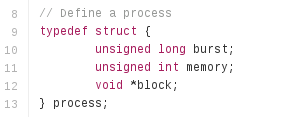
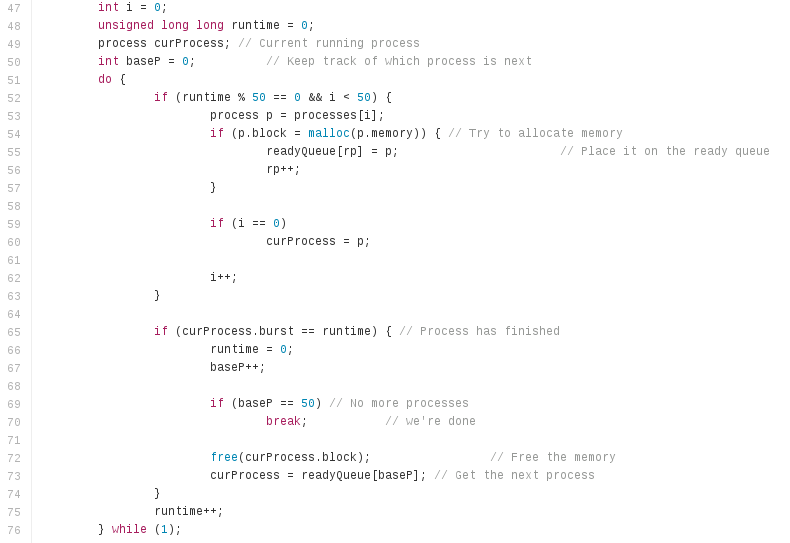
For this project, we developed a program that built upon the first program - simulates a scheduler by assigning processes to a set of processes. We generated the processes with the same requirements from project one. This includes the memory of each process and their corresponding burst times. But it now also includes a member “block” which is a pointer to nothing indicating the memory size requirement of the process. Given the project instructions, we assume that a random process arrives in the system every 50 cycles and that once a process has started, it will run until it has completed – no context switching. Also given, the process can only start if the memory requirements are met. Another aspect of this project is to compare the execution times of the system calls malloc() and free() versus our self-developed malloc() and free() calls.



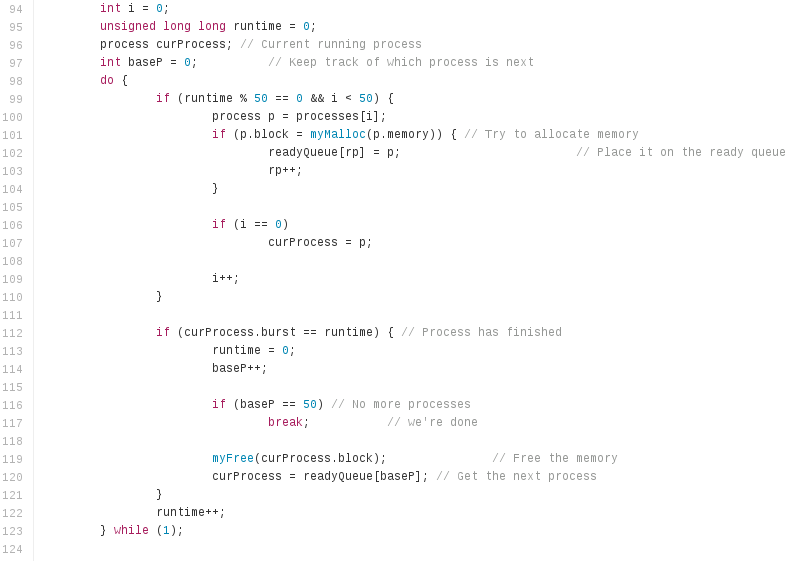
Struct that defines a process.

Problem 1 is mostly straightforward. We are assuming all fifty processes to take less than 10MB of space. We then measure the time it takes for the simulation system to run through all fifty processes dynamically allocating and de-allocating the memory for each one. Below is the main loop that accomplishes this.



In the loop, we test if the process can be run based on the amount of memory available. If so, the process gets put on the ready queue and we increase the process count. Then we test if the process has finished execution (from its burst time), free the memory, and get the next process in the ready queue. This continues until all fifty processes have run. Before and after this loop is where we calculate its running time using 'gettimeofday'.

For Problem 2, we developed our own memory management system with the corresponding myMalloc() and myFree() function calls. The goal here is to see an improvement in performance versus the malloc() and free() system calls because of there being less context switching in the system. There is only one initial malloc() call in the beginning, to get the 10MB chunk of memory for all of the myMalloc() and myFree() calls. The main loop in this function for Problem 2 can be seen below.



The only difference between the two loops (for Problems 1 & 2) are the calls to the aforementioned methods. Since the gettimeofday function is called again before and after this loop, we get an accurate measurement of the time it took for all fifty processes to complete. The myMalloc and myFree functions can be seen below. These are located in the blk.c file.

