Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Profesor: Jason Leitón, Leonardo Araya
CE 4303 Sistemas operativos
II Semestre 2024

Examen 1
Tiempo: 60 min
Total puntos: 22 pts
Puntos obtenidos: _____

Carné: _____ Nombre: _____ Nota: _____

INSTRUCCIONES GENERALES.

- Esta evaluación es individual.
- Sus respuestas deben estar escritas en lapicero, en caso contrario no se aceptan reclamos.
- Responda sólo lo que se le solicita y de forma *ordenada* identificando claramente la enumeración de cada ejercicio.
- Enumere en la esquina superior derecha cada página.
- Puede utilizar una página de material de apoyo escrita a mano.
- Cada ejercicio se debe realizar de manera ordenada.

I. Respuesta breve (15 minutos)

Responda las siguientes preguntas de forma ordenada, asegúrese de identificar cada ejercicio, de no ser así se tomarán las respuestas en orden.

1. ¿Cuál es la diferencia entre un programa, un proceso y un hilo? (3pts)
2. Respecto a los modos de operación de un sistema operativo ¿Un programa ejecutado en modo usuario podría tener acceso a algún recurso para el cual se requiera modo kernel? (2pts)
3. Explique qué es un *daemon* y mencione algunas de sus aplicaciones (2pts)

II. Análisis (10 minutos)

Responda de manera puntual lo que se le solicita.

1. Suponga que se tiene un programa en C que maneja interrupciones con diferentes prioridades. Se tiene un temporizador que genera una interrupción cada 200ms y un botón que genera una interrupción de mayor prioridad. Actualmente se tiene el siguiente código, **el cual funciona correctamente**, mas no todo ocurre de forma explícita.

```c
// Contadores globales
volatile uint32_t timer_count = 0;
volatile uint32_t button_count = 0;
// ISR del temporizador
void TimerISR(void) {
    timer_count++;
    printf("Timer Interrupt: timer_count = %u\n", timer_count);
    // Asuma que muchas otras cosas suceden aquí...
```

```
}
// ISR del botón (de mayor prioridad)
void ButtonISR(void) {
    button_count++;
    printf("Button Interrupt: button_count = %u\n", button_count);
    // Asuma que muchas otras cosas suceden aquí...
}
// Funciones para configurar el temporizador y el botón (pseudocódigo)
void setupTimer(void) {
    // Configura el temporizador para generar una interrupción cada 200 ms
}
void setupButton(void) {
    // Configura el botón para generar una interrupción
}
int main(void) {
    setupTimer();
    setupButton();

    // Bucle principal
    while (1) {
        // El sistema está en espera de interrupciones
        // Asuma que muchas otras cosas suceden aquí...
    }
    return 0;
}
```

Listing 1: Código en C para manejo de interrupciones

1.1. Cuando se genera una interrupción ¿cuál es la parte del código que se llama? (1 puntos)

1.2. ¿Qué debería de ocurrir cuando una interrupción termina su ejecución? (1 puntos)

1.3. ¿En qué parte del código se debería de establecer la prioridad de las interrupciones? Explique brevemente cómo lo haría. (2 punto)

III. Desarrollo (35 minutos)

Responda ampliamente y de manera clara los siguientes ejercicios. Muestre el procedimiento para llegar a la solución, realice un único ejercicio por página.

1. En un sistema computacional existen 5 procesos: A, B, C, D, E, con un tiempo de ejecución de 6, 7, 1, 9, 10 respectivamente. El tiempo de llegada de cada uno es 0, 4, 3, 2, 1.

   1.1. Calcule el tiempo de espera promedio utilizando el algoritmo SJF. (4 puntos)

   1.2. ¿El valor calculado es el tiempo de espera mínimo? (1 punto)

   1.3. En caso de que no sea mínimo. Proponga una calendarización donde el tiempo sea menor al calculado en el primer punto. (2 puntos)

2. Explique cuál es la diferencia entre una condición de carrera y un *deadlock*. ¿Es posible utilizar los mismos métodos para solucionar ambos problemas? (4 puntos)