EEL-4736/EEL-5737 Principles of Computer Systems Design

Homework #2 Assigned: 9/7/2015 To be done individually

Part-1 due 9/18/2015   Part-2 due 9/28/2015

This assignment is divided into two parts, Part-1 will test your understanding of the concepts covered in class by working out problems and Part-2 will involve designing and implementing a multi-level/hierarchical file system and extending it to support remote storage.

Part-1 (Due- 9/18)

Textbook exercise 2.2

Textbook exercise 2.5

Textbook exercise 4.3

Textbook exercise 4.7

Textbook exercise 4.8

Submission

submit a pdf file named homework2part1.pdf with solution to the above problems.

Part-2 (Due-9/28)

This part is to be implemented in two steps, Step-1 will require you to extend the flat in-memory file system implemented in memory.py to a hierarchical in-memory FS, much like you have in contemporary operating systems. Step-2 will then build upon the implementation in Step-1 to further enhance the functionality by moving the storage to a remote machine.

Step-1 - hierarchical name space

To proceed with this task, you must first familiarize yourself with memory.py to get a good understanding of how it works. If you look at the constructor for the Memory class (line 15), you will notice that memory.py uses two dictionaries (sometimes also called hash tables), one called “files” (line 16) which stores the metadata associated with a file system object (mode, creation time, modification time, access time, and number of links), and the other called “data” (line 17) which stores all of the data for a file. Recall that in Python, a dictionary provides a flexible primitive to create contexts where you can store and lookup name-value bindings. It is extremely important to understand how these two dictionaries are used for storage. Also note how the methods implementing FS functionality interact with these dictionaries to store, retrieve and modify content. Your goal is to design your system using a data structure that not only stores the contents but lets you support a multi-level namespace.

The solution to this problem is not complex, but requires that you consider carefully how to store and lookup objects that include contexts. Think about the answers to the following questions before you design your solution:

1. What is a file system, and how is it structured in terms of files and directories ? This assignment does not requires you to worry about low level details (e.g. inodes) but you need to consider the logical organization of files and directories.
2. How is a file structurally different from a directory?
3. How does a File system weaves together a skeleton holding all the files and directories, How does it connects them?
4. Which data structure I can use to build this skeleton?
5. How can I represent a File or Directory as a data structure?
6. How can I organize File/Directory data structures using a FS data structure.?
7. How will I travel the FS data structure to get to a desired File/Directory data structure?
8. How can I do the above (7) while parsing a hierarchical namespace like /dir1/dir2/hello.txt ?

If you have convincing answers to above questions you can start working on step-1. Once you have it ready, test it thoroughly to ensure that you can carry out regular FS operations. Your design will be graded on the ability to support complex file system operations, so it is to your benefit to test thoroughly.

Step-2 - Client/server

In this step you will design and implement a client-server model as an extension to the hierarchical name space of Step-1. The client process will communicate with the server process through the use of the XMLRPC protocol - an easy-to-use RPC framework available in Python. A goal in this assignment is to solidify your understanding of naming, the memory and interpreter abstractions, and illustrate differences in the design of single name-space vs. message-passing client/server systems.

Understanding the Python XMLRPC module

1. Read the following pages from the Python library pages:

a) http://docs.python.org/library/simplexmlrpcserver.html

b) http://docs.python.org/library/xmlrpclib.html

c) http://docs.python.org/library/simplexmlrpcserver.html#simplexmlrpcserver-example

d) https://docs.python.org/2/library/pickle.html

2. Download the server module and read the code thoroughly

a) http://www.acis.ufl.edu/~renato/pocsd/simpleht.py

b) Run the unit test locally (http://docs.python.org/library/unittest.html)

3. Write a simple XMLRPC client program called test-client.py that connects to simpleht.py and does the following operations (for the TTL value here and in the rest of this assignment, you can use 3000 seconds):

a) Stores and retrieves an integer

b) Stores and retrieves a string

c) Stores and retrieves a list

d) Stores and retrieves a dictionary

e) Use the write_file and read_file functions of the simpleht to save/restore data to/from the filesystem.

4. After you have established a hold of concepts in 3, modify the FS data structure to move File/Directory data structures to the simpleht server without breaking the FS skeleton that connects the File/directory data structures together, making it possible to traverse the multi-level hierarchy. You should be careful in designing the naming scheme for any content that is stored in simpleht, to be able to uniquely identify it.

When you are done with the implementation once again test it by trying out regular FS operations.

Submission

Use the Sakai E-learning system to submit the three python code files, as follows. (NOTE: to facilitate grading, please make sure you use exactly the file names as described)

1. hierarchicalFS.py -- This implements the local in-memory multi level FS.
2. test-client.py -- Tests methods exposed by simpleht.py
3. remoteHierarchicalFS.py -- This extends hierarchicalFS.py to support client-server model.

You should also submit a pdf file:

1. homework2design.pdf describing the design of your implementation, and the tests you conducted to verify the functionality.

It is suggested that you compress the above files into a single folder and name it -- H2PB_<your name>.<compression format>

Make sure your code is well commented and tested, it will be graded on the basis of its design and its functionality. There might be several feasible designs to solve the assignment choose the one that’s most concise and modular allowing you to make minimum changes when you extend the functionality.