知识点1【makefile的概述】（了解）

```
gcc a.c b.c c.c -o main
```

如果只修改了b.c 使用gcc编译 需要对所有文件重新编译。

make 是个命令，是个可执行程序，用来解析 Makefile 文件的命令

makefile 是个文件，这个文件中描述了咱们程序的编译规则。

采用 Makefile 的好处：

1、简化编译程序的时候输入得命令，编译的时候只需要敲 make 命令就可以了

2、可以节省编译时间，提高编译效率

知识点2【makefile的语法规则】（了解）

```
目标:依赖文件
 命令列表
```

gcc a.c b.c c.c -o main

目标：就是需要生成的文件

依赖文件：通过依赖文件 生成 目标文件

命令列表：实现 将依赖文件 生成 目标文件

```
main:a.c b.c c.c
 gcc a.c b.c c.c -o main
```

00_code.c

```c
#include <stdio.h>
int main(int argc, char const *argv[])
{
    printf("hello world\n");
```

```
    return 0;
}
```

makefile

```
main:00_code.c
    gcc 00_code.c -o main
```

```
edu@edu:~/work/c/day11$ make
gcc 00_code.c -o main
edu@edu:~/work/c/day11$ ls
00_code.c  main  makefile
edu@edu:~/work/c/day11$
```

make 默认在工作目录中寻找名为 GNUmakefile、makefile、Makefile 的文件作为 makefile 输入文件

make -f 自定义makefle文件名

make 工具默认会实现 makefile 文件内的第一个目标。

make 目标 ----->选择目标执行

知识点3【makefile的变量】（了解）

1、自定义变量

```
变量名=变量值
```

引用变量： $(变量名)或${变量名}

```
cc=gcc
exec=main
obj=main.o fun.o
$(exec):$(obj)
    $(cc) $(obj) -o $(exec)
main.o:main.c
    $(cc) -c main.c -o main.o
fun.o:fun.c
    $(cc) -c fun.c -o fun.o
clean:
    rm *.o $(exec)
```

2、系统环境变量

make 可以识别系统环境变量，在 makefile 中可直接读取或修改拷贝后的变量。

查看环境变量：env

3、预定义变量

makefile 中有许多预定义变量，这些变量具有特殊的含义，可在 makefile 中直接使用。

$@ 目标名
$< 依赖文件列表中的第一个文件
$^ 依赖文件列表中除去重复文件的部分

AR 归档维护程序的程序名，默认值为 ar
ARFLAGS 归档维护程序的选项
AS 汇编程序的名称，默认值为 as
ASFLAGS 汇编程序的选项
CC C 编译器的名称，默认值为 cc
CFLAGS C 编译器的选项
CPP C 预编译器的名称，默认值为$(CC) -E
CPPFLAGS C 预编译的选项
CXX C++编译器的名称，默认值为 g++
CXXFLAGS C++编译器的选项

高级的makefile

```
cc=gcc
exec=main
obj=main.o fun.o
flags=-Wall
$(exec):$(obj)
    $(cc) $^ -o $@ $(flags)
%.o:%.c
    $(cc) -c $< -o $@ $(flags)

clean:
    rm $(obj) $(exec)
```