Problema 1 (3 puntos).
Hemos escrito estos 2 programas en UNIX:

`programa1:`

padre.c

```
main()
{
      int id,estado;

      close(1);
      creat("salida.dat",0777);
      write(1,"linea de texto n 1\n",19);
      if((id=fork())==0) {
            execl("hijo","hijo",0);
            exit(1);
      }
      else {
4\n",19);
            while(wait(&estado)!=id);
            write(1,"linea de texto n 5\n",19);
            exit(0);
      }
}
```

hijo.c

```
#include <fcntl.h>

main() {
      int idf;

      write(1,"linea de texto n 2\n",19);
      idf=open("salida.dat",O_WRONLY);
      write(idf,"linea de texto n 3\n",19);
      write(idf,"linea de texto n 4\n",19);
      close(idf);
      close(1);
      exit(0);
 }
```

`programa2:`

```
 #include <unistd.h>
main()
   {
         int idf;

close(1);
creat("salida.dat",0777);
         write(1,"linea de texto n 1\n",19);
         write(1,"linea de texto n 2\n",19);
         idf=dup(1);
         lseek(idf,0L,SEEK_SET);
         write(1,"linea de texto n 3\n",19);
         write(idf,"linea de texto n

write(1,"linea de texto n 5\n",19);
         exit(0);
   }
```

Explica cuál será el contenido del fichero "salida.dat" en ambos casos y por qué.

Problema 2 (4 puntos).

Diseñar un programa llamado "thead" cuyo funcionamiento describimos a partir del siguiente ejemplo de utilización:

merlin_$ P1 | thead n F1 | P2

donde P1, P2 y thead son programas y F1 es el nombre de un fichero (pasado como segundo parámetro a "thead").

Como resultado de la ejecución de la linea anterior:

- P2 recibe en su entrada estandar las n primeras lineas procedentes de la salida estandar de P1 (n es pasado como primer parámetro a "thead" y cuando n=0, P2 recibe en su entrada estandar la salida estandar de P1).
- En F1 queda escrita la salida estandar de P1 (para cualquier valor de n).

Se pide:

Apartado 1. Implementar el programa "thead" en lenguaje C, usando sólo llamadas al sistema UNIX para la entrada/salida.

Apartado 2. Utilizando el programa "thead" del **apartado 1**, escribir las lineas de comandos que ejecuten de la forma más eficiente posible lo que se pide a continuación:

- La salida estandar de P1 quede escrita en F1 y F2 y sus n primeras lineas sean usadas como entrada estandar de P2.
- La salida estandar de P1 quede escrita en F1 y sus n primeras lineas aparezcan en pantalla.
- La salida estandar de P1 quede escrita F1 y en F2.

Problema 3 (3 puntos).

Hemos escrito el siguiente programa en UNIX:

programa.c:

```
#include <signal.h>

void sig_alrm()
{
        write( 2, "RUTINA\n", 7 );
        return;
}

main()
{
 int fd[2], n;
 char mensaje[10], *s;

        s="MENSAJE\n";
        signal(SIGALRM,sig_alrm);
        pipe(fd);
        if( fork() == 0 ){
                close(fd[1]);
                alarm(3);
                while ((n = read( fd[0], mensaje, 10)) > 0 );
                alarm(0);
                exit(0);
        };
        close(1);
        dup(fd[1]);
        close(fd[0]);
        close(fd[1]);
        while(1)        write( 1, s, 8 );
}
```

Explica cuál será el resultado de la ejecución de "programa" y por qué.