* 1. To run a program, the OS copies the data into the physical memory, sets aside an execution stack, to hold the local variables. It also holds aside the heap (memory region). The mode switches from kernel to user-mode, because we want the program to have restricted access, so it does not hurt the rest of the system. Its also nessesery so if the program has any malware or malfunction, it wont do any damage to the rest of the machine.
     1. Stores the process ID: pid\_t
     2. Accumulated virtual memory usage: acct\_vm\_mem1
     3. Besides pid and virt there are f.ex. NI, PR and RES.

|  |  |  |
| --- | --- | --- |
| **0xfffffff** | **Stack** | **Holds the procedure local variables** |
|  | **Heap** | **Holds dynamic allocated objects** |
|  | **Data segment** | **Contains initilized static variables which have pre-defined value and can be modified.** |
| **0x0** | **Text segment** | **Contains executable instructions and is fixed size and read-only.** |

* 1. Global – declared outside of all functions

Static – Static can be used outside and inside of functions, and will respectively be a global or local variable. They are only initialized only once at the time of declaration only.

Local – declared inside of a function, and it will have scope within a function and it will retain until the life of the program.

* 1. Var 1 = data segment - static variable

Var 2 = stack variable – local variable

Var 3 = heap variable – global variable

* 1. Text: 1838   
     Size: 616  
     Bss: 8
  2. Start address: 0x00000000000005f0
  3. 0x00000000000005f0 <\_start>

Its an entry point to the program, there it first sets up some stuff, then creates the argument array argv, counts how many arguments there are, calls main and then after main in returned, it calles exit.

* 1. The addresses change each time, because this will increase security vs viruses and attacks. Because aslr randomization is on, there it will change each time.
  2. The max stack size is 8192 Kb – 8Mb.
  3. It has created an infinite loop because we retrieve the method func() inside of the method func(). Therefor it will run until crash (segmentation fault (want to access non-existing memory)).
  4. Nr: 523565.

The address difference between each loop is -20. Therefor we know that an address use 2 byte. However we are printing out 2 lines per loop. So 523565/2\*40byte = 10.5 Mb. Since this is largen then the mac stack size found by ulimit, it will crash and give us and segment fault.owever

* 1. Since we know each function call uses 20 addresses, and there are 2 bytes per address. We use 40 bytes per recursive function call.