Problema 1 (3 puntos).

Hemos escrito escrito y compilado en nuestro directorio de trabajo los siguientes programas para UNIX:

`primero.c`

```
#include <stdio.h>

main()
{
      int id, estado;

      close(1);
      creat("salida.dat",0777);
      write(1,"linea\n",6);
      if(( id = fork() ) == 0 ){
            execl("segundo","segundo",0);
            exit(1);
      }
      else{
            while( wait(&estado) != id );
            write(1,"linea\n",6);
            exit(0);
      }
}
```

`segundo.c`

```
#include <stdio.h>
#include <sys/file.h>

main()
      {
            int idf, id;

      idf=open("salida.dat",O_WRONLY);
      write(idf,"linea\n",6);
            id = dup(1);
      lseek(id, -6L, SEEK_CUR);
            write(id,"linea\n",6);
            write(idf,"linea\n",6);
            close(idf);
            close(id);
            close(1);
            exit(0);
      }
```

Suponiendo que ejecutamos desde nuestro directorio de trabajo el programa `primero`:

- Explica cuál será el contenido final del fichero "`salida.dat`" y por qué.

- Sustituye la línea `lseek(id, -6L, SEEK_CUR);` del programa `segundo.c` por otra que tenga el mismo efecto que esta. Escribe al menos 2 sustituciones posibles.

Problema 2 (3,5 puntos).

Diseñar un programa (`shtwo.c`) que sea capaz de ejecutar dos aplicaciones en paralelo cuyos nombres recibirá en la línea de comandos. Las aplicaciones a ejecutar podrán tener parámetros y entre la especificación de la primera aplicación y la de la segunda habrá un símbolo +.

Ejemplo de utilización: **merlin_$** `shtwo ps -u teresa + ls -l *.c`

Deberá ejecutar "`ps -u teresa`" en paralelo con "`ls -l *.c`".

El programa `shtwo` no debe dejar ningún proceso `zombie` y deberá informar al final de su ejecución de la finalización de cada una de las aplicaciones lanzadas mediante una línea de mensaje hacia pantalla.

Responde ahora a la siguiente pregunta:

- Una vez diseñado el programa anterior `shtwo`, explica cuál sería el resultado de ejecutar la siguiente línea de comandos :

merlin_$ `shtwo date + who | wc -l`

Nota: Suponer que la salida del comando `who` es la siguiente:

merlin_$ `who`

```
juanm       pty/ttys0   Jun  9      08:03
teresa      pty/ttys1   Jun  9      10:18
gascon      pty/ttys4   May 29      14:42
```

Problema 3 (3,5 puntos).
Hemos escrito y compilado en nuestro directorio de trabajo el siguiente programa para UNIX:

`programa.c:`

```
#include <stdio.h>
#include <signal.h>
#include "error.h"  /* contiene definición de syserr para imprimir mensajes de error */
#define MSGSIZE 13

char *msg1 = "El mensaje 1\n";
char *msg2 = "El mensaje 2\n";

void rutina(){
 kill( getppid() , SIGKILL );
}
main()
{
 char inbuff[MSGSIZE];
 int p[2], j;

 signal( SIGALRM , rutina );
 if( fork() ){
       if( pipe(p) < 0 ) syserr("Error en la llamada pipe\n");
       write( p[1], msg1 , MSGSIZE );
       write( p[1] , msg2 , MSGSIZE );
       for ( j=0; j<2; j++ ){
             read( p[0] , inbuff , MSGSIZE );
             write( 1 , inbuff , MSGSIZE );
       }
 }
 else{
       if ( alarm(5) )  write( 1, msg1 , MSGSIZE );
       write( 1 , msg2 , MSGSIZE );
       alarm(0);
 }
}
```

- Explica qué hace nuestro `programa` y cuál sería su resultado final.
- Qué variaria del resultado si añadiésemos al final del código del proceso padre la siguiente línea :

```
wait(NULL);
```