CSCI4730/6730 – Operating Systems

Project #3

Due date: 11:59pm, 11/23/2015

Description

In this project, you will implement a simple UNIX-like file system simulator in order to understand the hierarchical directory and inode structures. You will be able to (1) browse the disk information, file and directory list, (2) create and delete files and directories, (3) read and write files. The functionality of the file system is similar to UNIX file systems, but it does not include per-process open file table, permission and user management.

Figure 1 shows the file system structure in our simulator. A size of block is 512 byte and the disk has 4096 blocks. Superblock, inode map, block map and inode blocks are loaded into the memory at file system mount time.

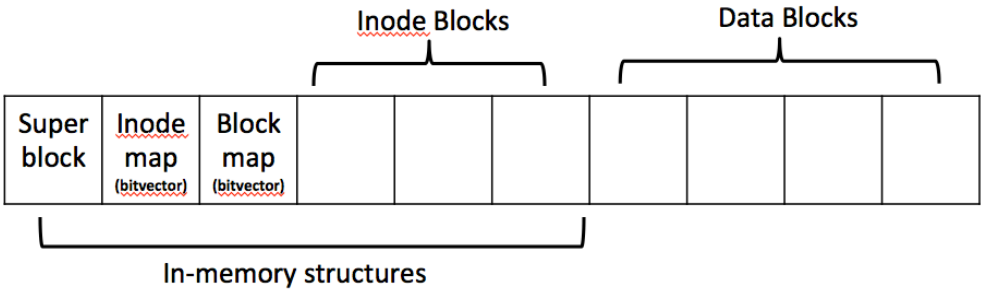

Fig 1. File System Structure

Part 1: Small size files - 50%

In this part, the file system will only use direct blocks in the inode to support small size files (up to 5120 bytes). Each file in a file system has an identification number, called inode number, which is unique in a file system. Inode structure (Inode) is defined in "fs.h" file.

You will implement the following functionalities to access small files:

Command	Arguments	Description
df		Show the file system information: number of free blocks / number of free inodes
create	<name> <size>	Create a file with <filename> as a name in the current directory, and fill it with <size> amount of random

		string.
stat	<name>	Show the status of the file or directory with name <name>. It displays the inode information; whether it is a file or directory; size of the file/directory; number of blocks allocated; other information stored about this file/directory.
cat	<name>	Print out the content of the file.
read	<name> <offset> <size>	Read <size> bytes from the file <offset>.
write	<name> <offset> <size> <string>	Write <size> bytes of <string> into <name> file from the <offset>
rm	<name>	Delete <name> file

Part 2: Directory – 50%

In this part, you will implement hierarchical, tree-structured directory. Each directory has an inode number and an inode block. In our simulator, each directory has up to 25 entries (files and sub-directories).
The first entry is always a current directory (".") and the second entry is a parent directory(".."). You will need to implement the following functionalities:

Command	**Arguments**	**Description**
ls		Show the content of the current directory
mkdir	<name>	Create a sub-directory <name> under the current directory.
rmdir	<name>	Remove the sub-directory <name>.
chdir	<name>	Change the current directory to <name>

Part 3: Large size files – Extra Credit 50%

In this part, you will need to extend the file system to support large size files, up to 70,656 bytes. You will need to implement file access functionalities to support a single indirect block.

Submission

Submit a tarball file using the following command
%tar czvf p2.tar.gz README Makefile *.c *.h

1. README file with:
    a. Your name

    b. List what you have done and how to test them. So that you can be sure to receive credit for the parts you've done.
    c. Explain your design.
2. All source files needed to compile, run and test your code
    a. Makefile
    b. All source files
    c. Do not submit object or executable files
3. Your code should be compiled in cf0-cf11 machine (cf0.cs.uga.edu – cf11.cs.uga.edu)
4. Submit a tarball through ELC.