Spring 2014 CS416 Operating System Design

Homework 7 - Report

Team Member: Yihan Qian, Zhenhua Jia, Yimeng Li,

Introduction

We finished all the required functions of this assignment. The parameters of each operation are in the directories of “files”.

Operations can handle both positive and negative legal parameters. When there is an exception caused by illegal input parameters. the system will automatically generate a notice about this error. Exception examples:

1. /div/a/b divider “b” is zero.
2. /factor/n n is negative or n is an integer that is smaller than 2.
3. /fib/n n is not an integer or n is smaller than 1.
4. Etc.

Operating System environment

1. Operating System name: GNU/Linux
2. Node Name: cp.cs.rutgers.edu
3. Operating System Version: #1 SMP Tue Jul 16 23:51:20 UTC 2013
4. Virtual machine or local: iLab Local
5. Hardware platform: x86_64

Implementation

Dynamic array

This array is used to store the results for Factor or Fibonacci functions.

```
/* dynamic array */
typedef struct
{
      uint64_t *array;
      size_t used;    /* count the number of integers */
      size_t size;    /* count the size of this array */
} Array;
```

Array Initialization

The initArray( ) function initializes the dynamic array with initial size.

```
/* initialize the array */
void initArray(Array *a, size_t initialSize);
```

Array Insertion

The insertArray( ) function pushes new data back.

```
/* insert at the end */
int insertArray(Array *a, uint64_t element)
```

Array print

The printArray( ) function dumps all the data in the array.

```
/*print the entire array */
void printArray(Array *a)
```

Array free space

The freeArray( ) function frees the array space.

```
/* free the array sapce */
void freeArray(Array *a)
```

Computation of Fibonacci numbers

The Fibonacci( ) function computes the first n Fibonacci numbers and stores the results in the dynamic array.

```
/* Fibonacci Number */
void Fibonacci (Array *a, double number)
```

Computation of Prime Factors

The primeFactors( ) function computes all the prime factors and stores the results in the dynamic array.

```
/* print all prime factors of a given number n */
void primeFactors(Array *a, double number)
```

Integer Check

The IsInt( ) function checks if the parameter is an integer.

```
/* check if a double is also an integer */
```

```
int IsInt(double d)
```

Operation table

We use the same operation table as the hello.c.

```
static struct fuse_operations math_oper =
{
    .getattr   = math_getattr,
    .readdir   = math_readdir,
    .open      = math_open,
    .read      = math_read,
};
```

Get attribute function

This function returns metadata concerning a file specified by path in a special stat structure.

1. Parse the string

The parseString( ) function gets a copy of the path, parses the copy into tokens and stores them in a global string pointers.

```
/* Parse the path string into tokens */
int parseString(const char *path)
```

2. Compare the path with directory

Loop around to check if there is a path match. If there is a match, trigger the setting of the attributes of the directory/file.

```
if (strcmp(arg[0], path[i]) == 0)
{
    /* we have a match */
    /* check for the presence of other elements in the path
    */
}
```

3. Define a directory

If getattr() is called with a path of "/add", “/sub”, etc., then set the attributes of the directory:

```
stbuf->st_mode = S_IFDIR| 0755;
stbuf->st_nlink = 3;
```

4. Define a file

If getattr() is called with a path of "/add/doc", “/sub/doc”, etc., then set the attributes of a file:

```
stbuf->st_mode = S_IFREG | 0444;
stbuf->st_nlink = 1;
stbuf->st_size = strlen(add_str);
```

Read directory function

```
static int math_readdir(const char *path, void *buf,
fuse_fill_dir_t filler, off_t offset, struct fuse_file_info
*fi);
```

If readdir() is called with a path of "/", then return a list of items:

```
filler(buf, ".", NULL, 0);
filler(buf, "..", NULL, 0);
filler(buf, add_path + 1, NULL, 0);
filler(buf, sub_path + 1, NULL, 0);
etc.
```

If readdir() is called with a path of "/add" (or "/mul", ...), then a list of items:

```
filler(buf, ".", NULL, 0);
filler(buf, "..", NULL, 0);
filler(buf, "doc", NULL, 0);
```

If readdir() is called with a path of "/add/123", then you return an empty directory:

```
filler(buf, ".", NULL, 0);
 filler(buf, "..", NULL, 0);
```

Open function

Check whether it is permitted to open the file. If it is not allowed, return error.

```
static int math_open(const char *path, struct fuse_file_info
*fi);
```

Read Function

Feed the user with data, the data is either from the file or generated by math functions.

```
static int math_read(const char *path, char *buf, size_t size,
off_t offset, struct fuse_file_info *fi);
```

1. Read the document with a path of “/add/doc”.(or “/sub/doc”, etc)

```
len = strlen(add_str);
if (offset < len)
{
   if (offset + size > len)
         size = len - offset;
         memcpy(buf, add_str + offset, size);
 }
else
      size = 0;
```

2. Div function (add, sub, mul, exp functions are similar, but have no exception check) If the divider is not 0, then compute the quotient.

```
/* check if the divisor is 0 */
if (d[1] != 0)
{
   result = d[0] / d[1];
   sprintf(buffer, "%lf\n", result);
}
else
   strcpy(buffer, "divide by zero error\n");
```

3. Compute Fibonacci. If there is no exception, call Fibonacci( ) to compute and store the Fibonacci numbers in the dynamic array. Then, dump all the numbers in buffer.

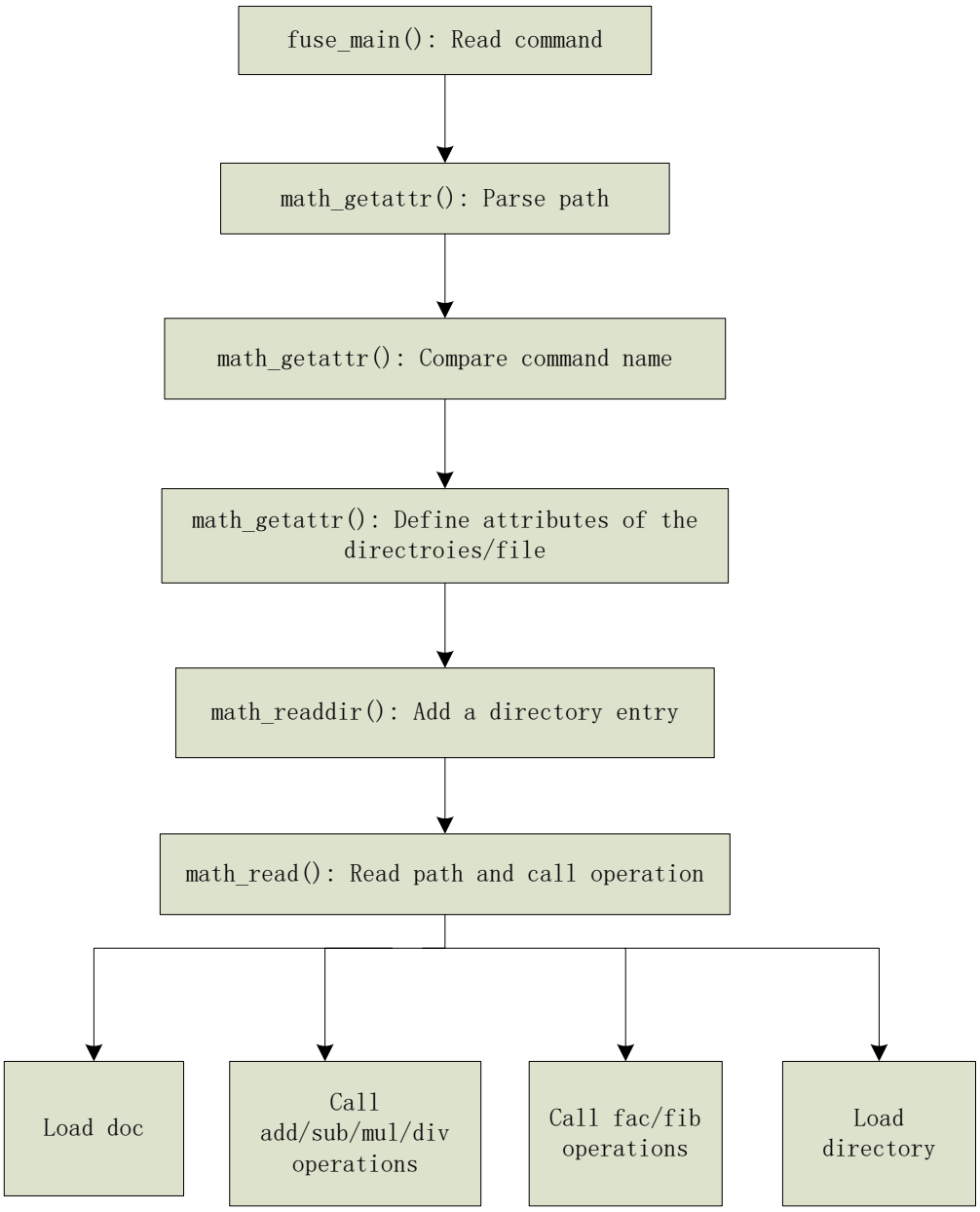
```
/* check if input number is a positve integer */
if (IsInt(d) && d >= 1)
{
   Fibonacci(&f, d);
   printArray(&f);

   /* put fibonacci numbers into one buffer*/
   for (i=0; i < (&f)->used; i++)
      {
         sprintf(tempbuf, "%"PRIu64"\n",
         (&f)->array[i]) ;
         strcat(buffer, tempbuf);
   }
}
else
```

```
   strcpy(buffer, "Error. Only integers that are larger
than 1 can be accepted.\n");
```

4. Compute prime factors. If there is no exception, call primeFactors( ) to compute and store all the prime factors in the dynamic array. Then, dump all the numbers in buffer.

```
/* check if the input number is a integer > 2*/
if (IsInt(d) && d > 2)
{
   primeFactors(&f, d);
   for (i=0; i < (&f)->used; i++)
   {
       sprintf(tempbuf, "%"PRIu64"\n", (&f)->array[i]) ;
      strcat(buffer, tempbuf);
   }
}
else
   strcpy(buffer, "Error. Only positive integer is
expected.\n");
```

Diagram



Test results

Test results are shown below:

1) Top-level directory listing:

```
File  Edit  View  Search  Terminal  Help
bash-4.1$ ls -l math
total 0
drwxr-xr-x 3 root root 0 Dec 31  1969 add
drwxr-xr-x 3 root root 0 Dec 31  1969 div
drwxr-xr-x 3 root root 0 Dec 31  1969 exp
drwxr-xr-x 3 root root 0 Dec 31  1969 fac
drwxr-xr-x 3 root root 0 Dec 31  1969 fib
drwxr-xr-x 3 root root 0 Dec 31  1969 mul
drwxr-xr-x 3 root root 0 Dec 31  1969 sub
bash-4.1$
```

2) Each top-level subdirectory contains a doc file:

```
bash-4.1$ ls -l math/*/*
-r--r--r-- 1 root root 56 Dec 31  1969 math/add/doc
-r--r--r-- 1 root root 65 Dec 31  1969 math/div/doc
-r--r--r-- 1 root root 54 Dec 31  1969 math/exp/doc
-r--r--r-- 1 root root 48 Dec 31  1969 math/fac/doc
-r--r--r-- 1 root root 55 Dec 31  1969 math/fib/doc
-r--r--r-- 1 root root 68 Dec 31  1969 math/mul/doc
-r--r--r-- 1 root root 64 Dec 31  1969 math/sub/doc
bash-4.1$
```

3) Show the contents of several directories:

```
bash-4.1$ ls -l math/add
total 0
-r--r--r-- 1 root root 56 Dec 31  1969 doc
bash-4.1$ ls -l math/div
total 0
-r--r--r-- 1 root root 64 Dec 31  1969 doc
bash-4.1$ ls -l math/exp
total 0
-r--r--r-- 1 root root 54 Dec 31  1969 doc
bash-4.1$ ls -l math/fac
total 0
-r--r--r-- 1 root root 48 Dec 31  1969 doc
bash-4.1$ ls -l math/fib
total 0
-r--r--r-- 1 root root 56 Dec 31  1969 doc
bash-4.1$ ls -l math/mul
total 0
-r--r--r-- 1 root root 65 Dec 31  1969 doc
bash-4.1$ ls -l math/sub
total 0
-r--r--r-- 1 root root 68 Dec 31  1969 doc
bash-4.1$
```

4) Add operation:

Normal calculation:

```
bash-4.1$ cat math/add/1234/5.52
1239.520000
bash-4.1$
```

Deal with other directories:

```
bash-4.1$ cat math/add
cat: math/add: Is a directory
bash-4.1$ cat math/add/13
cat: math/add/13: Is a directory
bash-4.1$ cat math/add/13/23/23
cat: math/add/13/23/23: Not a directory
bash-4.1$
```

Show documents:

```
bash-4.1$ cat math/add/doc
Add two numbers.
The file add/a/b contains the sum a+b.
```

5) Div operation:

Normal calculation:

```
bash-4.1$ cat math/div/138/4
34.500000
```

Handle exception:

```
bash-4.1$ cat math/div/138/0
divide by zero error
```

Deal with other directories:

```
divide by zero error
bash-4.1$ cat math/div
cat: math/div: Is a directory
bash-4.1$ cat math/div/34
cat: math/div/34: Is a directory
bash-4.1$ cat math/div/34/23/43
cat: math/div/34/23/43: Not a directory
```

Show documents:

```
bash-4.1$ cat math/div/doc
Divide two numbers.
The file div/a/b contains the quotient a/b.
```

6) Exp operation

Normal calculation:

```
bash-4.1$ cat math/exp/2/8
256.000000
```

Deal with other directories:

```
bash-4.1$ cat math/exp
cat: math/exp: Is a directory
bash-4.1$ cat math/exp/2
cat: math/exp/2: Is a directory
bash-4.1$ cat math/exp/2/3/4
cat: math/exp/2/3/4: Not a directory
```

Show documents:

```
bash-4.1$ cat math/exp/doc
The file exp/a/b contains a raised to the power of b.
```

7) Fac operation

Normal calculation:

```
bash-4.1$ cat math/fac/223092870
2
3
5
7
11
13
17
19
23
```

Handle exception:

```
bash-4.1$ cat math/fac/138.3
Error. Only positive integer is expected.
bash-4.1$ cat math/fac/-34
Error. Only positive integer is expected.
bash-4.1$
```

Deal with other directories:

```
bash-4.1$ cat math/fac
cat: math/fac: Is a directory
bash-4.1$ cat math/fac/2/3
cat: math/fac/2/3: Not a directory
```

Show documents:

```
bash-4.1$ cat math/fac/doc
The file fac/n contains the prime factors of n.
```

8) Fib operation

Normal calculation:

```
bash-4.1$ cat math/fib/15
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
bash-4.1$
```

Another implementation:

```
bash-4.1$ cat math/fib/50|tail -15
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
2971215073
4807526976
7778742049
12586269025
```

Deal with other directories:

```
bash-4.1$ cat math/fib/15.3
Error. Only integers that are larger than 1 can be accepted.
bash-4.1$ cat math/fib/-13
Error. Only integers that are larger than 1 can be accepted.
bash-4.1$
```

Show documents:

```
bash-4.1$ cat math/fib/doc
The file fib/n contains the first n fibonacci numbers.
bash-4.1$
```

9) Mul operation

Normal calculation:

```
bash-4.1$ cat math/mul/3/5
15.000000
bash-4.1$
```

Deal with other directories:

```
bash-4.1$ cat math/mul
cat: math/mul: Is a directory
bash-4.1$ cat math/mul/5
cat: math/mul/5: Is a directory
bash-4.1$ cat math/mul/25/2/3
cat: math/mul/25/2/3: Not a directory
bash-4.1$
```

Show documents:

```
bash-4.1$ cat math/mul/doc
Multiply two numbers.
The file mul/a/b contains the product a*b.
```

10) Sub operation

Normal calculation:

```
bash-4.1$ cat math/sub/67/3
64.000000
```

Deal with other directories:

```
bash-4.1$ cat math/sub
cat: math/sub: Is a directory
bash-4.1$ cat math/sub/34
cat: math/sub/34: Is a directory
bash-4.1$ cat math/sub/34/5/6
cat: math/sub/34/5/6: Not a directory
bash-4.1$
```

Show documents:

```
bash-4.1$ cat math/sub/doc
Subtract two numbers.
The file sub/a/b contains the difference abash-4.1$
```