CS 414 Operating Systems
Final, Spring 2001
Emin Gün Sirer

Name:_____ NETID:_____

- **This is a closed book examination. It is four (4) pages, 2.5 hours long. Brevity is key.**
- **Show your work and/or reasoning for partial credit.**
- **You may use semaphores, monitors, and condition variables for any synchronization problem. You may use C, Java, assembler or low-level pseudo-code for any coding problem. Abstract, non-specific, and unclear descriptions, in particular, descriptions written mostly in English, will get no credit as responses to coding questions.**

[10] **1. [Synchronization]** A common idiom found in real systems is that of an "optimistic read without a lock," where a check for a special case is made without holding the corresponding lock. For example, the filesystem code shown below avoids locking overhead in the common case, where the system reference monitor has already been initialized:

```
struct ReferenceMonitor *rm;
sema_t referenceMonitorSema = sema_init(1);

void file_read(File *fp, char *buf, int i) {
     if(rm == NULL) { // line 5
          P(&referenceMonitorSema);
          if(rm == NULL) { // line 7
               rm = initializeSystemReferenceMonitor();
          }
          V(&referenceMonitorSema);
     }
     // ok, we're initialized, proceed as normal
     if(rm->checkAccess()) {
          // perform read
          …
     }
}
```

Describe why the check, "if(rm == NULL)", on line 7 is necessary given the check on line 5, and explain what could have happened if the check on line 7 had been left out.

[10] **2. [Synchronization]** Ping and pong are two separate threads executing their respective procedures. The code below is intended to cause them to forever take turns, alternately printing "ping" and "pong" to the screen. The `minithread_stop`() and `minithread_start` () routines operate as they do in your project implementations: `minithread_stop`() blocks the calling thread, and `minithread_start` () makes a specific thread runnable if that thread has previously been stopped, otherwise its behavior is unpredictable.

```
main() {
     pingthread = minithread_fork(ping,NULL);
     pongthread = minithread_fork(pong,NULL);
```

```
}


void ping()
{
      while(true) {
            minithread_stop();
            Printf("ping is here\n");
            minithread_start(pongthread);
      }
}
void pong()
{
      while(true) {
            Printf("pong is here\n");
            minithread_start(pingthread);
            minithread_stop();
      }
}
```

a. The code shown above exhibits a well-known synchronization flaw. Briefly outline a scenario in which this code would fail, and the outcome of that scenario.

b. Show how to fix the problem by replacing the `minithread_stop` and `start` calls with semaphore P and V operations.

c. Implement ping and pong correctly using a mutex and condition variables.

3. [Networking]

[6] **a.** What purpose do stubs serve in an RPC system ?

[7] **b.** Describe how the DNS protocol resolves the address www.google.com to an IP address. Assume that all machines involved have an empty DNS cache.

[12] **c.** Networks-R-Us needs your help with their DSR implementation. The pseudo-code below describes a router capable of propagating DSR packets. Assume that network links are unidirectional. Also assume that no DSR packets are destined or originated from the host running this code. Point out three problems with this code and suggest solution strategies:

```
      network_packet_received(Packet *packet) {
            Switch(packet->type) {
            case ROUTE_REQUEST:
                  // add my name
                  add_self_to_packet(packet);
                  // send to all my neighbors;
                  if(packet->ttl>0) minimsg_send(BROADCAST, packet);
                  break;
            case ROUTE_REPLY:
                  // same as a unicast send, fallthrough
            case UNICAST:
                  // find self in header and compute next hop
                  nexthop = findnexthop(packet->sourceroute);
                  packet->ttl = packet->ttl-1;
                  // forward packet to the next hop
                  if(packet->ttl>0) minimsg_send(nexthop, packet);
```

```
        }
    }
```

[6] **4. [Disk Management]** Given the request stream for cylinders: "35, 10, 82, 8" for a 100-cylinder (i.e. the innermost cylinder is 1, outermost 100) disk, calculate the seek-distance FCFS, SCAN and C-LOOK algorithms would traverse to process these disk accesses. Assume that the disk head is initially at cylinder 53.

[6] **5. [Filesystems]** A filesystem may choose to allocate data blocks for a file on disk contiguously (where all of the data blocks reside one after another), using a linked structure (where every block contains a pointer to the next block), or using an indexed structure (where a separate, inode-like structure contains links to every data block).
a. Name one advantage for each allocation strategy.
b. Name one disadvantage of each strategy.

6. [Filesystems]

[4] **a.** How many disk block accesses does it take to determine if the file /usr/egs/mail exists under a UFS file system. Assume that the host OS has just been rebooted, and that no other files have been read from that disk before. Also assume that no indirect inodes are involved. Show your reasoning for partial credit.

[1] **b.** What does an NFS server have to do following a reboot ?

[4] **c.** Name two things that an AFS server has to do when a client overwrites an existing file.

[8] **7. [RAID]**
a. Briefly describe RAID levels 1 (mirroring) and 5 (block interleaved distributed parity). Which one offers higher read performance ? Which one is cheaper for a fixed file storage capacity ?

b. Briefly describe RAID levels 2 (bit level striping with hamming code) and 3 (dedicated parity disk). Which one offers better fault tolerance ? Which one is cheaper ?

[6] **8. [Security]** Recall that Lamport's one-time password scheme requires using y_n = f(f(f(….f(x)))) as a one-time password for login attempt number n, where N is the maximum number of times the user is allowed to log in, x is the secret key, f is a one-way hash function, and y_n is the result of applying f recursively to x N-n times. Users submit y_n to the system to log in. Provide at least two reasons why Unix and NT use simpler password schemes instead of Lamport's one-time password scheme. (Hint: consider what the user has to do, and what would happen if the secret password file were compromised).

[10] **9. [Security]** Suppose you are building a mail server that enforces the Bell-La Padula model. Briefly describe the properties of this security model, and what you need to do to enforce it for people reading and sending mail messages.

[10] **10. [Protection & Mobile Code]** A friend of yours gives you the following code written in assembler and asks you to execute it as part of the network write path inside your protected-mode operating system:

```
// r1 and r2 can hold arbitrary addresses, all other
// registers are free for use
ld r3, 25(r1)    // load a word
add r3,1,r3      // add one
st r3, 37(r2)    // store
```

Your friend insists that it is a benign routine. More concretely, she promises that it will never read or write any memory location outside the range 0x1000-0x1500. You have checked that all kernel data in that range belongs to her processes and would like to run her code. Briefly describe how software fault isolation (SFI) can help you run this code safely inside your OS. Show the modified instruction stream after your SFI mechanism has processed this code.