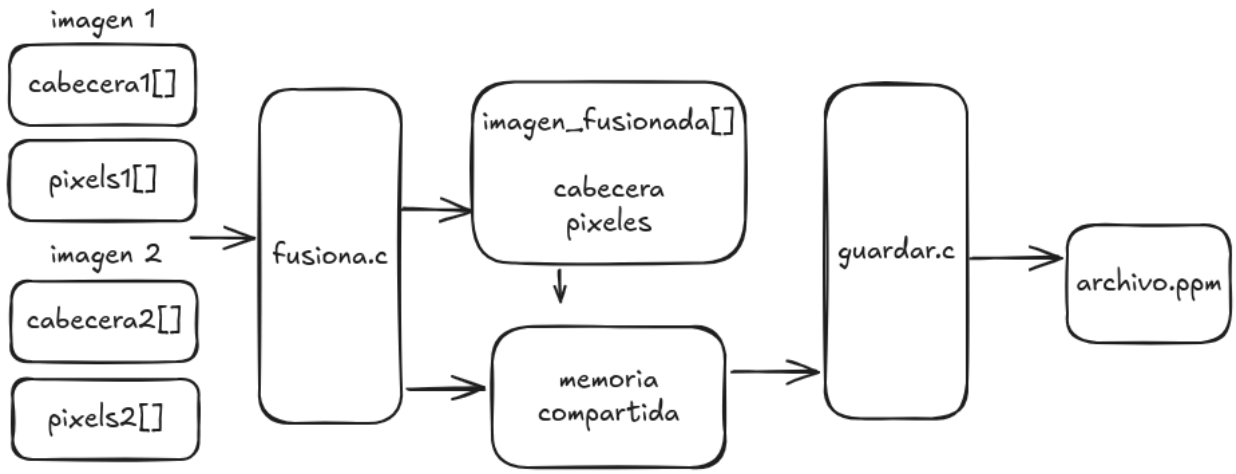
Sistemas Operativos I. Segundo examen parcial. 2025. Facultad de Informática. Universidad Nacional del Comahue

NOTA: para la resolución del examen se puede utilizar como material de referencia sus archivos de solución de los trabajos prácticos en sus cuentas Linux, o el material disponible en la web de la materia. http://se.fi.uncoma.edu.ar/so

En ningún caso se pueden utilizar herramientas online (google, chat, gpt, deepseek, etc).

Ejercicio 1. Implementar en Linux dos programas, que fusiona dos imágenes en formato ppm y las guarda en un archivo. Los dos programas realizarán esquemáticamente lo siguiente:



Utilice los fuentes en este archivo .tar.gz: http://se.fi.uncoma.edu.ar/so/misc/ej1-p2.tar.gz
En fusiona.c ya se encuentra "incluido" los arreglos que tienen las imágenes:

- cabecera1[] es el arreglo de la cabecera de la imagen 1.
- cabecera2[] es el arreglo de la cabecera de la imagen 2. //es igual a la cabecera1
- pixels1[] es el arreglo que contiene los píxeles de la imagen 1 (un byte = un píxel).
- pixels2[] es el arreglo que contiene los píxeles de la imagen 2 (un byte = un píxel).

a. Desarrollar el programa fusiona.c que fusiona las dos imágenes en un nuevo arreglo y la comparte por memoria compartida. Este programa COLOCA en un arreglo la fusión, de esta manera (este arreglo con la imagen fusionada contendrá):
   - La cabecera de la imagen.
   - El pixel 1 de la imagen 1.
   - El pixel 2 de la imagen 2.
   - El pixel 3 de la imagen 1.
   - El pixel 4 de la imagen 2.
   - etc…

   Es decir, los píxeles impares serán de la imagen 1, los píxeles pares de la imagen 2.

   La cabecera tiene 15 bytes.
   Cada imagen tienen 270.000 bytes (pixeles).

Luego, el programa solicita memoria compartida al sistema operativo (con el tamaño adecuado), coloca la imagen fusionada en la memoria compartida, y finaliza. Nota: Para copiar muchos bytes (por ej. un arreglo grande) a una memoria con un puntero se puede usar memcpy() (man memcpy).

b. Desarrollar un segundo programa guardar.c que solicita un nombre de archivo al usuario y guarda la imagen fusionada, que se encuentra en la memoria compartida, en ese archivo.

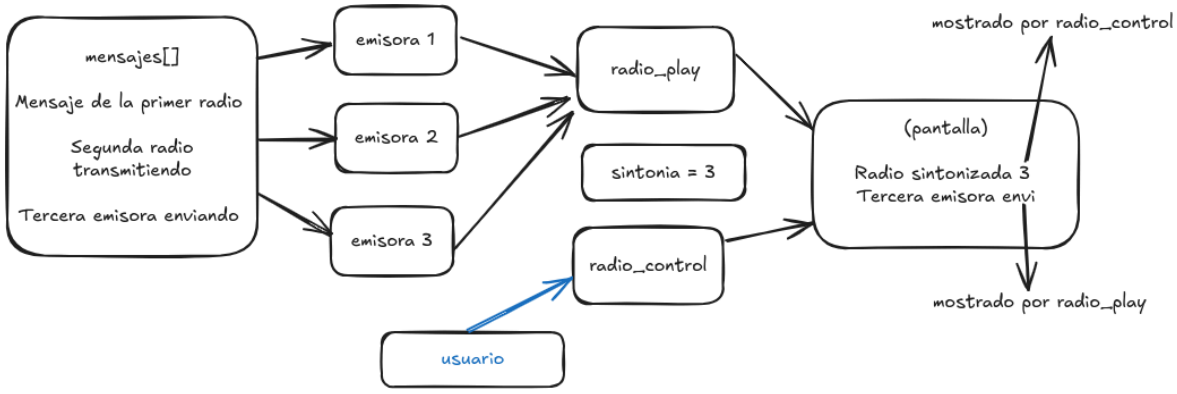
Ejemplo para guardar 10 bytes a un archivo:

```
int f = open("mi_archivo”,  O_RDWR|O_CREAT,  0666);
write(f,  buffer, 10);
close(f);
```

Imagenes originales

Imagen fusionada

Ejercicio 2. Se requiere implementar en XINU un sistema con varias emisoras. Y una radio compuesta de dos procesos: radio_control y radio_play.



Las emisoras deben enviar uno de estos mensajes:

```
const char * mensajes[] = {
    " Mensaje de la primer radio \n",
    " Segunda radio transmitiendo \n",
    " Tercera emisora enviando \n",
    };
```

Cada emisora envía su mensaje con un retardo entre letra y letra:

La emisora 1 envía cada letra de su mensaje cada 600 ms.
La emisora 2 envía cada letra de su mensaje cada 500 ms.
La emisora 3 envía cada letra de su mensaje cada 888 ms.

a. Desarrolle un programa “emisora”, que a partir de un número de emisora que obtiene como argumento envía su mensaje a la radio (al proceso radio_play detallado mas debajo). El mensaje se envía de una letra a la vez vía pasaje de mensajes, y el “retardo” entre letra y letra enviada también lo obtiene como argumento. Cuando finaliza empieza nuevamente a enviar el mismo mensaje.

   Los mensajes[] son globales, y para acceder, por ejemplo, al caracter de la posición 3 de la emisora 2 se puede acceder con mensajes[2][3].

   Cada letra/caracter se envía con un retardo de X ms (el retardo X es obtenido como argumento).

   Además, cada mensaje enviado por la emisora lleva el número de la emisora en el mismo mensaje: Como en XINU un mensaje es un entero (2 o mas bytes), se puede colocar el número de la emisora y la letra a enviar en el mismo mensaje (un byte es el número de la emisora y otro byte que representa la letra ASCII). Por ejemplo, se puede utilizar el operador desplazamiento tal cual aparece en el siguiente ejemplo. Ejemplo:

```
int n = 3;          // ejemplo de nro de emisora
char c = 'A';       // ejemplo de letra ASCII a enviar
int msg = (n << 8) | c;    //  colocamos en msg el nro de la emisora y el ASCII
// en la variable msg quedó el mensaje ya armado listo para enviar
```

   El mensaje debe ser enviado utilizando las primitivas de pasaje de mensajes a radio_play. El pid de radio_play tambien es obtenido por la emisora como argumento.

   En tiempo de ejecución se crean 3 procesos de este programa.

b. Desarrollar radio_play, el cual es un proceso que emula la radio. La radio está sintonizada para reproducir una unica emisora. radio_play recibe todos los mensajes de todas las radios y sólo muestra la letra de su sintonía. Si el mensaje está codificado como en el inciso a., conocer la letra y la emisora es similar. Use tal cual como en el siguiente ejemplo:

```
// msg es el mensaje recibido. Aquí lo definimos sólo para el ejemplo
int msg = 0x0341;        // mensaje de ejemplo: emisora 3 (0x03), y letra ‘A’ (0x41)
int n = (msg >> 8);      // obtenemos el numero de emisora
char c = (char) (msg & 0xFF);  // obtenemos la letra del mensaje
// en la variable n quedó el nro de la emisora
// en la variable c quedó la letra recibida del mensaje
```

Para mostrar una letra radio_play tiene que utilizar la pantalla de la consola, la cual es utilizada tambien por radio_control. Utilice algun mecanismo de sincronización (no utilice mutex, con un semáforo binario bien utilizado debería alcanzar por ejemplo).

c. Desarrollar radio_control, el cual es el programa principal (llamado desde el shell). Este es el programa que utiliza el usuario.
radio_control pone a ejecutar las tres emisoras y radio_play. Luego, interactua con el usuario:
   - Si el usuario presiona ‘q’ finaliza: finaliza todos los procesos, restablece la consola a modo normal si es necesario y finaliza.
   - Si el usuario presiona ‘1’ radio_control establece como sintonia global de la radio la emisora 1.
   - Si el usuario presiona ‘2’ radio_control establece como sintonia global de la radio la emisora 2.
   - Si el usuario presiona ‘3’ radio_control establece como sintonia global de la radio la emisora 3.

   Cuando radio_control obtiene un comando del usuario, muestra por pantalla,
   en una NUEVA LINEA, la sintonia actual de la radio. Por ejemplo, un mensaje así:

   La sintonia actual es la emisora 2

   Atención: la pantalla será utilizada entre radio_play y radio_control. Utilice por ejemplo un semáforo binario para sincronizar su uso (cada vez que un proceso quiera usar la pantalla debe esperar a que se libere si está en uso).