Taller de Drivers

Ejercicios

Sistemas Operativos

15 de mayo de 2024

Ejercicio 1: /dev/nulo

Se pide: **Implementar un módulo `/dev/nulo` que replique exactamente la funcionalidad de `/dev/null`**.

- Debe descartar toda la información que se escriba en él.
- No debe devolver ningún carácter cuando se intente leer.

Ejercicio 2: /dev/azar

Escribir un módulo para el dispositivo /dev/azar que debe comportarse de la siguiente manera:

- Cada vez que el usuario invoca a `write`, se debe interpretar el contenido del buffer como un string que representa un número entero, agregando un carácter de fin de string `'\0'`.
- Si la entrada no puede convertirse a entero, se debe fallar con `-EPERM`.
- Cada vez que el usuario realiza una lectura, se debe devolver una cadena con un número al azar entre 0 y el número escrito previamente, seguido por un carácter de fin de línea `'\n'`.
- Si no se ha escrito ningún número antes de leer, también debe fallar con `-EPERM`.

Funciones útiles:

- `copyfromuser / copytouser` (`<linux/uaccess.h>`)
- `kmalloc / kfree` (`<linux/slab.h>`)
- `kstrtoint` (`<linux/kernel.h>`)
- `get_random_bytes` (`<linux/random.h>`)
- `snprintf` (`<linux/kernel.h>`)

Ejercicio 3: /dev/letras123

Este módulo debe manejar hasta tres espacios de acceso, cuidando las condiciones de carrera.

- Cuando un usuario hace `open`, se le debe asignar un espacio disponible; si no hay espacios, debe fallar con `-EPERM`.
- Al hacer `close`, debe liberarse el espacio (si se había hecho `open` previamente).
- En el primer `write` del usuario (si tiene un espacio asignado), se debe guardar el primer carácter escrito. Los siguientes `write` deben ignorarse.
- Cada `read` debe devolver tantas copias del carácter guardado como se solicite. Si no se hizo `write` previamente, debe fallar con `-EPERM`.

Se recomienda el uso de:

- `semaphore` o `spinlock` para sincronización.
- El campo `private_data` de la estructura `file` para almacenar información del proceso.