Proposal: File and Directory Manipulation Language (FDL)

Cara J. Borenstein
Columbia University
cjb2182@columbia.edu

Daniel Garzon
Columbia University
dg2796@columbia.edu

Daniel L. Newman
Columbia University
dln2111@columbia.edu

Pranav Bhalla
Columbia University
pb2538@columbia.edu

Rupayan Basu
Columbia University
rb3034@columbia.edu

September 24, 2013

1 Motivation

With the *proliferation of storage devices*, and the rise of mobile and cloud computing, users must now manage a *large number of files scattered across several locations*. Thus the problem of accessing and organizing multiple files quickly and easily across storage media is becoming increasingly more important. Currently there is no language that enables users to efficiently and easily organize their files and directories according to various attributes such as last modified date, created by, size etc. and add customized tags to files such as scrap, important, work, and home.

2 Description

File and Directory Manipulation Language (FDL, pronounced fiddle) *solves this problem* by providing simple and intuitive syntax for manipulating *files* and *directories*. By providing the user with new *data types*, and an extensive list of *mathematical and logical operators*, what used to be tedious and time consuming will now be easy and fast.

Users can write programs that organize their file systems by conveniently copying files and directories to different locations, and removing files and directories from specific filepaths, through the use of mathematical operators. Users can loop through subdirectories and files contained within a chosen directory, with a template to browse the file/directory tree stemming from that directory by specifying different levels. One example is the ability to perform a function on all nodes of the tree at a certain level away from the root directory.

Files can be organized in this manner by the attributes spanning from last modified date to size, and additional, customized tags can be added to files for organizational purposes. Customized tags can be serialized and stored on the machine in XML format, to be loaded when users are navigating the file system.

3 Syntax

3.1 Basic Data Types

primitive	**Description**
int	The set of all positive natural numbers: $\mathbb{N}^0 = \{0, 1, 2, 3, \ldots, k\}$
bool	Used to compare two files or directories for equality. Returns 1 for *true* and 0 for *false*.
string	A sequence of *characters* surrounded by quotes.
dir	Object that holds the path to a *collection* of 0 or more *files* in memory. Directories can contain any number of *files* and any number of *sub-directories*.
file	Object that has a *file_type*, *modified_date*, *created_date*, and 0 or more customized *tags*.

3.2 File Attributes

attribute	**Description**
created_date	Field that holds the date when a *file* was created.
modified_date	Field that holds the date of the last time a *file* was modified.
file_type	Field that holds the type of a file. (ex. 'txt', 'jpeg').
tag	Field that holds a *customized association* of a *file* or *directory*.

3.3 Mathematical Operators

operator	**Description**
+	Used to add *files* to *directories* and also to append *strings*.
−	Used to remove 1 or more *files* from a *directory*.
,	Used to specify multiple objects that should be evaluated separately by the previous operator.
=	Assignment operator.
+ =	For a *directory* it is used to add a *file* or *sub-directory* to the *directory*. For integers, it is the *addition and assignment* operator.
− =	For a *directory* it is used to remove a *file* or *sub-directory* to from *directory*. For integers, it is the *substraction and assignment* operator.

3.4 Logical Operators

operator	**Description**
==	Equality operator.
! =	Inequality operator.
>	Used for checking the level of a *sub-directory* (select *files* at a level *greater than* the current *directory*), and for comparing *integers*.
>=	Used for checking the level of a *sub-directory* (select *files* at a level *greater than or equal to* the current *directory*), and for comparing *integers*.
<	Used for checking the level of a *sub-directory* (select *files* at a level *less than* the current *directory*), and for comparing *integers*.
<=	Used for checking the level of a *sub-directory* (select *files* at a level *less than or equal to* the current *directory*), and for comparing *integers*.

3.5 Control Statements

3.5.1 *if-then-else*

```
if <condition> then
  <expression>
else
  <expression>
end
```

3.5.2 *while*

```
while <condition> then
  <expression>
end
```

3.5.3 *for*

```
for <identifier> in <set of files> do
  <expression>
end
```

3.6 Function Definition

```
def <identifier> (<parameter list>)
  <expression>
end
```

4 Example Program

```
main
  directory D1 = "/SAMPLE_PATH";//path to the source directory
  string str = ""; //path to the destination folder

  // we expect file_temp will loop over all files in "D1" including subfolders
  for file_temp in D1
    if( file_temp.Extension == "jpg" || file_temp.Extension == "jpeg" ){
      // we wish to name the folders with date on which images were created
      // the below stmt creates(in case it didnt exist) or points dtemp to //the folder
      Directory dtemp = str + file_temp.Date;
      dtemp = dtemp + file_temp;
    }
  }
end
```

Listing 1: Code for ...