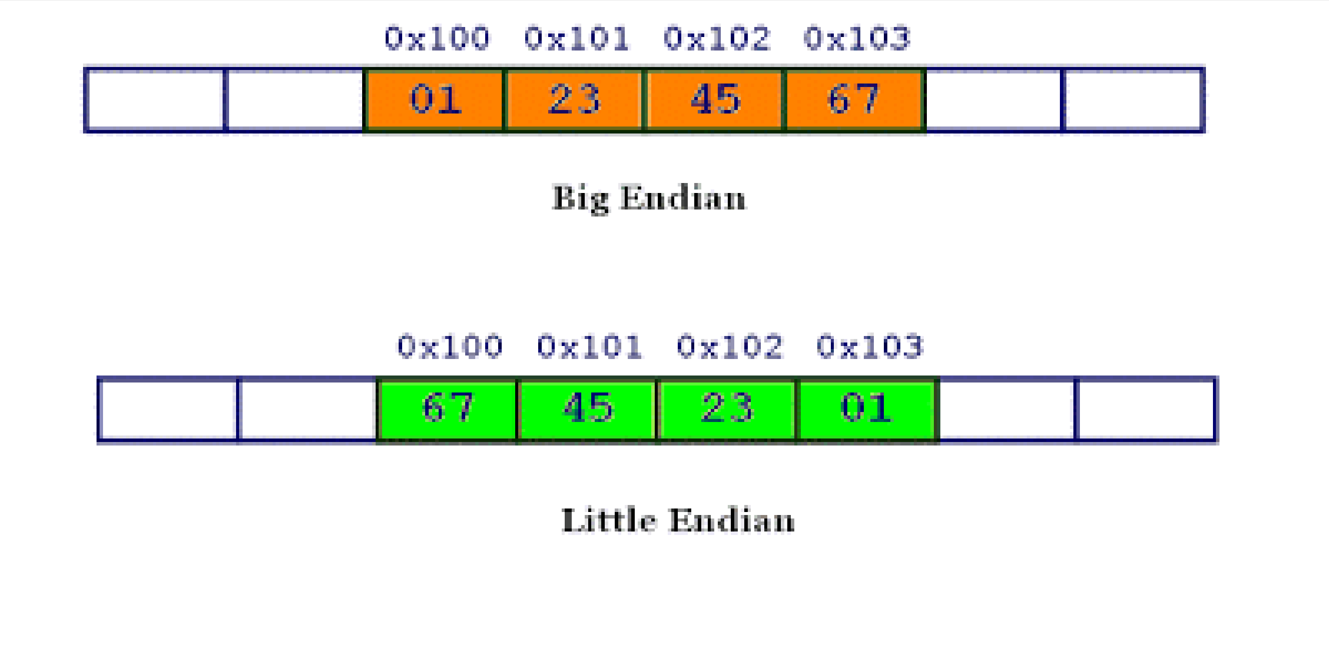
CS35L – Spring 2019

Slide set:	6.2
Slide topics:	Multithreaded Programming
Assignment:	6

Endianness



		0x100	0x101	0x102	0x103		
		01	23	45	67		

Big Endian

		0x100	0x101	0x102	0x103		
		67	45	23	01		

Little Endian

Systems for Representing Binary Numbers

- Most-significant bit
- 1’s complement
- 2’s complement
- Representing decimal numbers in different base systems

Note: refer to this link

Semaphores and Mutexes

Refer to this link and class notes/examples

- Solution to the race condition

Lab 6

- Evaluate the performance of multithreaded sort
- Add /usr/local/cs/bin to PATH
  - $ export PATH=/usr/local/cs/bin:$PATH
- Generate a file containing 10M random single-precision floating point numbers, one per line with no white space
  - /dev/urandom: pseudo-random number generator

Disk quota exceeded

- ◦ http://www.seasnet.ucla.edu/seasnet-account-quotas/

Lab 6

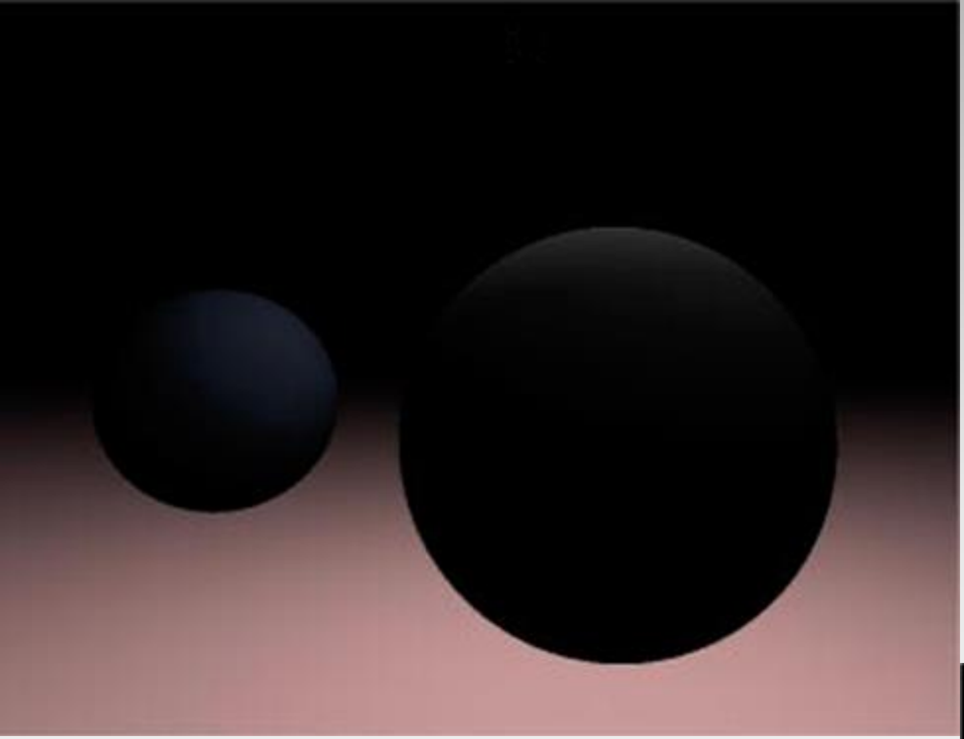
- od (refer to this guide)
  - write the contents of its input files to standard output in a user-specified format
  - Options
    - -t : select output format
    - -N <count>: Format no more than *count* bytes of input
- sed, tr
  - Remove address, delete spaces, add newlines between each float

Lab 6

- use `time -p` to time the command `sort -g` on the data you generated
- Send output to */dev/null*
- Run `sort` with the `--parallel` option and the `-g` option: compare by general numeric value
  - Use `time` command to record the real, user and system time when running sort with 1, 2, 4, and 8 threads
    - $ `time -p sort -g file_name > /dev/null` (1 thread)
    - $ `time -p sort -g --parallel=[2, 4, or 8] file_name > /dev/null`
  - Record the times and steps in log.txt

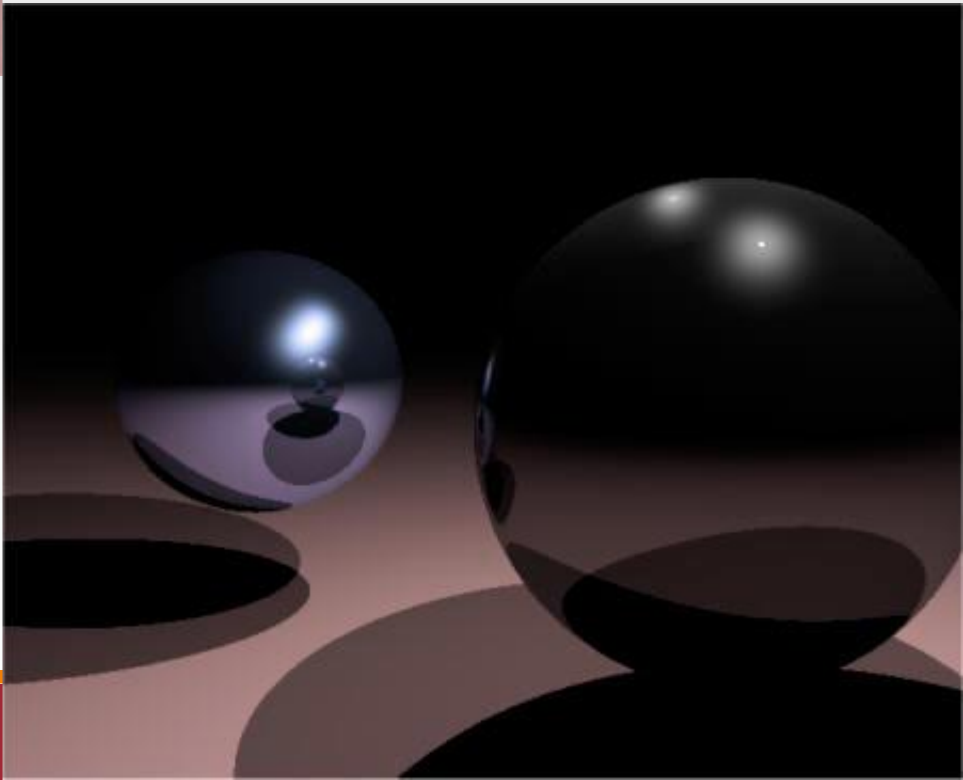
Ray Tracing

- 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
- Run the multithreaded version and compare resulting image with single-threaded one

Basic pthread Functions

include <*pthread.h*> and link with the *-lpthread* library

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:** creates a new thread and makes it executable
- Can be called any number of times from anywhere within code
- 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 completed 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

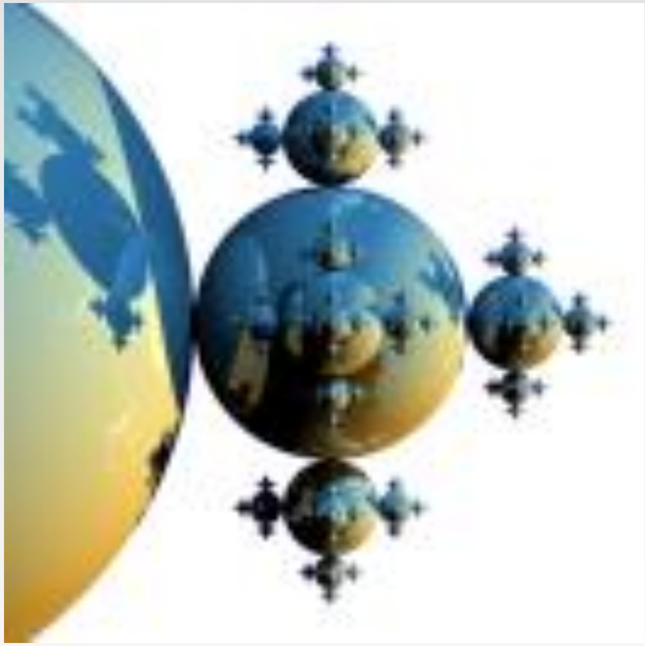
pthread_join Example

```
#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 errors }

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

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 (LDLIBS target in Makefile)
- make clean check
  - Outputs “1-test.ppm”
  - Can see “1-test.ppm” in GIMP/Image viewer

baseline.ppm & 1-test.ppm



Exercise 1

- You have an int array of size 100 . Write a sequential program to set all values to 10 in this array.
- Now change the program using the pthread library to do the same thing with 4 threads.
- Now change the program to write the values 1 to indices 0 to 24. 2 to indices 25-49, 3 to indices 50-74 and 4 to indices 75-99.
- Now change the program to equally divide the input based on the number of threads and do the same as above.

Exercise 2

- You have a 2d int array of dimensions 60 x 60. Write a sequential program to set all values to 10 in this array.
- Now change the program using the pthread library to do the same thing with 4 threads.
- Now change the program to write values 1 to first 15 columns, 2 to second 15 columns and so on.
- Now change the program to equally divide the input based on the number of threads and do the same as above.