TUTORIAL ONE

Review on Computer System Structures

1. Review the following concepts on Computer System Structures, and explain how they are related to the Operating Systems.

   a) Computer Organization

   b) Interrupt

   c) I/O Structure: Interrupt-Driven Data Transfer, DMA Data Transfer

OS Structures and Concepts

2. Indicate whether the following statements are true or false. Justify your answers.

   a) All I/O instructions are privileged instructions.

   b) Batch system is good for executing long jobs that need little interaction.

   c) Popular operating systems for personal computer use (such as Windows and Linux) are real-time systems.

3. Distinguish between multiprogramming and multiprocessing. What were the key motivations for the development of each?

TUTORIAL TWO

Processes and Threads

1. Indicate whether the following statements are true or false. Justify your answers.

   a) PCB (Process Control Block) is needed in a time-sharing system.

   b) Process control blocks of all processes are stored in the user space of the main memory.

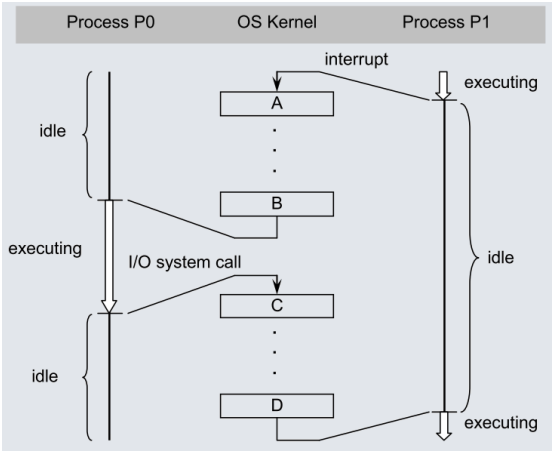
2. Explain the difference and the relationship between a program and a process. Is this difference important in a single process system? Why or why not?

3. A multiprogramming system *sometimes* has no running or ready process. Explain how this is possible and whether the system is therefore "hung".

4. The figure below shows the execution of processes P0 and P1 in a multiprogramming system.

   a) Identify state transitions of each process as shown in the figure.

   b) Describe operations A, B, C and D performed by the operating system kernel as shown in the figure.



TUTORIAL THREE

CPU Scheduling

1. State whether each of the following statements are true or false. Justify your answers.

   (a) A process scheduling discipline is pre-emptive if the CPU cannot be forcibly removed from a process.

   (b) Real-time systems generally use pre-emptive CPU scheduling.

   (c) Time-sharing systems generally use nonpre-emptive CPU scheduling.

2. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

   The process are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

   (a) Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpre-emptive priority (a smaller priority number implies a higher priority), and RR (quantum=1) scheduling.
   (b) What is the turnaround time of each process for each of the scheduling algorithm in part (a)?
   (c) What is the waiting time of each process for each of the scheduling algorithms in part (a)?
   (d) Which of the schedulers in part (a) results in the minimal average waiting time (over all processes)?

3. Measurements of certain system have shown that the average process runs for a time *T* before blocking on I/O. A process switch requires a time *S*, which is effectively wasted (overhead). Define what is meant by CPU efficiency. For round robin scheduling with quantum *Q*, give a formula for the CPU efficiency for each of the following:

   (a) $Q \rightarrow \infty$
   (b) $Q > T$
   (c) $S < Q < T$
   (d) $Q = S$
   (e) $Q \rightarrow 0$

TUTORIAL FOUR

Process Synchronisation- Part I

1. Consider an e-banking system that can handle many user requests. Two different processes are responsible for handling the login and logout procedures (LoginHandler() and ExitHandler(), respectively, as shown in the below figure). A global variable N is used to record the number of users in the system and MAX is the maximum number of users allowed in the system. Describe the potential danger of this program.

```
int N=0; //the number of users
void LoginHandler( ) {
  while (N<MAX) {
     allocate resources to the user;
     N ++;
  }
}
void ExitHandler( ) {
  while (N>0) {
     deallocate resources to the user;
     N --;
  }
}
```

2. (a) Describe the three requirements that should be satisfied by a solution to the critical section problem.

   (b) The following solution attempt to solve critical section problem for two processes.

   Uses a shared variable "flag" declared as:
   int flag[2];
   which is initialized to 0. The program for two processes are as follows:

```
Process 0                          Process 1
while (1) {                        while (1) {
    while (flag[1] == 1);              while (flag[0] == 1);
    flag[0] = 1;                       flag[1] = 1;
    critical section;                  critical section;
    flag[0] = 0;                       flag[1] = 0;
    remainder section;                 remainder section;
};                                 };
```

   Determine which of the three requirements in part (a) are not satisfied. Explain your answer.

3. Mutual exclusion primitives can be implemented with busy waiting or with blocking. Discuss the applicability and relative merits of each approach.

4. Consider a computer that does not have a *TestAndSet* instruction, but does have an instruction to ***swap*** the contents of a register and memory word in a single atomic command. Show how it can be used to implement the *entry section* and *exit section* which are before and after the critical section.

TUTORIAL FIVE

Process Synchronisation- Part II

1. Describe how a counting semaphore can be used to control a pool of identical resources.

2. Show that, if the *wait*(*S*) (see the implementation below, where *S* is a semaphore) operation is not executed atomically, then mutual exclusion may be violated.

```
wait(S){
  while(S<=0) ;
    S--;
}
```

3. Describe how *wait*(*S*) and *signal*(*S*) of a semaphore can be implemented using a *TestAndSet* instruction, given the below semaphore structure definition. *Hints: the semaphore value and the process queue L are shared variables among different processes when those processes access the semaphore.*

```
typedef struct {
    int value;
    struct process *L;
} semaphore;
```

TUTORIAL SIX

Deadlocks

1. Indicate whether the following statements are true or false. Justify your answers.

   (a) In the five philosopher dining problem, if we allow at most four philosophers to be hungry simultaneously, deadlock may still occur.

   (b) It is impossible to have a deadlock involving only one single process.

   (c) If a resource allocation graph contains a cycle, then a deadlock occurs.

2. Use resource-allocation graphs to model the following situations, determine if deadlock occurs in each case. There are three resource types, R, S, and T, each with a single instance.

Case 1	Case 2
P1 requests R	P1 requests R
P2 requests T	P2 requests T
P1 requests S	P1 requests S
P2 requests S	P2 requests S
P1 releases R	P1 requests T
P1 releases S	

3. A resource-allocation state is given below:

	Allocation	Max	Need
USER 1	1	6	5
USER 2	1	5	4
USER 3	2	4	2
USER 4	4	7	3

   Available = 2

   (a) If USER 4 asks for one more unit, does this lead to a safe state or an unsafe one?

   (b) Is USER 3 asks for one more unit, does this lead to a safe state or an unsafe state?

4. The status of a system involving four processes and four resource types at a specific time is as follows:

	Allocation				Request			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	0	0	0	0	1	0	0
P2	0	0	0	1	0	1	1	0
P3	1	0	0	0	0	1	0	0
P4	0	0	1	0	0	0	0	1

   At this moment in time, only two units of resource R2 are available. Use a deadlock detection algorithm to determine if the system is in a deadlock state.

TUTORIAL SEVEN

Memory Organization (Part 1)

1. A particular computer system provides a total memory of 4800 words to users. It supports multiprogramming with multiple-partition. At a given moment in time, the memory is occupied by four processes:

Starting Address of Process	Length (words)
1000	1200
3100	700
4000	400
4400	200

When a new process requests memory, the following method is employed:

STEP 1: Use the Best-fit algorithm to allocate memory for this new process. Do Step 2 only if there is no hole which fits the memory request.

STEP 2: Do memory compaction by moving occupied partitions toward lower memory addresses. Compaction stops when a sufficiently large hole is formed for the new memory request. If, at the end of compaction, there is still insufficient space, the process requesting for more memory is blocked.

The Operating System receives three new memory requirements from processes: 600, 250 and 1050 words (requested in this order).

a) Show the memory allocation after all three requests have been serviced.

b) If the algorithm in Step 1 had been First-fit, show the memory allocation after all three requests have been serviced.

c) In this particular case, which algorithm yields a lower overhead in terms of total relocation of occupied partitions?

2. a) Identify memory allocation schemes that suffer from internal fragmentation.

b) Explain why memory compaction cannot be performed if absolute address format is used in the code.

3. Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

4. Considering a computer system with a 10-bit logical address. Translate the logical address 1000011011 to the corresponding physical address for the following two cases:

   a) Assuming paging is used, the page size is 128 bytes and the page table is as given in Figure Q4a.

   b) Assuming segmentation is used, the maximum segment size that the system can support is 256 bytes, and the segment table is as given in Figure Q4b.

0	01011
1	00010
2	00010
3	10101
4	01001
5	01100
6	11010
7	00110

Figure Q4a

0	00010000	010000000000
1	00001000	100000000000
2	00100000	010011010000
3	00011000	011000110000

Figure Q4b

TUTORIAL EIGHT

Memory Organization (Part 2)

1. Consider a computer system with a 32-bit logical address and 1-Kbyte page size. The system has 1 Gbyes of physical memory.

   a) Give the format of both the logical and physical addresses of this system.

   b) How many entries are there in a page table?

   c) If an inverted page table is used, how many entries are there?

2. A paged memory system uses the page size of 1024 bytes. Size of a page table entry is 4 bytes and the logical address space is 2^{30} bytes.

   a) What is the size of the page table if single level of paging is used?

   b) What is the minimum number of levels of page tables needed in this system to ensure that the outmost page table will fit within a single page frame?

   c) Draw an address translation diagram to show how logical address translation is performed.

Virtual Memory (Part 1)

3. A computer has four page frames. The time of loading, time of last access, and the R bit for each page are as shown below (the times are in clock ticks):

PAGE	LOADED	LAST ACCESS.	R
0	126	279	0
1	230	260	0
2	120	272	1
3	160	280	1

   a) Which page will FIFO replace?
   b) Which page will second chance replace?
   c) Which page will LRU replace?

4. For each of the page replacement policies listed below, calculate the number of page faults encountered when referencing the following pages:

   0 1 6 0 3 4 0 1 0 3 4 6 3 4

   Assume the availability of 4 empty page frames.

   a) FIFO
   b) CLOCK
   c) LRU

TUTORIAL NINE

Virtual Memory (Part 2)

1. Consider a demand-paged virtual memory system with M page frames. Assume initially all frames are empty. Given a page reference string with length L and N distinctive page numbers (N < L),

   a) What are the lower and upper bounds of page faults that may be generated by such a reference string if the FIFO page replacement algorithm is used?

   b) Will you have different answers if the LRU (Least-Recently-Used), instead of FIFO, page replacement algorithm is used?

2. Suppose that a process is allocated four page frames in a demand-paged virtual memory system. The time of loading and the time of last access are as shown in the table below.

Frame #	Page #	Time Loaded	Time Last Accessed
0	1	60	161
1	0	130	160
2	3	26	162
3	2	20	163

   Given the above memory state before the fault and the remaining page reference string of the process: 4, 0, 0, 0, 2, 4, 2, 1, 0, 3, 2, answer the following two questions.

   a) Calculate how many page faults will be generated if Least-Recently-Used (LRU) algorithm is used. Show your workings.

   b) Assuming that the size of the working set window is four, calculate the number of page faults that would occur if only the pages in the working set were loaded into the memory instead of using a fixed allocation of four frames. Show your workings.

3. Consider a demand-paging system with the following time-measured utilizations:

   CPU utilization 20%
   Paging disk 97.7%
   Other I/O devices 5%

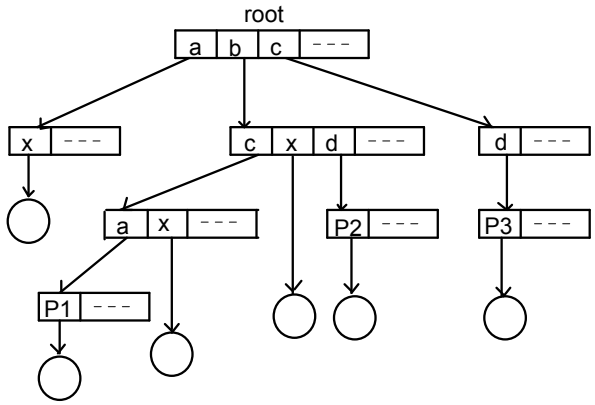
   Which (if any) of the following will (probably) improve CPU utilization?

   a. Install a faster CPU.
   b. Install a bigger paging disk.
   c. Increase the degree of multiprogramming.
   d. Decrease the degree of multiprogramming.
   e. Install more main memory.
   f. Install a faster hard disk or multiple hard disks.
   g. Increase the page size.

TUTORIAL TEN

File-Systems

1. The following figure shows part of a tree-structured directory of a file system. Assume that the components within a path name are separated by a slash (/).



a) Give the path names of each non-directory file.

b) The file system maintains an Accessed File Table (AFT) which contains every file accessed by the user during a given log-in session.

When a file is referenced, the following rules apply:

I. Search AFT. If file not found, try rule II.

II. Search caller's parent directory. If file not found try rule III.

III. Search current working directory. If file not found, try rule IV.

IV. Search library directory. If file not found, report error.

i) File P1, P2 and P3 are programs which contain references to a file x. Give the path name of the particular file x referenced when P1, P2 and P3 are executed separately. Assume that the current working directory is where the program is, the AFT is empty and the library directory is /a.

ii) Is it possible to execute P1 and have file /b/x referenced without changing the search rules or modifying directory entries? Explain your answer.

2. a) Consider a file system where a file can be deleted and its disk space reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

b) Some systems provide file sharing by maintaining a single copy of a file; other systems maintain several copies, one for each of the users sharing the file. Discuss the relative merits of each approach.

3. Assume that a file system uses Unix-like inodes with six direct pointers and one single-indirect pointer. The file system is block-oriented with both logical and physical block sizes of 1000 bytes. A user executes a program that contains the following code:

```
fd = open("/usr/ast/mbox");        // open a file
seek(fd, 5900);  // move the file pointer
read(fd, buf, 200);        // read 200 bytes of data from the file
```

a) How many disk read operations are required for the first system call (i.e., open) in the above program fragment? Assume that initially, the root directory is in memory, inodes are in disk, and that a directory can fit into a single block.

b) How many disk read operations are required for the last two system calls (i.e., seek and read) in the above program fragment?

c) What is the maximum file size the file system can support? Assume that each entry in the indirect block takes 2 bytes.

4. Some file systems use two block sizes for disk storage allocation, for example, 4-Kbyte and 512-byte blocks. Thus, a 6 Kbytes file can be allocated with a single 4-Kbyte block and four 512-byte blocks. Discuss the advantage of this scheme compared to the file systems that use one block size for disk storage allocation.

TUTORIAL ELEVEN

I/O Systems

1. Why is it necessary to separate the device driver layer from the kernel I/O subsystem?

2. Suppose that in a multiprogramming system, a process reads blocks of data from a file on disk for processing. As shown below, it reads one block of data at a time to a buffer using synchronous I/O and then processes the data.

```
while ( not end of file) {
        buffer <- read a block of data from disk using synchronous I/O;
        process data in buffer;
}
```

   a) Discuss how the performance of the above process can be improved.

   b) For a system running mainly with this type of processes, which file allocation scheme is best in terms of I/O performance?

Disk

3. a) During his presentation, a salesman emphasized on the substantial effort his company has made to improve the performance of their UNIX version - one example he quoted was that the disk driver used the SCAN algorithm and also queued multiple requests within a cylinder in sector order. You bought a copy and wrote a program to randomly read 10,000 blocks spread across the disk. The performance measured was the same as what would be expected from FCFS algorithm. Was the salesman lying?

   b) Another salesman claims that his system uses sector queueing to minimize head movement on moving head disks. What is wrong with this comment?

   c) Under what circumstances could a disk scheduling discipline not improve the performance or even degrade performance of the system?

4. Assume that a disk drive has 200 cylinders, numbered 0 to 199. The disk head starts at cylinder 0. A seek takes $(20 + 0.1 \times T)$ milliseconds, where T is the number of cylinders to move. Rotational latency is 2 milliseconds and data transfer per request takes 8 milliseconds, assuming each request accesses the same amount of data. The following table shows the arrival time and destination cylinder number of requests:

Arrive Time (ms)	0	15	20	23	30	35	50	65	70	88
Cylinder Number	45	132	35	4	23	50	70	40	10	35

   Compute the average time to service a request using the Shortest Seek Time First (SSTF) disk head scheduling algorithm.