EECS 678
Stephen Longofono
Lab 9, Procfs

1. The lack of change in a single process or thread is not sufficient to determine a deadlock situation. This is because the thread at hand could be waiting on another thread to finish with a resource, or sleeping while waiting for some sort of IO. The only way to be sure that a deadlock has been reached is if all active threads fail to make progress, that is, they all see no increase in user time and no increase in system time.

2. In order to have a safe time of observation, you would need to know the longest amount of time a thread might be stalled or otherwise blocked under normal operation. For example, if your threads blocked on I/O or some long, asynchronous event, you might mistakenly classify normal operation as deadlock. To address this, the time interval to check for deadlock should be just longer than the worst case interval of the longest task run among your system threads.

3. As the time increases, it appears that deadlock is less likely to occur. On such a small scale, the threads behave in a nearly deterministic manner; the time it will take them to initiate and do their work is fairly regular and predictable, and for varying values of ACTIVE_DURATION, we could assess whether the threads “sync up” in such a way that deadlock will occur.

It may be the case that as we increase ACTIVE_DURATION further, we will see a similar phenomenon and the threads will “sync up” again. In this case, the initiation time is small relative to the running time, so the offset is small. However, with larger duration times, the percentage of variation is also larger, so there may be new deadlock situations when one of the last threads overlaps the first of the next cycle.