what do you think my costume is this year?

clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Threading oblo

Updates

1. MP5 Allocator due today

3:15 – 6:00 4 people

6:00 – 9:00 wait longer

2. MP6 Wallet out today, due in 2 weeks

3. HW 5 due Thursday at 2:00pm

4. Exam 2 next Tuesday

no class

CBTF – study guide

coding @ Thursday

Agenda

1. MP6 - Wallet, related threading

2. Daixuan with AI systems

Does this code have a race condition?

```
6  int balance = 0;
7
8  void *transaction(void *arg) {
9      int delta = *((int*)arg);
10     balance += delta;
11     if(balance < 0) {
12         //$10 fee for negative balance
13         balance = -10;
14     }
15 }
16
17 int main() {
18  pthread_t threads[5];
19  int transactions[5] = {40, -5, -10, -4, -10};
20
21  for(int i = 0; i < 5; i++){
22     pthread_create(&threads[i], NULL, transaction, &transactions[i]);
23  }
24  for(int i = 0; i < 5; i++){
25     pthread_join(threads[i], NULL);
26  }
27  printf("Your balnce is now, %i", balance);
28 }
```

5 threads

← 1 thread

clicker.cs.illinois.edu

Q2

~Code~
340

Mutex

Definition - mutal exclusion lock — 1 thread

Pattern

Create mutex object

get lock ← mutex

critical

release lock ← mutex

Mutex Examples

```c
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t my_m = PTHREAD_MUTEX_INITIALIZER;
```

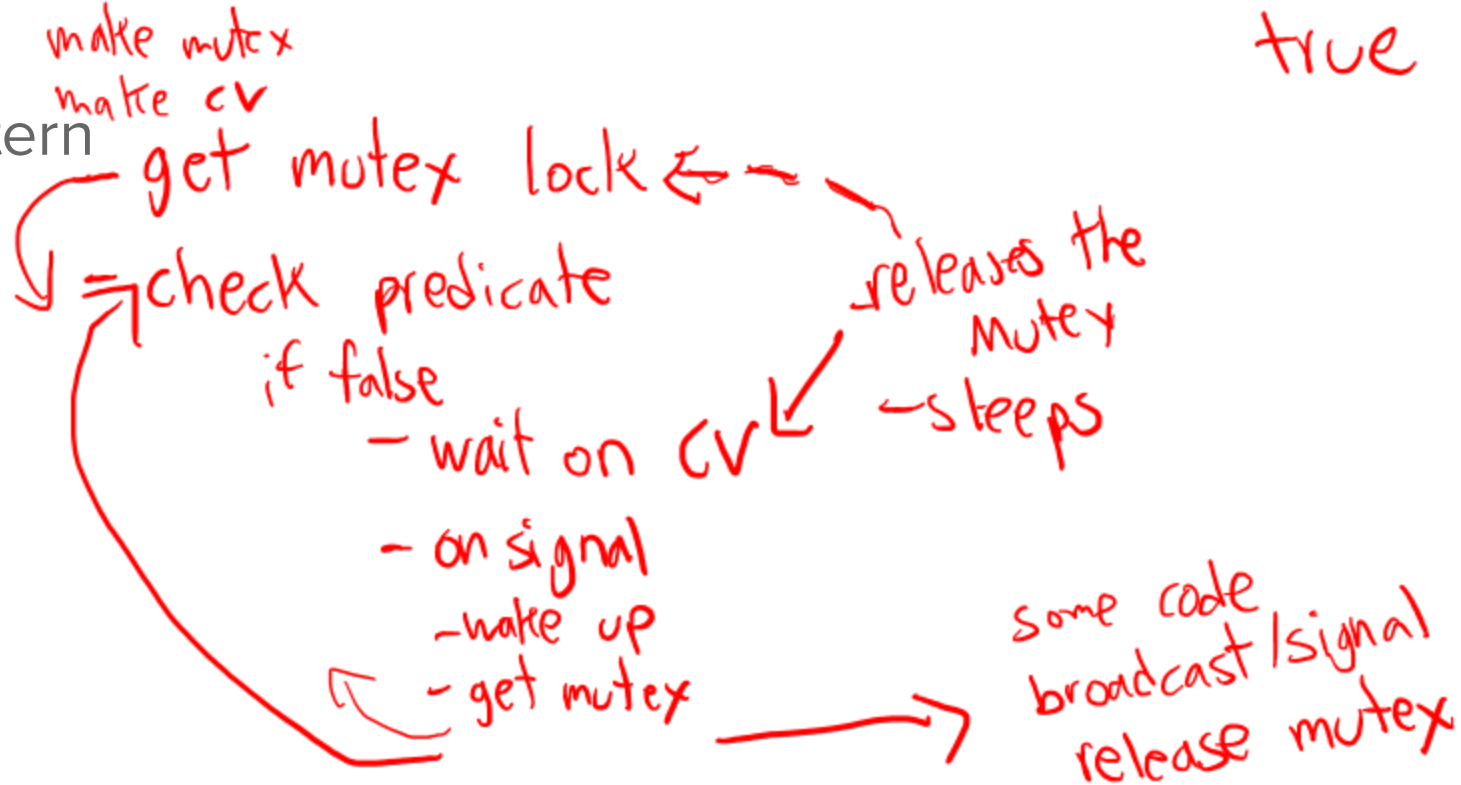
```c
int shared_int = 0;

void *shared_function(void* arg){
    pthread_mutex_lock(&my_m);
    shared_int += 3;
    printf("my val: %i", shared_int);
    shared_int = 0;
    pthread_mutex_unlock(&my_m);
}

//thread creation code and main function
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

pthread_mutex_t* my_mutex;

void init_mutex(){
    my_mutex = malloc(sizeof(pthread_mutex_t));
    pthread_mutex_init(my_mutex, NULL);
}

void destroy_mutex(){
    pthread_mutex_destroy(my_mutex);
    free(my_mutex);
}

//code that inits the mutex, uses it, then destroys it
```

Condition Variable

Definition - lets a thread sleep until a predicate true

Pattern



Condition Variable Examples

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

pthread_mutex_t my_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t my_cv = PTHREAD_COND_INITIALIZER;

int shared_int = 0;
int pred = 0;

void *shared_function(void* arg){
    pthread_mutex_lock(&my_mutex);
    while(pred == 0){
        pthread_cond_wait(&my_cv, &my_mutex);
    }
    shared_int = pred * 5;
    pred++;
    pthread_cond_broadcast(&my_cv);
    pthread_mutex_unlock(&my_mutex);
}

//thread creation code and main function, changes pred to 1 eventually
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

pthread_mutex_t my_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t* my_cv;

int shared_int = 0;
int pred = 0;

void init_cv(){
    my_cv = malloc(sizeof(pthread_cond_t));
    pthread_cond_init(my_cv, NULL);
}

void destroy_cv(){
    pthread_cond_destroy(my_cv);
    free(my_cv);
}

//code inits my_cv, runs code, then destroys cv
```

If I have an array of 5 items initialized at compile time and I want to protect them from race conditions could I use the static initialization method as shown below?

clicker.cs.illinois.edu

Q3

~Code~
340

```
5  pthread_mutex_t m1 = PTHREAD_MUTEX_INITIALIZER;
6  pthread_mutex_t m2 = PTHREAD_MUTEX_INITIALIZER;
7  pthread_mutex_t m3 = PTHREAD_MUTEX_INITIALIZER;
8  pthread_mutex_t m4 = PTHREAD_MUTEX_INITIALIZER;
9  pthread_mutex_t m5 = PTHREAD_MUTEX_INITIALIZER;
```

0

yes

3 threads

arr[2]

1 thread

arr[0]

If I have a linked list data structure and I want to protect each datum from race conditions, could I use the static initialization method as shown below?

clicker.cs.illinois.edu

Q4

~Code~

340

```
5  pthread_mutex_t m1 = PTHREAD_MUTEX_INITIALIZER;
6  pthread_mutex_t m2 = PTHREAD_MUTEX_INITIALIZER;
7  pthread_mutex_t m3 = PTHREAD_MUTEX_INITIALIZER;
8  pthread_mutex_t m4 = PTHREAD_MUTEX_INITIALIZER;
9  pthread_mutex_t m5 = PTHREAD_MUTEX_INITIALIZER;
```

NO

Static does not work for a dynamic DS

Linked List Example

```
struct node {
    struct node *next;
    int datum;
    pthread_mutex_t *m
};
```

insert val

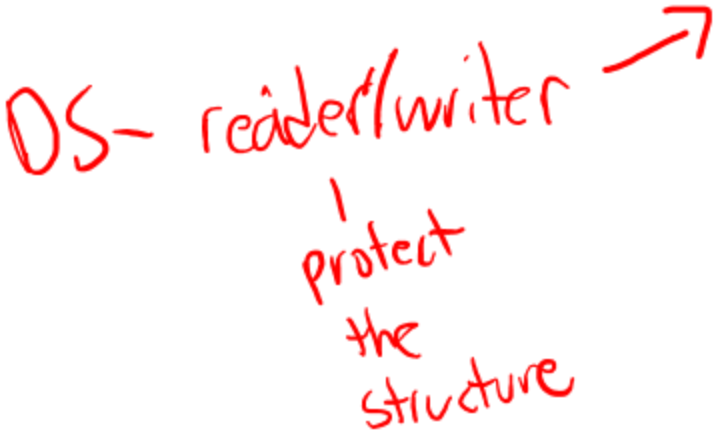
```
node *new_n = malloc(sizeof(node));
new_n->datum = val;
new_n->m = malloc(sizeof(pthread_mutex_t));
pthread_mutex_init(new_n->m, NULL);
```

pop off

```
pthread_mutex_destroy(old_n->m);
free(old_n->m);
free(old_n);
```

Reader Writer Lock

Many can read, only 1 can write



Reader Writer Lock Examples

```
5  typedef struct node {
6    struct node *next;
7    //could add mutex
8    int data;
9  } node;
10
11 typedef struct {
12   node *head;
13   pthread_rwlock_t *rwlock;
14 } my_list;
15
16 void my_list_init(my_list *list){
17     list->head = NULL;
18     list->rwlock = malloc(sizeof(pthread_rwlock_t));
19     pthread_rwlock_init(list->rwlock, NULL);
20 }
21
```

```
22 int find_element(my_list *list, int index){
23     pthread_rwlock_rdlock(list->rwlock);
24     //read things
25     pthread_rwlock_unlock(list->rwlock);
26 }
27
28 void add_element(my_list *list){
29     pthread_rwlock_wrlock(list->rwlock);
30     //change the structure of the list
31     pthread_rwlock_unlock(list->rwlock);
32 }
33
34 void my_list_destroy(my_list *list){
35     //delete all nodes and mutex's
36     pthread_rwlock_destroy(list->rwlock);
37 }
```

add_1

reader lock

reader unlock

Wallet

40% - map thread safe

reader writer lock

60% wallet_use

CV, mutex

Luther

made a helper function

m_setdefault2

Jule all in wallet_use