Sistemas Operativos

Primer Cuatrimestre - 2021

Docentes

Aquili, Alejo Ezequiel
Godio, Ariel
Merovich, Horacio Víctor
Mogni, Guido Matías

Trabajo Práctico Nro. 2
“Construcción del Núcleo de un Sistema Operativo y estructuras de administración de recursos.”

GRUPO 03

Integrantes

Catolino, Lucas - 61817
Cerdeira, Tomás - 60051
García Montagner, Santiago - 60352

Fecha de entrega

24/05/2021

Decisiones de desarrollo

Memory Manager

El SO cuenta con dos implementaciones distintas de un administrador de memoria: un memory manager que utiliza un bitmap y el buddy system.

Bitmap

Se inicia la memoria con una cantidad finita de bloques. Esta cantidad es calculada como el tamaño total de la memoria dividido el tamaño de los bloques. Al querer reservar memoria, se le asigna al pedido una cantidad de bloques dependiendo de cuánta memoria se pide. Luego se recorre el arreglo buscando esa cantidad de bloques contiguos libres para devolver la posición inicial de memoria. En caso de no haber la cantidad de bloques contiguos necesarios, se retorna null. Al momento de liberar la memoria, se marcan como liberados los bloques pertenecientes a la posición de memoria que se quiere liberar.

Buddy System

El sistema buddy es un sistema en el que la memoria disponible está representada por bloques de tamaño 2^n. Se decidió implementarlo con un árbol para ir referenciando a las distintas posiciones de memoria necesarias. Al querer reservar memoria, se calcula qué potencia de 2 es lo suficientemente grande como para almacenar lo pedido, y se devuelve el puntero correspondiente al nodo del árbol para esa altura. Al momento de liberar la memoria, se tiene en cuenta que si dos nodos hermanos de 2^{n-1} son liberados, el padre debe volver a unificar esos bloques de memoria en uno de 2^n.

Scheduler

Se decidió implementar un scheduler con un número fijo de procesos. De llegar a la cantidad máxima de procesos activos/bloqueados, la solicitud de crear uno nuevo retorna el valor -1, en vez del PID del mismo.
El algoritmo utilizado para cambiar entre los procesos activos es la “Planificación Round-robin” con prioridades discutida en clase. Las mismas, vienen dadas por el “timeSlot” asignado en su creación.
Una vez un proceso es elegido para ser corrido, toma el hilo de ejecución la cantidad de veces preestablecidas en su timeSlot, mientras el mismo mantenga un estado de “ACTIVO”. De ser bloqueado, cambiando su estado a “BLOQUEADO”, el algoritmo elige al siguiente proceso disponible en la cola, repitiendo el procedimiento.
La primera vez que se crea un proceso, se crea además un proceso “Halter” en la posición 0 del arreglo, el cual entra en acción cuando no hay ningún proceso activo. Este, se encarga de bloquearse en un halt a la espera de la creación de un nuevo proceso o el cambio de estado de alguno bloqueado.
Para retornar de un proceso, se usó la implementación de un wrapper quien se encarga de hacer el “exit”, matando al mismo y forzando una interrupción del timer tick para forzar al scheduler la elección del siguiente.

Observación: la función “initStack”, encargada de hacer el armado del stack de cada proceso, fue implementada en Assembler.

Inter Process Communication

Se decidió implementar pipes para conectar dos procesos. Un pipe está compuesto por un índice de lectura y otro de escritura, un file descriptor y un buffer por el que pasa la información. Además tiene un flag para determinar si está libre o no.
Al inicializar, se crea un arreglo estático de pipes liberados con una cantidad predefinida. A medida que se van creando pipes, se agregan a dicho arreglo.
La funcionalidad de los pipes es conectar la salida de un proceso con la entrada de otro.

Sincronización

Se decidió implementar los semáforos con un arreglo estático en el que se inicializan libres. Cuando un proceso quiere abrir un semáforo, se recorre el arreglo de semáforos para asignarle alguno libre. En caso de no haber, se retorna -1.
Los semáforos tienen un ID y un nombre. Además, guarda información sobre los procesos asociados a él.

Instrucciones de instalación

Junto con el código se adjunta un archivo make, por lo que el usuario deberá situarse en el directorio del archivo y correr los comandos “*make clean*” y “*make all*”.
En caso de querer utilizar la memoria con el sistema de bitmap o el sistema Buddy, se debe proceder de la siguiente manera:

- En kernel.c se tienen los siguientes includes

```
#include <buddy_system2.h>
#include <memory_manager.h>
```

  Se debe comentar el sistema que no se quiera utilizar y dejar descomentado el sistema que se quiera usar.
- En pipe.h se tienen los siguientes includes

```
#include <memory_manager.h>
#include <buddy_system2.h>
```

  Se debe comentar el sistema que no se quiera utilizar y dejar descomentado el sistema que se quiera usar.
- En scheduler.c se tienen los siguientes includes

```
#include <memory_manager.h>
#include <buddy_system2.h>
```

  Se debe comentar el sistema que no se quiera utilizar y dejar descomentado el sistema que se quiera usar.
- En test_mm.h se tienen los siguientes includes

```
#include <memory_manager.h>
#include <buddy_system2.h>
```

Se debe comentar el sistema que no se quiera utilizar y dejar descomentado el sistema que se quiera usar.

- Por último, se deben comentar los .c y .h que no se quieran utilizar (por ejemplo, si se quiere utilizar memory_manager se deben comentar los archivos de buddy_system2.c y buddy_system2.h).

Limitaciones

Entrada de teclado

El sistema no acepta el 9 como entrada del teclado. Esto es debido a un error arrastrado del trabajo práctico anterior, por lo que se decidió no dedicarle tiempo a correcciones que fueran más allá del alcance de esta materia.

Cat y filter

Los procesos cat y filter no toman los espacios. El scanf implementado anteriormente separa los strings cuando aparece un espacio. Dicha lógica se usa dentro de la shell por lo que cambiarlo conlleva un esfuerzo que no es relevante en este tp.

Memoria

Para el sistema de bitmap, la memoria total tiene un tamaño de 4096 * 20, y cada bloque un tamaño de 8. Para el sistema Buddy, la memoria total tiene un tamaño de 4096 * 16, y el bloque más chico tiene un tamaño de 4, representando un árbol asociado de 32767 nodos posibles.

Procesos

La cantidad máxima de procesos ejecutándose es de 11. El proceso Halter se posiciona en la posición 0 del arreglo, y el proceso Shell en la posición 1.

Semáforos

La cantidad máxima de semáforos que se pueden crear al mismo tiempo es de 10.

Pipes

La cantidad máxima de pipes que se pueden crear al mismo tiempo es de 10.

Test de memoria

Si al momento de ejecutar el test de memoria se hace un kill del mismo, y se termina el proceso antes de liberar la memoria, esta quedará inutilizable. Esto es así porque no es requisito del trabajo implementar un garbage collector que se encargue de liberar la memoria.

Problema de los filósofos

Si al momento de ejecutar el problema de los filósofos se hace un kill del proceso padre, los procesos hijos seguirán vivos ya que no se implementó un garbage collector que libere la memoria ni se encargue de los procesos huérfanos.
La cantidad máxima de filósofos está sujeta a la cantidad de procesos ya existentes.

Problemas encontrados

Lógica del scheduler

Luego de 4 o 5 implementaciones distintas, se llegó a una implementación relativamente simple y coherente. A lo largo de los distintos desarrollos se fue observando que el uso de distintas variables tipo flags terminaban complejizando el entendimiento del código y generando muchos escenarios distintos. Esto fue ocasionando errores imposibles de debuggear y depurar.

Configuración y uso de GDB

Muchos errores se fueron solucionando haciendo prints a lo largo del código para hacer un seguimiento del mismo. Esto generaba un proceso lento e ineficiente de debuggeo. La herramienta GDB propuesta por la cátedra resultó una herramienta eficaz y rápida, aunque la curva de aprendizaje resultó un poco lenta. Una vez familiarizados con los comandos, la resolución de errores resultó ser mucho más ágil.

Read bloqueante

Cumplir con este requisito derivó en una implementación nueva del scheduler. La versión anterior a la última tenía tantos escenarios posibles que fue imposible encontrar el error que no permitía que el read fuera bloqueante. Luego de más de 4 horas con la ayuda del docente, se decidió refactorizar todo, logrando una versión funcional.

Código de terceros

Problema de los filósofos

- https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/
- Libro: *“Modern Operating Systems”, Tanenbaum* (capítulo 2.5.1: The Dining Philosophers Problem)

xadd, semWait, semPost

- Clase práctica del 03/05/2021

Modificaciones realizadas a los tests provistos

Se portaron todas las funciones correctamente.

Warnings

1.

```
sysHandler.c:149:40: warning: passing argument 2 of 'startProcess' from incompatible
pointer type
        startProcess("test_processes", &test_processes, NULL, NULL, par3);
```

Este warning viene por el prototipo de la función definido en la materia anterior, por lo que se decidió ignorarlo.

2.

```
shell.c:214:17: warning: division by zero [-Wdiv-by-zero] int i = 1 / 0;
                                                                  ^
```

Esto es así para lanzar una excepción (viene de la materia anterior).

3.

```
sysHandler.c:204:15: warning: assignment makes integer from pointer without a cast
        *par1 =mallocNUESTRO((int)par2)
```

La solución de este warning implicó que no funcionara como se esperaba, por lo que se decidió ignorarlo.

4.

```
consoleManager.c: In function 'newLine':
consoleManager.c:71:10: warning: variable 'posY' set but not used
[-Wunused-but-set-variable]
     int *posY;
```

Warning de la materia anterior.

5.

```
scheduler.c: In function 'ps':
scheduler.c:291:32: warning: passing argument 2 of 'numToStr' makes integer from pointer
without a cast numToStr(auxStack, processes[i].stackPointer);
```

Este warning viene de la definición de numToStr, que debería recibir un int en vez de un uint64_t del stackPointer. No es posible castear la entrada, y no tiene sentido una implementación paralela al numToStr que reciba un uint64_t sólo para evitar un warning.

6.

```
videoDriver.c: In function 'calculatePosition':
videoDriver.c:58:9: warning: cast to pointer from integer of different size
[-Wint-to-pointer-cast]
  return (char *)(screenData->framebuffer + (x + screenData->width * y) * 3);
```

Warning de la materia anterior.

7.

```
sysHandler.c: In function 'read':
sysHandler.c:288:31: warning: 'keyboardBuffer' may be used uninitialized in this function
[-Wmaybe-uninitialized]  if (keyboardBuffer[i] == ESPACE)
```

Warning ignorado, ya que keyboardBuffer se utiliza para analizar la entrada del teclado.

8.

```
buddy_system2.c:79:7: note: expected 'struct node *' but argument is of type 'struct node
*' void *searchMemoryRec(node * n, int sizeRequired)
      ^
```

Warning ignorado.

Análisis con PVS

Siguiendo las instrucciones provistas por la cátedra se procedió a un análisis con PVS. Los errores encontrados fueron los siguientes:

- www.viva64.com/en/w 1 err Help: The documentation for all analyzer warnings is available here: https://www.viva64.com/en/w/.
  Esto no es considerado un error.

- /root/Bootloader/BMFS/bmfs.c 552 err V595 The 'disk' pointer was utilized before it was verified against nullptr. Check lines: 552, 575.
  Esto no es considerado un error nuestro ya que se encuentra en Bootloader.

- /root/Userland/SampleCodeModule/shell.c 246 err V609 Divide by zero.
  Justificado anteriormente, esto es así por una decisión de diseño de la materia anterior, para lanzar una excepción de dividido por cero.

- /root/Kernel/buddy_system2.c 204 err V575 The 'memcpy' function doesn't copy the whole string. Use 'strcpy / strcpy_s' function to preserve terminal null.
  Este error no es tenido en cuenta, ya que memcpy viene del trabajo anterior.

- /root/Kernel/font.c 4 err V1042 This file is marked with copyleft license, which requires you to open the derived source code.
  Esto no es considerado un error.

- /root/Kernel/synchro.c92 err V779 Unreachable code detected. It is possible that an error is present.
  Según la documentación de PVS, este error se presenta cuando existen bloques de código que jamás serán ejecutados (por ejemplo, si se quiere imprimir después de un return). No es el caso del synchro.c, por lo que este error no es tenido en cuenta.

- /root/Userland/SampleCodeModule/philo.c 162 err V522 Dereferencing of the null pointer 'status' might take place.
  Esto no es considerado un error, ya que la variable que PVS detecta que podría llegar a ser null es creada anteriormente, por lo que sus valores están controlados.

- /root/TP2/TP-Arqui-2/Kernel/scheduler.c 135 err V594 The 'halter.memory + 4096' pointer steps out of array's bounds.
  Esto no es considerado un error, ya que la constante fue actualizada y PVS no lo actualiza.

Análisis con CPPcheck

- [Userland/SampleCodeModule/shell.c:245]: (error) Division by zero.
  Justificado anteriormente, esto es así por una decisión de diseño de la materia anterior, para lanzar una excepción de dividido por cero.

Comentarios

Dentro de la implementación del problema de los filósofos se decidió agregar la opción “p” para que, al ejecutarla, se corra el comando ps mostrando el estado de los procesos. Esto es así por una decisión que se tomó para poder evaluar el correcto funcionamiento del bloqueo por semáforos y el estado de los procesos. Cabe aclarar que el comando ps se ejecuta de forma independiente del estado actual de los procesos al momento de ejecutarlo, por lo que no muestra el estado de los procesos en la última línea ejecutada, sino en un instante posterior.