Sistemas Operativos

Primer cuatrimestre 2016

TP Nº 2: Construcción del Núcleo de un Sistema Operativo

Profesor: Nicolas Rosner
Adjuntos: Ariel Godio, Rodrigo Rearden
Fecha de entrega: 16 de Mayo

Introducción

Durante el transcurso de la materia aprendieron a usar la API de sistemas operativos de tipo UNIX, ahora en cambio armarán su propio kernel simple en base al trabajo práctico final de la materia anterior, Arquitectura de Computadoras. Para ello implementarán mecanismos de IPC, Memory Management, Scheduling y Memory Protection.

Grupos

Se formarán grupos de hasta tres personas. En caso de quedar grupos de uno o dos integrantes, se resolverá durante la clase de práctica caso por caso.

Requisitos Previos

Es necesario contar con una versión funcional del trabajo práctico de Arquitectura de Computadoras (un kernel monolítico de 64 bits, manejo de interrupciones básico, con system calls, driver de teclado, driver de video en modo texto y binarios de Kernel Space y User Space separados). Si el sistema tenía problemas de memoria, como por ejemplo strings corruptos, es necesario arreglarlos antes de comenzar el trabajo.

En este paso se les recomienda la generación de un cross compiler de 64 bits, pero pueden utilizar el mismo toolchain (pipeline de desarrollo) que usaron en Arquitectura de Computadoras. Pueden encontrar más información al respecto acá: http://wiki.osdev.org/GCC_Cross-Compiler.

Requerimientos

El kernel debe implementar en las siguientes features. Se les recomienda implementarlas en el orden en el que son enumeradas.

System calls

Las system calls son la interface entre user y kernel space. El sistema deberá proveer system calls para que los procesos (explicados más adelante) interactúen con el kernel. Utilice exactamente el mismo mecanismo desarrollado en Arquitectura de Computadoras (interrupciones de software).

Physical Memory Management

El kernel debe contar con algún sistema simple para reservar y liberar páginas de al menos 4 KiB contiguos (page allocator), note que esto no está atado a memoria virtual/paginación, el requerimiento es solamente reservar y liberar bloques de memoria física. Quien utiliza esta memoria puede ser el kernel para sus estructuras internas, o un proceso en user space.

Syscalls involucradas

- Reservar memoria para el proceso que llama
- Liberar memoria del proceso que llama

Procesos, Context Switching y Scheduling

El sistema debe contar con multitasking pre-emptivo en una cantidad variable de procesos. Para ello el sistema deberá implementar algún mecanismo que permita suspender la ejecución de un proceso y continuar la ejecución de otro (context switching), que sea externo a los procesos en sí, es decir, sin que los mismos se enteren (pre-emptivamente) y con alguna estructura/algoritmo que permita seleccionar al siguiente proceso (scheduler).

Cada proceso en el scheduler tiene que tener al menos tres estados: ejecutando, dormido, pausado. Solo habrá un proceso ejecutando, los procesos dormidos son ignorados por el scheduler a la hora de seleccionar el próximo proceso a ejecutar, los demás se encontrarán pausados esperando a tener tiempo de procesador. Cada proceso tiene, por supuesto, su propio stack de tamaño fijo, alocado dinámicamente haciendo uso de las características del Kernel.

Syscalls involucradas

- Crear un proceso
- Finalizar un proceso
- Listar procesos

Opcionales (pueden servirles para implementar IPCs más adelante)

- En este paso posiblemente tengan que reescribir parte de la system call read (keyboard input) para que mande el proceso a dormir hasta que reciba el foreground y el usuario introduzca un enter, “\n”, en el buffer de teclado.
- Dormir el proceso actual hasta que suceda un evento x.
- Enviar una señal a un proceso x (lo despierta si está dormido)

- Ceder el procesador al siguiente proceso (yield)

Memory Protection

Utilizando el page allocator desarrollado y el sistema de de paginación de 64 bits de Intel, deberán implementar “identity mapping”, es decir, mapear las direcciones físicas alocadas a las mismas direcciones virtuales. No es requerida la protección de memoria entre procesos, todos los procesos son en realidad threads.

IPCs

Se debe implementar al menos un mecanismo de IPC, a elección, con las syscalls correspondientes. El mecanismo elegido debe contar con las primitivas para poder sincronizar y comunicar los procesos. El mecanismo **no** puede ser shared memory, ya que los procesos corren todos en el mismo espacio de memoria.

Syscalls involucradas

Dependen del IPC elegido, considerar que algunas de estas system calls posiblemente deberán dormir el proceso que las llama, y el kernel deberá despertarlo cuando se cumpla una cierta condición.

Drivers

Se requiere la implementación de cuatro drivers, teclado, sonido, video-texto, y video-imagen.

Sonido

Use el driver de PC Speaker implementado en el TP anterior.

Teclado

Use el driver de teclado implementado en el TP anterior. Agregue las syscalls necesarias para que un programa pueda reaccionar inmediatamente al presionar una tecla (por ejemplo, una system call bloqueante que retorna luego de que una tecla es presionada, sin consumirla del buffer de teclado original).

Video-Texto

Use el driver de video implementado en el TP anterior.

Video-Imagen

Se requiere implementar un driver VBE2 que soporte una resolución de al menos 1024x768 pixeles con color de 24-bits, investigue el estándar VESA y busque en la documentación de Pure64.

Aplicaciones de Userspace

Para mostrar el cumplimiento de todos los requisitos anteriores, deberán desarrollar varias aplicaciones, que muestren el funcionamiento del sistema llamando a las distintas system calls.

Aplicaciones mandatorias

- **sh:** shell de usuario que permita ejecutar las aplicaciones. Debe contar con algún mecanismo simple para determinar si va a ceder o no el foreground al proceso que se ejecuta, por ejemplo, bash cede el foreground cuando se agrega un & al final de un comando.
- **ps:** muestra la lista de procesos con sus propiedades, PID, nombre, estado, foreground, memoria reservada, etc.
- **ipcs:** muestra la lista de estructuras de IPC creadas en el sistema
- **help:** muestra una lista con todos los comandos disponibles
- **game:** explicada más adelante en el apartado Juego
- Agregue sus propias aplicaciones para demostrar el funcionamiento y las capacidades del sistema.

Juego

Se requiere implementar un juego **muy simple (apenas más complicado que un Snake o un Pong)** con sonido y video, el mismo debe tener tres threads comunicados entre sí usando las primitivas de IPC desarrolladas:

- Input: captura la entrada de teclado.
- Render: renderiza las imágenes del juego frame a frame, y calcula la lógica del juego.
- Sound: reproduce música continuamente, hasta que termine el juego.

Informe

El grupo debe presentar un informe el día de la entrega, el mismo debe incluir explicaciones de las system calls desarrolladas y del funcionamiento de los algoritmos.

Consideraciones finales

- El trabajo práctico es grande y no especifica limitaciones, elija sus propias limitaciones para simplificar el desarrollo del sistema, en particular, para evitar tener que programar un sistema con un diseño demasiado dinámico! Un ejemplo de esto es: el sistema solo soporta 512 procesos de hasta 512 MiB. Elija sus limitaciones cuidadosamente y compartalas con la cátedra por mail para verificarlas. A esto se le llama “Hacer asumpciones”.
- Todo el código tomado de cualquier otro sistema debe ser comentado y citado en el informe!

- Se sugiere tener dos consolas: una para la salida y entrada de los procesos, y una consola de debug para la salida del kernel, se puede alternar entre ambas con una tecla o combinación de teclas. La consola de debug del kernel ayuda a mostrar el funcionamiento del mismo!
- No es necesario que el sistema considere los múltiples cores del CPU, si los tiene (es una tarea extremadamente compleja).
- Implementar un algoritmo O(log(n)) para reservar de memoria y O(1) amortizado para liberarla, o un Scheduler de prioridades dinámicas en base a estadísticas es excelente y sube la nota final... **siempre y cuando todo lo demás funcione!** Prioricen el funcionamiento de los requerimientos por encima de la eficiencia.
- Mantenga su diseño simple en base a las limitaciones elegidas, Keep It Simple Student.
- Al igual que en Arquitecturas, no es posible acceder a Kernel Land desde User Land a menos que sea a traves de una System Call.
- El sistema **no requiere** múltiples terminales virtuales.

El sistema **no requiere** un filesystem. Dependiendo de su diseño y del IPC elegido, deberá tomar alguna convención para poder identificar apropiadamente un recurso de IPC usado por dos procesos.

Entrega preliminar

Fecha: Antes del Lunes xxxxxxxxxx
Entregables: Idea del proyecto, diseño de cada hilo.

Entrega Final

Fecha: Lunes xxxxxxxxxx
Entregables: Mail con un archivo ZIP adjunto con el código fuente y el informe, el mail debe incluir el hash MD5 del archivo.
Defensa del trabajo práctico: el mismo lunes de la entrega.