UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 2

SECCIÓN A

A-2.1 - Buffering

Nombre: Alvaro Emmanuel Socop Perez **Carne:** 202000194

Guatemala, 6 de Junio del 2023

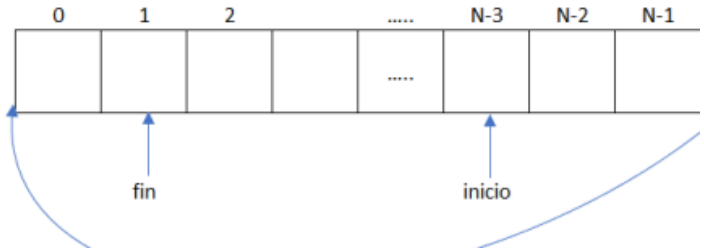
INTRODUCCION

Es indispensable contar con una arquitectura eficiente para gestionar de manera óptima los dispositivos de entrada y salida. Existen diversas formas en las que el procesador puede recibir los datos provenientes de estos módulos. Una de estas alternativas es mediante el uso de interrupciones, las cuales activan previamente una rutina registrada.

En este escenario, se implementa una interrupción específica para un dispositivo en particular, como el teclado. Cada vez que se presiona una tecla, mediante la interrupción se almacenan los códigos generados en una cola y posteriormente se procesan para obtener el código correspondiente.

PROBLEMA

Considere el problema de la aplicación de un buffering para un dispositivo de teclado, el cual es un dispositivo de caracter. En este caso el bloque de memoria buffer del sistema se ve como un arreglo de 0 a N-1, donde N es el tamaño del buffer, el cual se indiza como una cola circular, con índices de inicio y fin que se mueven dinámicamente a lo largo del buffer. Por ejemplo: en la siguiente gráfica, el inicio está en la posición N-3 y el fin en la posición 1, por lo tanto, hay 5 elementos en la cola:



En este esquema, el driver del dispositivo tiene dos componentes:

1. Rutina de servicio de interrupción: Se encarga de leer del puerto del teclado el scan code correspondiente y colocarlo en la cola del buffer. Si la cola está llena, se procesa un error (usualmente enviar un sonido al altavoz del sistema o marcar una bandera de error).

2. Rutina de procesamiento: Lee de la cola del buffer y realiza todas las tareas correspondientes: convertir el scan code en un código de un caracter visible o un control de control y notificar al proceso que esté activo. Esto es básicamente un esquema de productor consumidor con la restricción que la rutina de servicio de interrupción no puede utilizar ningún mecanismo de IPC (semáforos, monitores, etc.) que suspendan la interrupción, y por otro lado, la rutina de procesamiento puede ser interrumpida para procesar otra entrada desde el teclado.

Adicionalmente, la rutina de procesamiento es activada periódicamente por el sistema o cuando haya información para procesar. Realice dos algoritmos en C que trabajen sobre el buffer como cola circular así:

• interrupt rutina_interrupcion() Rutina de interrupción que se activa al presionar una tecla y toma la información del puerto y la escribe en la cola. No debe escribir en la cola si esta está llena.

• void procesamiento()

Rutina que se activa periódicamente o por demanda que lee de la cola del buffer, procesa y envía el mensaje al proceso activo .

Ambas rutinas deben mantener la integridad de la cola, es decir, no se pierda la información de cola ni se pierda el control del inicio y fin de la cola. Suponer las siguientes estructuras comunes a ambas rutinas:

long buffer[N] ; // N definido anteriormente

int inicio ; // marca la primera entrada de la cola

int fin ; //marca la última entrada de la cola

interrupt rutina_interrupción() ; // rutina de interrupción

void procesamiento() ; // rutina de servicio

Solución

Rutina de interrupción

La función de interrupción se encarga de leer la entrada del teclado y, por cada pulsación, genera un código utilizando el scancode correspondiente. Este código se guarda en una cola. La función de interrupción luego verifica si el siguiente índice en el buffer sería igual al índice de inicio, lo cual indicaría que la cola está llena. Si eso no ocurre, añade el nuevo elemento al buffer en la posición final y actualiza el índice de fin. El código sugerido para la función de interrupción es el siguiente:

```
void interrupción(code scancode)
{
    int siguiente = (fin + 1) % n //se incrementa en 1 para colocar posteriormente el
    //apuntador fin y el % para que vuelva a cero si
    //llega a n
    if (siguiente != inicio) //si la cola aun no está llena
    {
        Buffer[fin] = code; //almaceno el código de scancode
        fin = siguiente //incremento posición de fin
    }
    else
    {
        //la cola ya esta llena
    }
    }
```

Rutina de Procesamiento

La función de procesamiento se encarga de tomar los scancode que se guardaron en la cola durante la interrupción, hasta que el puntero de inicio sea igual al puntero de fin. En cada iteración, se toma el scancode, se guarda en una variable llamada "code" y luego se pasa a un método para obtener el carácter correspondiente. El código sugerido para la función de procesamiento es el siguiente:

```
void procesamiento()
{
        White (inicio != fin) {
                Code = buffer[inicio]; //lee el scancode de buffer y lo guarda en code
                Carácter = getCaracter(code); //este método convierte el código
                //obtenido en el carácter correspondiente
                inicio = (inicio + 1) % n; //se actualiza la posición de inicio
        }
}
```

Solución general

Aplicando los dos algoritmos se tiene el siguiente código:

```
#include <stdio.h>
#include "main.h"
 #define N 10
 //se define el tamaño de N
long buffer[N];
int inicio = 0; //se inicializa puntero de colas
int fin = 0; //se inicializa puntero de colas


void interrupcion(int scancode)
{
   int siguiente = (fin + 1) % N; //se incrementa en 1 para colocar posteriormente el
   //apuntador fin y el % para que vuelva a cero si llega a n

   if (siguiente != inicio)
   { //si la cola aun no está llena
      buffer[fin] = scancode; //almaceno el código de scancode
```

```
        fin = siguiente; //incremento posición de fin
    }
    else
    {
        printf("La cola está llena.\n"); //la cola ya esta llena
    }
}

char getCaracter(int code)
{
    // Lógica para convertir el código en el carácter correspondiente
    // Aquí se debe implementar la lógica real que mapea el scancode a un carácter
    // En este ejemplo, se utilizará un mapeo simple para los códigos 33, 44, 4B y 1C
    if (code == 33)
        return 'h';
    else if (code == 44)
        return 'o';
    else if (code == 0x4B)
        return 'l';
    else if (code == 0x1C)
        return 'a';
    else
        return ' ';
}
```

```
void procesamiento()
{
   while (inicio != fin)
   {
      int code = buffer[inicio]; //lee el scancode de buffer y lo guarda en code
      char caracter = getCaracter(code); //este método convierte el código
      //obtenido en el carácter correspondiente
      printf("%c", caracter);
      inicio = (inicio + 1) % N; //se actualiza la posición de inicio
   }
}
 //se procede a hacer una prueba ingresando algunos códigos del scancode con los
caracteres “h”, “o”, “l”, “a”

int main()
{
   interrupcion(33);
   interrupcion(44);
   interrupcion(0x4B);
   interrupcion(0x1C);

   procesamiento();

   return 0;
}
```