The first thing the main.cpp program does is to spawn a thread whose job it is to make three threads representing the three CPUs and three short-term schedulers. Main.cpp then spawns a thread simulating an I/O device which randomly unblocks processes from the blocked queue. It then calls the long-term scheduler. After the long-term scheduler returns, it attempts to join the initializing thread and the I/O device.

Processes are simulated with an array of instructions of random length containing randomly either CPU-bound computations or traps to I/O.

The CPUs take processes as input (usually from the ready queue) and loop through the instructions of a process. If the process calls for I/O, it is added to the blocked queue. If it is detected that a process is hogging the CPU, it is timed out and added back to the ready queue.

The three short-term schedulers attempt to join the CPU threads, and then make them again with a new process from the ready queue. If none is available, they block (in the method readyqueue::pop()).

The I/O device continually unblocks a random process from the blocked queue, and then waits.

The ready queue is where the processes stay which are ready to be given to the CPU because they have been allocated all the resources they need except the CPU. The blocked queue is where processes go which are waiting on an I/O event.

The long-term scheduler simply creates new processes and adds them to the ready queue. It terminates once it has added schedulers::TERMINATE\_NOW processes to the ready queue. The I/O device and the short-term schedulers terminate once this condition has been met and additionally there are no processes left in the ready or blocked queues.