Computer Science 311

Assignment 5 Write-up

Austin Dubina

1 Program Design

Overview:
Program Design Overview: The intent of this program was to find all prime numbers within the range of a 32 bit unsigned integer using parallelized processes and threads. In addition, all prime numbers needed to be stored using a bitmap, in this case an array of characters.

After doing quite a bit of research on bitmaps and attending a few office hour sessions, I concluded that my design was going to require bit masking, bit setting, and possibly a bit clearing functions. I quickly made use of the macros described on c-faq.com (http://c-faq.com/misc/bitsets.html) for my bit-setting functions.

Both the threaded and process portion of my program take advantage of the fact that each thread / process can modify the character array simultaneously without having to use any synchronization. I did this simply because it is much faster than having to lock and lock semaphore / mutexes. I accomplished this by passing a begin and end offset to each thread / process. Each thread / process finds all the primes within the range of numbers specified by my offsets.

Originally, I thought I was going to be able to reuse my marking algorithm from the first assignment in this project however this was not the case, in fact redesigning my algorithm to work with a segmented array of characters proved to be the most challenging part of this assignment! While although my program correctly marks all the primes within the range of a 32 bit unsigned integer, I first had to mark all non-base prime numbers up to the square root of n, where n is the max number of an 32 bit unsigned integer. While this number is small, roughly 50 million, it makes my program significantly slower because it is a serialized operation.

Once all the non-base prime numbers have been marked by the initial function in main, my program will then create a number of threads or fork a number processes, depending on what program the user is running, and begin calculating the number of primes for their segmented portion of the array.

I avoided using any mutexs in the threaded version of my program by simply passing an integer initialized at zero with my structure of arguments to sum the number of primes found by each thread. Once all threads have exited and have been joined, main sums each threads count of primes to get a grand total which is then printed to the terminal. For the process version of my program, I avoided using semaphores by simply having each process mark their segment of the array. Once each process exits and main counts the number of primes and prints to the terminal.

2 Additional Questions

1. what do you think the main point of this assignment is?
   I think the main intent of this assignment was to continue our learning of parallelized processing by comparing and incorporating the differences between forking processes and creating threads for extremely large numbers. Program design played a big factor on run times and execution because of the large size of data being processed.

2. how did you ensure your solution was correct? Testing details, for instance.
   I systematically went through each function to test every imaginable test case. As part of my check, I put error checking on most system calls and functions.

3. what did you learn?
   What did you learn? I have painfully learned a lot during this assignment. The most important thing I learned has nothing to with the assignment itself, but how program in general. After completing several assignments for this class and spending over 40 hours on this project alone, I have learned that brute forcing my way through these assignments does not work. Far too often I find myself going around in circles after coding for several hours straight. I have learned the hard way that it is best to take breaks especially when I catch myself not making any progress. I think starting earlier and sticking to a schedule would be a much better use of my time.

3 Difficulties

The biggest difficulty I had with this assignment was initially getting my head wrapped around using a bitmap, which I have never done before. As see in my work log attached in this document, my initial thoughts and assumptions about how to implement a bitmap was completely off. I spent quite a bit of time to convince myself on the use the bit masking for the purposes of this assignment.

4 Work Log

11/12/2012 1:19pm - Dissecting the problem
I started dissecting the assignment and so far I have made the following observations about the threading portion of the assignment. I will use the same method of finding prime numbers as I did in assignment one, however this time I will divide n (the total number of integers specified by the user) into threaded tasks, where each task will find a subtotal of prime numbers in the range of n/(number of threads). Second I will use an array chars to store a 32bit long int for the bitmap. I might represent the bitmap as a two dimensional array that is shared between each task however, this might cause synchronization issues as the number of threads increase. For the threaded portion of the assignment, my design will likely model a Manager/worker threaded program design where a single thread, the manager assigns work to the other threads, the workers. The manager handles all input and parcels out work to the other tasks.

After reading the posix thread tutorial posted on the class website, I think it is worth noting the performace diffrence between forking and thread creation. When compare to the cost of creating and managing a process, a thread can be created with much less operating system overhead. According to the text this is because managing threads requires fewer system resouces than managing processes. A thread duplicates only the essential resources it needs to be independently schedulable by the operating system.

11/12/2012 3:55pm - Thinking about bitmaps
After writting some stuff down in my notebook, I realized my char array size will be 1/8th the size

of n (range of integers from 0 - n). each bit in my char will be used as a bit marker. for example, lets say I want to find the total number of primes within the the range 1 - 8, n would be size 8. bits 0 , 1, 2, 4, and 6 would all be marked with one to represent prime numbers 1, 2, 3, 5, and 7. This would would give me a decimal value of 87 which I will cast as char to compactly represent my markings.

It is worth noting that a char is the smallest addressable unit I can use to represent my bitmap.

11/12/2012 5:34pm - Hello World! (threaded)
Huzzah! finished writting my first threaded program. Main creates a number of threads, specified by the user, and each thread prints their id number which was passed as an argument from main. Now I will try passing multiple arguments with a struct, and then attempt using my prime finding function from assighment 1.

11/13/2012 - 7:08am - Passing arguments to threads
I am still confused as the best way to pass arguments to my thread. The pthread tutorials posted on the class website give some good examples however I am not sure which will work best with my design. I am thinking about using a struct to pass each thread their own array of chars to manipulate, a thread id, and possibly a range offset to know what numbers each thread is responsible for checking.

11/13/2012 - 8:30am - Issues determining the total number of needed characters
I just realized that if the total number of needed characters to represent N (the number of integers to check for primes) does not divide evenly by the number of specified threads, one thread will need a larger array of chars than the rest. Of all things I don't know why this is giving me issues. I am thinking about doing a quick check to see if the "total number of needed chars" / "number of specified threads" is greater than zero, meaning the splitting of my bitmap array does not divide evenly, the thread with the most significant bits will get an extra character in it's array. Make sense? Good!

11/13/2012 - 12:30pm - Multiple arguments passed successfully
I decided to make an array of structs to pass multiple arguments to my threads. This will give me the flexibility in the future to add more data to the struct if needed without having to write a bunch of extra code. Plus I think learning how to pass multiple arguments is more challanging and a good thing to know how to do in the future. Next I will be working finding my prime numbers using the algorithm from the first assignment.

11/15/2012 - 10:02am - Algorithms
So I am just now realizing that the algorithm from assignment one is not going to work very well especially now that we are dealing with segmented version of Sieve of Eratosthenes. The algorithm in the first assignment worked just fine because we always started with 2 and moved through the array. However, now that we are dealing with segmented portions of a similar array, it is important to realize that that the starting value is offsetted. Also, not every segmented portion of the array is going to be the same size especially if the number of threads does not divided by the total number of needed characters in the array. After doing a little research online, I discovered that one possible way to deal with this is to mark all the primes up to the square root of n, where n

max size of unsigned long long. By marking all non primes up to the square root of n, I can then just mark multiples of of the known primes for each thread to find the rest. My only concern is this portion of my program will be serialized.

11/18/2012 - 1:38am - Finished threaded portion of the program
After many many many hours of blood sweat and tears, I have finally completed the threaded portion of the program. Currently the program computes all 203,280,221 primes in about 101 seconds with a thousand threads. I am almost positive my program would finish in less than 45 seconds if I could figure out how to get rid of the serialized portion of program that computes all the base prime numbers up to the square root of n. I will think on this however, If I dont start the process portion of the program, I am afraid I will not finish in time.

11/19/2012 - 8:37pm - Finished process portion of the program
Just finished the process portion of the program. it was very simple to do, I used the code provided in class to open, truncate, and map my shared memory which is the bitmap array from the threaded protion of the program. Because of my design, I was able to completly avoid using semaphores.