

ECE469 - Operating Systems Engineering

Exam #1

2004 March 25

“A computer lets you make more mistakes faster than any invention in human history with the possible exceptions of handguns and tequila.”

– Mitch Ratliffe, Technology Review, April, 1992

Instructions: Read carefully through the entire exam first, and plan your time accordingly. Note the relative weights of each segment as a percentage of the total exam score.

This exam is **closed book, closed notes**. You may *not* refer to any books or other materials during the exam.

Write your answers on this exam. You may use both sides of the page.

When you are done, present your completed exam and your student identification to the instructor or proctors at the head table. If leaving before the exam period is concluded, please leave as quietly as possible as a courtesy to your neighbors.

Name:

Student Number:

Signature:

1. (Semaphores - 15%)

- (a) (3%) Briefly, what are the two major kinds of semaphores, and what kind of synchronization is each used for? (Using the precise technical terms is a plus.)

Binary semaphores can be only 1 or 0, and are often used for mutual exclusion.

Counting semaphores can have non-binary values, and are often used for condition synchronization.

- (b) (5%) Semaphores are generally implemented in software. However, hardware support for synchronization primitives can make semaphore implementation easier and more efficient.

Recall the `TestAndSet` instruction available on many multiprocessor architectures. `TestAndSet` can be thought of as atomically performing the code below:

```
boolean TestAndSet(boolean *target)
{
    boolean rv = *target;
    *target = true;
    return rv;
}
```

Show how `TestAndSet` can be used to implement the semaphore primitives `P()` and `V()` for spinlocks on a multiprocessor system with shared memory. Assume a shared boolean, `lock`.

```

P(S)                                V(S)
{                                    {
    while (1)                        while (TestAndSet(&lock));
    {                                S++;
    while(TestAndSet(&lock));
    if (S <= 0)                       lock = false;
        { lock = false; }            }
    else
        {
            S--;
            lock = false;
            break;
        }
    }
}
}
```

- (c) (7%) In pseudo-code below, outline how to use the P() and V() primitives to solve the Producer/Consumer problem with a pool of empty (**EmptyPool**) buffers and full (**FullPool**) buffers. Assume there are N empty buffers initially, and no full buffers. Be careful to indicate which semaphores you are using where; note which kind each semaphore is (from your answer above), and the value it is initialized with.

```
Producer:                               Consumer:
while(1)                                 while(1)
{                                         {
    // Put producer                       // Put consumer
    // pseudocode here.                   // pseudocode here.

    P(empties)                            P(fulls)
    P(mutex)                              P(mutex)
    get empty buffer                       get full from pool of fulls
        from pool of empties
    V(mutex)                              V(mutex)
    produce data in buffer                 consume data in buffer
    P(mutex)                              P(mutex)
    put full buffer                        put empty buffer
        in pool of fulls                  in pool of empties
    V(mutex)                              V(mutex)
    V(fulls)                              V(empties)
}                                         }
```

empties is for condition synchronization, initialized to n .
fulls is for condition synchronization, initialized to 0.
mutex is for mutual exclusion, initialized to 1.

2. (Scheduling - 12%)

- (a) (3%) What are the relative advantages and disadvantages of STCF (Shortest Time to Completion First) process scheduling?

Advantage:

- STCF is provably optimal if we know the time to completion.

Disadvantages:

- STCF works well for offline scheduling, but requires that we accurately approximate the remaining runtime of a job in order to be used for general purpose scheduling.
- Can suffer from starvation.

- (b) (3%) What are the relative advantages and disadvantages of priority scheduling?

Advantages:

- Fairness
- Flexibility

Disadvantage:

- Starvation

- (c) (3%) Describe a technique for addressing the primary disadvantage of priority scheduling.

Aging: If process waits too long, increase priority.

- (d) (3%) One of the advantages of a lottery scheduling algorithm with dynamic priorities is that it provides fairness, preventing starvation. How does a lottery scheduler *reward* well-behaved processes, and why?

The lottery scheduler awards more tickets to processes that voluntarily block or yield before their quanta expires, thereby increasing the likelihood that I/O bound and other well-behaved processes run more frequently.

3. (Deadlock - 18%)

(a) (4%) List the four conditions required for deadlock to occur. Briefly explain each.

- Mutual Exclusion: a resource assigned to exactly one process
- Hold and Wait: multiple independent requests
- No preemption: can't revoke resources
- Circular chain of requests

(b) (6%) Consider the three processes below, (proc1 through proc3,) each of which competes for six shared resources, (A through F):

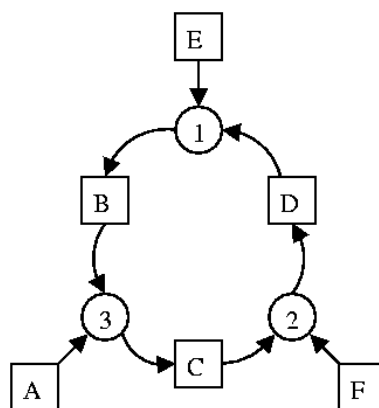
```

proc1()
{
    while (1)
    {
        lock(&D);
        lock(&E);
        lock(&B);
        // Use D, E,
        // and B
        unlock(&D);
        unlock(&E);
        unlock(&B);
    }
}

proc2()
{
    while (1)
    {
        lock(&C);
        lock(&F);
        lock(&D);
        // Use C, F,
        // and D
        unlock(&C);
        unlock(&F);
        unlock(&D);
    }
}

proc3()
{
    while (1)
    {
        lock(&A);
        lock(&B);
        lock(&C);
        // Use A, B,
        // and C
        unlock(&A);
        unlock(&B);
        unlock(&C);
    }
}
    
```

Could the three processes enter into deadlock? If so draw the resource allocation graph with the cycle that represents this deadlock. (Remember: Processes are circles, resources rectangles.)



- (c) (4%) One of the deadlock handling schemes discussed in class was called, “Detect and Recover”. How can the O/S detect deadlock? How can it recover? Note any major disadvantages to this scheme.

Detection requires only maintaining a resource graph and scanning for cycles, (which can be done in $O(n^2)$.)

Recovery requires killing offender processes or rolling back the actions of a deadlocked process to a safe state. Killing arbitrary processes detracts from the reliability of the system, and rolling back processes could involve a prohibitive amount of accounting.

- (d) (4%) Another of the deadlock handling schemes in class was labeled, “Prevention”; we discussed how each of the four conditions required for deadlock could be singled out for prevention. Pick one of the four conditions from your answer to part (a), and describe the major trade-offs prevention of this condition entails.

- Mutual Exclusion: Make resources sharable.
 - Reduces chances of deadlock.
 - Some resources are inherently unsharable, (i.e. printers,) so this scheme can at best reduce opportunities for deadlock, not eliminate.
- Hold and Wait: Require processes to request all of the resources they require upfront, so they never wait on another resource.
 - Eliminates Deadlock.
 - Reduces overall utilization of system. Starvation possible.
- No Preemption: Allow O/S to preempt (take back) resources from processes.
 - Eliminate deadlock.
 - Cannot soundly be applied to resources for which we cannot save state.
- Circular Wait: Require processes to request resources only in ascending order for some specified ordering of resources.
 - Eliminate Deadlock.
 - Low utilization. How to enforce?

4. (Dynamic Allocation Problem - 10%)

(a) (3%) What is the Dynamic Allocation Problem?

How do we satisfy a request of size n from a list of holes.

(b) (3%) What is the difference between internal fragmentation and external fragmentation? In the presence of random, general purpose traffic, what is the rule of thumb for how much memory we expect to waste on external fragmentation? How much on internal fragmentation?

Internal fragmentation is allocated space wasted within a block. External fragmentation is unallocated space wasted between blocks. Statistical analysis of first-fit reveals the *50% Rule*, which predicts that we'll lose $0.5 * N$ blocks to external fragmentation. We can expect to lose half a block on average to internal fragmentation per allocated chunk.

(c) (4%) Name and describe one of the three popular strategies used for the dynamic allocation problem. Explain its trade-offs.

- First Fit: Take the first hole in the list that fits.
 - Easy to implement; Fast.
 - Does not try to reduce fragmentation.
- Best Fit: Search entire list for smallest hole that is large enough to satisfy request.
 - Minimizes fragmentation.
 - Expensive to find best fit.
- Worst Fit: Search entire list for largest hole that can satisfy request.
 - Maximizes size of remaining holes.
 - Expensive to find worst fit.

5. (Paging and Segmentation - 16%)

A certain well-known 32-bit architecture uses segmentation to translate a *logical address* into a *linear address*. It then uses hierarchical, two-level, forward-mapped page tables to map the linear address down to a *physical address*. The processor has 16-bit segment selectors, and a segment descriptor register that points to the segment table. Page size is 4KB, and the first 10 bits of a linear address provide the offset into the top-level page directory, which is pointed to by a page directory base register.

- (a) (10%) Diagram the data structures involved in translating a logical address down to a final physical address. Label each table, and make clear how each step of translation takes place.

See Figure 9.21, page 311, in the textbook.

- (b) (3%) How many memory accesses does this lookup take in the worst case? What are they?

Four [or three]: 1) Segment lookup, 2) directory lookup, 3) page lookup, [and 4) memory reference in physical frame.]

- (c) (3%) How much memory is taken up by page directory and page tables for a full-sized process? (Do not include segment table size.)

$$\begin{aligned} & (2^{10} \text{ PDE's}) \times (4 \text{ bytes / PDE}) \\ & + (2^{10} \text{ PDE's}) \times (2^{10} \text{ PTE's / PDE}) \times (4 \text{ bytes / PTE}) \\ & = 2^{12} + 2^{22} \text{ bytes.} \end{aligned}$$

6. (More Paging - 17%)

- (a) (3%) How does a Translation Look-aside Buffer (TLB) improve the performance of paging?

By caching recent page table lookups, the system may save time by not having to go to main memory for subsequent lookups of the same page table entry (PTE).

- (b) (3%) What happens to the TLB with a context switch? (Careful: there are two very different answers here, depending on what information is stored in the TLB; specify what your assumptions are.)

- If the TLB does not store address space identifiers (ASID's) or process ID's, all TLB entries must be invalidated during context switch. We expect many subsequent memory references to be TLB misses.
- If ASID's are stored in the TLB, nothing special happens to the TLB during context switch; if the ASID of the new context does not match the ASID, we still expect that many subsequent memory references will be a TLB miss, but perhaps not as many as if all TLB entries has been invalidated..

- (c) (3%) What are the advantages and disadvantages of a system that can page out the user process page tables?

- More physical memory free for things other than page tables.
- Memory references in a process that has had its page tables paged out will generate multiple page faults – the first for the page table itself.

- (d) (8%) In class, we focused on paging mechanisms in which each process had its own page table, mapping virtual pages to physical frames. As a contrast to this mechanism, some systems (like the 64-bit UltraSparc and PowerPC,) use an *inverted page table*, a single system-wide table that maps physical frames to processes and their virtual pages. The inverted page table has one entry for each physical frame, and the entry contains an address-space identifier (ASID) and a virtual page number.

All virtual addresses in the system consist of an ASID, a page number, and an offset. If the memory system finds that the i^{th} entry in the inverted page table has a matching ASID and page number, the virtual address is translated to physical frame i plus the offset.

What do you suppose the greatest advantages and disadvantages of an inverted page table would be?

Single page table, takes up less space.

Searching inverted table very expensive; hashing can help, but this often entails specialized hashing hardware. Inverted table scheme makes sharing pages between address spaces more difficult.

7. (Page Replacement - 12%)

(a) (3%) Describe the FIFO page replacement algorithm.

Oldest page in queue (first page in) is the first to be replaced.

(b) (3%) What are the relative advantages and disadvantages of the FIFO page replacement algorithm?

- Fast and easy to implement.
- May replace heavily-used pages.

(c) (3%) Describe the FIFO with Second Chance algorithm.

All pages have a reference bit, set by hardware. When page reaches the end of the queue, if its reference bit is 0, it is chosen for replacement. If the page's reference bit is 1, the bit is cleared, and the page is given a second chance back at the head of the queue.

(d) (3%) What is the advantage of FIFO with Second Chance over the plain FIFO page replacement algorithm?

Heavily-used pages are not replaced easily.