Casos Prácticos Adicionales - Capitulo 2 UF1287

Caso práctico 7: Controlador de un dispositivo ficticio - SensorX

Enunciado:
Diseña un esquema de controlador para un dispositivo ficticio llamado SensorX que envía datos de temperatura cada segundo. El dispositivo se conecta por puerto serie y genera una interrupción cuando tiene un dato nuevo disponible.

Objetivos:

- - Identificar los componentes mínimos del controlador.
- - Diseñar la rutina de interrupción.
- - Indicar cómo se gestiona el búfer de recepción y cómo se comunica con el sistema.

Caso práctico 8: Diferencias entre controladores de bloque y de caracteres

Enunciado:
Explica las diferencias entre un controlador de dispositivo de bloque (por ejemplo, disco duro) y uno de caracteres (por ejemplo, teclado). Proporcione un ejemplo de implementación simplificada de cada tipo.

Objetivos:

- - Comprender el flujo de datos y acceso al hardware.
- - Diferenciar estructuras de manejo en el núcleo.

Caso práctico 9: Simulación de un driver de teclado con buffer circular

Enunciado:
Implemente en pseudocódigo el manejo de entradas de teclado utilizando un buffer circular. Incluya funciones para leer del buffer, escribir, y verificar si está lleno o vacío.

Objetivos:

- - Aplicar estructuras de datos al manejo de hardware.
- - Simular el comportamiento de entrada/salida continua.

```
Inicio
declarar bufer tipo fifo
para (i=0 ;i< bufer.lomgitud() ; i++)
    leer carácter
    escribir carácter en memoria
borrar buffer
```

Caso práctico 10: Medición del rendimiento de un driver

Enunciado:
Proponer un conjunto de métricas para evaluar el rendimiento de un controlador de dispositivo. Luego, describa un experimento controlado para medir el tiempo de respuesta y consumo de CPU.

Objetivos:

- - Introducir análisis de rendimiento y profiling.
- - Evaluar cuellos de botella en el acceso a dispositivos.

Caso práctico 11: Manejo de errores en controladores

Enunciado:
Plantea un conjunto de fallos típicos que puede producir un controlador (por ejemplo, pérdida de interrupciones, acceso nulo al hardware, corrupción de datos) y propone cómo podrían detectarse y manejarse desde el código del driver.

Objetivos:

- - Desarrollar buenas prácticas de robustez.
- - Introducir el uso de logs, validaciones y recuperación.

Caso práctico 7: Controlador de un dispositivo ficticio - SensorX

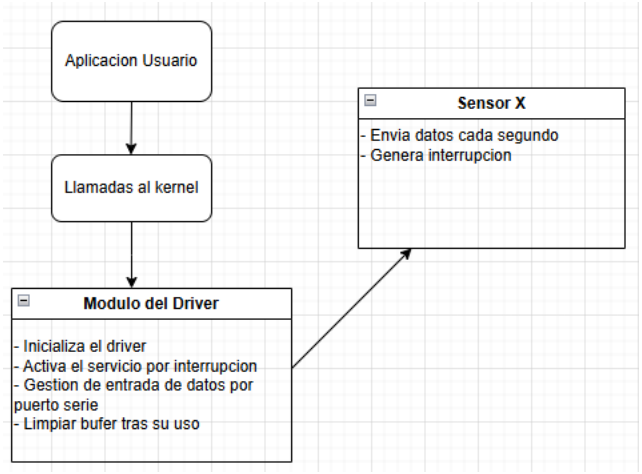
Enunciado:
Diseña un esquema de controlador para un dispositivo ficticio llamado SensorX que envía datos de temperatura cada segundo. El dispositivo se conecta por puerto serie y genera una interrupción cuando tiene un dato nuevo disponible.

Objetivos:

- - Identificar los componentes mínimos del controlador.

Unidad de control o CPU	Coordina el funcionamiento, ejecuta el código de lectura y temporización.
Temporizador	Genera interrupciones o señales cada segundo para iniciar la lectura.
Interfaz de comunicación	Permite conectar con el sensor (puede ser UART, I^2C, SPI, etc.).
Convertidor A/D (si aplica)	Si el sensor entrega datos analógicos, convierte a digital.
Registro de datos	Guarda la lectura de temperatura temporalmente antes de procesarla o enviarla.
Memoria (RAM/ROM)	ROM para el programa de lectura; RAM para almacenar temporalmente las lecturas.
Circuito de entrada/salida (E/S)	Coordina el intercambio de datos entre el sensor y el controlador.

- - Diseñar la rutina de interrupción.



Inicializador del dispositivo
Configura parámetros del puerto serie (baudios, bits de datos, paridad, etc.).
Registra la rutina de interrupción en el sistema operativo.
Rutina de interrupción (ISR)
Se ejecuta automáticamente al recibir la señal del SensorX.
Extrae el dato del puerto serie, lo guarda y notifica al sistema.
Gestor del búfer de recepción
Almacena los datos recibidos.
Permite a procesos del sistema acceder a ellos de forma segura.
Interfaz hacia el sistema
Funciones tipo read(), ioctl() o llamadas del sistema para que el usuario recupere los datos

- - Indicar cómo se gestiona el búfer de recepción y cómo se comunica con el sistema.

Elemento	Función clave
Inicialización	Configura puerto e interrupción
ISR	Lee el dato y lo coloca en el búfer
Búfer de recepción	Almacena datos en estructura circular
Interfaz al sistema	Permite que usuarios consulten la temperatura

Caso práctico 8: Diferencias entre controladores de bloque y de caracteres

Enunciado:
Explica las diferencias entre un controlador de dispositivo de bloque (por ejemplo, disco duro) y uno de caracteres (por ejemplo, teclado). Proporcione un ejemplo de implementación simplificada de cada tipo.

Objetivos:

- - Comprender el flujo de datos y acceso al hardware.
- - Diferenciar estructuras de manejo en el núcleo.

El controlador de bloques pasa bloques de datos (Discos duros, memorias USB) del mismo tamaño, que no necesitan leerse o escribirse de forma secuencial o continua, sino aleatoria. Se pueden realizar operaciones de lectura, escritura, y búsqueda

Y el de carácter una secuencia de caracteres que en binario son 8 bits/caracter. (puertos serie, impresoras, teclados, tarjetas de sonido) solo se pueden hacer operaciones de lectura y/o escritura

El carácter a carácter tiene que seguir un orden, ya que de ello depende que el flujo sea el correcto

como se ve la diferencias son muchas, a destacar el tamaño de cada entrada/salida y que en bloque también se tiene la opción de búsqueda, al ser e/s aleatoria

Caso práctico 9: Simulación de un driver de teclado con buffer circular

Enunciado:

Implemente en pseudocódigo el manejo de entradas de teclado utilizando un buffer circular. Incluya funciones para leer del buffer, escribir, y verificar si está lleno o vacío.

```
// variables globales
Variable buffer[caracteres del buffer] ; variable inicio = 0;  variable fin = 0; variable contador = 0; control= True

void Función escribir(tecla)
   Si contador == TAMANIO_BUFFER Entonces
      Mostrar "Buffer lleno, no se puede escribir"
   Sino
      buffer[fin] = tecla
      fin = (fin + 1) mod TAMANIO_BUFFER // para que si llega al final del buffer circular, vuelva a apuntar al principio
      contador = contador + 1
   Fin Si
Fin Función
char  Función leer() → tecla // lee y devuelve el carácter leido en la variable tecla
   Si contador == 0 Entonces
      Mostrar "Buffer vacío, no hay teclas para leer"
      Retornar NULL
   Sino
      tecla = buffer[inicio]
      inicio = (inicio + 1) mod TAMANIO_BUFFER /* incrementa el inicio  si llega al final vuelve a posicionarse al inicio ya que
es circular*/
      contador = contador – 1 // tecla leida descuenta el contador
      Retornar tecla
   Fin Si
Fin Función
booleano Función estaVacio() → booleano
   Retornar contador == 0 // si es correcto devuelve true
Fin Función
booleano Función estaLleno() → booleano
   Retornar contador == TAMANIO_BUFFER // si es correcto devuelve false
Fin Función
Inicio Programa
mientras  control = True
   Inicializar buffer circular
      Si NO estaVacio() Entonces              tecla = leer()
        si no                tecla= NULL        Fin Si
     Si tecla not NULL Entonces
         Si estaLleno() Entonces
            Mostrar "Buffer lleno. Tecla '" + tecla + "' no admitida"
         Sino
            escribir(tecla)
            Mostrar "Tecla '" + tecla + "' almacenada en el buffer circular en la posición"+ contador -1 +""
         Fin Si
      Fin Si
      Si pregunta("dispositivo desea continuar") == False Entonces
                  Mostrar "Proceso concluido"
                  liberar recursos
                  control= False
  fin mientras
Fin Programa
```

Objetivos:

- - Aplicar estructuras de datos al manejo de hardware.
- - Simular el comportamiento de entrada/salida continua.

Caso práctico 10: Medición del rendimiento de un driver

Enunciado:
Proponer un conjunto de métricas para evaluar el rendimiento de un controlador de dispositivo. Luego, describa un experimento controlado para medir el tiempo de respuesta y consumo de CPU.

latencia (velocidad)
uso de memoria
porcentaje de uso de CPU

Concepto	Alto	Medio	Bajo
latencia	<6	<12	>=12
uso de memoria	> 400MB	> = 100MB	< 100MB
Porcentaje de uso de CPU	> 10%	>=3%	<3%

Con esta tabla de valores y el monitor de rendimiento se puede ver si el controlador tiene un rendimiento alto, medio o bajo dependiendo de los valores que indique la herramienta de medición

Objetivos:

- - Introducir análisis de rendimiento y profiling.
- - Evaluar cuellos de botella en el acceso a dispositivos.

Caso práctico 11: Manejo de errores en controladores

Enunciado:
Plantea un conjunto de fallos típicos que puede producir un controlador (por ejemplo, pérdida de interrupciones, acceso nulo al hardware, corrupción de datos) y propone cómo podrían detectarse y manejarse desde el código del driver.

Tabla de fallos de controladores

Fallo	Descripción	Manejo de driver
No conecta	El dispositivo no es detectado no se puede acceder	Mostrar mensaje de conectar a la red electrica y encender el dispositivo, si es wifi esperar un tiempo prudencial para empezar de nuevo
Datos corruptos	El dispositivo no envia / recibe los datos para los que esta preparado	Enviar/recibir un bloque/paquete/carácter de prueba conocido, y comprobar si coincide con el enviado/recibido en caso negativo actualizar controlador y efectuar la misma prueba
Perdida de interrupciones	El dispositivo se haya bloqueado sin que la cpu admita su interrupción	Comprobar si el porcentaje de uso de cpu si es mayor del 90% mostrar aviso de sistema lento y posible perdida de datos si se sigue usando

Objetivos:

- - Desarrollar buenas prácticas de robustez.
- - Introducir el uso de logs, validaciones y recuperación.