CS 3214 Final Exam

This is a closed-book, closed-internet, closed-cell phone and closed-computer exam. However, you may refer to your 2 sheets of prepared notes. **Your exam should have 14 pages with 5 topics totaling 150 points. You have 120 minutes**. Please write your answers in the space provided on the exam paper. If you unstaple your exam, please put your initials on all pages. You may use the back of pages if necessary, but please indicate if you do so. Answers will be graded on correctness and clarity. You will lose points if your solution is more complicated than necessary or if you provide extraneous, but incorrect information along with a correct solution.
To be considerate to your fellow students, if you leave early, do so with the least amount of noise.

Name (printed) _____

I accept the letter and the spirit of the Virginia Tech undergraduate honor code – I have not given or received aid on this exam.

(signed) _____

#	Problem	Points	Score
1	Concurrency	30	
2	Synchronization	40	
3	Memory	20	
4	Communication	40	
5	Essay Question: Modern Software Development	20	
	Total	150	

1. Concurrency (30 points)

Show below is the partial code for a stack of integers.

```
struct stack_elem {
  int num;
  stack_elem *next;
};

struct stack {
  int size;
  struct_elem *top;
};

stack *stk =
malloc(sizeof(stack));
stk->size = 0;
stk->top = (stack_elem*)0;
```

```
void stack_push(stack_elem *s){
    s->next = stk->top;
    stk->top = s;
    stk->size++;
}

stack_elem* stack_pop() {
      (stack_elem*)tp = stack-
>top
      if (stk->size) {
        stk->top = stk->top-
>next;
        stk->size--;
      }
    return tp;
}
```

a) (6 points) Illustrate in detail how a race condition can occur in the stack_push function that leads to an incorrect result when the function is executed concurrently by two threads.

Here are at least two race conditions; other variations are possible; accept any valid scenario

1. concurrent execution of `stk->size++;` leading to incorrect size.

T1: load stk->size to register
T2: load stk->size to register
T1: increment register value and store result in `stk->size`
T2: increment register value and store result in `stk->size`

2. concurrent execution of: s->next = stk->top; stk->top = s; leading to incorrect stack contents

T1: executes `s->next = stk->top;`
T2: executes `s->next = stk->top;`
T2: executes `stk->top = s;`
T1: executes `stk->top = s;`

b) (6 points) Assume that all of the stack functions are thread safe. Is the following function also thread safe? Explain your answer.

```
void stack_add(stack* stk) {// add top two numbers on stack
 if (stk->size >= 2) {      // push the result on the stack
   stack_elem *e = stack_pop();
   stack_elem *f = stack_pop();
   stack_elem *g = malloc(sizeof(stack));
   g->num = e->num+f->num;
   stack_push(g);
 }
}
```

No, this is not thread safe. Here is one scenario of concurrent execution leading to an incorrect result.

T1: executes first stack_pop operation
T2: executes first stack_pop operation
T1: executes second stack_pop operation
T2: executes second stack_pop operation
T1: computes sum (second value may not be valid) and does stack_push
T2: computes sum (second value may not be valid) and does stack_push

The additions did not add consecutive stack elements.

c) (6 points) One of the above functions has a memory leak. Explain how the memory leak occurs and show the changed code that you need to fix the leak.

The stack_add function has a memory leak because the two elements popped from the stack are not freed. The change is as follows:

add: free(e) and free(f) somewhere after the computation of the sum.

d) (6 points) Three resources are used concurrently. To guarantee mutual exclusion the access to the resources is protected by locks named LA, LB, and LC as shown in the code below for three threads

Thread 1	Thread 2	Thread 3

lock(LA);
//use resource A
unlock(LA);
lock (LB);
//use resource B
unlock(LB);
	lock(LC);
	//use resource C
	unlock(LC);
	lock (LB);
	//use resource B
	unlock(LB);
		lock(LA);
		lock(LB)
		lock(LC)
		//use resources A,B, and C
		unlock(LA);
		unlock (LB);
		unlock(LC);

Can this code deadlock? If so explain how. If not explain what prevents deadlock from occurring.

No, deadlock cannot occur in this example. There can be no circular wait among these threads. Threads 1 and 2 only contend for ownership of one lock (LB) and cannot form a cycle. Threads 1 and 3 contend for two locks but acquire their locks in the same order and cannot form a cycle. Threads 2 and 3 contend over two locks and in different orders. However, if thread 2 is holding lock LC (blocking thread 3) it will release this lock before acquiring lock LB. In this case either thread 2 will acquire lock LB or thread 3 having acquired lock LB will be able to acquire lock LB.

e) (6 points) In general, name and briefly describe two situations closely related to deadlock that seriously hamper the progress of one or more threads.

livelock – two processes repeating same operations and blocking each other
indefinite postponement – thread never gets chance to run as writers in a readers-writers problem when there are always readers present

2. Synchronization (40 pts)

a) (10 points) How can you make stack_push function from Question 1 thread safe and avoid potential race conditions using the synchronization primitives from the standard Pthreads library. Show all changesneeded in the code.

```
struct stack {
  pthread_mutex_t mutex;
  …
};

stack->mutex = PTHREAD_MUTEX_INITIALIZER;  // in
initialization

void stack_push(stack_elem *s){
    pthread_mutex_lock(stk->mutex);
    …
    pthread_mutex_unlock(stk->mutex);
}
```

b) (10 points) How can you make stack_add function from Question 1 thread safe and avoid potential race conditions using the synchronization primitives from the standard Pthreads library. Show all changes that you make in the code.

```
void stack_add(stack* stk) {// add top two numbers on stack
 pthread_mutex_lock(stk->mutex);
 …
 }
 pthread_mutex_unlock(stk->mutex);
}
```

c) (20 points) *Condition Variables.* Threads in an application are synchronized via turn-taking using an integer service number that starts at zero. A thread requests a service number and at some later time waits for that service number to be the current service number. When the thread with the current service number finishes it increments the current service number by one.

i. (4 points) Show the definition of a struct type named service that you would use to implement turn-taking synchronization using Pthreads.

```
struct service {
   int current; // current turn
   int next;    // to assign turn request
   pthread_mutex_t mutex;
   pthread_cond_t turn;
};
```

ii. (4 points). Show the code needed to implement the function to initialize an instance of your service structure. The signature of this function is:

```
service* service_create(void);
```

```
service* service_create(void) {
   service *s = malloc(sizeof(service));
   s->current = 0;
   s->next = 0;
   s->mutex = PTHREAD_MUTEX_INITIALIZER
   s->turn = PTHRAD_COND_INITIALIZER;
}
```

iii. (4 points) Show the code needed to implement the function executed by a thread when it gets a service number. Assume that the service struct has been properly initialized. The signature of this function is:

```
int service_get(service *sp);
```

```
int service_get(service *sp) {
      pthread_mutex_lock(sp->mutex);
      int myturn = sp->next;
      sp->next++;
      pthread_mutex_unlock(sp->mutex);
      return myturn;
}
```

iv. (4 points) Show the code needed to implement the function executed by a thread when it waits for a service number. Assume that the service struct has been properly initialized. The signature of this function is:

```
void service_wait(service *sp, int myturn);
```

```
void service_wait(service *sp, int myturn) {
        pthread_mutex_lock(sp->mutex);
        while(myturn != sp->current){
           pthread_cond_wait(sp->turn);
        }
        pthread_mutex_unlock(sp->mutex);
}
```

v. (4 points) Show the code needed to implement the function executed by a thread when it has finished its turn. The signature of this function is:

```
void service_finish(service *sp);
```

```
void service_finish(service *sp) {
        pthread_mutex_lock(sp->mutex);
        sp->current++;
        pthread_cond_broadcast(sp->turn);
        pthread_mutex_unlock(sp->mutex);
}
```

3. Memory (20 points)

a) Multiple threads are executing in an address space. Some threads dynamically allocate and free blocks of memory of size B1 very rapidly. The other threads dynamically allocate and free blocks of memory of size B2 much more slowly.

i. (3 points) What strategy would you use for your memory allocator and how would you handle coalescing if B1=B2? Justify your choice.

The simplest strategy is first-fit with no coalescing. Equivalent answers are also acceptable.

ii. (4 points) What strategy would you use for your memory allocator and how would you handle coalescing if B2 was much larger than B1 (e.g., B2 was 10kb and B1 was 20 bytes)? Justify your choice.

Use some form of segregated storage first-fit and no coalescing

b) (4 points) What if any is the significant effect that a mode switch has on the virtual memory mapping for a given process?

The virtual-physical memory mapping is changed so that kernel portions of the virtual address space become accessible.

c) (4 points) What if any is the significant effect that a context switch has on the virtual memory mapping for a given process?

The virtual-physical memory mapping is changed so that the pages of a different process are now in effect.

d) (5 points) You are implementing a test to see the performance effects of high memory demands. You write a program that reads an integer value. N, from the command line and allocates that many 1Mb blocks of memory. The code looks something like:

```
for(i=0, i<N, i++ )
    char* p = malloc(1000000);
```

When you run your test using larger and larger values for N there is no discernible difference on the systems memory load. Keeping in mind what you know about dynamic memory allocation and virtual memory, explain what happened.

The memory is allocated in virtual memory but not in physical memory because no access has been made to the allocated addresses.

4. Communication (40 pts)

a) (10 points) Shown below is the code for a server's function to open a connection with a client using IPV6 as described in the comments. There are five errors in the function. Identify these errors and indicate how they should be fixed. Write you answers on the back of this page.

```
int get_client(int server_port)
{
   // open a TCP/IP socket connection on port server_port
   // return file descriptor for this connection

   int sd=-1, on=1;
   struct sockaddr serveraddr;

   if ((sd = socket(AF_INET6, SOCK_DGRAM, 0)) < 0)
      {
         perror("socket() failed");
         break;
      }

   if (setsockopt(sd, SOL_SOCKET, SO_REUSEADDR,
                      (char *)&on,sizeof(on)) < 0)
      {
         perror("setsockopt(SO_REUSEADDR) failed");
         break;
      }

   memset(&serveraddr, 0, sizeof(serveraddr));
   serveraddr.sin6_family = AF_INET6;
   serveraddr.sin6_port   = server_port;
   serveraddr.sin6_addr   = in6addr_any;

   if (bind(sd,(struct sockaddr *)&serveraddr,
               sizeof(serveraddr)) < 0)
      {
         perror("bind() failed");
         break;
      }


   if ((sdconn = accept(sd, NULL, NULL)) < 0)
      {
         perror("accept() failed");
         break;
      }

   return sd;
}
```

The five errors are:

1. SOCK_DGRAM should be SOCK_STREAM
2. sockaddr should be sockadd_in6
3. server_port should be htons(server_port)
4. the call to listen(int) is missing
5. wrong file descriptor returned

b) A NAT router with the internet address 192.168.5.62 is used to create a private local network. One of the machines on this private local network has the internet address 128.119.40.186. A client on this machine opens a connection to a server on a machine not in the private local network using, in part, this code:

int err = getaddrinfo("www.myserver.com", "3434", …)

The client's port is assigned the port number 2345 by the client's local host. Assume that the server www.myserver.com is at internet address 156.173.41.18. An open socket is identified by a quadruple S: (source address, source port), D: (destination address, destination port). The next port number to be assigned by the NAT is 6789.

i. (2 points) What is the quadruple for the socket on the client?

[S: 128.119.40.186,2345, D: 156.173.41.18, 3434]

ii. (2 points) What is the quadruple for the socket on the server?

[S: 156.173.41.18, 3434, D: 192.168.5.62, 6789]

iii. (2 points) What is the relevant table entry in the NAT translation table?

192.168.5.62, 6789 -> 128.119.40.186,2345

c) (9 points) Unix has both pipes and sockets for a TCP connection. Describe three similarities and three differences between them.

Similarities	Differences
accessed via file descriptor	**local vs. remote communication**
ordered byte stream	**socket has address associated with it**
buffered	**unidirectional vs. bidirectional**
may have short reads	**sockets used for accepting connections**
reliable, sequenced	**sockets have queue for pending connects**

d) The esh project allowed a user to create, monitor, and control the execution of processes on their local machine. This question explores resh, a shell to create, monitor, and control the execution of remote processes as well.

Version 1 of resh allows a process to be executed on a remote site and retrieve a single file that contains the output generated on STDOUT of the remote process. The resh shell communicates with a remote execution daemon (reshd) using the http 1.1 protocol. The reshd server executes requested programs and determines the local name of the file produced by the execution. These files are stored in the directory /remix/output.

i. (4 points) What would the resh HTTP request to execute a program look like and what would the reshd HTTP response look like for the command:

```
resh> www.cs.vt.edu/cs3214/ex/cmd
```

Be detailed in your answers. Use <CRLF> to indicate a pair of \r \l characters

```
POST /cs3214/ex/cmd HTTP/1.1<CRLF>
Host: www.cs.vt.edu <CRLF>
Connection: close<CRLF>
<CRLF>

HTTP/1.1 200 OK
Connection: close
Content-Length: 20
Content-Type: text
<CRLF>
/remex/output/file
```

ii. (3 points) Show the HTTP request for resh to retrieve the file when the remote execution is done.

```
Get /remex/output/file HTTP/1.1<CRLF>
Host: www.cs.vt.edu <CRLF>
Connection: close<CRLF>
<CRLF>
```

Version 2 of resh allow the data stream of STDOUT of the remote process to be the data stream of the STDIN of a local process. For example, the command line might look like:

>>resh remote | local

where the pipe symbol "|" denotes the connection of the data streams between the remote process and the local process.

iii. (8 points) Describe the important steps taken by resh and reshd.

Critical steps are: (1) the exchange of a port number between resh and reshd; the exchange can be initiated by either side; (2) opening a socet connection between resh and reshd; (3) reshd forks/execs “remote” with the socket’s file descriptor as the STDOUT; (4) resh forks/execs “local” with the socket’s file descriptor as the STDIN

5. Essay Question: The NSF/IEEE TCPP Curriculum (20 points)

In a recent paper [1] the authors wrote that:

"Computer programming has never been easy, and the cost of errors has always been high. Software failures have claimed lives, and expensive software project failures are the stuff of industry legend. Over time, however, improvement in programming languages, development tools, and education have ameliorated the difficulties of ordinary serial programming. Average programmers circa 2004 were as productive and competent as their counterparts in other engineering domains. Recent hardware trends, however, threaten to erode software dependability, programmer productivity, and the industry's rate of economic value creation."

Consider your learning experience in the Introduction to Computer Systems class. Discuss the above projection! Do you agree or disagree? Justify your opinion!

Note: *This question will be graded both for content/correctness of your technical points (12 pts) and for your ability to communicate effectively in writing (8 pts). Your answer should be well-written, organized, and clear.* ***Your answer must be legible – points will be deducted for parts that cannot be read with normal effort. Use the back of this page as needed.***

[1] Terence Kelly, Yin Wang, Stéphane Lafortune, and Scott Mahlke. 2009. Eliminating Concurrency Bugs with Control Engineering. *Computer* 42, 12 (December 2009), 52-60. DOI=10.1109/MC.2009.391 http://dx.doi.org/10.1109/MC.2009.391

(1) a description of the "recent hardware trends". An average answer will explain this to mean the rise of multi-core architectures. A great answer will also mention factors (density, thermal properties) being the driving forces for this change. A poor answer will give only vague information.

(2) recognition that architectures are leading to greater use of concurrent programming. An average answer will note that more cores imply more threads. A great answer will also note that the cores are growing in number but not in speed. A poor answer will give only vague description.

(3) statement of the difficulty of concurrent programming. An average answer will say that concurrent programming requires synchronization which is difficult to arrange. A great answer will use correct terminology to describe problems of atomiticy and order violations, the need for mutual exclusion, etc. A poor answer will not clearly state why concurrent programming is more difficult than sequential programming.