NIA: .....
Nombre y apellidos: .....

INSTRUCCIONES A SEGUIR:

- Es obligatorio seguir todas las siguientes instrucciones de realización del examen.
- Recuerde que ha de responder a lo pedido, de forma correcta, en el tiempo dado y en el espacio indicado. Evite ponerse nervioso/a.
- **No** desgrape el examen.
- Es obligatorio la entrega del examen.
-
- Indique su nombre, NIA y grupo en la primera hoja.
- Será necesario identificarse (DNI o carnet universitario) en todo momento.
-
- **No** se pueden utilizar: libros, apuntes, calculadora, móvil (o similares dispositivos).
- Todo aparato electrónico debe estar apagado y guardado.
- Los apuntes y libros deben estar en la mochila y fuera de alcance del estudiante.
- No se permiten estuches ni similares encima de la mesa.
-
- Para la realización del presente examen se dispondrá de **120 minutos.**
- Aunque las pruebas evaluables pueden calificarse de forma acumulativa (puntuar por apartados), recuerde que un error grave puede suponer un cero en su valoración total.
- El planteamiento solo de un problema no se tomará como solución válida.
- No se volverá a dar hoja de respuesta si no es por causa justificada.

NIA: .....
Nombre y apellidos: .....

Ejercicio 2 (2,5 puntos)

En la empresa en la que estamos trabajando disponemos de un sistema operativo básico que usa un *kernel* monolítico con un planificador *Round-Robin* (RR) para todos los procesos (de forma similar al introducido en la asignatura).
Esta empresa quiere que añadamos la siguiente funcionalidad: disponer una protección de denegación de servicio por parte de una aplicación que realiza una gran cantidad de llamadas al sistema (penalización por abuso de llamadas al sistema).

Para ello se pide indicar el diseño e implementación en pseudocódigo de los cambios que hay que hacer en el sistema operativo inicial para añadir la funcionalidad de la siguiente forma:

a) Disponer de una nueva llamada al sistema cuya declaración es:
   `void setThreshold_maxSyscalls ( long newMax )` ;
   que permite indicar el número máximo de llamadas al sistema en una rodaja de ejecución dada por el planificador (para todos los procesos).
   Si **`newMax`** indica un número negativo el *kernel* ignorará esta llamada.
   Si **`newMax`** es cero, se desactivará esta funcionalidad para todos los procesos, hasta que con una nueva llamada con un parámetro mayor que cero la vuelva a activar.
   Por defecto la funcionalidad está desactivada.
b) Disponer de un planificador actualizado que, si al final de la rodaja detecta que se supera el número máximo de llamadas al sistema indicado para la rodaja asignada por el planificador, deje 'dormido' el proceso durante 10 rodajas de reloj como castigo (y como forma de evitar una denegación de servicio para ese proceso).

NIA: .....
Nombre y apellidos: .....

Ejercicio 3 (2 puntos)

Una compañía dispone de un sistema operativo básico que usa un *kernel* monolítico con un driver para teclado que usa E/S programada (*polling*) para interactuar con el hardware (de forma similar a lo introducido en la asignatura).
La compañía desea disponer de una versión del driver de teclado usando interrupciones, por lo que la compañía nos contrata para hacer el diseño e implementación en pseudocódigo.
Buscando en documentación de otros drivers (en la web ficticia desbordamientoPila.es donde pueden ser incompletos), se han conseguido rescatar estos fragmentos de código para poder reusarse en el driver de teclado:

Id. Frag.	Fragmento de Código

1	• Proc = ObtenerPrimerProceso (Teclado.Bloqueados)
	• Si Proc != NULL
	  o Proc->estado = LISTO
	  o InsertarAlFinal(Listos, Proc)

2	• Insertar_proceso_actual(Teclado.Bloqueados) • procesoActual->estado = BLOQUEADO • procesoAnterior = procesoActual • procesoActual = extraerPrimero(listaProcesosListos) • procesoActual->estado = EJECUTANDO • procesoActual->rodaja = RODAJA • cambioContexto(procesoAnterior, procesoActual)
3	• Insertar_Interrupcion_Software(Despertar_bloqueados_teclado) • Generar_Interrupcion_Software()

Se pide que responda correctamente a los siguientes puntos en el espacio dado:

a) ¿Tiene alguna ventaja realizar la entrada/salida usando un diseño basado en interrupciones en lugar de E/S programada?
b) Describa brevemente para qué sirve los fragmentos de código recuperado (contestar en la columna descripción de la tabla asociada para la contestación).
c) Complete la funcionalidad del driver de teclado de la siguiente tabla. Los fragmentos de código anterior que use indíquelos por favor de la forma 'fragmento X' siendo X el índice de la tabla del enunciado

NIA: .....
Nombre y apellidos: .....

Ejercicio 4 (2 puntos)

Disponemos de una maquina monoprocesador de 32 bits y queremos implementar un sistema de ficheros para un sistema operativo UNIX con un *kernel* monolítico no expulsivo y con una caché de bloques que permite leer (bread) y escribir (bwrite) bloques de disco.

El sistema de ficheros a implementar tiene la siguiente estructura en disco:

1 bloque	**2** bloques	**n** bloques
Superbloque	Bloques con inodos	Bloques de datos
0	1	3 N

Donde los requisitos del sistema de ficheros incluyen:

1. El tamaño de bloque es de 4096 bytes (4 KiB).
2. Hay 50 bloques en total en el sistema de ficheros.
3. Solo hay ficheros (que pertenecen a un directorio raíz común para todos)
4. Hay como máximo 10 ficheros.
5. Cada fichero tendrá **dos** bloques de datos asociados.
6. Cada i-nodo ocupa 128 bytes y contiene, entre otras cosas, el nombre del fichero de un máximo de 12 caracteres y un carácter que con valor '0' indica que está libre y con el valor '1' indica que ese inodo está en uso.
7. El superbloque ocupará un bloque de disco y será el bloque inicial (bloque 0).
8. El superbloque contendrá la gestión de bloques libres/ocupados.
   Tiene un vector de 50 caracteres donde '0' indica que el bloque asociado a esa posición del vector está libre y '1' que está ocupado.
9. Cada fichero tiene su puntero de lectura y escritura (no compartido) donde varios procesos (máximo 100) pueden tener el fichero abierto simultáneamente y sus punteros de lectura/escritura pueden estar en posiciones diferentes.

Se pide:

a) Complete el diseño de las estructuras de disco dadas indicando los campos que tendrá el superbloque y un i-nodo de este sistema de ficheros.
b) Diseñar las estructuras en memoria que permitan trabajar con el sistema de ficheros anteriormente descrito.
c) Indicar en pseudocódigo la función de tratamiento de bloques bmap.
d) Indicar en pseudocódigo las funciones de tratamiento de i-nodos: ialloc, ifree y namei.
e) ¿Estima algún problema en el diseño dado respecto al aprovechamiento de espacio?

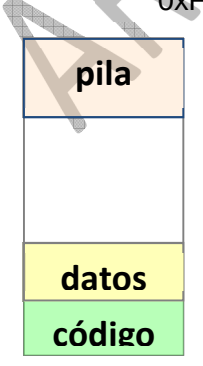
NIA: .....
Nombre y apellidos: .....

Ejercicio 5 (2 puntos)

Se dispone del siguiente código fuente de un programa:

```
pthread_t   thid;

void *task1 ( void * arg )
{
    arg = malloc (100*sizeof(int)) ;
    /* punto B */
}

main (int argc, char **argv)
{
        int *pint;
        int   pid, i;

         /* punto A */
        pthread_create(&thid,NULL,task1,(void *)pint) ;
        pint = malloc (100*sizeof(int)) ;
        pthread_join(&thid,NULL) ;

         pid =  fork() ;
         if (0 == pid) {
             free(pint) ;
             /* punto C */
             exit(0) ;
         }
         /* punto D */
         waitpid(pid, NULL, 0);
 }
```

Si en el punto A de ejecución, la imagen del proceso asociado se representa:



NIA: .....
Nombre y apellidos: .....

Se pide:

a) Utilizando la representación dada, dibuje la imagen de los procesos en el punto **B.**
b) Utilizando la representación dada, dibuje la imagen de los procesos en el punto **C.**
c) Utilizando la representación dada, dibuje la imagen de los procesos en el punto **D.**
d) ¿Considera que hay algún fallo en el uso de las llamadas de gestión de memoria en el fragmento de programa dado? Razone brevemente su respuesta.

NOTA: considere que la gestión de memoria **no** usa COW (*Copy-On-Write*)