CSCI UA.202-001: Operating Systems Final Fall 2021

Name: Xi Liu
NYU Net ID: xl3504

Instructions

- Go to the File menu and select “Make a Copy” to produce a copy of this document.
- Take the “begin” selfies
- Take the exam; fill out the answers in your copy.
- When done, go to File -> Download -> PDF Document and download a copy of the document as a PDF.
- Take the “end” selfies
- Go to Gradescope (https://www.gradescope.com/courses/302159/assignments/1729575) and submit the PDF as your midterm.
- Go to Box (https://nyu.app.box.com/f/9feb5b8baee242f9a7fa665f4d24e117) and upload your selfies

Name: **NYU Net ID:**

I C and Assembly

1. Write `find_leastsupperbound()` in syntactically valid C, following the requirements.

```
int find_leastupperbound(int* array, int length, int num, int* err) {

      for(int i = 0; i < length; i++)
      {
            if(array[i] > num)
            {
                  return array[i];
            }
      }
      *err = 1;
      return 0;
}
```

2. After this excerpt executes, what values are in each of the registers? Fill in the table below. Leave hexadecimal values in hexadecimal.

register	**value**
`%rdi`	1
`%rax`	2
`%rbx`	5
`%rcx`	0xff
`%rbp`	0xff0000
`%r8`	0xff0000
`%r9`	0xff
`%r10`	5
`%r11`	2
`%r12`	1

II Concurrent programming

3. Where indicated, fill in the variables and methods for the `Auction` object. Remember to follow the concurrency commandments.

```
class Auction {

    public:
        Auction();
        ~Auction();
        void RegisterBid(uint32_t amount);
        uint32_t SelectWinner();

    private:
        uint32_t bids[NUM_BIDS]; // bids[i] == 0 means that slot i is free

        // ADD MORE HERE
         cond all_full;
         mutex m;
        int n_bid;

};

Auction::Auction()
{
    memset(bids, 0, sizeof(bids));

    // FILL THIS IN
    cond_initialize(&all_full);
    mutex_initialize(&m);
    n_bid = 0;







}
```

```
void
Auction:RegisterBid(uint32_t amount)
{
    // We use 0 to mean "slot free"
    assert(amount > 0);

    // FILL THIS IN
      acquire(&m);
    while(n_bid == NUM_BIDS)
      {//no free slot
            cond_wait(&m, &all_full);
      }
      for(int i = 0; i < NUM_BIDS; i++)
      {
            if(bids[i] == 0)
            {
                  bids[i] = amount;
                  n_bid++;
                  signal(&m, &all_full);
                  break;
            }
      }
      release(&m);

}


uint32_t
Auction:SelectWinner()
{
    // FILL THIS IN
acquire(&m);
while(n_bid < NUM_BIDS)
{
      cond_wait(&m, &all_full);
}
int max = bids[0];
for(int i = 1; i < NUM_BIDS; i++)
{
      if(bids[i] > max)
      {max = bids[i];}
}
for(int i = 0; i < NUM_BIDS; i++)
{
     bids[i] = 0;
}
release(&m);
return max;





}
```

Page for code

Page for code

4. What did you do to avoid deadlock? Or, if you did not get this aspect of the assignment correct, what *should* you have done to avoid deadlock? Limit your answer to three sentences, but be precise.

```
sort(item_ids->begin(), item_ids->end()); //enforce lock ordering to avoid deadlocks
```

Because deadlock may happen when thread1 tries to acquire lock1 and lock2, and at the same time thread2 tries to acquire lock2 and lock1, then thread1 might wait for lock2 to be released and thread2 wait for lock1 to be released.

III Scheduling

5. Write down the process scheduled for each epoch:

0	1	2	3	4	5	6	7	8	9
P1	P2	P3	P3	P3	P2	P1	P1	P1	P1

IV Lab 4, virtual memory

6. Describe the code line by line, following the instructions.
9: set the kernel_pagetable variable to the address of 0th element of kernel_pagetables array, %cr3 will contain the physical address of the root of the page table
10: clear or set every cell of kernel_pagetable to be 0
11: put the address of L1 page table into the entry 0 of L0 page table
13: put the address of L2 page table into the entry 0 of L1 page table
15: put the address of L3 page table into the entry 0 of L2 page table
17: put the address of L4 page table into the entry 1 of L2 page table
20: virtual_memory_map function maps the virtual address in the range of [0, 0 + MEMSIZE_PHYSICAL] to physical address in the range of [0, 0 + MEMSIZE_PHYSICAL]

Page for answer continuation

V I/O

7. What is the storage capacity of the disk in bytes or gigabytes? Explain *briefly* (for example by showing your work).

(2^11 B / sector)(2^10(2^10 + 1)/2   sum of sectors of all tracks)
approx= 2^11 * 2^20 / 2 approx= 2^30 bytes

If the two-disk system is at capacity (that is, storing the maximum amount of data), what is the minimum time in seconds to read all logical sectors into memory? (Assume that there is enough RAM to store all of the data.) Explain briefly for example by showing your work.

sum of sectors of all tracks = 2^10(2^10 + 1)/2 approx= 2^19

Minimum time = (10.5 + 12 + 16)/3 * 2^19

Page for answer continuation

8. How many I/O requests from this process can be pending at the OS at once? Explain your answer in no more than two sentences.

8 I/O requests

V File systems, lab 5, and crash recovery

9. What is the maximum size of the entire `fs202` file system, in bytes (or megabytes, gigabytes, terabytes, petabytes, etc.)? Show your work.

16KB = (2^4)(2^10)B = 2^14 B
Maximum size = 8(16KB) = 2^3(2^14)B = 2^17 B

Based on your answer above, what is the absolute minimum length of a block pointer on this file system? Give your answer in bits.

17 bits

10. Is mkdir(`d`) idempotent, for some fixed value of `d`? Why or why not? Use no more than two sentences.

Not idempotent, because the system does not allow two directories to have the same name. Only the directory that first have the name can be successfully created

11. Assume that a crash can happen at any time. Does your friend's proposal work? If so, argue that it is correct. If not, explain why not. Use no more than four sentences.

The proposal does not work. To maintain crash consistency, the on-disk modifications must be done atomically, all done or nothing done. The file system has inconsistency when the on-disk modifications are partially completed

12. Fill in `firstbyte_fileblock1033()`, in syntactically valid C.

```
char firstbyte_fileblock1033(struct inode *ino)
{
   // FILL THIS IN.
   //   - Do not use inode_block_walk or inode_get_block.
   //   - Assume that file block 1033 is allocated.
   //   - It’s fine to specialize your code to file block 1033.

      int filebno = 1033;
      uint32_t * ptr = (uint32_t *)diskaddr(ino->i_indirect);
      int remain = filebno - N_DIRECT;
      return (char)ptr[remain];

}
```

VI Whole system, security, feedback

13. State (a) *which entity* processes the binary and ensures that the `.rodata` section of the binary becomes a read-only section of program memory, and (b) *what* that entity does, specifically, to make that section of program memory read-only. Write no more than two sentences.

a. The operating system
b. The operating system set the read-write permission bit to be read-only, so that whenever it checks if that part of program memory is not writable, any write attempts cause page faults

14. Does your friend's solution work? Why or why not? Write no more than two sentences.

The friend's solution does not work because ROP (return address oriented programming).

15. Please state the topic or topics in this class that have been least clear to you.

Crash consistency

Please state the topic or topics in this class that have been most clear to you.

Page tables