[toc]

课上测试

ch06

作业题目：商用密码标准实现

完成下面任务（29分）

1.在 Ubuntu 或 openEuler 中完成任务（推荐openEuler）

2.简述 GM/T0009 4种数据转换的功能，根据你的理解，每种转换功能给出至少一个例子 （8分）

3.参考课程代码sdfproject，基于一个模块utils.c，utils.h使用四个函数分别实现4种数据转换的功能（12分）

4.src中在testsdf.c中编写main函数 测试四个转换功能。 （6分）

5.提交git log结果（3分）

作业提交要求 (1')

0. 记录实践过程和 AI 问答过程，尽量不要截图，给出文本内容
1. (选做)推荐所有作业托管到 gitee或 github 上
2. (必做)提交作业 markdown文档，命名为“学号-姓名-作业题目.md”
3. (必做)提交作业 markdown文档转成的 PDF 文件，命名为“学号-姓名-作业题目.pdf”

- github链接

作业内容

简述 GM/T0009 4种数据转换的功能，根据你的理解，每种转换功能给出至少一个例子

- 四种数据转换功能:
    1. **位串到8位字节串的转换**:
        - 功能：将位串转换为8位字节串，如果位串长度不是8的整数倍，需要在左边补0以保证长度为8的倍数。
        - 例子：假设有一个位串`10110110`，需要转换为8位字节串。由于长度已经是8的倍数，直接转换为`10110110`。如果位串是`101101`，需要补0变为`101101000`，然后转换为`A3`（16进制表示）。
    2. **8位字节串到位串的转换**:
        - 功能：将8位字节串转换为位串。
        - 例子：假设有一个8位字节串`A3`（16进制表示），转换为位串就是`10100001`。
    3. **整数到8位字节串的转换**:
        - 功能：将一个整数转换为8位字节串，基本方法是先将整数用二进制表达，然后将结果位串转换为8位字节串。
        - 例子：假设有一个整数`255`，其二进制表示为`11111111`，转换为8位字节串就是`FF`（16进制表示）。

- 4. **8位字节串到整数的转换**：
  - 功能：将8位字节串转换为整数，可以简单地把8位字节串看成以256为基表示的整数。
  - 例子：假设有一个8位字节串`FF`（16进制表示），转换为整数就是`255`。

参考课程代码sdfproject，基于一个模块utils.c，utils.h使用四个函数分别实现4种数据转换的功能

- utils.h代码

```
#ifndef UTILS_H
#define UTILS_H

// 位串转8位字节串函数声明
void bitstring_to_bytes(char *bitstring, int blen, char *bytes);

// 8位字节串转位串函数声明
void bytes_to_bitstring(char* bytes, int mlen, char* bitstring);

// 8位字节串转整数函数声明
unsigned int bytes_to_int(unsigned char* M, int mlen);

// 整数转8位字节串函数声明
void int_to_bytes(int x, int mlen, unsigned char *M);

#endif
```

- utils.c代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 位串转8位字节串函数实现，与代码1中的函数实现一致
void bitstring_to_bytes(char *bitstring, int blen, char *bytes) {
    int mlen = (blen + 7) / 8;
    for (int i = 0; i < mlen; i++) {
        bytes[i] = 0;
        for (int j = 0; j < 8; j++) {
            int index = blen - 8 * (mlen - 1 - i) - j - 1;
            if (index >= 0 && bitstring[index] == '1') {
                bytes[i] |= 1 << j;
            }
        }
    }
}

// 8位字节串转位串函数实现，与代码2中的函数实现一致
void bytes_to_bitstring(char* bytes, int mlen, char* bitstring) {
    int index = 0;
    for (int i = 0; i < mlen; ++i) {
        char binary[9];
```

```
            byte_to_binary(bytes[i], binary);
            for (int j = 0; j < 8; ++j) {
                bitstring[index++] = binary[j];
            }
        }
        bitstring[index] = '\0';
    }

    // 辅助函数，将字节转换为二进制字符串，代码来自代码2
    void byte_to_binary(char byte, char* binary_str) {
        for (int i = 7; i >= 0; --i) {
            binary_str[7 - i] = ((byte & (1 << i))? '1' : '0');
        }
        binary_str[8] = '\0';
    }

    // 8位字节串转整数函数实现，与代码3中的函数实现一致
    unsigned int bytes_to_int(unsigned char* M, int mlen) {
        unsigned int x = 0;
        for (int i = 0; i < mlen; i++) {
            x += (1 << (8 * (mlen - 1 - i))) * M[i];
        }
        return x;
    }

    // 整数转8位字节串函数实现，与代码4中的函数实现一致
    void int_to_bytes(int x, int mlen, unsigned char *M) {
        for (int i = 0; i < mlen; i++) {
            M[i] = (x >> (8 * (mlen - 1 - i))) & 0xFF;
        }
    }
```

- 功能测试
  - testsdf.c代码

```
    #include <stdio.h>
    #include "utils.h"

    int main() {
        // 示例1：位串转8位字节串使用示例
        char bitstring[1000] = "1010101010101010";
        int blen = strlen(bitstring);
        char bytes[1000];
        bitstring_to_bytes(bitstring, blen, bytes);
        int mlen = (blen + 7) / 8;
        printf("位串转字节串示例结果：");
        for (int i = 0; i < mlen; i++) {
            printf("%02X ", (unsigned char)bytes[i]);
        }
        printf("\n");
```

```
    // 示例2：8位字节串转位串使用示例
    char bytes2[] = {0x12, 0x34, 0x56};
    int mlen2 = sizeof(bytes2) / sizeof(bytes2[0]);
    char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
    bytes_to_bitstring(bytes2, mlen2, bitstring2);
    printf("字节串转位串示例结果：%s\n", bitstring2);
    free(bitstring2);

    // 示例3：8位字节串转整数使用示例
    unsigned char M[] = {0x12, 0x34};
    int mlen3 = sizeof(M) / sizeof(M[0]);
    unsigned int x = bytes_to_int(M, mlen3);
    printf("字节串转整数示例结果：%u\n", x);

    // 示例4：整数转8位字节串使用示例
    int num = 12345;
    int mlen4 = 2;
    unsigned char *M2 = (unsigned char *)malloc(mlen4);
    int_to_bytes(num, mlen4, M2);
    printf("整数转字节串示例结果：");
    for (int i = 0; i < mlen4; i++) {
        printf("%02X ", M2[i]);
    }
    printf("\n");
    free(M2);

    return 0;
}
```

- 编译运行测试代码

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/test# nano testsdf.c
root@Youer:~/shiyan/test/bestidiocs2024/ch05/test# gcc -o my_program utils.c
testsdf.c
utils.c: In function ‘bytes_to_bitstring’:
utils.c:24:9: warning: implicit declaration of function ‘byte_to_binary’ [-
Wimplicit-function-declaration]
24 |         byte_to_binary(bytes[i], binary);
   |         ^~~~~~~~~~~~~~
utils.c: At top level:
utils.c:33:6: warning: conflicting types for ‘byte_to_binary’; have
‘void(char,  char *)’
33 | void byte_to_binary(char byte, char* binary_str) {
   |      ^~~~~~~~~~~~~~
utils.c:24:9: note: previous implicit declaration of ‘byte_to_binary’ with
type ‘void(char,  char *)’
24 |         byte_to_binary(bytes[i], binary);
   |         ^~~~~~~~~~~~~~
testsdf.c: In function ‘main’:
testsdf.c:7:16: warning: implicit declaration of function ‘strlen’ [-
Wimplicit-function-declaration]
    7 |     int blen = strlen(bitstring);
```

```
    |                ^~~~~~
testsdf.c:3:1: note: include ‘<string.h>’ or provide a declaration of
‘strlen’
    2 | #include "utils.h"
+++ |+#include <string.h>
    3 |
testsdf.c:7:16: warning: incompatible implicit declaration of built-in
function ‘strlen’ [-Wbuiltin-declaration-mismatc]
    7 |     int blen = strlen(bitstring);
    |                ^~~~~~
testsdf.c:7:16: note: include ‘<string.h>’ or provide a declaration of
‘strlen’
testsdf.c:20:31: warning: implicit declaration of function ‘malloc’ [-
Wimplicit-function-declaration]
20 |     char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
    |                               ^~~~~~
testsdf.c:3:1: note: include ‘<stdlib.h>’ or provide a declaration of
‘malloc’
    2 | #include "utils.h"
+++ |+#include <stdlib.h>
    3 |
testsdf.c:20:31: warning: incompatible implicit declaration of built-in
function ‘malloc’ [-Wbuiltin-declaration-mismatch]
20 |     char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
    |                               ^~~~~~
testsdf.c:20:31: note: include ‘<stdlib.h>’ or provide a declaration of
‘malloc’
testsdf.c:23:5: warning: implicit declaration of function ‘free’ [-
Wimplicit-function-declaration]
23 |     free(bitstring2);
    |     ^~~~
testsdf.c:23:5: note: include ‘<stdlib.h>’ or provide a declaration of
‘free’
testsdf.c:23:5: warning: incompatible implicit declaration of built-in
function ‘free’ [-Wbuiltin-declaration-mismatch]
testsdf.c:23:5: note: include ‘<stdlib.h>’ or provide a declaration of
‘free’
root@Youer:~/shiyan/test/bestidiocs2024/ch05/test# ls
my_program  testsdf.c  utils.c  utils.h
root@Youer:~/shiyan/test/bestidiocs2024/ch05/test# ./my_program
位串转字节串示例结果：AA AA
字节串转位串示例结果：000100100011010001010110
字节串转整数示例结果：4660
整数转字节串示例结果：30 39
```

- 验证运行结果是否准确：
  - 示例1：位串转8位字节串使用示例

    输入位串：`1010101010101010`
    - 长度：`blen = 16`（因为`strlen(bitstring)`返回的是字符数，每个字符代表一个位）

- 转换结果：`AA AA`（每个`AA`代表16进制的`10101010`，即`1010`（二进制）重复四次）
这个结果是正确的，因为`101010`（二进制）转换为16进制就是`AA`。

- 示例2：8位字节串转位串使用示例

输入字节串：`{0x12, 0x34, 0x56}`
- 转换结果：`000100100011010001010110`
这个结果也是正确的。`0x12`（16进制）转换为二进制是`00010010`，`0x34`是`00110100`，`0x56`是`01010110`。将它们连起来就是`000100100011010001010110`。

- 示例3：8位字节串转整数使用示例

输入字节串：`{0x12, 0x34}`
- 转换结果：`4660`
这个结果也是正确的。`0x1234`（16进制）转换为十进制就是`4660`。

- 示例4：整数转8位字节串使用示例

输入整数：`12345`
- 转换结果：`30 39`
这个结果也是正确的。整数`12345`在内存中的二进制表示（大端序）是`0x30 0x39`（16进制）。`12345`的二进制表示是`0110 0100 0011 0101 1001`，分成两个8位字节就是`0110 0100`和`0011 0101 1001`，即`0x30`和`0x39`。

- 综上所述，运行结果都是正确的。

src中在testsdf.c中编写main函数 测试四个转换功能

- 将代码文件等移动到sdftest中对应文件夹中
- 修改原先的testsdf.c文件，调用四个转换函数进行测试

```c
#include <stdio.h>
#include "sdf.h"
#include "utils.h"  // 引入之前定义的数据转换功能头文件

int main() {
    HANDLE deviceHandle;
    LONG openResult, closeResult;

    // 打开设备
    openResult = SDF_OpenDevice(&deviceHandle);
    if (openResult == 0) {
        printf("设备打开成功。\n");
```

```
        // 示例1：位串转字节串使用示例
        char bitstring[1000] = "1010101010101010";
        int blen = strlen(bitstring);
        char bytes[1000];
        bitstring_to_bytes(bitstring, blen, bytes);
        int mlen = (blen + 7) / 8;
        printf("位串转字节串示例结果：");
        for (int i = 0; i < mlen; i++) {
            printf("%02X ", (unsigned char)bytes[i]);
        }
        printf("\n");

        // 示例2：字节串转位串使用示例
        char bytes2[] = {0x12, 0x34, 0x56};
        int mlen2 = sizeof(bytes2) / sizeof(bytes2[0]);
        char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
        bytes_to_bitstring(bytes2, mlen2, bitstring2);
        printf("字节串转位串示例结果：%s\n", bitstring2);
        free(bitstring2);

        // 示例3：字节串转整数使用示例
        unsigned char M[] = {0x12, 0x34};
        int mlen3 = sizeof(M) / sizeof(M[0]);
        unsigned int x = bytes_to_int(M, mlen3);
        printf("字节串转整数示例结果：%u\n", x);

        // 示例4：整数转字节串使用示例
        int num = 12345;
        int mlen4 = 2;
        unsigned char *M2 = (unsigned char *)malloc(mlen4);
        int_to_bytes(num, mlen4, M2);
        printf("整数转字节串示例结果：");
        for (int i = 0; i < mlen4; i++) {
            printf("%02X ", M2[i]);
        }
        printf("\n");
        free(M2);

        // 关闭设备
        closeResult = SDF_CloseDevice(deviceHandle);
        if (closeResult == 0) {
            printf("设备关闭成功。\n");
        } else {
            printf("设备关闭失败，错误代码：%ld\n", closeResult);
        }
    } else {
        printf("设备打开失败，错误代码：%ld\n", openResult);
    }

    return 0;
}
```

- 编写makefile文件

```
# 编译器
CC = gcc

# 编译选项
CFLAGS = -I./include

# 目标可执行文件名称
TARGET = bin/sdftest

# 获取src文件夹下所有的.c源文件
SRC_FILES := $(wildcard src/*.c)

# 根据源文件生成对应的.o目标文件，放置在obj文件夹
OBJ_FILES := $(patsubst src/%.c, obj/%.o, $(SRC_FILES))

# 默认目标，用于生成最终的可执行文件
all: $(TARGET)

# 链接步骤，将所有的.o目标文件链接成最终的可执行文件
$(TARGET): $(OBJ_FILES)
    $(CC) $(CFLAGS) -o $@ $^

# 编译规则，将每个.c源文件编译成对应的.o目标文件
obj/%.o: src/%.c
    $(CC) $(CFLAGS) -c -o $@ $<

# 清理规则，用于删除生成的目标文件和可执行文件
clean:
    rm -f $(OBJ_FILES) $(TARGET)
```

- 编译运行结果：与原先结果一致，验证成功

```
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# make
gcc -I./include -c -o obj/testsdf.o src/testsdf.c
src/testsdf.c: In function ‘main’:
src/testsdf.c:16:20: warning: implicit declaration of function ‘strlen’ [-
Wimplicit-function-declaration]
   16 |         int blen = strlen(bitstring);
      |                    ^~~~~~
src/testsdf.c:4:1: note: include ‘<string.h>’ or provide a declaration of ‘strlen’
    3 | #include "utils.h"  // 引入之前定义的数据转换功能头文件
  +++ |+#include <string.h>
    4 |
src/testsdf.c:16:20: warning: incompatible implicit declaration of built-in
function ‘strlen’ [-Wbuiltin-declaration-mismatch]
   16 |         int blen = strlen(bitstring);
      |                    ^~~~~~
src/testsdf.c:16:20: note: include ‘<string.h>’ or provide a declaration of
```

```
‘strlen’
src/testsdf.c:29:35: warning: implicit declaration of function ‘malloc’ [-
Wimplicit-function-declaration]
   29 |         char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
      |                                   ^~~~~~
src/testsdf.c:4:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘malloc’
    3 | #include "utils.h"  // 引入之前定义的数据转换功能头文件
  +++ |+#include <stdlib.h>
    4 |
src/testsdf.c:29:35: warning: incompatible implicit declaration of built-in
function ‘malloc’ [-Wbuiltin-declaration-mismatch]
   29 |         char* bitstring2 = (char*)malloc(mlen2 * 8 + 1);
      |                                   ^~~~~~
src/testsdf.c:29:35: note: include ‘<stdlib.h>’ or provide a declaration of
‘malloc’
src/testsdf.c:32:9: warning: implicit declaration of function ‘free’ [-Wimplicit-
function-declaration]
   32 |         free(bitstring2);
      |         ^~~~
src/testsdf.c:32:9: note: include ‘<stdlib.h>’ or provide a declaration of ‘free’
src/testsdf.c:32:9: warning: incompatible implicit declaration of built-in
function ‘free’ [-Wbuiltin-declaration-mismatch]
src/testsdf.c:32:9: note: include ‘<stdlib.h>’ or provide a declaration of ‘free’
src/testsdf.c:57:47: warning: format ‘%ld’ expects argument of type ‘long int’,
but argument 2 has type ‘LONG’ {aka ‘int’} [-Wformat=]
   57 |             printf("设备关闭失败，错误代码：%ld\n", closeResult);
      |                                              ~~^     ~~~~~~~~~~~
      |                                                |     |
      |                                                |     LONG {aka int}
      |                                                long int
      |                                               %d
src/testsdf.c:60:43: warning: format ‘%ld’ expects argument of type ‘long int’,
but argument 2 has type ‘LONG’ {aka ‘int’} [-Wformat=]
   60 |         printf("设备打开失败，错误代码：%ld\n", openResult);
      |                                           ~~^     ~~~~~~~~~~
      |                                             |     |
      |                                             |     LONG {aka int}
      |                                             long int
      |                                            %d
gcc -I./include -o bin/sdftest obj/sdf.o obj/testsdf.o obj/utils.o
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# ls
Makefile  bin  include  obj  src
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# ./bin/sdftest
设备打开成功。
位串转字节串示例结果：AA AA
字节串转位串示例结果：000100100011010001010110
字节串转整数示例结果：4660
整数转字节串示例结果：30 39
设备关闭成功。
```

- 最终项目结构

```
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# tree
.
├── Makefile
├── bin
│   └── sdftest
├── include
│   ├── sdf.h
│   └── utils.h
├── obj
│   ├── sdf.o
│   ├── testsdf.o
│   └── utils.o
└── src
    ├── sdf.c
    ├── testsdf.c
    └── utils.c

4 directories, 10 files
```

提交git log结果（3分）

- 由于之前忘记克隆下来的仓库本身也可以提交，所以分成了两部分记录
- ch05/test文件夹（新仓库）中的git log记录

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/test# git log
commit 9bbf5f9ac206c58bb2324a3779025ca100feb5ff (HEAD -> master)
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:14:16 2024 +0800

    finish test

commit 528f6b71272f01aefe38b125712a4bf46181a1d4
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:06:14 2024 +0800

    finish utils.c

commit 892547df24f429b21c941eaf8abfcfdea057cdc0
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:05:16 2024 +0800

    finish  utils.h
```

- sdftest文件夹中的git log 记录

```
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# git add .
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# git commit -m "finish
sdftest"
```

```
[master 6a7314d] finish sdftest
 6 files changed, 142 insertions(+), 6 deletions(-)
 create mode 100755 ch06/sdftest/bin/sdftest
 delete mode 100644 ch06/sdftest/bin/testsdf
 create mode 100644 ch06/sdftest/include/utils.h
 create mode 100644 ch06/sdftest/src/utils.c
root@Youer:~/shiyan/test/bestidiocs2024/ch06/sdftest# git log
commit 6a7314d6c596130c238a1559a9aaa353b9f46c89 (HEAD -> master)
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:29:01 2024 +0800

    finish sdftest
```