Nombre:
DNI:

Primer control de laboratorio

Crea un fichero que se llame "respuestas.txt" donde escribirás las respuestas para los apartados de los ejercicios del control. Indica para cada respuesta, el número de ejercicio y el numero de apartado (por ejemplo, 1.a).

Justifica brevemente todas tus respuestas. Una respuesta sin justificar se considerará como no contestada.

Importante: para cada uno de los ejercicios tienes que partir de la versión de Zeos original que te hemos suministrado.

1. *(3 puntos + 1 punto) Mecanismos Entrada al Sistema*

Queremos añadir una nueva funcionalidad que nos permita mostrar un mensaje en una posición determinada de la pantalla. La interfaz de la nueva llamada es:

*int write_at(int col, int fil, char *buffer, int size)*

Que mostrará *size* carácteres de la cadena de carácteres *buffer*, a partir de la columna *col* y la fila *fil*. Esta función devuelve el número de bytes mostrados o -1 en caso de error.

No debes modificar el código del mecanismo habitual de llamadas a sistema, pero la solución propuesta tiene que permitir añadir nuevas funcionalidades de forma fácil (igual que ahora). Tienes que usar la interrupción 0x81 para generar esta nueva llamada a sistema.

a) Indica qué estructura de Zeos tienes que modificar para implementar esta nueva funcionalidad.
b) Indica qué nuevas estructuras de Zeos tienes que añadir.
c) Escribe el código para habilitar esta nueva funcionalidad.
d) Escribe el código del wrapper para esta funcionalidad.
e) Escribe el código del handler para esta funcionalidad.
f) (Opcional) Implementa en Zeos esta funcionalidad.

2. *(5 puntos) Gestión de procesos y memoria*

Estamos pensando en añadir a ZeOS la llamada a sistema *exec* (que NO tienes que implementar) con el siguiente interfaz:

*int exec (char *nombre_ejecutable)*

Esta nueva llamada a sistema recibe como parámetro el nombre de un ejecutable y se encarga de mutar a este nuevo programa. Es decir, liberar todos los frames que el proceso tenía asignado, crear un nuevo espacio de direcciones en el que se reservarán los suficientes frames para cargar el nuevo espacio de direcciones y pasar a ejecutar la primera instrucción del nuevo código.

Vamos a suponer que todo el código se situa de manera consecutiva en memoria lógica, que toda la región de datos también, pero ambas regiones no tienen por qué estar consecutivas (es decir, entre la zona de código y la zona de datos podemos tener un conjunto de direcciones que sea inválido para el proceso).

Para dar soporte a esta nueva funcionalidad:

- **los procesos NO compartirán la zona de memoria del código**.
- las zonas de código y datos pueden tener tamaños variables.
- y las direcciones iniciales de estas zonas no estan fijas.

Suponiendo que dentro de ZeOS tenemos ya implementadas las siguientes funciones, que sirven para interpretar el contenido del ejecutable:

- `int get_code_size(char *nombre_ejecutable)`: dado el nombre de un ejecutable devuelve el tamaño en bytes que ocupa su código.
- `int get_data_size (char * nombre_ejecutable)`: dado el nombre de un ejecutable devuelve el tamaño en bytes que hay que reservar para la zona de datos y pila de usuario.
- `int *get_first_instruction_address(char *nombre_ejecutable)`: devuelve la dirección lógica de la primera instrucción del ejecutable.
- `int *get_data_initial_address(char *nombre_ejecutable)`: devuelve la dirección lógica donde empezará la zona de datos.

Contesta a las siguientes preguntas:

a) Indica los cambios necesarios en el PCB para soportar esta nueva funcionalidad.
b) Indica los cambios necesarios durante la inicialización del sistema.
c) Indica los cambios necesarios en el fork para la herencia de datos de sistema.
d) Indica como calcular el número de páginas usadas por la zona de código del proceso actual (N).
e) Indica como calcular el número de páginas usadas por la zona de datos del proceso actual (M).
f) Indica donde colocar la dirección base de la pila de usuario y como calcularla.
g) Indica cómo obtener una página no usada del espacio de direcciones de este proceso (LIBRE).
h) (1 punto) Indica el código necesario en el fork para la herencia de la zona de código de usuario sin hacer comprobación de errores y suponiendo que tienes 1 única página libre en el espacio de direcciones.
i) Indica los cambios necesarios en el cambio de contexto.
j) Indica los cambios necesarios en el exit.

3. *(2 puntos) Cambio de contexto*

Supón que se modifica el código del cambio de contexto justo después de cambiar a la pila del nuevo proceso, para que haga lo siguiente:

```
asm volatile(
        "popl ebx\n\t"
        "popl ebp\n\t"
        "ret\n\t" );
```

a) Sin hacer ningún otro cambio, esto es incorrecto, ¿por qué?
b) ¿Cómo puedes modificar la rutina de cambio de contexto para que funcione con este cambio?
c) ¿Hay que modificar alguna otra cosa a parte de esta rutina?