1. yield() – process completes, yields the processor to perform an I/O request
2. wake\_up() – process that previously yielded completes its I/O, is ready for CPU operations
   1. called when a process in the NEW state becomes runnable
3. preempt() – for round robin; preempting a CPU-bound process
4. terminate() – process exits or is killed
5. idle() – waits for a new process to be added to the ready queue

void enqueue(\*queue, \*process);

* adds a process to the ready queue
* sets condition variable that process was added

pcb\_t dequeue(\*queue);

* finds the next process to remove according to the scheduling algorithm
* remove it from queue
* return pointer

void idle(cpu\_id);

* block while ready queue is empty
* uses a condition variable that is signaled whenever a process is added to the ready queue
* cpu\_id : cpu that is waiting for the process to come in

void preempt(cpu\_id);

* round robin handler
* preemptive priority scheduling
* places current process back in ready queue
* call schedule() to select a new runnable process
* cpu\_id : cpu we want to preempt

void yield (cpu\_id);

* process gives up CPU to perform an I/O request
* cpu\_id : cpu that is yielded by the process

void terminate(cpu\_id);

* handler for when a process completes
* cpu\_id : cpu to be terminated

void wake\_up (pcb\_t \*process);

* handler for when process’s I/O completes
* needs to handle priority
* use get\_current\_time()
* process : process that finished I/O and is ready to run CPU

A screen shot of a computer program

Description automatically generated