SO2 - Screen Splasher - Documento Final

Víctor Expósito Griñán
Francisco Javier Gallardo Rodríguez

Nivel alcanzado

Cuando empezamos el proyecto, nos planteamos alcanzar el nivel 2, cosa que finalmente hemos cumplido. Para el tercer nivel nos ha faltado tiempo, así que decidimos no tocarlo y centrarnos en el segundo escalón para acabar todas las tareas pendientes.

Decisiones, modificaciones, problemas y soluciones

Una vez llegados a la entrega final de este proyecto, se puede observar que hemos hecho cambios respecto a la entrega anterior. Hemos mantenido la forma de almacenar información de tareas con mínimo una pantalla abierta, y los datos de cada una tampoco se han visto muy alterados más allá de la mencionada modificación que teníamos en mente respecto al cursor.

Para mostrar el cursor por pantalla, hemos optado (como ya dijimos en la entrega de seguimiento) por utilizar únicamente dos variables x, y para posicionarlo y printar caracteres. Sin embargo, tuvimos pequeños bugs, ya que se imprimían caracteres extraños en lugar de lo que queríamos. Esto se pudo solucionar aplicando una máscara 0x00FF a los datos en la posición del cursor, que elimina los bits indeseados. Tal y como está ahora, si el cursor se encuentra en una posición donde hay un caracter, éste lo resalta haciéndolo parpadear, mientras que si no hay nada escrito, muestra una barra baja en estado blinking.

Respecto a la gestión de páginas, hemos implementado el proceso de herencia de pantallas con fork(). Para ello hemos decidido hacer un pequeño cambio para que la estructura sea un poco más comprensible. Esta modificación consiste en utilizar sólo la función set_pag_screens en la inicialización del proceso init, y en la creación de una nueva rutina llamada set_ss_screenpag que se utiliza en el fork(), de la misma manera que se trata con las páginas de datos.

No todo nos salió bien, ni mucho menos. Tuvimos problemas a la hora de utilizar la llamada a sistema close() cuando se producía una herencia de pantallas. En concreto, nos provocaba un page fault. Esto se debía a que, para acceder a los datos de pantalla de procesos en la readyqueue, hacíamos un cambio de contexto y luego lo deshacíamos. Como aún no habían pasado suficientes ciclos para que se ejecutase el hijo, al hacer el cambio de contexto el hijo hacía el ret_from_fork y pasaba a tener un stack deformado.

En cuanto al cambio de foco de pantalla, hemos conseguido solucionar el inconveniente que teníamos con el Tab y Shift + Tab. La solución en realidad era un detalle que pasamos por alto, y es que confundimos el bit de Make con el bit de Break a la hora de interpretar el código de una tecla, lo que provocaba el comportamiento errático que describimos en el documento anterior. También, como método para detectar si hacemos un cambio de foco cuando no hay pantallas disponibles (por ejemplo al hacer boot del SO o cuando se cierran todas las pantallas de todos los procesos) hemos incluido una pequeña función que muestra por pantalla un mensaje cuando no haya pantallas disponibles.

Por último, y para acabar las funcionalidades que teníamos pendientes, conseguimos implementar con éxito el método de backspace y delete, así como mover el cursor a cualquier posición de la pantalla, lo cual nos permite printar caracteres en posiciones concretas del dispositivo.

Listado de Tareas Acabadas

- Estructuras y funciones de las pantallas virtuales
    - Crear lista de pantallas global (hecho, cambios en idea y/o estructura)
    - Crear lista de pantallas de proceso (hecho, cambios en idea y/o estructura)
    - Crear el struct de pantalla (datos de pantalla) (hecho)
    - Código para copiar de pantalla virtual a pantalla del sistema (hecho)
    - Añadir funcionalidades especiales de la pantalla
        - Crear y mover el cursor (hecho)
        - Cambio de color de letras y fondo (hecho)
        - Suprimir caracteres (hecho)
- Añadir syscalls
    - Añadir entradas a syscall table (hecho)
    - Escribir sys_createScreen (hecho)
    - Escribir sys_close (hecho)
    - Escribir sys_setFocus (hecho)
- Modificar syscalls
    - Modificar sys_write para escribir en pantalla virtual y detectar caracteres especiales (hecho)
    - Modificar sys_fork para heredar pantallas. (hecho)
    - Modificar sys_exit para cerrar todas las pantallas del proceso (hecho)
- Modificar interrupción de teclado:
    - Habilitar el cambio de pantallas virtuales con Tab y Shift + Tab (hecho)
- Ejecutar juego de pruebas (hecho)

Aclaraciones

Al revisar el código hemos visto que no es del todo eficiente a la hora de asignar las páginas físicas a las pantallas. Hemos hecho otra implementación del código donde la asignación de los frames se realiza en el sys_createScreen y se liberan en el sys_close y sys_exit. Creemos que es correcto, pero al no haber tenido tiempo a testearlo, hemos creído más conveniente dejarlo a parte. En la carpeta allocate_in_create hemos incluido las funciones que se deben modificar para conseguir esta implementación más eficiente.