Hints for Programming Assignment 3

Bijan Soleymani

April 8, 2007

1 Objective

The goal of this assignment is to add networking to your file system and disk emulation code from PA1 and PA2. Specifically it will be done in a client server fashion, using TCP/IP for transmission and implementing a simple protocol for communication.

2 Client vs. Server

There is a sharp distinction between client and server. The client does all the input and output. That is to say it calls the functions to create files, store and retrieve data, etc. The server on the other hand is responsible for all the storage and retrieval. The server handles the FAT, directory table, and so on, and writes all of these plus the data to a file on the disk.

From the perspective of PA2, the client is now doing what the test code in main used to do, and the server is doing what the filesystem code used to do. Except in this case instead of having the test code directly call functions in the filesystem code, the client must send the request and receive the responses using TCP/IP.

3 TCP/IP

TCP/IP assures reliable in order delivery of data. We will be using it to move data as well as request from the client to the server and back again.

The first step in this direction is to write a simple client/server pair that can exchange an arbitrary block of data. A good example is an echo client/server. Whatever is typed at the client is sent to the server and is printed to the screen.

You can find the code for this at:

http://www.paulgriffiths.net/program/c/

Once that's working it's time to figure out how to send and receive actual filesystem requests.

4 Protocol

We need to establish rules so that the client and server know how to communicate with each other. One possibility is to use a fixed size header at the beginning of every transmission. All this header has to say is, what is being requested and how much extra data is being transmitted. Each of these can be represented as an int. And thus the header only needs to be 8 bytes long. You can transmit and receive it in the following manner:

```
int request;
int size;
char buffer[8];
//Put request at beginning of buffer
memcpy(buffer, request, sizeof(int));
//Put in second half of buffer
memcpy(buffer+sizeof(int), size, sizeof(int));
//Then send buffer
//On the receiving end do the reverse
memcpy(request, buffer, sizeof(int));
memcpy(size, buffer+sizeof(int), sizeof(int));
```

5 Requirements

We need to have a single client communicating with at least 2 servers. Each server should have a seperate filesystem (that means we can create a file with the same name on each server and put different data in each).

The only new data structures are on the client side. There should be a table indicating which filesystem is on which server. So it would contain the 3 fields: filesystem name, ip address, port number. Another table is necessary to indicate which file is on which server. It would have the 3 fields: file descriptor number on client side, server containing the file, corresponding file descriptor on that server.

6 Test Case

Here is an idea of what PA3 will be expected to do.

run a server on port 8000.

run another server on port 8001.

```
int i;
char buffer1[20];
char buffer2[20];
int array[1024];
int file1, file2, file3;
mount("s1", "localhost", 8000);
mount("s2", "localhost", 8001);
nfs_ls("s1");
nfs_ls("s2");
file1 = nfs_fopen("s1", "a.txt");
nfs_fwrite(file1, "hello", 5);
file2 = nfs_fopen("s1", "b.txt");
nfs_fwrite(file2, "goodbye", 7);
nfs_fclose(file2);
nfs_ls("s1");
file3 = nfs_fopen("s2", "c.txt");
for(i=0;i<1024;i++){
  array[i]=i*i;
}
nfs_fwrite(file3, array, 1024*sizeof(int));
file2 = nfs_fopen("s2", "a.txt");
nfs_fwrite(file2, "test", 4);
nfs_fwrite(file1, " world", 6);
nfs_fclose(file1);
nfs_fclose(file2);
nfs_fclose(file3);
nfs_ls("s2");
file1 = nfs_fopen("s1", "a.txt");
nfs_fread(file1, buffer1, 11);
file2 = nfs_fopen("s2", "a.txt");
nfs_fread(file2, buffer2, 4);
nfs_fclose(file1);
nfs_fclose(file2);
nfs_remove("s1", "a.txt");
nfs_remove("s2", "b.txt");
nfs_ls("s1");
nfs_ls("s2");
file1 = nfs_fopen("s2", "c.txt");
nfs_fread(file1, array, 1024*sizeof(int));
nfs_fclose(file1);
```

note: make sure to print out the output of the nfs_reads to make sure everything is worknig properly.