Sistemas Operativos

Primer cuatrimestre 2018

TP Nº 3: Estructuras de administración de recursos

Profesor: Horacio Merovich
Adjuntos: Ariel Godio, Rodrigo Rearden
Fecha de entrega: 06/06/2018

Introducción

En el TP Nº2 se implementó un kernel con procesos, IPCs, Memory Management y Scheduling. Ahora buscaremos implementar estructuras para agregarle mejor administración de recursos a nuestro sistema.

Grupos

Continuarán con los grupos ya formados.

Requisitos Previos

TP Nº2 completado.

Requerimientos

Deberán elegir 3 requerimientos para implementar entre los que se listan a continuación:

- Virtual Memory Management y Shared Memory
- Pipes
- Threads
- Mejoras al Physical Memory Management
- Filesystem

Virtual Memory Management y Shared Memory

El kernel debe contar con algún sistema para mapear y proteger zonas de memoria, impedir que los procesos accedan a zonas de otros (o incluso al Kernel), y además proveer algún mecanismo para compartir memoria entre varios procesos.

Syscalls involucradas

- Mapear y des-mapear memoria
- Modificar las syscalls anteriores para que implementen vmm.

Pipes

El kernel debe ofrecer un sistema de IPC por pipes, implementando una solución del problema del productor y el consumidor, asimismo la shell debe poder conectar dos procesos de forma similar a la de Unix.

Syscalls involucradas

- Crear, Escribir, Leer y Cerrar pipes.

Threads

Implementar kernel-threads (procesos que comparten su heap)

Syscalls involucradas

- Crear, Terminar y Esperar por Threads.
- Posiblemente haya que modificar la API de semáforos o implementar mutexes

Mejoras al Physical Memory Management

Implementar Buddy Allocation para poder reservar varias páginas físicas contiguas.

Syscalls involucradas

- Las mismas que en el TP anterior

Filesystem

Implementar alguna forma de persistir datos entre diferentes runs de un proceso, para organizar los datos persistidos (archivos) pueden usar jerarquías, tags, o cualquier sistema que se les ocurra, siempre y cuando sea simple de utilizar y ayude a organizar los datos y sea lo suficientemente general.

No es necesario que persistan a disco, pueden hacerlo a memoria, pero si es necesario que el filesystem sea manipulable por la shell. De todas formas si desean implementar persistencia a disco son bienvenidos

Syscalls involucradas

- Open, Close, Read y Write de archivos, las que sean necesarias para organizarlos.

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. 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.

Consideraciones

Es necesario que programen de forma modular y desacoplada. Asimismo se deberá respetar a rajatabla la separación a nivel binario y memoria entre kernelspace y userspace, las apps de usuario se comunican entre sí y con el kernel solo a través de system calls.

Armar casos de prueba, transformarlos en testeos unitarios y derivar las features del sistema operativo no es mandatorio, pero si es muy recomendado, en especial para los algoritmos más complejos.

Asimismo, si se programa de forma desacoplada y a partir de testeos unitarios, utilizar valgrind para revisar errores de acceso de memoria se vuelve trivial, esto es importante ya que debuggear un sistema operativo en runtime es posible, pero muy trabajoso, debería ser la última opción.

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

Fecha: 06/06/2018
Entregables: Mail con id de commit y branch del entregable en un repositorio como gitlab, github o bitbucket.
Defensa del trabajo práctico: 13/06/2018