Arquitectura de Computadoras
Trabajo Práctico Especial
Informe
(Presentado 2Q2022-Actualizado)

Alumna: Baron, María Mercedes 61187

Profesores: Vallés, Santiago
Merovich, Horacio
Federico Gabriel Ramos

Introducción:

El objetivo de este trabajo práctico especial es implementar un kernel booteable por Pure64 que administre los recursos de hardware de una computadora y muestre características del modo protegido de Intel. Estos recursos anteriormente mencionados deben poder ser utilizados por los usuarios.

El diseño del trabajo práctico especial consta de dos secciones, el Kernel Space y el User Space. La razón de esta separación se encuentra en el hecho de que un usuario no podrá editar secciones sensibles en cuanto al funcionamiento del sistema operativo y la comunicación con el hardware de la computadora. Sin embargo, para que el usuario pueda hacer uso de los periféricos, como la pantalla, el teclado, entre otros, el sistema operativo le ofrece las system calls, que son funciones para realizar ciertas acciones con un periférico desde la sección de código del usuario.
Esta decisión se tomó de la siguiente manera:

- **Kernel Space:**
    - Rutina de inicialización de la *Interrupt Descriptor Table* (IDT).
    - Rutinas de atención a las excepciones e interrupciones.
    - Implementación de las system calls.
    - Implementación del driver de pantalla.
    - Implementación del driver de teclado.
    - Implementación del driver de Timer Tick.

- **User Space:**
    - Implementación de la terminal, donde el usuario ingresa los comandos.
    - Implementación de los programas indicados en la consigna.
    - Implementación de la librería de funciones útiles

Desarrollo:

KERNEL SPACE

Inicialización de la Interrupt Descriptor Table:

En esta sección se cargan las entradas de la tabla IDT mediante la función load_idt() y setup_IDT_entry(), provistas por la cátedra. Ambas utilizan una estructura que simula una entrada de la tabla. Lo único que se modificó fue la máscara de los PIC (Programmable Interrupt Controller) que permite activar o desactivar las interrupciones de Hardware.

Interrupciones y excepciones:

Como se mencionó, nuestro objetivo fue imitar el comportamiento de Linux. Para lograr esto se definieron handlers para manejar las distintas interrupciones y excepciones, las cuáles son cargadas en la Interrupt Descriptor Table (IDT).
El Kernel maneja dos excepciones que son mencionadas en el Manual de Usuario, *zero division* e *invalid opcode*. Al lanzar alguna de ellas mediante un comando en la Shell nos pareció correcto aguardar a que el usuario pueda visualizar el mensaje de error, ver los registros al momento de la excepción y que, luego de presionar la tecla *enter*, se le devuelva el control al usuario, reiniciándose la Shell.

SystemCalls:

Como se dijo previamente, el sistema operativo ofrece servicios que pueden ser utilizados por el User Space y para poder hacer uso de los mismos, se provee una colección de funciones. Estas dos últimas se comunican directamente con los drivers de los periféricos y realizan la acción pedida.
Las syscalls fueron implementadas bajo el puerto 80h de la IDT para mantener el estilo que decidió adoptar Linux. Estas reciben por registro ciertos valores para ser invocadas siguiendo la convención de registros en 64 bits para adaptarlo a nuestra librería. La interrupción int 80h hace un llamado a la función "*sysCallDispatcher*" dependiendo de qué valor recibe, llama a las siguientes funciones:

```
sys_read(rdi, (char *)rsi, rdx);
sys_write(rdi, (char *)rsi, rdx);
sys_clearWindow();
sys_restartCursor();
sys_uniqueWindow();
sys_printmem((uint64_t *) rdi);
sys_date((char *)rdi);
sys_infoReg();
sys_paint((uint8_t*) rdi, (uint32_t) rsi);
sys_miliseconds_elapsed();
sys_seconds_elapsed();
sys_set_font((int) rdi);
sys_get_font();
```

Driver de pantalla:

Se proveen funciones para escribir a la pantalla del kernel, haciendo uso del modo gráfico. Se diseñaron métodos para imprimir a pantalla con diferentes colores de relleno y fondo.

Driver de teclado:

La función del driver de teclado es guardar el carácter ASCII de todas las teclas que se presionan. Dicho esto, se implementó un driver que dependiendo el *Scan Code* de la tecla presionada, se guarda en un buffer el carácter ASCII asociado a esa tecla1.
Se agregó la funcionalidad especial de reconocimiento de la tecla SHIFT, para guardar el carácter acorde.
Asimismo, se establecieron otras teclas especiales para el funcionamiento de los programas (véase el manual de usuario para más información).

Driver de Timer Tick

A las funciones provistas por la cátedra, como el contador de tics y los segundos transcurridos, se agregaron las funcionalidades de *seconds_elapsed()* y *milliseconds_elapsed()* que cuentan la cantidad de segundo y milisegundos transcurridos respectivamente.

USER SPACE

Implementación de la terminal

La funcionalidad de la terminal consta de una lectura constante de las teclas presionadas y la impresión de los caracteres correspondientes en la pantalla, mediante las system calls correspondientes. Todo aquello que se va leyendo, se guarda en un *buffer* y cuando se detecta un *ENTER* se decodifica lo escrito por el usuario. Si lo ingresado es un comando válido (véase el manual de usuario para ver los comandos) se procede a correr el programa correspondiente.:

Implementación de los programas indicados en la consigna:

Dentro del archivo shell.c podemos encontrar las implementaciones de los comandos disponibles en la Shell, las funcionalidades de cada comando se encuentran detallas en el Manual de Usuario:

- help
- Time
- inforeg
- printmem
- play
- dividezero
- invalidop
- setsmallfont
- setnormalfont
- setbigfont

Herramientas utilizadas:

Las herramientas que se utilizaron son las siguientes:

- Visual Studio Code: IDE.
- Terminal + Docker + Make + QEMU + GCC: compilación y ejecución del kernel.
- GDB: debuggeo del kernel.

Limitaciones del Kernel:

La principal limitación del Kernel es la falta del beeper al ganar en el juego de Tron. Tras una larga investigación, se implementó una system call, sys_beep() que intenta accionar el parlante de la pc. Lamentablemente, por unas fallas en la compilación, no se pudo testear esta funcionalidad por lo que el Kernel no dispone de esta función.