PRACTICE PROBLEMS

OPERATING SYSTEMS: INTERNALS AND DESIGN PRINCIPLES

SIXTH EDITION

WILLIAM STALLINGS

ACKNOWLEDGEMENTS

I would like to thank the following people for providing problems for this collection: Prof. Tracy Camp (Colorado School of Mines); Prof. Oskars J. Rieksts (Kutztown University); Dr. I-Ling Ye (U. of Texas, Dallas); Prof. Thomas N. Scanlan (U. of Wisconsin, Platteville); Klara Nahrstedt (U. of Illinois, Urbana-Champaign); Prof. Douglas Niehaus (U. of Kansas).

TABLE OF CONTENTS

CHAPTER 1 COMPUTER SYSTEM OVERVIEW

1.1 Discuss the mechanism for interrupt handling of I/O operations. Be sure to differentiate between hardware (or firmware) functions and software (OS) functions.

1.2 Consider the following pseudo assembly code for computing c = a + b. Assume that a, b, and c are assigned to consecutive memory "words" (memory is generally addressed byte by byte and assume that a word is 4 bytes) and address for "a" is 0x0000ec00. Also, we have a = 22, b = 158, and c = 0 at the starting time. Assume that the first instruction of the code is stored in 0x0000b128. Also, each instruction has the opcode in the first byte (most significant byte) and the remaining 3 bytes specify the corresponding address. The opcode for store is 1, load is 2, and add is 3.

```
0x0000b128 load a
0x0000b12c add b
0x0000b130 store c
```

a. Show the memory addresses and contents for all the instructions and data involved. Use the format as follows to express your answer (but the following is not the answer). For all data, use hexadecimal representation.

addresses	contents
0x00002104	0x00000001
0x00002108	0x00000002
……	……

b. Write the micro instructions for the code segment. Assume that current PC (program counter) is 0x00001018. For each micro-instruction, also indicate the data that is transferred (if it is a memory access). For all data, use the hexadecimal representation. The following are the first two micro-instructions and the data transferred. Complete the rest.

Micro-instructions	data
PC → MAR	0x0000b128
M → MBR	0x0200ec00

1.3 A computer has a cache, main memory, and a disk used for virtual memory. An access to the cache takes 10 ns. An access to main memory takes 100 ns. An access to the disk takes 10,000 ns. Suppose the cache hit ratio is 0.9 and the main memory hit ratio is 0.8. What is the effective access time (EAT) in ns required to access a referenced word on this system?

1.4 **a.** Consider a main memory system that consists of a number of memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address, and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The (internal) operation of different memory modules may overlap in time, but only one request can be on the bus at any time.

b. Sketch a graph of the maximum write rate (words per second) as a function of the module cycle time, assuming eight memory modules and a fixed bus busy time of 100 ns.

CHAPTER 2 OPERATING SYSTEM OVERVIEW

2.1 Suppose a short-term scheduling algorithm favors those processes that have used little processor time in the recent past.

a. Explain why this algorithm favors I/O-bound processes.

b. Explain why this algorithm does not permanently deny processor time to CPU-bound processes.

2.2 The classical batch processing system ignores the cost of increased waiting time for users. Consider a single batch characterized by the following parameters:

M	average mounting time
T	average service time per job
N	number of jobs
S	unit price of service time
W	unit price of waiting time per user

a. Show that the optimal batch size that minimizes the cost of service time and waiting time per user (within a single batch) is

$$N_{opt} = \sqrt{\frac{M}{T}\frac{S}{W}}$$

b. In an installation in which M = 5 min, T = 1 min, and S = $300/hr, the operators choose N = 50. Assuming that this is an optimal choice, find the unit cost of the user's waiting time W.

2.3 In the Exec II batch system, users would submit a large number of jobs in the morning. These jobs took hours to complete and thereby prevented fast response. Suggest a modification of the scheduling policy that would discourage users from doing this.

2.4 In the Scope system for the CDC 6600 computer, system resources (processor time, storage, etc.) can remain idle while running jobs wait for operators to mount magnetic tapes. Suggest a solution to this problem.

2.5 Measurements on the CTSS system showed that about half of all user requests could be classified as file manipulation, program input, and editing. How would you use this information about expected workload to improve processor utilization at a reasonable cost without degrading user response?

CHAPTER 3 PROCESS DESCRIPTION AND CONTROL

3.1 The following state transition table is a simplified model of process management, with the labels representing transitions between states of READY, RUN, BLOCKED and NONRESIDENT.

	READY	RUN	BLOCKED	NONRESIDENT
READY	–	1	–	5
RUN	2	–	3	–
BLOCKED	4	–	–	6

Give an example of an event that can cause each of the above transitions. Draw a diagram if that helps.

3.2 You have executed the following C program:

```
main ()
{   int pid;
    pid = fork ();
    printf ("%d \n", pid);
}
```

What are the possible outputs, assuming the fork succeeded?

3.3 Assume that at time 5 no system resources are being used except for the processor and memory. Now consider the following events:

At time 5: P1 executes a command to read from disk unit 3.
At time 15: P5's time slice expires.
At time 18: P7 executes a command to write to disk unit 3.
At time 20: P3 executes a command to read from disk unit 2.
At time 24: P5 executes a command to write to disk unit 3.
At time 28: P5 is swapped out.
At time 33: An interrupt occurs from disk unit 2: P3's read is complete.
At time 36: An interrupt occurs from disk unit 3: P1's read is complete.
At time 38: P8 terminates.
At time 40: An interrupt occurs from disk unit 3: P5's write is complete.
At time 44: P5 is swapped back in.
At time 48: An interrupt occurs from disk unit 3: P7's write is complete.

For each time 22, 37, and 47, identify which state each process is in. If a process is blocked, further identify the event on which is it blocked.

3.4 What would happen if you executed the following piece of code:

```
main()
{
   for(;;)
      fork();
}
```

Note: Don't try this on a real machine.

3.5 During a process switch, the operating system executes instructions that choose the next process to execute. These instructions are typically at a fixed location in memory. Why?

CHAPTER 5 CONCURRENCY: MUTUAL EXCLUSION AND SYNCHRONIZATION

5.1 In a multiprocessing environment, a single dispatch list may be accessed by each multiprocessor's dispatcher. Give an example of a race condition that can occur if two dispatchers access the dispatch list concurrently.

5.2 Nicole Kidman, Penelope Cruz, and Mimi Rogers, like all members of Hollywood, often eat at Spago's. But Nicole, Penelope, and Mimi would rather NOT eat at Spago's on the same evening as one of the other two. Think of Spago's as a system resource (e.g., file). Think of Nicole, Penelope, and Mimi as three processes needing access to this resource
Does the following solution to the critical section problem for three processes satisfy the desired mutual exclusion requirements (mutual exclusion, no deadlock, no starvation, process can enter empty critical section without delay)? Justify your response by discussing each of the critical section requirements, if the solution is correct; or one of the critical section requirements, if the solution is not correct.

```
have three flags (shared variables): Cole, Lope, and Mimi
have neon sign (shared variable) which flashes either
"Nicole" or "Penelope" or "Mimi-R"
initially: all flags are down and sign flashes "Nicole"
```

Nicole	**Penelope**	**Mimi**
`repeat`	`repeat`	`repeat`
`    [play with kids];`	`    [call Tom];`	`    [make a movie];`
`    raise Cole;`	`    raise Lope;`	`    raise Mimi;`
`    SIGN = Nicole;`	`    SIGN = Penelope;`	`    SIGN = Mimi-R;`
`    while (((Lope is up)`	`    while (((Cole is up)`	`    while (((Cole is up)`
`        OR (Mimi is up))`	`        OR (Mimi is up))`	`        OR (Lope is up))`
`        AND (SIGN == Nicole))`	`        AND (SIGN == Penelope))`	`        AND (SIGN == Mimi-R))`
`        [do nothing];`	`        [do nothing];`	`        [do nothing];`
`    [eat dinner];`	`    [eat dinner];`	`    [eat dinner];`
`    lower Cole;`	`    lower Lope;`	`    lower Mimi;`
`forever;`	`forever;`	`forever;`

5.3 Does the following solution to the critical section problem for two processes satisfy the desired mutual exclusion requirements (mutual exclusion, no deadlock, no starvation, process can enter empty critical section without delay)? Justify your response by discussing each of the critical section requirements, if the solution is correct; or one of the critical section requirements, if the solution is not correct.

```
have two flags (shared variables): Cole and Lope
have neon sign (shared variable) which flashes either
"Nicole" or "Penelope"
initially: both flags are down and sign flashes "Nicole"
```

Nicole	**Penelope**
`repeat`	`eat`
`  [play with kids];`	`  [call Tom];`
`  raise Cole;`	`  raise Lope;`
`  while (Lope is up) {`	`  while (Cole is up) {`
`    if (SIGN == Penelope) {`	`    if (SIGN == Nicole) {`
`      lower Cole;`	`      lower Lope;`
`      while (SIGN == Penelope)`	`      while (SIGN == Nicole)`
`         [do nothing];`	`         [do nothing];`
`      raise Cole;`	`      raise Lope;`
`    }`	`    }`
`  }`	`  }`
`  [eat dinner];`	`  [eat dinner];`
`  SIGN = Penelope`	`  SIGN = Nicole`
`  lower Cole;`	`  lower Lope;`
`forever;`	`forever;`

5.4 Some do not agree with the name of the semaphore's wait operation. They do not like the word: *wait*. Explain.

5.5 Consider the following functions that are shared by multiple processes:

```
static int count = 0;
int increment(void)
{
  count++;
  if (count > 5) {
       printf("counter %d reached value > 5", count);
       return 0;
  }
  return 1;
}

int decrement(void)
{
  while (count >5) {
       printf("counter %d is > 5:", count);
       count --;
  }
  if (count ==0) return 0;
  else return 1;
}
```

Use any synchronization primitives (semaphores, mutex) to make the two functions atomic.

5.6 The Busy Bank Problem. A bank has n tellers, all serving customers at teller windows. k of the n tellers are designated as "quick service" tellers, who only process single withdrawal transactions. Customers arriving at the bank join one of two queues, the "normal service" queue, or the "quick service" queue. They may only join the latter queue if they wish to make a single withdrawal transaction. Customers in the quick service queue are processed by the k quick service tellers, unless the normal service queue is empty, when they can also be processed by any of the normal service tellers. Customers in the normal service queue are processed by the (n-k) normal service tellers, unless the quick service queue is empty, when they can also be processed by any of the k quick service tellers.

Write a suite of suitable synchronized concurrent processes to model the processing of customers by the bank. You may use any (formal) programming language notation you prefer, as long as you clearly identify it. *Hint:* study the structure of the barbershop problem in Appendix A.

5.7 Explain what is the problem with this implementation of the one-writer many-readers problem?

```
int readcount;            // shared and initialized to 0
Semaphore mutex, wrt;     // shared and initialized to 1;

// Writer :               // Readers :
                          semWait(mutex);
                          readcount := readcount + 1;
semWait(wrt);             if readcount == 1 then semWait(wrt);
/* Writing performed*/    semSignal(mutex);
semSignal(wrt);           /*reading performed*/
                          semWait(mutex);
                          readcount := readcount - 1;
                          if readcount == 0 then Up(wrt);
                          semSignal(mutex);
```

5.8 Consider the following program:

```
P1: {                              P2:{
  shared int x;                      shared int x;
  x = 10;                            x = 10;
  while (1)  {                       while ( 1 ) {
     x = x - 1;                         x = x - 1;
     x = x + 1;                         x = x + 1;
     if (x != 10)                       if (x!=10)
       printf("x is %d",x)              printf("x is %d",x)
     }                                  }
  }                                  }
}                                  }
```

Note that the scheduler in a uniprocessor system would implement pseudo-parallel execution of these two concurrent processes by interleaving their instructions, without restriction on the order of the interleaving.

a. Show a sequence (i.e., trace the sequence of interleavings of statements) such that the statement "x is 10" is printed.

b. Show a sequence such that the statement "x is 8" is printed. You should remember that the increment/decrements at the source language level are not done atomically, i.e., the assembly language code:

```
LD      R0,X   /* load R0 from  memory location x */
INCR    R0     /* increment R0 */
STO     R0,X   /* store the incremented value back in X */
```

implements the single C increment instruction (x = x + 1).

5.9 Show where/how to add semaphores to the program in the preceding problem to insure that the printf() is never executed. Your solution should allow as much concurrency as possible.

5.10 Suppose that an operating system does not have access to an atomic test-and-set, compare and swap, or a lock instruction, but does have the ability to mask out interrupts with a spl(x) routine which sets the processor priority level to x; the return value of spl() is the old processor priority level. spl(x) is a privileged instruction (can only be executed in kernel mode) and is atomic. Sketch how you would make/implement an operating system service (system call) for the user that would allow her/him to easily implement mutually exclusive access to a critical section. A busy-waiting solution to the problem is OK, but remember that busy waiting inside the kernel will hang the system.

5.11 In the discussion of Figure 5.9, the book makes the following comment: " It would not do simply to move the conditional statement inside the critical section of the consumer because this could lead to deadlock (e.g., after line 8 of the table)." Explain how this could lead to deadlock.

CHAPTER 6 CONCURRENCY: DEADLOCK AND STARVATION

6.1 Consider the following snapshot of a system. There are no outstanding unsatisfied requests for resources.

available			
r1	r2	r3	r4
1	5	2	0

	current allocation				maximum demand			
process	r1	r2	r3	r4	r1	r2	r3	r4
p0	0	0	1	2	0	0	1	2
p1	1	0	0	1	1	5	1	1
p2	1	3	0	4	2	3	5	6
p3	0	6	3	2	0	6	7	2
p4	0	0	1	4	0	6	5	6

Is the system safe?

6.2 In the context of Dijkstra's Banker's Algorithm, discuss whether the following state is safe or unsafe. If safe, show how it is possible that all processes complete.

available	
A	B
10	15

	current allocation		**maximum demand**	
process	A	B	A	B
User 1	2	3	10	5
User 2	3	3	8	5
User 3	2	2	4	4
User 4	2	5	4	8

6.3 Given the following state for the Banker's Algorithm.
6 processes P0 through P5
4 resource types: A (15 instances); B (6 instances)
C (9 instances); D (10 instances)
Snapshot at time T0:

Available

A	B	C	D
6	3	5	4

	current allocation				**maximum demand**			
process	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

a. Verify that the Available array has been calculated correctly.
b. Calculate the Need matrix.
c. Show that the current state is safe, that is, show a safe sequence of processes. In addition, to the sequence show how the Available (working array) changes as each process terminates.
d. Given the request (3,2,3,3) from Process P5. Should this request be granted? Why or why not?

6.4 For this problem, consider the following ways of handling deadlock.
(i) banker's algorithm
(ii) detect deadlock and kill thread, releasing all its resources
(iii) reserve all resources in advance
(iv) restart thread and release all resources if thread needs to wait
(v) resource ordering
(vi) detect deadlock and roll back thread's actions
a. One criteria to use in evaluating different approaches to deadlock is which permits the greatest concurrency -- in other words, which allows the most threads to make progress without waiting when there isn't deadlock. Give a rank order from 1 to 6 for each of the ways of handling deadlock listed below, where 1 allows the greatest degree of concurrency, and 6 allows the least concurrency; if two approaches offer roughly equal concurrency, indicate that.
b. Another criteria to use in evaluating deadlock algorithms is efficiency -- in other words, which requires the least CPU overhead. Rank order the approaches according to efficiency, with 1 being the most efficient, assuming that encountering deadlock is a very rare event. Again, indicate if two approaches are equally efficient. Does your ordering change if deadlocks happen frequently?

6.5 Suppose the following two processes, foo and bar are executed concurrently and share the semaphore variables S and R (each initialized to 1) and the integer variable x (initialized to 0).

```
void foo( ) {
    do {
        semWait(S);
        semWait(R);
        x++;
        semSignal(S);
        SemSignal(R);
    } while (1);
}
```

```
void bar( ) {
   do {
       semWait(R);
       semWait(S);
       x--;
       semSignal(S;
       SemSignal(R);
   } while (1);
}
```

a. Can the concurrent execution of these two processes result in one or both being blocked forever? If yes, give an execution sequence in which one or both are blocked forever.

b. Can the concurrent execution of these two processes result in the indefinite postponement of one of them? If yes, give an execution sequence in which one is indefinitely postponed.

CHAPTER 7 MEMORY MANAGEMENT

7.1 Assume that we are using OS/MVT (variable size partitions) and we have batch processes arriving in order, with the following requests for storage:

20k, 30k, 10k, 100k, 60k

Assume that we have four holes in memory of size: 50k, 30k, 200k, 16k, 30k Where would each of the First-fit, Best-fit and Worst-fit algorithms place the processes?

7.2 Consider the following segment table:

Segment	**Base**	**Length**
0	1219	600
1	3300	14
2	90	100
3	2327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0, 4302 **b.** 1, 15 **c.** 2, 50 **d.** 3, 400 **e.** 4, 112

7.3 Suppose a fixed partitioning memory system with partitions of 100K, 500K, 200K, 300K, and 600K (in memory order) exists. All five partitions are currently available.

a. Using the best fit algorithm, show the state of memory after processes of 212K, 417K, 112K, and 350K (in request order) arrive.
b. Using the best available algorithm, show the state of memory after processes of 212K, 417K, 112K, and 350K (in request order) arrive.
c. At the end of part (a), how much internal fragmentation exists in this system?
d. At the end of part (b), how much external fragmentation exists in this system?

7.4 Consider a logical address space of eight pages of 1024 bytes each, mapped onto a physical memory of 32 frames.
a. How many bits are there in the logical address?
b. How many bits are there in the physical address?

7.5 A system uses (contiguous) dynamic partition memory management, with 110K of memory for user processes. The current memory allocation table is shown below:

Job	Base Address	Length
A	20	10
B	46	18
C	90	20

A new job, D, arrives needing 15K of memory. Show the memory allocation table entry for job D for each of the following memory allocation strategies: first fit, best fit, worst fit.

7.6 Consider a system with a 16KB memory. The sequence of processes loaded in and leaving the memory are given in the following.

P1	7K	loaded
P2	4K	loaded
P1		terminated and returned the memory space
P3	3K	loaded
P4	6K	loaded

Assume that when a process is loaded to a selected "hole", it always starts from the smallest address. E.g. P1 will be loaded in memory locations 0 to 8K – 1 since the entire memory is one hole. Give the memory map showing allocated portion and free portion after the end of the sequence (if a process cannot be loaded, indicate that) for the following placement algorithms. Also, indicate the internal/external fragmentations.

a. first fit
b. best fit
c. buddy
d. simple paging (assume that each page is of size 2K)

7.7 On a system with 2^{32} bytes of memory and fixed partitions with a partition size of 2^{20} bytes, what is the minimum number of bits needed in an entry in the process table to record the partition to which a process has been allocated? **7.8** On a system using fixed partitions with partition sizes of 2^8, 2^{24}, and 2^{64} bytes, how many bits must the bounds register (Figure 7.8) have?

CHAPTER 8 VIRTUAL MEMORY

8.1 Consider a demand paging system with the following time-measured utilizations.
processor utilization 20 %
paging disk 97.7 %
other I/O devices 5 %
Which of the following if any will probably improve processor utilization
a. Install a faster processor.
b. Install a bigger paging disk
c. Increase the degree of multiprogramming
d. Install more main memory
e. Decrease the time quantum allocated to each process

8.2 What is the probable cause of thrashing?
a. I/O drivers that are not properly connected to the I/O subsystem.
b. Repeated parity errors
c. Disk crashes
d. A local page replacement algorithm
e. Process (e.g., being unable to establish their working set of pages
f. A FIFO page replacement algorithm

8.3 Consider a paging system with a page table stored in memory.
a. If a memory reference takes 200 nanoseconds, how long does a paged reference take (reference is retrieved from page table in memory and then program data is retrieved from memory) Assume all pages are in memory?
b. If we add associative registers (TLB), and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time on average? (Assume that finding a page table entry in the associative registers takes 10 nanoseconds, if the entry is there.) Assume all pages are in memory.
c. What is the effective memory reference time if the hit ratio for the associative registers is 70%, the hit ratio for memory is 20%, and 10% of the time we must go out to the disk to load the faulted page in memory, where context switching, disk transfer time, etc takes 100 m seconds.).

8.4 Assume a demand paged memory management system with the following characteristics:

page size = 4K = 2^{12} bytes
physical address space = 2^{24} bytes
logical address space = 2^{32} bytes
TLB size = 2^6 bytes

a. How many bits are needed to specify a logical address? Give the number of bits for the page number and offset.

b. How many page table entries can fit in the TLB if each entry only contains the information needed for logical to physical translation. Show your work.

c. How many page table entries are needed when a program uses its full logical address space?

d. Part (c) indicates a serious problem that arises from having a very large logical address space. What is this problem and how could an OS solve it? Discuss the consequences of your solution for runtime overhead during program execution (just a few sentences).

8.5 Suppose:

TLB lookup time = 20 ns
TLB hit ratio = 80%
memory access time = 75 ns
swap page time = 500,000 ns
50% of pages are dirty
OS uses a single level page table

a. What is the effective access time (EAT) if we assume the page fault rate is 0%? (Show your work.) Assume the cost to update the TLB, the page table, and the frame table (if needed) is negligible.

b. What is the effective access time (EAT) if we assume the page fault rate is 10%? (Show your work.) Assume the cost to update the TLB, the page table, and the frame table (if needed) is negligible.

8.6 Consider the following page reference string:

0, 1, 7, 0, 1, 2, 0, 1, 2, 3, 2, 7, 1, 0, 3, 1, 0, 3

How many page faults would occur for the following replacement algorithms? Assume no prepaging occurs, and show what pages are in memory at each given time.

a. OPT replacement (three frames are allocated to the process)
b. FIFO replacement (three frames are allocated to the process)
c. Pure LRU replacement (three frames are allocated to the process)
d. Clock Policy (three frames are allocated to the process)

8.7 Suppose an instruction takes 1 nanosecond to execute (on average), a page fault takes 20 microseconds of processor time, and it takes 300 microseconds of disk time to read or write a single page. Suppose that on average, 1/2 of the pages are modified. What is the average number of instructions between page faults that would cause the disk to be busy doing page transfers all the time?

8.8 A virtual memory system exhibits the follow trace of page numbers:

1 2 3 2 6 3 4 1 5 6 1 6 4 2

Simulate the page replacements for the following scenarios:

a. FIFO with 3 page frames
b. FIFO with 4 page frames
c. LRU with 3 page frames
d. LRU with 4 page frames

and show the number of page faults that occur with each. Does Belady's Anomaly occur? Belady's anomaly states that it is possible to have more page faults when increasing the number of page frames under certain conditions. This was reported in "An Anomaly in Space-Time Characteristics of Certain Programs Running in a Paging Machine," by Belady et al, *Communications of the ACM*, June 1969.

8.9 Consider the following program.

```
#define Size 64
int A[Size; Size], B[Size; Size], C[Size; Size];
int register i, j;

for (j = 0; j < Size; j ++)
   for (i = 0; i < Size; i++)
      C[i; j] = A[i; j] + B[i; j];
```

Assume that the program is running on a system using demand paging and the page size is 1KB. Each integer is 4 bytes long. It is clear that each array requires 16-page space. As an example, A[0,0]-A[0,63], A[1,0]-A[1,63], A[2,0]-A[2,63], and A[3,0]-A[3,63] will be stored in the rst data page. A similar storage pattern can be derived for the rest of array A and for arrays B and C. Assume that the system allocates a 4-page working set for this process. One of the pages will be used by the program and three pages can be used for the data. Also, two index registers are assigned for i and j (so, no memory accesses are needed for references to these two variables).

a. Discuss how frequently the page fault would occur (in terms of number of times C[i,j] = A[i,j] + B[i,j] are executed).
b. Can you modify the program to minimize the page fault frequency?
c. What will be the frequency of page faults after your modification?

8.10 Certain algorithms that work very well when all of the data fit in memory do not work so well when the virtual memory exceeds the available physical memory, due to the amount of time that may be spent on page replacement. The following program multiplies two matrices, where each matrix is stored as a two dimensional array. The array is stored in row-order, meaning that consecutive elements in the same row of the array will be stored in consecutive locations in memory.

```
multiply(A,B,C)
   {
   int A[SIZE][SIZE]. B[SIZE][SIZE], C[SIZE][SIZE];
   int i,j,k,temp;

   for all i     /* this done in parallel */
      for (j=0; j<SIZE; j++)  {
          temp = 0;
          for (k=0; k<SIZE; k++)
             temp += a[i][k] * b[k][j];
          c[i][j] = temp;
   }
}
```

Suppose the above program was written for a shared-memory multiprocessor. Each processor has a number assigned to it to use for the variable i. Therefore, each processor is responsible for computing a row of C. Furthermore, assume that they all execute in parallel at the same rate. We will be interested in the performance of this program in light of very large arrays, which will not all fit in memory at the same time. For this problem, assume an optimal page replacement algorithm, where 1000 pages are available for the arrays (you can ignore i,j,k and the source program). Each page may hold up to 1000 elements of any array.

Assume there are 100 processors, assigned numbers from 0 to 99. Each will be responsible for doing a set of rows in the multiplication. For example, when multiplying 1000x1000 matrices, Processor 0 will find the first 10 rows, Processor 1 the next 10, and so on. The code each processor runs will be as follows:

```
int A[SIZE][SIZE]. B[SIZE][SIZE], C[SIZE][SIZE];
multiply(A,B,C)
     {
     int i,j,k,temp;
     int mult = SIZE / 100;

     for (i=mult*ProcN ; i<mult * (ProcN + 1); i++)
         for (j=0; j<SIZE; j++)
              {
              temp = 0;
              for (k=0; k<SIZE; k++)
                  temp += a[i][k] * b[k][j];
              c[i][j] = temp;
              }
     }
```

It is assumed that these are running roughly in parallel, though their memory accesses will be serialized by the memory unit. Therefore, accesses to the arrays will look something like this:

```
{ a[0][0], a[10][0], a[20][0], .... a[990][0] }   one access
{ b[0][0], b[ 0][0], b[ 0][0], .... b[  0][0] }    for each
{ a[0][1], a[10][1], a[20][1], .... a[990][1] }    processor
{ b[1][0], b[ 1][0], b[ 1][0], .... b[  1][0] }
```

etc.

An Optimal Page Replacement Policy is one that accomplishes all this in the fewest page faults -- based on clairvoyantly predicting what will be needed soon, and should not be replaced. Here, for example, several pages of A are used repeatedly, so should hang about in memory for a long time.
The circumstances to consider are the following:

a. SIZE = 1,000 (total memory required is 3 times that available)
b. SIZE = 10,000 (total memory required 300 times that available)
Estimate how many page faults occur in each case, (one significant digit is enough) and identify where most of them come from.

8.11 For this problem, assume a demand paged virtual memory system with a page size of 100 words (decimal values throughout). A process running on this system generates a sequence of logical addresses, given in the table below.

10	11	104	170	73	309	185	245	246	434	458	364

Assume the process is allotted exactly two page frames, and that none of its pages are resident (in page frames) when the process begins execution.

a. Determine the page number corresponding to each logical address and fill them into a table with one row and 12 columns. This is often called a reference string for the process.
b. Consider the reference string determined in part (a). Determine which references result in page faults, assuming FIFO page replacement is used, indicating your conclusions by placing Fs in the corresponding cells of a table with one row and 12 columns. The page fault rate is the number of page faults divided by the total number of references made. What is the page fault rate for this case? Round your answer to an integer percentage.
c. Repeat (b) for LRU.
d. Repeat (b) for optimal page replacement.

8.12 Assume a system that has 224 words of virtual memory and 218 words of physical memory, with a page size of 28 words. All addresses in this system are expressed in hexadecimal. A particular process running on this system is allocated 2700 words of virtual memory. The current page map table for that process is shown below:

Logical Page #	**Frame #**	**VM Page #**
000	212	0021
001	002	01A0
002		7FA3
003	3B2	7FA4
004	3F1	7FA9
005		901B
006		A113
007		CA00
008		CA01
009	012	CA02
00A		CA03

a. How many pages of virtual memory does this system have? Give a simplified decimal answer.

b. How many page frames does this system have? Give a simplified decimal answer.

In each of the following questions you are given a logical address generated by this process. Determine the virtual and physical addresses that correspond to each logical address. If there is no valid virtual or physical address indicate why. Express both physical and virtual addresses in hexadecimal.

c. 3F1

d. 214

e. A90

CHAPTER 9 UNIPROCESSOR SCHEDULING

9.1 Consider the following workload:

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

a. Show the schedule using Shortest Remaining Time, non-preemptive Priority (a smaller priority number implies higher priority) and Round Robin with quantum 30 ms. Use time scale diagram as shown below for the FCFS example to show the schedule for each requested scheduling policy.
Example for FCFS (1 unit = 10 ms):

```
| P1 | P1 | P1 | P1 | P1 | P2 | P2 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P4 | P4 | P4 | P4 |
0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20
```

b. What is the average waiting time of the above scheduling policies?

9.2 A processor scheduling workload is the sequence of jobs that arrive for processor time, when they arrive, and how much processor time they need. (In other words, the processor scheduling policy is a kind of an algorithm, and the workload is the input to the algorithm.) For each of the following processor scheduling policies, describe the set of workloads under which that policy is optimal (minimizes average response time) and the set of workloads under which the policy is pessimal (maximizes average response time). If there are no workloads under which a policy is optimal or pessimal, indicate that.

a. FIFO
b. round robin
c. shortest time to completion first with preemption
d. multilevel feedback
e. lottery scheduling where each job is assigned a number of lottery tickets inversely proportional to how much processor time it has consumed since blocking

9.3 Suppose the following jobs are to be executed in a uniprocessor system.

Job Number	Arrival Time	Service Time
1	0	4
2	1	8
3	3	2
4	10	6
5	12	5

Assume the overhead of context switching is one time unit. For each of the following scheduling methods, give (i) a timing chart to illustrate the execution sequence, (ii) the average job turnaround time, (iii) the normalized turnaround time for each job, and (iv) the processor efficiency.

a. FCFS
b. SPN
c. SRTN
d. RR, quantum = 3
e. Multilevel Feedback Queue with queues numbered 1-10, quantum = 2i, where i is the queue level number and processes are initially placed in the first queue (i.e., level 1). In this scheduling policy, each process executes at a particular level for one quantum and then moves down a level; processes never move up a level.

9.4 Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.
a. What would be the effect of putting two pointers to the same process in the ready queue?
b. What would be the major advantage of this scheme?
c. How could you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

9.5 Consider a page reference string for a process with a working set of m frames, initially all empty. The page reference string is of length p with n distinct page numbers in it. For any page replacement algorithm:
a. What is a lower bound on the number of page faults?
b. What is an upper bound on the number of page faults?

CHAPTER 11 I/O MANAGEMENT AND DISK SCHEDULING

11.1 For the following disk accesses, compute the number of head movements for the following list of seeks to disk cylinder: 26 37 100 14 88 33 99 12
Assume head is initially positioned over 26.

a. FCFS **b.** SSTF **c.** SCAN (going up) **d.** C-SCAN (going up)

11.2 Although DMA does not use the CPU, the maximum transfer rate is still limited. Consider reading a block from the disk. Name three factors that might ultimately limit the rate transfer.

11.3 Let us assume a disk with rotational speed of 15,000 rpm, 512 bytes per sector, 400 sectors per track and 1000 tracks on the disk, average seek time is 4ms. We want to transmit a file of size 1 MByte, which is stored contiguously on the disk.
a. What is the transfer time for this file?
b. What is the average access time for this file?
c. What is the rotational delay in this case?
d. What is the total time to read 1 sector?
e. What is the total time to read 1 track?

11.4 **Spooling** is an I/O technique developed in the early days of multiprogramming systems. It is a way of dealing with a dedicated I/O device. Each user process generates an I/O stream for a device that is directed to a disk file rather then to the device itself. The responsibility for moving data between the disk and the required device is delegated to a separate process, called a spooler. Spooling is appropriate for devices that cannot accept interleaved data streams, such as a printer.

Prefetching is a method of overlapping the I/O of a job with that job's own computation. The idea is simple. After a read operation completes and the job is about to start operating on the data, the input device is instructed to begin the next read immediately. The processor and input device are then both busy. With luck, by the time that the job is ready for the next data item, the input device will have finished reading that data item. The processor can then begin processing the newly read data, while the input device starts to read the following data. A similar idea can be used for output. In this case, the job creates data into a buffer until an output device can accept them.

Compare the prefetching scheme with the spooling scheme, where the processor overlaps the input of one job with the computation and output of other jobs.

11.5 Consider a disk system with 8 sectors per track and 512 bytes per sector. The disk rotates at 3000 rpm and has an average seek time of 15 msec. Also, consider a file consisting of 8 blocks. Compute the total time for accessing the entire file if the following allocation algorithms are used.

a. contiguous allocation,

b. indexed allocation

CHAPTER 12 FILE MANAGEMENT

12.1 A UNIX i-node has 13 direct pointers, 1 indirect pointer, 1 double indirect pointer and 1 triple indirect pointer. Assume that each 32-bit pointer identifies one block of 8KB. How large a file can the i-node handle?

12.2 Consider the organization of a UNIX file as represented by the I-Node. Assume there are 12 direct block pointers, and a singly, doubly, and triply indirect pointer in each I-Node. Further, assume that the system block size and the disk sector size are both 8K. If the disk block pointer is 32 bits, with 8 bits to identify the physical disk, and 24 bits to identify the physical block, then:

- **a.** What is the maximum file size supported by this system?
- **b.** What is the maximum file system partition supported by this system?
- **c.** Assuming no information other than the file I-Node is already in system memory, how many disk accesses are required to access the byte in position 13,423,956?

12.3 A sequential access file has fixed-size 150-byte records. Assuming the first record is record 1, the first byte of record 5 will be at what logical location?

12.4 A direct access file has fixed-size 150-byte records. Assuming the first record is record 1, the first byte of record 5 will be at what logical location?

12.5 On a system using 60-byte records and 500-byte blocks, how much space will be wasted in each block?

12.6 Consider a system using unspanned blocking and 100-byte blocks and a file containing records of 20, 50, 35, 70, 40, 20 bytes. What percentage of space will be wasted in the blocks allocated for the file?