CS 3113 Intro to Operating Systems

Name and ID Lucas Ho 113586574

Homework #3

Instructions:
1) To complete assignment one, you need to read Chapters 1, 3 and 4 of the textbook.
2) HW must be submitted on typed pdf or word document.
You must do your work on your own.

Q1. Using Amdahl's Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores. (15 points)

a. 1 / ((1 – 0.6) + (0.6 / 2))
1 / (0.4 + 0.3)
1 / 0.7
S_a = 1.43

b. 1 / ((1 – 0.6) + (0.6 / 4))
1 / (0.4 + 0.15)
1 / 0.55
S_a = 1.82

Q2. A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between start-up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread). (20 points)

c. How many threads will you create to perform the input and output? Explain.
d. How many threads will you create for the CPU-intensive portion of the application? Explain.

c. For one-to-one models, the operations are typically sequential. It is more efficient to handle these operations with minimal parallelism so one thread needs to be created. For example, opening a file during start-up and writing to that file before terminating that program.
d. The CPU is a dual-core system with four processors so four threads will need to be

created. Four threads allow maximum utilization of the processing power where each thread can work on its CPU-bound workload. This ensures efficient use of our dual-core processor system while not overloading the system.

Q3. Consider the following code segment: (15 points)

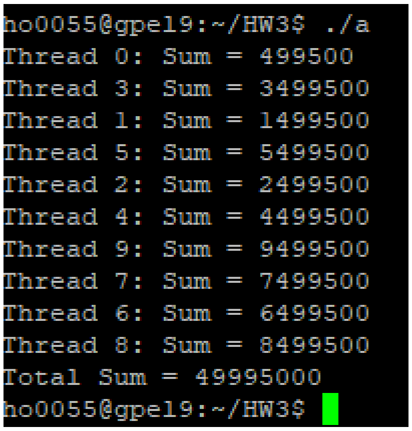
```
pid t pid;

        pid = fork();
        if (pid == 0) { /* child process */
                fork();
                thread create( . . .); /* for the purpose of this problem, you can ignore the lack
        of arguments to the function */
        }
        fork();
```

a. How many unique processes are created?
b. How many unique threads are created?

a. 4
b. 1

Q4. Pthread programming: writing a program to join on ten threads for calculating 0-9999. Each thread calculates the sum of 1000 numbers. Please attach screenshots of your execution results below. You also need to submit your code (along with a readme file) separately. (50 points)

All files (MUST INCLUDE: source codes, a readme file, and homework 3) should be zipped together.



```
ho0055@gpel9:~/HW3$ ./a
Thread 0: Sum = 499500
Thread 3: Sum = 3499500
Thread 1: Sum = 1499500
Thread 5: Sum = 5499500
Thread 2: Sum = 2499500
Thread 4: Sum = 4499500
Thread 9: Sum = 9499500
Thread 7: Sum = 7499500
Thread 6: Sum = 6499500
Thread 8: Sum = 8499500
Total Sum = 49995000
ho0055@gpel9:~/HW3$
```