Task 4: FS image extractor

Below is described a simple TAR-like FS image format, especially useful for highly-constrained embedded devices. Each FS image is a file, containing other files and directories inside without any compression.

The task is to create FS image extractor, that will unpack an `.img` file to some directory.

Example:

Here we extract 'image1.img' FS image to 'somedir' directory

```
[usr@pc ~]$ ./extract image1.img somedir

[usr@pc ~]$ ls -hal somedir
total 84K
drwxr-xr-x 2 usr users 4.0K may 14 15:02 .
drwxr-xr-x 6 usr users 4.0K may 14 16:00 ..
-rw-r--r-- 1 usr users  71K may 14 14:58 picture-registrar.jpg
-rw-r--r-- 1 usr users  209 may 14 14:58 zen.txt
```

Note

This task may look complicated. But really, it is not. If you have ever written programs reading and writing files – it is almost the same level of complexity. The hardest part is to understand how the FS image looks inside

Image file specification

Each FS image file consists of the following parts, in order from start to the end of the file:

- **HEADER**

  A header is

Size (bytes)	Name	Description
2	vermagic	Magic number. For forward compatibility. Not used during unpacking (but still present in `.img` file)
2	entnum	Number of files in FS image (and entries in FENTRY TABLE)

- **FENTRY TABLE**

  Contains entries for each file in the FS image, in sequential order. Each FENTRY has information on the file size and offset (from the beginning of the FS image) indicating where that file starts.

Size (bytes)	Name	Description
3	fsize	Size (in bytes) of the file contained in the image
4	foffset	Offset in bytes (from the beginning of the image) indicating where that file starts

  Thus, the first byte of the file is located at the address (`img_base + foffset`) and the last byte – at the address (`img_base + foffset + fsize - 1`), where `img_base` – is a base address of the FS image itself (`img_base = 0` in our case, presuming we do not need to place the whole FS image in RAM and are simply unpacking it)

  If there are numerous files in the image, the FENTRY TABLE will contain numerous FENTRY'es for every file

> ### *Note*
>
> If you read carefully, you may have probably noticed that our FS image format **does not** support empty files

- **FILE NODES**

  Contains entries for each file in the FS image, in the same order as entries in the FENTRY TABLE. Each node consists of the file itself plus the file name.

Name	Description
fdata	Size (in bytes) of the file contained in the image
fnpath	Null-teminated string naming the full file path inside the image (without / in the beginning)

  Therefore, if file `pic.jpg` is located at the root of the image, it's `fnpath` will be simply `pic.jpg\0` (or `"pic.jpg"` in C string literal notation).

  If file `other.txt` is located inside directory `somedir` inside the image, it's `fnpath` will be `somedir/other.txt\0` (or `"somedir/pic.jpg"` in C string literal notation).

  As you may have noticed, the `fnpath` begins at address `(img_base + foffset + fsize)` and ends at address (`where the next file starts - 1 - 1`).

> ### *Note*
>
> We are using the UNIX-way backslashes / for paths, not the slashes \.
>
> Every character of the `fnpath` is an ASCII byte for the means of simplicity.
>
> Once again, the path may contain directories, so your program must create them when needed.

byte order

The byte order is little-endian

Implementation advices

Make sure you use packed structures and fixed-width types where needed. Otherwise, your code may work differently on our machines than it works on yours.

We will test your code with erroneous FS images too. So make sure your program at least returns non-zero exit code in case of error detected. It would be nice if it prints an informative message to `stderr` as well.

In the current directory you may find FS images and their unpacked contents for you to be able to test your solution.

Should you have any questions regarding *this* task, finding the answers is left up to you. This task simulates a real-word scenario where a developer takes a specification as is.

Good luck!

P.S.

We left some Easter Eggs "here and there". We hope you will have lots of joy discovering them