Fall 2002 15-712 homework #0
— or —
Swapping in the state from undergraduate OS*

This homework assignment is meant to quickly and thoroughly restore your existing understanding of undergraduate OS course material. It may require review on your part, which is an appropriate action for the first week of class.

Turn it in at the start of class on September 16. Your answers must be typed (not handwritten).

1. *Definitions.* Define the following terms:
   (a) Internal and external fragmentation. Also, which of them can occur in a paging system? A system with pure segmentation?
   (b) Translation look-aside buffer (TLB).
   (c) Interrupt.
   (d) Distributed shared memory.
   (e) Stateful and stateless servers. Also, list two file systems, one that is stateful and one that is stateless, and explain how having or not having state affects file service.
   (f) Swapping.
   (g) Inverted page table.
   (h) Disk sector, track, and cylinder.
   (i) Thrashing.
2. *Short answer.* Provide a short answer to each of the following questions:
   (a) Give a reason why virtual memory address translation is useful even if the total size of virtual memory (summed over all programs) is guaranteed to be smaller than physical memory.
   (b) Compare and contrast access control lists and capabilities.
   (c) The length of the time slice is a parameter to round robin CPU scheduling. What is the main problem that occurs if this length is too long? Too short?
   (d) List an advantage and a disadvantage to increasing the virtual memory page size.
3. *Virtual memory addressing.* Suppose we have a machine that uses a three-level page table system. A 32-bit virtual address is divided into four fields of widths a, b, c, and d bits from left to right. The first three are indices into the three levels of page table; the fourth, d, is the offset. Express the number of virtual pages available as a function of a, b, c, and d.
4. *Page replacement.* Suppose a machine with 4 physical pages starts running a program (in other words, the physical pages are initially empty). The program references the sequence of virtual pages as follows:

   `A B C D E D C B A E D C B A C E`

*Questions taken from past undergraduate operating systems final examinations given at Berkeley and CMU.

For each of the following paging algorithms, replicate the reference pattern and underline each reference that causes a page fault (or make references that cause a page fault uppercase, and those that don't lowercase):

(a) LRU.

(b) FIFO.

(c) Optimum.

5. *Multiprocessing.* Suppose you have a large source program containing m files that you want to compile. You have a cluster of n "shared-nothing" workstations, where $n > m$, on which you may compile your files. At best you will get an m-fold speedup compared to a single processor. List at least three reasons as to why the actual speedup might be less than this.

6. *Achieving fast file reads.*

   (a) Give at least three strategies that a file system can employ to reduce the time a program spends waiting for data reads to complete.

   (b) For each strategy you listed, describe a read pattern for which the strategy would do well, and one for which the strategy would do poorly.

7. *Synchronization.* Your OS has a set of queues, each of which is protected by a lock. To enqueue or dequeue an item, a thread must hold the lock associated with the queue.

   You need to implement an *atomic transfer* routine that dequeues an item from one queue and enqueues it on another. The transfer must appear to occur atomically.

   This is your first attempt:

```
void transfer(Queue *queue1, Queue *queue2)
{
    Item thing; /* the thing being transferred */

    queue1->lock.Acquire();
    thing = queue1->Dequeue();
    if(thing != NULL) {
        queue2->lock.Acquire();
        queue2->Enqueue(thing);
        queue2->lock.Release();
    }
    queue1->lock.Release();
}
```

   You may assume that `queue1` and `queue2` never refer to the same queue. Also, assume that you have a function `Queue::Address()` which takes a queue and returns, as an unsigned integer, its address.

   (a) Explain how using this implementation of `transfer()` can lead to deadlock.

   (b) Write a modified version of `transfer()` that avoids deadlock and does the transfer atomically.

   (c) If the transfer does not need to be atomic, how might you change your solution to achieve a higher degree of concurrency? Justify why your modification increases concurrency.

8. *Networking.* You are developing a network protocol for the reliable delivery of fixed-sized messages over unreliable networks. You are using a sequence number in each message to allow the receiver to eliminate duplicates, but you still have three design alternatives to consider.

   The design alternatives are: (1) the sender must receive an acknowledgement for the previously sent message before it can send the next message in the sequence, (2) the sender can transmit up to n

unacknowledged messages, but the receiver will discard any messages that are received out of sequence (in other words, it will only acknowledge a message if it is received in sequence), and (3) the sender can transmit up to n unacknowledged messages, and the receiver will acknowledge each on receipt, even if they arrive out of order.

For each alternative, answer the following questions:

(a) Explain what state the receiver must keep around to implement each of the three alternatives (remember, the receiver must be able to detect and discard duplicates).

(b) Suppose two machines are communicating across a bi-directional network link with fixed-size messages. Acknowledgement packets are also fixed-size. One machine, the sender, is transmitting a very large file to the other machine, the receiver. That is, the sender is transmitting a large number of messages that make up this file. Discuss briefly how each alternative's data bandwidth is expected to vary given different underlying network characteristics.

9. *Serving multiple clients.* There are two main approaches to organizing a server daemon, such as a web server:

   (a) Create a new kernel thread for each client (for each web browser connection);

   (b) Use a single process responding to all clients, usually based on the `select()` system call.

   Compare and contrast these two approaches.