CS 111: Operating System Principles
Lecture 23

Final Review

1.0.0

Jon Eyolfson
June 3, 2021

The Exam Format Will Have More Short Answer

The tiny questions didn't give enough freedom

I'm leaning towards none, or a very limited amount

The short answers give more room to ensure you've understood the content

Being able to clearly explain yourself is a very important skill!

Question 1 (1)

Consider a memory architecture using two-level paging for address translation. The format of the virtual address, physical address, and PTE (page table entry) are bellow:

- Virtual address: 9 bit (index), 9 bit (index), 14 bit (offset)
- Physical address: 10 bit (physical page number), 14 bit (offset)
- PTE: 10 bit (physical page number), 6 bit (permissions)

(a) What is the size of a page?

(b) What is the size of the maximum physical memory?

(c) What is the total memory needed for storing all page tables of a process that uses the entire physical memory?

Question 1 (2)

Consider a memory architecture using two-level paging for address translation. The format of the virtual address, physical address, and PTE (page table entry) are bellow:

- Virtual address: 9 bit (index), 9 bit (index), 14 bit (offset)
- Physical address: 10 bit (physical page number), 14 bit (offset)
- PTE: 10 bit (physical page number), 6 bit (permissions)

(d) Assume a process that is using 512KB of physical memory. What is the minimum number of page tables used by this process? What is the maximum number of page tables this process might use?

(e) Assume that instead of a two-level paging we use an inverted table for address translation. How many entries are in the inverted table of a process using 512KB of physical memory?

Question 2 (1)

Consider a set of 3 queues, and the following code that moves an item from a queue (denoted “source”) to another queue (denoted “destination”). Each queue can be both a source and a destination.

```
void AtomicMoveItem (Queue *source, Queue *destination) {
  Item thing; /* thing being transferred */
  if (source == destination) {
    return; // same queue; nothing to move
  }
  source->lock.Acquire();
  destination->lock.Acquire();
  thing = source->Dequeue();
  if (thing != NULL) {
    destination->Enqueue(thing);
  }
  destination->lock.Release();
  source->lock.Release();
}
```

Question 2 (2)

(a) Give an example involving no more than three queues illustrating a scenario in which AtomicMoveItem() does not work correctly.

(b) Modify AtomicMoveItem() to work correctly.

(c) Assume now that a queue can be either a source or a destination, but not both. Is AtomicMoveItem() working correctly in this case? Use no more than two sentences to explain why, or why not. If not, give a simple example illustrating a scenario in which AtomicMoveItem() (given at point (a)) does not work correctly.

Question 3

One of the innovations of the BSD file system is that it tries to allocate large files in long contiguous chunks. Assuming that a disk transfers at a peak rate of 200 MByte/s, and that a combined seek and rotation take, on average, a total of 20 milliseconds. What is the minimum size of each contiguous run of a large file, in order to achieve 80% of peak transfer rate for large files when they are accessed sequentially?

Question 4 (1)

We have seen different filesystems that support fairly large files. Now let's see just how large a file various types of filesystems can support. Assume, for all of the questions in this part, that filesystem blocks are 4 KiB.

i) Consider a really simple filesystem, directfs, where each inode only has 10 direct pointers, each of which can point to a single file block. Direct pointers are 32 bits in size (4 bytes). What is the maximum file size for directfs?

ii) Consider a filesystem, called extentfs, with a construct called an extent. Extents have a pointer (base address) and a length (in blocks). Assume the length field is 8 bits (1 byte). Assuming that an inode has exactly one extent. What is the maximum file size for extentfs?

Question 4 (2)

iii) Consider a filesystem that uses direct pointers, but also adds indirect pointers and double-indirect pointers. We call this filesystem, indirectfs. Specifically, an inode within indirectfs has 1 direct pointer, I indirect pointer, and 1 doubly-indirect pointer field. Pointers, as before, are 4 bytes (32 bits) in size. What is the maximum file size for indirectfs?

iv) Consider a compact file system, called compactfs, tries to save as much space as possible within the inode. Thus, to point to files, it stores only a single 32-bit pointer to the first block of the file. However, blocks within compactfs store 4,092 bytes of user data and a 32-bit next field (much like a linked list), and thus can point to a subsequent block (or to NULL, indicating there is no more data). How many blocks does a file of 10KB contain?

v) What is the maximum file size for compactfs (assuming no other restrictions on file sizes)?

Question 5

Consider a system with four processes P1, P2, P3, and P4, and two resources, R1, and R2, respectively. Each resource has two instances. Furthermore: - P1 allocates an instance of R2, and requests an instance of R1; - P2 allocates an instance of R1, and doesn't need any other resource; - P3 allocates an instance of R1 and requires an instance of R2; - P4 allocates an instance of R2, and doesn't need any other resource.

(a) Draw the resource allocation graph

(b) Is there a cycle in the graph? If yes name it.

(c) Is the system in deadlock? If yes, explain why. If not, give a possible sequence of executions after which every process completes.

Question 6 (1)

Professor Harry writes the following program for grading students' projects. Per-project grades are stored in a set of input files, and the following program's goal is to compute a final course grade for each student and write it to file name.grade.

```
main():
    remove all files ending with ".”grade
    for each student name s in alphabetical order:
        read assignment scores for student s
        calculate final grade filename = s + ".”grade
        fd = creat(filename)
        write(fd, final grade, ...)
        close(fd)
        printf”(finished with %s\”n, s)
```

Harry uses a laptop with a journaling file system in a mode that the journal contains both file content and the metadata. He runs the following command:

```
program | cat
```

Question 6 (2)

Harry sees “finished with x” for all students with names up through “p”, and then his laptop crashes. Bob reboots his laptop. Harry thinks he may have to re-run the program for some or all students. Explain what guarantees he has about which final grades will be on disk after the restart.

Question 7

Complete the put below using compare_and_swap so that concurrent invocations will run correctly without using locks

```
static void put(int key, int value) {
  struct entry *n, **p;
  struct entry *e = malloc(sizeof(struct entry));
  e->key = key;
  e->value = value;
  for (p = &table[key%NBUCKET], n = table[key % NBUCKET]; n != 0;
       p = &n->next, n = n->next) {
    if (n->key > key) {

    }
  }
  done: return;
}
```

Question 8 (1)

Consider the following implementation of a reader writer lock. A reader writer lock allows either multiple readers to have access to a critical section or a single writer.

```
struct rwlock {
  sem_t *sem;
  int readers;
  int writers;
};

void rwlock_init(struct rwlock *lock) {
  sem_init(&lock->sem, 1);
  lock->readers = 0;
  lock->writers = 0;
}
```

Question 8 (2)

```
void readlock(struct rwlock *lock) {
  while(1) {
    sem_wait(lock->sem);
    if(lock->writers == 0) { lock->readers++; break; }
    sem_post(lock->sem);
  }
}

void writelock(struct rwlock *lock) {
  while(1) {
    sem_wait(lock->sem);
    if(lock->readers == 0 && lock->writers == 0) { lock->writers = 1; break; }
    sem_post(lock->sem);
  }
}
```

Question 8 (3)

```
void unlock(struct rwlock *lock) {
  sem_wait(lock->sem);
  if(lock->readers > 0) lock->readers--;
  else lock->writers--;
  sem_post(lock->sem);
}
```

(a) What is the problem with this implementation?

(b) Starvation is a problem where one thread is unable to acquire a resource. After fixing the problem, is starvation possible? How?

Thank you!