# Dispatcher

Author: Zachary Clute

## Functions

* run()
  + Inputs: void
  + Outputs: pcb\*
  + Purpose: To get a new pointer to a pcb process which will be set to running. run() will get the new pcb pointer from a list of pointers passed in from addToReady(vector<pcb\*>)
* getRunningProcess()
  + Inputs: void
  + Outputs: pcb\*
  + Purpose: Returns the currently running process which was set by run()
* moveProcToExit(pcb\*)
  + Inputs: pcb\*
  + Outputs: void
  + Purpose: to move a running process to exit. The passed in pcb must match getRunningProcess() or else an error is thrown.
* getReadyProcs()
  + Inputs: void
  + Outputs: vector<pcb\*>
  + Purpose: returns the list of processes waiting to be run which is initially passed in by addToReady(vector<pcb\*>)
* getExitProcs()
  + Inputs: void
  + Outputs: vector<pcb\*>
  + Purpose: returns a list of processes which were moved to exit by moveProcToExit(pcb\*)
* addToReady(vector<pcb\*>)
  + Inputs: vector<pcb\*>
  + Outputs: void
  + Purpose: Sets the state of passed in pcb\* to ready and adds them to the readyQueue list.s
* debug()
  + Input: void
  + Output: void
  + Purpose: prints debug information to the screen

## Data Structures Used

* vector<pcb\*> readyQueue
  + Purpose: to hold a list of pcb\* which are ready to be ran
* vector<pcb\*> exitList
  + Purpose: to hold a list of pcb\* which were sent to exit
* pcb\* runningProcess
  + Purpose: holds the currently running process

## High-Level Description

The purpose of this class is to move processes from a ready queue to running and then to an exit list one at a time. The initial ready queue is set outside the class by calling addToReady() and run() pops off the front of the ready queue and sets that popped process to running. Finally moveToExit() takes the running process and pushes it onto the exitList.

## Low-Level Description

A Dispatcher object has the following life cycle:

1. Initialization with the default constructor Dispatcher(). The constructor takes no arguments and returns a Dispatcher object.
2. addToReady(vector<pcb\*>) must be called before anything else to ensure “vector<pcb\*> readyQueue” is filled. This is important since the other methods manipulate readyQueue. addToReady not only fills the readyQueue but also sets the state of the passed in pcbs to READY by looping through the passed in vector and setting each pcb\*->state = READY
3. run() is called which pops the readyQueue and returns a pcb\* process which can be used outside the Dispatcher object. Run() will also set it’s own internal “pcb\* running” state to the same returned process to ensure Dispatcher has an accurate environment. Run can only be called if the readyQueue is filled, otherwise it will fail and throw an error.
4. moveProcToExit(pcb\*) can then be called to move a running process to exit when it’s completed. It’s important to note you can only move the running process to exit so the dispatcher will perform a check on what is passed into moveToExit to ensure it’s the running process or throw an error.

If a process needs to be moved out of running by another object (such as the interruptmanager) simply call run again to get a new process to replace it. There is also a debug function which will print out the readyQueue, and exitList in a column.