Isaac Hendrickson and James Ahrens

COMP 340 Project 3

May 8th, 2018

**Task 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | RSS (KiB) | VSZ (KiB) | % MEM |
| Main, no allocation | 652 | 10220 | 0.0 |
| First child, no allocation | 128 | 10220 | 0.0 |
| Second child, HUGE malloc() loop | 2,029,420 | 698,859,460 | 52.2 |
| Main after second child | 712 | 10232 | 0.0 |
| Third child, no allocation | 92 | 10232 | 0.0 |

**Task 2**

3 forks:

|  |  |  |
| --- | --- | --- |
| Process | RSS (KiB) | VSZ (KiB) |
| Main before children | 636 | 6424 |
| Main after 1 child fork | 732 | 6432 |
| Main after 2 child forks | 736 | 6436 |
| Main after 3 child forks | 740 | 6440 |
| First child | 288 | 6432 |
| Second child | 292 | 6436 |
| Third child | 296 | 6440 |

Execv with big\_function

|  |  |  |  |
| --- | --- | --- | --- |
| Process | RSS (KiB) | VSZ (KiB) | % MEM |
| Main at start | 644 | 6428 | 0.0 |
| Child at start | 288 | 6432 | 0.0 |
| Main after child calls execv | 736 | 6432 | 0.0 |
| Child after execv | 200720 | 97804436 | 5.1 |

Execv with empty\_function

|  |  |  |  |
| --- | --- | --- | --- |
| Process | RSS (KiB) | VSZ (KiB) | % MEM |
| Main at start | 644 | 6428 | 0.0 |
| Child at start | 288 | 6432 | 0.0 |
| Main after child calls execv | 736 | 6432 | 0.0 |
| Child after execv | 344 | 4164 | 0.0 |

**Analysis**

Task 1: New processes are given about 640 KiB on their start and that does not vary based on things currently running or the type of process. If the process needs more space it allocates it when the code calls it to, so the memory allocation of a process will change over time, the OS will allocate and deallocate space as it runs.

Task 2: Child processes are given about 288 KiB, for our function each subsequent child process was given what the previous child was given plus another 4 KiB, this may be a coincidence though. Initially, Linux allocates the normal 288 KiB to a child process that loads another process but will allocate that child more space when he calls execv and needs the extra space. In our example with an empty function the child was given an extra 56 KiB, however when the child ran an execv on a function that needed a large amount of space he was given a massive 200720 KiB.

From this assignment I learned that even small functions can take quite a bit of space. I also learned that an execv call will not completely destroy the process that called it but rather replace it.