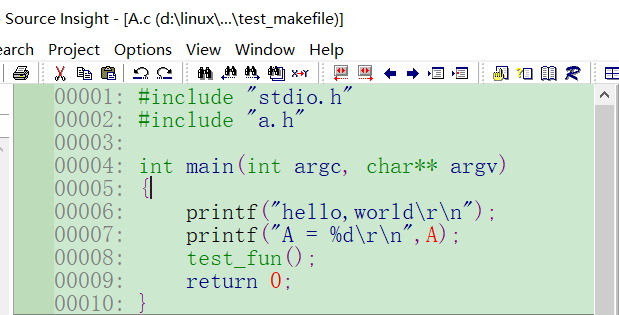
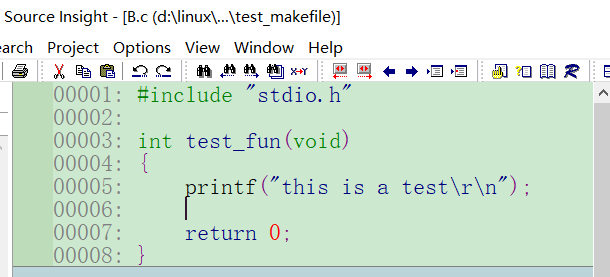
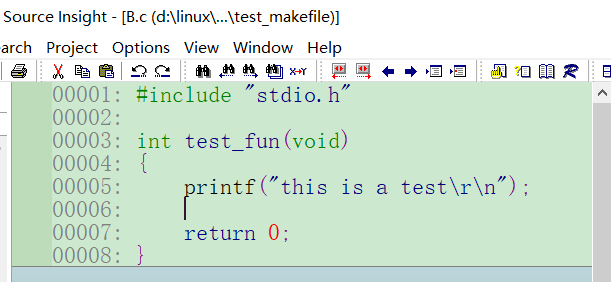
首先简单理解makefile中依赖关系，某个文件依赖于a.h，如果a.h文件被改动，所以依赖于a.h的文件会被全部重新编译，反之如果不动则不会。举例如下

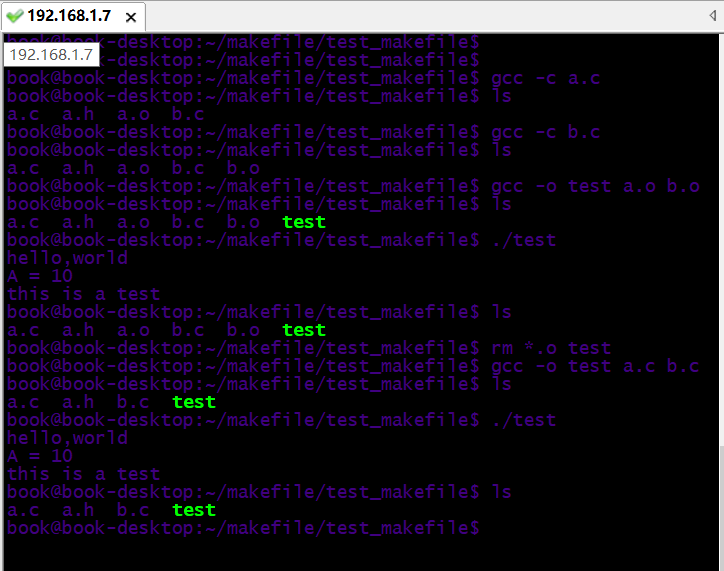
有 a.c b.c a.h三个文件







ubuntu中操作如图



gcc –c a.c/gcc a.c/gcc -o

gcc –c只编译C文件，成功输出文件a.o

gcc –o是将目标文件连接成可执行二进制文件

-c -o是不能同时使用的

gcc –o test a.c b.c

对于a.c b.c均会执行预处理，编译，汇编的操作

最后用-o链接成可执行文件

写makefile

核心：规则

目标：依赖1 依赖2 依赖3……

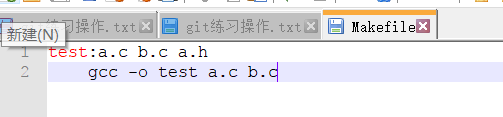
TAB命令

命令执行的条件：

i 依赖 比 目标 的时间新

ii 没有“目标”这个文件

举例



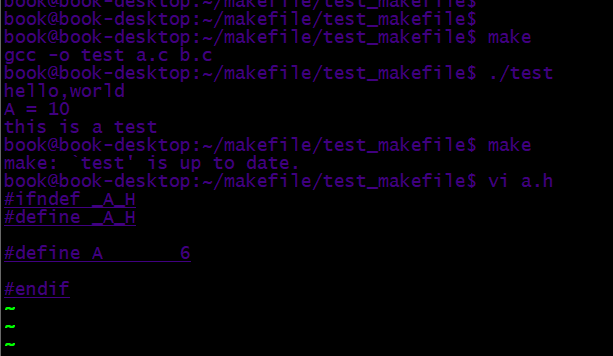
我们有makefile这个文件后，一执行make操作就回去自动读取当前路径下的makefile，然后根据规则看是否执行。如上图所示，test是目标文件，没有的话就回去执行gcc的操作。

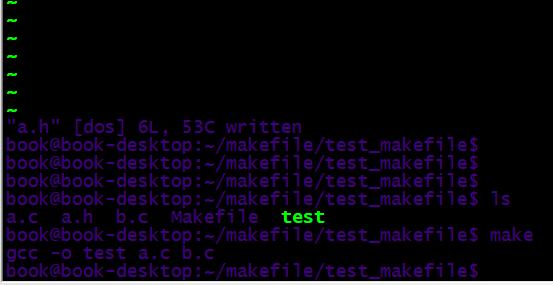
命令执行的条件

i 没有test

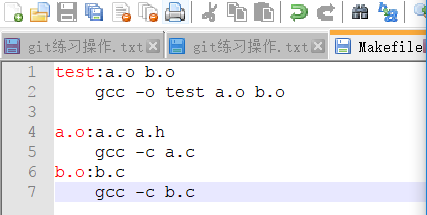
ii 依赖比目标文件新

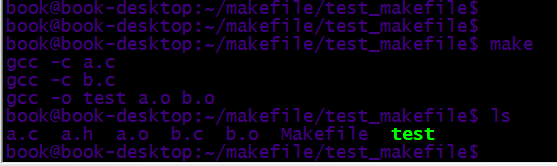
如下图所示，分别是没有目标文件以及依赖更新的情况下执行make操作的结果：





这是第一个makefile，比较low，我们可以继续优化，如下所示：





首先用a.o b.o去生成test，a.o不存在继续往下找，找到为止。

所以执行的顺序为

gcc -c a.c

gcc -c b.c

gcc -o test a.o b.o

核心：

目标：依赖1 依赖2

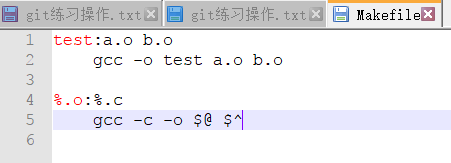
生成目标命令

依赖1：依赖1.1 依赖1.2

生成依赖1命令

这个makefile还是有缺陷，如果我有很多依赖怎么办？

通配符的引入

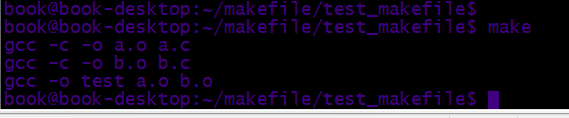


$@:目标文件

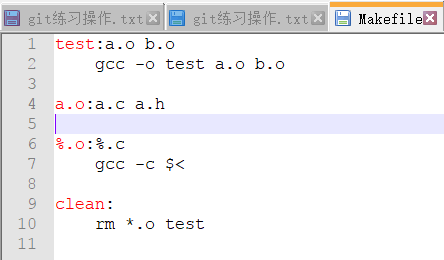
$<:第一个依赖

$^:所有的依赖

%：通配符，所有的



这个makefile更智能一点，但是有个地方不对，a.o依赖于a.h的，没有写进去，怎么办呢，这个时候你去修改a.h是不会有任何效果的。还需要改进一下。



添加了一个进去之后就可以了。