CS 481 Lab 2

Due by 11:59pm on Monday, 23 April, as an e-mail to the instructor (jedcrandall@gmail.com). Please send only PDF files and C code files. It's unlikely that there will be any extensions, since the TA needs time to grade your submissions.

I strongly recommend submitting early versions of your C code so that TA or I can test them for you and let you know how you're doing along the way. If you wait until the deadline and submit something that doesn't even work on most of the machines, that will have a huge negative impact on your grade.

100 points (out of 100 total for Lab 2), the writeup is 50 points and the C code is 50 points

The purpose of Lab 2 is to understand virtual memory and virtual memory swapping.

You may work in groups of up to three. You can work alone or in a group of two, but the grading standard will be based on a group of three.

Refer to the syllabus about cheating and collaboration. If a single sentence of your writeup is copied from another source without proper attribution the entire group will receive a 0 on the writeup. You may refer to the C code of other groups for high-level concepts, but each group is expected to write their own C code for lab 2. If a significant part of the code is copied you will receive a 0 on the C code part of the grade. Feel free to use any C code you find on Google as a guide. For example, if there's C code online for forking a child with a pipe or using signals between the parent and child feel free to integrate their example code into your code. This will not be considered plagiarism. For this assignment, I'm more worried about plagiarism from other groups than plagiarism from Google since Google can only give you pieces of what you need to do.

If you do not want me to share your lab 2 writeup with others (such as showing it to the class as a good example or giving it to other students in the future who are curious about Linux), please indicate this clearly at the top of your writeup.

Part 1: C code (50 points)

You are to write C code that answers the following questions (by printing the questions followed by the answer for each to stdout):

1. How much main memory (RAM) does the system have?'

2. How much swap space does it have?
3. Is it using multi-level page tables or an inverted page table?
4. How many entries are there in the last level of page table?
5. What is the page size?

Note that you'll need to answer these questions in this order when printed to stdout, but that doesn't necessarily mean you should calculate them in this order. Also, don't assume that all machines have a power of 2 amount of RAM or have a large amount of RAM, the machines your code will be running on are from right around the turn of the millennium.

You are restricted to using only the following library functions:

- You may use stdio.h only for printing to stdout, you may not use any named file with stdio.h
- You may use any math functions from stdlib.h (e.g., atoi), as well as calloc, malloc, free, and realloc
- You may use sleep() and usleep(), fork(), and anything related to pipes or PIDs in unistd.h
- You may use anything in sys/time.h, but keep in mind that about half the machines have a timer resolution of only 1/100th of a second.

If you feel like I've missed a critical function or something that would be useful, let me know and maybe we can add it. Your code should be a single C file that compiles with gcc without any special options, libraries, or Makefiles, though on OpenBSD we will include “-lrt” so you can use sys/time.h. You'll need to account for big vs. little endian and 32- vs. 64-bit in your code in a portable way. We'll compile and run your code on five machines, where each of the answers your code prints out will be worth 2 points. If your code crashes, we'll only grade the parts printed up to the point where it crashed on that machine.

NO TRICKS. You should be using your knowledge of virtual memory management to answer the questions reliably. If your code is trying to use tricks (such as dereferencing a pointer into a text section and inferring from the machine code what machine you're on, then just getting the answers specific to that machine off Google) then we won't grade that submission. Your answers to all five questions should be based on either return values from memory allocation attempts or timing measurements.

Also, all three group members should be involved in the C code, with the work divided as evenly as possible.

Part 2: Writeup (50 points, 20 to the whole group and 30 individually)

In addition to your C code, also submit a PDF with a short writeup. The writeup should contain:

1. An overview paragraph.
2. One short paragraph for each question describing your strategy for getting the code to answer that question reliably.
3. A paragraph explaining how you ensured that the work was divided evenly among the three group members for the C code. (If there's a large discrepancy in C coding skill levels in your group, you should describe what you did in terms of mentorship to get all group members up-to-speed and able to contribute).
4. A paragraph from each group member explaining how they did some research that supports life-long learning.

The last part will be 30 points to individual group members, while the rest of the writeup as a whole will be 20 points for the whole group.

Life-long learning: you should use something other than the textbooks or the Internet to do some research that will help you with this lab, and then describe this in a paragraph. This is a personal thing that will be graded for each individual group member, you each need to do this separately. You should find some information that helps you with the lab that is not easily available online and is not in the textbook. Then you should write a paragraph describing how the information helped you and how you found it. For example, you might find something in the OpenBSD source code, or go to the library and check out a book about UltraSparc memory management hardware. You should also explain how you confirmed that this information wasn't easily available online or in the textbooks (easiest way to confirm this is to Google it, and presumably you've read the textbooks so you know what's in there).

If you read the OpenBSD or Linux source code to understand something, then as long as what you understood from it is not readily available in written English somewhere then that counts as not being online or in the textbooks, even though the source code for both OpenBSD and Linux are online.

The last thing you want to do is follow your natural instincts to use Google to find something. The whole point of this last part is to assess whether you can do some basic research for information beyond Googling it or finding it in the textbook. Your best bets are probably to either read the OpenBSD source code or go to the library.

Make sure the paragraphs from each group member about life-long learning have their name attached to them somehow.