```c
/* Implements a first-come, first-served scheduler.
 *
 * Created by Henry Walker, 27 September 2004
 * Last modified by Janet Davis, 25 September 2010
 * Revised by Jerod Weinman, 10 August 2012
 */

#include <stdlib.h>
#include <stdio.h>
#include "scheduler.h"

/* The ready queue */
job_queue_t ready;

/* Initializes the ready queue.  Call before any other functions. */
void ready_queue_init(void) {
  ready.first = NULL;
  ready.last  = NULL;
}

/* Returns true or false, according to whether any jobs are waiting
 * in the ready queue.
 */
int ready_queue_empty(void) {
  return (ready.first == NULL);
}

/* Adds the specified job to the ready queue.
 *
 * Preconditions:
 *   job != NULL
 * Postconditions:
 *   Creates a new node for the job
 *   job is inserted at the end of the queue
 */
void ready_queue_insert(job_t* job) {
  job_queue_node_t* node
      = (job_queue_node_t *)malloc(sizeof(job_queue_node_t));

  if (!node) {
    perror("Unable to allocate job node");
    exit(EXIT_FAILURE);
  }

  /* copy event data to new node */
  node->job = job;

  /* insert node into ready queue*/
  node->next = NULL;
  if (ready_queue_empty()) {
    ready.first = node;
    ready.last  = node;
  } else {
    ready.last->next  = node; /* add after current last */
    ready.last  = node;       /* make new node last */
  }
}

/* Removes and returns the job at the head of the ready queue.
 *
 * Postconditions:
 *   If ready_queue_empty(), returns NULL
 *   Otherwise, returns head job and frees the associated node
 */
job_t* ready_queue_select(void) {
  job_t* job;
  job_queue_node_t* old_node;

  /* if no jobs are ready, return NULL */
  if (ready_queue_empty())
    return NULL;

  /* next job is at front of queue */
  job = ready.first->job;

  /* record node at front of queue */
  old_node = ready.first;
  ready.first = ready.first->next;

  /* check if queue is -now- empty */
  if (ready_queue_empty()) {
    ready.last = NULL; /* make last pointer consistent */
  }

  /* return old front of queue to memory pool */
  free(old_node);

  return job;
}
```