University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2016 Anthony D. Joseph

Midterm Exam #1

March 9, 2016
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA Name:	
Discussion Section Time:	

General Information:
This is a **closed book and one 2-sided handwritten note** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

QUESTION	POINTS ASSIGNED	POINTS OBTAINED
1	**18**	
2	**24**	
3	**27**	
4	**19**	
5	**12**	
TOTAL	**100**	

NAME: ____

1. (18 points total) Short answer

a. (9 points) True/False and Why? **CIRCLE YOUR ANSWER.**

i) If every thread acquires locks in the same order (for example, sema_down(A); sema_down(X); sema_down(Y)), deadlock cannot occur.

TRUE FALSE

Why?

ii) It is safe to call `sema_up()` or `lock_release()` from an interrupt handler in Pintos, because they cannot cause the current thread to sleep.

TRUE FALSE

Why?

iii) The function getchar(), which reads a character from standard input, can sometimes run without making any system calls.

TRUE FALSE

Why?

NAME: ____

b. (6 points) Kernels.

i) Briefly, in two to three sentences, explain what a kernel panic is and why it causes a system crash.

ii) Briefly, in two to three sentences, explain the two mechanisms used by the Operating System kernel to prevent user programs from overwriting kernel data structures.

c. (3 points) Briefly, in two to three sentences, explain why the space shuttle failed to launch on April 10, 1981 – be specific but brief in your answer.

NAME: ______________________________

2. (24 points total) Scheduling. Consider the following set of processes, with associated processing times and priorities:

Process Name	Processing Time	Priority
A	4	3
B	1	1
C	2	3
D	1	4
E	4	2

For each scheduling algorithm, fill in the table with the process that is running on the CPU (for timeslice-based algorithms, assume a 1 unit timeslice). Notes:

- A smaller priority number implies a higher priority.
- For RR and Priority, assume that an arriving thread is run at the beginning of its arrival time, if the scheduling policy allows it.
- All of the processes arrive at time 0 in the order Process A, B, C, D, E.
- Assume the currently running thread is not in the ready queue while it is running.
- Turnaround time is defined as the time a process takes to complete after it arrives

Time	**FIFO**	**RR**	**SRTF**	**Priority**
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
Average Turnaround Time				

NAME: ____

3. (27 points total) Synchronization

a. (5 points) Consider the following procedure written in C:

```
struct X data;

struct X *getX(const char key[]) {
    computeDatafromKey(key, &data);
       // a value, based on key, is computed
       //  and stored in data
    return &data;
}
```

i) (2 points) In a single-threaded program, one would call *getX* to obtain an item of type *struct X*, based on the value of *key*. Briefly, in one or two sentences, explain what problems would occur if one used *getX* in a multithreaded program?

ii) (3 points) Rewrite getX to address the problem you mentioned in 3.a.i.

```
struct X *getX(const char key[]) {








}
```

NAME: ________________________________

b. (15 points) Consider a multithreaded operating system that includes monitors and condition variables with the following primitives:

```
mon_t *mon_create()     /* Creates a new monitor */
void mon_lock(mon_t *m) /* Acquires the monitor's lock */
void mon_release(mon_t *m)    /* Releases the monitor's lock */

cv_t *cv_create(mon_t *m)     /* Creates a condition variable
                                 associated with monitor m */
void cv_wait(cv_t *cv)  /* Blocks on the condition variable */
void cv_signal(cv_t *cv)      /* Wakes a thread waiting on cv */
void cv_broadcast(cv_t *cv) /* Wakes all threads waiting on cv */
```

Your task is to implement general purpose semaphores under this system.

```
typedef struct {
  mon_t *m;
  cv_t *c;
  int value;
} sema_t;
```

You may use these syscalls: malloc(), free(), sbrk(), open(), close(), read(), write()

```
sema_t *sema_create(int initval) {










}

void sema_down(sema_t *s)
{








}
```

NAME: ____

```
void sema_up(sema_t *s)
{








}
```

c. (7 points) Consider the following three threads in a concurrent program that uses semaphores Sem1, Sem2, and Sem3.

Thread 1	Thread 2	Thread 3

`L1: sema_down(Sem3);`
`    print("2");`
`    sema_up(Sem2);`
`    goto L1;`
	`L2: sema_down(Sem1);`
	`    print("6");`
	`    sema_up(Sem3);`
	`    goto L2;`
		`L3: sema_down(Sem2);`
		`    print("1");`
		`    sema_up(Sem1);`
		`    goto L3;`

i) (4 points) Are there initial values that can be given to the semaphores so that the threads cooperate to print a string that begins with 16216216216216? If so, give the initial values (tell which value is to be used for which semaphore).

ii) (3 points) Suppose the initial values are Sem1=2, Sem2=6, Sem3=1. Is it possible for the threads to cooperate to produce a string that begins with 1122622? Explain your answer.

4. (19 points) Coding questions.
   a. (13 points) We would like to implement the Unix utility, `tee`, which reads data from standard input and writes that data to standard output, as well as all of the files that are passed on the command line. For example, if you run the following command in bash:

```
$ echo "CS162 is the best!" | tee letter_to_mom.txt personal_motto.txt
```

Then, `tee` would print out "`CS162 is the best!`" to the terminal (standard output), and it would also store "`CS162 is the best!`" into a file named "`letter_to_mom.txt`" and a file named "`personal_motto.txt`".

i) (7 points) We have started writing the code for you. Fill in the **blank spots** in the following code:

```
char buffer[1024];
int main(int argc, char **argv) {
 int files[argc];
 files[0] = ________________________________;
 for (int i = 1; i < argc; i++) {
   files[i] = fileno(fopen(argv[i], "w"));
 }
 while (1) {
   int error = read(____________________________);
   if (error <= 0) break;
   for (int i = 0; i < argc; i++) {
     write(____________________________________);
   }
 }
 return 0;
}
```

ii) (2 points) What does this program do if we cannot open one of the destination files?

NAME: ____

iii) (2 points) In our first for loop, what would change if we started the for loop at i = 0?

iv) (2 points) What does this program do if the same destination file appears more than once on the command line?

b. (6 points) Pintos questions.

i) (3 points) Will the Pintos implementation of semaphores work on a multiprocessor machine? Explain why they will work, or explain how you could fix them.

ii) (3 points) In Pintos, how does the `thread_current()` function find the address of the currently running thread's TCB? (If you did not have access to this function, how could you get the address of the current thread's TCB?)

NAME: ______________________________________

5. (12 points total) Resource Allocation.
Suppose we have the following snapshot of a system with five processes (P1, P2, P3, P4, P5) and 4 resources (R1, R2, R3, R4). There are no outstanding queued unsatisfied requests.

	Current Allocation				**Max Need**				**Still Needs**			
Process	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	1	2	0	0	0	0
P2	1	0	0	0	1	7	5	0	0	7	5	0
P3	1	3	5	4	2	3	5	6	1	0	0	2
P4	0	6	3	2	0	6	5	2	0	0	2	0
P5	0	0	1	4	0	6	5	6	0	6	4	2

Currently Available Resources

R1	**R2**	**R3**	**R4**
1	5	2	0

a. (8 points) Is this system currently in a SAFE, UNSAFE, or deadlocked state? Explain your answer and if possible, give an execution order.

b. (4 points) If a request from process P2 arrives for (0, 4, 2, 0), can the request be granted immediately? Explain your answer and if possible, give an execution order.