CS 3113 Intro to Operating Systems

Name and ID ____Kyumin Lee 113591341____

Homework #3

Due date: 10/29/2023 at 11:59 pm

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.
<u>You must do your work on your own.</u>

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)

Amdahl's Law says, Speedup <= 1/(S+(1-S)/N). Where S is the fraction of serial tasks and N is the number of processors. 60% parallel component means S = 1 - 0.6 = 0.4

(a) N = 2, speedup <= 1/(0.4 + (1- 0.4)/2 = 1/(0.4 + 0.6/2) = 1/(0.4 + 0.3) = 1/0.7 = 1.42857

(b) N = 4, speedup <= 1/(0.4 + (1 - 0.4)/4 = 1/(0.4 + 0.6/4) = 1/(0.4 + 0.15) = 1/0.55 = 1.81818

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)

(a) How many threads will you create to perform the input and output? Explain.

One for each(2 total). Because we are only reading and writing a single file. Parallelizing it could cause issues. Also, since all input is performed at the start and all output is performed at the end of the program, They won't overlap with each other or with the CPU-intensive part of the application.

(b) How many threads will you create for the CPU-intensive portion of the application? Explain.

4 threads. Because our system has two dual-core processors, which means 4 cores in total. To fully utilize the hardware I'll use all 4 cores to achieve the best performance.

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?

6 in total

```
(pid = x) — (pid = 0) — ()
    I            I
   ()           () — ()
```

(b) How many unique threads are created?

8 in total

Each 6 processes will have its initial 6 threads. But the child that has pid of 0 gets forked in the if block and so two processes run 'thread create(...);'.
So in total there will be 8 threads.

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)

```
lee0436@gpel8:~/homework3$ pwd
/home/lee0436/homework3
lee0436@gpel8:~/homework3$ ls
homework3  homework3.c
lee0436@gpel8:~/homework3$ gcc homework3.c -o homework3 -lpthread
lee0436@gpel8:~/homework3$ ./homework3
Total Sum: 49995000
lee0436@gpel8:~/homework3$
```

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