

**实验报告**

**实 验（二）**

题 目 DataLab 数据表示

专 业 计算学部

学　　 号 1190202405

班　　 级 1903005

学 生 林逸灏

指 导 教 师 史先俊

实 验 地 点 G712

实 验 日 期

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc20326083)

[1.1 实验目的 - 4 -](#_Toc20326084)

[1.2 实验环境与工具 - 4 -](#_Toc20326085)

[1.2.1 硬件环境 - 4 -](#_Toc20326086)

[1.2.2 软件环境 - 4 -](#_Toc20326087)

[1.2.3 开发工具 - 4 -](#_Toc20326088)

[1.3 实验预习 - 4 -](#_Toc20326089)

[第2章 实验环境建立 - 5 -](#_Toc20326090)

[2.1 Ubuntu下CodeBlocks安装 - 5 -](#_Toc20326091)

[2.2 64位Ubuntu下32位运行环境建立 - 5 -](#_Toc20326092)

[第3章 C语言的数据类型与存储 - 6 -](#_Toc20326093)

[3.1 类型本质（1分） - 6 -](#_Toc20326094)

[3.2 数据的位置-地址（2分） - 6 -](#_Toc20326095)

[3.3 main的参数分析（2分） - 6 -](#_Toc20326096)

[3.4 指针与字符串的区别（2分） - 6 -](#_Toc20326097)

[第4章 深入分析UTF-8编码 - 7 -](#_Toc20326098)

[4.1 提交utf8len.c子程序 - 7 -](#_Toc20326099)

[4.2 C语言的strcmp函数分析 - 7 -](#_Toc20326100)

[4.3讨论：按照姓氏笔画排序的方法实现 - 7 -](#_Toc20326101)

[第5章 数据变换与输入输出 - 8 -](#_Toc20326102)

[5.1 提交**cs\_atoi.c** - 8 -](#_Toc20326103)

[5.2 提交**cs\_atof.c** - 8 -](#_Toc20326104)

[5.3 提交**cs\_itoa.c** - 8 -](#_Toc20326105)

[5.4 提交**cs\_ftoa.c** - 8 -](#_Toc20326106)

[5.5 讨论分析OS的函数对输入输出的数据有类型要求吗 - 8 -](#_Toc20326107)

[第6章 整数表示与运算 - 9 -](#_Toc20326108)

[6.1 提交fib\_dg**.c** - 9 -](#_Toc20326109)

[6.2 提交**fib\_loop.c** - 9 -](#_Toc20326110)

[6.3 fib溢出验证 - 9 -](#_Toc20326111)

[6.4 除以0验证： - 9 -](#_Toc20326112)

[6.5 万年虫验证 - 9 -](#_Toc20326113)

[6.6 2038虫验证 - 9 -](#_Toc20326114)

[第7章 浮点数据的表示与运算 - 10 -](#_Toc20326115)

[7.1手动float编码： - 10 -](#_Toc20326116)

[7.2特殊float数据的处理 - 10 -](#_Toc20326117)

[7.3验证浮点运算的溢出 - 10 -](#_Toc20326118)

[7.4 类型转换的坑 - 10 -](#_Toc20326119)

[7.5 讨论1：有多少个int可以用float精确表示 - 10 -](#_Toc20326120)

[7.6 讨论2：怎么验证float采用的向偶数舍入呢 - 10 -](#_Toc20326121)

[7.7 讨论3：float能精确表示几个1元内的钱呢 - 10 -](#_Toc20326122)

[7.8 Float的微观与宏观世界 - 11 -](#_Toc20326123)

[7.9 讨论：浮点数的比较方法 - 11 -](#_Toc20326124)

[第8章 舍尾平衡的讨论 - 12 -](#_Toc20326125)

[8.1 描述可能出现的问题 - 12 -](#_Toc20326126)

[8.2 给出完美的解决方案 - 12 -](#_Toc20326127)

[第9章 总结 - 13 -](#_Toc20326128)

[9.1 请总结本次实验的收获 - 13 -](#_Toc20326129)

[9.2 请给出对本次实验内容的建议 - 13 -](#_Toc20326130)

[参考文献 - 14 -](#_Toc20326131)

# 第1章 实验基本信息

## 1.1 实验目的

熟练掌握计算机系统的数据表示与数据运算

通过C程序深入理解计算机运算器的底层实现与优化

掌握VS/CB/GCC等工具的使用技巧与注意事项

## 1.2 实验环境与工具

### 1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

### 1.2.2 软件环境

Windows7 64位以上；VirtualBox/Vmware 11以上；Ubuntu 16.04 LTS 64位/优麒麟 64位；

### 1.2.3 开发工具

Visual Studio 2010 64位以上；CodeBlocks；vi/vim/gpedit+gcc

## 1.3 实验预习

* **上实验课前，必须认真预习实验指导书（PPT或PDF）**
* **了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。**
* **采用sizeof在Windows的VS/CB以及Linux的CB/GCC下获得C语言每一类型在32/64位模式下的空间大小**
  + Char /short int/int/long/float/double/long long/long double/指针
* **编写C程序，计算斐波那契数列在int/long/unsigned int/unsigned long类型时，n为多少时会出错 （linux-x64）**
  + 先用递归程序实现，会出现什么问题？
  + 再用循环方式实现。

答：1.采用递归方式：

|  |
| --- |
| int fib\_recursive (datatype a) {  if (a == 0) {  return 0;  } else if (a == 1)  {  return 1;  }  return fib\_recursive(a - 1) + fib\_recursive(a - 2);  } |

时间复杂度为O(2^n)，太大了，运行时间缓慢。

2.采用循环方式：

|  |
| --- |
| datatype fib[1000];  fib[0] = 0;  fib[1] = 1;  int n;  scanf("%d", &n);  for (int i = 2; i <= n; i++) {  fib[i] = fib[i - 1] + fib[i - 2];  } |

时间复杂度为O(n)

在不同的数据类型下，n 为多少时会出错？

int : 47

long : 47

unsigned int : 48

unsigned long : 48

* **写出float/double类型最小的正数、最大的正数（非无穷）**

答：float 的最小值为：1.4E-45

最大值为：3.4028234663852886E38

double 的最小值为：4.9E-324

最大值为：1.79769313

* **按步骤写出float数-10.1在内存从低到高地址的字节值-16进制**

答：cd cc 8c bf

* **按照阶码区域写出float的最大密度区域范围及其密度，最小密度区域及其密度（表示的浮点数个数/区域长度）**

答：最大：-1.1111(23个1)×2-126~1.1111(23个1)×2-126(包含正负浮点数)；225个浮点数 ；

最小：1.0000(23个0)×2127~1.11111(23个1)×2127；223个浮点数

# 第2章 实验环境建立

## 2.1 Ubuntu下CodeBlocks安装

CodeBlocks运行界面截图：编译、运行hellolinux.c

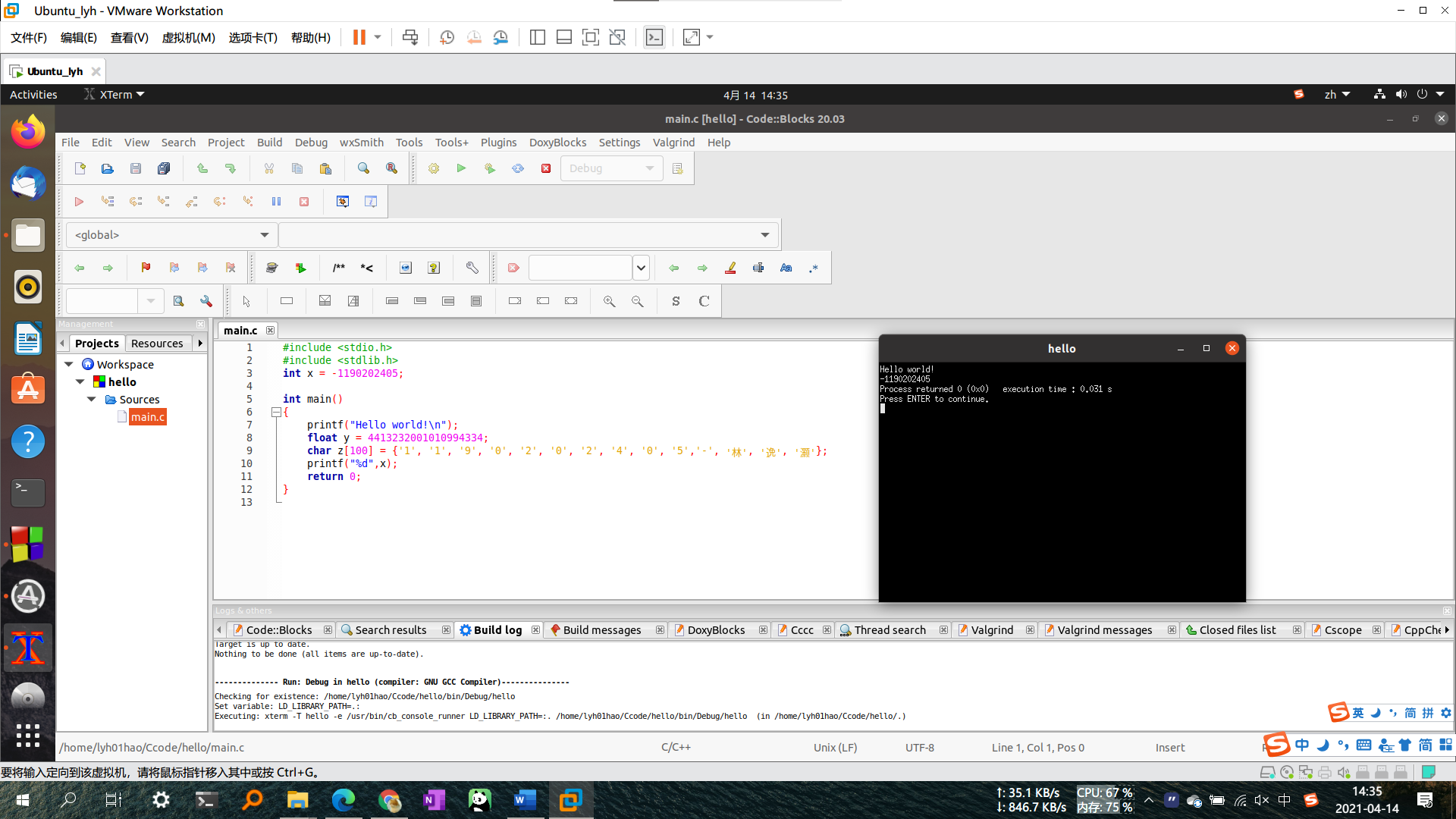


图2-1 Ubuntu下CodeBlocks截图

## 2.2 64位Ubuntu下32位运行环境建立

在终端下，用gcc的32位模式编译生成hellolinux.c。执行此文件。

Linux及终端的截图。

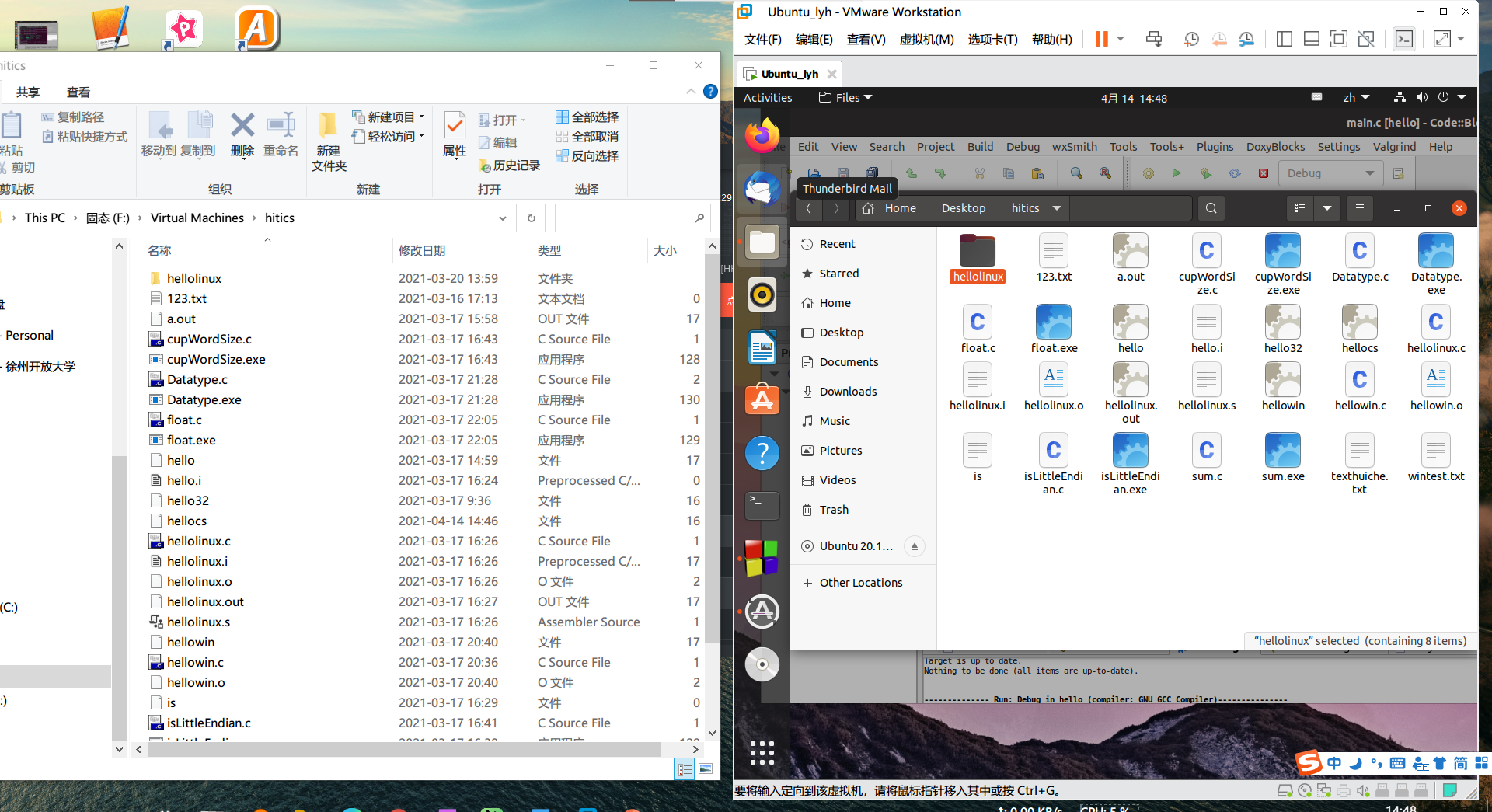
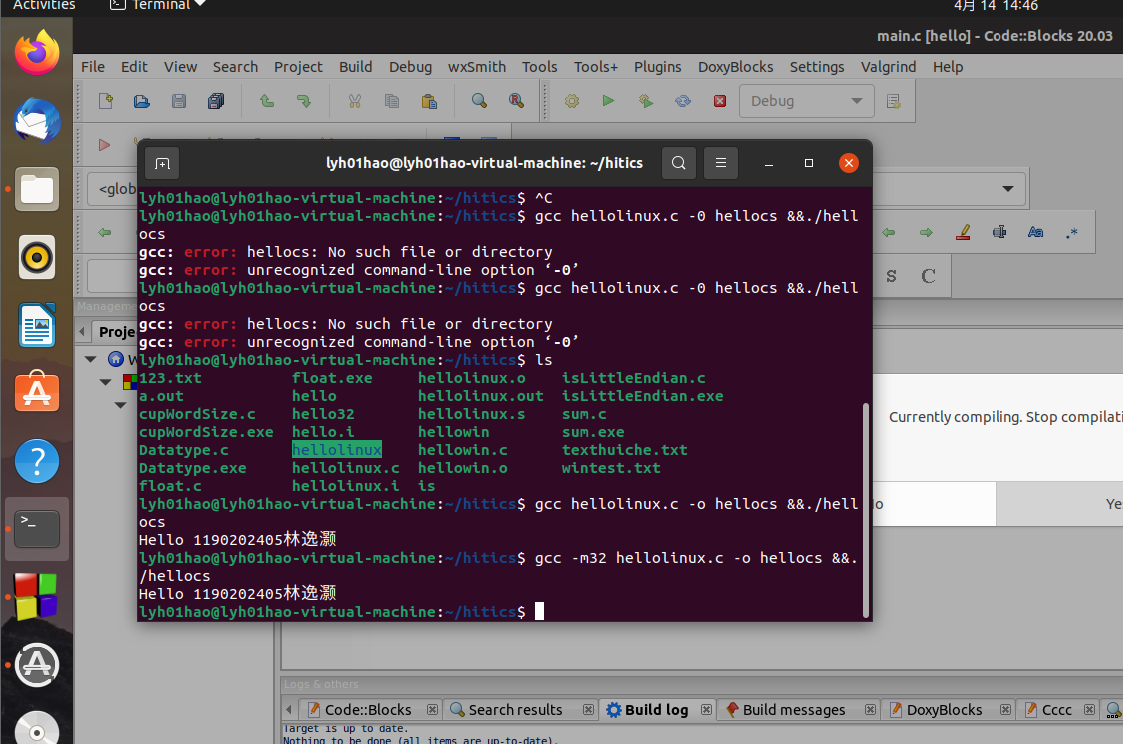


图2-2 Ubuntu与Windows共享目录截图

# 第3章 C语言的数据类型与存储

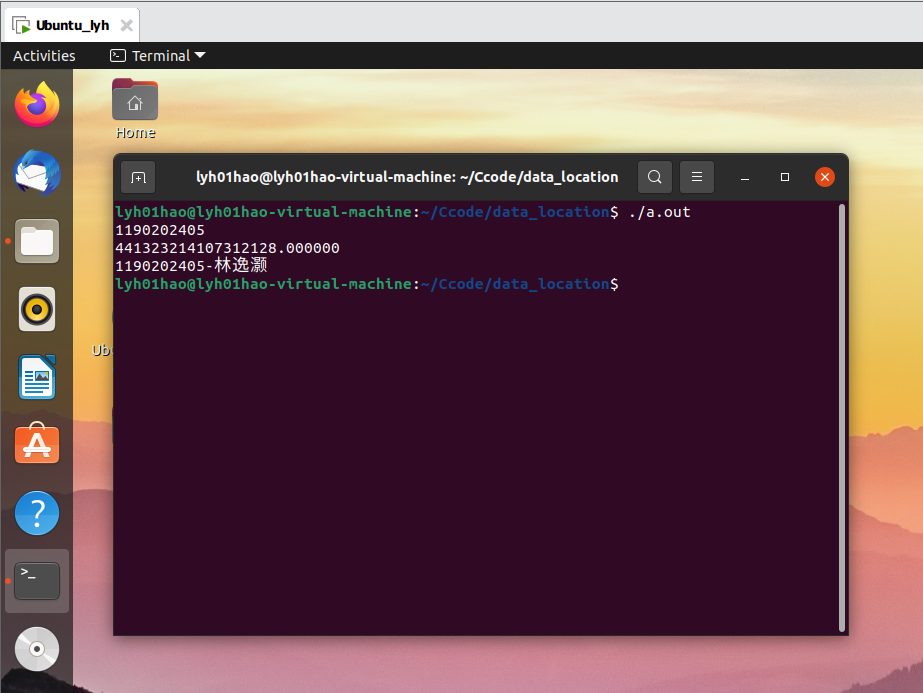
## 3.1 类型本质

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Win/VS/x86 | Win/VS/x64 | Win/CB/32 | Win/CB/64 | Linux/CB/32 | Linux/CB/64 |
| char | 1 | 1 | 1 | 1 | 1 | 1 |
| short | 2 | 2 | 2 | 2 | 2 | 2 |
| int | 4 | 4 | 4 | 4 | 4 | 4 |
| long | 4 | 4 | 4 | 8 | 4 | 8 |
| long long | 8 | 8 | 8 | 8 | 8 | 8 |
| float | 4 | 4 | 4 | 4 | 4 | 4 |
| double | 8 | 8 | 8 | 8 | 8 | 8 |
| long double | 8 | 8 | 12 | 16 | 12 | 16 |
| 指针 | 4 | 8 | 4 | 8 | 4 | 8 |

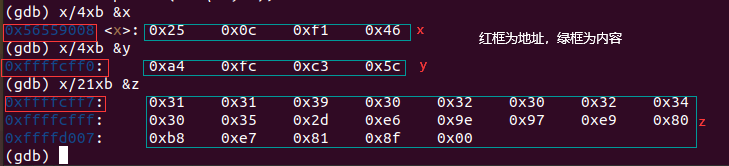
C编译器对sizeof的实现方式：sizeof 是编译时确定的，由编译器的编译参数确定，在运行时 sizeof 视为一个常量。

## 3.2 数据的位置-地址

打印x、y、z输出的值：截图1

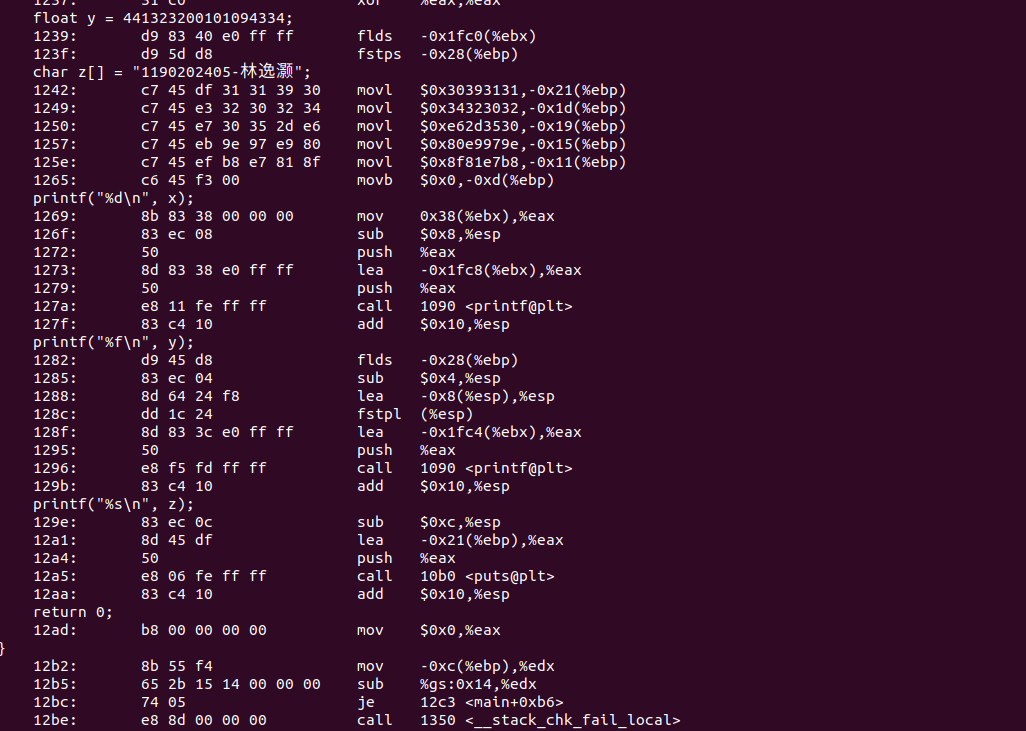


反汇编查看x、y、z的地址，每字节的内容：截图2，标注说明



x, y, z的地址与每字节的内容

反汇编查看x、y、z在代码段的表示形式。截图3，标注说明



由反汇编代码可以看出，对于x和z，通过ebx加上对应的偏置进行访问，而y是通过ebp来进行访问的。因为x和z在全局变量数据区，而y在局部变量栈中。

x与y在\_\_\_\_编译\_\_\_\_阶段转换成补码与ieee754编码。

数值型常量与变量在存储空间上的区别是：\_\_数值型常量存储在常量区，而数值型变量则存储在动态区

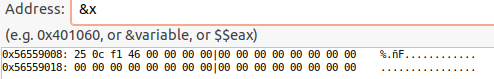
字符串常量与变量在存储空间上的区别是：\_\_\_字符串常量存储在常量区，为只读数据，而字符串变量存储在静态全局初始化区，在数据段内。

常量表达式在计算机中处理方法是：常量表达式在编译时，其值就存储在内存中，不可直接修改值的内容

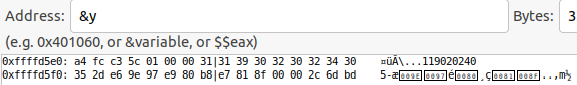
## 3.3 main的参数分析

反汇编查看x、y、z的地址，argc的地址，argv的地址与内容，截图4

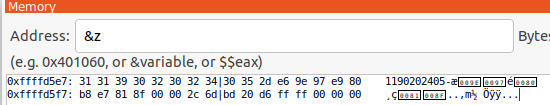
x：



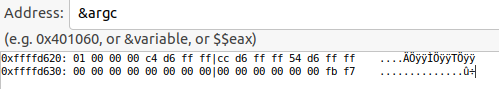
y：



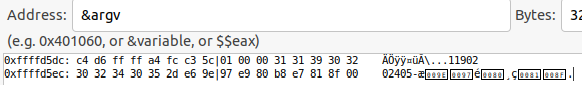
z：



argc：

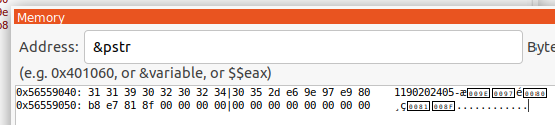
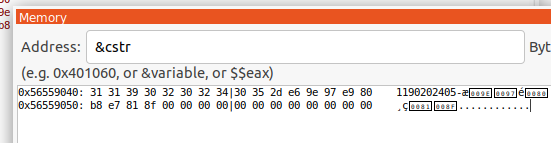


argv：



## 3.4 指针与字符串的区别

cstr的地址与内容截图，pstr的内容与截图，截图5



pstr修改内容会出现什么问题\_\_\_\_\_\_\_\_字符串指针为常量无法修改，显示段错误\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 第4章 深入分析UTF-8编码

## 4.1 提交utf8len.c子程序

## 4.2 C语言的strcmp函数分析

分析论述：strcmp到底按照什么顺序对汉字排序

按照汉字对应的 Unicode 编码进行排序。用 strcmp 比较两个由汉字组成的字符串时，先比较第一个汉字的 Unicode 编码大小，若一样，则比较下一位，以此类推。

## 4.3讨论：按照姓氏笔画排序的方法实现

分析论述：应该怎么实现呢？

创建一个数据库或新文件存储每个汉字及其对应的笔画数目，按姓氏笔画排序时，先比较汉字的笔画数目，若相同，则比较这两个汉字的 Unicode 编码，从而实现对姓氏笔画的排序。

# 第5章 数据变换与输入输出

## 5.1 提交cs\_atoi.c

## 5.2 提交cs\_atof.c

## 5.3 提交cs\_itoa.c

## 5.4 提交cs\_ftoa.c

## 5.5 讨论分析OS的函数对输入输出的数据有类型要求吗

论述如下：OS的函数对输入输出有类型要求。因为OS的程序是通过调用read和write函数来执行读取（输入）和写入（输出）的。

这两个函数的函数原型为：

|  |
| --- |
| #include<unistd.h>  ssize\_t read (int fd,void \*buf,size\_t n)  return：若成功，则为读的字节数，若是EOF则为0；若出错则为-1  ssize\_t write (int fd,void \*buf,size\_t n)  return：若成功，则为读的字节数，若出错则为-1 |

# 第6章 整数表示与运算

## 6.1 提交fib\_dg**.c**

## 6.2 提交fib\_loop.c

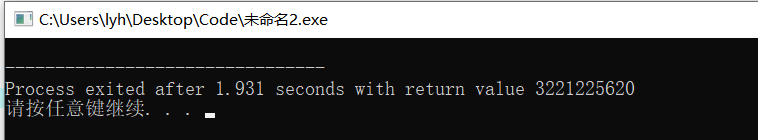
## 6.3 fib溢出验证

int 时从n=\_\_\_\_47\_\_\_\_\_时溢出，long时n=\_\_\_47\_\_\_\_\_\_\_\_\_时溢出。

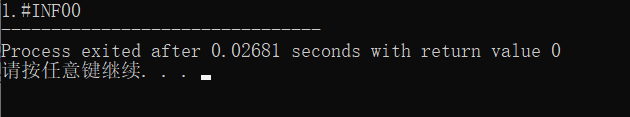
unsigned int 时从n=\_\_\_48\_\_\_\_\_\_时溢出，unsigned long时n=\_\_48\_\_\_\_\_\_时溢出。

## 6.4 除以0验证：

除以0：截图1

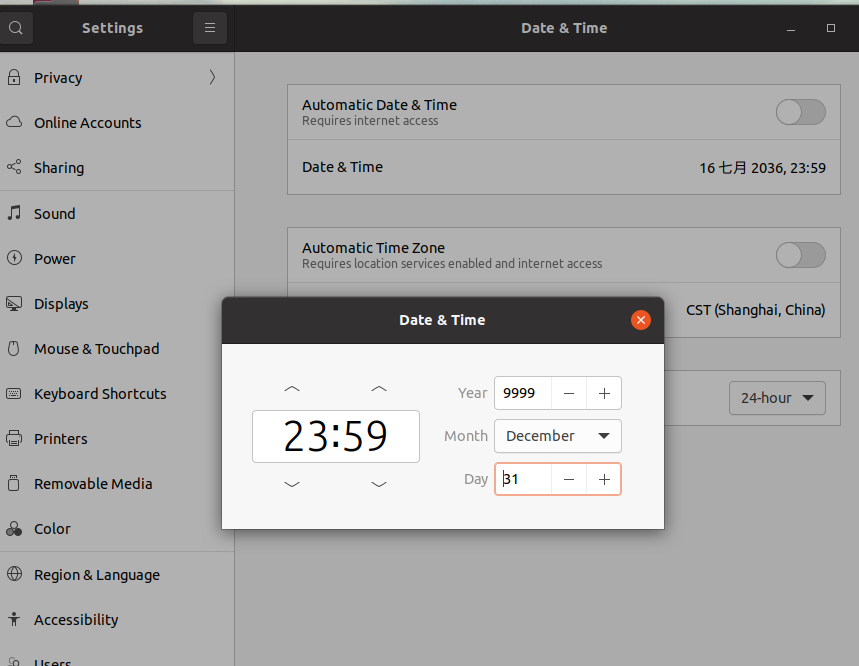
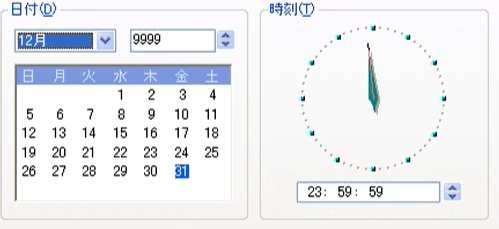


除以极小浮点数，截图：



## 6.5 万年虫验证

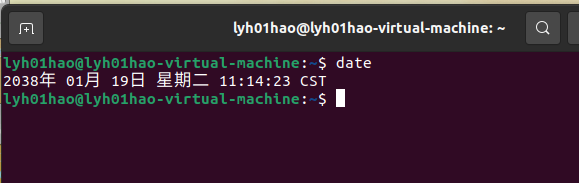
你的机器到9999年12月31日23:59:59后，时钟怎么显示的？Windows/Linux下分别截图：



windows xp在过了这个时间点之后变为10000年的1月1日，然后时间不动了，系统出了bug。

## 6.6 2038虫验证

2038年1月19日中午11:14:07后你的计算机时间是多少，Windows/Linux下分别截图



# 第7章 浮点数据的表示与运算

## 7.1手动float编码：

按步骤写出float数-10.1在内存从低到高地址的字节值（16进制）。

答：S=1;

数值：-10.1(10) =-1010.00011001100110011010(2)

尾数：M=-1.01000011001100110011010(2)

frac=01000011001100110011010

阶码：E=3

Bias=127

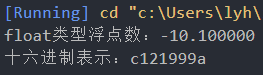
Exp=130=10000010

补码：11000001001000011001100110011010

16进制：C121999A

从低地址到高地址的数值：A99121C

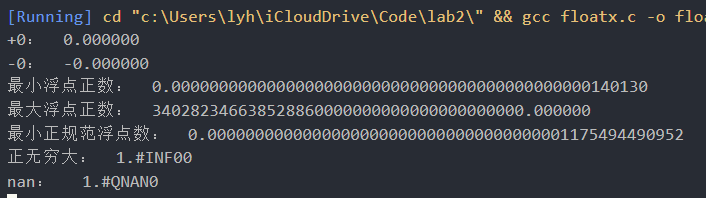
编写程序在内存验证手动编码的正确性，截图。



## 7.2特殊float数据的处理

提交子程序floatx.c，要求：

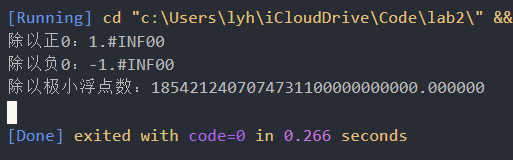
构造多float变量，分别存储+0-0，最小浮点正数，最大浮点正数、最小正的规格化浮点数、正无穷大、Nan,并打印最可能的精确结果输出（十进制/16进制）。截图。



## 7.3验证浮点运算的溢出

提交子程序float0.c

编写C程序，验证C语言中float除以0/极小浮点数后果，截图



## 7.4 类型转换的坑

实验指导PPT第5步骤的x变量，执行 x=(int)(float)x 后结果为多少？

原x=\_\_\_-1190202405\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_，现x=\_\_\_-1190202368\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 7.5 讨论1：有多少个int可以用float精确表示

有\_\_\_\_\_\_\_\_\_\_\_\_\_\_个int数据可以用float精确表示。



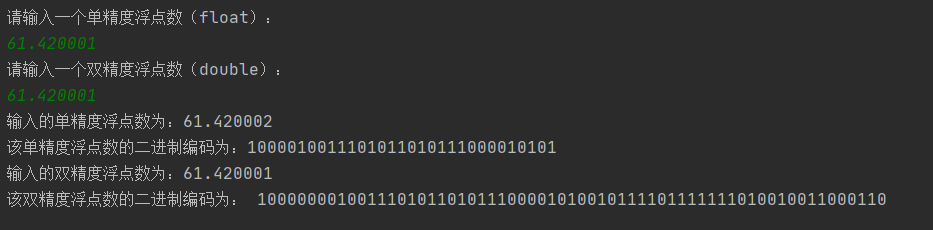
是哪些数据呢？\_\_因为浮点数尾数的位数为23，因此可以精确且连续的表示从-到的整数，当阶码的值大于等于24之后，浮点数只能离散地表示整数，尾数位共有种组合，正数的剩余阶码位有104个，负数部分有103个，合计为个\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



## 7.6 讨论2：怎么验证float采用的向偶数舍入呢

基于上个讨论，开发程序或举几个特例用C验证即可！

截图与标注说明！



可以观察到，float类型将输入的61.42001的编码截断，发生了向偶数舍入，实际输出的浮点数为：61.42002。

而double类型能更加精确地表示该数。

## 7.7 讨论3：float能精确表示几个1元内的钱呢

人民币0.01-0.99元之间的十进制数，有多少个可用float精确表示？

是哪些呢？

能够完全精确表示的只有三个：0.5，0.25，0.75其他的小数都有一定的舍入。

## 7.8 Float的微观与宏观世界

按照阶码区域写出float的最大密度区域的范围及其密度，最小密度区域及其密度（区域长度/表示的浮点个数）：\_\_ -1.1111(23个1)×2-126~1.1111(23个1)×2-126\_\_\_\_\_\_\_\_\_、\_\_\_ 225个浮点数\_\_\_\_\_\_\_\_、\_\_\_\_ 1.0000(23个0)×2127~1.11111×2127 (23个1)\_\_\_\_\_\_\_\_、\_\_\_ 223个浮点数\_\_\_\_\_

微观世界：能够区别最小的变化\_0.00000…1(2)× (22个0)



\_\_\_\_\_\_\_，其10进制科学记数法为\_\_\_1.401298464324817e-45\_\_\_\_\_\_\_

宏观世界：不能区别最大的变化\_\_1.11111×2127 (23个1)\_\_\_\_\_\_，其10进制科学记数法为\_\_\_3.4028234663852886e37\_\_\_\_\_\_\_\_

## 7.9 讨论：浮点数的比较方法

从键盘输入或运算后得到的任意两个浮点数，论述其比较方法以及理由。

比较方法：使用差值的方法，定义一个很小的接近于0的常量epsilon，设输入的两个浮点数为a与b，若fabs(a-b)<=epsilon，则可近似认为这两个浮点数相等。若fabs(a-b)>epsilon，则这两个浮点数不相等，若a-b>0.0，a比b大，若a-b<0，a比b小。

原因：浮点数并非真正意义上的实数，只是在一定范围内的近似数，因此难以判断两个浮点数是否相等，只能采取近似的手段，判断这两个浮点数的差值是否近似为0。

# 第8章 舍尾平衡的讨论

## 8.1 描述可能出现的问题

对于一般的舍尾处理，往往会对原始数据采取四舍五入的近似方法，例如：将0.5近似为1，四舍五入会产生误差，而当多组数据都采用四舍五入的方法的时候，产生的误差就会累计，例如将0.5近似为1，和式0.5+0.5=1就会变成1+1=2，使得结果与真实值存在着较大的偏差，以致于出现严重的后果。这种后果在经济统计、数据报表等方面的影响是很严重的。

## 8.2 给出完美的解决方案

定义：

1. 设平衡差=求和后的舍尾值-舍尾后的求和值；
2. 最小调整值：舍尾后的最小精度的单位值，例如在对分数取整中，最小精度就是个位，最小调整值就是±1。

方案：

如果舍尾后合计值变小，则需要将数据的值扩大，那么最小调整值就是1；如果舍位后合计值变大，则需要将数据减小，最小调整值就是-1。而调整只针对绝对值比较大的数据，这样它们的相对偏差就会比较小。用平衡差的绝对值除以最小调整值得到需要调整数据的个数n，然后对原始数据的绝对值进行排序，从大到小的n个数加上最小调整值或减去最小调整值（取决于舍位后求和值是变大还是变小），最后将调整后的值和未调整的值重新求和。

举例说明：

假设有四个数2.8、5.2、8.3、2.4。取整到个位，求和后再舍尾，值为19。舍尾后再求值，值为18，则平衡差为1,。对四个数舍尾后绝对值从大到小进行排序，为8、5、3、2，对第一个值加上最小调整值1，得9、5、3、2，此时相加的和为19，达到了舍尾平衡的目的。

# 第9章 总结

## 9.1 请总结本次实验的收获

本次实验我学到了数据如何在计算机中存储，了解了各种变量在不同的操作系统与不同位数机器的大小，了解了反汇编代码，学会使用gdb、gcc、objdump，了解了字符的编码方式，对于IEEE 754标准的浮点数有了更加深刻的了解。

## 9.2 请给出对本次实验内容的建议

关于万年虫实验，改完时候后电脑直接崩溃了，并且难以更改到9999年的时间，希望能调整一下该实验部分。

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.