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. (6 puntos + 1 punto de implementación) ZZzzzz...

Queremos implementar una nueva llamada a sistema:

int sleep(int seconds)

que permita *dormir* un proceso durante *seconds* segundos. Esta llamada es bloqueante y, por lo tanto, durante este tiempo el proceso no se ejecuta. Al finalizar este tiempo, el proceso se despierta y puede volver a ejecutarse. Esta función retorna 0 si se despierta normalmente o un número negativo en caso de error (EINVAL, si el parametro es <0 o EINTR, si se despierta antes de tiempo).

Tambien queremos implementar otra llamada a sistema:

int wakeup(int pid, int NOW)

que permita *despertar* a un proceso dormido, es decir, permite al proceso volver a ejecutarse. Esta función tiene un parámetro (*NOW*) que, con un valor 1, indica que se tiene que pasar a ejecutar el proceso dormido inmediatamente. Con valores distintos, el proceso debe respeter el orden de otros procesos que esten pendientes de ejecución.
Esta función retorna 0 si ha podido despertar al proceso correctamente o un número negativo en caso de error (EINVAL, algun parámetro no es valido o EEXIST, si el proceso no estaba dormido).
Para implementar estas nuevas funcionalidades del sistema no se debe usar el mecanismo habitual de llamadas a sistema. Debes usar las interrupciones 0x60 y 0x61 para llamar de forma directa a las funciones de sistema *sleep* y *wakeup* respectivamente.
Para controlar el tiempo, puedes tener en cuenta que el reloj está programado a 18Hz (o sea, que genera 18 interrupciones por segundo).
Aunque actualmente haya una limitación de 10 procesos, la solución propuesta tiene que poder tratar de forma eficiente con miles de procesos.

a) Indica qué estructuras de Zeos tienes que modificar para implementar estas nuevas funcionalidades.
b) Indica qué nuevas estructuras de Zeos tienes que añadir.
c) Escribe el código para habilitar estas nuevas funcionalidades.
d) Indica el código del wrapper de la llamada al sistema sleep.
e) Indica el código del wrapper de la llamada al sistema wakeup.
f) Indica el código del handler para la llamada al sistema wakeup.

Nombre:
DNI:

g) ¿Donde puedes detectar que hay que despertar a un proceso?
h) ¿Cuando y como lo detectas?
i) Indica el código necesario en este caso para despertar al proceso.
j) Indica el código de la llamada sistema que tienes que realizar en el caso de NOW=1. Puedes suponer que tienes implementada la siguiente función: *struct task_struct * find_task_by_pid(int pid)* que dado un pid te devuelve su PCB (o NULL) si no existe.
k) (Opcional) Implementa en Zeos los cambios propuestos en los apartados anteriores.

2. *(2 puntos)* Gestión de procesos

Queremos modificar el punto de retorno de los nuevos procesos creados en la llamada a sistema fork. Concretamente queremos que los nuevos procesos empiecen su ejecución en la rutina *ret_from_fork2*, definida en ensamblador como:

```
ENTRY(ret_from_fork2)
        movl $0, %eax
        ret
```

a) Indica los cambios necesarios en el sys_fork.
b) Indica los cambios necesarios en el task_switch.
c) Indica los cambios necesarios en la rutina init_idle.
d) ¿Qué opción es mejor, la antigua o esta nueva?

3. *(2 puntos)* Gestión de memoria

Si el espacio lógico de un proceso en modo usuario ocupa la región de memoria 0x150000-0x170000:

a) ¿Cuantas páginas usa esta zona?
b) (1.5 puntos) Suponiendo que tienes un vector *frames* con las páginas físicas necesarias, indica como seria el código de la llamada a sistema fork para copiar toda esta zona del padre al hijo.