

Sample Midterm Exam

1) Suppose that there are three processes, as described in the table below. Suppose that the operating system uses preemptive, round-robin scheduling, and that T11 is running when the quantum expires.

Process	Threads within the process
P1	T11, T12, T13
P2	T21, T22
P3	T31

(a) (5 points) If the threads are implemented entirely at the user level, which threads might possibly execute at the beginning of the next quantum?

T21, T22, or T31.

(b) (5 points) If threads are supported by the operating system, which threads might possibly execute at the beginning of the next quantum?

T12, T13, T21, T22, or T31.

(c) (15 points) Assuming these are kernel-level threads running on a multiprocessor with 4 CPUs with gang scheduling, describe an efficient assignment of threads to CPUs. What can the scheduler do when the quantum for T11 expires?

An efficient assignment would be running T11, T12, T13, and T31 at the same time. When the quantum for T11 expires, we could execute T21, T22, and T31 at the same time.

2) (20 points) Synchronization mechanisms are typically implemented using atomic operations provided by the underlying architecture; e.g., test-and-set or compare-and-swap. Consider an operation: fetch-and-increment (fai), which behaves as an atomic counter that is sampled, then incremented. It can also be initialized to 0. In other words, if you create a fai variable using `fai a`; you can issue two atomic operations, namely `int j = a++`; and `a=0`;. Your task is to use fai to implement a mutual exclusion lock. You need not guarantee fairness among multiple processes. Assume no fai variable can overflow.

```
fai inside = 0; // inside is an atomic counter
int a;
repeat
    a = inside++;
until (a == 0);
/* C.S. */
inside = 0;
```

3) (20 points) Linear page tables can become extremely large. Why is this a problem? Describe two page table organizations that address this problem. Compare these organizations to linear page tables.

A large linear page table is a problem because the entire table needs to reside in physical memory at all times. Two approaches: multi-level page tables and inverted page tables. Multi-level page tables allow the page table to be paged. The main problem with multi-level page tables is that they involve multiple memory accesses to complete a virtual/physical translation.

Inverted page tables use space that is proportional to the size of the physical memory. There are two main approaches to implementing IPTs: search-based and hash-based. Inverted page tables require less physical

memory than linear page tables, since their size is proportional to the size of physical memory rather than virtual memory. However, inverted page tables have their drawbacks. With search-based inverted page tables, searching can take very long and it is harder to map two different virtual pages to the same physical page frame. Hash-based do not have the search and mapping problems, but cause an extra memory reference (to the hash table), compared to a linear page table.

4) (a) (10 points) In the absence of future knowledge, the most desirable page replacement policy is one that chooses as a victim the page that has gone unused for the longest period of time. Why is this least-recently-used (LRU) policy most desirable? Also, explain why a pure implementation of LRU is currently not feasible.

LRU is desirable because it matches well with the behavior of applications. Pure LRU requires that the hardware keep track of the order in which pages are referenced. Real hardware doesn't do that (it's too expensive).

(b) (10 points) Assuming second-chance replacement and three memory frames, list the pages that are replaced for the following sequence of page accesses: 0, 1, 2, 3, 2, 4, 2, 5.

In this simple example, the pages that are replaced are 1, 0, and 3.

(c) (15 points) Compare replacement policies with local and global scopes.

Global-scope policies consider all physical frames in deciding which page to replace. Local-scope policies replace pages from the faulting processes. Global-scope policies are easier to implement, but may replace a page in the working set of a process. Local-scope policies are harder to implement because the OS periodically has to resize the resident sets of the processes.