实验 2　make　& 位级运算

本实验的目标:

1. 初步掌握 make 工具
2. 熟练运用 C 语言的位级运算
3. 深刻理解整数和浮点数的表达形式

一、 make

make 是一个工具程序（Utility software），经由读取一个名字为“Makefile”的文件，自动化建构软件。它构建的目标被称为“target”；与此同时，它也检查文件的依赖关系，如果需要的话，它还会调用一些外部软件来完成任务。它的依赖关系检查规则非常简单，主要根据依赖文件的修改时间进行判断。

对于大型、复杂的 project，多个程序文件之间存在依赖和调用关系，程序员很难通过一条简单的 gcc 编译指令就能够生成整个 project 的可执行代码。我们往往会通过写 Makefile 来来确定 target 文件的依赖关系，并把生成这个 target 的相关命令传给 shell 去执行。

有了 Makefile，就可以通过相关命令 make，编译源代码，生成结果代码，然后把结果代码连接起来生成可执行文件或者库文件。

这个博客是一个简单的 Makefile 例子，帮你起步
（ http://blog.chinaunix.net/uid-25838286-id-3204219.html ）
这里有一个非常棒的 Makefile 教程: (http://wiki.wlug.org.nz/MakefileHowto).
官方手册在这儿:　（ http://www.gnu.org/software/make/manual/make.html) .
B 站上的视频，也能帮你了解它的来龙去脉:
https://www.bilibili.com/video/BV1B4411F7EK?from=search&seid=7439875720245299496

接下来开始我们的实验:
先下载 datalab.tar 文件
#解压缩:

```
$ tar -xvf datalab.tar
```

#进入文件夹

```
$ cd datalab
```

查看文件夹下的文件

```
$ ls
```

你可以看到一些头文件（ .h ）和源文件 （ .c ）以及 Makefile 文件。 这些文件之间的关系，你通过可以阅读 README 文件了解。

使用 vim 或者 gedit，或者任意一个你喜欢使用的文字编辑软件，打开 Makefile 文件，查看它的内容。

要求：在实验报告中回答以下问题：

- 本程序的编译使用哪个编译器？Which compiler is currently being used?
- 采用哪个命令，可以将所有程序全部编译？Which target is part of a rule that makes all

the compiled programs?

- 采用哪个命令，可以将所有上次编译的结果全部删除？Which target is part of a rule that deletes all the compiled programs?
- 文件中第几行生成 btest 的目标文件？What line creates the btest program from its object files? (Give its line number.)
- 文件中第几行生成 fshow 的目标文件？What line creates the fshow program from its object files? (Give its line number.)
- 如果在 Makefile 文件中用要引用变量“FOO“， 怎么表示？How would we reference a variable “FOO” in a makefile?

二、位级运算、数的编码

执行命令

$ make clean

$ make all

运行：

$ ./fshow 一个浮点数例如 34.5

你可以看到这个数的 float 机器编码

$ ./ishow 一个整数例如 20

你可以看到这个数的十六进制的机器编码

$ ./btest

你会看到程序错误的提示：

```
Score   Rating  Errors  Function
ERROR: Test allOddBits(-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test isLessOrEqual(-2147483648[0x80000000],-2147483648[0x80000000]) faile
d...
...Gives 2[0x2]. Should be 1[0x1]
ERROR: Test logicalNeg(-2147483648[0x80000000]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test floatScale2(0[0x0]) failed...
...Gives 2[0x2]. Should be 0[0x0]
ERROR: Test floatFloat2Int(0[0x0]) failed...
...Gives 2[0x2]. Should be 0[0x0]
Total points: 0/20
```

你需要做的是：修改 bit.c 文件中的几个函数，完成规定的功能，仔细阅读 bit.c 中各函数前的注释，了解各函数应该能达到的功能。这些函数有：

```
// 判断所有奇数位是否都为 1
//可以使用的运算符：~ & ^ | + << >>
int allOddBits(int x) {
   return 2;
}
```

```
//使用位级运算符实现<=
//可以使用的运算符：  ~ & ^ | + << >>
int isLessOrEqual(int x, int y) {
   return 2;
}

//使用位级运算求逻辑非！
//可以使用的运算符：  ~ & ^ | + << >>
int logicalNeg(int x) {
   return 2;
}

//求 2 乘一个浮点数，
//  可使用任意整数的合法运算符，例如: &, |, ^ , ||, &&, +，  -，   if, while
unsigned floatScale2(unsigned uf) {
   return 2;
}

//  将将浮点数 uf 转换为整数的，返回其 32 位的位级表达
//  可使用任意整数的合法运算符，例如: &, |, ^ , ||, &&, +，  -，   if, while
int floatFloat2Int(unsigned uf)
 {
   return 2;
}
```

修改后重新编译

$ make clean

$ make all

执行：

$ ./btest

如果你的实现全部正确，应该得到以下结果，这 5 个函数的得分情况如下，满分为 20 分。

```
Score   Rating  Errors  Function
 2      2       0       allOddBits
 4      4       0       isLessOrEqual
 4      4       0       logicalNeg
 5      5       0       floatScale2
 5      5       0       floatFloat2Int
Total points: 20/20
```

如果有扣分，说明函数实现没有符合要求，使用了不允许使用的运算符。使用命令

$ ./dlc bits.c

可以调用文件包中提供的规则检查器，检查哪个运算符是不合规定的。

要求：在实验报告中，把你的运行结果、以及你实现的五个函数的源代码贴上来。

注：本实验选自 CMU CSAPP，如果你想了解 CMU CSAPP：Datalab 的完整要求，可以查看：http://csapp.cs.cmu.edu/3e/labs.html 找到相关的文档和代码。