Collin Bauer
CSCI 340 – Operating Systems

1. pmap utility

a) Both bash instances have the exact same libraries in them. They have different virtual addresses, but their offset values and permissions are the same

i. `libnss_files-2.27.solibnsl-2.27.so`

ii. `libnss_nis-2.27.so`

iii. `libnss_compat-2.27.so`

The above three are all parts of the Name Service Switch library, which deals with database configuration and name resolution. It largely handles Linux users, passwords and permissions.
- `files` contain the bulk of the library, which allows support for much of the database info found in `/etc/`.
- `nis` is a plugin for Network Information Service, which is a program for distributing system config data between computers on a network.
- `compat` allows a “compatibility” mode enables extra permissions for additional users and netgroups to access system files.

iv. `libc-2.27.so`

This is the standard C library, or at least, one version of it. Modern Linux distros specifically use `glibc`, or the GNU C library, version 2.0+. This contains some of the most common functions used by almost all C programs.

v. `libdl-2.27.so`

This library provides an implementation of dynamic loading and/or linking (found conflicting reports on this) when attempting to open files. `libc` makes a reference to this whenever a C program needs it included, but it seems to not be inside `libc` for compatibility purposes (again, conflicting reports).

vi. `libtinfo.so.5.9`

I couldn’t find a clear answer to what this library is,but it seems to be another submodule of `libc`, but I found references to something called ncurses as well.

b) Both shells have exactly three instances of Bash running, with the same permissions and memory space allocated,but different addresses. I'm guessing the different addresses implies that they are different process.

2. Threads

a) A thread is almost identical in function to a process, except two threads of one process share the same address space, thus resources, while two processes cannot do this.

b) Many processes require tedious computation that may be sped up by splitting the load into different threads, allowing different processor cores to split up the work.

c) Slack uses lots of threads!

   i. Proccess pid: 13708

   ii. Thread pids: {13708 13713 13716 13718 13720 13722 13724 13726 13728 13732 13737 13739 13769 13815 13711 13715 13717 13719 13721 13723 13725 13727 13729 13736 13738 13752 13791 23798 24118}

3. Memory and Concurrency

Quick note: My Linux machine is actually the *host* computer, so it has access to a whole...two cores. *(It's a really cheap laptop.)*

a) My loop had to run for around 1,000,000 times before the values started mismatching, though on some runs it still matches.

b) The more times a critical section of code runs, the more likely two threads will collide in those sections and cause redundant or conflicting executions. This means that the odds of them colliding with just a few executions is really, really low (but possible).

c) Having more cores it means threads can run uninterrupted more often, so both will be loaded into memory at the same time more often. This leads to faster execution overall, but also more collisions during critical sections.