CS310 Lab 4

October 4, 2023

1. Write a `Hello World` C program to observe the behavior of the virtual address space associated with the program.

   (a) Check out the `/proc/<pid>/maps` file. Understand the various segments of the virtual memory described in this file. Identify the various properties of each segment and how it relates to the program. The page `https://www.baeldung.com/linux/proc-id-maps` may help.

2. Modify the program to allocate blocks of memory dynamically, using repeated calls to `malloc` in a `for` loop.

   (a) Note how the address of the heap changes after the dynamic allocation. You may have to request a sufficiently large block in each loop iteration to ensure that this happens. *How is the size of the block relevant in this context?*

   (b) You can also observe the memory used by the process using the `ps` command with the right options. In particular see the `SZ` and `RSS` fields.

3. Have a look at the system calls `brk` and `sbrk`. These can be used to change the current limit of the heap, called the process' *program break.* Modify the program above to undo the memory allocation (using the library function `free` to free each allocated block in an iterative manner.) Print the value of the program break in each case.

4. Write a C program that has a main program which starts a thread `t1`. Print for both the main thread and the thread `t1`, the following: (a) the pid, and (b) the thread id. There are some issues to handle here, such as:

   (a) What if the main thread finishes before `t1` starts?

   (b) What if `t1` starts running before main returns from `pthread_create`?

   See the man pages of `pthread_create` and `pthread_self`.

5. Write a C program that creates a thread. The thread creates a structure containing two integer fields (with the values 100 and 150) and returns it to the main program. Use `pthread_join` for this.

   (a) Note what happens when the item to be returned is declared local to the thread.

   (b) How will you fix the issue?