```c
/* This file contains information dump procedures. During the initialization
 * of the Information Service 'known' function keys are registered at the TTY
 * server in order to receive a notification if one is pressed. Here, the
 * corresponding dump procedure is called.
 *
 * The entry points into this file are
 *   handle_fkey:       handle a function key pressed notification
 */

#include "inc.h"

/* Define hooks for the debugging dumps. This table maps function keys
 * onto a specific dump and provides a description for it.
 */
#define NHOOKS 18

struct hook_entry {
        int key;
        void (*function)(void);
        char *name;
} hooks[NHOOKS] = {
        { F1,   proctab_dmp, "Kernel process table" },
        { F2,   memmap_dmp, "Process memory maps" },
        { F3,   image_dmp, "System image" },
        /* { F4,        privileges_dmp, "Process privileges" }, */
        { F4,   messaging_dmp, "Messaging activity" },
        { F5,   monparams_dmp, "Boot monitor parameters" },
        { F6,   irqtab_dmp, "IRQ hooks and policies" },
        { F7,   kmessages_dmp, "Kernel messages" },
        { F9,   sched_dmp, "Scheduling queues" },
        { F10,  kenv_dmp, "Kernel parameters" },
        { F11,  timing_dmp, "Timing details (if enabled)" },
        { SF1,  mproc_dmp, "Process manager process table" },
        { SF2,  sigaction_dmp, "Signals" },
        { SF3,  fproc_dmp, "Filesystem process table" },
        { SF4,  dtab_dmp, "Device/Driver mapping" },
        { SF5,  mapping_dmp, "Print key mappings" },
        { SF6,  rproc_dmp, "Reincarnation server process table" },
        { SF7,  holes_dmp, "Memory free list" },
        { SF8,  data_store_dmp, "Data store contents" },
};

/*===========================================================================*
 *                              handle_fkey                                  *
 *===========================================================================*/
#define pressed(k) ((F1<=(k)&&(k)<=F12 && bit_isset(m->FKEY_FKEYS,((k)-F1+1)))\
        || (SF1<=(k) && (k)<=SF12 && bit_isset(m->FKEY_SFKEYS, ((k)-SF1+1))))
PUBLIC int do_fkey_pressed(m)
message *m;                                     /* notification message */
{
  int s, h;

  /* The notification message does not convey any information, other
   * than that some function keys have been pressed. Ask TTY for details.
   */
  m->m_type = FKEY_CONTROL;
  m->FKEY_REQUEST = FKEY_EVENTS;
  if (OK != (s=sendrec(TTY_PROC_NR, m)))
      report("IS", "warning, sendrec to TTY failed", s);

  /* Now check which keys were pressed: F1-F12, SF1-SF12. */
  for(h=0; h < NHOOKS; h++)
      if(pressed(hooks[h].key))
          hooks[h].function();

  /* Don't send a reply message. */
  return(EDONTREPLY);
}

/*===========================================================================*
 *                              key_name                                     *
 *===========================================================================*/
PRIVATE char *key_name(int key)
{
        static char name[15];

        if(key >= F1 && key <= F12)
                sprintf(name, " F%d", key - F1 + 1);
        else if(key >= SF1 && key <= SF12)
                sprintf(name, "Shift+F%d", key - SF1 + 1);
        else
                sprintf(name, "?");
        return name;
}


/*===========================================================================*
 *                              mapping_dmp                                  *
 *===========================================================================*/
PUBLIC void mapping_dmp(void)
{
  int h;

  printf("Function key mappings for debug dumps in IS server.\n");
  printf("        Key   Description\n");
  printf("-------------------------------------");
  printf("------------------------------------\n");

  for(h=0; h < NHOOKS; h++)
      printf(" %10s.  %s\n", key_name(hooks[h].key), hooks[h].name);
  printf("\n");
}


```