SOA (20/10/2015)

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).
Todas las respuestas se tienen que justificar brevemente. Una respuesta sin justificar se dará como no contestada.

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

Queremos implementar completamente, en Zeos, **la jerarquía de procesos**. Para ello, se tienen que añadir las llamadas al sistema *exit* y *wait*

```
int exit(int status);
```

El proceso pasa a un nuevo estado llamado Zombie (sin liberar su PCB) y guarda en su PCB el valor de status para que su proceso padre lo pueda consultar.
Si el proceso tiene algún proceso hijo (en cualquier estado), éste pasa a ser hijo del proceso Init.

```
int wait(int *status);
```

Consulta el estado de finalización de los procesos hijos. Dependiendo de éstos, el funcionamiento de wait es:
-Si no tiene procesos hijos, wait devuelve error (ECHILD)
-Si tiene procesos hijos, pero ninguno de ellos está en el estado de Zombie, es bloqueante hasta que alguno de ellos acabe.
-Si tiene un proceso hijo en estado de Zombie, devuelve su PID (valor de retorno) junto con su valor de status, y libera el PCB de ese proceso.
-Si tiene más de un proceso hijo, selecciona el PCB de cualquiera, solo uno, de ellos.

1. (4 puntos + 0,5 puntos de implementación) Llamadas al sistema

Como es un soporte experimental, se habilitará una nuevo punto de entrada al sistema (0x83) donde colgaremos las versiones experimentales de las llamadas al sistema. Es decir, crearemos una réplica de fork (llamada sys_exp_fork), de exit (sys_exp_exit) y añadiremos wait (sys_exp_wait) con los números de servicio, respectivamente, 3, 4 y 5. De todas formas, desde el código de usuario, lo único que se verá es que se ha añadido la llamada al sistema wait que, por ahora, devolverá siempre -1. Del resto (fork y exit) redireccionaremos los wrappers.

a) Indica qué estructuras de Zeos tienes que modificar para implementar este nuevo punto de entrada al sistema.
b) Indica qué nuevas estructuras de Zeos tienes que añadir
c) Escribe el código para habilitar la nueva entrada de la IDT
d) Indica cómo se modifican los wrappers para las llamadas al sistema fork y exit.
e) Indica el código del wrapper de la llamada al sistema wait

f) Indica el código del nuevo handler para las llamadas al sistema fork, wait y exit
g) ¿Se tiene que hacer alguna modificación en el código de usuario para que utilice la nueva implementación propuesta?
h) Implementa en Zeos los cambios propuestos en los apartados anteriores

2. (6 puntos + 0,5 puntos de implementación) Creación de procesos

Es momento de implementar el funcionamiento de estas llamadas al sistema:

A tener en cuenta:

- Aunque ZeOS solamente permite tener 10 procesos vivos a la vez, se tiene que implementar el mecanismo pensando que pueden haber miles de ellos en el sistema.
- El mecanismo de saber/encontrar procesos hijos dado un proceso, tiene que ser eficiente.
- El mecanismo de saber/encontrar procesos hijos en estado de Zombie tiene que ser eficiente.
- **El proceso Init nunca llamará a *wait*.** En el caso de que lo haga, la llamada al sistema devolverá el error ECHILD.

a) Indica qué campos se tienen que añadir al task_struct de un proceso, describiéndolos brevemente.
b) Indica qué estructuras se tienen que modificar y añadir al sistema para implementar el mecanismo propuesto, junto con una breve explicación.
c) Indica en qué punto del código de Zeos se tienen que inicializar las estructuras nuevas descritas en el apartado b.
d) Explica cuando se liberarán los PCBs de los procesos hijos, en estado Zombie, de Init en Zeos.
e) ¿Se tiene que poner alguna restricción especial al task_struct del proceso Init?
f) Indica qué modificaciones se tienen que hacer a la llamada al sistema fork.
g) Indica qué modificaciones se tienen que hacer a la llamada al sistema exit.
h) Escribe el código de la llamada al sistema wait.
i) Implementa en Zeos los cambios propuestos en los apartados anteriores.

3. (1 punto) Generic Competences Third Language (Development Level: mid)

The following paragraph belongs to the book *Windows Internals* by Mark E. Russinovich and David A. Solomon:

> *"File objects are the kernel-mode constructs for handles to files or devices. File objects clearly fit the criteria for objects in Windows: they are system resources that two or more user-mode processes can share, they can have names, they are protected by object-based security, and they support synchronization. Although most shared resources in Windows are memory based resources, most of those that the I/O system manages are located on physical devices or represent actual physical devices. Despite this difference, shared resources in the I/O system, like those in other components of the Windows executive, are manipulated as objects."*

Nombre:
DNI:

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):

1.- Can system resources be shared among user-mode processes?

2.- Which mechanism ensures that a process holds enough privileges to access a given resource?

3.- What kind of devices are represented by file objects in Windows?

4. 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, si lo has hecho, de cada ejercicio utiliza:

```
> tar zcfv ejercicioX.tar.gz zeos
```