Sistemas Operativos I 2022
Trabajo Práctico Obligatorio 1

Objetivos:

- Modificar, compilar y testear el sistema operativo Xinu.
- Crear procesos, finalizar procesos.
- Conocer los componentes de la tabla de procesos en Xinu.

La versión de Xinu que utilizaremos es para arquitectura PC (x86). Utilizaremos el sistema operativo Xinu utilizando 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.

Ejercicio 1

Descargue el código fuente de Xinu para PC, desde la web de la materia. El sistema operativo ya contiene, también, el código fuente del shell de Xinu.

Compilar y verificar el funcionamiento de Xinu utilizando las instrucciones en la web de la materia.

- ¿Qué componentes trae esta versión de Xinu?
- ¿Qué periféricos de PC son accesibles desde QEMU?
- ¿Cómo se accede al puerto serie de la PC en QEMU, el cual permite utilizar el shell?
- ¿Cuántos procesos existen en ejecución? ¿Cómo lo obtuvo?

Ejercicio 2

Modifique Xinu. Editar el archivo main.c y emita un mensaje a la pantalla gráfica a colores VGA de PC. Compilar y verificar.

Ejercicio 3. *Incorporación de un programa al shell de Xinu.*

Agregue un nuevo programa a Xinu. Deberá incorporarlo al shell de Xinu de la siguiente manera:

1. Escriba un programa hello world en un archivo .c debajo del directorio **`shell/`**. Atención, en Xinu la función principal de un programa no debe llamarse main(). Ejemplo:

```
#include <xinu.h>

mi_programa() {
        printf("Hola mundo! \n");
}
```

2. Incorpore su programa a la estructura que define los programas del shell, dentro del archivo **`shell/cmdtab.c`** y en **`include/shprototypes.h`**
3. Compilar y verificar que su programa se encuentra incorporado al sistema y puede ejecutarlo.
4. ¿Por qué en Xinu no se llama main() la función principal de cada nuevo programa?

Ejercicio 4. *Creación de procesos en Xinu.*

Incorpore a su programa anterior código para crear los siguientes dos procesos que se observan en el ejemplo:
https://github.com/zrafa/xinu-avr/blob/master/apps/example2/main.c

- ¿Qué sucede al ejecutar el programa?
- ¿Cómo se podría lograr que el sistema operativo conmute la CPU entre los dos procesos de manera más seguida?

Ejercicio 5. *Finalización de procesos en Xinu.*

Incorpore a main, luego de creado los procesos del Ejercicio 4, una espera de 10 segundos. Y que luego de la espera finalice los dos procesos iniciados anteriormente. (Ayuda: almacenar los PIDs de los procesos y averiguar cómo se llama la system call para finalizar procesos).

Ejercicio 6

Varios procesos reutilizando código ejecutable.
Incorpore a su programa código para crear los siguientes dos procesos que se observan en el siguiente ejemplo:
https://github.com/zrafa/xinu-avr/blob/master/apps/example3/main.c

Ejercicio 7

Utilice el ejemplo de la clase para crear un programa en Linux, que le pida al sistema operativo crear un nuevo proceso hijo.

Ejercicio 8

Douglas Comer en el libro que documenta la implementación de Xinu dice:

The name stands for Xinu Is Not Unix. As we will see, the internal structure of Xinu differs completely from the internal structure of Unix (or Linux). Xinu is smaller, more elegant, and easier to understand.

Ahora compare las funciones que se utilizan en Linux y en Xinu para pedirle luego al sistema crear procesos. ¿Cuál es más fácil en su opinión?, ¿las funciones para realizar system calls en Linux o en XInu? Si su respuesta fue Linux, entonces ¿por qué piensa que Douglas Comer diría que Xinu es más elegante y más fácil de entender?

Ejercicio 9

Observar el archivo **`include/process.h`**. Que campos contiene la tabla de procesos en Xinu. ¿Piensa que falta algo más que deba contemplar el sistema operativo para gestionar un proceso?