SISTEMAS OPERATIVOS

CURSO 2017-2018

SIMULADOR DE UN SISTEMA INFORMÁTICO MULTIPROGRAMADO: Gestión de un dispositivo de E/S

V5

Introducción

Una vez revisados en la parte teórica de la asignatura los esquemas básicos de gestión de dispositivos, procedemos a incorporar un dispositivo de entrada/salida simple al simulador de Sistema Informático.

DISEÑO

El dispositivo de entrada/salida

Es un dispositivo muy simple y únicamente de salida. Dispone de dos registros:

- Registro de estado. Mantiene el estado del dispositivo, que puede ser:
  - FREE: el dispositivo está esperando a que se le dé carga de trabajo.
  - BUSY: el dispositivo está realizando una entrada/salida encomendada por el SO.
- Registro de datos. Contiene la información necesaria para la realización de la operación de entrada/salida en curso.

El dispositivo de entrada/salida tiene un funcionamiento 100% fiable, es decir, todas las operaciones finalizan con éxito. Finalizada una operación de entrada/salida, el dispositivo eleva una interrupción para advertir de tal hecho al resto del sistema informático.

El procesador

El procesador cambia para dar el tratamiento hardware a un nuevo tipo de interrupción: la que señala el final de la realización de una operación de entrada/salida en el dispositivo.

El sistema operativo

El sistema operativo se comunicará con el dispositivo utilizando entrada/salida por interrupciones. La funcionalidad del sistema operativo a este respecto se dividirá entre:

- Manejador independiente del dispositivo. Recibe una solicitud de realización de entrada/salida realizada por un proceso vía una llamada al sistema y construye una petición (IORB). Si es necesario, encolará peticiones que no pueden ser atendidas de manera inmediata por estar ocupado el dispositivo. La petición es enviada al manejador dependiente del dispositivo para que le dé el tratamiento oportuno.
- Manejador dependiente del dispositivo. Recibe peticiones del manejador independiente y las traduce en órdenes que puede entender el dispositivo.
- Rutina de tratamiento de interrupción de fin de entrada/salida. Se ejecuta a instancias del procesador, que es el informado en primer lugar de la terminación de la operación de entrada/salida cuando el dispositivo eleva la interrupción. Dado que las operaciones de entrada/salida siempre terminan con éxito, no es necesario que compruebe cómo ha terminado. Únicamente tendrá que llamar la atención del manejador dependiente del dispositivo para que planifique la siguiente petición, en caso de que hubiera alguna pendiente.

IMPLEMENTACIÓN

El dispositivo de entrada/salida

- Estructuras de datos
  - Registros
    - Registro de estado (`registerStatus`). Contendrá uno de los dos valores del siguiente enumerado

      ```
      enum DeviceStatus {FREE, BUSY};
      ```

    - Registro de datos. Contiene dos valores. El valor `info` será el valor a mostrar por

pantalla y el valor `IOEndTick`, indicará el instante de reloj en el que la entrada/salida se dará por terminada.

```
typedef struct {
     int info;
     int IOEndTick;
} IODATA;
```

- Registro de duración de la operación (`IOLapTime`). Contiene el número de tics de reloj que dura una operación de E/S en el dispositivo. Se establece su valor en la inicialización del dispositivo.
- Registro con el nombre del dispositivo (`deviceName`). Contiene el nombre que se le dará al dispositivo y se establece su valor en la inicialización.

- Funcionalidad
  - `Device_Initialize(name, delay)`. Se invoca para inicializar el dispositivo antes de poder ser utilizado. Se le pasa el nombre que se le asigna y el número de tics de reloj que dura cada operación de E/S sobre el dispositivo.
  - `Device_StartIO(value)`. Invocada por el manejador dependiente del dispositivo del SO para pedirle al dispositivo que se disponga a realizar una entrada/salida. Como argumento le suministrará un valor entero que será utilizado en la operación de entrada/salida.
  - `Device_UpdateStatus()`. Esta función permite al dispositivo reconocer el instante de tiempo en que una entrada/salida se ha completado. En el momento en que se completa la operación, el dispositivo informa de tal hecho al procesador elevando una interrupción.

El procesador

El procesador deberá saber atender la ocurrencia de interrupciones de fin de entrada/salida. Para ello, deberá anotar en la posición apropiada de la tabla de vectores de interrupción la dirección de la rutina del SO que tratará dicho tipo de interrupciones. El manejador de interrupciones de entrada/salida será `OperatingSystem_HandleIOEndInterrupt`.

El sistema operativo

El sistema operativo divide sus responsabilidades entre el manejador independiente y el manejador dependiente del dispositivo:

- Manejador independiente del dispositivo.
  - Recibe una petición de entrada salida desde la rutina de tratamiento de llamadas al sistema (`TRAP 1`).
  - La inserta en la cola de peticiones asociadas al dispositivo.
  - Avisa al manejador dependiente del dispositivo de que hay una nueva petición.
- Manejador dependiente del dispositivo.
  - Espera la llegada de una petición.
  - La envía al dispositivo.
  - Espera a que el dispositivo complete la entrada/salida.
- Manejador de interrupciones de fin de entrada/salida.
  - Como en nuestro caso las operaciones de entrada/salida siempre se completan con éxito, la aparición de la interrupción provoca que se invoque al manejador dependiente del dispositivo para que gestione la siguiente petición pendiente.
  - Además, el proceso bloqueado esperando el fin de la entrada/salida, se desbloqueará.

La cola de peticiones pendientes es una cola FIFO (`QueueFIFO.c` y `QueueFIFO.h`) y se define así:

```
int IOWaitingProcessesQueue[PROCESSTABLEMAXSIZE];
int numberOfIOWaitingProcesses=0;
```

La operación de entrada/salida en curso ocupa la primera posición de la cola. El valor almacenado en cada posición de la cola se corresponde con el PID del proceso que solicitó la entrada/salida.

La cola FIFO

Utiliza un array de enteros que contiene los elementos de la cola, y otro entero que contiene el número de elementos que hay en la cola en cada momento.

Se añaden elementos a la cola usando la función:

`int QueueFIFO_add(int elemento, int fifo[], int *numElementos, int limiteMaximo)`

Que añade `elemento` a la cola FIFO apuntada por `fifo` en la que caben hasta `limiteMaximo` elementos, incrementando si cabe `numElementos`. Devuelve 0/-1 si cabe/no cabe el elemento a insertar.

Se saca el primero de la cola usando la función:

`int QueueFIFO_poll(int fifo[], int *numElementos)`

Que saca el primer elemento de la cola apuntada por `fifo` si lo hay (lo devuelve) y decrementa `numElementos`. Si no hay elementos devuelve -1.

Se consulta el primer elemento de la cola sin sacarlo:

`int QueueFIFO_getFirst(int fifo[], int numElementos)`

Que devuelve el primer elemento de la cola apuntada por `fifo` si hay algún elemento. Si no hay elementos devuelve -1.

SIMULADOR DE UN SISTEMA INFORMÁTICO MULTIPROGRAMADO: Gestión de un dispositivo de E/S

Tareas V5

Tareas iniciales

Saca un duplicado de tu directorio `V4` (una vez completada esta versión) denominándolo `V5`. El trabajo a realizar en los ejercicios siguientes se desarrollará sobre la copia indicada de los ficheros contenidos en el directorio `V5`, dentro de tu directorio personal. Copia también en dicho directorio `V5` los recursos disponibles para esta versión en el directorio `/var/asignaturas/ssoo/2017-2018/V5-studentsCode` de `ritchie`.

Ejercicios

1. Modifica el **sistema** para que sepa reconocer interrupciones de fin de entrada/salida (`IOEND_BIT = 8`) e invoque, llegado el momento, a la rutina del SO de tratamiento de dichas interrupciones, que deberá ser `OperatingSystem_HandleIOEndInterrupt(){}`.

2. Implementa el manejador independiente del dispositivo con el nombre `OperatingSystem_IOScheduler`. La responsabilidad de dicho manejador consiste en añadir una nueva petición a la cola de peticiones pendientes e informar de tal hecho al manejador dependiente del dispositivo. Definido así hasta que se complete en el ejercicio 4-a: `OperatingSystem_DeviceControlerStartIOOperation(){}`

   La cola de peticiones pendientes se implementa como una cola FIFO en esta estructura del sistema operativo, que hay que incluir en el `OperatingSystem.c`:

   ```
   int IOWaitingProcessesQueue[PROCESSTABLEMAXSIZE];
   int numberOfIOWaitingProcesses=0;
   ```

   La implementación de una cola FIFO está en `QueueFIFO.c` y `QueueFIFO.h` (ver lo que es cada parámetro en ellos) y se maneja usando las funciones siguientes:

   `int QueueFIFO_add(int, int[], int*, int)`; Para añadir al final de la cola.

   `int QueueFIFO_poll(int[], int*)`; Para sacar el primer elemento de la cola.

   `int QueueFIFO_getFirst(int[], int)`; Para devolver el primer elemento de la cola sin sacarlo.

3. Vamos a implementar un dispositivo de E/S, para ello:

   a. Añade la siguiente línea de código dentro del fichero `ComputerSystem.h` como nueva sección para los mensajes de depuración.

```
#define DEVICE 'v'
```

b. Añade la llamada a la función siguiente (está en `Device.c`) justo antes de intentar la creación de los procesos de la inicialización:

```
Device_Initialize("OutputDevice-2018", 7);
```

4. Implementa el manejador dependiente del dispositivo con dos funciones:

   a. `OperatingSystem_DeviceControlerStartIOOperation()` será invocada por el manejador independiente del dispositivo para pedirle al dispositivo que realice una operación de entrada/salida. El proceso que hace la entrada/salida será el primero de la cola de solicitudes. Si el dispositivo está **libre**, la información que debe enviar al dispositivo para que la muestre (`Device_StartIO(value)`) será el PID del proceso que solicita la operación.

   b. `OperatingSystem_DeviceControlerEndIOOperation` será invocada por la rutina de tratamiento de interrupciones de fin de entrada/salida y se encargará de gestionar la siguiente petición de la cola, en el caso de que exista. Devuelve el PID del proceso que ha terminado su operación de E/S

5. Implementa la rutina del SO de tratamiento de interrupciones de fin de entrada/salida, `OperatingSystem_HandleIOEndInterrupt`. La rutina tendrá tres responsabilidades fundamentales:

   a. Pedir al manejador dependiente del dispositivo que se disponga a gestionar la siguiente petición pendiente.

   b. Desbloquear al proceso cuya entrada/salida ha finalizado. Como hay cambio de estado de procesos, hay que sacar los mensajes de cambio de estado pertinentes y como implica cambios en las colas, se debe hacer una llamada a `OperatingSystem_PrintStatus()`

   c. Requisar el procesador al proceso en ejecución si fuese necesario, en cuyo caso se llamaría a `OperatingSystem_PrintStatus()`

6. Implementa una nueva llamada al sistema (`SYSCALL_IO=1`) que podrá ser utilizada por los procesos de usuario para realizar entrada/salida sobre el dispositivo. El nuevo servicio realizará, fundamentalmente, dos cosas:

   a. Bloquear al proceso que realiza la llamada. Como hay cambio de estado, se debe mostrar el mensaje de cambio de estado pertinente.

   b. Invocar al manejador independiente del dispositivo para que se ponga en marcha la operación.

   c. Hacer una llamada a `OperatingSystem_PrintStatus()`

Escuela Ingeniería Informática de Oviedo. Sistemas Operativos: Simulador V5 Final

MainMemory

MAR

MBR

mainMemory

0
1
2
3
…
(**)
…
(*)

(**) =PartitionsSizeSum -1
(*) = MAXMEMORYSIZE - 1

Clock

tics

Device

Status
IOLapTime
*deviceName
regIODData

MMU

Base

Limit

MAR

ComputerSystem

ProgramList

*PROGRAMDATA

*Name Arrival Type

0
1
2 null
(*) null

(*) = PROGRAMMAXNUMBER-1

Arrival
(index in programList)

0
1
2
(*)

(*) = PROGRAMMAXNUMBER-1

NumArrival

Message Subsystem
….

Processor

accum

PC

IR

MAR

MBR

PSW

A

B

Int

VInt.

0	null
1	null
2	SysCall Entry
3	null
4	null
5	null
6	Exception Entry
7	null
8	null
9	ClockInt Entry
…	
(*)	null

(*) = INTERRUPTTYPES -1

OperatingSystem

executingProcessID

sipID

NonTerminated

numberOfClockInterrupts

processTable

PID busy initialAddr size state priority Copy PC programListIndex queueID whenToWakeUp …

0
1
2
*

partitionsTable

occupied initAddress size PID

0
1
2
…
(**)

(**) 2*PROCESSTABLEMAXSIZE – 1

ReadyToRun (PID)

0 1 2 … (*)

USER
DAEM

NumReadyToRun

USER
DAEM

Sleeping (PID)

0 1 2 … (*)

NumSleeping

IOWaiting (PID)

0 1 2 … (*)

NumIOWaiting

(*) PROCESSTABLEMAXSIZE – 1