COMP3911 Suggested Exercises #2a

These suggested exercises are not to be handed in for marking, but should be used as a study aid. Solutions will be posted on the course website.

I suggest that you not look at the solutions until after you have attempted to solve the problems. **Please note that Suggested Exercises #2 supplements Suggested Exercises #1, it does not replace it.** The final will cover the entire course, not just material discussed after the midterm.

1) Referring to a SCSI disk without a cache that has:

- an average seek time of 8msec
- 7,200 RPM
- 480 sectors/track
- 10 tracks/cylinder
- 512 bytes/sector

a) What is the average time it takes to do a read operation of 16Kbytes from this drive and how many of these operations can be performed/second?
b) What is the average time it takes to do a read operation of 64Kbytes from this drive and how many of these operations can be performed/second?
c) What is the I/O bandwidth (Kbytes/sec) for both a) and b)?
d) How much time is saved per I/O operation, if the drive is replaced with one that is identical, except that it spins at 10,000RPM?
e) Does a read operation on this disk take longer than a write operation?

2) Briefly explain why file system writing performance will be very different from file system reading performance?

3) For the following I/O operations on the files, determine what data, if any, is returned and what changes, if any, occur to the file system? (Include all changes, including ones that will occur sometime later if delayed writing is used.) Do each case from the file system in the Diagram. (In other words, for v), do not use the modified file system created by iv).)

a) Using ABC.DAT in the FAT file system attached as the second-last page
b) Using the file represented by i-node #5 in the Unix like file system attached as the last page

i) Read 55 bytes at offset 1010
ii) Read 200 bytes at offset 4000
iii) Read 2000 bytes at offset 4400
iv) Write 100 bytes of "xxxxx..." at offset 5060
v) Write 200 bytes of "yyyyy..." at offset 6080

4) For a Unix-like file system that uses cylinder groups in its disk layout in an effort to keep the average seek distance small, how will the data blocks of a large file be allocated?

5) Here are three examples of file system block size layouts:

i) 16,384 byte block size
ii) 16,384 byte block size, with the last block being divided into 8 fragments that can be used by up to 8 different files
iii) 1024 byte block size

a) Which of the above designs will waste the least amount of disk space?

b) Which of the above will achieve the fastest average write performance?

c) Explain why increasing the size of a file could take longer for ii) than i)?

6) Suppose that you had a Unix-like (indexed) file system with a block size of 8192 bytes, where the first 8 blocks are referenced directly in the i-node and the rest are referenced by a single indirect block holding the next 2048 block numbers.

a) What is the maximum size of file that can be supported by this file system?

b) If nothing is in memory at the start of a read for this file, how many disk read operations will be required to read 1000 bytes at a file offset of 65,000?

c) to f) are related to the following operation on the file:
Suppose the indirect block number in the i-node is NULL and then 500 bytes is written at an offset of 67000.

c) How many blocks must be allocated to the file from the free list for this write?

d) What size will the file be after the write completes?

e) What will the data value of file byte 66600 be?

f) What will be the value of the first block# in the indirect block?

7) When a file is deleted on a FAT type file system, is the file data no longer on the disk?

8) Write the C code, using the semaphore synchronization primitive along with the semaphore operations WAIT_SEM() and SIGNAL_SEM(), for the following variant of the Producer/Consumer problem. (Use as many semaphores and WAIT_SEM, SIGNAL_SEM operations as you need.) In this case, the queue has requests for disk drives numbered 0->3 in it, with the drive # in the request entry. As such, the **get()** function, must get a request for the drive specified as an argument. Assume that there are several processes making disk I/O requests and calling **put()**, along with four processes calling **get()** to get requests for the corresponding disk drives.

Write three variations of the solution that implement the following three queuing disciplines, respectively:

a) FIFO
b) Shortest Seek distance first (ie. select closest request next)
c) SCAN (similar to b), but goes in the one direction until there are no more requests then reverses direction (direction refers to increasing cylinder number or decreasing cylinder number)

```
#define     QSIZE     10
struct diskreq {
        int     drive_number;
        int     cylinder;
        int     head;
        int     sector;
        char    *buffer_address;
        int     read_write_flag;
};
struct diskreq que[QSIZE];
int start_pos = 0, end_pos = 0;

/*
 * This function gets the next request for drive_num and
 * returns the request by filling in the fields of next_req,
 * that is passed in as an argument.
 */
void get(int drive_num, struct diskreq *next_req)
{
. . .
}

/*
 * This function puts the I/O request given in the structure
 * passed in as an argument (by reference) as req into the que.
 */
void put(struct diskreq *req)
{
. . .
}
```

9) You are managing a machine that is running as a dedicated web server. There have been some complaints that the web server is slow when under heavy load and the boss is concerned since the load on the web server is expected to increase in the near future.

You run "vmstat 5" and "uptime" repeatedly on the machine when it is under a typical heavy load period and observe the following:

- On average the CPU is idle 20% of the time
- The machine always has a lot of free memory pages
- The SCSI disk is doing about 10 I/O operations/second.

Given the above, determine which of the following machine upgrades to recommend and how much additional load it will handle, giving the same or better performance as compared to the current system?

- A CPU that is twice as fast
- The addition of 256Mbytes more memory
- The replacement of the SCSI disk with one that has a 10,000RPM spindle instead of the current one with a 7,200RPM spindle.

Hints: Remember that Po (the proportion of time a queue is empty) is equal 1 - ρ, which is the load factor and equal to λ / μ. For the disk, you need to estimate the average time taken for an I/O operation in order to estimate how busy the disk is and what effect the faster spindle will have. In other words, for the disk, you will have to figure out how many I/O operations/second that it can handle and use that to estimate what effect the change will have on the server's load handling capacity. (A typical current generation SCSI disk has an average seek time of about 10msec.) In general, the busiest device will saturate first, and is typically referred to as the bottleneck device. This device will end up with the long queue and will be the predominate factor in overall performance. (The only exception to this is when the system is balanced such that the queue lengths remain about the same length for all queues.)

10) The following page reference string was collected by a hardware monitor for a program running on an Acme computer system. (The Acme system is known to support a pure demand paging virtual memory subsystem.)

0x7 0x6 ↓ 0x7 0x8 0x9 ↓ 0x7 0x8 0x7 0x8 0xa 0x8 ↓ 0x8 0xa 0xb 0xb ↓ 0xc 0x8 0x8 0xc 0xa ↓ 0xc 0xa 0x7

The ↓ arrows indicate where the operating system cleared the Use bit on all frames in memory.

For a fixed page frame allocation of 3, simulate each of the following page replacement algorithms and count the number of page faults that occurs:

- FIFO
- LRU
- Belady's optimal replacement
- Use Bit (falling back to FIFO)

11) What are the tradeoffs between using a small vs large page size in paged virtual memory systems? How do these relate to the tradeoffs between small versus large block-size in a filesystem?

12) If a machine that supports paged virtual memory is observed to be doing a lot of swapping, what change(s) should be made to the machine, in order to improve performance and why?

13) For the following I/O operations on the files, determine what data, if any, is returned and what changes, if any, occur to the file system? (Include all changes, including ones that will occur sometime later if delayed writing is used.)

a) Using ABC.DAT in the FAT file system in diagram #1
b) Using the file represented by i-node #5 in the Unix like file system in diagram #2

i) Read 55 bytes at offset 1010
ii) Read 200 bytes at offset 4000
iii) Read 2000 bytes at offset 4400
iv) Write 100 bytes of "xxxxx..." at offset 5060
v) Write 200 bytes of "yyyyy..." at offset 6080

14) Write a C code sequence that would create a data packet with the following fields in it for transmission across the internet. (Ensure that the data is tightly packed and that it is some known architectural format. For bit efficiency, the numeric data should not be transmitted as ascii data.)

- A 16 bit integer in a variable of type short called "num"
- A 32 bit integer in a variable of type long called "value"
- An ascii string with a maximum of 9 characters in a character array called "name"

15) If an RPC service is built directly on top of UDP/IP transport, what does the client have to do so that it will never have to wait forever for a reply?

16) In a multi-processing operating system running on a multiprocessor, we use semaphores inside of the disk subsystem. Why do we need these semaphores?

17) In the filesystem, there is an option to the `open()` call which is described as follows:

`O_EXCL`

When used with O_CREAT, if the file already exists it is an error and the open will fail.

a) Describe how this flag can be used to create a syncronization primitive, based around a lock file (a lock file is a file placed in a directory to indicate which process currently is writing to a common resource, where only one process is permitted to write to this common resource at a time). Note that the flag `O_CREAT` will create a file.

b) The man page goes on to describe this functionality as "broken on NFS file systems". Why would this be much harder to implement on NFS than on local file systems?

18) Outline the list of steps taken when writing data to the disk. For each example, assume that you need to write out 16k of data. Start by assuming that you have just opened an existing file of 30k, and that none of the file data is in the cache.

a) Where the user level writes are 1024 bytes, and the disk blocksize is 4096 bytes.

b) Where the user level writes are 4096 bytes, and the disk blocksize is 1024 bytes.

b) Where the user level writes are 1 byte each, and the disk blocksize is 512 bytes.

d) Some of these operations are arguably more "efficient" than others (ie; they get to avoid some of the steps). Which ones are more "efficient"? What makes them more efficient?

19) Some versions of telnet allow you to execute "line-at-a-time" mode, where an entire line of text (up to a carriage return) is sent. Regular telnet sends each character separately. Why would someone implement "line-at-a-time"?

20) What premise does the RAID-1 (mirroring) work on in order to recover data? What premise does RAID-4 work on? Why does RAID-4 require fewer disks than RAID-1?

In the text, a description is given for RAID 0+1, which requires a huge number of disks. What advantage does this system have over other RAID setups which use fewer disks?