Single- Versus Multi-threading

Shaun Hoadley

CPT304: Operating Systems Theory and Design

Joshua Reichard

June 14, 2020

According to Silberschatz, Galvin, and Gagne, the traditional, or heavyweight, process has a single thread and may only do one thing at a time; whereas, a process with multiple threads can perform more than one task at a time (2014, 4.1).

**Part One**

**Explain why a multi-threaded or single-threaded solution is more effective for each. Support your statements with evidence from your sources.**

* **A printer that is performing the job of printing a set of documents**

A Printer that is printing a set of documents would be more effective with a single-threaded solution. A printer can only print one page at a time and is going to print the pages in the order they receive them.

* **An application that allows the graphical user interface (GUI) to remain responsive while processing a large number of files**

A multi-threaded solution would be the most effective with an application which allows the graphical user interface to remain responsive while processing a large number of files. One thread could be working on maintaining GUI responsiveness while the remaining threads process the other files.

* **A Python webserver that listens for requests, reads them, and pushes the data in the database**

A webserver would definitely benefit from a multi-threaded solution. After receiving a request, the process could be passed on to a separate thread to process the request, freeing up the first thread to listen for the next request.

* **A *shell* program that closely monitors its own working space, such as open files, environment variables, and current working directory**

Having a thread to handle each process would allow a *shell* program to monitor each part without dividing its resources. The use of a multi-thread solution would be effective.

* **A program that calculates a large number of payments where each calculation is independent of other payment calculations**

A multi-thread solution in this type of program would be effective because it could perform its calculations for multiple accounts simultaineously; whereas, if it were a single-thread, it would have to perform each calculation consecutively rather than concurrently, taking more time.

**Part Two:  
Use the following guidelines to create three additional programming scenarios similar to the ones given in Part One.**

* **Write Scenario One such that the most effective solution is a single-threaded process.**

An example of a scenario in which a single-threaded process would be the most effective solution is a microwave oven. As the microwave can only perform one function at a time, having multiple threads would not do any good. The microwave will not come on until you set the timer, the timer cannot start counting down before it has a starting point to count from, and it should not shut off until the time is up unless otherwise interrupted.

* **Write Scenario Two such that the most effective solution is a multi-threaded process.**

For scenario two, a multi-threaded process would be the most effective, and likely not possible with a single-thread process, in a virtual reality (VR) headset. A VR rig has to be able to process multiple sensors simultaniuosly to determine position, direction, and velocity, while synchronizing these inputs to the two miniature high definition screens positioned in front of your eyes.

* **Write Scenario Three such that the most effective solution is either a single-threaded or multi-threaded process.**

For scenario three, some time in the past, I had created a very simple crane game that was powered by an Arduino Uno maker board. It is an excellent example, in my opinion, of an application which is well suited for single-threaded processes. By pressing a button, the program initiated the game. First, the LCD display lights up with a three second countdown. When the counter reaches zero it resets to a 60-second timer and two more buttons control movement either from left to right or from front to rear, but not in reverse. When the counter reaches zero again, the crane bucket lowered, closed, re-raised, and moved back to its initial position and drops anything it may have picked up. As everything is done one at a time and nothing else can be done before the previous completes, the program which controlled the crane would not have benefitted from multiple threads.

**References**

Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). [Operating system concepts essentials](https://ashford.instructure.com/courses/66667/modules/items/3374055)(2nd ed.). Retrieved from https://redshelf.com/