CS241 Lecture 13  
Working With pThreads. Introducing mutex locks

1. // downloads a web resource in the background
2. void\* download(void\*url) {
3. void\* mem = malloc(2048);
4. ... cs241 network magic to download file
5. FILE\* file = fopen(shortname,"w");
6. if(file&&bytes) fwrite(mem, bytes,1, file);
7. fclose(file);
8. free(mem);
9. return NULL;
10. }
11. int main() {
12. pthread\_t tid1,tid2;
13. pthread\_create(&tid1, NULL, download, "https://en.wikipedia.org/wiki/Spanish\_dollar");
14. pthread\_create(&tid2, NULL, download, "...1888\_México\_8\_Reals\_Trade\_Coin\_Silver.jpg");
15. // 2 ways to wait for threads to complete?

1. You call malloc from two threads?

Yes because it is "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

2. Why is it that *mem* will point to two different heap areas?

3. Which one of these is also safe to be used by two threads?

char \*strerror(int errnum);

int strerror\_r(int errnum, char \*strerrbuf, size\_t buflen);

4. Complete this code to print the thread id and an initial starting value. What does this code actually print? Why?

|  |
| --- |
| 1. void\* myfunc(void\*ptr) { 2. printf("My thread id is %ld   and I’m starting at %d\n",   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); 3. return NULL; 4. } 5. int main() { 6. // Each thread needs a different value of i 7. pthread\_t tid[10]; 8. for(int i =0; i < 10; i++) { 9. pthread\_create(& tid[i], 0, myfunc, &i); 10. } |

5 What is a critical section?

6 What is a mutex?

7 What are the two ways to create a mutex?

8 How do you lock and unlock a mutex?

9 When can you destroy a mutex?

10. What does this code print? Will it always print the same output?

|  |
| --- |
| 1. int counter; 2. void\*myfunc2(void\*param) { 3. int i=0; // stack variable 4. for(; i < 1000000;i++) counter ++; 5. return NULL; 6. } 7. int main() { 8. pthread\_create(&tid1, 0, myfunc2, NULL); 9. pthread\_create(&tid1, 0, myfunc2, NULL); 10. pthread\_join(tid1,NULL); 11. pthread\_join(tid2,NULL); 12. printf("%d\n", counter ); 13. } |

11 Common pattern: Use heap memory to pass starting information to each thread.

Example: Create two threads. Each thread will do half the work. The first thread will process 0..numitems/2 in the array. The second thread will process the remaining items. Any gotchas?

|  |
| --- |
| 1. typedef struct work\_ { 2. } work\_t; 3. void calc (int \* data, size\_t nitems) { 4. size\_t half = numitems/2; 5. pthread\_create(&tid1, 0, imagecalc,\_\_\_\_); 6. } 7. // Gotchas odd number of numitems. Memory leak? |