# HLS.cpp – High Level Scheduler

## Functions

a. addToNew()

Inputs: pcb\* proc

Outputs: Void

Function Purpose: The purpose of this function is to add PCB objects into the New State.

b. run()

Inputs: Dispatcher\* disp, MemoryManager\* mm

Outputs: Boolean value

Function Purpose: The purpose of this function is to iterate through the New State and then to switch jobs, when necessary, to the Ready State. Jobs are checked before the switch, however, to ensure that memory can be correctly allocated by querying the Memory Manager. The Boolean output will allow for checking as to whether a job was successfully switched to the Ready State.

c. getNewProcs()

Inputs: void

Outputs: Vector of PCB object pointers

Function Purpose: The purpose of this function is simply to return references to the PCB objects in New State for system status printing purposes.

d. getNewStateSize()

Inputs: Void

Outputs: Size of the New State vector of PCB object references

Function Purpose: The purpose of this function is simply to allow other classes to request and receive the length of the New State vector.

## Data Structures Used

a. New State, Vector of PCB object pointers

## High-Level Description

The New State, as a definition, will contain PCB objects which have entered the system, but do not yet have memory allocated to them. The Ready State, as a definition, will contain PCB objects which have entered the system and do have memory allocated to them. The overall high-level purpose of the HLS will be to add jobs to the New State, and then to add those jobs to the Ready state when appropriate. When appropriate, here, refers to when memory is successfully allocated to the jobs. The functions of this HLS module do exactly this, with the added functionality of being able to request the New State in its entirety by making use of the “Print New State Stats” function described above.

## Low-Level Description

The High-Level Scheduler will remain idle until the main controller (in this case the main method) or another class calls upon its functionalities. Thus, the inputs will all come from whichever class called upon the HLS, and the destination of the outputs follows suit.

If the addToNew() is called, the function will:

1. Set the passed-in PCB object’s “State” value to New

2. Insert the passed-in job references into the New State vector

NOTE: Decisions about inputting new jobs are made in the main controller. This function of the HLS only adds the functionality of adding jobs to the New State.

If the run() function is called, the function will:

1. Create a new temporary vector of PCB object references

2. Iterate through the New State vector of PCB object references and transfer the PCB object reference from NewState to the temporary vector if the Memory Manager successfully allocates memory to it (See Memory Manager System Documentation).

3. Send an instance of the temporary vector created to the Dispatcher’s addToReady() function (See Dispatcher System Documentation).

4. Return false is the temporary vector created is empty, true otherwise.

If the getNewProcs() function is called, the function will:

1. Return the New State vector of PCB object references.

If the getNewStateSize() function is called, the function will:

1. Return the size of the New State vector of PCB object references.