CECS 326: Project 4 (team of 2 students)

Objectives: Interprocess communication with semaphores and shared memory.

Create 4 groups/segments of shared memory. Each "group" holds three 512-byte "chunks". The 3 chunks in the first group will be initialized with random lower case letters. Everything else will be initialized with random upper case LETTERs.

Create 5 processes. To slow down processing a bit: Each process generates a 32-bit random integer (speed_check). When its speed_check < 5000, a process [generates 4 additional random values to] randomly <u>selects 2 groups</u> and randomly <u>chooses one chunk from each</u> of these 2 groups to operate on. The process swaps the contents of these 2 chunks atomically (while a chunk is being swapped, no other process can do its swap on the same chunk). The process repeatedly swaps 2 random chunks whenever its speed_check < 5000, until it had completed its required number of operations (inputed by the user). The last process to terminate removes all resources.

Be carefully with deadlocking: A situation where all 5 processes block indefinitely because they have to wait for each other to finish swapping their chunks. It needs to be handled explicitly. Some additional theories on deadlock (though not really necessary) will be covered in lecture for CH7.

You have permission to use SEMAPHORE class for this assignment. Or, you may use equivalent semaphores from another library.

Demo your software and submit hard copy of your source code for grading. Along with a software description, include a small paragraph to explain whether or not your solution exhibits starvation.