Juan Li Puma, 55824
Diego de Grimaudet de Rochebouët, 55821
Arquitectura de las Computadoras (72.08), Comisión S
Segundo Cuatrimestre 2015

Informe TP Especial Arquitecturas de las Computadoras

El objetivo de este trabajo especial fue implementar un kernel que administre los recursos de hardware de una computadora y muestre características del modo protegido de Intel. Para poder mostrar estos conocimientos, se pidió crear un kernel que además de hacer lo antes mencionado también tenga una terminal interactiva con distintos comandos, entre ellos uno para reproducir sonidos por el speaker interno de la computadora (PC speaker).

Se tuvo entonces que crear un espacio de usuario (User Space) y un espacio de kernel (Kernel Space) separados; el kernel space encargado de administrar el hardware y el user space de todo lo demás (interpretar los comandos del usuario, correr distintos programas, etc.). La única forma de acceder al kernel space desde el user space se diseñó a ser una interrupción, en particular la 0x80. Mediante esta interrupción se accede a distintas funciones del kernel ("syscalls") de forma organizada y regulada. De estas syscalls se crearon simplemente las que fueron necesarias, que administran entrada y salida de datos de los dispositivos de hardware como el PC speaker, el teclado y el video. Esta separación es necesaria para que no cualquier aplicación tenga libre albedrío con el hardware, lo que sería desorganizado e incluso peligroso. Esto tiene la ventaja de proteger el hardware contra aplicaciones maliciosas o simplemente negligentes que podrían querer causar algún daño al hardware, pero por otro lado podría también plantearse la desventaja que limita lo que puede hacer el usuario.

Un ejemplo de la separación de user space y kernel space es el conjunto de syscalls "sysread" y "syswrite" (llamadas desde las funciones de user space "fread" y "fwrite" entre otras) que reciben por parámetro un descriptor de archivo (entrada estándar, salida estándar,

speaker, etc.) y un buffer de donde leer o adonde escribir. Estas funciones necesariamente se acceden mediante int80 con los parámetros deseados y una función del kernel space se encarga de validar el pedido de lectura o escritura, y únicamente si es válido lo realiza. Esta utilización de *fread* y *fwrite* abstrae al usuario del hardware (facilitando significativamente su trabajo como programador) y le da acceso genérico a distintos dispositivos. En caso de añadir un dispositivo nuevo basta con agregar el descriptor de archivo en el header apropiado y la función de atención del lado del kernel, y el usuario ya tiene la funcionalidad nueva sin hacer nada de su lado. En el unico hardware en donde no se cumplio esto fue en sonido, puesto que generalmente se quería escribir notas de una a la vez y entonces no era necesario pasar por un buffer, nos pareció entonces más claro pasar entonces por una "syscall" distinta.

Video

Gran parte de la funcionalidad de video fue provista por la cátedra en el commit original. Si bien se agregaron algunas cosas, la tarea principal era regularizar las funciones de video para el usuario de la misma manera que se hizo con los demás dispositivos. Se definió la salida estándar (STDOUT) como el video, y la salida de errores (STDERR) como el video pero con otros colores. La función más importante de video es "ncPrintChar" que imprime un carácter especificado y avanza el puntero del video para imprimir en el siguiente lugar. En particular se definieron algunos caracteres especiales como \b y \n que manipulan el video para obtener el comportamiento esperado al introducir backspace y newline. Todas las demás funciones de video se basan en *ncPrintChar*, por lo que todas las syscalls que corresponden al video terminan siendo

una serie de llamadas a ésta. Todo esto es transparente para el usuario; él sólo conoce las funciones *putchar* y *printf*.

Teclado

Para el teclado se diseñó un buffer interno circular en el cual se van encolando los scan codes de las teclas presionadas (o soltadas) a medida que se reciben por interrupciones de teclado, y se desencolan mediante se solicitan teclas por syscalls. El buffer es único y las teclas que se desencolan se envían al programa que las solicite. Si hubiesen varios programas corriendo "al mismo tiempo" que necesiten entrada del teclado esto sería un problema, pero en nuestro caso sólo hay un programa que lee del teclado a la vez: la terminal o el programa de piano (llamado desde la terminal; cuando corre el piano la terminal está esperando que retorne el piano y por lo tanto no lee del teclado). Hay dos descriptores de archivo que corresponden al teclado: STDIN, que convierte los scan codes a caracteres ASCII conocidos o devuelve 0 para teclas sin representación ASCII; y STDIN_RAW, que devuelve directamente los scan codes. El primer formato es utilizado mayormente por la terminal, mientras el segundo lo utiliza el piano.

Terminal

La terminal funciona casi únicamente con syscalls. La lectura del teclado y la salida a pantalla son funciones básicas de la terminal que se acceden mediante syscalls. Los programas que ejecuta la terminal también funcionan con syscalls. Se crearon algunas syscalls en especial

para la terminal, como “scroll” para dar el efecto de que el texto se traslada hacia arriba cuando la pantalla está llena y “reboot” para reiniciar el sistema. La terminal utiliza un buffer de entrada propio (independiente del buffer de teclado pero alimentado de éste) en el cual va guardando la entrada del usuario para luego interpretarla.

Sonido

Para el sonido optamos por tener dos buffers internos en los cuales se guardan las frecuencias de los sonidos a reproducir y las duraciones de cada sonido. En estos buffers, cuyo comportamiento se basa en el del buffer del teclado, se van encolando los sonidos que se quieren reproducir que después se irán “consumiendo.” Para registrar el paso del tiempo se utilizó el timer tick: En cada interrupción del timer tick se verifica si hay sonido a reproducir, y si lo hay se lo reproduce y se decrementa el tiempo restante. Se repite esto hasta que el tiempo llega a 0 y los buffers avanzan al siguiente sonido, si hay alguno. No se guardan tiempos futuros para saber cuándo terminar, sino que cada sonido corresponde a una cantidad especificada de ticks. Se eligio poder encolar sonido para que si fuese necesario, se pueda seguir usando la terminal mientras corre un sonido de fondo que se eligió reproducir. Una desventaja que esto crea es que al basarse en el Timer Tick para medir el tiempo, el tiempo mínimo para consumir un sonido es un “Tick” del Timer Tick, que en caso de encolar muchos sonidos que uno quiere que se reproduzcan rápido puede crear un efecto de “retraso”. //sonar notas incluso después.

Las implementaciones de sonido entonces que se hicieron fueron primero un piano, que mapea las teclas del teclado (en particular sus scan codes, no su representación ASCII) con

distintas notas musicales, y un reproductor de melodías que tienen que ser previamente cargadas al módulo de datos con un formato definido por el grupo.

Piano

Para el piano lo que se hizo fue tener escritas en memoria en una matriz las diferentes frecuencias para las distintas notas musicales de distintas octavas. De entre ellas simplemente se utilizaron las octavas que se consideraron mejor audibles a partir del speaker, que fueron la 4, 5 y 6 (aunque se podría también agregar más como la 7 y 8, no pareció necesario el pequeño piano diseñado). Se dispuso entonces que 12 teclas conjuntas de 3 líneas del teclado hagan sonar las 12 notas de cada octava (notas sostenidas incluidas) y se las dispuso consecutivas para una comodidad de la programación, aunque se podrían haber dispuesto de otra forma para respetar la distribución de teclas de un piano "real." Como cada tecla encola un sonido en el buffer con duración "1," en caso de apretar rápidamente múltiples teclas, los sonidos se encolan con duración mayor a lo que duró el presionas de cada tecla y por lo tanto suenan con un retraso. Se planteó entonces que las teclas suenen hasta que llegue el código de que la tecla se había soltado, pero no se consideró necesario. Agregar esta funcionalidad no sería tarea difícil ya que el teclado registra estos eventos, bastaría filtrarlos.

Reproductor de Melodías

A pesar de que el piano solo puede hacer reproducir sonidos de notas musicales, que son ciertas frecuencias armónicas y que son las frecuencias mejor audibles al ser humano, se permitió al reproductor de melodías reproducir frecuencias mucho menores que por limitaciones

de hardware podrían sonar mal o ni siquiera sonar. Se prefirió dejarle la responsabilidad del usuario que crea las melodías de hacer con ellas lo que él prefiere, puesto que esto le permite una mayor versatilidad y comodidad de uso al usuario. Se utilizó entonces el siguiente formato para recibir en el archivo: Primero un número de 32 bits indicando la cantidad de notas de la canción. Este número está seguido de n notas, representadas con un número de 32 bits que indica la frecuencia de la nota (en caso de ser 0, se toma como un silencio) y otro de 8 bits que indica su duración. Para crear los archivos se optó por hacer un "SongMaker" en el cual se escriben las frecuencias de las notas a reproducir y sus tiempos, y éste crea el binario correspondiente. Fue un forma simple de crear el binario a partir del cual se reproducen la melodías para facilitar ese trabajo.

El conjunto de todas las utilidades descritas conforma el trabajo presentado. Hay algunas componentes que funcionan exclusivamente del lado del kernel, otras exclusivamente del lado del usuario, y otras que trabajan con el kernel mediante syscalls. Si bien el sistema es sistema intuitivo de utilizar, el manual de usuario adjunto (y el programa de ayuda del sistema) resolverá cualquier duda de uso de parte del usuario.