

Objetivos

- Analizar diferentes mecanismos de **comunicación entre procesos**.
- Examinar la implementación de **drivers en Xinu**.
- Implementar funciones de alto nivel para un **controlador de periférico**.

Referencias

- [1] Tanenbaum, Bos – Modern Operating Systems - Prentice Hall; 4 edition (March 10, 2014) - ISBN-10: 013359162X
[2] Douglas Comer - Operating System Design - The Xinu Approach. CRC Press, 2015. ISBN : 9781498712439
[3] Silberschatz, Galvin, Gagne - Operating Systems Concepts - John Wiley & Sons; 10 edition (2018) – ISBN 978-1-119-32091-3

Software y Hardware

Linux y Xinu. La versión de Xinu que utilizamos es para arquitectura PC (x86). Ejecutamos el sistema operativo Xinu en una máquina virtual llamada QEMU, que emula una PC básica.

El trabajo puede realizarse sobre las máquinas de los laboratorios (RECOMENDADO).

Quienes tengan Linux en sus casas, podrían intentar instalar todo lo necesario y llevarlo a cabo ahí también. Una tercera posibilidad es el acceso remoto RDP comentado en la web de la materia.

Ejercicio 1. Implementar un programa en Linux que utilice **memoria compartida (comunicación entre procesos)**.

- Desarrollar un programa A que solicite al sistema operativo Linux una región de memoria compartida. Luego, el proceso A debe abrir el archivo
`/usr/share/doc/util-linux/source-code-management.txt`
(archivo de texto) , leer su contenido, y colocarlo en la región de memoria compartida creada.
- Desarrollar un segundo programa, B, que le solicite a Linux el acceso a la memoria compartida, y presente en pantalla el contenido de esa región (la cual será el contenido del archivo de texto).
- Utilice las funciones `open()`, `read()`, `close()` en a. para leer el contenido del archivo.

Documentación:

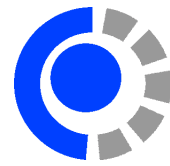
`man 2 open`

`man 2 read`

`man 2 close`

`man 3 shm_open`

`man 3 shm_unlink`



Ejercicio 2. Implementar un programa en Xinu que utilice *pasaje de mensajes (comunicación entre procesos)*.

Usted ha sido contratado por Shigeru Miyamoto para implementar en Xinu un video juego.

Él le explica que ha portado una versión de Galaga de la consola Game Boy Advance (GBA) a Xinu, y le solicita que lo termine.

Luego de una revisión usted detecta que el juego está lleno de bugs y hacks.

La paleta de colores es incorrecta, los disparos no siempre alcanzan al objetivo correctamente (detector de colisiones defectuoso), no existe puntuación, el código es horrible.

- a. Divida el videojuego en al menos 5 procesos:
 - Un proceso 1 lee el teclado.
 - Un proceso 2 mantiene la posición de la nave del jugador.
 - Un proceso 3 detecta colisiones.
 - Un proceso 4 actualiza la posición de las naves enemigas, y de los disparos.
 - Un proceso 5 que finaliza el juego.
- b. El proceso 1 lee el teclado y a través del mecanismo de pasaje de mensajes le indica al proceso 2 (que mantiene el estado del jugador), si debe moverse a la izquierda, a la derecha o disparar.
- c. El proceso 3 debe detectar colisión entre los disparos y naves enemigas. Si detecta una colisión, debe enviar un mensaje al proceso 4 (que mantiene el estado de las naves enemigas), para indicar que esa nave fue destruida y el disparo finalizó. Luego de verificar si hubieron colisiones, entonces envía un mensaje al proceso 3 para que simplemente actualice el estado de las nuevas posiciones de los disparos y naves enemigas.
Luego, este mismo proceso 3 debe detectar la colisión entre la nave del jugador y las naves enemigas. Si detecta una colisión, envía un mensaje al proceso que debe finalizar el juego (o restar vidas).
Luego de verificar si hubieron colisiones, envía un mensaje al proceso 4 para que simplemente actualice el estado de las nuevas posiciones de los disparos y naves enemigas.
- d. BONUS TRACK: agregar una puntuación al juego, o vidas.