TRABAJO PRÁCTICO NRO. 1

INTER PROCESS COMMUNICATION

Nicolás Clozza - 56268
Constanza de Rienzo - 56659
Federico Mamone - 56255
Lucía Tay - 56244

INTRODUCCIÓN

En este informe intentaremos detallar las decisiones tomadas a la hora de desarrollar el trabajo realizado. Éste consiste en aplicar md5sum a los archivos que se encuentran en un directorio enviado por parámetro a un proceso aplicación. Dicho proceso crea procesos esclavos y distribuye dichos archivos entre ellos para que calculen el hash md5 sobre dichos archivos. Luego, al ejecutar el proceso vista, se muestran los resultados por salida estándar (de no ser invocado, no se muestra nada). Por ultimo, el proceso aplicación vuelca los resultados en un archivo en disco (independientemente de si se ejecuta el proceso vista o no).

DECISIONES TOMADAS DURANTE EL DESARROLLO

PROCESO APLICACIÓN

En un principio quisimos utilizar pipes para la comunicación entre el padre y los hijos, proporcionando un pipe individual desde el padre a cada hijo, y un único pipe entre todos los hijos y el padre (Se puede ver la implementación en el commit 151d218f8735e43584ba65482cbc56d916f752af). El problema de esta solución es que dos hijos podrían escribir al mismo tiempo en el pipe, quedando un contenido no deseable en el mismo. Se podría haber solucionado este problema utilizando semáforos de manera que dos hijos no puedan escribir al mismo tiempo, pero implicaba cierta complejidad de código. Además, como no sabíamos cuántos caracteres leer, lo que decidimos fue calcular el largo del nombre del path y concatenarlo al principio del mensaje enviado al pipe.

Otra solución podría haber sido tener un pipe exclusivo entre cada hijo y el padre, pero caíamos en el problema en que el padre podría detectar como que dos hijos terminaron al mismo tiempo cuando en realidad no fue así (por ejemplo, el hijo “a” termina en el tiempo 1 y el hijo “b” termina en el tiempo 2 pero el padre se fija cada 3 tiempos, entonces levanta los dos al mismo tiempo). Esto supone un problema porque no garantiza que después el padre los encolaría en orden de llegada.

Por lo tanto optamos por usar dos message queues, una que almacena todos los paths de los archivos a hashear (llamada mqSendPaths) y otra en donde los hijos envían los hashes producidos (llamada mqReceiveHashes). Al comenzar el proceso el padre toma todos los archivos dentro del directorio recibido por parámetro y los agrega a la primer message queue. De esta manera nos aseguramos de que cada hijo consuma directamente del buffer de la message queue una vez que se liberó del trabajo que estaba realizando, y de la misma forma al escribir en el buffer de los hashes procesados, ya que si dos hijos quieren escribir al mismo tiempo se realiza una escritura por orden de llegada (FIFO).

Si bien las message queues son más complejas de usar que los pipes, a nuestra forma de usarlas nos resultó más sencillo (utillizamos las provistas por POSIX ya que son muy eficientes y nos permiten no tener que preocuparnos por la sincronización).

Se tomó la decision de poner un sleep de 10 segundos antes de empezar a distribuir los archivos para que el usuario tenga tiempo de correr el proceso vista.

PROCESO VISTA

Se decidió utilizar semáforos para limitar el acceso a la memoria compartida entre este proceso y la aplicación principal y de esta forma evitar que uno lea mientras el otro está escribiendo.

INSTRUCCIONES DE COMPILACIÓN Y EJECUCIÓN

PROCESO APLICACIÓN

- Compilación: `gcc main.c -o main queue.c hash.c appendStringToFile.c -lpthread –lrt`
- Ejecución: `./main ./dir`

PROCESO VISTA

- Compilación: `gcc view.c -o view -lrt –lpthread`
- Ejecución: `./view <PID del proceso aplicación>`

LIMITACIONES

- El proceso aplicación sólo puede recibir un directorio que contenga no más de diez archivos.
- Todos los archivos a procesar tienen que tener un path menor o igual a 216 caracteres.
- Al ejecutar la aplicación pasando por parámetro un directorio que contiene subdirectorios adentro se imprime por salida estándar un mensaje de error por cada subdirectorio ya que el proceso intenta calcular el hash de éstos pero no es posible (sin embargo, los archivos que se encuentran dentro de los subdirectorios son hasheados con normalidad).
- Los nombres de los archivos no pueden tener espacios porque habría que parsear ese string y remover los caracteres especiales.

CONCLUSIÓN

Este trabajo nos permitió aprender sobre el uso de procesos y la comunicación entre ellos para poder distribuir tareas y compartir recursos. Si bien al inicio tuvimos problemas para comunicar al proceso padre con los procesos hijos lo pudimos resolver mediante el uso de message queues. Además, tuvimos que aprender a usar semáforos para poder coordinar los accesos a la memoria compartida.