CS241 #14  
Working With threads and locks

1. Would you expect the following to work on your 64 bit VM?

(How about a 32bit machine?)

|  |
| --- |
| 1. int bad = (int) "Hello"; 2. puts( (char\*) bad); |

2. Which of the following calls will block?

pthread\_mutex\_init

pthread\_mutex\_lock

pthread\_mutex\_unlock

pthread\_mutex\_destroy

2b. You call to *pthread\_mutex\_X* (what is X?) blocks. When will it return i.e. when will it unblock?

2c. Why might pthread\_mutex\_X not block?

3. Where are the *critical sections* in the following two code examples?

Fix any errors you notice.

Modify the code to be thread safe

|  |
| --- |
| 1. link\_t\* head; 2. void list\_prepend(int v) { 3. link\_t\* link = malloc( sizeof(link\_t\*)); 4. link -> value = v; 5. link -> next = head; 6. head = link; 7. } 8. int list\_remove\_front() { 9. link\_t\* link = head; 10. int v = link ->value; 11. head = link->next; 12. free(link); 13. return v; 14. } |

*4. Meanwhile the code continued… (check for errors)*

|  |
| --- |
| 1. size\_t capacity = 64; 2. size\_t size = 0; 3. char\*\* data = malloc(capacity); 4. void push(char\*value) { 5. if(size == capacity) { 6. capacity \*= 2; 7. realloc(data,capacity); 8. } 9. data[size++] = value; 10. } 11. char\* pop() { 12. char\* result = data[--size]; 13. return result; 14. } |

5. Lock Contention and likelihood of discovering race conditios

A thread at a random time executes for 1ms code inside an unprotected critical section with 1s total running time. If there are now 2 threads that run for 1second each, estimate the probability of both threads in the critical section at the same time.

6. Remember me?Notice any mistakes? What will happen exactly?

|  |
| --- |
| 1. pthread\_t tid1,tid2; 2. pthread\_mutex\_t m; 3. int counter; 4. void\* myfunc2(void\*param) { 5. int i = 0; // stack variable 6. for(; i < 1000000;i++) { 7. pthread\_mutex\_lock( &m ); 8. counter ++; 9. } 10. return NULL; 11. } 12. int main() { 13. pthread\_create(&tid1, 0, myfunc2, NULL); 14. pthread\_create(&tid2, 0, myfunc2, NULL); 15. pthread\_join(tid1,NULL); 16. pthread\_join(tid2,NULL); 17. printf("%d\n", counter ); 18. } |

7. Case study1: Critical Sections and functions that are not thread safe

1. static FILE\* file;
2. void logerror(int errnum, char\*mesg) {
3. char\* error = strerror(errnum);
4. if(!file) {
5. file = fopen("errorlog.txt","a+");
6. }
7. fprintf(file,"%s:%s", mesg, error);
8. fflush(file)
9. }

8. Meet your next *Synchronization Primitive*: What is a  
 *Counting Semaphore*?

9. Case study2: Parallelize *AngraveCoin* miner for fun and profit!

|  |
| --- |
| void search(long start, long end) {  printf("Searching from 0x%lx to 0x%lx\n", start , end);  for(long i = start; i < end; i++) {  char message[100];  sprintf(message,"AngraveCoin:%lx", i);    unsigned char \*hash; // 256 bit result ( = 32 bytes )  hash = SHA256(message, strlen(message), NULL);  int iscoin; // first three bytes must be zero  iscoin =(hash[0]==0)&&(hash[1]==0)&&(hash[2]==0);  if(iscoin)  printf("%lx %02x %02x %02x '%s'\n",   i, hash[0],hash[1],hash[2] , message);  }  printf("Finished %lx to %lx\n", start, end);  }  // I want to speed up search of 233 possible coins  long array[] = {0L, 1L <<25, 1L <<27, 1L <<33};  int main() {  search(array[0], array[1]);  search(array[1], array[2]);  search(array[2], array[3]);  return 0;  } |