**Operating Systems**

**Spring 2016 – Quiz 2**

**Question 1:**

The following code compiles with no errors or warnings (header files not shown for space).

#define THREADS 10

void \*print\_hello( void \*tid )

{

int \*id = (int \*) tid;

printf( "Hello from thread %d.\n", \*id );

pthread\_exit( NULL );

}

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

{

pthread\_t threads[THREADS];

int status, i;

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

{

printf( "Creating thread %d.\n", i );

status = pthread\_create( &threads[i], NULL,

print\_hello, (void \*) &i );

if ( status != 0 )

exit( -1 );

}

pthread\_exit( 0 );

}

Almost every time this program is run, it prints different output. One time it was run, the output was:

Creating thread 0.

Creating thread 1.

Creating thread 2.

Creating thread 3.

Creating thread 4.

Creating thread 5.

Creating thread 6.

Creating thread 7.

Creating thread 8.

Creating thread 9.

Hello from thread 1.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

Hello from thread 10.

* **Explain** how/why this output was produced.

**Question 2:**

The following code compiles with no errors or warnings (header files not shown for space).

int main()

{

char \*sentences;

int a;

sentences = (char \*) malloc(sizeof(char)\*10);

for (a = 0; a < 12; a++){

sentences[a] = 'b' + a;

}

for (a = 0; a < 12; a++){

printf( "%c\n", sentences[a] );

}

printf("\n");

sentences = (char \*) malloc(sizeof(char)\*5);

for (a = 0; a < 5; a++){

sentences[a] = 'l' + a;

}

for (a = 0; a < 10; a++){

printf( "%c\n", sentences[a]);

}

free(sentences);

}

* There are several problems in this code. Can you find them all?
* What will this program do?
* What will this program print out?
* Where is a memory leak in this program?
* What is freed with the free() command at the end of the code?

**Question 3:**

One of the things that makes working with threads convenient is that all the threads in a process share the same **heap** area, so that they can easily share data. However, each thread has its own private **stack** area. Explain the reason for this (not sharing the stack)?

**Question 4:**

The following code compiles with no errors or warnings (header files not shown for space).

int main()

{

int pid;

int status, i;

printf( "bird\n" );

pid = fork();

if ( pid != 0 )

{

waitpid( pid, &status, 0 );

pid = fork();

printf( "dog\n" );

}

else

{

printf( "fish\n" );

execlp( "date", "date", NULL );

printf( "cow\n" );

}

printf( "end\n" );

}

* What will this program print out?

**Question 5:**

**Explain** the difference between the 2 sockets, **ssock** and **msock**, referenced in the line of server code below.

**ssock** = accept( **msock**, (struct sockaddr \*)&fsin, &alen );

**Question 6:**

Describe what the output of the following shell commands would be:

ls

ls –alrt

ls -alrt | grep “.png”

cat file |grep “string”