Actividad Práctica capitulo 2 UF1287

Enunciado:

En este documento se desarrollan dos ejercicios prácticos relacionados con el Capítulo 2, el cual trata sobre la programación de controladores de dispositivos. En este capítulo se aborda la necesidad de disponer de código específico, los denominados drivers, para permitir que el sistema operativo se comunique adecuadamente con los distintos dispositivos de hardware.

A través de estos ejercicios, se busca afianzar los conocimientos sobre el comportamiento de los drivers, el uso de interrupciones, la transferencia de datos, y la clasificación de errores comunes que pueden ocurrir en la interacción entre el hardware y el sistema operativo.

Ejercicios:

Ejercicio 1: Clasificación de errores de dispositivos

Se dispone de una lista de errores producidos y se necesita clasificar por el tipo de error para tratar de elaborar el plan de actuación correspondiente. La lista de errores es la siguiente:

- Enviar un texto para visualizar a un dispositivo de tipo teclado.
- Intentar leer de una llave USB que no está conectada.
- Imprimir un documento y la impresora está apagada.
- Se pretende utilizar de forma errónea la estructura de directorios.
- Se ha intentado escribir en una dirección de memoria protegida.
- En una función que tiene que devolver un mensaje de éxito o fracaso, se devuelve una dirección de memoria.

Ejercicio 2: Cálculo del uso de CPU por dispositivos

Determine el porcentaje de tiempo que está ocupando la CPU con los siguientes dispositivos: un ratón y un disco duro.
Se supone que, para conectarse a la CPU, los dispositivos lo hacen mediante una E/S programada.

- El ciclo completo de la operación de E/S es de 400 unidades y la CPU trabaja a 500 MHz.
- Es necesario leer el ratón 30 veces por segundo para que no se pierda ningún movimiento.
- El disco duro transfiere datos con el procesador en bloques de 4 palabras a una velocidad de 4 Mb/s. (Mb es megabit --- MB es megabyte)

Ejercicio 3: Simulación de interrupciones sin soporte hardware

Se está utilizando un sistema que no permite el tratamiento hardware de interrupciones. La idea es simular el comportamiento que realizaría el sistema de interrupciones sin tenerlo implementado.

¿Se podría conseguir un efecto similar a la gestión de interrupciones en un procesador sin este sistema? Si es así, ¿cómo?

Ejercicio 4: Comparación de modos de transferencia

Se va a realizar una transferencia de una palabra (byte), pero se necesita saber cuál será la opción que implicará menos a la CPU mientras se realiza la transferencia.

Las tres opciones que se pueden dar son:
- TP: tiempo de E/S controlado por programa.
- TI: tiempo de transferencia con interrupciones.
- TD: tiempo de transferencia usando DMA.

Ejercicio 5: Clasificación de requerimientos funcionales

La empresa X, que ha solicitado el diseño del driver 'Hola, mundo', necesita determinar con los requerimientos obtenidos cuáles pertenecen a requerimientos funcionales y cuáles no.
La tabla de requerimientos es la siguiente:
Requerimiento:
- Realizar bajo Linux
- Realizar en modo kernel
- Mensaje de carga
- Mensaje de descarga
- Compatible sistema

Clasifique cada requerimiento como funcional o no funcional.

Ejercicio 6: Orden de generación de archivos

En un directorio están almacenados los archivos que se listan a continuación. ¿En qué orden se han generado los archivos?, ¿qué herramientas se han utilizado para la generación del archivo?

Lista de archivos:
- new_prog.s
- new_prog.out
- new_prog.o
- new_prog.c

Forma de Entrega:

El caso práctico deberá entregarse en formato pdf:

ApellidoNombre_CasoPractico2 UF1287.pdf

Ejercicio 1: Clasificación de errores de dispositivos

Se dispone de una lista de errores producidos y se necesita clasificar por el tipo de error para tratar de elaborar el plan de actuación correspondiente. La lista de errores es la siguiente:

- Enviar un texto para visualizar a un dispositivo de tipo teclado. → Error de Programación
- Intentar leer de una llave USB que no está conectada. → Error de E/S o hardware
- Imprimir un documento y la impresora está apagada. → Error de E/S / dispositivo no disponible
- Se pretende utilizar de forma errónea la estructura de directorios. → Error sin clasificar a resolver por el sistema
- Se ha intentado escribir en una dirección de memoria protegida. → Error de programación punteros erroneos
- En una función que tiene que devolver un mensaje de éxito o fracaso, se devuelve una dirección de memoria. → Error de tipo de parametro devuelto

Ejercicio 2: Cálculo del uso de CPU por dispositivos

Determine el porcentaje de tiempo que está ocupando la CPU con los siguientes dispositivos: un ratón y un disco duro.

Se supone que, para conectarse a la CPU, los dispositivos lo hacen mediante una E/S programada.

- El ciclo completo de la operación de E/S es de 400 unidades y la CPU trabaja a 500 MHz.
- Es necesario leer el ratón 30 veces por segundo para que no se pierda ningún movimiento.
- El disco duro transfiere datos con el procesador en bloques de 4 palabras a una velocidad de 4 Mb/s. (Mb es megabit --- MB es megabyte)

Raton:

400 ciclos * 30 ciclos /s = 12000.

500 mega-ciclos → 500,000,000 ciclos que es el 100%; 12000 es el x%

la regla de tres → x =12000 * 100 / 500.000.000 → 0,0024 %

Disco duro:

si estamos en un sistema de 32 bits, que es lo que sugiere por la frecuencia y la transferencia en bloques de 4 palabras, cada palabra es de 4 bytes pero para asegurar, si estamos en linux podemos verlo con el comando getconf WORD_BIT. 1. – pasamos la velocidad a bytes/s 4Mb / 8 bits son 500.000 bytes. 2.- cuantos bloques de 16 bytes /s 500,000 /16 = 31.250 bloques/s. 3.- el ciclo completo es 400 uds x 31.250 bloques 12.500.000. 4.- Ya tenemos claro que el porcentaje es 12,500,000 *100% / 500,000,000= 2,5%

Ejercicio 3: Simulación de interrupciones sin soporte hardware

Se está utilizando un sistema que no permite el tratamiento hardware de interrupciones. La idea es simular el comportamiento que realizaría el sistema de interrupciones sin tenerlo implementado.

¿Se podría conseguir un efecto similar a la gestión de interrupciones en un procesador sin este sistema? Si es así, ¿cómo?

Sí se podría por software, tras cada instrucción consultar si se ha producido alguna operación E/S, en caso afirmativo que se dirija a la rutina que maneje la E/S y cuando acabe que vuelva a la secuencia del programa, pero no sería eficiente, ya que los recursos liberados estarían a la espera

Ejercicio 4: Comparación de modos de transferencia

Se va a realizar una transferencia de una palabra (byte), pero se necesita saber cuál será la opción que implicará menos a la CPU mientras se realiza la transferencia.

Las tres opciones que se pueden dar son:
- TP: tiempo de E/S controlado por programa.
- TI: tiempo de transferencia con interrupciones.
- TD: tiempo de transferencia usando DMA. → la CPU cede el control a la DMA para la gestión de las transacciones, con lo que esta es la opción donde la CPU esta menos implicada

Ejercicio 5: Clasificación de requerimientos funcionales

La empresa X, que ha solicitado el diseño del driver 'Hola, mundo', necesita determinar con los requerimientos obtenidos cuáles pertenecen a requerimientos funcionales y cuáles no. La tabla de requerimientos es la siguiente:

Requerimiento:
- Realizar bajo Linux NO FUNCIONAL
- Realizar en modo kernel NO FUNCIONAL
- Mensaje de carga FUNCIONAL
- Mensaje de descarga FUNCIONAL
- Compatible sistema NO FUNCIONAL

Clasifique cada requerimiento como funcional o no funcional.

Ejercicio 6: Orden de generación de archivos

En un directorio están almacenados los archivos que se listan a continuación. ¿En qué orden se han generado los archivos?, ¿qué herramientas se han utilizado para la generación del archivo?

Lista de archivos:

2.- new_prog.s → COMPILADOR sudo gcc -S new_prog.c

4.- new_prog.out → LINKADOR GENERA EL EJECUTABLE sudo gcc new_prog.c -o new_prog.out

3.- new_prog.o → ENSAMBLADOR, no realiza el ejecutable sudo gcc -c new_prog.c

1.- new_prog.c → EDITOR DE TEXTO ya sea integrado en IDE como cualquier editor de texto plano por ejemplo sudo nano new_prog.c