ISO-2-2

PROYECTO MYPACKZIP

DOCUMENTO RESUMEN

Asignatura:
Introducción a los Sistemas Operativos
2º Ingeniería Informática, EHU/UPV
Profesores:
Gonzalo Alvarez Balbas
Mikel Larrea Alava

Asier Septién Molano
✉aseptien001@ikasle.ehu.eus
Eneko Pizarro Liberal
✉epizarro001@ikasle.ehu.eus
Katrin Mariova Lukanova
✉kmariova001@ikasle.ehu.es

1 de abril del 2022

RESUMEN

Este documento pretende resumir la implementación y verificación del proyecto mypackzip; el cual trata sobre la creación de un fichero regular que contiene directorios, enlaces simbólicos y otros ficheros regulares los cuales están comprimidos, con el propósito de interiorizar el sistema de ficheros de los Sistemas Operativos Unix, linux para ser exactos.

Este proyecto, además, requiere de funciones específicas para poder crear y hacer uso de mypackzip, las cuales también estarán brevemente explicadas a continuación.

El punto de partida ha sido la estructura de datos s_headers y las especificaciones tipo man que sirven como documento que describen el correcto comportamiento que cada función que compone mypackzip debe de cumplir.

Estás se encuentran actualizadas en el directorio “especificaciones” adjunto a este fichero y son las siguientes:

- Especificación-inserta_fichero
- Especificación-extrae_fichero
- Especificación-ls_mypackzip

ESTRUCTURA DEL DIRECTORIO DEL PROYECTO

La estructura del proyecto está presentada en un **readme.txt** adjunto a este documento.

FUNCIONES

A pesar de ser tres funciones principales, estas están compuestas por funciones más específicas, y su especificación e implementación se ha ido actualizando a medida que las iteraciones del proyecto han sido completadas.

Es por eso por lo que vamos a resumir cada función por iteración.

ESTRUCTURA DE LA PROGRAMACIÓN (EN C)

- **mypackzip_functions.h**
    - Este fichero regular contiene la definición de todas las funciones de mypackzip a implementar en “mypackzip_functions.c”.
- **mypackzip_functions.c**
    - Implementación de todas las funciones definidas en “mypackzip_functions.c”.
    - Este fichero no contiene ninguna función a ejecutar.
- **insertmypackzip.c, extractmypackzip.c y lsmypackzip.c**
    - Ficheros de c que llaman a sus funciones respectivas de “mypackzip_functions.c”.
    - Estos contienen el main, y son los que se incluyen en la última fase de compilación para finalmente hacer el ejecutable que recibe parámetros.
    - Además, estos ficheros también controlan la salida de errores, mensajes de precaución y de correcto funcionamiento (dependiendo del valor de retorno de la función a la que llaman).
- **insertmypackzip, extractmypackzip y lsmypackzip**
    - Ejecutables de los ficheros “.c” previamente compilados.
    - Estos son los que se han de llamar para hacer uso de las funciones de mypackzip.
    - Cada uno acepta diferentes argumentos y cumplen su respectiva especificación descrita en los pseudo-man adjuntos en el directorio adjunto a este fichero.
    - He aquí los argumentos necesarios para poder utilizar cada ejecutable:
        - **./insertmypackzip f_mypackzip f_dat índice**
        - **./extractmypackzip f_mypackzip índice**
        - **./lsmypackzip f_mypackzip**

ITERACIÓN - P05.3

En esta iteración se ha implementado insertar y extraer mypackzip para ficheros regulares.

*Todas las funciones están comentadas en el fichero de implementación “mypackzip_functions.c”.

- **inserta_fichero(char * f_mypackzip, char * f_dat, int index)**
  - Parámetros:
    - f_mypackzip: Nombre del fichero mypackzip a crear o en donde insertar los datos.
    - f_dat: Nombre del fichero de datos a insertar dentro de f_mypackzip en el índice index.
    - index: Índice dónde guardar f_dat dentro de f_mypackzip. Rango de valores desde -1 hasta NUM_HEADERS-1, estos inclusive.
  - Si f_mypackzip no existe lo crea utilizando la función **crea_fichero(char * f_mypackzip)** que inicializa todos los headers a vacío y crea el fichero utilizando como nombre y ruta el parámetro “f_mypackzip”.
  - Después de haber sido creado o abierto “f_mypackzip”, se insertan los datos de “f_dat” dentro del índice especificado por parámetro. Si este es -1 se busca la primera posición vacía (si hay una disponible, si no se devuelve error E_POS3) y se utiliza ese índice.
  - A continuación se insertan los datos del fichero “f_dat” previamente abierto en su header correspondiente por índice; estos datos son rellenados dependiendo del fichero f_dat o calculados mediante las funciones **sizeOfFile(char * f_dat)** que obtiene el tamaño de fichero desde su inode utilizando la estructura stat y st_size, **amountOfBlocks(unsigned long blockSize, unsigned long compSize))** que calcula cuantos bloques de tamaño blockSize se necesitan para albergar compSize cantidad de bytes.
  - Una vez insertados los datos del fichero se dispone a leer los datos internos que contiene el fichero “f_dat” mediante un bucle que lee con “read” desde “f_dat” y se escriben en “f_mypackzip” con “write” empezando desde el final del fichero previamente dicho.
  - Una vez terminado de leer y escribir los datos de “f_dat” se rellena el espacio del bloque que aún queda por rellenar con ceros utilizando bzero.

- **extrae_fichero(char * f_mypackzip, int index)**
    - Parámetros:
        - f_mypackzip: Nombre del fichero f_mypackzip de donde extraer el fichero correspondinte del índice index.
        - index: índice correspondiente a la posición del fichero que se quiere extraer de f_mypackzip.
    - Extrae el fichero que se referencia en el índice i con i 0 >= i < NUM_HEADERS, de “f_mypackzip”.
    - Para ello se abre f_mypackzip y se extrae su información del header, se leen sus datos utilizando un bucle desde la posición indicada en el header y se escriben en el fichero previamente creado con “open” y O_CREAT.

ITERACIÓN - P06.3

En esta iteración se ha añadido la funcionalidad de insertar y extraer ficheros de tipo directorio.

(*Ha sido necesario introducir un campo más en el s_header “FileType” que describe el tipo de fichero.)

Para esta iteración se han implementado dos funciones adicionales para saber si f_dat es un fichero regular o es un directorio. Estas dos, obtienen la información gracias a st_mode en la estructura estat:

- **int es_fichero_regular(char * fileName)**
    - Devuelve 1 (TRUE) si el fichero es regular 0 (FALSE) en caso contrario.
- **int es_directorio(char * fileName)**
    - Devuelve 1 (TRUE) si el fichero es directorio 0 (FALSE) en caso contrario.
- **inserta_fichero**
    - Esta función se ha dividido en dos subfunciones para mayor claridad según el tipo de fichero que se quiere insertar que referencia el parametro f_dat.
        - **inserta_fichero_regular(inserta_fichero_regular(int zipFD, int datFD, char * f_dat, struct s_mypack_headers *headers, int index)**
            - Implementa la funcionalidad anterior de “inserta_fichero” únicamente para ficheros regulares.

    - **inserta_directorio**
        - Inserta un directorio y todos sus ficheros regulares contenidos si realmente hay índices libres suficientes.
        - En esta función, en vez de utilizar la llamada al sistema “open” utilizamos la función de biblioteca más específica “opendir” y un bucle “readdir” para leer uno a uno cada entrada dentro del directorio “f_dat” que devuelve una estructura dirent con la información de cada fichero asociado a su directorio padre.
        - Para cada fichero dentro del directorio se introduce dentro de una estructura s_headers diferente y se comprueba que todos los ficheros puedan ser insertados en f_mypackzip si hay suficientes índices libres. Si ese es el caso, se llama a la función “inserta_fichero_regular” para poderlos insertar individualmente definitivamente en “f_mypackzip”.
        - Finalmente se introduce el mismo directorio en el índice especificado en “f_mypackzip”.
- **extrae_fichero**
    - Se ha incluido una condición que depende del tipo de fichero en el header en el índice indicado, se extrae un fichero regular utilizando la implementación anterior de “extrae_fichero” o se extrae un fichero directorio utilizando la función “mkdir”. Notese, que si el fichero es directorio solo se exporta el propio directorio y no los ficheros albergados en él.

ITERACIÓN - P06.3 RECURSIVO

Se ha implementado el insertar y extraer recursivo de mypackzip, esto es, si el usuario inserta un directorio que contiene subdirectorios y todos los ficheros contenidos caben dentro de mypackzip se insertan desde el índice introducido por el usuario. De misma manera, si un usuario elige extraer cualquier directorio o subdirectorio de mypackzip se extraen los ficheros dentro de el mismo.

- **inserta_fichero**
    - Para insertar el fichero se ha creado un campo nuevo vectorial en el “s_header.h” “childIndex[]” que contiene los índices de sus ficheros hijo.
    - Para poder saber si hay suficiente espacio para almacenar todos los ficheros dentro de un directorio y guardar los índices de sus hijos se ha creado una función nueva llamada “**obtener_ficheros_recursivo(char * dirPath, struct s_mypack_headers * tempHeaders, int * tempI, int * nZip)”.**

    - Una vez obtenidos todos los subdirectorios, subficheros y haber comprobado que hay suficiente espacio para insertarlos se escriben en “f_mypackzip” utilizando las funciones implementadas anteriormente “inserta_fichero_regular” y “inserta_directorio”.
- **extrae_fichero**
    - Llama a la función “**extrae_fichero_recursivo**”, que hace de función “worker” recursiva.
- **extrae_fichero_recursivo**
    - Extrae un directorio recursivamente utilizando el vector “childIndex” que dentro de un loop, por cada índice se llama recursivamente a la misma función para poder finalmente extraer todos los ficheros.

ITERACIÓN - P07.3

Esta iteración hace posible que se puedan insertar y extraer iterativamente como recursivamente enlaces simbólicos.

- **inserta_fichero**
    - Para saber si un fichero es enlace simbólico se ha implementado la función “**es_enlace_simbólico**” que llama a lstat.
    - Una vez sabido que “f_dat” es enlace simbólico se inserta mediante la función “**inserta_enlace_simbolico”** que lee el nombre del fichero con lstat y que lee mediante “readlink” la ruta del fichero original e inserta en “f_mypackzip” el enlace introduciendo la información correspondiente en el “s_header” y la ruta al fichero original en el campo datos.
- **extrae_fichero**
    - Se extrae el fichero recursivamente utilizando la función “symlink” y leyendo los datos desde “f_mypackzip”.

ITERACIÓN - P08.2

En esta iteración se ha implementado la función “lsmypackzip” y su ejecutable correspondiente con el mismo nombre para que muestre el contenido de mypackzip por la terminal en formato tipo ls con los índices, permisos, tamaños originales y comprimidos y el nombre del fichero con sus respectivos colores para poder controlar las propiedades del terminal.

ITERACIÓN - P09.3

En la penúltima iteración se ha introducido un campo nuevo en el “s_header” llamado “permisos” que contiene el st_mode con los permisos de cada fichero para, con cada inserción y extracción se lean los permisos con la función “obtener_permisos” y “lstat” se guarden en “f_mypackzip” en los s_headers y a la hora de extraer se cambien los permisos del fichero extraído utilizando “chmod()” si tienes los permisos pertinentes.

ÚLTIMA ITERACIÓN

En la última iteración se ha usado la utilidad “zPipe” para poder comprimir y descomprimir los ficheros regulares a la hora de insertarlos y extraerlos respectivamente.

A medida que se van leyendo los datos desde “f_dat” se van comprimiendo y escribiendo en “f_dat” y a la hora de extraer de manera inversa.