simplefs - a simple file system for Linux

The file system "simplefs" is helpful to understand Linux VFS and file system basics.
The Linux VFS supports multiple file systems. The kernel does most of the work while the file system specific tasks are delegated to the individual file systems through the handlers. Instead of calling the functions directly the kernel uses various Operation Tables, which are a collection of handlers for each operation (these are actually structures of function pointers for each handlers/callbacks).

The super block operations are set at the time of mounting. The operation tables for inodes and files are set when the inode is opened. The first step before opening an inode is lookup. The inode of a file is looked up by calling the lookup handler of the parent inode.

Current features

- Directories: create, remove, list, rename;
- Regular files: create, remove, read/write (through page cache), rename;
- Hard/Symbolic links (also symlink or soft link): create, remove, rename;
- No extended attribute support

Prerequisite

Install linux kernel header in advance.

```
$ sudo apt install linux-headers-$(uname -r)
```

Build and Run

You can build the kernel module and tool with `make`.
Generate test image via `make test.img`, which creates a zeroed file of 50 MiB.

You can then mount this image on a system with the simplefs kernel module installed.
Let's test kernel module:

```
$ sudo insmod simplefs.ko
```

Corresponding kernel message:

```
simplefs: module loaded
```

Generate test image by creating a zeroed file of 50 MiB. We can then mount
this image on a system with the simplefs kernel module installed.

```
$ mkdir -p test
$ dd if=/dev/zero of=test.img bs=1M count=50
$ ./mkfs.simplefs test.img
$ sudo mount -o loop -t simplefs test.img test
```

You shall get the following kernel messages:

```
simplefs: '/dev/loop?' mount success
```

Here `/dev/loop?` might be `loop1`, `loop2`, `loop3`, etc.

Perform regular file system operations: (as root)

```
$ echo "Hello World" > test/hello
$ cat test/hello
$ ls -lR
```

Remove kernel mount point and module:

```
$ sudo umount test
$ sudo rmmod simplefs
```

Design

At present, simplefs only provides straightforward features.

Partition layout

```
+------------+-------------+-------------------+-------------------+-------------+
| superblock | inode store | inode free bitmap | block free bitmap | data blocks |
+------------+-------------+-------------------+-------------------+-------------+
```

Each block is 4 KiB large.

Superblock

The superblock is the first block of the partition (block 0). It contains the partition's metadata, such as the number of blocks, number of inodes, number of free inodes/blocks, ...

Inode store

Contains all the inodes of the partition. The maximum number of inodes is equal to the number of blocks of the partition. Each inode contains 72 B of data: standard data such as file size and number of used blocks, as well as a simplefs-specific field `ei_block`. This block contains:

- for a directory: the list of files in this directory. A directory can contain at most 40920 files, and filenames are limited to 255 characters to fit in a single block.

```
  inode
  +-----------------------+
  | i_mode = IFDIR | 0755 |            block 123
  | ei_block = 123    ----|-------->  +----------------+
  | i_size = 4 KiB        |         0 | ee_block  = 0  |
  | i_blocks = 1          |           | ee_len    = 8  |      block 84
  +-----------------------+           | ee_start  = 84 |---> +-----------+
                                      |----------------|    0 | 24 (foo)  |
                                    1 | ee_block  = 8  |      |-----------|
                                      | ee_len    = 8  |    1 | 45 (bar)  |
                                      | ee_start  = 16 |      |-----------|
                                      |----------------|      | ...       |
                                      | ...            |      |-----------|
                                      |----------------|   14 | 0         |
                                  341 | ee_block  = 0  |      +-----------+
                                      | ee_len    = 0  |
                                      | ee_start  = 0  |
                                      +----------------+

```

- for a file: the list of extents containing the actual data of this file. Since block IDs are stored as `sizeof(struct simplefs_extent)` bytes values, at most 341 links fit in a single block, limiting the size of a file to around 10.65 MiB (10912 KiB).

```
inode
+-----------------------+
| i_mode = IFDIR | 0644 |          block 93
| ei_block = 93     ----|------>  +----------------+
| i_size = 10 KiB       |       0 | ee_block  = 0  |
| i_blocks = 25         |         | ee_len    = 8  |      extent 94
+-----------------------+         | ee_start  = 94 |---> +--------+
                                  |----------------|     |        |
                                1 | ee_block  = 8  |     +--------+
                                  | ee_len    = 8  |      extent 99
                                  | ee_start  = 99 |---> +--------+
                                  |----------------|     |        |
                                2 | ee_block  = 16 |     +--------+
                                  | ee_len    = 8  |      extent 66
                                  | ee_start  = 66 |---> +--------+
                                  |----------------|     |        |
                                  | ...            |     +--------+
                                  |----------------|
                              341 | ee_block  = 0  |
                                  | ee_len    = 0  |
                                  | ee_start  = 0  |
                                  +----------------+
```

Extent support

The extent covers consecutive blocks, we allocate consecutive disk blocks for it at a single time. It is described by `struct simplefs_extent` which contains three members:

- `ee_block`: first logical block extent covers.

- `ee_len`: number of blocks covered by extent.
- `ee_start`: first physical block extent covers.

```
struct simplefs_extent
  +----------------+
  | ee_block =  0  |
  | ee_len   =  200|              extent
  | ee_start =  12 |-----------> +---------+
  +----------------+    block 12 |         |
                                 +---------+
                              13 |         |
                                 +---------+
                                 | ...     |
                                 +---------+
                             211 |         |
                                 +---------+

```

TODO

- Bugs
  - Fail to show `.` and `..` with `ls -a` command
- journalling support

License

`simplefs` is released under the BSD 2 clause license. Use of this source code is governed by a BSD-style license that can be found in the LICENSE file.