Write-Up for Project 1: Transparent Remote File System

Adam Li *zli3@andrew.cmu.edu*

1 Data-marshalling Protocol

The data structures for data-marshalling protocol is defined in `marshall.h`. A `general_wrapper` struct is defined to act as the outter header, which provides general information about the total packet length and the procedure call code to be operated.
The payload is define as a zero-length array because the payload length is dynamic, i.e., can only be determined after having the information about the call parameters and so on.

```
typedef struct general_wrapper {
    int total_len;
    int op_code;
    char payload[0];
} general_wrapper;
```

The payload may be populated with one of many kinds of structs designed for different RPCs. For example, like the `read_write_payload` presented below, provides the parameters such as file descriptor, the number of byte needed to read/write, and the information section `buf` (which is also a zero-length array which needed to be malloced ad-hoc, because the length of the information to write may vary).

```
typedef struct read_write_payload {
    int fildes;
    size_t nbyte;
    char buf[0];
} read_write_payload;
```

There are other data structures such as `close_payload`, `lseek_payload`, etc.

2 Serialization of `dirtreenode`

I designed the algorithm for serialization such that the non-binary tree structure is serialized into a string using DFS with time complexity O(N). First, DFS the whole tree to calculate how much space is needed (for malloc). Second, DFS the tree and pack it into a string. In that the path name have varied length, I also put the path length in front of the actual path name, so that it is easier for deserialization Element structure packed into string: `[size_t path_len, int children_num, char *path]`