Trabajo Práctico Especial

Arquitectura de las Computadoras

# Maio, Sebastián - 50386

# Costesich, Pablo Alejandro - 50109

# 2015Q1

Reporte

Introducción y Objetivos

# Objetivos

Desarrollar un Sistema Operativo de 64 bits en arquitectura Intel EM64T para la cátedra de Arquitectura de las Computadoras, que administre los recursos de Hardware de una computadora y muestre características de Modo Protegido.

# Introducción

El sistema operativo desarrollado, *the Barely Compiles Operating System* (BCOS, *because*), está basado en el Bootloader de Pure64 provisto por la cátedra. Presenta una separación en Kernelspace y Userspace mediante syscalls a la interrupción 80h.

Userspace está compuesto sólo por Shell, un programa de intérprete de comandos que interactúa con el usuario. Éste es compilado junto a una librería estándar que provee los llamados a syscalls.

Kernelspace está dividido en drivers y un módulo de control de interrupciones. Éste último es el nodo central entre todas las operaciones del SO y controla el pasaje de datos entre la frontera de Kernel/Userspace.

Syscalls

Las llamadas a sistema (syscalls) son generadas mediante la interrupción 80h y la siguiente configuración:

Caller

RAX: Indica la syscall a llamar.

RDI, RSI, RDX, RCX, R8, R9: Registros de propósito propio.

Callee

RAX: Resultado de la syscall.

Syscall Reference Table

Las syscalls respetan la convención de llamados a función de la ABI EM64T ó x86-64, System V Style. Los registros usados para llamar a la syscall deben ser considerados volátiles. Los parámetros de cada syscall corresponden en posición a RDI, RSI, RDX, RCX, R8, y R9, respectivamente.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Write | 0 | File Descriptor | Source Buffer | Size |  |  |  |
| Read | 1 | File Descriptor | Destination Buffer | Size |  |  |  |
| IO Control | 6 | File Descriptor | Request Type | \* (Depends on Request) |  |  |  |
| Pause | 34 |  |  |  |  |  |  |
| Towel | 42 |  |  |  |  |  |  |
| Halt | 48 |  |  |  |  |  |  |
| Get System Time | 200 | Time Struct Pointer |  |  |  |  |  |
| Set System Time | 201 | Time Struct Pointer |  |  |  |  |  |

# Write

Escribe Size bytes a un File Descriptor de un Buffer especificado por el Usuario.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Write | 0 | File Descriptor | Source Buffer | Size |  |  |  |

Prototipo: int syscall\_write(unsigned int fd, char \*str, unsigned int size);

# Read

Lee Size bytes de un File Descriptor a un Buffer especificado por el Usuario.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Read | 1 | File Descriptor | Destination Buffer | Size |  |  |  |

Prototipo: int syscall\_read(unsigned int fd, char \* buf, unsigned int size);

# IO Control

Controla propiedades de un File Descriptor. Para más información, referirse a la sección de Librería Estándar y LibC.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| IO Control | 6 | File Descriptor | Request Type | \* (Depends on Request) |  |  |  |

Prototipo: int syscall\_ioctl(unsigned int fd, unsigned long request, void \* params);

# Pause

Duerme la pantalla (cualquier llamado a escritura la despierta). Es equivalente a realizar un llamado de screensaver.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Pause | 34 |  |  |  |  |  |  |

Prototipo: void syscall\_pause(void);

# Halt

Detiene el sistema.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Halt | 48 |  |  |  |  |  |  |

Prototipo: void syscall\_halt(void);

# Towel

Imprime 42 en un loop infinito.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Towel | 42 |  |  |  |  |  |  |

Prototipo: void towel(void);

# Get System Time

Almacena la hora del sistema a una estructura de destino. Más información disponible en Librería Estándar y LibC.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Get System Time | 200 | Time Struct Pointer |  |  |  |  |  |

Prototipo: void gettime((struct rtc\_time \*) time\_struct);

# Set System Time

Actualiza la hora del sistema desde una estructura. Más información disponible en Librería Estándar y LibC.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYSCALL | RAX | RDI | RSI | RDX | RCX | R8 | R9 |
| Set System Time | 201 | Time Struct Pointer |  |  |  |  |  |

Prototipo: void settime((struct rtc\_time \*) time\_struct);

Drivers

Los drivers de sistema definen acceso a Hardware o abstracciones al mismo. Éstos incluyen el RTC, Video y Teclado.

# Video

El driver de video expone funcionalidad muy básica para el manejo de un buffer (“flip” buffer) y un cursor en la pantalla. Apunta a emular las funcionalidades básicas de una terminal en cuanto a que soporta scroll, saltos de línea y tabs.