Manual de Usuario - Trabajo Práctico Especial (TPE)

Instituto Tecnológico de Buenos Aires - Arquitectura de las computadoras (72.08)

Grupo 21

4 de junio de 2024

Ejecución

Una vez compilado el proyecto y generada la imagen, se puede ejecutar el mismo mediante el siguiente comando.

```
$> ./run.sh
```

Notar que el proyecto utiliza flags de qemu para poder emitir sonido. Los flags necesarios para esto pueden cambiar dependiendo del entorno.

Shell

El shell cuenta con varios módulos que se pueden ejecutar. Al iniciarse el shell por primera vez se ejecuta el módulo help. El módulo help lista todos los módulos disponibles, junto con una descripción de su funciónalidad.

Lista de comandos:

- exit: vuelve al kernel.
- help: muestra los módulos disponibles.
- clear: limpia el buffer de texto de la pantalla.
- sysinfo: muestra información del sistema.
- fontsize: permite cambiar el tamaño de letra.
- time: imprime la hora actual.
- regs: muestra los registros capturados.
- beep: emite un sonido.
- song: toca una canción mientras se muestran gráficos por la pantalla.
- calculator: calculadora de enteros positivos (al dividir por 0 se produce excepción).
- eliminator: juego eliminator.
- invalid_opcode: mediante un jump a la posición 0 en memoria se produce una excepción de tipo invalid opcode.

Los registros se pueden capturar en cualquier momento presionando la tecla escape.

Desarrollador

El shell permite facilmente agregar nuevos módulos. Para crear un modulo es necesario agregar una entrada al array de módulos que contiene structs de ModuleDescriptor.

```
typedef struct {
    char* module_name;
    char* module_description;
    void (*module)();
} ModuleDescriptor;
```

El module_name es el que será utilizado como nombre del comando en la shell. El module_description es usado por el comando help. Y el tercer elemento del struct es un puntero a la función del módulo.

Ya existen implementadas las siguientes funciones en la librería estandar: putchar, puts, printf, strlen, strcmp, strcpy, atoi, itoa, getchar, scanf. Y existen adaptadores para los syscalls.

Los syscalls disponibles se listan en la tabla 1. Están modelados a partir de la API de Linux 64bit.

rax	syscall	rdi	rsi	rdx	r10	r8
0	sys_read	const char* buff	uint64_t len			
1	sys_write	uint64_t fd	const char* buff	uint64_t len		
2	sys_put_text	const char* str	uint32_t len	uint32_t hexColor	uint32_t posX	uint32_t posY
3	sys_set_font_size	uint32_t font_size				
4	sys_draw_square	uint32_t hexColor	uint32_t posX	uint32_t posY	uint32_t size	
5	sys_get_screen_width					
6	sys_get_screen_height					
7	sys_get_time	int time_zone				
8	sys_get_key_pressed					
9	sys_get_character_pressed					
10	sys_clear_text_buffer					
11	sys_get_cpu_vendor	char* buff				
12	sys_beep	uint64_t freq	uint64_t secs			
13	sys_delay	uint64_t millis				
14	sys_clear_screen	uint32_t clearColor				
14	sys_print_registers					

Tabla 1: syscalls implementadas para el llamado de funciones de kernel desde el userspace