

MP4 Overview Session

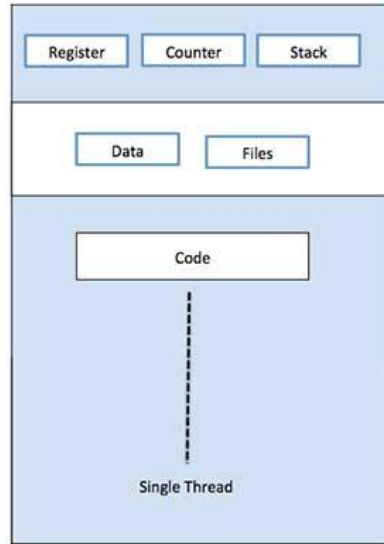
CS 340 - Introduction to Computer Systems

Goals of the MP

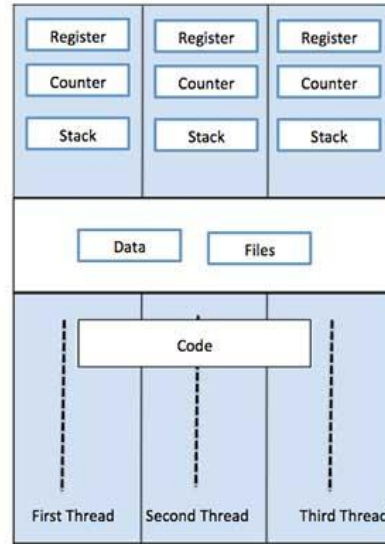
- Write multithreaded programs in C
 - Create thread safe data structures
 - Use mutex, condition variable, etc
- Implement a wallet that will hold resources

Multithreading Overview

- **What is a thread?:** A thread is a path of execution within a process
- **Can a process have multiple threads:** Yes, there can be multiple threads within a single process. A process usually starts with only the main thread



Single Process P with single thread



Single Process P with three threads

Synchronization

- **Race Condition:** A thread touches a piece of shared memory at the same time as another thread
- **Critical Section:** A piece of shared memory that only one thread should be able to access at a time

A 🛍️ with 40 💎	
Thread 1: Reads 40 💎	Thread 2: Reads 40 💎
Withdraws 40 💎	
0 💎 in the wallet	Adds 20 💎
	60 💎 in the wallet
The wallet should have 20 💎, not 60 💎	


Mutex

Programming in C language

- `pthread_mutex_init`: create a mutex
- `pthread_mutex_destroy`: destroy a mutex
- `pthread_mutex_lock`: block execution for all other threads trying to acquire mutex
- `pthread_mutex_unlock`: unblock execution for all other threads trying to acquire mutex and allow another thread use the mutex

Avoid Busy Waiting

- Avoid checking **repeatedly** if a condition is met
 - Don't keep checking if your wallet has a positive balance for a resource to allow you to withdraw
- This can cause issues with **race conditions**
- Busy waiting will waste **system resources**

```
while( is not positive) {  
    // loop until condition is true  
}
```

Condition Variable

Programming in C language

- **pthread_cond_init:** create a condition variable
- **pthread_cond_destroy:** destroy a condition variable
- **pthread_cond_wait:** release a mutex and block on the current thread using the condition variable
- **pthread_cond_signal:** unblock at least one thread that is blocked on a condition variable
- **pthread_cond_broadcast():** unblock all threads that are blocked on a condition variable

Spurious Wakeup

- **Spurious Wakeup:** A thread may randomly wake up for no reason
- This can happen where another thread **changes** the condition before the waiting thread runs
- We want to call `pthread_cond_wait` in a loop to avoid issues with **spurious wakeup**

```
// mutex is locked  
while(💎 is not positive) {  
    pthread_cond_wait()  
}  
  
// finish thread task
```


MP4 Functions

Implementing the Resource Manager

Structs in wallet.h

- `wallet_t` - maintain the `state` of the wallet
- `wallet_resource` - represent the `resource` in a wallet
- Feel free to add any other variables to these structs

Wallet Functions

- `wallet_init()` - initialize the wallet structure
 - The wallet is initially empty with no resources
- `wallet_get()` - return the amount of a given resource
 - Remember to ensure that access to the wallet is thread-safe
- `wallet_change_resource()` - change the amount of a resource by a given delta
 - The amount of resource **can not** go negative
 - The thread must wait until the request can be satisfied (**avoid busy waiting**)
- `wallet_destroy()` - free any memory associated with the wallet structure

Memory Correctness

- You do not have write any additional code for this part.
- Your code need to run “**valgrind clean**”:
 - Zero memory error, no memory leak
 - free() any memory allocated with malloc/calloc
 - fclose() any file opened with fopen

All heap blocks were freed -- no leaks are possible

- Valgrind does not work on macOS. Use it with a **Docker** container.