Instituto Tecnológico de Buenos Aires

72.11 - Sistemas Operativos

Segundo Cuatrimestre 2023 - Grupo 5

TP2

Autores:

Luca Valentin Bloise (63004)
Cristian Nicolás Tepedino (62830)
Florencia Carrica (62040)

Decisiones tomadas durante el desarrollo

En cuanto a los memory managers, además de la implementación del memory manager Buddy requerido por el enunciado, implementamos uno utilizando dos listas doblemente enlazadas. Una lista para el espacio libre y otra para el espacio ocupado. Inicialmente intentamos implementar un memory manager basado en el de K&R, sin embargo nos pareció algo complicado y confuso en cuanto al código por lo que optamos por una lista.

En cuanto al algoritmo de scheduling, optamos por utilizar colas de procesos, cada una representando un nivel de prioridad. La prioridad más baja es la que corre primero. Optamos por incrementar la prioridad al finalizar cada quantum, y aumentarla más si se bloquea antes de que eso ocurra.

Instrucciones de compilación y ejecución

1. Moverse a la carpeta toolchain.
2. Ejecutar el comando “make all”
3. Volver al directorio del proyecto
4. Ejecutar nuevamente “make all”
5. Para compilar con un memory manager específico:
    a. “make all MM=BUDDY” compila con el memory manager buddy. (Este es el comportamiento predeterminado si no se aclara ninguno)
    b. “make all MM=FREE_LIST” compila con el memoria manager de lista.
6. Para ejecutar correr “./run.sh” desde el directorio del proyecto.

Pasos a seguir para demostrar el funcionamiento de cada uno de los requerimientos

Para demostrar el funcionamiento del memory manager, los procesos, la prioridad de procesos y los semaforos, correr los siguientes comandos desde la shell:

- testmm (Recibe como único argumento la memoria máxima)
- testprocess (Recibe la cantidad máxima de procesos)
- testprio (No recibe argumentos)
- testsync (Recibe la cantidad de iteraciones y si usar semáforos (1) o no (0))
- phylo

Cabe destacar que el testsync presenta un problema detallado más adelante, por lo que la demostración del funcionamiento de los semáforos es con el programa de los filósofos.

Si se desea correr en el background, escribir el comando seguido de un “&” al final, con un espacio. Por ejemplo:

“$ testprocess 10 &”

Para el funcionamiento de los pipes, correr dos programas con un símbolo “|” entre ambos. Por ejemplo, para los programas “cat” y “wc” correr:

“$ cat | wc”

Además de estas opciones, la shell provee otros comandos para probar el funcionamiento de los requerimientos.

Limitaciones

Solo se puede hacer pipe entre dos procesos desde la shell utilizando “|”, no más de dos.
Luego de cada context switch, la prioridad del proceso que estaba corriendo sube. Esto se decidió para poder permitir a cada proceso correr de forma más justa. Se puede ajustar el algoritmo de context switch para evitar esto de forma sencilla, en caso de que se quiera correr hasta terminar primero los procesos de mayor prioridad y luego los de menos.

Problemas encontrados durante el desarrollo

El principal problema, el cual no pudimos resolver, es que el test sync no funciona. No pudimos determinar la causa, pero, dado que en las pruebas que hicimos previamente y en el problema de los filósofos parecen funcionar correctamente, no creemos que sea un problema con los semáforos en sí. Por falta de tiempo, no hemos podido arreglarlo.