Program 3 Analysis

Task 1

For task1, we created several applications that we also applied to later code. We ended up with some interesting and unexpected results. First of all, we created task1.c, which calculates an expensive function that takes several seconds to run, and meanwhile it looks at the memory usage by reading from meminfo. After a while we decided to create a separate piece of code, freemem, which just reads the available memory. We can then run a process in one window and use another to run freemem. With the original task1.c, it usually allocated somewhere in the 150-200 range. Interestingly, memory usage went down when running ping commands, and it happened consistently enough that it didn’t seem to be coincidence. It was tested with a number of bash commands, all of which used more memory.

Task 2

Like task 1, task 2 had strange and unexpected results. The first forked process sometimes resulted in a change in memory usage, but the 2 following forks didn’t seem to change anything, and they both returned 0 for their individual process usage, and the free and available memory counts remained unchanged. Sometimes there was no reported change in memory, so it seems that a number of our test processes, aside from memhog had relatively low effects on memory. This was true for both identical forked processes and execv forked processes. However, we created a program called memhog, that forks 15 children and does seem to make a large dent on memory usage. This was especially true when we ran multiple copies of it. In task2.c we tested identical children, and in task2b we tested children using execv and there was no noticeable difference.

Task 3

Task 4

Overall Analysis