Sistema de ficheros
3ª entrega (final)

niveles 11 a 13 (y anteriores)

En esta tercera entrega se ejecutarán prioritariamente la **simulacion** y **verificación** de las 50 escrituras de 100 procesos, pero se ha de enviar también todo lo anterior para revisar y almacenar como prueba de evaluación de la práctica completa.

Se entregará **un solo archivo,** comprimido con **tar.gz** o **tar.xz** que es un **directorio** que llevará por nombre el primer apellido de cada uno de los miembros del grupo, ordenados alfabéticamente y separados por un guión, por ejemplo: "Arenas_Fornes_Martínez" y lo entregará cualquiera de ellos.

El directorio contendrá:

- El programa ficticio **leer_sf.c** para mostar el **superbloque**.
- El programa **mi_mkfs.c** para inicializar el sistema de ficheros.
- Los siguientes ficheros de funciones de la **librería del sistema de ficheros** (y sus cabeceras correspondientes **.h**): **bloques.c**, **ficheros_basico.c**, **ficheros.c** y **directorios.c**
- El **semáforo** **semaforo_mutex_posic.c** y su cabecera.
- Los programas correspondientes a los **comandos** implementados (como mínimo **mi_mkdir.c**, **mi_ls.c**, **mi_chmod.c**, **mi_stat.c**, **mi_escribir.c**, **mi_cat.c**, **mi_link.c**, y **mi_rm.c**)
- Los programas ficticios **leer.c**, **escribir.c**, **permitir.c**, **truncar.c**
- Los programas correspondientes a la simulación y verificación: **simulacion.c** y **verificacion.c** (y sus cabeceras correspondientes **.h** si las habéis implementado).
- El **makefile** necesario para compilar todos los programas [1]
- El **scripte1.sh** (con los programas ficticios **escribir.c**, **leer.c**, **truncar.c** y **permitir.c**), el **scripte2.sh** adaptado a vuestros comandos [2] y el **scripte3.sh**
- El fichero de **texto** **texto2.txt**.
- Un fichero **README.txt** con las observaciones que consideréis (mejoras realizadas, restricciones del programa, sintaxis específica, y los nombres de los miembros del grupo).

En todos los programas se ha de poner el nombre de los autores en el código. Si algún componente del grupo no ha participado realmente en la implementación de la práctica NO SE HA DE INCLUIR SU NOMBRE y se me ha de notificar **ANTES de la entrega** para escindir el grupo y que así no pueda descargarse él la práctica del resto del grupo desde el aula digital.

En caso de algún funcionamiento incorrecto se os puede solicitar visualizar las impresiones de los tests de cualquiera de los niveles anteriores. Los mensajes de

---

[1] Ha de contener esta regla para limpiar:

```
.PHONY: clean
clean:
	rm -rf *.o *~ $(PROGRAMS) disco* ext* res*
```

[2] El script ha de acabar haciendo un make clean para limpiar los objetos, disco virtual y fichero externo de redireccionamiento del cat

depuración no teneís que haberlos eliminado, tan sólo deben estar comentados o con visualización sujeta a alguna variable de depuración, como por ejemplo:

```
#define DEBUG 0 //a 1 para mostrar los mensajes de depuración
...
#if DEBUG
  fprintf (stderr, "[buscar_entrada()→ inicial: %s, final: %s, reservar: %d]\n", inicial, final, reservar);
#endif
```

La evaluación final **cualitativa** de la práctica será **síncrona** y han de asistir **TODOS** los miembros de cada grupo. Se realizará en el aula informática con los grupos que se puedan evaluar en tiempo de clase o por Discord para el resto (se creará una tabla con mi disponibilidad horaria para elegir).
Una vez revisada conjuntamente con vosotros la ejecución de la práctica, procederé posteriormente de forma **asíncrona** a examinar el código para realizar la evaluación final **cuantitativa** y la detección de posibles **fraudes académicos**.

Se os recuerda que, de acuerdo con el artículo 37 del Reglamento Académico, la realización que se demuestre fraudulenta de alguno de los elementos de evaluación incluidos en las guías docentes de las asignaturas puede conllevar, a criterio del profesorado, la calificación final de «suspenso 0» de la asignatura. La existencia de un fraude también puede ser motivo de apertura de un expediente disciplinario contra el estudiante infractor.

Para **superar** los niveles 11 a 13, deberá funcionar correctamente:

- La **escritura** de los 50 registros en números de registro aleatorios[3] (salvo excepcional coincidencia de posición aleatoria) siendo el offset = nRegistro_aleatorio * sizeof(registro), para cada proceso.
  - Podéis comprobad para vosotros que os escribe cada registro leyéndolo justo a continuacion para ese mismo offset (previa inicializacion del buffer de lectura a 0s).
- La **verificación** de las escrituras mostrando el informe con los 4 registros más significativos de cada proceso.
  - Hay que comprobar que las lee todas (50 para cada proceso, salvo coincidencia de posición aleatoria).
- El **tamaño en bytes logico del informe** generado ha de ser el mismo que el del **fichero exterior** al que direccionemos el resultado de la verificacion (utilizad la salida de errores estándard para mostrar el tamaño del informe, así como para todos los mensajes de pantalla).
- El **informe** no ha de contener **basura.**

Además se **valorarán** los siguientes aspectos:

[3] Limitado a REGMAX=500.000

- Granularidad de las secciones críticas especialmente para la lectura y escritura: en **mi_read_f()** y **mi_write_f()** que controle la lectura y escritura del inodo, la actualización de sus campos, y la reserva de bloques, (en vez de serializar mi_read() y mi_write()), lo cual implica controlar el código reentrante.
- En la verificación, explorar en un buffer de memoria varios registros, en vez de leer las escrituras una a una accediendo cada vez al dispositivo virtual.
- Tiempo de ejecucion de la simulacion y de la verificacion (inferior a 1' en cada una).
- Uso de fechas con definición de microsegundos (con el struct timeval).

Ejemplo de test manual (ver ejecución en niveles 12 y 13):

```
$ rm disco #si ya lo teníais creado
$ make
$ ./mi_mkfs disco 100000
$ time ./simulacion disco  #mostrar dir_simulacion
$ time ./verificacion disco dir_simulacion
$ ./mi_cat disco dir_simulacion/informe.txt > res.txt
$ ls -l res.txt
$ cat res.txt
$ ./leer_sf disco
```

Si queréis realizar un testeo automatizado, podéis utilizar el **scripte3.sh** (tiempo real de ejecución aproximadamente 20").

Observaciones lógicas:

- Siempre se ha de cumplir que la **fecha de la primera escritura** ha de ser inferior a la de la **última**.
- La de **mayor posición** ha de tener el nº de registro mayor de las 4 escrituras mostradas para ese proceso, y la de **menor posición** el menor valor, y ambas han de estar comprendidas en el rango [0, REGMAX-1]
- La de mayor posición no tiene porqué coincidir con la posición de la última escritura, ni la de menor posición con la posición de la primera escritura, ya que las posiciones se calculan al azar.
- El **nº de escritura** tiene que estar compendido entre 1 y 50.
- Entre la **1ª y la última escritura** de un mismo proceso puede haber una diferencia de 2 a 3 segundos.