Trabajo Práctico Nro. 1
72.11 - Sistemas Operativos
2C 2022

Fecha de entrega: 12/09/2022

Magdalena Flores Levalle 60077
Pedro López Guzmán 60711
Camila Sierra Pérez 60242
Martín E. Zahnd 60401

Grupo 11

Índice

1. Decisiones tomadas durante el desarrollo

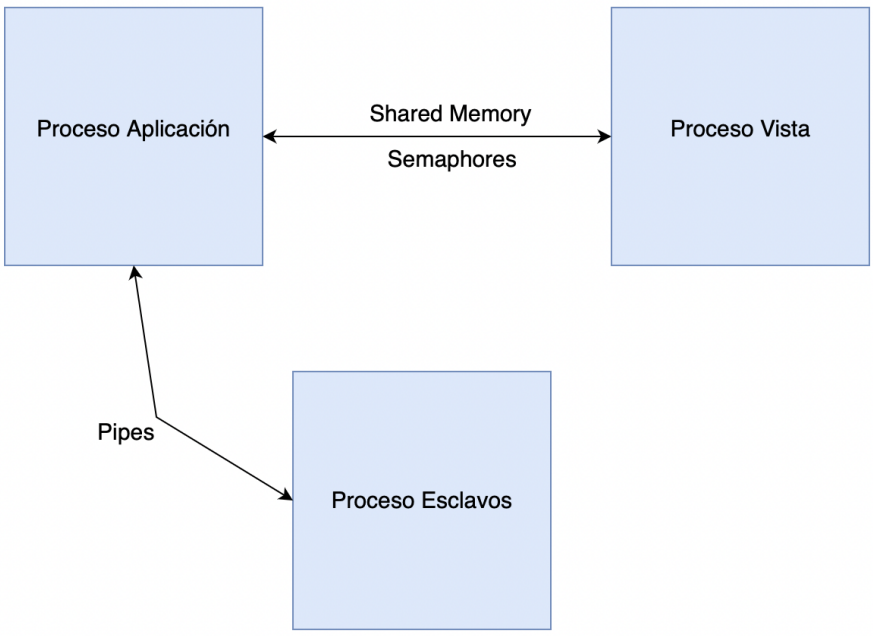
Para el desarrollo del trabajo tuvimos que tomar una serie de decisiones. En primera instancia, optamos por la utilización de pipes para comunicar a los slaves con el main, debido que, al conectar directamente padres con hijos, consideramos que no era necesario el uso de named pipes.

Además, decidimos que el proceso de *view* reciba el tamaño que tendrá la *shared memory* de dos posibles formas:

- Una forma es que el proceso *view* reciba el tamaño de la *shared memory* directamente por entrada estándar, en caso de que el proceso *main* se pipee al *view*.
- La otra opción es que se pase el tamaño de la memoria como argumento en la linea de comandos, para poder ejecutarlos desde la otra terminal.

Por último, decidimos no pasarle una cantidad fija de trabajos al *view*, ya que este es dinámico y la condición de corte no depende de dicho valor. En cambio, se utiliza un caracter especial definido como `END_CHAR`.

2. Diagrama ilustrando cómo se conectan los diferentes procesos



3. Instrucciones de compilación y ejecución

Para poder compilar es necesario encontrarse dentro del directorio raíz del proyecto. Una vez ahí con el comando make se hará la compilación completa. Una vez compilado, se puede ejecutar de dos formas:

```
./bin/md5 [archivos separados por espacio] | ./bin/view
```

O, teniendo otra terminal, se pueden ejecutar los comandos por separado. Esto va a funcionar de la misma forma ya que el view accede a la shared memory. Para ejecutar los programas de esta manera se debe hacer:

```
./bin/md5 [archivos separados por espacio]
```

Y en otra terminal:

```
./bin/view
```

4. Problemas encontrados durante el desarrollo y cómo se solucionaron

Durante el proceso de realizar el trabajo nos encontramos con dos dificultades que, afortunadamente, pudimos solucionar.

En primer lugar, los primeros *slaves* no tenían funcionalidad y todas las tareas terminaban siendo realizadas prácticamente en su totalidad por el último *slave*. Esto era debido a que realizamos un cambio para poder deterctar varios outputs de un mismo esclavo: implementamos tokens en el output para ver cuántas tareas se habían completado. Lo que no notamos fue que esto solo se estaba decrementando en 1 sin importar cuántas tareas finalizaban. Traducido a código, faltaba decrementar la variable *remaining tasks* en el correspondiente ciclo.

El otro problema al cual nos tuvimos que enfrentar fue que, al correr el proceso *view* en otra terminal, funcionaba y terminaba de manera exitosa. En cambio, cuando se corría este mismo proceso con un pipe, no imprimía. Nos resultó un tanto inusual este problema pero en el proceso de debuggeo nos dimos ceunta de que el proceso *main*, cuando lo corríamos utilizando pocos archivos, terminaba antes de que pudiera inciarse el proceso *view*.

En otras palabras, cuando terminaba el proceso del *main*, este cerraba la memoria compartida, por lo que cuando iniciaba el proceso *view* abría otra *shared memory* distinta a la que utilizó el proceso anteriormente mencionado.

Al mismo tiempo, si no cerrábamos la memoria compartida, funcionaba el pipe. En ese momento fue que nos dimos cuenta que necesitábamos saber cuando el proceso main estaba siendo pipeado para poder comunicarle que no cierre la memoria compartida. Así será el proceso *view* quien tiene la responsabilidad de cerrarla.

5. Citas de fragmentos de código reutilizados de otras fuentes

Tomamos como referencia para el desarrollo del trabajo los siguientes dos libros: *The Linux Programming Interface - Michael Kerrisk* y *"Operating Systems Concepts"(Silberschatz, Galvin, Gagne).*

Por último, para la resolución del problema con la *shared memory* (mencionado anteriormente), utilizamos el siguiente : link.

6. Repositorio

Repositorio del trabajo práctico: mzahnd/sistemas-operativos-tp1

Hash del commit: `cad248e63607c39527d7fd0f24c78288474a557b`

Branch del commit: `main`