Sistemas Operativos

Primer Cuatrimestre - 2021

Docentes

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

Trabajo Práctico Nro. 1
“Inter Process Communication”

GRUPO 03

Integrantes

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

Fecha de entrega

05/04/2021

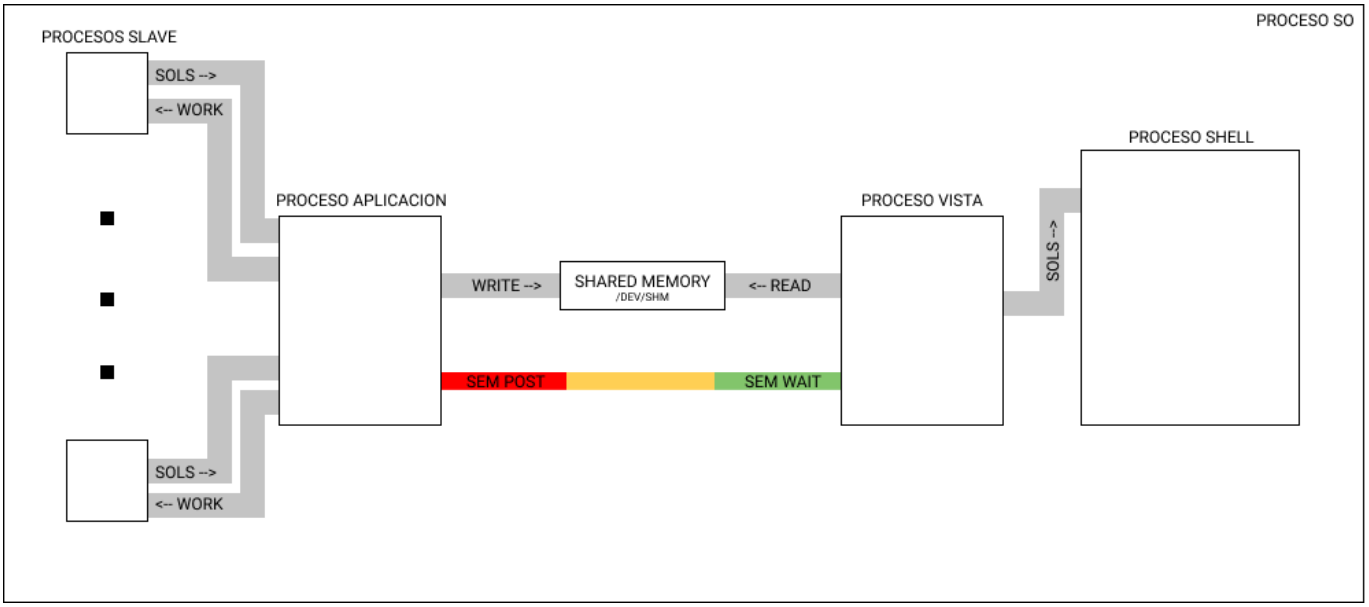
Decisiones de desarrollo

Se decidió seguir la cantidad de procesos esclavos propuesta en el enunciado del trabajo práctico, por lo que se crearon 5 procesos esclavos que reciben, inicialmente, 2 archivos para resolver.
Para la comunicación entre el master y los esclavos se decidió utilizar dos unnamed pipes:

- fd_work: por este pipe, el master envía los archivos a los esclavos para que estos resuelvan.
- fd_sols: por este pipe, los esclavos envían las soluciones para que el master procese.

Para la comunicación entre el master y la vista se creó un espacio de memoria compartida con un semáforo. De esta forma se coordinan los accesos de lectura y escritura, para evitar condiciones de carrera y deadlocks.
El proceso solve vuelva los resultados en un archivo llamado “output_solve”. En caso de no existir lo crea, y en caso de existir guarda los nuevos resultados, dejando un historial de antiguas resoluciones.

Diagrama de procesos



(LINK)

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*”. Una vez ejecutados, se crearán 3 archivos: solve, slave y vista. Estos procesos pueden ser ejecutados de la siguiente forma:

- ./solve *path*: se ejecuta el proceso padre y se le envían los archivos dentro de *path*. Por consola no se mostrará ningún resultado, sino que serán exportados al archivo “output_solve” que aparecerá en el directorio.

- ./solve *path* | ./vista: se ejecuta el proceso padre y el proceso vista, quien se conectará a un espacio de memoria compartida para leer lo que escriba el proceso padre.
- ./slave (enter) *file* (control + d): se ejecuta el proceso slave y procesa el archivo (o archivos) que se le envíe dentro de la ejecución. El control + d corresponde al EOF de la shell para que corte el proceso slave.

Limitaciones

Por decisiones de implementaciones, el master tiene 5 procesos esclavos, a los cuales les envía 2 archivos a cada uno en una primera pasada. Por este motivo, **se toma como precondición que se manden al menos 10 archivos para procesar.**

Problemas encontrados

Algunos problemas encontrados fueron:

- Cómo parsear la salida: se utilizó la función *popen*, que junto con la ayuda provista por la cátedra en el apartado “Hint” logró parsear la salida de la forma buscada y en pocas líneas de código.
- La función *select* no funcionaba como se esperaba: en un primer momento se tuvo una mala interpretación de la función, creyendo que con sólo chequear qué file descriptor había cambiado éste iba a quedar marcado como leído y no volvía a ser recibido por la función en una salida posterior. Leyendo más detenidamente el manual y consultando con la cátedra, se entendió que había que leer la respuesta retornada para que no volviera a ser evaluado ese file descriptor.
- El munmap devolvia “EINVAL”: esto era porque se usaba un único puntero a memoria compartida para hacer la escritura, incrementándolo con cada resolución de cada archivo. Al momento de llamar a dicha función, se le pasaba este puntero “corrido” haciendo que munmap fallara. Se soluciono creando dos punteros, uno auxiliar para ser modificado y otro para guardar la posición inicial.

Código prestado

- Ejemplo de uso de Shared Memory:
    - https://github.com/WhileTrueThenDream/ExamplesCLinuxUserSpace
- Comando *tar* para eliminar espacios en blanco del parseo de *grep*
    - https://stackoverflow.com/questions/12763944/shell-removing-tabs-spaces