man(ISO) Especificación de la función inserta_fichero man(ISO)

NOMBRE

inserta_fichero - inserta un fichero regular, directorio o enlace a un fichero con formato gnu tar.

SINOPSIS

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

DESCRIPCIÓN

La función inserta_fichero inserta un fichero regular, directorio o enlace simbólico (f_dat) a un fichero de formato tar gnu (f_mytar), almacenando su información de control como permisos, propietario y grupo.

Si f_mytar contiene ficheros con un formato gnu tar correcto, se añadirá el fichero fdat al final del último bloque de datos del último fichero incluido en f_mytar (es decir justo dónde comienza el EOF).

Si f_mytar no existe o está vacío, la función crea lo crea y le añade el fichero f_dat tal y como corresponda dependiendo del tipo de fichero que sea (lea 'detalles').

El formato de f_mytar se describe en la estructura c_header_gnu_tar (ver s_mytarheader.h).

DETALLES

Para diferenciar el tipo de fichero en la inserción, internamente se usa la llamada ***stat()***.

Si *f_dat* es de tipo fichero regular:

La función inserta_fichero inserta el fichero *f_dat* en el fichero *f_mytar*. Primero escribe una cabecera *c_header_gnu_tar* con la información correspondiente del fichero y si procede, escribe por bloques los datos del fichero en cuestión.

Si *f_dat* es de tipo Directorio:

Debe introducir primero un elemento de nombre *f_dat* pero sin datos. Solo se añade una cabecera (*c_header_gnu_tar*) con la información de control correspondiente.

Seguido, la función inserta solamente **los ficheros regulares** contenidos en el primer nivel del directorio *f_dat*. Estos tendrán de nombre (en el campo *name* de la cabecera) *f_dat/xxx* dónde *xxx* es el nombre de cada fichero regular encontrado en el directorio.

Si *f_dat* es de tipo enlace simbólico:

Se inserta una cabecera con la información correspondiente. En el campo *name*, se almacena el nombre del enlace y en el campo *linkname* el nombre del fichero al cual se enlaza.

Si *f_dat* no fuera de ninguno de estos tipos: Se reporta el error por la salida estándar de errores y se devuelve E_DESCO (ver errores).

VALOR DE RETORNO

Si todo funciona correctamente, inserta_fichero devolverá el número correspondiente del último elemento insertado dentro del fichero f_mytar (número de ficheros contenidos en f_mytar).

En caso contrario no actualizará/creará el fichero f_mytar y retornará los errores indicados en el apartado de ERRORES.

En el caso de que el error sea E_DESCO o en errores muy particulares, la función también puede escribir un mensaje aclarativo de error por la salida de estándar de errores.

ERRORES

E_OPEN_TAR (-1)

No se puede abrir o crear f_mytar.

E_OPEN_DAT (-2)

f_dat hace referencia a un archivo (o directorio) que no se puede abrir.

E_TARFORM (-3)

f_dat no tiene el formato de gnu tar.

E_DESCO (-99)

Otro tipo de errores

NOTAS

Nota 1: El programa que utilice esta función, deberá informar por la salida de error estándar un mensaje indicando el tipo de error.

Ejemplo de utilización:

Se supone que se utiliza compilación separada y el código (en lenguaje C) de la función inserta_fichero la cual se encuentra en un fichero diferente.
En el ejemplo de uso, la función inserta el fichero "./fichero3.dat" 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 inserta_fichero(char * f_mytar, char * f_dat);
...
unsigned int ret;
```

```
n = inserta_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

inserta_fichero() debería funcionar en cualquier sistema UNIX.

VEASE TAMBIEN

create_mytar(ISO), extrae_fichero(ISO).

AUTOR

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