

CS 3113 Intro to Operating Systems  
Name and ID - Roberto Cervantes, 113603977  
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 a 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)

To calculate the speedup gain we can use Amdahl's law  $S = \frac{1}{(1-P) + \frac{P}{N}}$

S = Speedup

P = Proportion of the program that can be parallelized

N = Number of processors (or cores)

Using the numbers given for 2 and four cores we can solve for the speedup gain

- a. 1.43
- b. 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.

1 Thread for both input and output.

The input and output (I/O) in this application are simple tasks: a single file is opened at the start, and another file is written to at the end.

These I/O tasks are not CPU-bound, and the system performs the input at the beginning and the output at the end. Since these tasks are isolated and occur only once (and not concurrently with the CPU-intensive work), you only need one thread for input and one thread for output to manage them effectively.

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

4 Threads (one for each core, maximizing CPU utilization).

The system has 4 available cores (two dual-core processors). Since the application is entirely CPU-bound after the input phase and before the output phase, you want to maximize the usage of all available cores to parallelize the work.

In the one-to-one threading model, each thread maps directly to a kernel thread, which allows you to fully utilize each processor core. Therefore, to make efficient use of all the available cores, you should create one thread per core, i.e., 4 threads.

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();
```

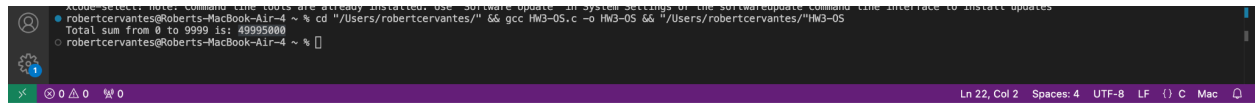
a. How many unique processes are created?

6. Each fork() call creates a new process. Since there are three fork() calls (one in the parent, one in the child, and one in all processes), a total of 6 processes are created: the original process and 5 new ones

b. How many unique threads are created?

1. A thread is only created in the child process (P2) within the conditional block, so there is only 1 unique thread created in total. No other processes create additional 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) All files (MUST INCLUDE: source codes, a readme file, and homework 3) should be zipped Together.



```
robertcervantes@Roberts-MacBook-Air-4 ~ % cd "/Users/robertcervantes/" && gcc HW3-05.c -o HW3-05 && "/Users/robertcervantes"/HW3-05
Total sum from 0 to 9999 is: 49995000
robertcervantes@Roberts-MacBook-Air-4 ~ %
```

The screenshot shows a macOS terminal window with a dark background. The prompt is 'robertcervantes@Roberts-MacBook-Air-4 ~'. The user enters 'cd "/Users/robertcervantes/" && gcc HW3-05.c -o HW3-05 && "/Users/robertcervantes"/HW3-05'. The output is 'Total sum from 0 to 9999 is: 49995000'. The terminal status bar at the bottom shows 'Ln 22, Col 2', 'Spaces: 4', 'UTF-8', 'LF', 'C', 'Mac', and a bell icon.