TPE - Arquitectura de Computadoras

Agustin Alonso; 63316
Magdalena Cullen; 63065
Tomas Becerra; 63732

Manual de Usuario

A la hora de iniciar el sistema operativo, se inicia un prompt del shell en el cual el usuario puede interactuar con el sistema mediante los respectivos comandos. Luego de la ejecución de cualquier funcionalidad, la shell se prepara para recibir otro input. A continuación, un breve listado de todas las funciones implementadas.

man:

Imprime en pantalla un breve manual con descripciones de todas las distintas funcionalidades disponibles para su ejecución.

time:

Imprime en pantalla la hora local en formato “hh:mm:ss”.

div0:

Fuerza que el sistema genere una excepción de división por cero.

invalid op:

Fuerza que el sistema genere una excepción de opcode invalido.

registers:

Imprime en pantalla los valores de todos los registros guardados previamente. Para guardar los registros, basta con presionar en el prompt de comandos “SHIFT + ALT”. En caso de que no se guardó ninguna copia de los registros, la función no procede con la impresión de los mismos.

snake:

Función para jugar al snake, en sus variantes para uno y dos jugadores.

clear:

Al ser llamada, “clear” borra todo el contenido de la shell y vuelve a su estado original.

zoom in:

Funcionalidad que incrementa el tamaño de los caracteres ingresados en pantalla, al ser ejecutada reinicia el prompt de comandos.

zoom out:

Esta función decrementa el tamaño de los caracteres impresos en pantalla, al igual que “zoom in” su ejecución reinicia la shell.

set theme:

Funcionalidad para cambiar la combinación de colores de la shell. Su ejecución reinicia la terminal.

Justificación de Diseño

A continuación, vamos a exponer los distintos motivos y explicaciones para nuestra implementación de las distintas funcionalidades del trabajo práctico especial.

Interrupciones:

Para implementar el funcionamiento de las interrupciones, previo al salto a Userland, el programa inicializa las entradas de la IDT con las funciones a ejecutar en caso de una interrupción. Por un lado, seteamos la interrupción de teclado nos permite interactuar con el teclado para el input de teclas, a su vez con el Timer-tick, para implementar el funcionamiento del "sleep" del programa y la pseudo-randomización de números. En cuanto a la propia implementación de la IDT seguimos la implementación provista por la cátedra.

Driver de Teclado:

En primer lugar, para el funcionamiento del driver del teclado, debemos configurar la entrada correspondiente a las interrupciones de hardware relacionadas al manejo del teclado. Nuestro sistema, a la hora de recibir un input por parte del teclado, identifica que se está recibiendo y procede principalmente de dos maneras. En caso que el carácter sea una tecla especial, tanto como el "CTRL" o el "SHIFT", se guarda un flag que indica su estado, diferenciado sí presionado o ignorado. Para el resto de los caracteres, se los guarda en un buffer en donde posteriormente se puede leer para obtener los inputs recibidos. A su vez, implementamos la función del guardado de registros mediante la combinación de teclas "SHIFT + ALT".

Driver de Video:

El driver de video es el encargado de modificar los valores de los píxeles del modo video que dispone Pure64. Implementamos diversas funcionalidades tales como poder dibujar caracteres, formas geométricas y en caso de si se precisa la totalidad de la pantalla. A su vez, diseñamos un sistema de "cursor", para facilitar la escritura de textos, manteniendo la posición en donde se requiere escribir posteriormente nuevos caracteres.

El cursor, utiliza la lógica de desplazarse en la pantalla utilizando coordenadas x e y para seguir la funcionalidad de una terminal, de escribir a lo largo de una línea. Decidimos implementarlo dentro del *videoDriver.c,* ya que las syscalls de dibujar caracteres y formas pertenecen a la parte de video del sistema operativo.

System Calls:

Para hacer uso de las distintas funcionalidades del Kernel desde otro módulo, pero sin tener acceso a este mismo, se nos hizo indispensable crear un sistema de llamadas al kernel,

por medio de interrupciones. Esto permite proteger a nuestro sistema operativo de posibles fallas producidas por el usuario.

Estas system calls (llamadas al sistema operativo), nos permiten interactuar con el driver de video y el driver de teclado, manejar excepciones, manipulación tanto tiempo y chequeo de los registros del procesador.

El manejo de las system calls se realiza a través de una la interrupción 0x80, la cual recibe en el registro RDI el ID de la system call que quiera utilizar y en los demás registros, los parámetros que precise dicha función.

Las system calls proporcionadas en este trabajo fueron pensadas para la implementación de las diferentes funcionalidades de nuestro sistema operativo.

Excepciones:

Para el manejo de excepciones, se configuró las dos entradas solicitadas en las primeras posiciones de la IDT. Nuestra implementación del manejo de las mismas, consiste en definir en Userland la función que se debe ejecutar ante una interrupción, y luego asignarle dicha función a una excepción en específico.

En particular, tanto como para la excepción de división por cero, como la de opcode invalido, definimos un handler genérico que actúa de manera similar en ambos casos. Para la recuperación de la excepción, debemos realizar un salto a una posición segura, restableciendo los registros de manera que el programa pueda seguir ejecutando correctamente. Es por esto, que implementamos las funciones “saveReference” y “jumpToReference”.

La función “save Reference” guarda un estado de programa al cual se deseara volver luego, mantenemos los registros que no se modifican en el pasaje de argumentos con el mismo valor y sobre todo el RIP para luego poder efectuar el salto. Por otro lado, “jumpToReference” recupera el estado deseado y salta a la dirección del RIP previamente guardado, de esta manera se recuperan los registros luego de la excepción y se vuelve a ejecutar el shell.

Time:

El manejo del tiempo se realizó de forma tal que, llamando al correspondiente módulo “time”, se imprima por pantalla horas, minutos y segundos actuales en Argentina. Para esto se utilizó una system call la cual hace uso de RTC (real time clock), para obtener el tiempo. La hora que muestra el RTC se basa en la hora y la fecha establecidas durante la configuración inicial de la computadora, es por esto que tuvimos que ajustar la hora a la de Argentina.

Shell:

La shell es una pieza fundamental de cualquier sistema operativo, proporcionando una interfaz de usuario poderosa y versátil para interactuar con las funcionalidades del sistema operativo. En nuestro programa dicha interfaz de usuario permite interpretar diferentes tipos de comandos (explicados en el manual de usuario).

Dado que los programas a ejecutar se podrían considerar de baja complejidad, decidimos no aceptar argumentos por líneas de comando, en nuestros scripts, y así facilitar el manejo de errores, en los posibles scripts proporcionados por el usuario.

Snake:

El juego snake, se implementó desde el lado de usuario (userland), ya que consiste en crear un juego interactivo con el usuario. En esta implementación, se separa en tres partes principales: la parte de consola de juego, el manejo del snake, y la creación de la comida.

Para la parte de manejo de consola de juego, se creó una matriz con dimensiones modificables en los defines, donde toda la lógica parte de los cuadrados del tablero. Dado que es un juego, decidimos dejar la primera fila de la matriz como el menú para el juego, donde se ve la actualización de puntos y mensaje de salida. La tecla ESC se implementó como la salida, ya que nos pareció práctica, ya que su función es escapar. También implementamos las funcionalidades de perder el juego, salir del juego, y seguir jugando otra ronda. Se guarda el valor mayor de puntajes obtenidos en la transcurso del ejecución del programa y una vez vuelta al shell, se reinicia este seguimiento.

En el TPE, se pidió brindar la funcionalidad de doble jugador. Decidimos crear un *struct* que contiene la información necesaria para cada snake, como su carácter identificador, tamaño, head, un vector con la posiciones del cuerpo, último movimiento, color, y puntaje. Esto nos pareció la forma ordenada para poder acceder a todos estos comportamientos que componen a la serpiente. A medida que la serpiente se mueve en la pantalla, se realizan comprobaciones a través de la matriz *boardStatus* para verificar si se encuentra vacía o no, tanto como las colisiones con los límites del tablero. Estas colisiones se acompañan con la implementación del sonido para alertar al jugador, ya sea cuando la serpiente crece (es decir, cuando come) o cuando colisiona consigo misma, con otra serpiente o con el límite del tablero.

Por último, utilizamos una función inspirada en *rand()* de stdlib para generar posiciones aleatorias para la comida de la serpiente y su color. Esta función consiste en ajustar los límites del rango de números a elegir, es decir, en nuestro caso, las dimensiones del tablero.