CS 35L Software Construction Lab
Week 6 – Multithreading

Threads

- A flow of instructions, path of execution within a process
- It is a basic unit of CPU utilization
- Each thread has its own:
  - Stack
  - Registers
  - Thread ID
- Each thread shares the following with other threads belonging to the same process
  - Code
  - Heap
  - Global Data
  - OS resources (files,I/O)
- A process can be single-threaded or multi-threaded
- Threads in a process can run in parallel

  (provide another type of parallelism)

Thread safety/synchronization

- **Thread safe function** - safe to be called by multiple threads at the same time. Function is free of 'race conditions' when called by multiple threads simultaneously.
- **Race condition** - the output depends on the order of execution
  - Shared data changed by 2 threads
    - int balance = 1000
  - Thread 1
    - T1 - read balance
    - T1 - Deduct 50 from balance
    - T1 - update balance with new value
  - Thread 2
    - T2 - read balance
    - T2 - add 150 to balance
    - T2 - update balance with new value

Thread safety/synchronization

- Order 1
  - balance = 1000
  - T1 - Read balance (1000)
  - T1 - Deduct 50
    - 950 in temporary result
  - T2 - read balance (1000)
  - T1 - update balance
    - balance is 950 at this point
  - T2 - add 150 to balance
    - 1150 in temporary result
  - T2 - update balance
    - balance is 1150 at this point
  - **The final value of balance is 1150**
- Order 2
  - balance = 1000
  - T1 - read balance (1000)
  - T2 - read balance (1000)
  - T2 - add 150 to balance
    - 1150 in temporary result
  - T1 - Deduct 50
    - 950 in temporary result
  - T2 - update balance
    - balance is 1150 at this point
  - T1 - update balance
    - balance is 950 at this point
  - **The final value of balance is 950**

Thread synchronization

- Only one thread will get the mutex. Other thread will **block in Mutex.lock()**
- Other thread can start execution only when the thread that holds the mutex calls **Mutex.unlock()**

Thread synchronization

- **Mutex (mutual exclusion)**
  - Thread 1
    - Mutex.lock()
      - Read balance
      - Deduct 50 from balance
      - Update balance with new value
    - Mutex.unlock()
  - Thread 2
    - Mutex.lock()
      - Read balance
      - Add 150 to balance
      - Update balance with new value
    - Mutex.unlock()
  - balance = 1100

Pthread in C

- compile with -pthread (or -lpthread) compiler option

-tells the compiler that your program requires threading support

- include pthread.h in program header

Basic pthread Functions

There are 5 basic pthread functions:

1. **pthread_create:** creates a new thread within a process

2. **pthread_join:** waits for another thread to terminate

3. **pthread_equal:** compares thread ids to see if they refer to the same thread

4. **pthread_self:** returns the id of the calling thread

5. **pthread_exit:** terminates the currently running thread

pthread_create

- **Function:** starts a new thread in the calling process
- Return value:
  - Success: zero
  - Failure: error number

Parameters

int pthread_create(pthread_t *tid, const pthread_attr_t *attr, void *(my_function)(void *), void *arg);

- **tid**: unique identifier for newly created thread
- **attr**: object that holds thread attributes (priority, stack size, etc.)
  - Pass in NULL for default attributes
- **my_function**: function that thread will execute once it is created
- **arg**: a *single* argument that may be passed to my_function
  - Pass in NULL if no arguments

pthread_create Example

```
#include <pthread.h> …
void *printMsg(void *thread_num) {
        int t_num = (int) thread_num;
        printf("It's me, thread #%d!\n", t_num); }

int main() {
        pthread_t tids[3];
        int t;
        for(t = 0; t < 3; t++) {
                ret = pthread_create(&tids[t], NULL, printMsg, (void *) t);
                if(ret) {
                        printf("Error creating thread. Error code is %d\n", ret");
                        exit(-1); }
        }
}
```

Possible problem with this code?

If main thread finishes before all threads finish their job -> incorrect results

pthread_join

- **Function:** makes originating thread wait for the completion of all its spawned threads' tasks
- Without join, the originating thread would exit as soon as it completes its job

  ⇒A spawned thread can get aborted even if it is in the middle of its chore
- Return value:

  ⇒Success: zero

  ⇒Failure: error number

Arguments

int pthread_join(pthread_t tid, void **status);

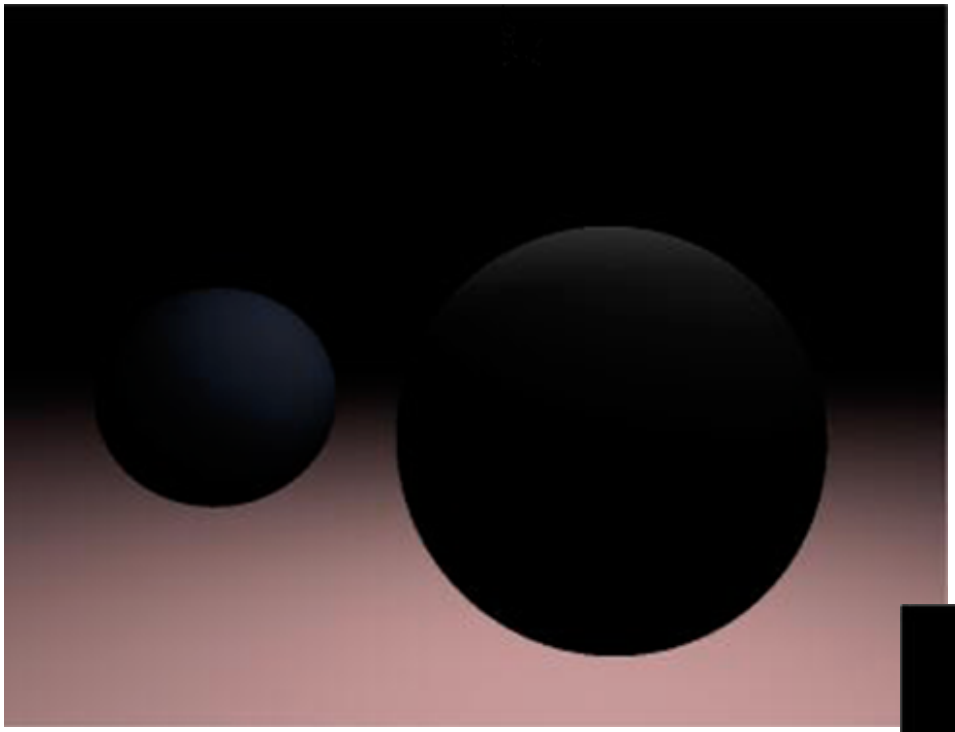
- **tid**: thread ID of thread to wait on
- **status:** the exit status of the target thread is stored in the location pointed to by *status
  - Pass in NULL if no status is needed

pthread_join Example

```
#include <pthread.h> …
#define NUM_THREADS 5

void *PrintHello(void *thread_ num) {
        printf("\n%d: Hello World!\n", (int) thread_num); }

int main() {
        pthread_t threads[NUM_THREADS];
        int ret, t;
        for(t = 0; t < NUM_THREADS; t++) {
                printf("Creating thread %d\n", t);
                ret = pthread_create(&threads[t], NULL, PrintHello, (void *) t);
                // check return value }

         for(t = 0; t < NUM_THREADS; t++) {
                ret =  pthread_join(threads[t], NULL);
                // check return value }
}
```

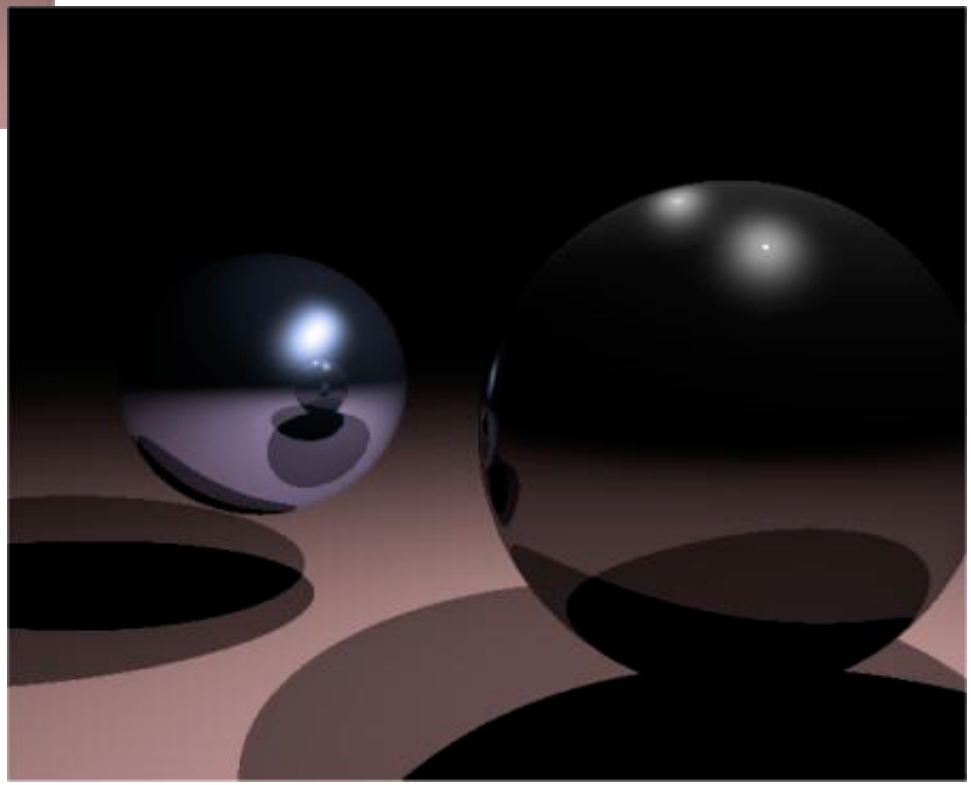
Ray Tracing

- An advanced computer graphics technique for rendering 3D images
- Mimics the propagation of light through objects
- Simulates the effects of a single light ray as it's reflected or absorbed by objects in the images



Without ray tracing

With ray tracing



Computational Resources

- Ray Tracing produces a very high degree of visual realism at a high cost
- The algorithm is *computationally intensive*

=> Good candidate for multithreading (embarrassingly parallel)

Homework 6

- Download the single-threaded ray tracer implementation
- Run it to get output image
- Multithread ray tracing
  - Modify main.c and Makefile
- Run the multithreaded version and compare resulting image with single-threaded one

Homework 6

- Build a multi-threaded version of Ray tracer
- Modify “main.c” & “Makefile”
  - Include <pthread.h> in “main.c”
  - Use “pthread_create” & “pthread_join” in “main.c”
  - Link with –lpthread flag
- make clean check
  - Outputs “1-test.ppm”
  - Can’t see “1-test.ppm”
    - sudo apt-get install gimp (Ubuntu)
    - X forwarding (lnxsrv)
    - gimp 1-test.ppm

1-test.ppm

Figure. 1-test.ppm
& baseline.ppm