ARQUITECTURA DE LAS COMPUTADORAS

TRABAJO PARACTICO ESPECIAL

1ER CUATRIMESTRE 2015

INFORME

INTEGRANTES:

MAGDALENA VEGA	55206
NICOLAS KUYUMCIYAN	55165
LUCAS CASAGRANDE	55302

INTRODUCCIÓN:

El trabajo práctico especial consistió de la realización de un Sistema Operativo para computadoras con arquitectura de 64 bits. Se utilizó el Pure64 para configurar el booteo de la máquina y configurarla en modo Flat.

DESARROLLO:

DIVISIONES ENTRE EL KERNEL-LAND Y USER-LAND

Para evitar que el usuario tenga acceso directo a las funciones que los corresponden al Kernel, o peor acceso directo al Hardware, se divide el espacio en dos. Uno donde están todos los drivers de HW y la lógica del Kernel, y la otra donde se encuentra todo a lo que el usuario puede acceder.

Para que el UserLand pueda interactuar con el Kernel y así tener acceso al HW se implementó la interrupción 80h. Cuando se llama a esta interrupción (desde el UserLand) se le pasan los parámetros dependiendo de que se desee hacer. Luego el Kernel recibe la orden y actúa en respuesta. De esta forma si el Kernel decide que el usuario no debería poder acceder simplemente le niega el acceso.

Además, con esta implementación, se cuenta con la ventaja de que para el usuario (y el código que ejecuta) la lógica interna del Kernel no le afecta, y si el día de mañana de decide hacer un cambio, siempre y cuando se mantengan las interrupciones todo va a seguir funcionando.

Para facilitar la llamada a la INT 80h hicimos una función en assembler que (aprovechando de que lo mismo que recibe tiene que pasarlo, y lo mismo que devuelve es lo que te tiene que devolver...) lo único que hace es llamar a la interrupción:

Nosotros definimos los siguientes comandos de la INT 80h

1. Escribe en pantalla el carácter que se le pasa con su perspectivo modificador(color)
2. Borra la pantalla y hace apunta el puntero a la primera posición
3. Lee una cadena de caracteres del buffer del teclado
4. Lee un solo carácter del buffer del teclado
5. Permite leer el Real time Clock
6. Escribe el Real Time Clock
7. Modifica los colores con los que aparecen los caracteres en pantalla
8. Cambia el tiempo hasta que aparezca el screensaver.

La función que recibe las interrupciones no sabe que puede llegar a recibir (debido a que cada interrupción tiene diferentes parámetros), por lo tanto los 2 argumentos que recibe son del tipo de dato más grande (***uint64_t***) y en caso de necesitar otro, se castea.

VIDEO

El driver de video es el único que tiene acceso a la posición de memoria donde se guarda los caracteres que se muestran en pantalla. Y se encarga de:

- Escribir los caracteres que le pasen.
- Borrar toda la pantalla cuando se lo soliciten
- Manejar el scroll, cuando se llega al final de la pantalla se mueve todo para arriba dejando una nueva línea vacía
- Mostrar con una animación donde se encuentra el cursor en todo momento
- Manejar los colores de la consola
- Guardar la pantalla antes del screensaver y volverla a restaurar cuando se presiona una tecla.

Se podría discutir si es correcto que el driver de video se encargue de manejar la animación de standby, considerando que es algo que afecta a la consola, pero decidimos que si le delegamos la responsabilidad a la consola, le estábamos dando demasiados accesos a cosas que la consola no tendría que controlar (debido a que cuando se imprime el standby se tiene que guardar el modificador de color que había anteriormente para restaurarlo) y también la consola tendría que estar al tanto cuando alguien hace un Enter o borra un carácter porque también ahí hay que restaurar el modificador anterior.

TECLADO

Para manejar el teclado se implementó la interrupción 21h que se activa cada vez que alguien presiona una tecla. La interrupción llama a la función encargada de manejar el teclado pasándole como parámetro la tecla presionada ya convertida a un Char. Esto se consigue teniendo un Array con todos los valores y sus respectivas conversiones a caracteres, y donde también se evalúa si se está presionando Shift o está activado el Caps-Lock para ver correctamente que carácter hay que imprimir.

Una vez que el buffer recibe el carácter ya convertido lo evalúa, si lo que recibió es un 0 lo ignora (es una tecla que no está implementada o que no muestra nada) y lo que recibió es un símbolo que representa que se presionó borrar entonces borra un carácter. Si lo que recibió es un carácter que se muestra en pantalla lo guarda en el buffer (siempre y cuando tenga capacidad). Y si lo que recibió es un Enter entonces copia todo lo que tenga en el buffer a un segundo buffer, llamado clean_buffer, donde queda guardado para el usuario lo pida.

Este sistema de doble buffer permite que el usuario solo tenga acceso a lo que se escribió una vez que se presiona Enter, y que además lo que reciba sea directamente la cadena que se escribió y no tenga que estar haciendo ningún manejo raro para conseguir lo que le importa, que es lo que se escribió.

LIBRERÍA C

Debido a que no era necesario implementar todas las funciones de C se hicieron solo las siguientes:

- `memset` y `memcopy` hechas por la catedra
- `print_message` – Imprimir un String y definirle el modificador
- `print_number` – Imprimir un numero
- `str_cmp` – Comparar String (para ver que comando se escribió)

- `atoi` – Convertir un String a un numero
- `strlen` – Tamaño del String
- `isNumber` – Ver si el String es un numero
- `pow` – Calcular la potencia
- `putChar` – Escribir un solo caracter
- `printf` – imprimir String concatenados con números u otros String

CONSOLA (SHELL)

La consola es lo que se ejecuta luego de que se inicialice el Kernel, y es lo que siempre va a estar corriendo hasta que se apague la computadora. Es lo que se utiliza para poder llamar a todas las demás acciones que puede hacer el usuario. La consola está constantemente buscando leer lo que escribió el usuario para ver si se corresponde con alguno de los comandos válidos, y si este es el caso llamar a la función que ejecuta el comando.

Para esto creamos una estructura que tiene un String de nombre uno de descripción y un puntero a la función que se tiene que ejecutar.

Los comandos validos son:

- ✓ clear – Borra toda la pantalla
- ✓ time – Muestra la hora y la fecha
- ✓ help – Muestra todos los comando disponibles
- ✓ change time – te permite cambiar el tiempo y la fecha
- ✓ whoami – te dice quien sos
- ✓ keyboard – muestra gráficamente la distribución del teclado
- ✓ colors – Permite cambiar los colores de la consola
- ✓ screen time – Permite cambiar el tiempo de inactividad hasta que aparezca el salva pantallas

Vale aclarar que una vez que se ejecutan estos comandos la consola no vuelve a tener el control hasta que se terminen de ejecutar, independientemente de que se siga viendo lo que se escribió antes.

INTERRUPCIONES

Nuestro TPE tiene tres interrupciones implementadas:

- ❖ Interrupcion del TimerTick, esta interrupción se ejecuta cada 55 ms. En nuestro caso esta interrupción se encarga de 2 cosas:
  - ➢ Contar el tiempo transcurrido, y en caso de que no se registre actividad en el tiempo determinado por el usuario primero llama a la función que se encarga de guardar la pantalla como estaba antes de mostrar el screensaver, y cada 550 ms llama a la función del screensaver para que muestre otra de las “imágenes”

- Tiene un contador que se encarga de manejar el cursor que parpadea para que el usuario sepa dónde está escribiendo. Cada 550 ms se encarga de llamar a una función que varía el estado de dicho cursor.

- Interrupción del Teclado: esta interrupción se ejecuta cada vez que el usuario presiona una tecla. Esta interrupción se encarga de 3 cosas:
  - Si en ese momento estaba activado se encarga de llamar a la función que restaura el estado de la pantalla a como estaba antes, si el screensaver no está activado se encarga de resetear el Timer que calcula el tiempo de inactividad
  - Pasarle al keyboard buffer la tecla que se presionó ya convertida al carácter
  - Llamar a la función que se encarga de escribir en pantalla y pasarle el carácter que se escribió(también ya convertido)

Nosotros decidimos independizar lo que es la escritura en el buffer con lo que es la escritura en pantalla, entonces si se quisiera podría hacerse que se guarden las teclas en el buffer y no se muestren, o la inversa, no guardando las teclas en el buffer pero si en la pantalla.

- Interrupción de Software 80h: esta interrupción es llamada por el usuario cuando quiere acceder al HW o a alguna función del Kernel. Cuando se llama a la interrupción se ejecuta la función sys_manager que se encarga de llamar a la función pertinente. Las funciones que se pueden llamar son:
  - Una función que se encarga de escribir en pantalla lo que le pasen, esta función recibe un char y el modificador de color con el que se lo quiere imprimir. Dependiendo del carácter que le pasen hace diferentes cosas
    - Si le pasan un 0 no hace nada, ni siquiera mueve el puntero a la próxima posición
    - Si le pasan un backspace llama una función que se encarga de borrar el ultimo carácter, siempre y cuando pueda(en caso de que este al principio de la línea no borra)
    - Si le pasan un carácter común lo muestra en pantalla, si le pasan un modificador de vale 0xFF (decisión nuestra) usa el modificador por default de la consola, (que depende de la gama de colores que eligió el usuario) y si le pasa otro modificador se utiliza ese.
    - Si le pasan un Enter entonces llama a una función que se encarga de dibujar una nueva línea

    En todos los casos se fija si llego al final de la pantalla y en caso afirmativo mueve todo una línea para arriba.

  - Una función que se encarga de borrar todo lo que hay en pantalla y hacer que el cursor apunte al comienzo de la pantalla
  - Unas funciones que te permiten leer un carácter o una cantidad finita de caracteres juntos (que se guardan en un Char* pasado por el usuario). Ambas funciones en caso de encontrarse el buffer vacío devuelven 0, debido a que si se quedan esperando nunca saldrían de la interrupción, lo cual consideramos que no es correcto, entonces hay una función getChar en el UserLand que se queda esperando si le devuelven 0 y la sigue llamando hasta que devuelva algo diferente. Cabe aclarar que en el caso de que se pida una cantidad mayor a un carácter no hay problema de que el 0 afecte el funcionamiento, ya que como esta implementado el buffer del teclado cuando se llena, se hace con toda la palabra a la vez, y si se piden más caracteres que los que están en el buffer se devuelve el String hasta el '\n'
  - Dos funciones que te permiten leer y cambiar la hora del sistema, los valores que se le pasan están en decimales, y después la función de encarga en convertirlos al formato que usa el sistema para guardarlos. Evitando molestar al usuario con la conversión

- Una función que te permite cambiar el color de defecto que se usa en la consola. Cuando se llama a esta función no solo se cambian los colores desde ese momento, sino que también se recorre todo lo que ya está escrito convirtiéndolo al color que se eligió, para mantener la armonía visual
- Una última función que modifica el tiempo hasta que se muestre el screensaver, esta función recibe el valor en segundos y se encarga de convertirlo a Ticks del timer.

CONSIDERACIONES:

- No vimos necesario hacer una interrupción donde te permita escribir un Char*, ya que todo se puede hacer fácilmente poniéndolo de a 1.
- Para evitar que el usuario pueda borrar más allá de la línea donde está escribiendo implementamos un puntero que apunta al principio de la línea donde tiene que escribir, y si la distancia donde esta ahora es menor que 4(porque en cada línea escribe ">:" al principio de la misma) no se le permite borrar.
- Decimamos que si quieres cambiar el reloj tienes que cambiar todos los valores juntos.
- No se puede llamar a un comando pasándole los argumentos escribiéndoles después los valores, ya que en nuestra estructura de comandos la función es de la forma `void función()`. Esto también nos permite escribir comandos que tengan más de una palabra separadas por un espacio.
- Debido a la cantidad de String que teníamos nos encontramos que nos quedamos sin espacio, debido a esto modificamos el archivo linkeditor del UserLand para que en vez que dividir cada 4 KB lo haga cada 8 KB.
- Más allá de que el buffer de teclado tiene espacio para 255 teclas, solo se pueden poner 253, debido a que si se dejarían poner 255 no habría espacio para el Enter y además al ser circular cuando se ocupa la última posición el puntero que determina donde termina la cadena queda apuntando al mismo lugar que el que indica el comienzo.
- También por una cuestión de claridad para el usuario se decidió que si el buffer del teclado se llena que se deje de escribir en pantalla para avisar que no hay más espacio en el buffer

BUGS

- Si se presiona Enter y rápidamente otra tecla esta se escribe en la pantalla antes de que el comando invalido se muestre, mas allá de que si se presiona borrar se puede sacar, genera que parezca que se introdujo correctamente un comando cuando, en realidad, quedo la letra en el buffer.