EECE 315
Project/Assignment #5

This is a **group assignment**. All group members must equally contribute, cooperate and coordinate.

Deliverable: Submit one zip file that includes all your files. Please use folders to organize your project files. Please find the guidelines for the submission and on the project report on the next page.

Any one of your group members may submit for the group (i.e. all group members will see the same submission dropbox), please coordinate.

Github: As before, every group member must use github to collaborate. You are to record regular snapshots of your progress to github. Create a folder PA5 and two subfolders for each project; save your files in the corresponding directory.

Due date: Friday March 20, 2015 by 3:00 PM

==========

Submission: Submit one .zip file that includes two .pdf files and two directories.

- The first .pdf file is your project report for Project 1, and the second .pdf is your project report for Project 2 (see below. Include one contribution page (as described below) for each report and clearly identify the contribution of each member for each project and the overall contribution percentage.
- The two directories include the c file(s) and the related files for the projects.

- Please only use the formats specified. That is .zip for the main file (should easily be opened in Ubuntu or Windows), .pdf (or html, to be portable, there are many free pdf creator software), and the .c file formats. Please name the zip file `GgroupNum.zip` where `groupNum` is *your group number*.
- All .c code files should include sufficient comments, and the projects should compile and work correctly according to the specification.
- Some marks will be dedicated to following the guidelines.

Project report: Each project is explained in a formal report that has the following sections:

Front page - names and student numbers of all members in your group, title, date sunmitted, ...

Contribution page - Clearly state the contribution of each group member for each project.

This info page should be in percentage, as well as in words specifying which tasks, design, coding, debugging, etc each group member contributed to.

Introduction - A brief description of the introductory and required information related to the project, such as the problem statement, high-level flowchart and the project benefits.

Solution - A complete description of your approach to solving the problem, such as the implementation, techniques, and explanation of your code (the code itself is submitted by the .c files; see submission guidelines).

Debugging - An outline of any problems you ran into, any test sets used to check the correctness of the code, ...

The report should include relevant figures, comments, test cases, conclusion, and any appendixes needed. Do not include the complete code with the report, though including small code segments that are necessary for your explanations is fine.

The length of the report is up to you. It should be just long enough to sufficiently explain the project as outlined above. Use an 11pt font size (Times New Roman). It should not be longer than 4 pages (excluding the front page, contribution page and any possible appendixes).

You may be invited later on to defend/present your project or your approach.

==========

1) Project1: VMem Manager

- The description of the project is on page 458 to page 461 of the textbook.
- As usual, use Ubuntu and include a make file. The make file should include all dependencies that build your program. See the guideline.
- Any accompanying file needed for this project (such as addresses.txt) is posted on Connect.

2) Project2: Shared Memory

Shared memory refers to a common block of memory that is mapped into the address spaces of two or more processes. You will be using the POSIX library to implement shared memory.

This exercise should be solved in Ubuntu and please submit a makefile as usual. It is from the G. Nutt's book.

You may use either of the following set of functions:

1) shmget, shmat, and shmdt, shmctl,… (more Linux specific shared memory API functions, see slide 3-36)
2) or shm_open, mmap, shm_unlink, … (see page 132 and 133 for examples)

Your program originally acts as the parent process and creates a shared memory block and then creates the required child processes.

The program is to solve linear $n \times n$ systems of equations of the form $Ax=b$ using Successive Over-relaxation (SOR). Note that A is the $n \times n$ coefficient matrix, b is a vector and x is the vector of unknown variables.

SOR is an iterative technique and finds new value of x based on the previous values until the equation is numerically solved. See http://en.wikipedia.org/wiki/Successive_over-relaxation for the algorithm. You are allowed to use any correct implementation of SOR including the one provided in the above Wikipedia page. Make sure to explain the algorithm you use in the report.

Implement SOR on an n-process system by having n different processes (all running the same program) where the ith process computes the x_i (ith variable in vector x).

You may include A and b as arrays insides your program but provided enough test cases. The output can simply be printed out on *stdout*. n must be a symbolic constant that you can set to a value between 3 and 6.