SOA (04/04/2017)

Nombre:
DNI:

Primer control de laboratorio-Grupo 12

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).

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

1. (3 puntos) Comprensión del código de Zeos

Indica para las siguientes líneas de código (si es posible): fichero, línea de código, función y para qué sirven:

a) `if ((remaining_quantum==0)&&(!list_empty(&readyqueue))) return 1;`

b) `movl $-ENOSYS, %eax;`

c) ¿En qué dirección lógica de memoria se encuentra la rutina de servicio de la excepción de "page fault"? ¿Cómo la has hallado? ¿Cuál es su correspondiente dirección física? ¿Cómo la has hallado?

2. (5 puntos + 0,5 puntos de implementación) Estadísticas

Queremos contar, de forma aproximada, el número medio de task_struct libres. Para ello, queremos crear un proceso de sistema, llamado *stats*, que lo haga. Este proceso se creará en tiempo de boot del sistema, el código de creación estará dentro de la función *init_stats y*, una vez creado, se encolará en la cola de READY. Cuando este proceso pase a RUN, contará el número de task_structs que, en ese momento, estén disponibles, y forzará un cambio de planificación al primer proceso de la cola de READY o a IDLE en caso de que la cola esté vacía. El proceso *stats* también contará el número de veces que pase a RUN.

Además, para poder acceder a estas estadísticas, crearemos una nueva llamada al sistema con la siguiente sintaxis:

```
int get_free_stats(struct free_stats *stats);
```

La definición de *struct free_stats* es:

```
struct free_stats {
  int free;          // Accumulated number of free task_struct
  int times;         // Number of times in RUN state
};
```

Free és el número acumulado de task_struct's que están libres y *times* es el número de veces que el proceso *stats* ha entrado en el estado de RUN. Así, *free/times* nos da el promedio de task_struct's que están libres.

Get_free_stats solamente se puede invocar a través de la posición 0x90 de la IDT y tiene cómo número de servicio, el 2.

Nombre:
DNI:

Contesta a las siguientes preguntas:

a) ¿Qué estructuras nuevas se tienen que añadir al sistema operativo y cuando y donde se inicializan?
b) ¿Qué estructuras actuales se tienen que modificar del sistema operativo?
c) ¿Desde qué punto del sistema (fichero y línea de código) se tiene que invocar a *init_stats*?
d) Escribe el código del wrapper de la llamada get_free_stats
e) Escribe el código del handler de la llamada get_ free_stats
f) Escribe el código de la rutina de servicio sys_get_ free_stats
g) Escribe el código de la función *init_stats*
h) Escribe el código del proceso *stats*
i) Implementa todo el código necesario en ZeOS.

3. (2 *puntos) Fork*

Modifica el código del fork para que el proceso padre use solamente la región de memoria 0x210000-0x224000 para acceder a la zona de datos del hijo y realizar la herencia de datos de usuario, en lugar de usar la región consecutiva a la zona de datos del padre.

a) ¿Cuantas páginas tiene esta región temporal? (SIZE)
b) ¿Cual es la primera dirección de la zona de datos de un proceso de usuario? (START_DATA)

Suponiendo que el bucle de copia fuera:

```
for(i=0; i < NUM_PAG_DATA; i++) {
        set_ss_page(...[1]...
        copy_data(...[2]...
        del_ss_page(...[3]...
        ...[4]...
}
```

Muestra como seria el código para hacer esta copia. Puedes usar las variables SIZE y START_DATA de los apartados anteriores, así como usar otras variables/funciones de ZEOS o nuevas que creas convenientes siempre y cuando lo indiques y muestres su inicialización.

c) Indica el código de [1]
d) Indica el código de [2]
e) Indica el código de [3]
f) Indica el código de [4]

4. (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:

Nombre:
DNI:

> *"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."*

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?

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, si lo has hecho, de cada ejercicio utiliza:

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