Sistemas Operativos

TP2 - Construcción del Núcleo de un Sistema Operativo

Primer Cuatrimestre de 2019

Grupo 8

Integrantes

- Karpovich, Lucia 58131
- Martin, Fernando 57025
- Reyes, Santiago 58148
- Tarradellas, Manuel 58091

Introducción

El trabajo consiste en la implementación de un kernel propio, usando como base el Trabajo Práctico Especial de la materia *Arquitectura de las Computadoras.* El proyecto implementa mecanismos de Memory Manager, Scheduling y de IPC.

Physical Memory Manager

Para la implementación del mismo se utilizó la técnica de Best Fit. La memoria se divide en diferentes bloques o particiones las cuales llamaremos *ListNodes*. Cada proceso se asigna de acuerdo con el requerimiento de espacio. Cuando hay más de una partición disponible para acomodar la solicitud de un proceso, se selecciona una partición de acuerdo con el esquema de Best Fit. Este le asigna la primera partición lo más pequeña posible (que cumple con los requerimientos de espacio pedidos) entre la particiones disponibles.

Procesos, Context Switching y Scheduling

Para habilitar multiprocesos se implementó un sistema *Scheduler* que maneja una lista de procesos a ejecutar mediante un sistema de lotería. Los procesos se encuentran representados en una lista de nodos que llamaremos *tPList,* los cuales contienen información sobre cada proceso como su estatus, prioridad, stack pointer, etc.

El *Scheduler* se ejecuta por cada interrupción del timer tick. En cada una de sus ejecuciones chequea si la condición de ejecución para el proceso en ejecución expiró. Si el proceso no completó su quantum (la cantidad de interrupciones de timer ticks que se le asigna a un proceso como tiempo de ejecución continua) se decrementará el valor de quantum según corresponda y finalizará la ejecución del scheduler. De lo contrario se procederá a hacer un sorteo de tickets.

Ante la creación de un proceso se le asigna una cantidad de “tickets” para la lotería y la cantidad dependerá de su prioridad (un proceso con alta prioridad recibirá muchos tickets lo cual significa más chances de ser seleccionado para ejecutarse ante un sorteo). El proceso ganador del sorteo es el que se ejecutará, eligiendolo de la lista de procesos listos para correr.

El scheduler mantiene en paralelo una lista de procesos bloqueados, los cuales no son partícipes del sorteo.

Cuando se cambia el proceso en ejecución por otro que fue elegido según el procedimiento descrito con anterioridad, se genera una copia de respaldo del stack pointer en el instante inmediatamente anterior a la ejecución del código del scheduler en su estructura de proceso. A continuación, se devuelve el stack pointer correspondiente al proceso que comenzará, o continuará, su ejecución para garantizar que la instrucción iretq levantará la información de su contexto a fin de continuar con la ejecución del código.

A fin de poder reutilizar este comportamiento para manejar la inicialización de un proceso nuevo, se recrea manualmente el stack frame de interrupciones cargándose con los valores apropiados que permitan que la efectiva ejecución de su código no se diferencie de un cambio de contexto ya en ejecución.

IPCs

Se implementó un mecanismo de exclusión mutua (Mutex) sencillo que funciona mediante un lock y unlock. Si se pide bloquear y el mutex ya está en modo bloqueado, entonces el proceso que intentaba bloquear, entra en la cola de procesos que están esperando al mutex. El unlock es solo llamado por el dueño del mutex (el que lo bloqueo), y al ejecutarse desbloquea el primer proceso en la cola de bloqueados y le cede posesión del mutex.

Fue implementado, también, un método de señalización entre procesos de semáforos. Este funciona de manera similar al mutex, utilizando un mutex propio y un valor entero. Cuando el valor se incrementa, se libera el primer proceso (si existe) de la cola de procesos en espera del mutex. Cuando el valor se trata de decrementar, si este era mayor a 0, el valor se decrementa. Pero si el valor ya era 0, el proceso que intentó decrementarlo se bloquea y se agrega a la lista de bloqueados del mutex.

Aplicaciones de User space

ps: muestra la lista de procesos con sus propiedades, PID, nombre, estado, foreground, memoria reservada, prioridad, etc.

prodcons: muestra una resolución para el problema productor consumidor de buffer acotado, se debe poder incrementar / decrementar en runtime la cantidad de consumidores y productores (hasta 5). Muestra el funcionamiento de mutex y semáforos.

memtest: Se implementó una función que pide y libera memoria para demostrar el funcionamiento del manejo de memoria.

ptest: muestra el funcionamiento del scheduler mostrando procesos en simultáneo que pueden ser terminados con el comando killtest.

Instrucciones de compilación y ejecución

Requisitos: Tener instalado docker (versión 1.8 o posterior) y quemu.

Para el correcto compilado y ejecución del programa, se recomienda ingresar el comando dado por terminal, con todos los archivos en el mismo directorio:

$> cd src

$> sudo ./docker.sh

Limitaciones

El sistema tiene un límite de memoria a manejar (dado por una variable MEM_SIZE) y una cantidad máxima de particiones de la memoria (dado por la variable MAX_NUM_NODES).

Hay un máximo para la longitud del id de un mutex dado por una variable llamada MAX_MUTEX_ID.

Problemas Encontrados

Los principales problemas se generaron entorno a la creación de procesos, el scheduling y el context switch.

Uno de ellos fue la implementación de EOI (end of interrupt). cuando se generaba una interrupción resultaba que como entraba en acción el scheduler, nunca se retornaba el EOI al pic ya que no volvía a la misma función que manejaba la interrupción. Esto estaba generando que luego de una primera interrupción no se pudiera hacer otras. La solución consistió en hacer que al ejecutar el siguiente proceso el scheduler avisa de la finalización de la interrupción

Otro problema similar surgió cuando se ejecutaba un proceso y al final producía un error. Esto se debía a que al hacer return, hacía pop de basura ya que nadie había "llamado" a esa función. Esto se solucionó mediante el uso de una función wrapper que crea el proceso.

Hubo problemas con las listas genéricas creadas para manejar distintos sistemas, como los mutex, por un tema de que daba errores cuando los elementos eran punteros.

Por otro lado, resultó complejo hacer tests por un tema de que en el TP todo se encuentra muy entrelazado, de modo que probar y arreglar el funcionamiento de una función o parte del proyecto de manera unitaria resultaba imposible porque algunas cosas solo funcionan cuando son ejecutadas en el contexto del trabajo práctico. En otras palabras, se hizo muy poco unit testing.

Fuentes

Algoritmo Best Fit

https://courses.cs.vt.edu/csonline/OS/Lessons/MemoryAllocation/index.html

https://www.geeksforgeeks.org/operating-system-memory-management-partition-allocation-method/

Lottery

https://intro2operatingsystems.wordpress.com/tag/lottery-scheduling/

http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-lottery.pdf

IPC

https://people.mech.kuleuven.be/~bruyninc/ecs/AsynchronousSynchronization.pdf