Midterm

Due No due date **Points** 34 **Questions** 30 **Available** Oct 21 at 2pm - Oct 21 at 3:15pm about 1 hour **Time Limit** 75 Minutes

This quiz was locked Oct 21 at 3:15pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	49 minutes	27.17 out of 34

Score for this quiz: **27.17** out of 34
Submitted Oct 21 at 2:50pm
This attempt took 49 minutes.

Question 1 — 0.5 / 0.5 pts

UNIX fork does all the following except

Correct!
- Initialize the address space with a copy of only part of the contents of the address space of the parent
- [illegible]
- [illegible]
- [illegible]

Question 2 — 0 / 0.5 pts

Banker's algorithm is a solution of which of the following type

Correct Answer
- Deadlock avoidance

You Answered
- Deadlock prevention
- [illegible]
- [illegible]

Question 3 — 0.5 / 0.5 pts

A lock should ensure the following three properties except

- [illegible]
- [illegible]
- [illegible]

Correct!
- Thread ordering

Question 4 — 0.5 / 0.5 pts

Involuntary thread context switch has to perform three steps including (i) Chooses another thread to run; (ii) Save the state of the running thread; (iii) Restoring the state of the thread to the processor. The correct order of the steps is:

Correct!
- ii, i, iii
- [illegible]
- [illegible]
- [illegible]

Question 5 — 0.5 / 0.5 pts

Thread TCB includes the following information except

- [illegible]
- [illegible]
- [illegible]

Correct!
- Thread source code

Question 6 — 0.5 / 0.5 pts

All of the following can be used to implement multi-threaded processes except

- [illegible]
- [illegible]
- [illegible]

Correct!
- Multi-threaded processes using kernel processes

Question 7 0.5 / 0.5 pts

Kernel to user mode transfer can be triggered due to

Correct!

- [blurred]
- (x) ALL OF ABOVE
- [blurred]
- [blurred]

Question 8 0.5 / 0.5 pts

At a minimum, the hardware must support the following except

Correct!

- (x) Kernel threads
- [blurred]
- [blurred]
- [blurred]

Question 9 0.5 / 0.5 pts

If an individual thread is unable to take advantage of the overlap of CPU and I/O operations, the OS can overlap the CPU execution of one thread with the I/O operation of other threads.

Correct!

- (x) True
- [blurred]

Question 10 0.5 / 0.5 pts

most system crashes are due to bugs in device drivers rather than in the operating system itself.

Correct!

- (x) True
- [blurred]

Question 11 0.5 / 0.5 pts

The interrupt hander is a kind of kernel thread.

Correct!

- [blurred]
- (x) False

Question 12 0.5 / 0.5 pts

Almost all commercial operating systems today support kernel-supported threads.

Correct!

- (x) True
- [blurred]

Question 13 0.5 / 0.5 pts

Lock variable is sufficient to solve all synchronization problems.

Correct!

- [blurred]
- (x) False

Question 14 0.5 / 0.5 pts

Deadlock happens mostly due to inappropriate OS implementation by programmers.

Correct!

- [blurred]
- (x) False

Question 15 1.5 / 2 pts

After being put in the ready list, a thread is in [Select] state. If an I/O event occurs, it transfers to RUNNING state. During RUNNING state, a thread may be voluntarily or involuntarily switched to [Select] state.

Answer 1:

Correct! READY

Answer 2:

You Answered RUNNING

Correct Answer WAITING

Answer 3:

Correct! RUNNING

Answer 4:

Correct! READY

Question 16 2.25 / 3 pts

The four conditions of deadlocks are (names starts with capital for only the first letter):

1. Bounded resources
2. No preemption
3. Wait-while-holding
4. Circular waiting

Since they are required conditions, one can prevent deadlock by failing just one of these four conditions. For example, for dinning lawyer problems of 5 lawyers, providing 1 additional chopstick(s) may prevent the deadlock. This approach fails condition # 1 . We may also require all lawyers to get "either all or none" chopsticks to prevent deadlock by failing condition # 4 .

Answer 1:

Correct! Bounded resources

Answer 2:

Correct! No preemption

Answer 3:

You Answered Wait-while-holding

Correct Answer Wait while holding

Answer 4:

Correct! Circular waiting

Answer 5:

You Answered required

Correct Answer necessary

Answer 6:

Correct! 1

Answer 7:

Correct! 1

Answer 8:

You Answered 4

Correct Answer 3

Question 17 1 / 1 pts

The output of the following code is

6

6

```
// Program 1
main() {
    int val = 5;
    int pid;

    if (pid = fork())
        wait(pid);
    val++;
    printf("%d\n", val);
    return val;
}
```

Answer 1:

Correct! 6

Answer 2:

Correct! 6

Question 18 1.31 / 3.5 pts

Now suppose that we will implement a RWLock class as follows using lock variables and condition variables. The requirements are that (i) readers do not conflict readers; (ii) writers conflict with both readers and writers; (iii) if there is any writer waiting to write, readers will have to yield.

Note: Please **make sure** to leave one blank space before and after each operator such as ==, %, +, / , etc.

```
class RWLock {
   private:
  Lock lock;
  CV readGo;
```

```
    CV writeGo;

    int activeReaders;
    int activeWriters;
    int waitingReaders;
    int waitingWriters;
    bool readShouldWait();
    bool writeShouldWait();
     public:
    RWLock();
    ~RWLock();
    void startRead();
    void doneRead();
    void startWrite();
    void doneWrite();
};
```

(1) Please complete the following function for readShouldWait().

```
bool RWLock::readShouldWait() {
    return ( activeWriters>0 || waitingWriters>0 );
}
```

(2) Please complete the following function for startWrite().

```
bool RWLock:: startWrite() {
    lock.acquire();
    waitingWriters++;
    while( writeShouldWait() ) {
        writeGo.wait(&lock)
    }
    waitingWriters--;
    activeWriters++;
    lock.release();
}
```

(3) Please complete the following function for doneRead().

```
bool RWLock:: doneRead () {
    lock.acquire();
    activeReaders--;
    if( waitingWriters>0 && !writeShouldWait() )
        writeGo.signal(&lock);
    lock.release();
}
```

Answer 1:

Correct! activeWriters>0

Correct Answer activeWriters > 0

Answer 2:

Correct! waitingWriters>0

Correct Answer waitingWriters > 0

Answer 3:

Correct! writeShouldWait()

Answer 4:

You Answered writeGo.wait(&lock);

Correct Answer writeGo.Wait(&lock)

Answer 5:

You Answered waitingWriters--;

Correct Answer waitingWriters--

Correct Answer waitingWriters --

Answer 6:

You Answered activeReaders--;

Correct Answer activeReaders--

Correct Answer activeReaders --

Answer 7:

You Answered waitingWriters>0

Correct Answer activeReaders==0

Correct Answer activeReaders == 0

Answer 8:

You Answered !writeShouldWait()

Correct Answer waitingWriters>0

Correct Answer waitingWriters > 0

Question 19 1 / 2 pts

Please fill in the missing part of the following pseudo code for bounded buffer problem, assuming Mesa Semantics for condition variable. Buffer is an array/vector named "buf". Please make sure to leave one blank space before and after each operator such as ==, %, +, / , etc.

```
get() {
    lock.acquire();
    while( buf == empty ) {
        empty .wait(&lock);
    }
```

item = buf[0] ;

front++;

full.signal(&lock);

lock. release ();

return item;

}

Initially: front = tail = 0; MAX is buffer capacity and **empty/full** are condition variables.

Answer 1:

You Answered: buf == empty

Correct Answer: front == tail

Correct Answer: tail == front

Answer 2:

Correct!: empty

Answer 3:

You Answered: buf[0]

Correct Answer: buf[front % MAX]

Correct Answer: buf[front % MAX]

Correct Answer: buf[front%MAX]

Answer 4:

Correct!: release

Question 20

0 / 0.5 pts

All of the following are atomic operations except

Correct Answer: Acquire all or none

[blurred]

You Answered: Unix file open, i.e., check if it exists, create if it does not exist, and then open the file

[blurred]

Question 21

0 / 0.5 pts

All of the following are possible approaches of inter-process communication except

[blurred]

You Answered: File read/write

[blurred]

Correct Answer: Multi-threading

Question 22

4.74 / 5 pts

Suppose that three threads A, B, and C in a system are competing for a total of 8 pages of memory. A, B, and C needs 4, 5, and 5 pages to complete respectively. A, B, and C currently holds 3, 2, and 2 pages respectively. Now, A, B, and C all try to obtain the next page they need. B is the next thread to continue. Show the detailed steps such that with banker's algorithm, all threads eventually get the pages they need and release all pages they hold.

For each of the blanks, enter either the number of pages currently being allocated for a specific thread. If a thread is blocked due to an unsafe state (from banker's algorithm), enter W.

Note: only one action at a time, e.g., release, assign, wait. There is no cell intentionally left as blank.

Process		Allocation												
A	3	3	3	4	0	0	0	0	0	0	0	0	0	0
B	2	w	w	w	w	3	3	4	4	5	0	0	0	0
C	2	2	w	w	w	w	3	3	w	w	w	4	5	0

Answer 1:

Correct!: 3

Answer 2:

Correct!: 3

Answer 3:

You Answered: 4

Correct Answer: 3

Answer 4:

You Answered: 0

Correct Answer: 4

Answer 5:

Correct!: 0

Answer 6:

Correct! 0

Answer 7:

Correct! 0

Answer 8:

Correct! 0

Answer 9:

Correct! 0

Answer 10:

Correct! 0

Answer 11:

Correct! 0

Answer 12:

Correct! 0

Answer 13:

Correct! 0

Answer 14:

Correct! W

Answer 15:

Correct! W

Answer 16:

Correct! W

Answer 17:

Correct! W

Answer 18:

Correct! 3

Answer 19:

Correct! 3

Answer 20:

Correct! 4

Answer 21:

Correct! 4

Answer 22:

Correct! 5

Answer 23:

Correct! 0

Answer 24:

Correct! 0

Answer 25:

Correct! 0

Answer 26:

Correct! 0

Answer 27:

Correct! 2

Answer 28:

Correct! W

Answer 29:

Correct! W

Answer 30:

Correct! W

Answer 31:

Correct! W

Answer 32:

Correct! 3

Answer 33:

Correct! 3

Answer 34:

Correct! W

Answer 35:

Correct! W

Answer 36:

Correct! W

Answer 37:

Correct! 4

Answer 38:

Correct! 5

Answer 39:

Correct! 0

Question 23 0.67 / 1 pts

The three "formal" properties "M-P-B" that must be satisfied for a lock are [mutual exclusion], [progress], and [bounded resources], respectively.

Answer 1:

Correct! Mutual Exclusion

Correct Answer mutual exclusion

Answer 2:

Correct! Progress

Correct Answer progress

Answer 3:

You Answered bounded resources

Correct Answer Bounded waiting

Correct Answer bounded waiting

Question 24 1.5 / 1.5 pts

The three operations on a condition variable are, **in alphabetic order**, [broadcast], [signal], [wait], where [signal] is used to unblock one waiting thread and [broadcast] is used to unblock "all" waiting threads for a specific variable.

Answer 1:

Correct! broadcast

Answer 2:

Correct! signal

Answer 3:

Correct! wait

Answer 4:

Correct! signal

Answer 5:

Correct! broadcast

Question 25 0.5 / 0.5 pts

An alternative approach to multi-threading for concurrency is called [event]-driven programming, which uses asynchronous I/O.

Answer 1:

Correct! event

Correct Answer events

Question 26 1 / 1 pts

```
Line#   // Thread A                 // Thread B
1       lock1.acquire();            lock1.acquire();
2       ...                         ...
3       lock2.acquire();            lock2.acquire();
4       while(need to wait) {       ...
5           cv.wait(&lock2);        cv.signal();
6       }                           lock2.release();
7       ...                         ...
8       lock2.release();            lock1.release();
9       ...
10      lock1.release();
```

For above pseudo code, assuming that Thread A obtained lock1 and then lock2 successfully, deadlock will happen when Thread A just executed Line # [5] and is now busy waiting for Thread B to execute Line # [5], which never happens since Thread A holds the lock.

Answer 1:

Correct! 5

Answer 2:

Correct! 5

Question 27 0.5 / 0.5 pts

Which of the choice include all possible outcomes of the following program? x is initialized to 0.

Thread A	Thread B
x = x + 1;	x = x + 2;

Correct!

- 1,2,3
- [illegible]
- [illegible]

Question 28 2.7 / 3 pts

Please finish the following thread programming. Please use the simple Threads API from the textbook.

```
#define NTHREADS 10
static thread_t threads [NTHREADS] ;
void go(int n) {
    cout << "child thread running!" << endl;
    thread_exit (100 + n); // terminate the thread
}

int main() {
    for(int i = 0;i < NTHREADS ;++i)
        create_thread (& threads[i] , &go , i ); // create ith thread with go function and pass i as parameter
    }
    for(int i = 0;i < NTHREADS ;++i)
        int exitValue = thread_join ( threads[i] ); // wait for ith thread to finish
        cout << exitValue << endl;
    }
    return 0;
}
```

Answer 1:
Correct! [NTHREADS]
Correct Answer NTHREADS

Answer 2:
Correct! thread_exit

Answer 3:
Correct! NTHREADS

Answer 4:
You Answered create_thread
Correct Answer thread_create

Answer 5:
Correct! threads[i]

Answer 6:
Correct! &go

Answer 7:
Correct! i

Answer 8:
Correct! NTHREADS

Answer 9:
Correct! thread_join

Answer 10:
Correct! threads[i]

Question 29 1 / 1 pts

The data structure that stores all the information the operating system needs about a particular process is called process control bloc (no abbreviation), and similarly, the data structure that stores all the information the operating system needs about a particular thread is called thread control block (no abbreviation).

Answer 1:
Correct! process control block
Correct Answer Process control block

Answer 2:
Correct! thread control block
Correct Answer Thread control block

Question 30 1 / 1 pts

In user mode, the processor checks each instruction before executing it to verify that it is permitted to be performed by that process. In kernel mode, the operating systems executes with protection checks turned off. Together, this is called dual -mode operation.

Answer 1:

Correct!	user
	Answer 2:
Correct!	kernel
Correct Answer	kernal
	Answer 3:
Correct!	dual

Quiz Score: **27.17** out of 34