man(ISO) Especificación de extrae_fichero man(ISO)

NOMBRE

extrae_fichero - extrae un fichero regular, directorio o enlace de un fichero con formato gnu tar.

SINOPSIS

```
#include "s_mytarheader.h"
int extrae_fichero(char * f_mytar, char * f_dat);
```

DESCRIPCIÓN

La función extrae_fichero extrae un fichero regular, directorio o enlace simbólico (f_dat) de un fichero tar gnu f_mytar, recuperando su información de control original como permisos, propietario y grupo.

Si el archivo f_mytar existe, no está dañado y cumple con el formato gnu tar, entonces se procederá a la extracción de f_dat.

El fichero (fichero regular, directorio o enlace) a extraer (nombre, datos,...) corresponde con la información indicada en un "file system object" de f_mytar cuyo nombre coincide con el valor de f_dat.

Los ficheros a extraer del archivo tar, se diferencian en la cabecera mediante el campo *typeflaq*.

- Si ese objeto corresponde a un elemento de tipo directorio, se crea el directorio indicado en el campo name de c_header_gnu_tar.

Si el nombre del elemento (contenido en el c_header_gnu_tar) es un fichero y contine una ruta(directorio), si la ruta no existe, debe crearla.
Por ejemplo, si el nombre del archivo contenido en el c_header_gnu_tar correspondiente a index, es dir1/fdatos.dat, y no existe el directorio dir/ debe crear el directorio dir/ y sobre ese directorio crear el archivo fdatos.dat (extrae directorio y los ficheros **regulares** que hubiera en el **primer nivel**)

- Si lo que contine es un fichero regular, se extrae como tal.

- Si corresponde con un enlace simbólico, se extrae y se crea el enlace. Sin embargo, no se comprueba si el enlace está roto o es válido.

VALOR DE RETORNO

Si todo funciona correctamente, extrae_fichero devolverá cero. En caso contrario no creará el fichero a extraer y retornará los errores indicados en el apartado de ERRORES.

ERRORES

E_OPEN (-1)

No se puede abrir f_mytar.

E_TARFORM (-3)
f_mytar no tiene el formato de gnu tar.
E_NOEXIST (-2)
f_mytar no contiene el fichero de nombre f_dat.
E_CREATDEST (-4)
No se puede crear el directorio f_dat o algunos de los ficheros contenidos dentro
E_DIR1 (-5)
No se puede crear el directorio indicado en (index) .

NOTAS

Ejemplo de utilización:
Se supone que se utiliza compilación separada y el código (en lenguaje C) de la función extrae_fichero se encuentra en un fichero diferente.

En el ejemplo de uso, la función extrae el fichero de nombre "./fichero3.dat" se encuentra en el fichero "./ejemplo.tar".
- "./ejemplo.tar" deber ser un fichero con formato gnu tar. (sino ocurrirían errores; ver “errores”)

```
#include "s_mytarheader.h"
extern int extrae_fichero(char * f_mytar, char * f_dat);
...
int ret;
n = extrae_fichero("./ejemplo.tar", "./fichero3.dat");
if (n < 0) // Error
{
   ....
}
```

FORMATO GNU TAR

Atención (sobre la cabecera): Nótese que el campo *typeflag* sigue el estándar **POSIX “ustar”** el cual define los tipos de ficheros en caracteres de 0 al 7, en vez de letras como el define el estándar GNU TAR actual. Sin embargo, el resto de la cabecera se rige por el estándar GNU.

```
/**
* @file s_mytarheader.h
* @author   Gonzalo Alvarez - Dpto. ATC/KAT - UPV-EHU
* @date     10/02/2023
* @brief    Include file with struct c_header_gnu_tar
* @details  A header file with the definition of c_header_gnu_tar of
*         gnu tar file format
*         (1) source: https://manpages.ubuntu.com/manpages/bionic/en/man5/tar.5.html:
*         "... A tar archive consists of a series of 512-byte records.  Each file system object requires a
```

```
*       header record which stores basic metadata (pathname, owner, permissions, etc.) and zero or
*       more records containing any file data.  The end of the archive is indicated by two records
*       consisting entirely of zero bytes.
*       ..."
*       The last block of file system object is completed with zero bytes.
*
*       Size of a .tar file
*       --------------------
*       The size of a .tar file has to be a multiple of 10K bytes (20 x 512 blocks).
*       To accomplish this, the tar file is populated with the 512-byte zero blocks needed to make
*       its size a multiple of 10KB.
*       (https://www.gnu.org/software/tar/manual/html_node/Blocking-Factor.html)
*
*       GNU Tar File Format
*       ++++++++++++++++++++++++
*       + Header Record   0   +
*       +----------------------+
*       +  Data File 0       +
*       + 0... N blocks of    +
*       + of 512 bytes       +
*       ++++++++++++++++++++++++
*       + Header Record   1   +
*       +----------------------+
*       +  Data File 1       +
*       + 0... N blocks of    +
*       + of 512 bytes       +
*       ++++++++++++++++++++++++
*       +       ...          +
*       ++++++++++++++++++++++++
*       + Header Record  N-1  +
*       +----------------------+
*       +  Data File N-1     +
*       + 0... N blocks of    +
*       + of 512 bytes       +
*       ++++++++++++++++++++++++
*       +    End of archive   +
*       +    2 blocks of      +
*       +    of 512 bytes     +
*       ++++++++++++++++++++++++
*       +    Padding Data     +
*       +  to  tar file size  +
*       +  equal to N * 10K   +
*       +  block size (zeros) +
*       ++++++++++++++++++++++++
*
*/

#define ERROR_OPEN_DAT_FILE (2)
```

```
#define ERROR_OPEN_TAR_FILE (3)
#define ERROR_GENERATE_TAR_FILE (4)
#define ERROR_GENERATE_TAR_FILE2 (5)

#define FILE_HEADER_SIZE     512
#define DATAFILE_BLOCK_SIZE  512
#define END_TAR_ARCHIVE_ENTRY_SIZE  (512*2)
#define TAR_FILE_BLOCK_SIZE  ((unsigned long) (DATAFILE_BLOCK_SIZE*20))

#define HEADER_OK (1)
#define HEADER_ERR (2)


struct c_header_gnu_tar {
      char name[100];           // file name
      char mode[8];             // stored as an octal number in ASCII.
      char uid[8];              // (idem)
      char gid[8];              // (idem)
      char size[12];            // (idem)
      char mtime[12];           // (idem)
      char checksum[8];         // see (1).
      char typeflag[1];         // see (1).
      char linkname[100];       // see (1).
      char magic[6];            // see (1).
      char version[2];          // see (1).
      char uname[32];           // user name
      char gname[32];           // group name
      char devmajor[8];         // not used (zeros)
      char devminor[8];         // not used (zeros)
      char atime[12];           // stored as an octal number in ASCII.
      char ctime[12];           // stored as an octal number in ASCII.
      char offset[12];          // not used (zeros)
      char longnames[4];        // not used (zeros)
      char unused[1];           // not used (zeros)
      struct {
            char offset[12];
            char numbytes[12];
      } sparse[4];              // not used (zeros)
      char isextended[1];       // not used (zeros)
      char realsize[12];        // not used (zeros)
      char pad[17];             // zeros
};


/**
* end @file s_mytarheader.h
**/
```

COMPATIBILIDAD

extrae_fichero() debería funcionar en cualquier sistema UNIX.

VEASE TAMBIEN

create_mytar(ISO), inserta_fichero(ISO).

AUTOR

Grupo ISO-1-10 (Telmo Sendino, Marcos Chouciño y Mikel Amundarain)