Trabajo Practico Especial
Arquitectura de computadoras

Martin Victory 56086
Florencia Cavallin 56015
Segundo Fariña 56176

Introducción

Se implemento un mini sistema operativo en base a un kernel booteable por Pure64 que fue provisto por la cátedra. El mismo consta de un espacio de kernel y un espacio de usuario claramente definidos los cuales se comunican por medio de la interrupción de software 80h.

Kernel Space

IDT

Lo primero que se hizo fue implementar la *“Interrupt Descriptor Table”* , la cual es la estructura por la que el procesador determina que rutina de atención debe llevar acabo cuando se lo interrumpe. Existen tres tipos de interrupciones las cuales son excepciones de procesador, interrupciones de hardware e interrupciones de software. Se manejaron solo unas pocas interrupciones que formaban parte de este trabajo, las cuales se detallan a continuación.

Las interrupciones de hardware llegan al procesador por medio del PIC, por lo cual se debió habilitar la escucha de estas en este controlador, las cuales realizan una “*Interruption Request” (IRQ)* especifica al periférico. Una vez que recibe esta IRQ el procesador busca en la IDT la rutina a realizar.

Perisfercio	Indice en la tabla	IRQ	Plc
Teclado	0x21	1	Master
Placa ethernet	0x2B	11	Slave
System Calls	0x80	-	

Driver de Teclado

El manejo de las interrupciones de teclado se hacen por medio de un pequeño driver en el cual se recibe un *scan-code* representando una tecla la cual es convertida a su correspondiente carácter ASCII. Esos caracteres se almacenan en un buffer circular, el cual proporciona las teclas ingresadas al *systemcall read* cuando es invocado.

Driver de Video

La salida estadard del Sistema Operativo es la pantalla, por lo que para poder escribir en esta, fue necesario crear un pequeño driver de video. Se decidió que este driver de video debía saber como imprimir cada uno de los caracteres especificados. La pantalla se encuentra mapeada en memoria en una dirección fija y especifica, donde un byte representa al carácter a imprimir y el byte siguiente el color y fondo. Para este trabajo, los colores son fijos, por lo que solo se manejó la manipulación de los caracteres en pantalla.

RTL

El centro de este trabajo fue crear un *driver* para la placa de red RTL8139, comúnmente utilizada en muchas computadoras. El *driver* debía permitir la comunicación con otras placas de red y así enviar y recibir mensajes.

Lo primero fue buscar la ubicación de la placa RTL en los buses de la PCI, para así poder identificar en que posición se la mapeaba en memoria, y que interrupción generaba. Una vez encontrada, estos valores se *harcoderon* en el código, ya que el trabajo no constaba de realizar el proceso de búsqueda dinámica de la placa. Para que la placa pueda enviar y recibir mensajes fue necesario activarle el *Direct Memory Access*, para que pueda escribir directamente en la memoria, ya que la placa funciona con un buffer de lectura en el que se almacenan los mensajes recibidos, y 4 *buffers* de escritura en los que se colocan los mensajes a enviar (aunque nosotros utilizamos uno solo).

Al momento de enviar mensajes, el *driver* crea un *frame ethernet* el cual se conforma por la dirección MAC de la placa de destino, la dirección MAC de la placa que envía el mensaje, un mensaje, la longitud del mismo y un *frame check* (el cual esta en des uso en este trabajo). Este *frame* es copiado en el buffer de escritura y luego se escribe la longitud del mensaje a enviar en un registro de la placa, proceso que directamente envía el mensaje.

Por otro lado, la placa esta todo el tiempo en escucha de mensajes nuevos, los cuales los guarda en el buffer de lectura y lanza inmediatamente después una interrupción de hardware. Dado que recibe todos los mensajes, cuando se ejecuta la rutina de atención, se guardan solo los mensajes que son destinados a esta, en un buffer circular, del cual luego se extraen mensajes al solicitarse desde *userland*.

Para este trabajo solo trabajamos con dos tipos de mensajes: *Broadcast* que son capturados por todos los dispositivos, *Private* que solo son guardados cuando la MAC coincide con la del dispositivo receptor.

SystemCalls

La comunicación entre el kernel y el user se realizan puramente mediante interrupciones de software, denominadas *system calls.* Existe una única función de *system call* genérica, la cual luego se divide en los distintos *system calls* específicos.

Dos de estas implementaciones de system call en especifico son los casos de *read* y *write*. Estas funciones son utilizadas para la leer y escribir en un *file descriptor.* En nuestra implementación se usan dos file descriptor distintos: para la entrada estándar (file descriptor 1) y para la lectura y escritura en la placa RTL (file descriptor 2).

En segundo lugar se encuentran las funciones *clear screen* para borrar lo escrito en la pantalla y memory management con el objetivo de asignar y liberar la memoria. Como en el objetivo de este trabajo no es el dominio de distribución de memoria, la asignación devuelve un puntero por arriba del 10 mega y la liberación de esta no tiene efecto.

Como ya antes mencionado todas estas funciones utilizan la función principal system call; esta recibe un numero que representa la lectura, escritura, borrado de pantalla o manejo de memoria, al igual que un file descriptor, un buffer que representa a donde se copiara o de donde se leerá la información y por ultimo el tamaño a leer o escribir.

User space

Shell

La *shell* adquiere el papel de redistribuidor de tareas a la hora de relacionarse con el usuario. Le brinda al mismo un menu de acciones posibles para ejecutar. En este caso, los comandos a ingresar son:

- help: dispone en pantalla todos los comandos disponibles
- echo: imprime por pantalla los parámetros insertados
- 2048game: implementacion del juego 2048
- chat: inicia un chat grupal entre todos los usuarios online
- clear: borra todo lo escrito en la pantalla
- hola: saludo de la consola

Libreria estandar

En este trabajo se implementaron las librerías stdlib, stdio y string debido a su importancia a la hora de desarrollar las especificaciones del trabajo practico.

En el caso de stdlib, su propósito fue facilitar la gestión de memoria dinámica y control de procesos. Son ejemplos malloc, realloc y free, los cuales son funciones destacadas en esta librería para el correcto funcionamiento de la asignación de memoria.

Por otro lado se necesito stdio para realizar operaciones de entrada y salida, principalmente para el uso de la pantalla. En esta librería, las funciones de mayor uso serian scanf, printf, puts y sus derivados; cada una realizara su correspondiente systemcall para llevar a cabo su objetivo sin darle el mando al usuario en su camino.

Finalmente, la librería string contiene un paquete de funciones para operar con strings de todas las formas posibles y así facilitar su manejo a la hora de realizar trabajos mas complejos. Entre las mas comunes, se pueden encontrar strlen, strcmp, strcpy y strcat.

Chat

El chat es el encargado de comunicar dos o mas terminales mediante sus Mac Address ya sea creando un chat conjunto de todos los usuarios que se encuentran online o mandando mensajes privados a un usuario especifico.

El funcionamiento del mismo comienza al ingresar el comando *chat,* luego se elige un nombre y se abre automáticamente un chat en el que podrán participar todos los usuarios online. Al unirse al chat, se manda un mensaje a todos los

demás usuarios de tipo *online,* el cual reciben un mensaje con el nombre de usuario y la Mac Address de esa persona que acaba de conectarse y esa información es guardad por cada terminal.

El chat esta pensado como una sala de chat en la cual por default todos los mensajes son mandados a todas las terminales conectadas, con la posibilidad de poder mandar un mensaje privado a alguien conectado mandando @(nombre de usario).

Al tener guardado el nombre de usuario y la Mac Address es posible mandar un mensaje privado a una terminal usando su nombre de usuario el cual a partir de este da la Mac Adress y se manda un mensaje únicamente a esa Mac Adreess.

Por otro lado a la hora de desconectarse se podrá ingresar el comando *exit* el cual actualiza su estado a offline avisando a todos los usuarios su nuevo estado. También es posible visualizar todos los usuarios que se encuentran disponibles tipeando el comando *online*.

Con una mirada mas bien técnica, cuando un nuevo usuario se conecta al chat, su nombre y Mac Address son agregados a una lista. De esta forma será mas fácil localizar ambos atributos cuando el mismo desee mandar un mensaje. Al abandonar la conversación es removido de la lista, ya que la misma contiene solo los usuarios online.

Cuando un usuario desee mandar un mensaje privado deberá ingresar el nombre del contacto, anticipado por el carácter @. Si el usuario esta conectado, es decir si se encuentra en la lista, se encontrara el Mac Address correspondiente y se le enviara el mensaje solo a el. En caso de que el usuario ingresado no se encuentre online no se podrá llevar a cabo esta acción. Por otro lado si el usuario no especifica con que contacto desea comunicarse, entonces el mensaje le llagara a todos los usuarios online por default.