Práctica 2 (1.5 ptos)

Mini−shell

1.− Objetivo

Construir una pequeña librería con una función que permita trocear cadenas y emplearla en el diseño de una versión simplificada de una shell.

2.− Librería `fragmenta.a`

La función que se debe construir debe cumplir la siguiente documentación:

```
FRAGMENTA(3)                                          FRAGMENTA(3)

NAME
    fragmenta, borrarg − Utilidades para trocear cadenas

SYNOPSIS
   #include "fragmenta.h"

   char **fragmenta(const char *cadena);
   void borrarg(char **arg);

DESCRIPTION
    fragmenta() crea una array de char* con tantos elementos como el número
de fragmentos que encuentre en cadena más uno, el último vale siempre NULL y es el
único con tal valor, con lo que sirve para determinar el final del array. Cada elemento
de este array es un puntero a una zona de memoria donde se encuentra uno de los
fragmentos de cadena y en el mismo orden.. Los fragmentos de cadena vienen
definidos por estar separados por uno o más espacios, pudiendo terminar en un fin de
línea.
    borrarg() libera la memoria asociada con el puntero arg, así como las
zonas de memoria apuntadas por cada uno de los char* apuntados por arg y
colocados uno tras otro hasta uno que valga NULL.

RETURN VALUES
    fragmenta() devuelve el puntero al array creado o NULL si no puede
realizar su función.
```

El contenido del header es simplemente:

```
/* Contenido de fragmenta.h */
char **fragmenta(const char*);
void borrarg(char **);
/* Fin de fragmenta.h */
```

3.− Mini−shell

```
MSH(1)                                                    MSH(1)

NAME
    msh − Mini Shell

SYNOPSIS
    msh

DESCRIPTION
    Las funcionalidades de esta shell son:
```

- Ejecución de comandos con un número indeterminado de argumentos.
  Ejemplo:
  ```
  msh%> cp −r sources backup
  ```
- Redirección de la salida estándar a fichero mediante `>`
  Ejemplo:
  ```
  msh%> ls −alF > listado
  ```
- Redirección de la entrada estándar de fichero mediante `<`.
  Ejemplo:
  ```
  msh%> wc −l < listado
  ```
- Redirección simultánea de entrada y salida estándar en cualquier orden.

4.− Ficheros

En el directorio `$(HOME)/solucion/prac2` debe encontrarse un `Makefile` así como todos los ficheros `.c` y `.h` necesarios para crear `fragmenta.a` y `msh`. La acción por defecto del `Makefile` (la cual debe funcionar con solo hacer `make` en ese directorio) debe ser crear ambos. Igualmente debe responder correctamente a `make fragmenta.a` y a `make msh`.

En el directorio `$(HOME)/../ejemplos/prac2` se pueden encontrar el fichero `fragmenta.h` así como unos ejemplos de `fragmenta.a` y `msh` para comprobar cuál es el funcionamiento deseado. Igualmente se deja disponible `msh_main.o` que es el compilado de las función principal de msh para poder ser linkado con otra librería `fragmenta.a`.

Para la corrección de la práctica se borrarán todos los ejecutables, se hará un touch a todos los ficheros fuente y se recompilará mediante el `Makefile`.

5.− Aplicaciones, funciones y llamadas al sistema útiles

`fork(2), execvp(3), wait(2), open(2), close(2), dup2(2).`