CS342301: Operating System

Machine Problem 4 - File System

Deadline: 2016/1/15 23:59

1. Problem Description

NachOS native file system only supports up to 4KB file size and only has root directory. In this assignment, you need to study NachOS FS and find out the reason of limitations. Also, you are required to enhance the file system to let NachOS support larger file size and subdirectory structures.

2. Assignment

Part I. Understanding NachOS file system

Trace the file system call and answer the following questions:

(1) Explain how does the NachOS FS manage and find free block space? Where is this information stored on the raw disk (which sector)?

(2) What is the maximum disk size can be handled by the current implementation? Explain why.

(3) Explain how does the NachOS FS manage the directory data structure? Where is this information stored on the raw disk (which sector)?

(4) Explain what information is stored in an inode, and use a figure to illustrate the disk allocation scheme of current implementation.

(5) Why a file is limited to 4KB in the current implementation?

Part II. Modify the file system code to support file I/O system call and larger file size

(1) Combine your MP1 file system call interface with NachOS FS

For your implementation simplicity, you may assumed that all of the userprograms for testing does not contain any messy operations. (E.g. try to create a file that already exist, read/write exceeds the file size, etc.)

There are five system calls to be implemented:

```
int Create(char *name, int size);
```

Create a file with the `name` and with `size` bytes in the root directory. The character in name only contains [A-Za-z0-9.] and with length not greater than 9. Here, this operation will always success and return 1.

```
OpenFileId Open(char *name);
```

Open the file with `name` and return its `OpenFileId`. Only at most one file will be opened at same time. Here, any `OpenFileId` larger than 0 are considered as a successful open.

```
int Read(char *buf, int size, OpenFileId id);
int Write(char *buf, int size, OpenFileId id);
```

Read/Write `size` characters from/to the file to/from `buf`. Return number of characters actually read/written from/to the file. Here, id will always be valid and no messy operations will be given.

`int Close(OpenFileId id);`

Close the file by id. Here, this operation will always success and return 1.

(2) Enhance the FS to let it support up to 32KB file size

You can use any approach including modify the allocation scheme or extend the data block pointer structure, etc.

Important: You **ARE NOT** allowed to change the sector size!!!

For your implementation simplicity, you may assumed that all of the operations will not be messy. (E.g. copy a file larger than 32KB, try to print a non-existing file, etc.)

We will use these commands to check your correctness:

`>nachos -f`

Format the disk on NachOS.

`>nachos -cp <file_to_be_copied> <destination_on_NachOS_FS>`

Copy a file from Linux FS to NachOS FS.

`>nachos -p <file_to_be_dumped>`

Print the content of a file on NachOS disk.

Part III. Modify the file system code to support subdirectory

For your implement simplicity, you may assume that all the operation will not be messy. (E.g. try to remove a non-existing file, try to copy file into a non-existing directory, create a directory in a non-existing directory, etc.)

(1) Implement the subdirectory structure

Use '/' as path name separator

Path has maximum length of 255.

Length of directory and file name does not exceed 9.

All path are absolute (e.g. /testing/num100, /1000, etc.)

Support recursively list the file/directory in a directory

(2) Support up to 64 files/subdirectories per directory

We will use these commands to check your correctness:

`>nachos -f`

Format the disk on NachOS.

`>nachos -mkdir <directory_to_be_created>`

Create a directory on NachOS disk.

`>nachos -cp <file_to_be_copied> <absolute_path_on_NachOS_FS>`

Copy a file from Linux FS to NachOS FS.

`>nachos -r <file_to_be_deleted>`

Delete a file (not a directory) from NachOS FS.

`>nachos -l <list_direcotry>`

List the file/directory in a directory.

`>nachos -p <file_to_be_dumped>`

Print the content of a file on NachOS disk.

`>nachos -lr <directory_to_be_listed>`

Recursively list the file/directory in a directory. The Directory will always exist.

Other Work. Memory Leakage Check

In this assignment, you are required to use valgrind to ensure that your NachOS does not memory leak at any time.

You can use the following command to check whether your NachOS leaks memory:

```
>valgrind ../build.linux/nachos <NachOS_command>
```

TA will use several test cases to detect your memory leakage.

Only “definitely lost”, “indirectly lost”, “possibly lost” blocks will be counted.

N leakage block(s) will receive $\lceil log_2(N+1) \rceil$ point(s) penalty.

3. Bonus Assignment

Design your own test cases to show your implementation is correct. There is no output format limitations, any way to show your result is acceptable.

You do not have to handle messy operations. (E.g. copy a file larger than 64MB, delete/list a non-existing directory)

Bonus I. Enhance the NachOS to support even larger file size

(1) Extend the disk from 128KB to 64MB

(2) Support up to 64MB single file

Bonus II. Multi-level header size

(1) Show that smaller file can have smaller header size.

(2) Implement at least 3 different size of headers for different size of files

Bonus III. Recursive Operations on Directories

(1) Support recursive remove of a directory

We will use these commands to check your correctness:

```
>nachos -rr <file/directory_to_be_removed>
```

Remove the file or recursively remove the directory

The directory to be removed will always exist and will not be root directory.

4. Grading

Demo - 60%

- a) Part II - 30%
- b) Part III - 30%
- c) Memory Leakage Check - 0%

Report - 40%

- a) Part I - 20%
- b) Report of Implementation - 20%

Bonus - 15%

- a) Bonus I - 5%
- b) Bonus II - 5%
- c) Bonus III - 5%

5. Remainder

1. iLMS
   - a) You do not have to upload your NachOS to iLMS, but we will use your latest modification time as your submission time.
   - b) Please upload your Part I and implementation report in PDF format to iLMS.
2. Demo policy
   - a) All demo will be performed on our server.
   - b) You are responsible to make sure your NachOS works on our server.
3. Please refer to syllabus for late submission penalty.
4. 0 will given to cheaters. Do not copy & paste!
5. Feel free to ask questions on iLMS!