

UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER 2003 EXAMINATIONS

CSC 369H1F

Duration - 3 hours

Examination aids allowed: one 8.5 x 11" (letter-sized), double-sided "fact sheet" from the student.

Name: _____

Student #: _____

Notes to students:

1. There are 11 questions and 17 pages in total (including this cover sheet) for this exam. There are 141 points possible (for the 180 minutes available). Budget your time appropriately. FYI, this is approximately 47 points per hour.
3. Answer all questions directly on the examination paper. Generally, the space allowed is a clue to the size of answer expected. Use the final page of the exam, or the backs of pages if more space is needed, and provide clear pointers to your work.
4. Write your name and student number at the top of each page.
4. In general, show your intermediate work and BRIEFLY state your assumptions where appropriate.
5. Write clearly and legibly. If we can't read your answer, we can't grade it.
6. Read the questions carefully and answer the question that is being asked.

| Question | Marks | Question | Marks |
|---------------|-------|----------|-------------|
| 1 | /28 | 6 | /10 |
| 2 | /16 | 7 | /10 |
| 3 | /15 | 8 | /8 |
| 4 | /6 | 9 | /10 |
| 5 | /6 | 10 | /12 |
| | | 11 | /20 |
| Total: | | | /141 |

1. Ultra-Short Answer [28 points, 2 each]

Answer with a single term or phrase, or circle the appropriate option for each of the following.

- (a) What is the primary hardware feature that allows operating system kernels to be protected from user programs?
- (b) For a multi-threaded process, indicate whether each of the following items should be part of the process descriptor (P), the thread descriptor (T), or both (B):
- | | |
|------------------------------|-----------|
| a. Stack pointer | P / T / B |
| b. Priority | P / T / B |
| c. Page table | P / T / B |
| d. Resource usage accounting | P / T / B |
- (c) What is one major problem with using atomic hardware instructions for synchronization?
- (d) What key property of concurrent transactions is guaranteed by the two-phase locking protocol?
- (e) An unsafe state ALWAYS/SOMETIMES/NEVER leads to deadlock.
- (f) True or False: The Windows XP (or NT) CPU scheduler uses feedback to adjust the priority of threads in the real-time class.
- (g) What are the two components of a segment table entry in a system using pure segmentation?

This question continues on
the following page

Name: _____
Student #: _____

- (h) A TLB miss ALWAYS/SOMETIMES/NEVER results in a page-in from disk.
- (i) What is the primary motivation for two-level page tables?
- (j) In general, are there MORE/SAME/FEWER inodes than directories in a file system such as Unix?
- (k) Disk scheduling is MORE/SAME/LESS likely to be useful on a multi-user server than on a desktop PC.
- (l) What name is given to the distributed computing design used by NFS and AFS?
- (m) If you think _____ can solve your security problem, then you don't understand <same blank> and you don't understand your problem either. (attributed to Bruce Schneier)
- (n) You have an unsigned integer which has been initialized with a memory address (i.e., "unsigned int x = 0x4001fff0"). Write correct C code, including the necessary casting, to initialize an integer pointer variable with the contents of that memory address.

```
int *y =
```

Name: _____
Student #: _____

2. Short Answer [16 points, 4 each]

i) What is the difference between a capability and an access control list?

ii) Give two reasons why a programmer needs to be aware that a remote procedure call (RPC) is different from an ordinary procedure call.

iii) The principle of locality is often exploited by operating systems to improve performance. Give one example involving *spatial locality* and one example involving *temporal locality*.

iv) Why is it better for disk scheduling to be performed by the disk controller itself, rather than by the operating system?

3. Synchronization [15 points]

A tennis club has two courts that are available for “pick-up” doubles games. The rules for using these courts are as follows: When players arrive at the club, they must wait until a group of four players can be formed. Once four players are ready, one of them must reserve a court. If both courts are in use, they must wait for a court to become available before beginning their match. When the match is over, the player who reserved the court must let others know that it is available.

Using Posix mutexes (locks), condition variables, and semaphores, implement the tennis court scheduling policy described above (**Hint: use exactly one of each**). You do not need to worry about how the players know which of the two courts they are allowed to use. The code below provides the framework for your solution. On the following page, fill in the functions **readyToPlay()** and **donePlaying()**. Also identify any global variables that you need, and add any initialization code to the **main()** function.

```
int total_players = N;

void* tennis_player(void *arg) {
    int my_id = (int)arg;
    readyToPlay(my_id); /* wait for players and court */
    playMatch(my_id);
    donePlaying(my_id); /* release court if necessary */
    pthread_exit((void *)0);
}

main() {
    pthread_t t;

    for(int i=0; i < total_players; i++) {
        pthread_create(&t, NULL, tennis_player, i);
    }
}
```

To refresh your memory, the functions for semaphores, mutexes, and conditions that you can use are:

```
sem_wait(sem_t *sem)
sem_post(sem_t *sem)
```

```
pthread_mutex_lock(pthread_mutex_t *mutex)
pthread_mutex_unlock(pthread_mutex_t *mutex)
```

```
pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)
pthread_cond_signal(pthread_cond_t *cond)
pthread_cond_broadcast(pthread_cond_t *cond)
```

This question continues on
the following page

Name: _____
Student #: _____

```
/* Add global variables here */
```

```
void readyToPlay(int my_id) {
```

```
}
```

```
void donePlaying(int my_id) {
```

```
}
```

```
void main() { /* Add initialization code here */
```

```
}
```

Name: _____
Student #: _____

4. Policy vs. Mechanism [6 points]

Discuss the difference between policy and mechanism in operating systems, using CPU scheduling as an example.

5. Distributed File Systems [6 points]

“AFS was designed for scalability.” Define what is meant by scalability in this context, and discuss two features of AFS that improve scalability over the NFS design.

Name: _____
Student #: _____

6. Virtual Memory Part I [10 points; 2 each]

A system has a 32-bit virtual address space and uses a two-level page table. 9 bits are used to index the top-level page table, and 11 bits are used to index the second-level page table.

i) How large are the pages (in bytes)?

ii) How many *virtual* pages can the process have?

iii) What is the size (in bytes) of an entry in the second-level page table, assuming the second-level page table is stored in a single physical page frame?

iv) If each page table entry includes valid, reference, and dirty bits in addition to the page frame number, what is the maximum number of physical page frames in the system?

v) Assuming the page size is fixed, how can the address translation scheme be altered to support more physical page frames?

7. Virtual Memory Part II [10 points]

In Assignment 3, your implementation of *malloc* and *free* was evaluated on the maximum size of the heap after handling a series of requests, and on the time to handle those requests. Someone argues that saving space is unnecessary since it is only virtual addresses that are being wasted – the same amount of physical space is needed to store the allocated blocks, regardless of where they are located in the virtual address space. They claim that time is the only criteria that matters, and that the best time performance could be achieved by the following strategy:

- expand the heap to its maximum size initially using the *sbrk* system call
- use a “next fit” allocation policy, splitting blocks off the start of the heap until the end of the heap is reached
- only coalesce free blocks when the “next fit” pointer returns to the start of the heap

Since the maximum size of the heap is extremely large, most processes will never need any coalescing, and will never have to search the free list for a block that is large enough; in addition, only a single system call is needed to expand the heap.

Using your knowledge of how virtual memory works, explain two (2) reasons why the proposed strategy is likely to result in *lower* overall performance for a program that uses dynamic memory allocation, than a strategy that tries to minimize the size of the heap.

Name: _____
Student #: _____

8. Computer Security [8 points]

Explain briefly how a *buffer overflow* attack works and describe how an operating system could defend against them. Identify what hardware support (if any) would be required.

Name: _____
Student #: _____

9. Disk I/O [10 points]

The original Unix file system reserved space for inodes at the beginning of the disk, with the remaining disk sectors used to store data blocks.

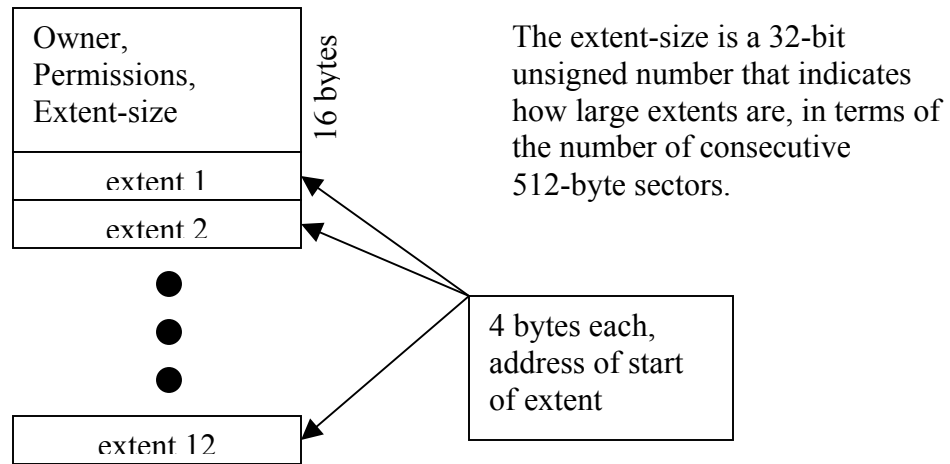
i) [2 pts] Identify one major problem with this strategy.

ii) [2pts] Describe briefly how FFS deals with this problem.

iii) [6pts] Suppose space for inodes is reserved from the middle cylinders of the disk instead of at the beginning. **(a) State** how this strategy can improve performance over the original Unix file system layout, and **(b) State one advantage and one disadvantage** of this strategy as compared to the FFS approach.

10. File Representation [12 points, 3 each]

Consider a file system with the following file representation in the inode:



Files are created with an initial extent-size of 1 (that is, each extent is a single 512-byte sector). Whenever the file grows larger than the maximum file size supported by the current extent-size, the extent-size is doubled.

i) What is the maximum file size supported by this system? Express your answer as $n \cdot 2^m$.

ii) Assuming that the inode is already in memory, how many disk accesses are required to read the byte at offset 1,000,000 in a file? Briefly explain your answer.

This question continues on the following page

Name: _____
Student #: _____

iii) Identify the free space representation you would prefer to use with this file system, and explain why.

iv) What operations are required when the file grows larger than the maximum file size supported by the current extent-size (in addition to doubling the extent-size field in the inode)?

11. Performance [20 points; 4 each]

Alice and Bob are working on optimizing a program to analyze a very large data file (many times larger than the size of physical memory on their system). The existing program repeatedly reads a single block of data from the file, analyzes it, and stores the results into a buffer, until all the data has been processed. The results are then summarized, formatted, and written to a new disk file. Their problem has the property that each block of data can be analyzed independently of the others (in any order) without changing the results. Their hardware platform will be a uniprocessor with a single secondary storage device (a fixed magnetic disk).

Alice proposes that the program should be multi-threaded – each thread will repeatedly read a single block of data from the file system, analyze it, and store the results into a shared buffer, until all the data has been processed. The last thread to finish will summarize and output the results. Alice argues that even though only one thread can use the CPU at a time, her plan will allow one thread to compute while others wait for their data blocks to be read, resulting in an overall program speedup.

i) Explain one reason why Alice’s plan may lead to an overall program *slowdown* rather than a speedup.

Measurements of the original program have produced information about the worst-case and average time to read a block of data from the file system.

ii) Assuming that the time to analyze a single data block is *larger* than the *worst-case* time to read a block from the disk, how many threads should Alice use? Explain briefly.

| |
|---|
| This question continues on the following page |
|---|

Name: _____
Student #: _____

iii) Assuming that the time to analyze a single data block is *smaller* than the *average* time to read a block from disk, should Alice use more threads, fewer threads, or the same number of threads as in (ii)? Explain your answer.

Bob agrees with Alice's logic, but knows that correctly synchronizing multithreaded programs is hard. He also remembers that disk performance improves with larger transfer sizes. Bob argues that the program should simply be modified to read a large number of data blocks from the file at one time, thus reducing the total I/O time.

iv) Explain one reason why Bob's plan may not be substantially faster than the original program.

Name: _____
Student #: _____

v) Alice and Bob can't agree on which proposal to implement so they have come to you for a decision. Make a choice and explain your reasons to them (and the grader!). Your argument must include additional factors; do not just re-state the points that Alice and Bob have already made. Identify any additional assumptions you make.

Name: _____
Student #: _____

Extra Space – Use if needed. Indicate clearly what questions you are answering here.

End of Exam
Total Pages = (17)
Total Marks = (141)