

Sistemas Operativos.
PrácticaTarea 04

Alumno:
Abrego Álavrez Jonathan

Ayudante de Laboratorio:
Cinthia Rodríguez Maya

Ayudante de Teoría:
Yessica Martínez Reyes.

30 de marzo de 2014

1. Lo primero que hice fue ubicar la dirección de los archivos a usar
 - **tty.c** que se encuentra en **/usr/src/drivers/tty/tty.c**
 - **config.h** que se encuentra en **/usr/src/include/minix/config.h**



Figura 1: Screenshot tty.c

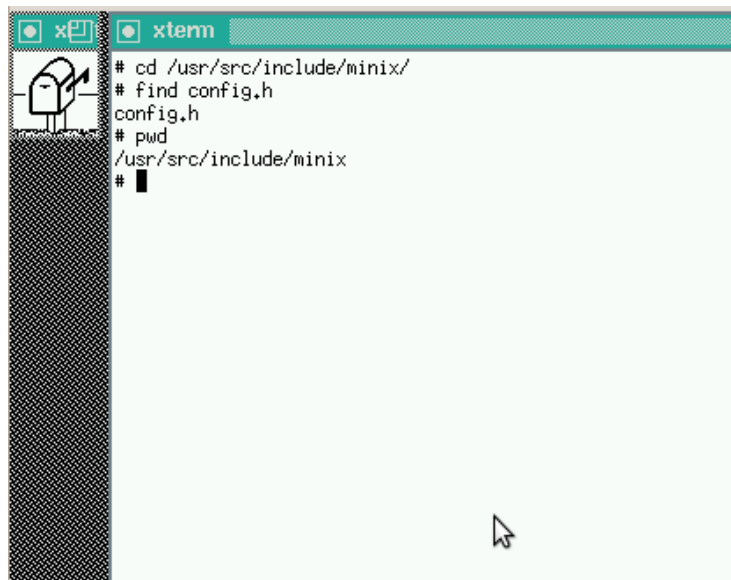
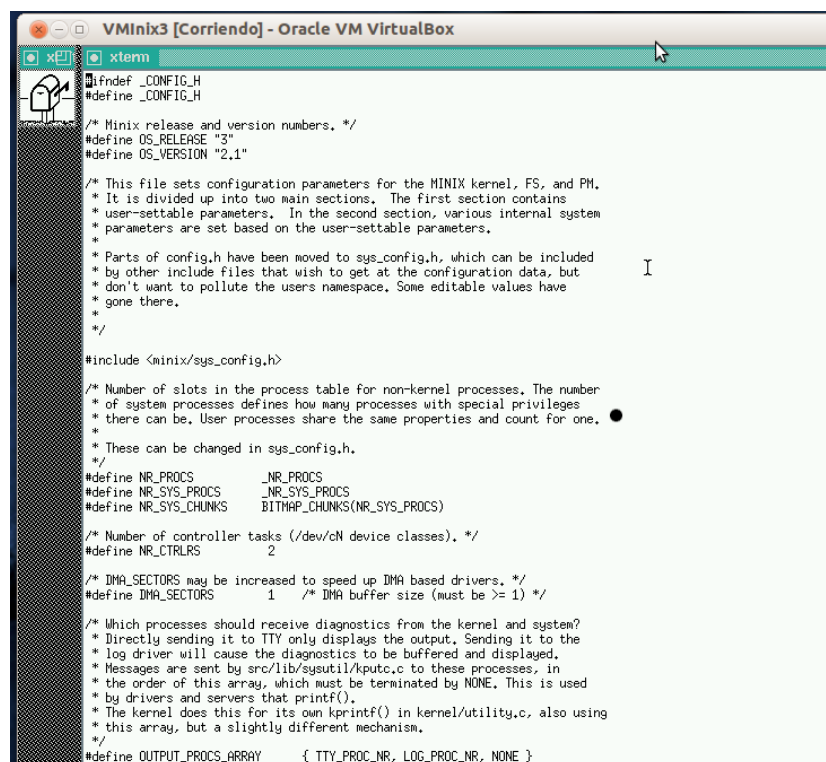


Figura 2: Screenshot config.h

2. Vista del archivo **config.h**



```
#ifndef _CONFIG_H
#define _CONFIG_H

/* Minix release and version numbers. */
#define OS_RELEASE "3"
#define OS_VERSION "2.1"

/* This file sets configuration parameters for the MINIX kernel, FS, and PM.
 * It is divided up into two main sections. The first section contains
 * user-settable parameters. In the second section, various internal system
 * parameters are set based on the user-settable parameters.
 *
 * Parts of config.h have been moved to sys_config.h, which can be included
 * by other include files that wish to get at the configuration data, but
 * don't want to pollute the users namespace. Some editable values have
 * gone there.
 */

#include <minix/sys_config.h>

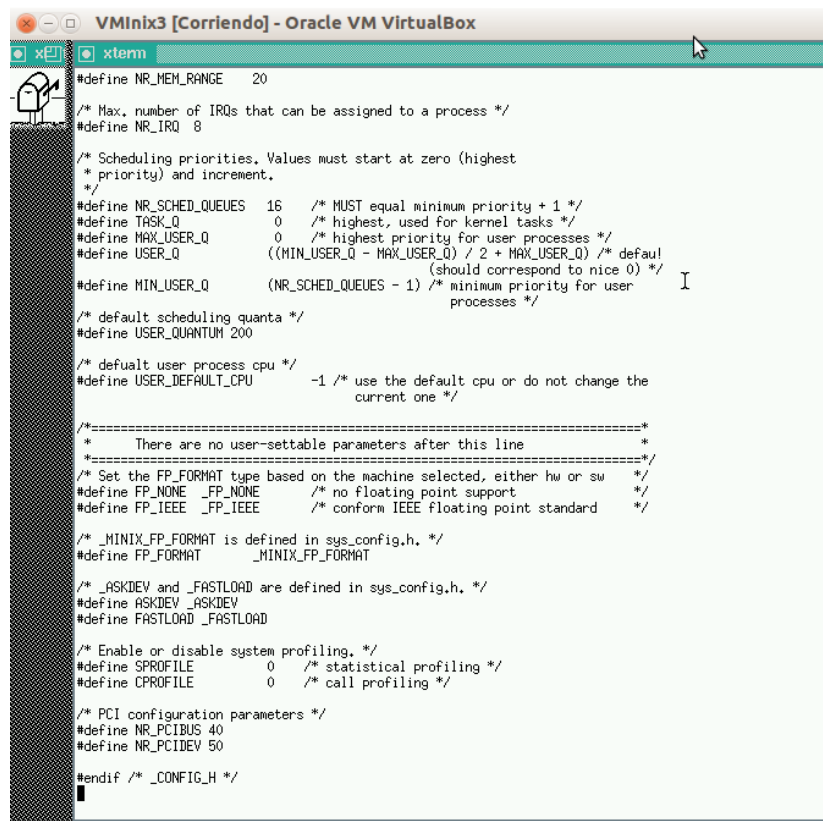
/* Number of slots in the process table for non-kernel processes. The number
 * of system processes defines how many processes with special privileges
 * there can be. User processes share the same properties and count for one.
 *
 * These can be changed in sys_config.h.
 */
#define NR_PROCS      _NR_PROCS
#define NR_SYS_PROCS  _NR_SYS_PROCS
#define NR_SYS_CHUNKS BITMAP_CHUNKS(NR_SYS_PROCS)

/* Number of controller tasks (/dev/cN device classes). */
#define NR_CTRLRS     2

/* DMA_SECTORS may be increased to speed up DMA based drivers. */
#define DMA_SECTORS   1 /* DMA buffer size (must be >= 1) */

/* Which processes should receive diagnostics from the kernel and system?
 * Directly sending it to TTY only displays the output. Sending it to the
 * log driver will cause the diagnostics to be buffered and displayed.
 * Messages are sent by src/lib/systutil/putc.c to these processes, in
 * the order of this array, which must be terminated by NONE. This is used
 * by drivers and servers that printf().
 * The kernel does this for its own kprintf() in kernel/utility.c, also using
 * this array, but a slightly different mechanism.
 */
#define OUTPUT_PROCS_ARRAY { TTY_PROC_NR, LOG_PROC_NR, NONE }
```

Figura 3: Screenshot config.h



```
/* config.h */

#define NR_MEM_RANGE 20

/* Max. number of IRQs that can be assigned to a process */
#define NR_IRQ 8

/* Scheduling priorities. Values must start at zero (highest
 * priority) and increment.
 */
#define NR_SCHED_QUEUES 16 /* MUST equal minimum priority + 1 */
#define TASK_Q 0 /* highest, used for kernel tasks */
#define MAX_USER_Q 0 /* highest priority for user processes */
#define USER_Q ((MIN_USER_Q - MAX_USER_Q) / 2 + MAX_USER_Q) /* default
 * (should correspond to nice 0) */
#define MIN_USER_Q (NR_SCHED_QUEUES - 1) /* minimum priority for user
 * processes */

/* default scheduling quanta */
#define USER_QUANTUM 200

/* default user process cpu */
#define USER_DEFAULT_CPU -1 /* use the default cpu or do not change the
 * current one */

/*-----*
 * There are no user-settable parameters after this line
 *-----*/

/* Set the FP_FORMAT type based on the machine selected, either hw or sw */
#define FP_NONE _FP_NONE /* no floating point support */
#define FP_IEEE _FP_IEEE /* conform IEEE floating point standard */

/* _MINIX_FP_FORMAT is defined in sys_config.h. */
#define FP_FORMAT _MINIX_FP_FORMAT

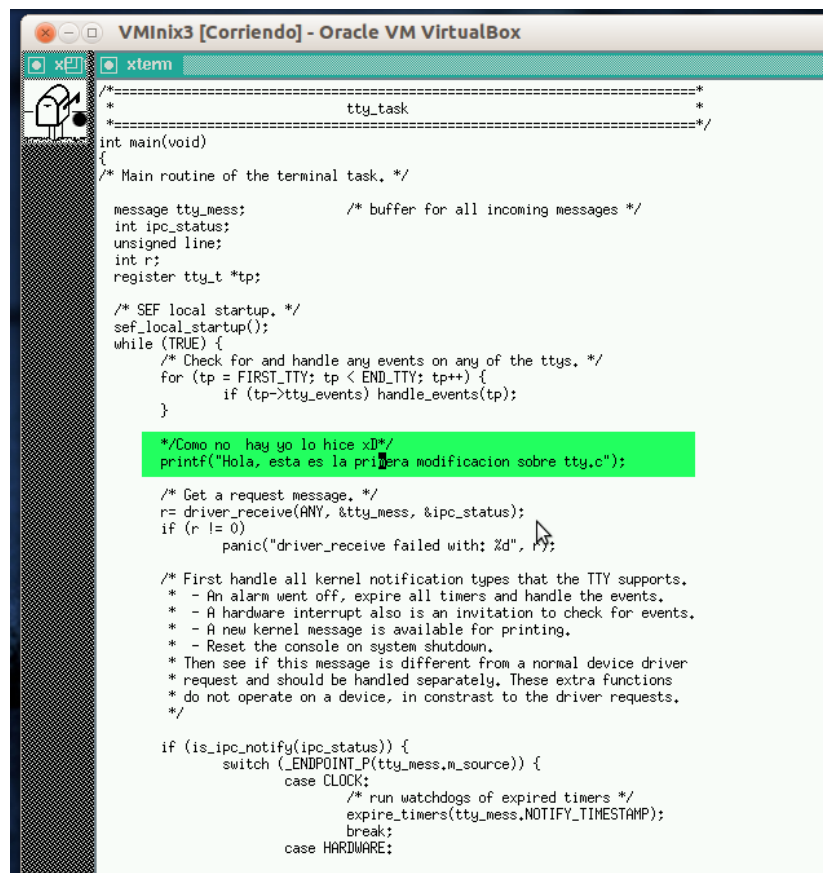
/* _ASKDEV and _FASTLOAD are defined in sys_config.h. */
#define ASKDEV _ASKDEV
#define FASTLOAD _FASTLOAD

/* Enable or disable system profiling. */
#define SPROFILE 0 /* statistical profiling */
#define CPROFILE 0 /* call profiling */

/* PCI configuration parameters */
#define NR_PCIBUS 40
#define NR_PCIDEV 50

#endif /* _CONFIG_H */
```

Figura 4: Screenshot config.h



```
/*=====
 *                               tty_task
 *=====*/

int main(void)
{
    /* Main routine of the terminal task. */

    message tty_mess;          /* buffer for all incoming messages */
    int ipc_status;
    unsigned line;
    int r;
    register tty_t *tp;

    /* SEF local startup. */
    sef_local_startup();
    while (TRUE) {
        /* Check for and handle any events on any of the ttys. */
        for (tp = FIRST_TTY; tp < END_TTY; tp++) {
            if (tp->tty_events) handle_events(tp);
        }

        /* Como no hay yo lo hice xD */
        printf("Hola, esta es la primera modificacion sobre tty.c");

        /* Get a request message. */
        r = driver_receive(ANY, &tty_mess, &ipc_status);
        if (r != 0)
            panic("driver_receive failed with: %d", r);

        /* First handle all kernel notification types that the TTY supports.
         * - An alarm went off, expire all timers and handle the events.
         * - A hardware interrupt also is an invitation to check for events.
         * - A new kernel message is available for printing.
         * - Reset the console on system shutdown.
         * Then see if this message is different from a normal device driver
         * request and should be handled separately. These extra functions
         * do not operate on a device, in contrast to the driver requests.
         */

        if (is_ipc_notify(ipc_status)) {
            switch (_ENDPOINT_P(tty_mess.m_source)) {
                case CLOCK:
                    /* run watchdogs of expired timers */
                    expire_timers(tty_mess.NOTIFY_TIMESTAMP);
                    break;
                case HARDWARE:
            }
        }
    }
}
```

Figura 5: Screenshot Modificación de tty.c