UCLA Computer Science 111 (Winter 2017) Final
180 minutes total
Open book, open notes, closed computer

Name:_____ Student ID:_____

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	total

Problems 1 through 10 are multiple choice. For each such problem, list the single best answer, where the possibilities are (a) through (e). Also, for each such problem briefly explain either why (a) is not the best answer, or (if (a) is in fact the best answer) why (b) is not the best answer.

1 (8 minutes). In RAID:

a. Mirroring is generally better-performing than striping.

b. RAID-4 can survive up to 4 disk failures without losing data.

c. For random writes, RAID-0 and RAID-1 are typically better-performing than RAID-4 and RAID-5.

d. In RAID-4, if an application decides to write a block, the operating system must always first read at least one other block.

e. RAID-5 dominates RAID-4 so much that RAID-4 is almost never used nowadays.

2 (8 minutes). Some security questions:

a. If prime numbers were quite rare, algorithms like RSA that are based on factoring products of primes would be insecure.

b. Traditionally in Unix, the password file /etc/passwd was world-readable, so anybody could read anybody else's password and the system was insecure.

c. A larger trusted computing base helps to increase the security of a system.

d. Shared-secret cryptography is rarely used nowadays; almost all encryption now uses public keys.

e. Principals in GNU/Linux are uid_t values, which are integers.

3 (8 minutes). In NFS:

a. Each call to 'open' (without a corresponding 'close') corresponds to a distinct NFS file handle.

b. An NFS file handle's generation number helps avoid race conditions when a parent and child process close a shared file descriptor at about the same time.

c. The NFS model assumes a smart file server and a dumb client, rather than the reverse.

d. In practice, most NFS operations are idempotent; this keeps the protocol simple.

e. Although NFS clients use caches heavily, this does not introduce correctness problems since the caches are always validated before use.

4 (8 minutes).

a. On a multi-CPU machine, single- and multi-queue schedulers can both make effective use of cache affinity.

b. Because hardware is generally agnostic about which OS it should run, a GNU/Linux platform can host a macOS guest about as efficiently as it can host a GNU/Linux guest.

c. To gain flexibility, some virtual-memory systems let an application specify different page sizes for different arrays in the program.

d. In a multilevel page table, two page table entries at different levels cannot map to the same physical page, because the hardware prohibits this.

e. For best performance a virtual-memory manager should typically choose dirty pages as victims, as opposed to choosing clean pages.

5 (8 minutes).

a. The main practical problem with 'fsck' and similar programs is that they are too unreliable.

b. Log-structured filesystems are a mistake for flash devices, since their main goal is to avoid seek overhead for writes, and flash devices do not seek.

c. Checksums are ineffective for data integrity in filesystems, since a device that can lose or permute ordinary data can lose or permute a checksum too.

d. Unlike 'fdatasync', the 'fsync' system call can arrange for the contents of a symbolic link to be safely synchronized to secondary storage.

e. Although programs like 'git' maintain version histories, these histories are independent of the version histories used to implement filesystems, which means that running 'git' atop a log-structured filesystem will maintain two sets of histories.

6 (8 minutes).

a. Every filesystem has a structure called an inode, which stores the metadata for a file.

b. Although filesystems typically store user data in fixed-sized blocks, they store directory data in varying-length structures instead of blocks.

c. In a Unix-style filesystem, directory entries cannot cache any file metadata other than inode numbers, since metadata is kept in inodes and the cache might become invalid.

d. Security is a good reason why GNU/Linux lacks a system call openi(I, FLAGS) that would act like open(NAME, FLAGS) except with an inode number I as a parameter rather than a filename NAME.

e. Symbolic links are more flexible than hard links, because they allow links to cross directory boundaries, and allow links to files in a parent directory.

7 (8 minutes).

a. Hard links can be used to create two different directory entries in the same directory, i.e., entries that have the same name but different inodes.

b. Every Unix filesystem contains at least two directory entries, though they may both have the same inode number.

c. Link counts prevent cycles from being created in the directory structure.

e. Two different filesystems cannot contain the same inode number.

f. The last-modified time of a file cannot decrease; it can only increase or stay the same.

8 (8 minutes). In this question, assume that no files are being modified (perhaps by some other process) during the actions described, and that all system calls succeed.

a. If you call 'stat' with two different file names and 'stat' reports the same filesystem and inode number (st_dev and st_ino) both times, the link count (st_nlink) might also be 1 both times.

b. If you call 'stat' with two different file names and 'stat' reports the same st_dev and st_ino both times, the last-modified times (st_mtime) might disagree.

c. If you call 'stat' twice with the same file name, it might report differing st_dev, st_ino pairs.

d. The readdir system call is redundant, since you can find the directory entries individually by calling 'stat' on each of the directory's files. The main reason for readdir is efficiency, not functionality.

e. If you call readdir several times in succession on the same file descriptor, readdir could return the same file name more than once.

9 (8 minutes).

a. The plain elevator algorithm is fairer than the circular scan algorithm.

b. Anticipatory scheduling is more efficient than eager (work-conserving) scheduling.

c. Batching typically increases throughput and fairness when accessing blocks on a disk.

d. Flash-based storage devices are simpler to deal with than hard disks, because if the block size is kept constant then one can assume that all input and output operations have roughly the same cost.

e. GNU/Linux is not a good match for an RT-11-style file system, because when a GNU/Linux application creates a file it cannot easily communicate the file's desired size to the kernel.

10 (8 minutes).

a. Suppose a thread T owns a spin lock L. Then T must unlock L; that is, no other thread can unlock L.

b. Suppose a thread T has successfully called sem_wait on a semaphore S. Then T must call sem_post on S; that is, no other thread can call sem_post on S.

c. Because two different threads can simultaneously call sem_wait on the same semaphore successfully, race conditions can still occur even if semaphores are used, if they are not used carefully.

d. Suppose a thread T has successfully acquired a binary semaphore (blocking mutex) B. Then T must release B; that is, no other thread can release B.

e. It can make sense to have two condition variables for the exact same condition, to improve parallelism when many threads all depend on the same condition.

---END OF MULTIPLE-CHOICE QUESTIONS ---

11 (20 minutes). Would it make sense to use SSL/TLS within an operating system operating on a single multicore machine? For example, a CPU might use SSL/TLS when issuing commands to a flash drive. If so, explain the advantages of using SSL/TLS internally, and how you'd go about setting it up. If not, explain why not, give a more-conventional approach that would make more sense for security, and explain why.

12 (12 minutes). Suppose you execute the debugfs command 'clri 2'. What bad thing will happen to your ext2 file system? Is this something you can recover from, at least partially? If so, how; if not, why not?

13 (12 minutes). Suppose you have two programs S and M, both of which use a single lock to control access to a large shared array with A elements. The programs are identical except that S uses a spin lock and M uses a mutex. Both programs bottleneck with accesses to the array, so you decide to use finer-grained locking, and divide the array into N subarrays of size A/N, each subarray with its own independent lock. Which program, S or M, will benefit more from this change? If your answer depends on the sort of operations that the program is doing, explain the dependency.

14 (15 minutes). Suppose you want to extend GNU/Linux by having conditional symbolic links where the condition is an arbitrary user-defined program. For example, the symlink from 'a' to 'sort -c foo?b:c' would resolve to 'b' if the command 'sort -c foo' exited with status zero, and would resolve to 'c' otherwise. Describe the main challenges you see in implementing this extension.

15a (3 minutes). Why can't the following shell script be translated to simpsh format?

```
#!/bin/sh
sort | uniq
```

15b (6 minutes). Give a simple change to the spec for Lab 1 that would allow simpsh to execute the equivalent of the above script, in a relatively-natural way.

16. Suppose Intel and IBM announce a new processor, the Ozette Lake processor. It has an unusual feature: in user mode, it executes only unprivileged x86-64 instructions, but in kernel mode it executes only 64-bit z/Architecture instructions (either privileged or unprivileged). The z/Architecture ABI is quite different from the x86-64 ABI; among other things, it is a big-endian processor designed for mainframes. GNU/Linux has been ported to the z/Architecture, using a C compiler that generates z/Architecture instructions instead of x86-64 instructions.

16a (12 minutes). Would it be feasible to implement a Linux-based operating system on the Ozette Lake processor, without going to extreme lengths such as using a software-based emulator for the x86-64 or for the z/Architecture? If so, explain the most important problems you'd need to overcome to support such an OS; if not, explain why it would not be feasible.

16b (8 minutes). Assume that your answer to (a) is "yes", or that it's "no" and you're willing to put up with the performance problems of a software-based emulator. Explain the relationship between system calls and marshaling/unmarshaling in your operating system.

17 (12 minutes). Suppose we're worried about the power consumption of spin locks, in that the CPU chews up energy while spinning. Implement the lock and unlock functions for spinlocks that use the following primitives, while using less power than the implementation discussed in class when the code is spinning. The 'pause' primitive tells the CPU to wait for a bit and to remove speculative compare instructions from its pipeline; this uses less power.

```
void
pause(void)
{
  asm volatile("pause\n": : :"memory");
}

unsigned
xchgl(unsigned *ptr, unsigned x)
{
  asm volatile
    ("xchgl %0,%1"
     :"=r" ((unsigned) x)
     :"m" (*(volatile unsigned *) ptr), "0" (x)
     :"memory");
  return x;
}
```