Assignment2 Report

20181016 KwonYongmin

1. Implementing 'is_elf' function.

```
bool is_elf(Elf64_Ehdr ehdr)
{
    if(memcmp(ehdr.e_ident, ELFMAG, SELFMAG) == 0){ // compare ehdr.e_ident with ELFMAG defined in elf.h, as much as SELFMAG size
        return true;
    }else{
        return false;
    }

}
```

The first four bytes of every elf file equal to 0x7f, E, L, F. When I watch the source code of elf.h, there was useful variables. In Elf64_Ehdr type, the attribute called 'e_ident' means the first four bytes of elf file. And ELFMAG means the special value, 0x7f, E, L, F. Also, SELFMAG means the size of 'e_ident', so I compare these values with 'memcmp'.

2. 'Section info' struct.

```
typedef struct
{
    Elf64_Addr sh_addr;
    Elf64_Off sh_offset;
    Elf64_Xword sh_size;
}Section_Info;
```

In given source code, especially 'read_elf_header' and 'read_section' function show us how to read the whole elf file practically. But I want to make some variable contain the information only about '.rodata section'. I defined a struct which has three attribute, section address, section offset from starting point of the file, and the size of section.

3. 'find_rodata_info' function.

```
Section_Info* find_rodata_info(int32_t fd, Elf64_Ehdr eh, Elf64_Shdr sh_table[])
{
    uint32_t i;
    char* sh_str;
    Section_Info* rodata_info = (Section_Info *)malloc(sizeof(Section_Info));

    assert(lseek(fd, (off_t)eh.e_shoff, SEEK_SET) == (off_t)eh.e_shoff);

    for(i=0; i<eh.e_shnum; i++) {
        assert(read(fd, (void *)&sh_table[i], eh.e_shentsize) == eh.e_shentsize);
    }

    sh_str = read_section(fd, sh_table[eh.e_shstrndx]);

    for(i=0; i<eh.e_shnum; i++) {
        if(!strncmp((sh_str + sh_table[i].sh_name), ".rodata", 7))
        {
            rodata_info->sh_addr = sh_table[i].sh_addr;
            rodata_info->sh_offset = sh_table[i].sh_offset;
            rodata_info->sh_size = sh_table[i].sh_size;
        }
    }

    return rodata_info;
}
```

With file descriptor and section table, this function fills empty variable of which type is 'Section info' with the information of .rodata section.

4. 'find_and_change_difficult' function.

```
Section_Info* rodata_info = (Section_Info *)malloc(sizeof(Section_Info));

rodata_info = find_rodata_info(fd, ehdr, sh_tbl);
char* data = malloc(rodata_info->sh_size);
lseek(fd, rodata_info->sh_offset, SEEK_SET);
read(fd, data, rodata_info->sh_size);

find_and_change_difficult(fd, data, rodata_info->sh_offset, rodata_info->sh_size);
```

```
void find_and_change_difficult(int32_t fd, char* data, int32_t offset, int32_t size){
    const char* str = "difficult!";
    const char* str2 = "funny!";
    const int str_len = strlen(str);
    const int str2_len = strlen(str2);

    int i = 0;
    while(i < size){
        if(strlen(data + i) != 0){
            char* target = (data + i);
            int target_len = strlen(target);
            char* ptr = strstr(target, str);
            while(ptr != NULL){
                int d_index = ptr - target;
                for(int j = 0; j < str2_len; j++){
                    target[d_index + j] = str2[j];
                }
                int shift_index = d_index + str_len;
                for(int k = 0; k < target_len - shift_index; k++){
                    target[shift_index - (str_len - str2_len) + k] = target[shift_index + k];
                }
                for(int l = 1; l <= (str_len - str2_len); l++){
                    target[target_len - l] = ' ';
                }
                ptr = strstr(target, str);
            }
            if(strlen(data + i) == strlen(target)){
                lseek(fd, offset + i, SEEK_SET);
                write(fd, target, strlen(target));
            }
            i += strlen(data + i);
        } else {
            i++;
        }
    }
}
```

Before using this function in assignment.c, I make a buffer of which size is equal to .rodata section and fill this buffer with contents of .rodata section using read function.

The arguments of this function is the buffer filled with contents of .rodata section, file descriptor, the offset of .rodata section and size. In this function, for the elements in the buffer which is not empty, if the string in element has "difficult!", "difficult" changes to "funny!", and remaining blanks are sent to the end of string. Finally, the function write this modified string on original location of .rodata section, which is found by the offset.