Arquitecturas de Computadoras 72.08
2do Cuatrimestre 2016
TPE - Informe

Debrouvier, Ariel 55382
Matorras Bratsche, Tobías 55376
Nuñez Kohler, Damian Nicolas 54365

Introducción

En este informe se detalla la estructura básica del TPE, así como las diferentes decisiones que se tomaron en cuanto a su implementación.

Objetivo

Realizar un sistema operativo de 64 bits que administre los recursos de hardware de una computadora y muestre las características del modo protegido de Intel.

Estructura Básica

Se diseñó respetando la distinción entre Kernel space y User space. La interacción entre ambos se hace únicamente mediante system calls (a través de la instrucción int 80h de Intel), provistas para acceder a los recursos de la computadora. Las llamadas que se implementaron son las siguientes:

Syscall	RDI	RSI	RDX	RCX
EXIT	0	CODE	-	-
CLEAR	1	-	-	-
READ	2	FILE DESCRIPTOR	BUFFER	SIZE
WRITE	3	FILE DESCRIPTOR	BUFFER	SIZE
VIDEO	4	WIDTH	HEIGHT	BITS PER PIXEL
DRAW	5	ROW	COL	COLOR
SBRK	6	INCREMENT	-	-
TIME	7	-	-	-
DATE	8	-	-	-
SET_DATE	9	DAY	MONTH	YEAR
SET_TIME	10	HOURS	MINUTES	SECONDS
WAIT	11	MILLISECONDS	-	-
SEND	12	DEST	MESSAGE	LENGTH
GET_PACKAGE	13	SOURCE	DEST	MESSAGE

Kernel Space

Kernel

El kernel está conectado directamente con el hardware y es quien provee el acceso al mismo a través de interrupciones. Al bootear el sistema el mismo limpia la pantalla y carga las interrupciones de hardware y software en la IDT, asociando cada una de ellas con su respectiva rutina de atención (ISR). Se utilizó la dirección 0x0 para ubicar la IDT, espacio que Pure ya tiene reservado para colocar esta tabla y de esta manera se evitó hacer el load de la IDT. Las interrupciones de hardware son habilitadas mediante una máscara de bits en el PIC (en nuestro caso, el Timer Tick y el Teclado). Para las interrupciones de software se agrega una entrada en la posición 80 de la IDT, que puede ser accedida mediante la instrucción INT 80h de Intel. Por último, se carga el módulo de código y se espera a que el usuario lo interrumpa.

Keyboard Driver

Cuando se produce una interrupción de teclado, se llama a la función *keyboard_handler*. Esta función lee el puerto 0x60 del mapa de entrada salida para saber si el buffer tiene alguna tecla sin consumir. De ser así se extrae su correspondiente keycode (leyendo el puerto 0x64) y se utiliza un vector para obtener su código ASCII. Una vez obtenida la tecla presionada se la guarda en un buffer (si no es una tecla especial: shift, capslock, etc). El driver de teclado NO imprime los caracteres en pantalla, ésta tarea es realizada por la Shell a través de las system calls READ y WRITE.

Video Driver

Tiene como objetivo administrar la memoria de video en modo texto. Se lo llama cada vez que se tiene que imprimir por pantalla. Cada posición de pantalla está compuesta por dos bytes, uno indica la letra a escribir y el otro el atributo (color). Se decidió que por pantalla se muestre la hora actual del sistema en la esquina superior derecha (esta opción puede ser deshabilitada por el usuario), por lo que hay que considerar que la primer línea de la pantalla no se puede escribir.
Una función de interés es scroll, la cual se invoca cuando la posición actual de de la pantalla es la última (esquina inferior derecha). Dicha función copia toda la pantalla (excepto la primer fila, en la que se encuentra el reloj) a una fila superior, dejando la última libre para su escritura.

Modo Video

QEMU implementa la VGA BIOS del emulador Bochs y se decidió utilizar esta alternativa para ingresar al modo de video. Para acceder al hardware de gráficos emulado, se utilizan dos puertos de entrada/salida de 16 bits y de esta manera se entra en el modo de video con una resolución de 1024x768 y 24 bits de profundidad.

Fue necesario agregar a la línea de ejecución de QEMU, el parámetro -vga std.

La modificación de la memoria de video se realiza a través de una dirección de buffer lineal, la cual se obtiene en el bootloader, realizando la interrupción int 10h. Esta interrupción, se encarga de verificar que el modo VESA es soportado, y almacena en una estructura la dirección requerida, junto con otra información.

La entrada al modo video se realiza a través de la system call VIDEO. La generación de los fractales que se implementaron en el TPE utiliza además la system call DRAW que permite dibujar un pixel en particular.

Vale aclarar que una vez que se entra en modo video no es posible volver al modo texto. Esta sería una de las desventajas del modo que propusimos, pero se decidió aprovechar esta funcionalidad del emulador, sin desechar lo que ya se había realizado sobre la shell de Userland.

RTC (Real Time Clock)

Se programó el RTC para que el usuario pueda obtener y modificar la hora del sistema. La hora actual, los minutos y los segundos se obtienen leyendo el puerto 70h. Se los imprime en la esquina superior derecha de la pantalla durante toda la ejecución de la shell (esta opción se puede habilitar/deshabilitar mediante la syscall TIME). La modificación de la hora y la fecha se realizan mediante las system calls SET_TIME y SET_DATE respectivamente.

PIT (Programmable Interval Timer)

Se programó el timer para suspender el sistema por una cantidad de milisegundos dada (mediante la syscall WAIT). Además se implementó un sistema básico de tareas para que sea posible habilitar/deshabilitar el reloj en la parte superior derecha de la pantalla, entre otras cosas. El sistema de tareas funciona de la siguiente manera:
Se puede agregar una tarea mediante la función *add_task*, que recibe 3 parámetros:

- *seconds*: cada cuantos segundos se ejecuta la tarea
- *handler*: la función que ejecuta la tarea
- *forever*: un booleano que indica si la tarea debe repetirse indefinidamente.

Para eliminar una tarea se usa la función *remove_task*, que recibe el handler correspondiente a la tarea. Se decidió no proveer acceso directo a esta funcionalidad al usuario, es decir, no se agregaron system calls para agregar/remover tareas.

Network Driver

Se emuló con QEMU la placa de red RTL-8139 y se recreó una red con interfaces TAP y un bridge, para así lograr comunicación entre distintas instancias de QEMU.

Para la transmisión de datos se implementó el siguiente formato de paquete.

Mac Destino	Mac Fuente	Data Size	Data
6 bytes	6 bytes	2 bytes	

Desde Userland se permite tanto enviar mensajes privados como broadcast.

Para enviar un mensaje se hace uso de la syscall SEND. Cuando se recibe un mensaje se almacena en un buffer y desde Userland se pueden solicitar aquellos mensajes recibidos que aún no han sido leídos con la syscall GET_PACKET.

User Space

Shell

La interacción con el usuario se realiza mediante un intérprete de comandos. El mismo lee caracteres por entrada estándar hasta que se presiona enter y valida la línea ingresada contra una tabla de comandos disponibles. Si el string representa un comando válido, el mismo se ejecuta produciendo un resultado por pantalla.
La shell contiene un buffer circular de 256 caracteres que se llena a medida que el usuario presiona el teclado. Si la tecla presionada es imprimible, la shell hace el putchar correspondiente. Es diferente al buffer del driver de teclado ya que este no imprime los caracteres, solo los almacena. Vale aclarar que si no se ejecutase la shell, los caracteres presionados igualmente serían almacenados en el buffer de teclado, pero no se verían.

Para una lista completa de los comandos disponibles ver el Manual de Usuario.

Librería estándar de C

Se implementaron las siguientes librerías:

- **stdio**: provee funciones básicas de entrada/salida como putchar, getchar, printf y scanf.
- **stdlib**: solo contiene 3 funciones: malloc, calloc y free. Se utilizó la implementación provista por la cátedra, con la diferencia de que se agregó una system call para aumentar el tamaño del segmento de datos (SBRK - similar a la de Linux)
- **string**: contiene funciones básicas para operar sobre strings (strlen, strcpy, etc)
- **ctype**: contiene funciones básicas para clasificar y operar sobre caracteres.

- **syscalls**: provee una interfaz para acceder a las system calls del Kernel de manera eficiente y clara.

Conclusión

Luego de la realization de este proyecto, se logró determinar que la separación entre kernel space y user space a la hora de diseñar un sistema operativo es indispensable para poder administrar los recursos de la computadora (hardware). Es necesario delimitar correctamente a que puede acceder el usuario y que no pueda interactuar directamente con el hardware, sino que exista una interfaz o intermediario entre este y el sistema.

Con respecto a la implementación, una posible mejora sería el agregado de una consola en modo video, para poder seguir operando luego de entrar en modo video. Solo se debería diseñar la parte de interfaz gráfica, ya que la lógica ya desarrollada, puede ser reutilizada.