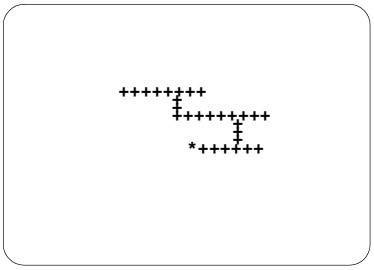
Práctica 3: (Temas 3 y 4) Dibuja serpiente con threads

- **Objetivos:**
  - **descubrir la necesidad de la concurrencia**
  - **practicar la creación y la gestión de threads**
  - **practicar la creación de threads periódicos**

Programa secuencial

Se dispone del programa `serpiente_secuencial.c` que permite dibujar una "serpiente" en la pantalla

- **cuyo movimiento se controla con las flechas del teclado**
- **cada vez que se pulsa una flecha, la serpiente avanza una posición en el sentido indicado**



Estructura del programa secuencial

```
// lazo infinito
while(1) {
  // lee la tecla pulsada por el usuario
  tecla = getchar();
  // determina el sentido del movimiento
  sentido_mov = obtenido en base a la tecla pulsada
  // lo que era cabeza se convierte en parte de la cola
  set_cursorxy(x_serpiente, y_serpiente);
  printf("+");
  // calcula la nueva posición de la cabeza
  x_serpiente=suma o resta una posición (según sentido_mov)
  y_serpiente=suma o resta una posición (según sentido_mov)
  // dibuja la cabeza en su nueva posición
  set_cursorxy(x_serpiente, y_serpiente);
  printf("*");
  // espera al próximo periodo
  clock_nanosleep(...);
}
```

el programa se bloquea hasta que el usuario pulsa una tecla

Modificación del programa secuencial

Se desea que el programa se comporte de forma diferente:

- **la "serpiente" deberá desplazarse a velocidad constante por la pantalla sin esperar la pulsación de las teclas**
- **como antes, el sentido de su desplazamiento será el correspondiente a la última tecla pulsada**

Solución propuesta:

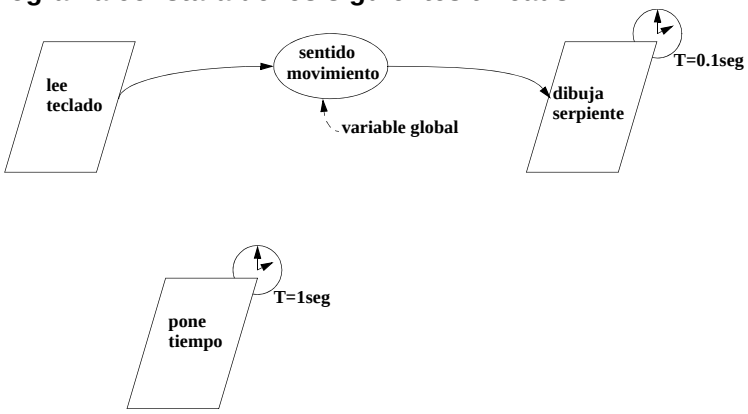
- **transformar la aplicación secuencial en una aplicación concurrente según la arquitectura mostrada a continuación**

En este caso tan sencillo el problema también podría resolverse utilizando una llamada al teclado no bloqueante

- **pero en el caso general una solución de este tipo puede no estar disponible**

Arquitectura

El programa constará de los siguientes threads:

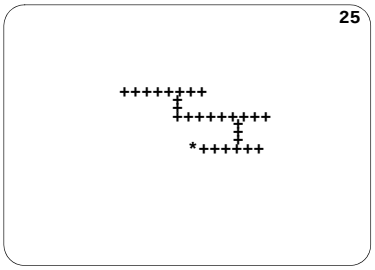


Funcionalidad de cada thread

- **Thread lee_teclado**
  - **lee cada tecla pulsada y la almacena en la variable global `tecla_pulsada`**
- **Thread dibuja_serpiente**
  - **dibuja la serpiente avanzando una posición cada periodo**
  - **el sentido del avance está determinado por la última tecla pulsada por el usuario**
- **Thread pone_tiempo**
  - **muestra el tiempo de ejecución (en segundos) en la esquina superior derecha de la pantalla**

Desarrollo

1. Compilar y ejecutar el programa proporcionado y observar su funcionamiento
2. Modificar el programa para convertirlo en una aplicación concurrente según la arquitectura propuesta
3. Añadir el thread "pone tiempo"
   - muestra los segundos que vamos ejecutando el programa en la esquina superior derecha de la pantalla



Gestión de la consola

MaRTE OS proporciona funciones para manejar la consola en modo texto en `<misc/console_management.h>` (en el directorio `marte/x86_arch/include`)

- Posicionamiento del cursor:
  `set_cursorxy()`
- Cambio del color del texto y del fondo
  `set_text_background_color(), set_text_color()`
- Configuración del teclado para que no espere la pulsación de la tecla enter
  `set_raw_mode()`
- Configuración de la consola para que no muestre las teclas pulsadas
  `disable_echo()`

Entrega

Enviar por e-mail al profesor (`aldeam@unican.es`):

- Código desarrollado