# Main Controller

**Author: Zachary Conner**

*Note: This document will cover the main method and it’s supporting functions. The I/O functions are not included.*

## Functions

* allProcsFinished()
  + Inputs: void
  + Outputs: bool
  + Function Purpose: This function loops through each of the processes, and checks to see if they are all in the EXIT state.
* Delay()
  + Inputs: *void*
  + Outputs: *void*
  + Function Purpose: This is a simple platform independent delay function that uses a loop and counter. It is used to slow down execution in auto run mode.

## Variables

* Vector<pcb> masterArray
  + This is a global that holds all of the processes in the system. The processes never move from this data structure once they are placed in it. All references to processes within the system are done using pointers to the objects within this data structure.
* **Control Variables**
  + Bool changeOccured: This is set to false at the beginning of each loop iteration. It is then set to true within the loop if some sort of change occurs. This is used to decide whether or not to output a new version of the display.
  + Bool autoPlay: This is set to true if the user decides to start the run-until-completion feature.
  + PCB\* runningJob: This is a pointer to the current running job. It valid, this job will be “executed” during the cycle.
  + Int timesliceTracker: This is a counter used to track how long the current running job has been running.
  + Int totalCycleTracker: This counts how many cycles the system has been running for.
* **Module Instances**
  + HLS hls: This is an instance of the High Level Scheduler module. See HLS.docx for further information on this module.
  + MemoryManager memoryManager: This is an instance of the Memory Manager module. See MemoryManager.docx for further information on this module.
  + Dispatcher dispatcher: This is an instance of the Dispatcher module. See Dispatcher.docx for further information on this module.
  + InterruptManager interruptManager: This is an instance of the Interrupt Manager module. See InterruptManager.docx for further information on this module.

## High-Level Description

The overall function of the main controller is to act as the glue that brings all the different parts of the operating system together. It initializes the system, and it implements the execution cycle logic. It is the main entry point of the program.

## Execution Cycle Steps

1. Check if all processes have completed. If so, break out of the loop.
2. If not auto playing and a change occurred last cycle, get the user input to move along.
   1. Otherwise, just keep executing.
      1. If a change occurred and we are auto playing, run delay()
   2. If the user inputs ‘r’, set autoPlay to true.
3. Reset the changeOccured control variable to false.
4. Check each process’s entry time.
   1. If it is the current cycle, use HLS to add the process to NEW.
5. Run the HLS. This will move NEW processes to READY.
   1. See HLS documentation for details.
6. Run the Interrupt Manager. This will return any processes that have finished their IO request.
   1. If processes are returned, use the Dispatcher to add them to READY.
7. If the running job is invalid or NULL, run the Dispatcher to get a new running job.
   1. If we do this, reset the time slice tracker.
   2. See Dispatcher documentation for details.
8. Execute the running process if we have one.
   1. Increments the job’s currentExecutionCycle variable.
   2. Increments the time slice tracker.
9. Check to see if the running job has an I/O request on the current execution cycle.
   1. If so, use the Interrupt Manager to add it to BLOCKED.
   2. Reset the time slice tracker, and set the running job to NULL.
10. Check to see if the current running job has finished. If so:
    1. Move it to COMPLETE.
    2. Set the running job to NULL.
    3. Deallocate memory for the job using the Memory Manager.
       1. See Memory Manager documentation for more info.
    4. Reset the time slice tracker.
11. Otherwise, check to see if the current running job has hit its time slice. If so:
    1. Use the Dispatcher to add it to READY.
    2. Set the running job to NULL.
    3. Reset the time slice tracker.
12. If a change occurred during the cycle, print the display.
    1. See display documentation for more information.