```
/* This file contains the main program of MINIX as well as its shutdown code.
 * The routine main() initializes the system and starts the ball rolling by
 * setting up the process table, interrupt vectors, and scheduling each task
 * to run to initialize itself.
 * The routine shutdown() does the opposite and brings down MINIX.
 *
 * The entries into this file are:
 *   main:              MINIX main program
 *   prepare_shutdown:  prepare to take MINIX down
 */
#include "kernel.h"
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <a.out.h>
#include <minix/callnr.h>
#include <minix/com.h>
#include <minix/endpoint.h>
#include "proc.h"

/* Prototype declarations for PRIVATE functions. */
FORWARD _PROTOTYPE( void announce, (void));
FORWARD _PROTOTYPE( void shutdown, (timer_t *));

/*===========================================================================*
 *                                main                                       *
 *===========================================================================*/
PUBLIC void main()
{
/* Start the ball rolling. */
  struct boot_image *ip;        /* boot image pointer */
  register struct proc *rp;     /* process pointer */
  register struct priv *sp;     /* privilege structure pointer */
  register int i, s;
  int hdrindex;                 /* index to array of a.out headers */
  phys_clicks text_base;
  vir_clicks text_clicks, data_clicks, st_clicks;
  reg_t ktsb;                   /* kernel task stack base */
  struct exec e_hdr;            /* for a copy of an a.out header */

  /* Clear the process table. Anounce each slot as empty and set up mappings
   * for proc_addr() and proc_nr() macros. Do the same for the table with
   * privilege structures for the system processes.
   */
  for (rp = BEG_PROC_ADDR, i = -NR_TASKS; rp < END_PROC_ADDR; ++rp, ++i) {
        rp->p_rts_flags = SLOT_FREE;            /* initialize free slot */
        rp->p_nr = i;                           /* proc number from ptr */
        rp->p_endpoint = _ENDPOINT(0, rp->p_nr); /* generation no. 0 */
        (pproc_addr + NR_TASKS)[i] = rp;        /* proc ptr from number */
        rp->p_recent_time = 0;                  /* recent cpu time */
  }
  for (sp = BEG_PRIV_ADDR, i = 0; sp < END_PRIV_ADDR; ++sp, ++i) {
        sp->s_proc_nr = NONE;                   /* initialize as free */
        sp->s_id = i;                           /* priv structure index */
        ppriv_addr[i] = sp;                     /* priv ptr from number */
  }

  /* Set up proc table entries for processes in boot image.  The stacks of the
   * kernel tasks are initialized to an array in data space.  The stacks
   * of the servers have been added to the data segment by the monitor, so
   * the stack pointer is set to the end of the data segment.  All the
   * processes are in low memory on the 8086.  On the 386 only the kernel
   * is in low memory, the rest is loaded in extended memory.
   */

```