Evaluación Práctica en Computadora

# Temas: Programación de E/S, Manejadores de Dispositivos y Programación de Interrupciones

**Instrucciones Generales:**

1. Trabaja individualmente en tu equipo.
2. Usa un entorno Linux o Windows con compilador C/C++ o lenguaje similar.
3. Guarda tu trabajo en una carpeta llamada **Evaluacion\_EyS\_ApellidoNombre**.
4. Responde las preguntas y completa los ejercicios solicitados.

# Parte I – Preguntas de Selección Múltiple (2 puntos cada una)

1. ¿Cuál de las siguientes afirmaciones describe mejor la función de la E/S (Entrada/Salida) en un sistema operativo?
   1. Solo se encarga de enviar datos a la impresora.
   2. Permite la comunicación entre el procesador y los dispositivos periféricos.
   3. Ejecuta programas almacenados en la memoria.
   4. Controla los procesos en segundo plano.
2. Un manejador de dispositivo (driver) es:
   1. Un programa que traduce las órdenes del hardware al usuario.
   2. Un componente del sistema operativo que actúa como intermediario entre el hardware y el sistema operativo.
   3. Una aplicación que controla la interfaz gráfica.
   4. Una interrupción generada por el teclado.
3. ¿Qué ocurre cuando se activa una interrupción de hardware?
   1. El sistema operativo entra en modo usuario.
   2. El procesador ignora la señal hasta que termina el proceso.
   3. El procesador detiene temporalmente la ejecución actual y ejecuta una rutina específica.
   4. El procesador borra la memoria RAM.
4. El tipo de interrupción software (INT) se genera:
   1. Por un fallo eléctrico.
   2. Por la ejecución explícita de una instrucción en el programa.
   3. Por el reloj del sistema.
   4. Por un error del disco duro.
5. ¿Cuál de las siguientes opciones muestra un ejemplo de E/S programada?
   1. El procesador espera a que el periférico esté listo antes de continuar.
   2. El procesador continúa mientras el periférico trabaja en paralelo.
   3. Se utiliza un canal DMA (Acceso Directo a Memoria).
   4. El procesador no interviene en la transferencia.

# Parte II – Ejercicios Prácticos (3 puntos cada uno)

1. En C, escribe un fragmento de código que lea desde el teclado una cadena y la guarde en un archivo llamado datos.txt.

#include int main() { FILE \*f; char texto[100]; printf("Ingrese texto: "); gets(texto); f = fopen("datos.txt", "w"); fprintf(f, "%s", texto); fclose(f); return 0; }



Pregunta: ¿Qué tipo de E/S se utiliza en este ejemplo y qué función cumple el sistema operativo?

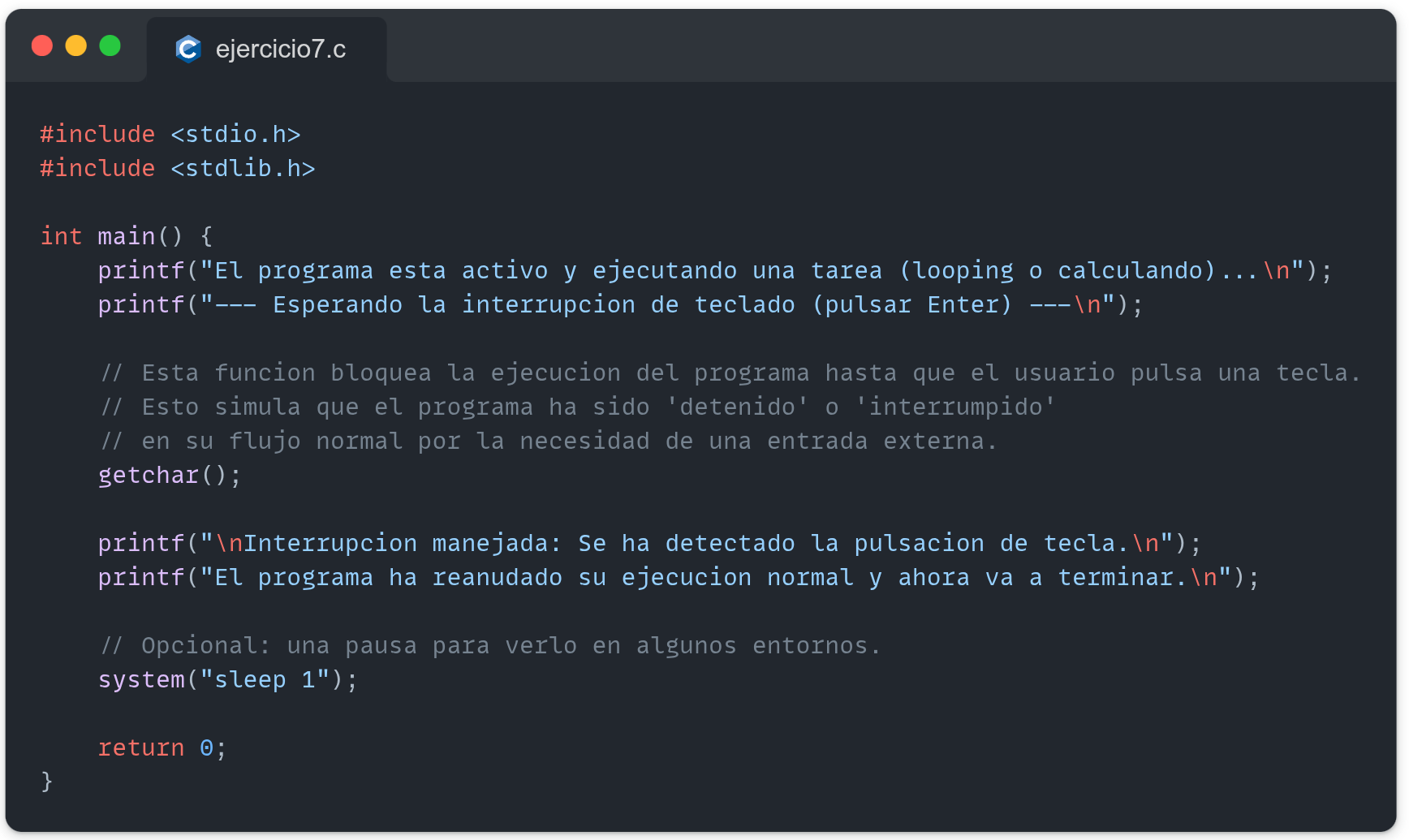
El tipo de E/S que se utiliza en este ejemplo es E/S a nivel de usuario (manejo de datos), implementada a través de las funciones de la biblioteca estándar de C (stdio.h). Específicamente, se utilizan:

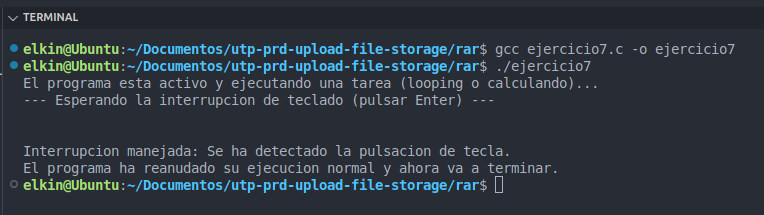
**Entrada interactiva (printf, gets):** Se lee de un periférico de entrada (teclado).

**E/S de archivos (fopen, fprintf, fclose):** Se escribe en un periférico de almacenamiento (disco duro).

|  |  |  |
| --- | --- | --- |
| **Función** | **Tipo de E/S** | **Periférico** |
| printf/gets | E/S interactiva | Terminal/Teclado |
| fopen/fprintf | E/S de archivos | Disco duro |

1. **Simula una interrupción de teclado usando un lenguaje o entorno que detecte pulsación de teclas.**



****

**Pregunta: Explica qué componente del sistema operativo gestiona esta interrupción y cómo se reanuda la ejecución normal del programa.**

El componente clave del sistema operativo que gestiona la interrupción de teclado (generada al pulsar Enter para el getchar()) es el Núcleo (*Kernel*), específicamente a través de su Manejador de Interrupciones (o ISR). Cuando se pulsa la tecla, el *hardware* notifica a la CPU, y el *Kernel* toma el control, guarda el estado actual del programa, lee el carácter del *hardware* del teclado y lo almacena en un *búfer* interno. Una vez que el *Kernel* ha procesado la interrupción y el dato está disponible, utiliza una instrucción de Retorno de Interrupción (RTI) para restaurar el estado del programa (su contador de programa y registros), permitiendo que la llamada al sistema (getchar()) finalice y entregue el dato leído, con lo cual el programa reanuda su ejecución normal a partir de la siguiente instrucción.

1. **Investiga y muestra con capturas o comandos los drivers del sistema operativo. Pregunta: ¿Cuál es la relación entre un driver y el kernel del sistema operativo?**

En el sistema operativo basado en Linux (como Ubuntu), los *drivers* se implementan como módulos del *Kernel*. Los siguientes comandos son útiles para ver la información de *hardware* y los módulos (*drivers*) cargados:

|  |  |  |
| --- | --- | --- |
| **Comando** | **Función** | **Ejemplo de Captura/Salida** |
| lspci | Lista todos los dispositivos conectados a través del bus PCI (tarjeta gráfica, red, etc.). | 01:00.0 VGA compatible controller: NVIDIA Corporation [...] |
| lsusb | Lista todos los dispositivos conectados al bus USB. | Bus 001 Device 002: ID 8087:0a2a Intel Corp. Bluetooth |
| lsmod | Muestra el estado de los módulos del Kernel (los drivers) actualmente cargados en memoria. | nvidia\_modeset 1179648 1 nvidia |
| dmesg | Muestra los mensajes del buffer del Kernel, a menudo indicando la carga de drivers y errores de hardware durante el arranque. | [ 3.456789] usbcore: registered new interface driver usb-storage |

Relación entre un Driver y el Kernel del Sistema Operativo son la **dependencia funcional** y **jerarquía de privilegios**.

Los *drivers* son **extensiones fundamentales del *Kernel*** (módulos) que se ejecutan en el **Modo *Kernel***, el nivel más privilegiado del sistema. Su función es actuar como **traductores** entre las peticiones de alto nivel del *software* y los comandos de bajo nivel que entiende el *hardware* específico. De esta manera, el *Kernel* logra **abstraer** la complejidad del *hardware* y gestionar la E/S (Entrada/Salida). Como se ejecutan en el núcleo, un fallo en un *driver* puede causar un **fallo crítico** del sistema (*kernel panic*).

# Parte III – Análisis y Reflexión (2 puntos)

1. Explica con tus palabras:

**¿Por qué las interrupciones aumentan la eficiencia del sistema operativo?**

Las interrupciones aumentan significativamente la eficiencia del sistema operativo porque permiten que la CPU se libere de la ineficiente tarea de espera activa (*polling*) a los dispositivos de Entrada/Salida (E/S) lentos. En lugar de que el procesador esté preguntando constantemente si un dispositivo (como un disco duro o una tarjeta de red) ha terminado su trabajo, el sistema operativo utiliza interrupciones, lo que significa que el hardware avisa asíncronamente a la CPU solo cuando necesita atención. Esto permite que el procesador utilice ese tiempo para ejecutar otras tareas y procesos de manera concurrente, maximizando el uso del tiempo de CPU.

**¿Qué riesgos existirían si los controladores de dispositivos no estuvieran correctamente programados?**

El principal riesgo de tener controladores de dispositivos (drivers) mal programados es la inestabilidad total del sistema (llevando a fallos como el *kernel panic* o el Pantallazo Azul), ya que los *drivers* se ejecutan en el modo *kernel*, el nivel de mayor privilegio, y un error en su código puede corromper la memoria central del sistema operativo. Además, estos fallos crean graves vulnerabilidades de seguridad, permitiendo que *software* malicioso explote el *driver* para escalar privilegios y obtener control total sobre el sistema, comprometiendo su seguridad y los datos del usuario.

**Criterios de Evaluación**

|  |  |  |
| --- | --- | --- |
| Criterio | Descripción | Ponderación |
| Comprensión teórica | Define correctamente los conceptos básicos | 30% |
| Aplicación práctica | Ejecuta correctamente los ejercicios en computadora | 40% |
| Análisis y reflexión | Argumenta con claridad y coherencia | 20% |
| Presentación | Orden, claridad y evidencias (capturas o código) | 10% |