Quiz Submissions - Midterm Exam

Daniel Davis (username: DanielDavis.dkd6)

Attempt 1

Written: Oct 14, 2013 3:01 PM - Oct 14, 2013 4:47 PM

Submission View

released: Oct 23, 2013 2:12 PM

Question 1 **1 / 1 point**

Global variables must be stored in the process control block when a process is taken off the CPU.

- True
- ✓ ● False

Question 2 **1 / 1 point**

Time-shared operating systems allow users to interact with a system while the system is running multiple processes in memory.

- ✓ ● True
- False

Question 3 **1 / 1 point**

Using the basic definition of a thread, stack is shared between two threads.

- True
- ✓ ● False

Question 4 **0 / 1 point**

The message passing model uses shared memory for two processes to communicate.

✗ [✓ True]

➡ False

Question 5 **1 / 1 point**

The file descriptor table survives a `fork()` system call.

✓ [✓ True]

False

Question 6 **1 / 1 point**

The system calls trigger a context switch allowing the kernel to take over execution.

✓ [✓ True]

False

Question 7 **1 / 1 point**

User-level threads are more vulnerable to priority inversion than kernel-level threads.

✓ [✓ True]

False

Question 8 **0 / 1 point**

Switching from kernel mode to user mode requires execution of a privileged instruction.

✗ [✓ True]

➡ False

Question 9 **1 / 1 point**

The stack pointer must be written into the PCB during a context switch.

✓ [✓ True]

False

Question 10 **1 / 1 point**

In user mode, privileged instructions are disabled by the CPU.

✓ [✓ True]

False

Question 11 **4 / 4 points**

A process that accesses an invalid memory location must switch from the ___running___

✓ state to the ___terminated___ ✓ state.

Question 12 **4 / 4 points**

A system call triggers a ___interrupt___ ✗ **(software interrupt)** causing the running process to become ___blocked___ ✓ .

Question 13 **2 / 2 points**

Mutual exclusion requires that a process or a thread acquires a ___lock___ ✓ before entering its critical section.

Question 14 **0 / 2 points**

Elapsed time from when the system becomes aware of a new process to until the process terminates and exits is called ___response___ ✗ **(turn-around)** time.

Question 15 **2 / 2 points**

A ___semaphore___ ✓ has memory so that the order of `wait()` and `signal()` calls can cause the `wait()` to return immediately without sleeping.

Question 16 **2 / 2 points**

A ___busy loop___ ✓ is a loop that executes until a condition is met, but the condition can't be altered until the quantum expires and another process is scheduled.

Question 17 **2 / 2 points**

More computer efficiency is achieved by scheduling ___IO___ ✓ -bound processes first.

Question 18 **2 / 2 points**

___Shortest Job First___ ✓ scheduling is non-preemptive but may cause long jobs to suffer from starvation.

Question 19 **2 / 2 points**

The *test-and-set* instruction can be used to implement locks and semaphores without disabling ___interrupts___ ✓ .

Question 20 **2 / 2 points**

A ___race condition___ ✓ is a situation in which the outcome is determined by the timing of quantum expiration.

Question 21 **2 / 2 points**

The `fork()` system call returns ___0___ ✓ for the child process.

Question 22 **2 / 2 points**

A process in ___new___ ✓ or ___blocked___ ✓ state cannot be selected for execution.

Question 23 **2 / 2 points**

Actions that cannot be interrupted in the middle (either execute completely or not at all) are ___atomic___ ✓ .

Question 24 **4 / 4 points**

Consider the following code fragment that shows access control to SharedResource().

Semaphore guard = _____.

```
guard.Wait();
SharedResource();
guard.Signal();
```

What value should be used to initialize guard to ensure mutual exclusive access to SharedResources()? Give an example of a resource that would be accessed through such a guard.

Guard should be initialized to a value of 1 for mutual exclusive access. An example is a printer that is a device that is shared, but should only be shared by one device at a time.

The correct answer is not displayed for Long Answer type questions.

Question 25 **4 / 4 points**

Consider the following code fragment that shows access control to SharedResource().

Semaphore guard = _____.

```
guard.Wait();
SharedResource();
guard.Signal();
```

What value should be used to initialize guard to allow a maximum 5 processes to simultaneously access SharedResources(). Give an example of a resource that would be accessed through such a guard.

Guard should be initialized to a value of 5 for a maximum of 5 devices to simultaneously access. A good example is reading from a file and being able to have 5 devices read from this shared resource simultaneously. However, these can only be File Readers and not File Writers, i.e. cannot write to the shared resource!

The correct answer is not displayed for Long Answer type questions.

Question 26 **2 / 2 points**

Consider the following code fragment that shows access control to SharedResource().

Semaphore guard = _____.

```
guard.Wait();
SharedResource();
guard.Signal();
```

What would be the result of setting the initial value of guard to 0 (zero) in the above code? Briefly explain.

If the guard is set to 0, then when the first thread or process attempts to acess the shared device, it will be blocked. Also any other threads or process or device that attempt to access the shared resource will be blocked as well. This is a deadlock condition that should be avoided.

The correct answer is not displayed for Long Answer type questions.

Question 27 **8 / 10 points**

Write a multi-threaded program that uses the divide-and-conquer principle to solve a linear search problem with multiple threads searching simultaneously in a portion of an array of integers. For your solution, use 4 worker threads that work on an evenly divided portion of the array. The main thread waits for the worker threads to complete their task and then prints the result of the search. Worker threads are given information about the element to search and the portion of the array they need to examine. The result is an index location of the array that stores the searched element. If the value to be searched occurs at multiple locations any of the locations in which the value occurs may be printed. If the value to be search is not contained in the array, a value of -1 must be printed. Use the code below to get started.

PThread API:

pthread_create (pthread_t *thread, const pthread_attr_t *attr, void * (*start_routine)(void *), void *arg);

Example: pthread_create (&tidA, NULL, tFunc, (void *)“threadA”);

pthread_join(pthread_t thread, void **retval);

Example: pthread_join(tidA, NULL);

pthread_exit (void *retVal);

```
#define N 1024
int numbers[N];

struct userInput {

int searchfor, startindex, finishindex, foundAtIndex;

};

void* tlinSearch(void *val) {

struct userInput *var = (struct userInput *) val;

int i = val->startindex;

for(i = val->startindex; i <= val->finishindex; i++)

if(numbers[i] == val->searchfor) val->foundAtIndex = i;

val->foundAtIndex = -1; //value not in array

pthread_exit(0);

}

int main(int argc, char** argv)
```

```
{
    // variable declarations
    // you must complete this
pthread_t workerthreads[4];

struct userInput input;

    // prompt user for data
    // you do not have to implement this

    // start worker threads
    // you must implement this

int i = 0;

for(i = 0; i < 4; i++) {

input.startindex = ;

input.finishindex = N/4;

 pthread_create(&workerthreads[i], NULL, tlinSearch, (void *) &input);

}

    // wait for worker threads
    // you must implement this

for(i = 0; i < 4; i++)

pthread_join(workerthreads[j], NULL);

    // print result
    // you must implement this

 printf("%i\n", input.foundAtIndex);

    return 0;
}
```

The correct answer is not displayed for Long Answer type questions.

Hide Feedback

Incomplete code of dividing array into multiple ranges. You need an array structs so that the threads can work on separate ranges. Sharing one struct will cause race conditions. (-2)

Question 28 **2 / 5 points**

For the multi-threaded program above, describe a snapshot of the process in memory once the threads have been fully started. Include in your snapshot stack, global data, and heap memory, TCB and PCB. For TCB and PCB only include made-up IDs of processes and threads, not any registers and their values.

Stack: workerthreads[], input, i = 4

Heap:

Global data: numbers[]

TCB: TID 307, TID 308, TID 309, TID 310

PCB: PID 306

The correct answer is not displayed for Long Answer type questions.

Hide Feedback

Each thread has its own TCB and stack. the different thread IDs are not stored in the same TCB

Question 29

10 / 10 points

Consider the following algorithm that is claimed to be a “good solution” to the critical section problem between **two** processes. I have numbered the statements so that you can refer to the numbers in your explanation.

```
          bool flag[2] = {false, false}; /* shared variable */
          ...
          while ( 1 ) {
/* 1 */       flag[i] = true;
/* 2 */       while ( flag[j] )
              {
/* 3 */           flag[i] = false;
/* 4 */           while ( flag[j] ) ;
/* 5 */           flag[i] = true;
              }
/* 6 */       Critical_Section();
/* 7 */       flag[i] = false;
          }
```

–

1. Does the algorithm ensure mutual exclusion? Briefly explain using the code and line numbers above?

2. Does the algorithm allow progress? Briefly explain using the code and line numbers above?
3. Does the algorithm ensure a bounded wait? Briefly explain using the code and line numbers above?

Mutual exclusion: Yes. Because a process checks to see if another process has set it's flag to True indiciating that it wants to go into the Critical Section. The process waits on lines 2 and 4 in the while loop if another process is already accessing the Critical Section.

Progress: Yes, If a process set the flag to True indicating that it wants to go into the Critical Section on Line 1, then it can go into the Critical Section by skipping the while loop on Line 2.

Bounded wait: NO, Line 3 could execute in a process and then it could get interrupted and switch to the other process, then the other process could set it's flag to True. This can lead to starvation for a process.

The correct answer is not displayed for Long Answer type questions.

Question 30 **6 / 10 points**

Write a distributed solution using multiple forked processes for computing Collatz number sequences for a set of user entered numbers. Each Collatz number sequence must be computed by a separate process and printed to the screen as numbers in the sequence are computed. A Collatz number series is computed using the following formula:

n = n/2, if n is even
n = 3n + 1, if n is odd

The first number in the sequence is the original value of n. Subsequently, values of n are updated using the above formula and printed until n is 1. For example, starting with 5, the Collatz sequence is 5, 16, 8, 4, 2, 1. All Collatz number sequences end with a value of 1. Use the starter code below to implement a solution. You must use waitpid() to wait for all processes that have been forked before you may terminate the main program. You may assume that k < N and that no negative values have been entered.

System Calls:

pid_t fork(void);

pid_t_waitpid (pid_t pid, int *status, int options);

WEXITED   Wait for children that have terminated.
WNOHANG   Return immediately if no child has exited.

```
#define N 8

int main(int argc, char** argv)
{

    // variable declarations
    // you must complete this
    int input[N], k, i;
    int status;

    // prompt user for data
    printf ("How many Collatz sequences: ");
    scanf ("%d", &k);

    for (i=0; i < k; i++) {
        printf ("Enter a number: ");
        scanf ("%d", &input[k]);
    }

    // compute Collatz number sequences

for(i = 0; i < k; i++)

 calculate(input[i]);

    return 0;
}

void calculate(int n) {

 pit_t pid = fork();

//child

if(pid == 0) {

   printf("%i, ", n);

   if(n % 2 == 0) n = n / 2;

   else n = 3 * n + 1;

}

else { //parent

   waitpid(pid, &status);

   if(n == 1) { //print 1 and return

     printf("%i.\n", n);

     return;

     }

    else  calculate(n);

}

}
```

The correct answer is not displayed for Long Answer type questions.

The child process must compute the entire Colatz number sequence. You need a while loop to do that. The recursive calls of the parent will only fork unecessarily processes. And you are waiting for the child process to finish rather than starting all processes in parallel to compute the Colatz sequences.

Question 31 **3 / 5 points**

In the previous problem, what issues may arise when the processes computing the Collatz numbers print the number sequences to the screen? Explain.

The numbers could print in random order and not in the correct Collatz sequence, because of scheduling. Is a race condition.

The correct answer is not displayed for Long Answer type questions.

Hide Feedback

if print statements are scheduled, then the order is not random. the sequences will print in order, however they could be mixed together.

Question 32 **7 / 10 points**

A system runs 5 processes A, B, C, D, and E. Process A and B are CPU-bound processes that require 15 and 10 ticks to complete, C and D perform file IO after 2 ticks each, while E sends a message over the network after 3 ticks. The arrival times for each process are shown below. Illustrate the ready and IO queues for the processes at time 5, 15, 25, and 35assuming that we select a round-robin scheduler with 10 ticks and 1 tick in handling context switches. The IO time shows the time it takes to wait for completion of IO. Use the following notation to describe a queue:

Ready Queue: A -> B -> C.

This queue contains three processes starting with A at the top and C at the end of the queue.

Processes	**Arrival Time (in ticks)**	**CPU Burst Time (in ticks)**	**IO Time (in ticks)**
A	0	15	0
B	4	10	0
C	8	5	7
D	14	8	12
E	12	7	10

time 5:

Ready Queue: B

IO Queue:

time 15:

Ready Queue: C -> A -> E -> D

IO Queue:

time 25:

Ready Queue: E -> D

IO Queue: C

time 35:

Ready Queue: E -> C

IO Queue: D

The correct answer is not displayed for Long Answer type questions.

Hide Feedback

time5:ready:B io:null network:null

time15:ready:C>A>E>D io:null network:null

time25:ready:E>D io:C network:null

time35:ready:C io:null network:E

Attempt Score:	82 / 100
Overall Grade (highest attempt)**:**	82 / 100

Close