Todos los programas que se piden deberán construirse en **C** y realizar su labor **utilizando únicamente las llamadas al sistema tratadas en los guiones de prácticas.**

PREGUNTA 1 . [2,5 puntos] Lea la documentación que se le proporciona sobre el archivo `maps`.

Construya un programa que se llamará `conSoporte.c`. Cuando se ejecute desde el shell bash, obtendremos con getppid() el pid de dicha ejecución del intérprete de órdenes padre de `conSoporte`. Formamos la ruta del archivo que contiene el mapa de memoria de dicho proceso padre:

```
char mempadre[60];
sprintf(mempadre, "/proc/%d/maps", getppid());
```

Este programa `conSoporte.c` deberá lanzar un proceso con lo que tendremos dos procesos, padre e hijo, que se comunicarán con un cauce (con nombre o sin nombre, decídalo usted).

El proceso hijo debe llamar a `exec` para lanzar una ejecución de la orden `cat` que obtenga el contenido del archivo `/proc/<pid-del-padre>/maps.` Debe conseguirse que estos datos vayan al cauce, de donde los tomará el padre.

El proceso padre leerá carácter a caracter del cauce y procesará esta secuencia de forma que cuando el carácter leido sea "/" es porque comienza la ruta de un archivo aludido en el archivo maps, y deberá llevar a la salida estándar el resto de la línea (hasta encontrar un carácter que sea "\n").

Para cada carácter leido el procesamiento podria seguir esta pauta:

```
if (strcmp(c,"/")==0) deboescribir=1;
if (deboescribir) {...
.....escribir el caracter que acabamos de leer ...}
if (strcmp(c,"\n")==0) deboescribir=0;
```

PREGUNTA 2 . [2,5 puntos] Descargue del trabajo asociado a esta prueba el archivo `anilloNprocesos.c` que crea un anillo de N procesos siendo N el número que pasamos como argumento. Incluya el código adecuado para conseguir lo siguiente.

Antes de crear ningún proceso debe escribirse en "su salida" el caracter "0" como primer dato que deberá circular en el anillo. Cada proceso del anillo deberá leer 1 carácter de "su entrada". Si es menor estricto que 9, sumarle 1 y escribirlo en pantalla y en "su salida". Si ya ha llegado a 9, terminar.

Los procesos deberán ir mostrando mensajes que permitan visualizar que el anillo está funcionando.

PREGUNTA 3. [5 puntos] El programa se llamará `verEnlaces.c` y tendrá como argumento la ruta de un archivo. El programa deberá explorar en su totalidad el árbol de archivos y mostrar en pantalla aquellas rutas que sean un enlace duro al dado como argumento.

Minimice la búsqueda haciendo uso del valor que tome el contador de enlaces para la ruta dada como argumento.

Programe la actuación adecuada ante las distintas situaciones de error que puedan ocurrir.