Assignment 1-1: Practice with Linux

Build and install Linux Kernel on the Raspberry Pi 4 (RPi4).

- The goal is to familiarize yourself with the Linux development environment and setup the RPi for future assignments and projects.
- Build the Linux Kernel using Native Linux or Virtual Machine.
    - We provide a detailed guideline in another file.
    - You must use the custom kernel that we provide via Github.
- After building and installing it, you must run a provided program, which checks the kernel integrity.
- Turn in the encrypted file generated by the program to the LMS.
    - You must build the kernel by yourself. We will check with information stored in the encrypted file, which has various kernel build setups and running hardware.

Getting Help

If you need any help, send us an email.

- 김예성: yeseongkim@dgist.ac.kr
- 이호연: lhyzone@dgist.ac.kr
- 이준영: lolcy3205@dgist.ac.kr

Assignment 1-2: Bit Operations

In addition to the kernel build, we are going to practice basic C programming.
Modify the provided skeleton code and implement the following five functions in C as shown below.

- You should use bit operations to solve the problems.
- Do not modify function declaration.
- You can not use any C/C++ library functions other than malloc and free. Do not include any other header files.
- If no assumption is made, all implementations should be executed without error. For example, your implementation should check all corner cases that may cause segmentation faults.
- Assume the "pointer" type is defined as "unsigned char *". (Provided with the skeleton)

```
void print_bit(pointer a, int len); (2pt)
void reverse_bit(pointer a, int len); (2pt)
void split_bit(pointer a, pointer out1, pointer out2, int len); (2pt)
unsigned int convert_endian(unsigned int a); (2pt)
void get_date(unsigned int date, int* pYear, int* pMonth, int* pDay);  (2pt)
```

Function Specification

---

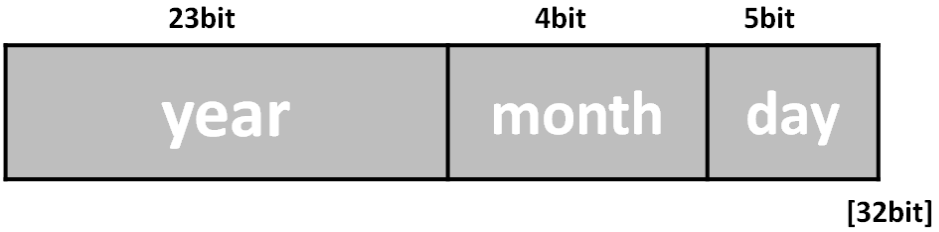
1. void print_bit(pointer a, int len);
    a. Print the bit representation for given byte data.
    b. e.g., for unsigned int a = 0x1234CDFF, the function will print 11101111 11001101 00110100 00010010(2)
    c. Parameters
        i. a: data to be printed (byte array)
        ii. len: the size of a in bytes
2. void reverse_bit(pointer a, int len);
    a. Reverse the bit order for the data stored in a
    b. e.g., for two-byte data, a = 1111000010111101(2), the function manipulates it by 1011110100001111(2).
    c. Parameters:
        i. a: data to be reversed (byte array)
        ii. len: the size of a in bytes

3. void split_bit(pointer a, pointer out1, pointer out2, int len);
    a. For the data stored in a, take all odd bits and store it into out1 / take all even bits and store it into out2

b. e.g., for e.g., for two-byte data, a = 1111000010111101(2), the function stores 11001110(2) and 11000111(2) to out1 and out2, respectively.
c. Assume the length of a, i.e., len, is multiplies of 2. Also assume that the byte lengths in out1 and out2 are len/2.
d. Parameters:
    i. a: data to be split (byte array)
    ii. out1 / out2: odd/even bits as the outputs
    iii. len: the size of a in bytes

4. unsigned int convert_endian(unsigned int a);
    a. Convert the endian of the data given in the variable a to the other way
    b. For example, assuming that a is formatted with the little endian, you can change it to the big endian, or vice versa
    c. e.g., for a = 0x12345678, the return value will be 0x78563412
    d. Parameters:
        i. a: the unsigned integer value to be converted

5. void get_date(unsigned int date, int* pYear, int* pMonth, int* pDay);
    a. Assume the date is encoded with an unsigned integer, date, as follows:



    b. Decode it and store the results into pYear, pMonth and pDay.

Sample

Note: We will check with other example inputs for grading.

Sample Code

```c
int main() {
    printf("Problem 1\n");
    unsigned int v1 = 0x1234CDEF;
    print_bit((pointer)&v1, sizeof(v1));
    reverse_bit((pointer)&v1, sizeof(v1));
    print_bit((pointer)&v1, sizeof(v1));

    printf("Problem 2\n");
    unsigned int v2 = 0x1234CDEF;
    unsigned short out1 = 0, out2 = 0;
    print_bit((pointer)&v2, sizeof(v2));
    split_bit((pointer)&v2, (pointer)&out1, (pointer)&out2, sizeof(v2));
    print_bit((pointer)&out1, sizeof(out1));
    print_bit((pointer)&out2, sizeof(out2));

    printf("Problem 3\n");
    unsigned int v3 = 0x12345678;
    unsigned int v3_ret = convert_endian(v3);
    print_bit((pointer)&v3, sizeof(v3));
    print_bit((pointer)&v3_ret, sizeof(v3_ret));

    printf("Problem 4\n");
    unsigned int date = 1035391;
    int year, month, day;
    print_bit((pointer)&date, sizeof(date));
    get_date(date, &year, &month, &day);
    printf("%d -> %d/%d/%d\n", date, year, month, day);

    return 0;
}
```

Output

```
Problem 1
11101111 11001101 00110100 00010010
01001000 00101100 10110011 11110111
Problem 2
11101111 11001101 00110100 00010010
11111010 01000001
10111011 01100100
Problem 3
01111000 01010110 00110100 00010010
00010010 00110100 01010110 01111000
Problem 4
01111111 11001100 00001111 00000000
1035391 -> 2022/3/31
```

Due

Submit your files before Apr 12, Friday, 11:59:59pm, to LMS.

Submission

1. For Assignment 1-1, submit the encrypted file that you finally get. Before the submission, rename the text file with this format: “hw1_YOURSTUDENTID.txt”. For example, “hw1_200012345.txt”
2. For Assignment 1-2, submit the c file that has your implementation. You have to implement all the functions in a single c file. Before the submission, rename the c file with this format:
   “hw1_YOURSTUDENTID.c”. For example, “hw1_200012345.c”