# Final Laboratorio SO2223q2

## Makefile (0.5 puntos)

Crea un **Makefile** que permita generar todos los programas del enunciado a la vez y cada uno de ellos por separado. Añade una regla (clean) para borrar todos los binarios y/o ficheros objeto, y dejar sólo los ficheros fuente. Los programas deben generarse si, y solo si, ha habido cambios en los ficheros fuente.

## Control de errores (0.5 puntos)

Para todos los programas que se piden a continuación deben comprobarse los errores de TODAS las llamadas a sistema (excepto el write por pantalla), controlar los argumentos de entrada y definir la función Usage().

## Ejercicio 1: Ficheros (1,5 puntos):

Escribe un programa (proc_time.c) que reciba, como primer y único argumento, un PID y escriba por la salida estándar la cantidad de tiempo que el proceso con ese PID ha estado en ejecución. Para hacerlo, el programa debe abrir el fichero /proc/PID/stat en modo lectura, leer (carácter a carácter) los valores número 14 y 15 del fichero y sumarlos (ver man proc). Si el fichero no existe (no existe un proceso con ese PID) el programa debe devolver el valor 255 en el "exit()". Junto a este enunciado se entrega un ejecutable de este programa (proc_time_ok) para que puedas utilizarlo en los siguientes apartados si no te funciona.

## Ejercicio 2: Procesos (2 puntos)

Escribe un programa (nproc_time.c) que reciba una lista de PIDs como argumento y escriba por la salida estándar el tiempo que ha estado en ejecución cada uno de ellos, en el mismo orden en que se han recibido los PID. Para obtener el tiempo debe utilizarse el programa anterior (proc_time). Si alguno de los PID no existen (el valor de finalización es 255), el programa debe acabar con un mensaje de error.

## Ejercicio 3: Pipes con nombre (2 puntos)

Modifica el programa anterior (llámalo nproc_time_max.c) para que ahora, en vez de mostrarse el tiempo de cada proceso, se muestre únicamente el tiempo del proceso que ha estado más tiempo en ejecución. En caso de empate, se muestra cualquiera de ellos. Para obtener el tiempo de cada proceso (para calcular el máximo) debe utilizarse una única pipe CON nombre, donde cada hijo escribirá el tiempo de ejecución. Si la pipe no existe el programa debe crearla. Junto a este enunciado se entrega un ejecutable de este programa (nproc_time_max_ok) para que puedas utilizarlo en el siguiente apartado si no te funciona.

## Ejercicio 4: Signals (1,5 puntos)

Escribe un programa (max_5sec.c) que, dada una lista de PIDs, muestre cada 5 segundos el tiempo del proceso que ha estado más tiempo en ejecución. Para hacerlo debe utilizar el programa anterior (nproc_time_max) e implementar una espera bloqueante. Si recibe un SIGUSR1 el proceso debe mostrar el número total de segundos que lleva en ejecución y acabar.

## Ejercicio 5: Pipes sin nombre y memoria (2 puntos)

Modifica el programa del Ejercicio 3 (llámalo nproc_time_max2.c) para que la creación de los procesos sea concurrente. Ahora para comunicar los procesos debes utilizar pipes SIN nombre. Concretamente, debes utilizar una pipe sin nombre diferente para cada hijo. Para guardar los

canales necesarios para acceder a cada pipe debes utilizar memoria dinámica y liberarla antes de acabar.

## Qué hay que hacer

- El Makefile
- Los códigos de los programas en C
- La función Usage() para cada programa

## Qué se valora

- Que sigas las especificaciones del enunciado
- Que el uso de las llamadas al sistema sea el correcto
- Que se comprueben los errores de **todas** las llamadas al sistema
- Que el código sea claro y correctamente indentado
- Que el Makefile tenga bien definidas las dependencias y objetivos
- Que la función Usage() muestre por pantalla como debe invocarse correctamente el programa en el caso que los argumentos recibidos no sean los adecuados

## Qué hay que entregar

Un único fichero tar.gz con el código de todos los programas y el Makefile:

```
tar zcvf finallab.tar.gz Makefile *.c
```