COMP 340: Operating Systems

Spring 2018

Project 3: Memory Manager

Date Released: 04/25/2018

In this project, you will characterize how Linux allocates memory and create tools to react to current memory allocation. Your main source of information about memory allocations will be one of the following: either read /proc/meminfo or run free or vmstat.

Write code and run experiments to complete each of the following tasks. Each piece of code you write must include a header comment (name, course #, date, description, how to invoke compiler) and liberal comments in the code. In the code, I expect you to define all variables with comments and explain each control structure with comments.

When you run an experiment (defined by the questions associated with a task), be sure your document clearly identifies the task, what you are trying to accomplish, the inputs of the test, the data collected and an analysis of the results. Always describe your procedure for testing, such as:

1. read /proc/meminfo and record the available memory at the start
2. start processes A and B running in the background
3. read /proc/meminfo and record current available memory
4. etc.

Run each of these tasks/experiments in your Linux VM. If something does not compile or work correctly, clearly identify that in your submission.

Linux administrators (and users who like to control their environment) constantly create and use tools to monitor and control the system. This assignment is a small step in that direction.

### Task 1 - Characterize Memory Allocation of Individual Processes

Run a variety of programs, existing functions and some you create, and record how much memory is allocated to each. Set some processes running in the background and see if that influences how much memory is allocated to a specified process when it starts in the foreground. Be sure that your test set has processes of varying sizes.

How much memory does Linux allocate to a new process? Does that vary based on the size of the process or what else is running in the system? Does the memory allocation of a process change over time?

The code you write for this task can be fairly mindless: it just needs to exhibit different size processes. However, you do need to provide comments in all code and submit all code, even processes that just take up space.

### Task 2 - Characterize Memory Allocation of Forked Processes

Write a program that forks at least 3 children. See how much memory is allocated when each child process starts. Write a separate program where a child process uses execv() to load another process and measure the memory allocation after the execv() call.

How much memory does Linux allocate to an identical child? How much memory does Linux allocate to a child that loads another process?

### Task 3 - Signal Memory Allocation that Exceed Thresholds

Write a program that runs in the background and monitors allocated memory. You should check memory every time a new process starts or set a timer and check on defined intervals. This program must write a message to the screen when total memory allocation exceeds 75% of available memory. Your code should send one message every time system memory allocation increases above the 85% barrier - i.e. if system allocation goes back below 85%, then the code should signal if it goes back above.

The 85% value should be a globally defined variable so it is easy to modify for testing purposes.

Based on knowledge from Tasks 1 and 2, start your monitor process running and then start other processes to approach and then exceed 85% memory allocation. Verify your Task 3 program works correctly.

### Task 4 - Keep Memory Allocation Below Thresholds

Write a program to detect when current system memory allocation exceeds 85% and start halting processes until it goes back below 85%. Make sure you only halt user processes; do not halt any system processes. You should check memory every time a new process starts or set a timer and check on defined intervals.

The 85% value should be a globally defined variable so it is easy to modify for testing purposes.

Based on knowledge from Tasks 1, 2 and 3, start your monitor process running and then start other processes to approach and then exceed 85% memory allocation. Verify your Task 4 program brings memory allocation back down below 85%.

### What to Turn In:

* turn in all source code (tasks 2,3 & 4)
* turn in all experiment documentation/analysis in a Word document
* turn in an overall analysis and summary about what you learned from this assignment in a Word document
* *(these can all be the same Word document - just be clear to label the individual parts)*

### Grading

The code and analysis for each task are worth 20 points each.

The overall comments across all your source code are worth 15 points (5 points deducted per task for insufficient comments).

The overall analysis and summary of what you learned (across all tasks) is worth the last 5 points.

All code must compile and work correctly to receive maximum points. In Tasks 1 and 2, the analysis of your experiments must be complete and show insight to receive maximum points. In Tasks 3 and 4, you must document thorough testing to receive maximum points. The overall analysis and summary of what you learned should be a paragraph or two.

Any error/incompleteness in the analysis deducts 5 points.

**Project Policies:**

* Assignments must be submitted electronically via my.gcc. Be sure to upload your files correctly the first time.
* This project is a group project. Every student needs to work as a part of a 2-people team. Students are expected to declare keep their project 2 teams. If you wish to change your team, contact Dr. Al Moakar and she will assign you a team.
* 20% of the grade will be weighed with the peer evaluation. Students are expected to turn in the peer evaluation form posted on mygcc.