3.2 编译器gcc的使用

1. 多文件编译

（1）创建一个文件夹

mkdir

（2）创建三个.c文件

（2.1）other1.c

```
void welcome() {
    printf("Welcome to the world of Linux\n");

}
```

（2.2）other2.c

```
int add(int x, int y) {
    return x+y;
}
int sub(int x, int y){
    return x-y;
}
```

（2.3）app.c

```
#include <stdio.h>.
    void main(){
int a=15,b=3,c;
printf("test in app\n");
welcome();
c = add(a, b);
printf("%d + %d = %d\n", a, b, c);
 c = sub(a, b);
printf("%d - %d = %d\n", a, b, c);
}
```

（3）编译

```
gcc other1.c other2.c app.c -o app
```

（4）运行

```
./app
```

2. 增加头文件

（1）增加头文件

（1.1）other1.h

```
#ifndef OTHER1_H
#define OTHER1_H
#include <stdio.h>
    void welcome();
#endif
```

（1.2）other2.h

```
#ifndef OTHER2_H
#define OTHER2_H
#include <stdio.h>
        int add(int, int);
        int sub(int, int);
#endif
```

（2）修改other1.c和other2.c

（2.1）other1.c

```
#include “other1.h”
        void welcome() {
 printf(“Welcome to the world of Linux\n”);
}
```

（2.2）other2.c

```
#include “other2.h”
int add(int x, int y) {
return x+y;
}

int sub(int x, int y){
return x-y;
}
```

（3）修改app.c

```
#include “other1.h”
#include “other2.h”
void main(){
int a=15,b=3,c;
printf(“test in app\n”);
welcome();
c = add(a, b);
printf(“%d + %d = %d\n”, a, b, c);
 c = sub(a, b);
printf(“%d - %d = %d\n”, a, b, c);
}
```

（4）编译并运行

```
gcc other1.c other2.c app.c -o app

./app
```

3.2 编译器gcc的使用

Linux库的创建与使用

什么是库

静态库的创建和使用

动态库的创建和使用

1. 什么是库?

（1）库：事先已经编译好的代码，经过编译后可以直接调用的文件，本质上来说是一种l执行代码的二进制形式，可以被操作系统载入内存执行。

（2）系统提供的库的路径

/usr/lib

/usr/lib64

（3）Linux库文件名的组成

前缀（lib）+库名+后缀（.a静态库；.so动态库）

libmm.a：库名为mm的静态库；

libnn.so：库名为nn的动态库。

2. 静态库与动态库

载入的顺序是不一样的

（1）静态库的代码在编译时就拷贝到应用程序中，因此当有多个程序同时引用一个静态库函数时，内存中将会调用函数的多个副本。由于是完全拷贝，因此一旦连接成功，静态库就不再需要了，代码体积大。

（2）动态库在程序内留下一个标记，指明当程序执行时，首先必须要载入这些库。在程序开始运行后调用库函数时才被载入，被调用函数在内存中只有一个副本，代码体积小。