Trabajo Práctico 1 - Grupo 7

(72.11) Sistemas Operativos

Integrantes

- Martina Schvartz Tallone - 62560
- Josefina Míguez - 63132

Aclaraciones:

Los tests en userland, al crearlos como procesos, no nos funcionaban por algo que no pudimos descifrar: cuando debuggueamos notamos que la dirección de la función test_processes era un número ridículamente alto, alejado del rango de alcance del Userland. Notamos que esto se debía a que estaban en archivos separados, ya que si lo probabamos con funciones dentro del mismo archivo en el cual creabamos el proceso, la dirección era lógica. En un principio nos funcionaba en Kernel, y luego de estar buscando el error por días, tomamos la decisión de entregarlo con lo que tenemos para que al menos se pueda evaluar nuestra creación de procesos y salto a userland desde Kernel.

Lamentamos la desprolijidad de la entrega e incumplimiento de los requerimientos de la misma, ya que el tiempo se nos hizo muy justo y no pudimos arreglar ni probar todo lo que nos quedó pendiente.

Decisiones tomadas durante el desarrollo:

Cada proceso está definido en un struct donde se almacena la información que creemos necesaria guardar: el nombre, el pid, el pid del proceso que lo creó, la prioridad, su estado (*ready*, *blocked*, *killed*), el *stack pointer* y una lista de sus hijos. En la creación de dicho proceso todos estas características se pasan por parámetros de la función *my_create_process* excepto el nombre. El mismo está dentro de *argv* el cual (junto con *argc*) son necesarias para el salto a la función en el momento que scheduler indique.

Almacenamos la información de todos los procesos que fueron creados en un vector de procesos e incluimos también una lista circular de los procesos que tienen estado *ready* los cuales esperan su turno para correr.

El *scheduler* es llamado en cada una de las interrupciones del *timer tick* y el mismo recibe el rsp del proceso que estaba corriendo para *back-upearlo* y agarrar el del proceso apuntado por la lista *readys*.

Instrucciones para compilar y ejecutar:

Correr el contenedor y posicionarse en el directorio `root`.

1. Posicionarse en la carpeta `Toolchain` y correr en la terminal del contenedor `make clean all`.
2. Dentro de `root` otra vez, correr `make clean all`.
3. Por fuera del contenedor correr `./run.sh` sin ningún argumento extra.
4. Dentro del qemu, ejecutar los comandos `testprocess` o `testprio` para testear la creación y prioridades de los procesos respectivamente.

Limitaciones

El sistema tiene un máximo de 200 procesos.