

Sistemas Operativos

Segundo cuatrimestre 2017

TP Nº 2: Construcción del Núcleo de un Sistema Operativo

Profesor: Nestor Esquivel

Adjuntos: Ariel Godio, Rodrigo Rearden

Fecha de entrega: a definir

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

Continuarán con los grupos ya formados.

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 el uso de una de dos soluciones:

- Generar un cross compiler de 64 bits, pueden usar [este](#) sitio como referencia.
- Utilizar Docker como ambiente de desarrollo host portable entre distintas máquinas y OSes (visto en clase).

Requerimientos

El kernel debe implementar 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, Threads, 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), sin requerir que los mismos entreguen el control del CPU, y con alguna estructura/algoritmo que permita seleccionar al siguiente proceso (scheduler).

Debe contar además con implementación de threads, con su propio scheduler dentro de cada proceso.

Syscalls involucradas

- Crear, Finalizar, y Listar procesos
- Crear, Finalizar, y Listar threads

Opcionales (pueden servirles para implementar IPCs más adelante)

- 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 protección de memoria entre procesos, es decir, un proceso no debería poder acceder a las zonas de memoria de otro. Que zona de memoria ocupa cada proceso, y si esas zonas son dinámicas o estáticas, queda a criterio del grupo.

IPCs

Se debe implementar envío y recepción de mensajes bloqueantes, de cantidad fija de bytes, enviados a un identificador común de tipo cadena de caracteres definida entre los procesos que van a comunicarse (puede ser una combinación de PIDs, un nombre de dominio, u otra cosa).

También deberán implementar mutexes, los mismos pueden ejecutar sus operaciones de up y down sobre un identificador acordado.

Syscalls involucradas

Send y Receive para mensajes, siendo la segunda bloqueante. Up y Down para mutexes. El diseño del scheduler debería contemplar estas syscalls. Un thread esperando por un mensaje o mutex no debería bloquear a los otros threads.

Drivers

Se requiere la implementación de dos drivers, teclado, y video en modo texto.

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 en modo texto

Use el driver de video implementado en el TP anterior.

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.
- **philosophers:** muestra una resolución del problema de los filósofos. Se puede incrementar/decrementar en runtime la cantidad de filósofos.

- **prodcons:** muestra una resolución para el problema productor consumidor de buffer acotado, se puede incrementar/decrementar en runtime la cantidad de consumidores y productores.
- **help:** muestra una lista con todos los comandos disponibles
- Es muy importante que agreguen sus propias aplicaciones prácticas **para demostrar el funcionamiento de cada una de las capacidades del sistema**, de otra forma la existencia de las mismas no podrá ser evaluada. Por ejemplo, para mostrar la protección de memoria, debería haber una aplicación que intente acceder a una zona inválida, y sea bloqueada por el kernel.

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.

Entrega Final

Fecha: a definir

Entregables: Mail con o bien un archivo ZIP adjunto con el código fuente y el informe, o bien el id de commit en un repositorio como gitlab, o bitbucket.

Defensa del trabajo práctico: el mismo martes de la entrega.