SOA (18/01/2022)

Final 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, 1.b, ...).

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

*1. (6 puntos + 0,5 puntos) **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. Esta función tiene un parámetro (*NOW*) que, con un valor 1, indica que se tiene que pasar a ejecutar el proceso dormido inmediatamente. Si NOW es diferente de 1, el proceso debe respetar el orden de otros procesos que estén pendientes de ejecución.

Esta función retorna 0 si ha podido despertar al proceso correctamente o un número negativo en caso de error (EINVAL, algún parámetro no es válido 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 entradas 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) Implementa el wrapper de la llamada al sistema *sleep*.
e) Implementa el wrapper de la llamada al sistema *wakeup*.
f) Implementa el handler para la llamada al sistema *wakeup*.
g) ¿Dónde puedes detectar que hay que despertar a un proceso? Escribe el pseudocódigo para detectar procesos dormidos que hay que despertar.
h) Indica el código de la llamada sistema *wakeup* para 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 un puntero a su PCB (o NULL) si no existe.
i) Explica como detecta la llamada a sistema *sys_sleep* que debe devolver el valor –EINTR.
j) ***(0.5 puntos)*** Implementa correctamente estas funcionalidades en Zeos.

2. (1 punto) Memoria

a) ***(0.75 puntos)*** Modifica el código de sys_fork para que la copia de los datos y pila del proceso padre al hijo se realice a partir de la dirección lógica 0x120000 utilizando una única página lógica.

b) ***(0.25 puntos)*** ¿Qué inconveniente tiene el sistema visto en clase en el que se usan tantas páginas extra como ocupa la zona de datos comparado con el sistema del apartado anterior?

3. (3 puntos + 0,5puntos) Memoria Dinámica

Decidimos añadir un par de llamadas a sistema nuevas:

*void * malloc4K(void);*
int free4K(void buff);*

malloc4K reserva una región de 4096 bytes en memoria de usuario (coge un frame físico libre y lo mapea en algun lugar vacio del espacio lógico del proceso) devolviendo la dirección usada. *free4K* libera una de estas regiones. El parámetro *buff* de *free4K* corresponde a un puntero devuelto por la función *malloc4K* con anterioridad. Estos bloques de memoria **sólamente serán válidos para el proceso que los crea**.
Se valorará la eficiencia y la eficacia de la solución, y que la solución propuesta sea escalable a sistemas con muchos más procesos.
Contesta a las siguientes preguntas:

a) ¿Qué estructuras nuevas se tienen que añadir al sistema? Indica también su utilidad.
b) ¿Qué estructuras actuales del sistema operativo se tienen que modificar?
c) Escribe el pseudocódigo de la función *busca_espacio_libre_user*() que retorna la dirección inicial de una zona libre del espacio de direcciones de usuario donde guardar un bloque de memoria.
d) Implementa la función *sys_malloc4K*.
e) Implementa la función *sys_free4K*.

f) Comenta si habría que modificar alguna/s rutina/s de sistema adicional/es y el motivo sin entrar en demasiados detalles.
g) ***(0.5 puntos)*** Implementa correctamente estas funcionalidades en Zeos.

4. (1 punto) Generic Competences Third Language (Development Level: mid)

The following paragraph belongs to the book *Understanding the Linux Kernel* by D. Bovet and M. Cesati:

"The translation of linear addresses is accomplished in two steps, each based on a type of translation table. The first translation table is called the Page Directory, and the second is called the Page Table.
The aim of this two-level scheme is to reduce the amount of RAM required per-process Page Tables. If a simple one-level Page Table was used, then it would require up to 2^20 entries (i.e. at 4 bytes per entry, 4MB of RAM) to represent the Page Table for each process (if the process used a full 4GB linear address space), even though a process does not use all addresses in that range. The two-level scheme reduces the memory by requiring Page Tables only for those virtual memory regions actually used by a process"

Create a text file named "generic.txt" and answer the following questions (since this competence is about text understanding, you can answer in whatever language you like):

a) What is the name of the second translation table used for translating linear addresses?
b) Is it usual that a process uses 4GB of linear address space?
c) Why a two-level linear address translation reduces the amount of memory required to store all the translations?

5. Entrega

Sube al Racó los ficheros "respuestas.txt" y "generic.txt" junto con el código que hayas creado en cada ejercicio.

Para entregar el código utiliza:

```
> tar zcfv examen.tar.gz respuestas.txt generic.txt ejercicio1
ejercicio2 ejercicio3
```