CS 3113 Intro to Operating Systems

Name and ID ___Thomas Duffy__113603769

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)

Amdahl's law S=(1/((1-P)+P/N)

A) P = 0.60, N =2, S=1/((1-.60)+0.60/2), 1/0.70, S = 1/0.70 or 1.43

B) P = 0.60, N = 4, S = 1/((1-0.60)+0.60/4), 1/0.55, S = 1/0.55 or 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)

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

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

A) I will use 1 thread to perform the input and output because there is one input during the program startup and one output that occurs right before the program terminates, These tasks are I/O bound and are not CPU intensive, and also happen at different times so they do not need to run concurrently or throughout the programs entire execution Creating multiple threads shouldn't improve the performance and will only add more complexity.

B) I will create 4 threads to improve the performance of the CPU-intensive portion by utilizing all 4 of the CPU cores efficiently. Each thread can run on a different core, this will use parallelism, and any given core will not be left with all or none of the load.

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?
The first fork creates a child of process pid t, the second fork executes in the child process and creates another child in the child process there are now 3 processes, and the third fork executes both the parent and the child from the first fork as well as the new child created inside the if condition. **There are now 6 unique processes.**

b. How many unique threads are created? 1 unique thread is created
inside the if statement the child of the parent creates another child, only the first child creates a thread because the second child is created inside the if condition and doesn't reach the create thread() function, so there is only **1 unique thread that is created.**

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

```
cs027@cs027:~/homework3$ gcc -o homework3 homework3.c
cs027@cs027:~/homework3$ ./homework3
Total sum of numbers from 0 to 9999: 49995000
```