Computer Science and Engineering Department
University of South Florida
Ph.D. Qualifiers Exam Spring 2008
Operating Systems
January 25

You have 3 hours to solve 10 problems, all equal weight. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

1. Your task is to develop a scheduler that is optimized to handle workloads containing many memory-intensive, fairly long-running non-interactive jobs.
    a. What information does the scheduler need from the virtual memory system to make the best decisions? Describe how the virtual memory can obtain, track, and record the needed information.
    b. What scheduling policy would you develop to optimize both throughput and fairness for this workload? Describe your algorithm in either sentences or pseudo-code.

2. Show how to emulate general (counting) semaphores safely using binary semaphores.

3. Sketch rough graphs (“back of napkin”) illustrating the relationships among the following quantities. Note any important assumptions underlying your analysis.
    a. Page fault rate as a function of page size. What page sizes are typical for current processors and operating systems?
    b. Disk access latency as a function of rotation speed.
    c. Peak disk read bandwidth and read latency as a function of read block size (request size).
    d. Peak read bandwidth, I/O operation throughput (IOPS), and access latency as a function of the number of drives in a RAID storage unit.
    e. CPU utilization as a function of the degree of multiprogramming. Consider a uni-processor system.
    f. Memory fragmentation as a function of page size. Is that internal or external?

4. Most modern processors and operating systems enforce protection boundaries that prevent programs from interfering with one another or with the operating system, and that allow the operating system to securely mediate and monitor all accesses to shared resources in accordance with a security policy. Briefly summarize the most important mechanisms underlying OS protection and security. Your answer should be comprehensive and precise, but succinct.

5. The original Unix file system allocated pages to files essentially randomly, so that the pages of a file would be scattered all over the disk. The result was that sequentially scanning (*reading*) a single large file was quite slow (only a few percent of the raw disk bandwidth). However, performance in a multi-user timesharing environment was acceptable, as was performance for sequentially *writing* (or creating) a file. Explain why.

6. Clocks in distributed systems:

    a. Why are clocks difficult to synchronize in distributed systems? What are the factors limiting the accuracy of the synchronization?
    b. What features in a system assume that the clocks are (reasonably) synchronized? Give two examples of cases where a problem would occur if the clocks were not synchronized.

7. What is the output of the following code and why?

```
#include <pthread.h>
#include <stdio.h>

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* the thread */

main(int argc, char *argv[])
{
      pthread_t tidl, tid; /* the thread identifier */
      pthread_attr_t attr; /* set of thread attributes */
      if (argc != 2) {
            fprintf(stderr, ''usage: a.out<integer value>\n'');
            exit();
      }
      if (atoi(argv[1] < 0) {
            fprintf(stderr, ''%d must be >= 0\n'', atoi(argv[1]));
            exit();
      }
      pthread_attr_init(&attr);
      /* create the threads*/
      pthread_create(&tid,&attr,runner,argv[1]);
      pthread_create(&tidl,&attr,runner,argv[1]);
      pthread_join(tid,NULL);
      pthread_join(tidl,NULL);
      printf(sum = %d\n,sum);
}

/* The thread will begin control in this function */
void *runner(void *param)
{
      int upper = atoi(param);
      int i;
      sum = 0;
      if (upper > 0) {
            for (i = 1; i <= upper; i++)
            sum += i;
      }
      pthread_exit(0);
}
```

8. Should an operating system for a 4 processor machine have a single kernel for process scheduling? Discuss the strengths vs. weaknesses of such an approach.

9. Present possible techniques implemented in the operating system for preventing the “fork bomb” (i.e., a process which infinitely forks child processes) and discuss their potential costs.

10. Name a new trend in operating systems research. List and discuss some of its challenges.