Trevor Lund
ComSci 352
2-12-13

HW 4

1. Circumstances where a multithreaded program performs better than a single-threaded one when it's only on a single-processor system would be in the instance that the program would have a high risk of page faults or other operations that would cause the processor to pause and wait for other system events. Even with a single processor, a multithreaded program would be able to continue doing events even when the parent thread has to pause and wait. An example of this would be a program that allocates a large amount of memory and thus would have a lot of page faults when trying to access that memory. Multiple threads would allow the program to continue executing as expected even when one of the threads gets a page fault.
2. In the program, 3 threads will be created. The first one is created as a result of the main method, and the next two are created as part of the if statement, since the variable i is shared between threads and is decremented each time a new thread is created.
3. The outputs of the program are "var = 1\nvar = 5\nThread 2: var=3\nThread 2: var=15"

   As the program executes, it first hits fork(), where the parent process changes the variable and continues on, continuing execution before the child process, thus printing out "var = 1\nvar = 5\n"

   Next, the parent process creates a thread with a copy of its memory. However, it doesn't wait for the thread to complete, so it continues on to make a second thread, printing out "Thread 2: var=3 \n". Meanwhile, the child process is making a thread that executes do_1, but again it doesn't wait and goes on to create a thread for do_2, printing "Thread 2: var=15 \n"