XJTU-ICS Lab 1: Data Lab

Data Lab

Data Lab 实验指南

实验简介

相信大家在前几节课中学习中已经学到了不少东西，这个小实验的目的是让大家更加熟悉计算机中信息表示的常见模式和整数的位级表示。你将通过解决一系列编程“谜题”来做到这一点。其中许多谜题可能看起来构造的非常刻意（human-made），但你会发现自己在处理它们的过程中更多地考虑了计算机中比特的表示法。

这些题目并不简单，但是相信在尝试求解他的过程中你能感受到一些乐趣，并提升一些动手与编程能力。

Enjoy and Have fun! 😉

注意事项

1. 这是一个“个人”作业，请不要抄袭或者尝试几个人一起完成哦~
2. 这些问题并不是一些完全全新的问题，我们鼓励“Google”学习解决一些代码之外的问题（环境问题/工具使用问题），但是频繁的查询代码谜题相关的解将使得这个实验与课程失去意义~

开发环境

推荐使用 ICSServer + Visual Studio Code + Remote-SSH插件 进行开发。

也自己搭建开发环境，环境要求为：

- Linux 或 WSL (推荐 Ubuntu 22.04)
- gcc-11
- GNU Make
- clang-format

实验准备

如果你已经顺利的在本地安装或者申请到了一个稳定的Linux环境，那么就可以顺利的开始你的实验~

资源下载与解压

1. 下载文件 datalab-handout.tar

2. 通过Vscode/scp等方式将文件压缩文件上传到 ICSServer 对应的 Linux 目录下
3. 通过如下命令解压文件

```
tar xvf datalab-handout.tar
```

打开编辑器或终端并连接远端服务器

详见 VSCode-Setting

实验流程

一个比较推荐的实验流程如下：

检查文件完整性

解压完成后你将获得如下的一个完整目录，请检查你下载并解压的文件中是否包含如下目录：

```
slyang@sugon:~/datalab$ ls
bits.c  bits.h  btest.c  btest.h  check-format  decl.c  fshow.c  helper.mk
```

如出现文件完整性问题请及时询问同学或者找助教解决相关问题。

尝试make

`Makefile` 文件描述了 `Linux` 系统下 `C/C++` 工程的编译规则，它用来自动化编译 `C/C++` 项目。分发下去的实验包内已经有编写好的 `Makefile` ，我们只需要在本机中尝试 `make` ，就可以获得一些 `Linux` 下的可执行文件。

（想要仔细了解Makefile相关信息：

https://opensource.com/article/18/8/what-how-makefile

https://www.math.colostate.edu/~yzhou/computer/writemakefile.html）

`make` 方法：在对应路径下命令行输入 `make` .

```
linux > make
```

如果 `make` 成功，一个典型的输出结果如下所示：

```
slyang@sugon:~/XJTU-ICS/lab1/lab1-description/datalab-handout$ make
/usr/bin/gcc -std=gnu99 -m64 -O1 -fwrapv -g -Werror -Wall -Wextra -Wstrict
```

```
/usr/bin/gcc -std=gnu99 -m64 -O1 -fwrapv -g -Werror -Wall -Wextra -Wstrict
/usr/bin/gcc -std=gnu99 -m64 -O1 -fwrapv -g -Werror -Wall -Wextra -Wstrict
```

出错处理

如果`make`不成功，并且他的报错如果如下：

```
slyang@sugon:~/datalab-handout$ make
CLANG_FORMAT=clang-format ./check-format bits.c
/bin/sh: 1: ./check-format: Permission denied
make: *** [helper.mk:126: .format-checked] Error 126
```

则尝试：

```
linux> chmod +x check-format
```

其他一些可能出现的典型环境问题和解决方法将在附录中做简要解释，一个比较推荐的通用的问题解决方法的路径是：

1. 查看报错，根据报错分析产生原因
2. 直接咨询搜索引擎或者ChatGPT
3. 若无法成功解决则咨询助教或登录相关软件官方社区询问。

尝试运行

如果成功`make`此时你的实验目录将会变成如下：

```
slyang@sugon:~/datalab$ ls
bits.c  bits.h  btest  btest.c  btest.h  check-format  decl.c  fshow  fshc
```

简单验证可用性，我们简单运行一下`make`之后获得可执行文件btest

运行方法：命令行输入

```
linux > ./btest
```

此时若运行成功，则会出现：

```
slyang@sugon:~/XJTU-ICS/lab1/lab1-description/datalab-handout$ ./btest
Score  Rating  Errors  Function
ERROR: Test implication(0L[0x0L],0L[0x0L]) failed...
...Gives 2L[0x2L]. Should be 1L[0x1L]
ERROR: Test leastBitPos(-9223372036854775808L[0x8000000000000000L]) failed
...Gives 2L[0x2L]. Should be -9223372036854775808L[0x8000000000000000L]
ERROR: Test distinctNegation(-9223372036854775808L[0x8000000000000000L]) f
...Gives 2L[0x2L]. Should be 0L[0x0L]
ERROR: Test fitsBits(-9223372036854775808L[0x8000000000000000L],1L[0x1L])
...Gives 2L[0x2L]. Should be 0L[0x0L]
ERROR: Test trueFiveEighths(-9223372036854775808L[0x8000000000000000L]) fa
...Gives 2L[0x2L]. Should be -5764607523034234880L[0xb000000000000000L]
ERROR: Test addOK(-9223372036854775808L[0x8000000000000000L],-922337203685
...Gives 2L[0x2L]. Should be 0L[0x0L]
ERROR: Test isPower2(-9223372036854775808L[0x8000000000000000L]) failed...
...Gives 2L[0x2L]. Should be 0L[0x0L]
ERROR: Test rotateLeft(-9223372036854775808L[0x8000000000000000L],0L[0x0L]
...Gives 2L[0x2L]. Should be -9223372036854775808L[0x8000000000000000L]
ERROR: Test isPalindrome(-9223372036854775808L[0x8000000000000000L]) faile
...Gives 2L[0x2L]. Should be 0L[0x0L]
ERROR: Test bitParity(-9223372036854775808L[0x8000000000000000L]) failed..
...Gives 2L[0x2L]. Should be 1L[0x1L]
ERROR: Test absVal(-9223372036854775807L[0x8000000000000001L]) failed...
...Gives 2L[0x2L]. Should be 9223372036854775807L[0x7fffffffffffffffL]
Total points: 0/33
```

出现如上或类似如上的输出结果，恭喜你！这证明您的本地环境已经准备好了~，之后就可以快乐的开始Coding啦 ~

实验任务

实验准备中，我们在实验路径看到非常多的文件，你是否感到了一丝慌张？别慌！大多数的东西都是来帮助你更好的完成你的实验，你本地中唯一需要**更改与提交的代码文件**就是“**bits.c**”(This is all you need.)。找到这个文件，采用你心仪的编辑器，开始这个实验吧！

代码编写规则：

bits.c文件包含了针对11个编程谜题的框架。在这个实验中你的任务就是在一个严格的Coding Rules（代码编写规则）之下完成每个函数编写（一个函数就代表一个小谜题）。这个严格的代码代码编写规则如下：

不可以新添加库

并不可以尝试使用

```
#include ...
```

添加新的库，我们在bits.c中包含了printf以供大家debug使用，其余只允许使用C语言基础的运算符进行求解。

！建议优先使用的C算数和逻辑运算符

具体来说，在这个实验中建议你优先尝试使用如下8个运算符解决问题：

```
! ~ & ^ | + << >>
```

这并不是一个强制要求，但是很多问题我们可以通过这些基础运算符进行解决。

可以确定的假设：

在编写代码的时候，你可以建立如下假设：

- 数据类型int 的值为32 位。
- 数据类型long 的值为64 位
- 有符号数据类型使用二进制补码表示。
- 带符号数据的右移以算术右移方式执行。
- 当移动一个w 位值时，移动量应该在0 和w－1 之间。
- 谓词运算符（Predicate operators），包括一元运算符！和二元运算符 ==、!=、<、>、<= 和 >=，无论参数类型如何，都返回 int 类型的值。

一个例子：

看完了以上的一些冗长的限制之后，你可能感受到了疑惑。没事，接下来我们将从一个小例子出发，来解释这个实验到底需要做些什么。

bits.c中的每个小的谜题都会有谜题本身的限制，限制的具体内容写在了谜题函数上面的注释格式中。一个典型的谜题例子如下：

```
/*
 * minusOne - return a value of -1
 *   Rating: 1
 */
int minusOne(void) {
  return 1L;
}
```

如上题

- 题名：minusOne
- 题目要求：return a value of -1 ， 返回一个-1
- 分值：1分

因此我们编写如下的程序代码：

```
int minusOne(void) {
  return ~0;
}
```

谜题

这个小节 **简要** 地介绍了你将要面临的小谜题~

位操作：

（请忽略图中的Max Ops限制）

如下表 1 描述了一组操作和测试位集的函数。 “Rating”字段给出了难度评级。有关更多详细信息，请参阅 bits.c 中的每个题面上的注释。函数的所有参数和返回值都是 long 类型。

Name	Description	Rating	Max Ops
`implication(x, y)`	Given input x and y, which are binary (taking the values 0 or 1), return x → y in propositional logic - 0 for false, 1 for true	2	5
`leastBitPos(x)`	Return a mask that marks the position of the least significant 1 bit. If $x = 0$, return 0	2	6
`distinctNegation(x)`	Returns 1 if x ≠ -x and 0 otherwise	2	5
`fitsBits(x, n)`	Return 1 if x can be represented as an n-bit, two's compliment integer $1 \leq n \leq 64$	2	15
`rotateLeft(x, n)`	Rotate x to the left by n bits	3	25
`bitParity(x)`	Returns 1 if x contains an odd number of 0's	4	22

Table 1: Bit-Level Manipulation Functions.

补码运算：

（请忽略图中的Max Ops限制）

表 2 描述了一组使用整数的补码表示的函数。全部参数和返回值都是 long 类型。参考bits.c中的注释了解更多信息。

Name	Description	Rating	Max Ops
`trueFiveEighths(x)`	Multiplies by $\frac{5}{8}$ rounding toward 0, avoiding errors due to overflow	4	20
`addOK(x, y)`	Determine if can compute x+y without overflow	3	20
`isPower2(x)`	Return 1 if x is a power of 2, 0 otherwise	3	20
`isPalindrome(x)`	Return 1 if bit pattern in x is equal to its mirror image	4	70
`absVal(x)`	Returns the absolute value of x	4	10

Table 2: Arithmetic Functions

评分

标准评分标准

你的分数会按照如下标准计算出来（满分为33分）

提供给你的11个小谜题，每道题的Rating（分值）不同，他们的Rating加起来一共33分，如果你可以通过他的所有检查，并且结果正确，您将得到33分~

本地自动测试你的工作

上一节中，我们简要地谈到了你的实验成绩的组成，你可能感到了慌张。我该怎么保证我本地的程序是bugfree的呢？难不成要我自己手动人肉测试吗！？这显然是一种苛求，事实上，我们将**完整的测试程序**在**实验分发包**中给了你，你可能自行使用测试程序btest去测试你的程序的正确分。这与我们最终测试你提交上来文件的方法一致，换句话说，**你如果在本地测试得到了满分，那么你在最终的考核中也会得到一致的分数**。

正确性

如果你的bits.c的程序编写满足C语言要求并可以通过编译，在本地根目录下make之后你将会获得：

```
linux > make
```

如下的目录：

```
slyang@sugon:~/datalab$ ls
bits.c  bits.h  btest  btest.c  btest.h  check-format  decl.c  fshow  fshc
```

其中btest 程序通过多次调用它们来检查 bits.c 中函数的正确性不同的参数值。要构建和使用它，请键入以下两个命令：

```
linux> make
linux> ./btest
```

需要注意的第一点是，你每次修改你的bits.c文件之后，你需要通过`make`重新build之后再进行测试：

即如果你新完成了一个谜题，你想要重新测试，则需要完成如下的流程：

```
linux> make
linux> ./btest
```

通常来说你会获得如下类型的运行结果：

```
slyang@sugon:~/XJTU-ICS/lab1/lab1-description/datalab-handout$ ./btest
Score   Rating  Errors  Function
 2  2  0  implication
 2  2  0  leastBitPos
 2  2  0  distinctNegation
 2  2  0  fitsBits
 4  4  0  trueFiveEighths
 3  3  0  addOK
 3  3  0  isPower2
 3  3  0  rotateLeft
ERROR: Test isPalindrome(-9223372036854775808L[0x8000000000000000L]) faile
...Gives 1L[0x1L]. Should be 0L[0x0L]
 4  4  0  bitParity
 4  4  0  absVal
Total points: 29/33
```

如上输出中告诉你每一题的得分，**如果你的程序出错，则会输出对应的出错的题目与出错时候的输入和输出。**

如上中，btest告诉我们：

- *Test isPalindrome*测试用例出差错
- 出错的输入是：-9223372036854775808L[0x8000000000000000L]
- 你的输出是：1L[0x1L]
- 正确的解应该是0L[0x0L]

根据输出的结果我们返回去继续修改我们的代码。

进制转换工具

在题目编写过程中，您可能需要一些工具的帮助来确定一些10进制数的16进制格式。

我们为您提供了相应的便捷工具ishow、fshow，您可以使用提供的程序 ishow 查看整数的十进制和十六进制表示形式。首先编译代码如下：

```
linux> make
```

然后使用它来检查在命令行中键入的十六进制和十进制值：

```
linux>  ./ishow 0x8000000000000000
Hex = 0x8000000000000000L,  Signed = -9223372036854775808L,  Unsigned = 92
```

他会帮助你完成进制的转换~

同理我们还提供了fshow工具，这是一个浮点数的转换器，但是这次的实验没有涉及，所以不做要求，大家有兴趣可以自行尝试，使用方法与ishow类似。

代码规范

代码规范在大型项目的维护和开发中是很重要的，虽然我们的实验并没有对代码规范做出一些硬性的要求，并不会与得分有任何的挂钩，但是为了帮助大家体验代码规范，我们为大家提供了 clang-format 工具来帮助格式化您的代码。他会帮你显式的检查你的代码的格式是否规范，如果不规范，在你尝试 `make` 时，他会出现如下报错：

```
slyang@xjtu-ics:~/1.datalab/datalab-handout$ make
CLANG_FORMAT=clang-format ./check-format bits.c
ERROR: Your code's formatting does not match clang-format.
    For details, see https://www.cs.cmu.edu/~213/codeStyle.html
    To reformat your code, run "make format".
    You must fix this before submitting to Autolab.
    Files needing reformatting:
bits.c

make: *** [helper.mk:126: .format-checked] Error 1
```

你可以使用自动化的代码格式修改工具，他会帮助你修改你的代码格式以符合规范，要调用它，运行：

```
linux> make format
```

他会帮您自动修改您的代码为符合代码格式要求的代码。

您可以修改 .clang-format 文件以反映您喜欢的代码风格。

代码提交

每次在 `datalab-handout` 目录下执行 `make` 命令时，都会生成一个名为 `<userid>-handin.zip` 的 tarball 文件（其中 `<userid>` 为Linux系统中你的用户名）。

在在线学习平台 上的作业模块中，将该文件作为附件提交即可。

一些小的建议～

- **Start Early**
- 谜题可能比较复杂，可能一时间不可以直接想出优秀的满足要求的解法，但是不要气馁，没有人可以马上给出最优的解法。一个比较好的解题尝试路径是：
  - 先尝试在纸上写出一般的解法
  - 从一般解中找出**更一般性的规律**，这些规律可能通常需要你的灵光一现，但是对相信聪明的你来说这不是问题~
- 如果您是采用自己本地的环境，那么总会出现一些依赖缺失，软件未安装等等之类的错误。请大家不要慌张，一个好的程序员需要更好的了解自己手里的工具，**而自己搭建环境往往就是其中很重要的一部分，请大家自行查询相关环境问题，自己动手解决问题（折腾（笑））的能力在计算机专业的学习中是尤为重要的~**
- 找到适合自己的好工具的能力：请大家在不断地动手中学习并找到适合自己的工具，以提高自己的效率。

最后祝大家玩的愉快~

附录

附录1 典型报错和解决方法

clang-format缺失

如果在运行时报错clang-format找不到，请您用本地的包管理工具直接去安装clang-format

- Mac：

```
OS X> brew install clang-format
```

- Ubuntu：

```
linux> sudo apt install clang-format
```

Operation not Permitted

`make`成功后，您将获得一些可执行文件，例如我们在这个实验中我们将会获得如下文件

```
btest fshow ishow
```

这些二进制文件可以在我们的Linux环境中被加载运行，所有我们尝试执行程序都会在对应目录下以如下的形式执行：

```
linux > ./filename
```

可能在运行各种可执行文件的时候经常会遇到一个报错：Operation not permitted，这其实是你并没有赋予可执行文件在你的系统中执行的权利。

典型报错场景：

```
slyang@sugon:~/datalab-handout$ make
CLANG_FORMAT=clang-format ./check-format bits.c
/bin/sh: 1: ./check-format: Permission denied
make: *** [helper.mk:126: .format-checked] Error 126
```

一个比较直接的解决方法就是：

```
linux> chmod +x filename
```

（filename是您需要添加执行权限的文件）

这也是为什么之前，在make之前，我们需要先执行

```
linux > chmod +x check-format
```

文件权限相关学习文档：Linux File Permissions and Ownership Explained with Examples