1 Rules and Hints for the Exam

This examination is open book and open note (your notes only please). Calculators are permitted. Networked devices are strictly prohibited. The questions are marked as to their relative value, the exam will be scored out of 100% but is worth 25 points towards your course grade. Relax and try to do what you can.

Here are some hints for doing well:

• You have about 80 minutes, so using 4 minutes per 5 percentage point of work is a good rate.

• You can do the problems out of order, just be sure to clearly label the problem numbers so that the grader can find the order.

• Please be neat (illegible answers cannot get credit).

• We want to see your thought process, show how you got to the result, otherwise it looks like guessing.

• Please put the following on each blue book cover your name (write this clearly please, some students have hard to read signatures), the page number and the total number of pages your grader should expect (in case the pages become detached). So for example if your name was James Taylor, then you would write: James Taylor 1 of 2.
2 The Problem Set

1. Extracurricular Knowledge (readings and miscellaneous stuff) (20 %):
   
   (a) How did the Linux operating system get its name (5 %)?
   
   (b) Give a brief summary of your favorite operating systems paper that was not assigned in the readings, be sure to mention the title, author, and publication details (which conference/journal and date)(10 %)?
   
   (c) Consider the file hoarding approach of Satyanarayanan and Kistler’s Coda file system. Would this approach work well for an environment where there are many small and seldom shared files or would it perform better in an environment where there are a few large and highly shared files (e.g. a large distributed database)? (5 %)

2. Fault Tolerance (20 %)
   
   (a) What is the difference between a Byzantine and Fail-stop fault model (5 %)?
   
   (b) Show that consensus cannot be established in a system with 3 nodes if one node has a single Byzantine failure (5 %).
   
   (c) Consider the system shown in Figure 1, where Lieutenants 1 and 3 are indirectly connected so that all messages between them must pass through lieutenant 2, and there is at most one (unidentified) traitor in the system.

   ![Figure 1: A Byzantine Generals Network for Problem 2c](image)

   - Can this system reach consensus using the oral messages (the OM) algorithm (that is messages are unsigned)? Why or why not (5 %)?
   - Can this system reach consensus when signed messages are used (5 %)?

3. Synchronization (30 %): Consider a bi-directional one-lane north-south bridge. Cars heading in the same direction can cross the bridge at the same time, but cars going in the opposing direction cannot (e.g. they must wait). After a car exits the bridge, the car is parked while the owner does some business, then the car drives back across the bridge in the other direction. Give a solution to this problem using semaphores. Do not worry about fairness and do not prefer any car. Be careful to both label variables to indicate if they are in shared memory and show initial values when appropriate.

4. Concurrency networking and threading (10 %) Suppose that a program has the following sequence of operations

   ```
   1 x = x + z;
   2 w = x * z;
   3 z = y;
   4 y = w;
   ```

   Listing 1: An Instruction Sequence

   Which Statements have data dependencies? Show the maximally parallel code using parbegin/parend notation. What is the maximal degree of parallelism in this problem (10 %).
5. Networking: What does the session layer do in the OSI/ISO model (5 %)?

6. Threading: Suppose that you are developing a multi-threaded operating system to run on a N-way simultaneous multi-threaded architecture (i.e. it has hardware support for managing N contexts, much like the 2-way hyperthreading used in the current Intel Xeon processors). Your manager has noticed some fine work from the IBM and the University of Washington that suggests that using the hardware performance measures of the number of instructions per cycle (IPC) is a good idea. Suppose that you have a function that returns the IPC of the specified thread, and efficient libraries for managing stacks, queues, priority queues, binary search trees and bitmaps. Your boss wants you to implement the IPC priority scheduler, which when there are M, M > N eligible jobs, your scheduler should make the N jobs with the largest IPC values resident on the processor. You can assume that you have the following routines and data structures.

```c
// reclaims the resources used by a terminated thread
void thread_reclaim(int thread_id);

// Should only be called by the scheduler, not by the thread
double get_ipc(int thread_id);

// Evicts the thread from the hardware scheduler
void thread_demote(int thread_id);

// Allocates a thread to the hardware scheduler
void thread_promote(thread_id);

// Returns the number of threads resident in the hardware scheduler
int resident_threads(int thread_id[], int thread_id_list_length);
```

Listing 2: Routines for IPC Scheduler

Newly arrived jobs should be treated as having the maximum IPC (to ensure that they get to run). You will need to implement:

```c
// Schedule's a newly runnable job
void schedule_job(int thread_id);

// Called by a job when it wants to relinquish the cpu
void thread_exit(int thread_id);

// Called by the scheduler when a job is removed from the CPU
// e.g. when it finishes or is preempted
void thread_remove_from_processor(int thread_id);
```

Listing 3: Routines for IPC Scheduler

Be careful not to leave the processor idle when there are runnable jobs nor to try have the processor manage more than N contexts concurrently (20 %).

7. Systems Programming: In our projects 1 and 2 we compiled audit trail logic directly into the kernel (by extending LTT). Another approach we could have tried was to compile in support for dynamic probes (dprobes). Dprobes are a facility supporting run time insertion of instrumentation into a running Linux kernel using system calls. The kernel invokes functions that the user specifies in a loadable kernel module (LKM). What are the security implications of using dprobes for audit trail creation (10 %)?