Final Exam

Total: 120 points (including bonus 20 points).
(closed books and notes)

1. (10 pt) Consider the following snapshot of a system. P_0, P_1, P_2 and P_3 are processes and A and B are resources.

Available	
A	B
1	4

	Max		Allocation		Need	
	A	B	A	B	A	B
P_0	8	5	0	1	8	4
P_1	3	2	2	0	1	2
P_2	10	4	4	1	6	3
P_3	3	2	3	1	0	1

Answer the following questions using the banker's algorithm:

(a) Is the system in a safe state? (explain it, don't just answer 'yes' or 'no'.)

(b) If a request from process P_1 arrives for (1, 1), can the request be immediately granted? (explain it, don't just answer 'yes' or 'no'.)

(c) If a request from process P_0 arrives for (1, 2), can the request be immediately granted? (explain it, don't just answer 'yes' or 'no'.)

2. (6 pt) Consider the following snapshot of a system. P_0, P_1, P_2 and P_3 are processes and A and B are resources. Show that this system is not deadlocked now.

Available	
A	B
0	1

	Allocation		Request	
	A	B	A	B
P_0	1	0	0	1
P_1	2	0	4	2
P_2	1	0	3	2
P_3	2	1	1	1

3. (6 pt) (a) Describe four necessary conditions for deadlock.
(b) Describe deadlock prevention.
(c) Describe deadlock avoidance.

4. (6 pt) Consider a paging system with the page table stored in memory.
   (a) If a memory reference takes 80 nanoseconds, how long does a paged memory reference take?
   (b) If we add TLBs, and 90 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 3 nanoseconds, if the entry is there.)

5. (6 pt) Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 32 GB of physical memory. How many entries are there in each of the following?
   (a) A conventional single-level page table
   (b) An inverted page table

6. (6 pt) Compare the main memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:
   (a) external fragmentation
   (b) internal fragmentation
   (c) ability to share code across processes
   (d) allocation and deallocation
   (e) compaction

7. (6 pt) Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or if the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 60 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 180 nanoseconds?

8. (10 pt) Consider the following page reference string: 1, 2, 3, 1, 4, 2, 1, 5, 6, 2, 1, 3, 7, 6, 3, 2, 1, 2, 3, 6. How many page faults would occur for the following replacement algorithms, assuming four frames? Remember that all frames are initially empty, so your first unique pages will cost one page fault each.
   a. LRU replacement
   b. FIFO replacement
   c. Optimal replacement

9. (6 pt) (a) What is the cause of thrashing?
   (b) Calculate successive working sets for the following reference string, with a working set window (Δ) of 5.
   1 2 6 1 2 5 4 7 5 1 6 2 3 4 1 2 3 4 3 4 3 4 1 3 2 3 7 4 3 4 5 6

10. (10 pt) Suppose that a disk drive has 4,500 cylinders, numbered 0 to 4,499. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is:

    2069, 1212, 2296, 2800, 544, 1618, 336, 1523, 4475, 3681

    Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

    a. FCFS
    b. SSTF
    c. SCAN
    d. LOOK
    e. C-SCAN
    f. C-LOOK

11. (6 pt) Consider a mirrored drive system. Supposed that the failures of the two drives are independent. If the mean time between failures (MTBF)of a single drive is 120,000 hours and the mean time to repair is 6 hours, what is the mean time to data loss?

12. (6 pt) (a) Draw a figure to describe synchronous I/O.
    (b) Draw a figure to describe asynchronous I/O.

13. (6 pt) In the current working directory, there is a file ‘intro.ps’ shown in the following:
    -rw-rw-r-- 1 pbg staff 31200 Sep 3 08:30 intro.ps
    Suppose that user ‘Jim’ is not in the group ‘staff’. Can user ‘Jim’ write to file ‘intro.ps’?

14. (6 pt) Consider a file system that uses inodes to represent files. Disk blocks are 2-KB in size and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, plus single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

15. (6 pt) Consider a file system similar to the one used by UNIX with indexed allocation. How many data block I/O operations and inode I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the data blocks and inodes is currently being cached.

16. (6 pt) (a) Draw a figure with in-memory file system structures to explain ‘File open’ operation.
    (b) Draw a figure with in-memory file system structures to explain ‘File read’ operation.

17. (12 pt) After executing the program shown below, what are the output on the screen and the contents of files "output1" and "output2"? Assume that file descriptors 0, 1 and 2 are for standard input, standard output and standard error in the beginning of this process, respectively.
(Hint: The dup(oldfd) system call creates a copy of the file descriptor oldfd, using the lowest-numbered unused file descriptor for the new descriptor.)
(Hint: The close(fd) system call closes the file descriptor fd.)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
main()
{
	int pid, fd1, fd2;
	write(1, "*demo*1\n", 8);
	fd1 = dup(1);
	write(1, "*demo*2\n", 8);
	close(1);
	open("./output2", O_CREAT|O_WRONLY|O_TRUNC, 0666);
	write(1, "*demo*3\n", 8);
	fd2 = dup(1);
	write(1, "*demo*4\n", 8);
	close(1);
	open("./output1", O_CREAT|O_WRONLY|O_TRUNC, 0666);
	write(1, "*demo*5\n", 8);
	if ((pid = fork()) != 0) {
		wait();
		write(1, "--parent--\n", 11);
	} else {
		write(1, "--child--\n", 10);
	}
	write(fd1, "*demo*6\n", 8);
	write(fd2, "*demo*7\n", 8);
	write(1, "*demo*8\n", 8);
	close(1);
	dup(fd1);
	write(1, "*demo*9\n", 8);
	close(1);
	dup(fd2);
	write(1, "*demo*10\n", 9);
}
```