a computing environment, where a process is given the privilege of accessing an object only n times. Suggest a scheme for implementing this policy. **(10%)**

背面尚有試題

系級	資訊科學系碩士班A組	考試時間	100 分鐘
科目	作業系統	本科總分	100 分

6. Following questions are related to *deadlock*: **(15%)**
   (1) What are the four necessary conditions needed before deadlock can occur? **(5%)**
   (2) Also, propose the corresponding typical method of handling deadlock for four following classes of resources. **(10%)**
   (a) Internal resources (e.g. PCBs)
   (b) Central memory
   (c) Process resources
   (d) Swappable space

7. Following questions are related to *memory management*: **(15%)**
   (1) Differentiate "*simple paging*" and "*demand paging*". Also, How many frames are needed for each page in the simple paging and the demand paging, respectively? **(8%)**
   (2) Simply define the "*Thrashing*". Also, propose two strategies to prevent the thrashing in a demand paging. **(7%)**

8. Consider the following set of processes, with the length of the CPU burst given in millisecond: **(15%)**

Process ID.	Arrival time	Burst time	Priority
P_1	0	3	3
P_2	1	5	1
P_3	3	2	3
P_4	9	5	4
P_5	12	5	2

   (a) What is waiting time of each process using *FCFS*, *SJF*, *RR* (quantum = 1), and *non-preemptive priority* (a smaller priority number implies a higher priority) schedules? **(12%)**
   (b) Which one of the schedules can get the minimal average waiting time (over all processes)? **(3%)**