Matt Kennedy
William Shreeve

CIS 452 - 10

Lab 4 - Threads

1. In the original, unmodified program, the initial thread executes and exits before the spawned thread that runs the do_greeting function terminates. Because this is the initial thread, and the second thread is spawned off this, the program will simply exit and not wait for the program to print before exiting.
2. When the program is run, the output appears to be that one thread is run before the other. In the program we ran, we saw that “World\n” was printed 10 times, then “Hello “ was printed 10 times. This indicates that one thread happened before the other, almost in an atomic fashion.
3. After a one second sleep is inserted, the thread appear to be running simultaneously. The first line is just “World”, then most of the output is “Hello world”, which a few lines not being conformant, such as “Hello Hello World”.
4. We believe Linux uses a 1-1 mapping. This is due to multiple print statements, which are system calls. This would require multiple kernel threads. This also allows for more concurrency on the system because in many-to-one, a blocking thread could block for multiple other threads.
5. In this program, there are 3 threads running; the initial thread, and the two spawned threads. Thread 1 is passed ‘a’, and thread 2 is passed ‘b’. Each thread has accessed to the global int “shared_memory”. The output for our example is as follows:

   ```
   Parent sees 5
   Child receiving b initially sees 5
   Child receiving a initially sees 5
   Child receiving b now sees 7
   Child receiving a now sees 8
   Parent sees 8
   ```

   What is happening is that all 3 threads print before reaching the increment statement in the given functions. Threads 1 and 2 reach the increment first, then thread 2 prints. Immediately following this, the parent reaches the increment, then thread 1 prints, followed by the initial thread.

6. From the source code for pthread_create.c:

   ```
   /* Store the address of the start routine and the parameter.  Since we do not start the
   function directly the stillborn thread will get the information from its thread descriptor.  */
   pd->start_routine = start_routine;
   pd->arg = arg;
   pd->c11 = c11;
   ```

   The new thread is created in the pthread struct, then the attributes are stored in the struct and are retrieved from the thread descriptor later.