Trabajo Práctico 2:

Mecanismos de administración de recursos

Integrantes:

- Mateo Reino
- Jonatan Blankleder
- Agustin German Ramirez Donoso

1 Introducción

El objetivo del tp se puede dividir en 4 objetivos principales: Crear un manejador de memoria, un manejador de procesos, una sincronización dentro de dichos procesos y diferentes aplicaciones de UserSpace pedidas por la cátedra. Fuimos capaces de implementar las 4 funcionalidades aunque tuvimos un problema principal el cual vamos a explicar más adelante

2. Manejo de memoria

2.1 Implementación

Hemos implementado dos formas de manejar la memoria, la primera es la memoria con capacidad de reservar la memoria y liberarla, la segunda forma de manejarla es con el Buddy allocator que consiste en tener listas de bloques de tamaño igual para ser más eficiente.

make usa bump allocator con free (lista simple).

2.1 Limitaciones

make buddy compila un buddy system de 2^k páginas.

Hemos utilizado un bloque de 1 MiB para el heap.

3. Procesos

3.1 Implementación

PCB: Array fijo de 128 entradas (struct pcb); campos : PID, estado, prioridad, punteros de stack, regs salvados.

Scheduling basado en el algoritmo **Round-Robin** con prioridades.

Además, concluimos que la prioridad máxima en nuestro proyecto iba a ser de 7 ya que consideramos que la diferencia entre prioridades 1 y 7 es lo suficiente. Esta diferencia se puede ver corriendo test_priority.

3.2 Limitaciones

Puede haber hasta 128 procesos al mismo tiempo.

4. Semáforos

4.1 Implementación

Buscamos hacer que la implementación de los semáforos sea lo más parecida a la vida real viéndolo desde el lado del usuario. Creamos syscalls para sem_init, sem_post,

sem_wait y sem_destroy. 2 procesos se pueden conectar al mismo semáforo sabiendo su nombre.

Cuando un proceso se bloquea pasa a estar en una cola. Siempre que se hace un sem_post se liberan todos los procesos en la cola de ese semáforo.

4.2 Limitaciones

Puede haber hasta 128 procesos esperando en un semáforo.
Puede haber hasta 32 semáforos.
La longitud máxima del nombre de un semáforo es de 32 caracteres.

5. Pipes

5.1 Implementación

La implementación de los pipes también se buscó que sea lo más parecida a la vida real posible (vista desde el usuario). Además, 2 procesos se pueden conectar a una misma pipe sabiendo su nombre.

5.2 Limitaciones

Tamaño máximo del buffer de la pipe: 1924.
Longitud máxima del nombre: 32 caracteres.
Máxima cantidad de pipes: 32.

6. Aplicaciones

6.1 Implementación

Buscamos diseñar todas las aplicaciones y se las pueden ver corriendo el comando help. Aun así, hubo algunas que queremos aceptar que no están perfectamente hechas

- cat: imprime el stdin y no está hecho para conectarse con otros procesos
- wc: lo mismo, siempre termina imprimiendo 1

Aparte de esos 2, el resto debería funcionar sin grandes problemas.

7. El problema

Tuvimos 2 grandes problemas: El primero fue que nos costó mucho tiempo ser capaces de poner un proceso a correr. Y el segundo, que **no lo solucionamos**, el cual es simplemente que el proyecto no tiene la suficiente memoria cuando se pone en ejecución, lo que causa que una variable global pierda su valor y no podamos llamar a las diferentes aplicaciones.

Por mail fue mandado un instructivo de cómo replicar el error en una versión vieja del proyecto, pero el error es el mismo y la solución debería ser la misma.

Aun así, a pesar del error seguimos realizando el tp aunque nuestra única posibilidad de debugeo es la compilación y no correr el proyecto y probar los comandos. Esperamos que la cátedra nos de la posibilidad de solucionar cualquier problema que haya debido a la falta de pruebas en el tp debido a que es una solución rápida

2.3 IPC - Pipes Unidireccionales

Se crean con un buffer máximo de 1024, con lecturas y escrituras bloqueantes, idénticas a los semáforos. El nombre de los pipes al igual que los semáforos tienen como longitud máxima 32.

2.4 Semáforos

Diseño inspirado en los semáforos de Linux. Con down/up sin busy wait. sem_wait/post con spinlock + wait-queue. Bloquea al hilo y planifica.

Con una cantidad máxima de semáforos de 32, con un nombre de longitud máxima de 32 caracteres. La de waiters en un semáforo es de 128.

3 Compilación y ejecución

Para compilar y ejecutar, se encuentran hechos archivos bash para crear el contenedor (create-container.sh), compilar (compile.sh), limpiar los archivos .o (clean.sh) y ejecutar los binarios/imágenes (run.sh). Si está usando WSL probablemente sea necesario hacer dos2unix a los mismos para que se ejecuten correctamente

Comandos a ejecutar (en orden):

3.1 Crear contenedor

./create-container

3.2 Compilar kernel y userland

./compile.sh

3.3 Ejecutar en QEMU

./run.sh

4. Instrucciones para demostrar el funcionamiento de los requerimientos

Filósofos comensales(semáforos sin busy waiting): sh phylo 5

Memoria física: **sh mem** //imprime total

Pipes & redirección: **sh cat < /etc/motd**