Trabajo Práctico 2 - Grupo 7

(72.11) Sistemas Operativos

Integrantes

- Martina Schvartz Tallone - 62560
- Josefina Míguez - 63132

1. Instrucciones de Compilación y Ejecución

Para poder tener los permisos necesarios para correr el proyecto usando `./run.sh` se deberán correr los comandos `sudo groupadd docker` y `sudo usermod -aG docker $USER`. Opcionalmente se puede correr `newgrp docker`, de esta forma no se tiene que reiniciar la terminal para que los cambios sean aplicados.

Para poder compilar y correr el trabajo, se deberán seguir los siguientes pasos:

1. Correr el contenedor y posicionarse en el directorio `root`.
2. Posicionarse en la carpeta `Toolchain` y correr en la terminal del contenedor `make clean all`.
3. Dentro de `root` otra vez, correr `make clean all`. Si se quisiera compilar el proyecto para que funcione con el buddy system compilar con el comando `make clean buddy`.
4. Por fuera del contenedor correr `./run.sh` sin ningún argumento extra, para poder finalmente ejecutar el trabajo.

   Con respecto a la corrección de la entrega parcial 1, agregamos los permisos del run.sh en la compilación del trabajo.

2. Decisiones Tomadas durante el Desarrollo y Limitaciones

Durante el desarrollo del trabajo, se han tomado decisiones acerca de las implementaciones de los requerimientos en las diferentes entregas. En conjunto a estas, también se han definido ciertas limitaciones que simplifican el trabajo de análisis de posibles casos.
A continuación se presentan los principales:

- No se hizo uso de una variable `current_process` ya que se toma como current el puntero al “inicio” de la lista o front. Con “inicio” nos referimos justamente al proceso que se está corriendo actualmente.
- En las correcciones nos han dicho de cambiar el scheduler, pero por el gran esfuerzo que requería y el poco tiempo que disponíamos se optó por dejarlo como está, es decir una lista circular de nodos que apuntan a la estructura del proceso, y la prioridad se maneja según cuántas instancias de este nodo hayan. La limitación que se decidió

poner es que como máximo un proceso puede tener prioridad 4, y como mínimo 1, así se reducen lo más posible las repeticiones.

- Como máximo hay dos procesos BLOCKED/KILLED/ASLEEP consecutivos. En el scheduler se chequean manualmente los casos en el que el proceso current no sea uno, y en el que, de no serlo el current, el próximo en la lista tampoco lo sea.
- Se decidió a último momento cambiar la implementación del driver de teclado para que en lugar de ser un array de char, sea de int. Esto nos permite un mejor manejo en el caso de querer enviar EOF. Esto no se propagó a la implementación de pipes ya que el EOF se maneja de otra forma.
- En relación al punto anterior, se decidió que los pipes haya a lo sumo un file descriptor de escritura y uno de lectura. Esto se debe a que los pipes se usan para comunicar manualmente dos procesos, por ende no se requiere la noción de cantidad de fd abiertos. Si el de escritura se cierra, entonces el pipe entiende que hay un EOF, y lee hasta que ya no haya más elementos dentro, devolviendo 0 en ese caso.

- Luego de varios intentos encontramos que nuestra implementación de phylos tiene race conditions que no pudimos resolver, lo que hace que a veces su comportamiento no sea el esperado.

3. Falsos Positivos y Warnings

Debido a la compilación del trabajo con el flag `-Wall` surge un warning debido a un casteo en el archivo videoDriver.c que se intentó resolver pero no se pudo. Dado que se debe a un código provisto por la cátedra se decidió no invertir más tiempo en arreglarlo.
Por otro lado, PVS-Studio arroja warnings con respecto al Bootloader. Al igual que en el caso anterior, el código es código provisto por la cátedra, por lo que modificarlo podría llegar a dañar el trabajo realizado.
Se listan a continuación falsos positivos que arroja PVS:

- naiveConsole.c : 11
  Debido a cómo se inicializa el array, PVS arroja warning. Debido a que es el comportamiento que se espera, inicializar el array con 0, no se tiene en cuenta el warning

- naiveConsole.c : 12 - naiveConsole.c : 13 - kernel.c : 33 - kernel.c : 34
  Arroja warning por un casteo de un int a puntero, se decide obviarlo ya que no es un casteo inseguro en el contexto que se trabaja.