CS-GY 6233 Midterm Exam

Jia Chen

TOTAL POINTS

85 / 108

QUESTION 1

1 1 (a) **0 / 4**

- **0 pts** Correct

✓ **- 4 pts** **Incorrect**

- **0.5 pts** Calculation Error

QUESTION 2

2 1 (b) **0 / 4**

- **0 pts** Correct

✓ **- 4 pts** **Incorrect**

QUESTION 3

3 1 (c) **1 / 2**

- **0 pts** Correct

✓ **- 1 pts** **Partial Correct**

- **2 pts** Incorrect

QUESTION 4

4 2 (a) **1 / 2**

- **0 pts** Correct

✓ **- 1 pts** **Incorrect**

- **2 pts** Blank

QUESTION 5

5 2 (b) **4 / 5**

- **0 pts** Correct

- **1 pts** Missing or Incorrect explanation of +4

✓ **- 1 pts** **Missing or Incorrect explanation of *4**

- **2 pts** Incorrect

- **5 pts** Blank

QUESTION 6

6 2 (c) **3 / 3**

✓ **- 0 pts** **Correct**

- **3 pts** Incorrect

- **3 pts** Blank

QUESTION 7

7 3 (a) **5 / 5**

✓ **- 0 pts** **Correct**

- **5 pts** Incorrect

- **2 pts** Job submission time should be 0, 0, 0, 0, 3, 5, 5

- **2 pts** Job E F not correct

- **3 pts** Order does matter

- **1 pts** Calculation Error

QUESTION 8

8 3 (b) **5 / 5**

✓ **- 0 pts** **Correct**

- **2 pts** Submission time

- **5 pts** Incorrect

- **0.5 pts** Calculation Error

QUESTION 9

9 4 (a) **5 / 5**

✓ **- 0 pts** **Correct**

- **4 pts** Incorrect

- **2 pts** Didn't show 4 processes

- **2 pts** Missing page directory

- **5 pts** Blank

- **1 pts** Incorrect calculation

QUESTION 10

10 4 (b) **5 / 5**

✓ **- 0 pts** **Correct**

- **4 pts** Incorrect

- **2 pts** Didn't show 4 processes

- **5 pts** Blank

QUESTION 11

11 5 (a) **6 / 6**

+ **6 pts** Correct

+ **0 pts** Wrong

✓ **+ 2 pts** **Description is correct.**

✓ **+ 4 pts** **Code is correct**

+ **4 pts** Description is correct, part of the code is correct

QUESTION 12

12 5 (b) **4 / 4**

✓ **+ 4 pts** **Correct**

+ **3 pts** Mostly correct

+ **2 pts** Ambiguous

+ **1 pts** Mostly wrong

+ **0 pts** Wrong

QUESTION 13

13 6 (a) **0 / 4**

+ **4 pts** Correct

+ **3 pts** round robin

+ **1 pts** fair

✓ **+ 0 pts** **Wrong**

QUESTION 14

14 6 (b) **4 / 4**

✓ **+ 4 pts** **Correct**

+ **2 pts** Ambiguous

+ **0 pts** Wrong

QUESTION 15

15 6 (c) **2 / 2**

✓ **+ 2 pts** **Correct(Context switching and register)**

+ **1 pts** Ambiguous

+ **0 pts** Wrong

QUESTION 16

16 7 (a) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Incorrect

QUESTION 17

17 7 (b) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** InCorrect

QUESTION 18

18 7 (c) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 19

19 7 (d) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 20

20 7 (e) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 21

21 7 (f) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 22

22 7 (g) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 23

23 7 (h) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 24

24 8 (a) **0 / 2**

- **0 pts** Correct

✓ **- 2 pts** **Click here to replace this description.**

QUESTION 25

25 8 (b) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Click here to replace this description.

QUESTION 26

26 8 (c) **2 / 2**

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 27

27 8 (d) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 28

28 8 (e) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 29

29 8 (f) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 30

30 8 (g) 0 / 2

- 0 pts Correct

✓ - 2 pts Click here to replace this description.

QUESTION 31

31 8 (h) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 32

32 9 (a) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 33

33 9 (b) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 34

34 9 (c) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 35

35 9 (d) 2 / 2

✓ - 0 pts Correct

- 2 pts Click here to replace this description.

QUESTION 36

36 9 (e) 0 / 2

- 0 pts Correct

✓ - 2 pts Click here to replace this description.

QUESTION 37

37 9 (f) 0 / 2

- 0 pts Correct

✓ - 2 pts Click here to replace this description.

- 1 pts 1

QUESTION 38

38 9 (g) 2 / 2

✓ - 0 pts Correct

QUESTION 39

39 9 (h) 2 / 2

✓ - 0 pts Correct

Name Jia Chen NetID jc8773

Name: Jia Chen

NetID: jc8773

CS6233 Midterm

October 31, 2018

Answer the questions in the spaces provided in the question sheets. If you run out of room for an answer, you can continue on the back of the page.

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	16	
8	16	
9	16	
Total	108	

Name Jia Chen NetID jc8773

1. Consider the following C program:

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
      int i = 0;
      for (int i = 0; i <= 4; i++) {
            if (i % 2 == 0) {
                  fork();
            }
            printf("foo\n");
      }
      return 0;
}
```

0. 2. 4

(a) (4 Points) How many times will "foo" be printed?

"foo" will be printed 5 times ~~20~~ 20

(b) (4 Points) How many processes will be created (including the initial process)?

Creat 4 processes. 8

(c) (2 Points) If `fork()` can create processes, why do we still have `exec()`?

exec() is used to jump to another command or function.
with some parameters.
fork() ~~is~~ to create process.
their functions are different.

Name Jiachen NetID jc8773

2. The dup() function will create a copy of the file descriptor. The following is the code that implements the dup() system call in xv6:

```
1  int
2  fetchint(uint addr, int *ip)
3  {
4    if(addr >= proc->sz || addr+4 > proc->sz)
5      return -1;
6    *ip = *(int*)(addr);
7    return 0;
8  }
9
10 int
11 argint(int n, int *ip)
12 {
13   return fetchint(proc->tf->esp + 4 + 4*n, ip);
14 }
15
16 static int
17 argfd(int n, int *pfd, struct file **pf)
18 {
19   int fd;
20   struct file *f;
21
22   if(argint(n, &fd) < 0)
23     return -1;
24   if(fd < 0 || fd >= NOFILE || (f=proc->ofile[fd]) == 0)
25     return -1;
26   if(pfd)
27     *pfd = fd;
28   if(pf)
29     *pf = f;
30   return 0;
31 }
32
33 int
34 sys_dup(void)
35 {
36   struct file *f;
37   int fd;
38
39   if(argfd(0, 0, &f) < 0)
40     return -1;
41   if((fd=fdalloc(f)) < 0)
42     return -1;
```

Name Jia Chen NetID jc8773

```
43    filedup(f);
44    return fd;
45 }
```

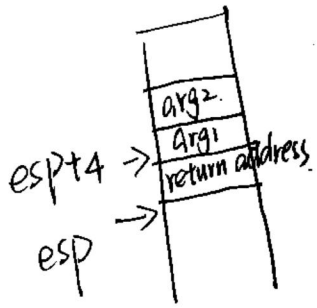
(a) (2 Points) On line 39, why do we have to use `argfd(int n, int *pfd, struct file **pf)`, instead of directly passing the value to `fd` to `sys_dup`?

~~Because first we need to find the file descriptor of file.~~

~~Becuse first we need to check if the file is valid. Maybe the file's file descr is out of the boundary of process.~~

Because we need to check if the file is valid.
Specifically, is the file in the boundary of process stack.

(b) (5 Points) In the `argint` function, why do we add `4 + 4*n` to `proc->tf->esp`?



esp is the stack pointer.

We use proc->tf->esp+4+4*n to go to user-mode stack to get the argument we want.

Name Jia Chen NetID jc8773

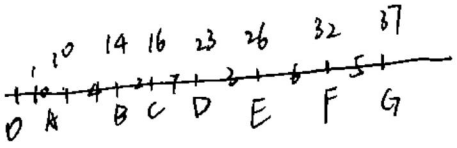
(c) (3 Points) What could go wrong if xv6 didn't have the check at line 4 of `fetchint` function?

Maybe they gonna go to other ~~process's~~ processes' stack. And cause damage to their data.

Name Jia Chen NetID jc8773

3. Four batch jobs, A through D, arrive at a computer center at the same time, They have estimated running times of 10, 4, 2, 7 minutes respectively. At time 3, job E arrives, which takes 3 minutes. At time 5, jobs F and G arrive, which take 6 and 5 minutes respectively. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

(a) (5 Points) First-come, first-served (run in order of sequence: A, B, C, D, E, F, G).



For A: 10 min

B: 10 + 4 = 14 min

C: 14 + 2 = 16 min

D: 16 + 7 = 23 min

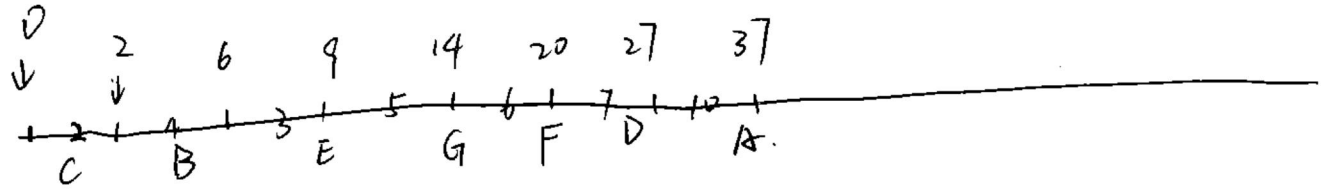
E: ~~23+3=~~ 23 − 3 + 3 = ~~20 min~~ 23 min

F: 26 − 5 + 6 = 27 min

G: 32 + 5 − 5 = 32 min

$$\frac{10+14+16+23+23+27+32}{7} = 20.\$ \text{ min}$$

(b) (5 Points) Shortest job first.



C: 2 min

B: 6 min

E: 9 − 3 = 6 min

G: 14 − 5 = 9 min

F: 20 − 5 = 15 min

D: 27 min

A: 37 min

$$\frac{2+6+6+9+15+27+37}{7} = 14.6 \text{ min}$$

Name Jia Chen NetID jc8773

4. Recall that in 32-bit x86, page directories and page tables are each made up of 1024 32-bit entries. Suppose we have 4 processes on a system, each of which has every possible virtual address mapped.

(a)(5 points) How much memory is used to store the page directories and page tables if 4KB pages are used?

For each process:

page directories: 4kB

page tables: $\frac{2^{32}}{4KB} \times 4 = 4MB$

Total: $4 \times (4kB + 4MB) \approx 16\ MB$

(b)(5 points) If 4MB pages (superpages) are used, then the entries in the page directory point directly to the page frame (i.e., no second-level page tables are used). How much memory would be taken up by page directories in this case?

For each process:

$$\frac{2^{32}}{4MB} \times 4 = 4kB$$

Total: $4 \times 4kB = 16kB$

Name Jia Chen NetID jc8773

5. (a)(2 + 4 points) What is a zombie process? Write a program that creates a Zombie.

Zombie process is a process whose ~~who's exit~~ parents doesn't call wait() to catch the exit of this process. And Zombie process exit. But the entry still exists in the process table. Because parents doesn't delete it.

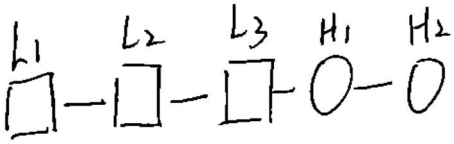
```
int main() {
    if(fork() > 0){
        sleep(5); // let the child exit before parent.
    }
    exit(1);

    return 0;
}
```

Name Jia Chen NetID jc8773

(b)(4 points) Explain (with diagram and a paragraph) how a priority round-robin scheduler works when the system has processes of two distinct priorities -- low priority and high priority. Work through an example with 2 high priority processes and 3 low priority processes.

O : high priority process

□ : low priority process.

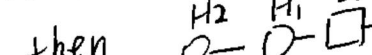


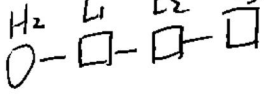
~~First, let H1 run a time slice. while it uses up slice or yield.~~

~~then choose H2 run a time slice. while it uses up slice or yield.~~

~~let L1 run a time slice until no time left or L1 yield~~

~~let L2 run a time slice until no time left or L2 yield~~

First. H1 run a time slice. ① If no time left and H1 still unfinished/ then  H1 is added to list again and ~~in the~~ according to the priority. It will stands before L1. L2 and L3.

② If H1 finished or yield. H1 is not ~~run~~ ready any more  then we run H2 for a time slice.

So. Priority ~~de~~ weights most. And if Priority is equal/ And run according to the original sequence. New process or just ~~fro~~ runned process will be add to the end (but not finish) of sequence.

Name Jia Chen NetID jc8773

6. Below is the code for the scheduler function in xv6, which picks the next process to run. Refer to it as you answer the following questions:

```
1 // Per -CPU process scheduler.
2 // Each CPU calls scheduler () after setting itself up.
3 // Scheduler never returns . It loops , doing :
4 // - choose a process to run
5 // - swtch to start running that process
6 // - eventually that process transfers control
7 // via swtch back to the scheduler .
8 void
9 scheduler ( void )
10 {
11     struct proc *p;
12
13     for (;;){
14           // Enable interrupts on this processor .
15           sti ();
16
17           // Loop over processes table looking for process to run .
18           acquire (& ptable . lock);
19           for (p = ptable . proc ; p < $ ptable . proc [ NPROC ]; p ++){
20                 if (p-> state != RUNNABLE)
21                       continue;
22
23                 // Switch to chosen process . It is the process 's job
24                 // to release ptable . lock and then reacquire it
25                 // before jumping back to us.
26                 proc = p;
27                 switchuvm (p);
28                 p-> state = RUNNING ;
29                 swtch (& cpu -> scheduler , proc -> context );
30                 switchuvm ();
31
32                 // Process is done running for now .
33                 // It should have changed its p-> state before coming back .
34                 proc = 0;
35           }
36           release (& ptable . lock);
37
38     }
39 }
```

Name Jia Chen NetID jc8773

(a)(4 points) What scheduling algorithm is implemented by this code? Is it fair (i.e., does each process get an equal share of the CPU)?

Just look up in the page table, find first process that is runnable. And then execute that process. like ~~FIFO~~. First-come, first-served

NO. It's not fair.

(b)(4 points) On lines 20 and 21, we skip the rest of the loop if the process state is not RUNNABLE. What is an example of another state that a process could be in? What would go wrong if we tried to run a process in that state?

the process could be in blocked state.

Maybe the process is still ~~waiting~~ waiting for an event. ~~If~~ Run a process in that state is a waste of CPU time. Because ~~Cause~~ the resources are not ready. The process can not be executed.

(c)(2 points) The `swtch` function, called on line 29, is written in assembly. What is its purpose, and why does it need to be written in assembly rather than C?

It is a switch of context. We change the context of scheduler to context of process that is going to be executed. We need to change the content in registers. So that ~~should~~ should be written is assembly

Name Jia Chen NetID jc8773

7. (16 points) For the following questions CIRCLE True/False. Each question is worth 2 points.

True/**False** (a) Throughput means the time between the moment of submission of a job and the time of its completion.

True/**False** (b) Virtual memory space cannot exceed the limit of physical memory space.

True/False (c) The OS can suspend a process even though the process didn't yield.

True/False (d) A segment is of variable size and a page is of fixed size.

True/False (e) Threads can be used to accelerate I/O heavy processes, for example, a web server process.

True/False (f) In x86 the execution stack grows towards decreasing memory addresses.

True/**False** (g) Batch systems can guarantee best response time.

True/**False** (h) A user process can yield its time slice to another specific user process.

Name Jia Chen NetID jc8773

8. (16 points) For the following questions, circle the correct option for the answer. Each question is worth 2 points.

(a) A process, which has voluntarily given up the CPU, it should be set to ____ state.

A. Running
B. Blocked
C. Ready
D. Zombie

(b) Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

A. first-come, first-served scheduling
B. shortest job scheduling
C. priority scheduling
D. none of the mentioned

(c) In operating system, each process has its own:

A. address space and global variables
B. open files
C. pending alarms, signals and signal handlers
D. all of the mentioned

(d) A thread shares its resources (like data section, code section, open files, signals) with:

A. other processes similar to the one that the thread belongs to
B. other threads that belong to the similar processes
C. other threads that belong to the same process
D. All of these

(e) Swap space exists in:

A. primary memory
B. secondary memory
C. CPU
D. none of the mentioned

(f) With FIFO page replacement algorithm, when a page must be replaced:

A. oldest page is chosen
B. newest page is chosen
C. random page is chosen
D. none of the mentioned

Name Jia Chen NetID jc8773

(g) A process is thrashing if:

A. it is spending more time paging than executing
B. it is spending less time paging than executing
C. page fault occurs
D. swapping can not take place

(h) During memory allocation, external fragmentation will not occur when:

A. first fit is used
B. best fit is used
C. worst fit is used
D. no matter which algorithm is used, it will always occur

Name Jia Chen NetID jc8773

9. (16 points) For the following questions fill in the blank with the correct answer. Each question is worth 2 points.

(a) To overcome the problem of doubling the memory access time, most virtual memory schemes make use of a special high-speed cache for page table entries called a Translation Lookaside Buffer.

(b) When instead of using one page table entry per virtual page, the system keeps one entry per physical page frame, it's called: Inverted Page table.

(c) A process in the blocked state is waiting for user's input.

(d) Pages that have been used a lot recently will probably be used again soon, so Least Recently Used is devised as a page replacement algorithm.

(e) PC Instruction Register. contains the next instruction to execute.

(f) Operating System are special system software that controls which process to run next.

(g) My favorite part of the course so far has been:

The workload is not very big and I can finish it with the instruction of TA.

(h) My least-favorite part of the course so far has been (besides this test):

Go through slides too fast in class hard for me to follow.