

哈尔滨工业大学

实验报告

实 验（二）

题 目 DataLab 数据表示

专 业 计算机系

学 号 1180300811

班 级 1803008

学 生 孙骁

指 导 教 师 吴锐

实 验 地 点 G712

实 验 日 期 2019-09-28

计算机科学与技术学院

目 录

| | |
|-------------------------------------|---------------|
| 第 1 章 实验基本信息 | - 4 - |
| 1.1 实验目的..... | - 4 - |
| 1.2 实验环境与工具..... | - 4 - |
| 1.2.1 硬件环境..... | - 4 - |
| 1.2.2 软件环境..... | - 4 - |
| 1.2.3 开发工具..... | - 4 - |
| 1.3 实验预习..... | - 4 - |
| 第 2 章 实验环境建立 | - 6 - |
| 2.1 UBUNTU 下 CODEBLOCKS 安装..... | - 6 - |
| 2.2 64 位 UBUNTU 下 32 位运行环境建立..... | - 6 - |
| 第 3 章 C 语言的数据类型与存储 | - 8 - |
| 3.1 类型本质（1 分） | - 8 - |
| 3.2 数据的位置-地址（2 分） | - 8 - |
| 3.3 MAIN 的参数分析（2 分） | - 10 - |
| 3.4 指针与字符串的区别（2 分） | - 11 - |
| 第 4 章 深入分析 UTF-8 编码..... | - 13 - |
| 4.1 提交 UTF8LEN.C 子程序..... | - 13 - |
| 4.2 C 语言的 STRCMP 函数分析 | - 13 - |
| 4.3 讨论：按照姓氏笔画排序的方法实现 | - 13 - |
| 第 5 章 数据变换与输入输出 | - 14 - |
| 5.1 提交 CS_ATOI.C | - 14 - |
| 5.2 提交 CS_ATOF.C | - 14 - |
| 5.3 提交 CS_ITOA.C | - 14 - |
| 5.4 提交 CS_FTOA.C | - 14 - |
| 5.5 讨论分析 OS 的函数对输入输出的数据有类型要求吗 | - 14 - |
| 第 6 章 整数表示与运算 | - 15 - |
| 6.1 提交 FIB_DG.C..... | - 15 - |
| 6.2 提交 FIB_LOOP.C | - 15 - |
| 6.3 FIB 溢出验证..... | - 15 - |
| 6.4 除以 0 验证： | - 15 - |
| 第 7 章 浮点数据的表示与运算 | - 16 - |
| 7.1 正数表示范围..... | - 16 - |
| 7.2 浮点数的编码计算 | - 16 - |

| | |
|----------------------------|---------------|
| 7.3 特殊浮点数值编码 | - 17 - |
| 7.4 浮点数除 0 | - 17 - |
| 7.5 FLOAT 的微观与宏观世界 | - 17 - |
| 7.6 讨论：任意两个浮点数的大小比较 | - 17 - |
| 第 8 章 舍尾平衡的讨论 | - 19 - |
| 8.1 描述可能出现的问题 | - 19 - |
| 8.2 给出完美的解决方案 | - 19 - |
| 第 9 章 总结 | - 20 - |
| 9.1 请总结本次实验的收获 | - 20 - |
| 9.2 请给出对本次实验内容的建议 | - 20 - |
| 参考文献 | - 21 - |

第 1 章 实验基本信息

1.1 实验目的

1. 熟练掌握计算机系统的数据表示与数据运算
2. 通过 C 程序深入理解计算机运算器的底层实现与优化
3. 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

i7-8550U X64 CPU; 1.80GHz; 2G RAM; 256GHD Disk

1.2.2 软件环境

Windows10 64 位; Vmware 15.1.0; Ubuntu 18.04 LTS

1.2.3 开发工具

Visual Studio 2019 ; CodeBlocks; gcc

1.3 实验预习

1. 上实验课前，必须认真预习实验指导书（PPT 或 PDF）
2. 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。
3. 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
 - a) Char /short int/int/long/float/double/long long/long double/指针实验内容中有展示
4. 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

a) 先用递归程序实现，会出现什么问题？

b) 再用循环方式实现。

实验中有展示

5. 写出 float/double 类型最小的正数、最大的正数（非无穷）

float 最小正数 0 0000 0000 000 0000 0000 0000 0000 0001

float 最大正数 0 1111 1111 111 1111 1111 1111 1111 1111

double 最小正数 0 000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001

double 最大正数 0 111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111

6. 按步骤写出 float 数-1.1 在内存从低到高地址的字节值-16 进制

1.1 的二进制表示为 $1.0001100110011 \times 10^0$

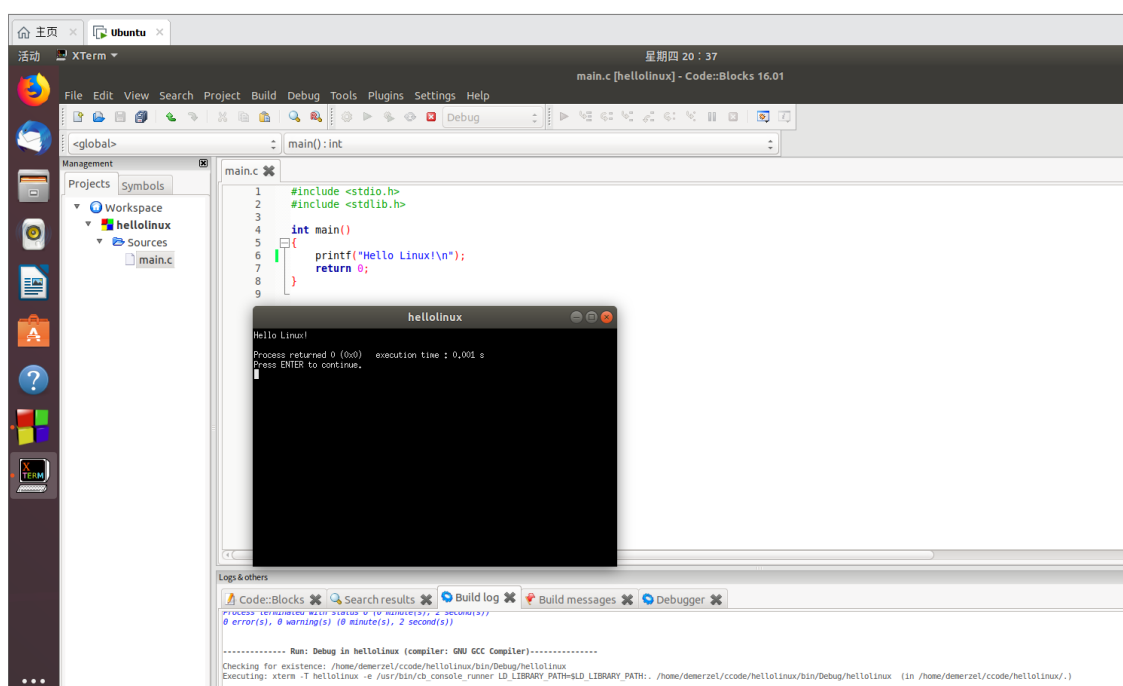
1 0111 1111 000 1100 1100 1100 1100 1101

7. 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（区域长度/表示的浮点个数）

第 2 章 实验环境建立

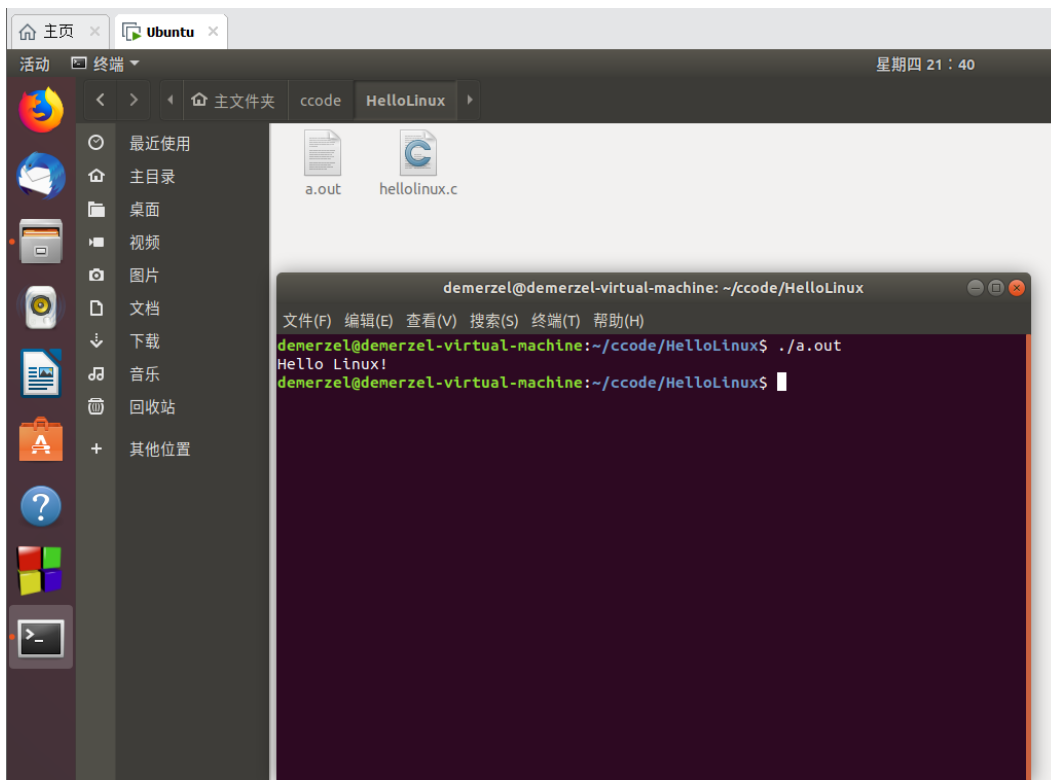
2.1 Ubuntu 下 CodeBlocks 安装

CodeBlocks 运行界面截图：编译、运行 hellolinux.c



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。



第 3 章 C 语言的数据类型与存储

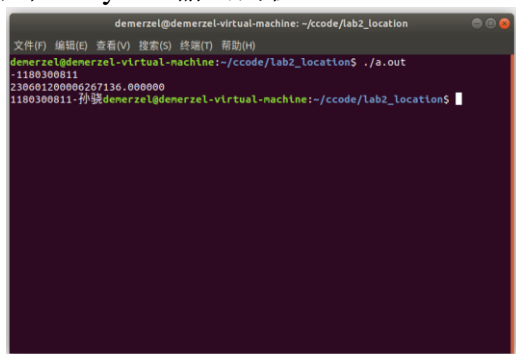
3.1 类型本质 (1 分)

| | 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 | 4 | 4 | 8 |
| long long | 8 | 8 | 4 | 4 | 8 | 8 |
| float | 4 | 4 | 8 | 8 | 4 | 4 |
| double | 8 | 8 | 8 | 8 | 8 | 8 |
| long double | 8 | 8 | 12 | 16 | 12 | 16 |
| 指针 | 4 | 8 | 4 | 4 | 4 | 8 |

C 编译器对 sizeof 的实现方式：sizeof 实际上不是函数，是一个宏，在编译时会做预处理。

3.2 数据的位置-地址 (2 分)

打印 x、y、z 输出的值：



```

demerzel@demerzel-virtual-machine: ~/code/lab2_location
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/code/lab2_location$ ./a.out
-1180300811
2386601200006267136.000000
1180300811- 孙斌demerzel@demerzel-virtual-machine:~/code/lab2_location$

```

反汇编查看 x、y、z 的地址，每字节的内容：


```

demerzel@demerzel-virtual-machine: ~/cocode/lab2/lab2_location
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_location$ gcc main.c
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_location$ ./a.out
-1180300811
230601200006267136.000000
1180300811-孙晓
x_address=0x55ee0741c010,y_address=0x7ffc20735320,z_address=0x55ee0721b810
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_location$

```

```

demerzel@demerzel-virtual-machine: ~/cocode/lab2/lab2_location
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
13      }
(gdb) l
Line number 14 out of range; main.c has 13 lines.
(gdb) x/35i main
0x6aa <main>:      push    %rbp
0x6ab <main+1>:     mov     %rsp,%rbp
0x6ae <main+4>:     sub     $0x20,%rsp
0x6b2 <main+8>:     mov     %fs:0x28,%rax
0x6bb <main+17>:    mov     %rax,-0x8(%rbp)
0x6bf <main+21>:    xor     %eax,%eax
0x6c1 <main+23>:    movsd   0x15f(%rip),%xmm0      # 0x828  y的初始地址
0x6c9 <main+31>:    movsd   %xmm0,-0x10(%rbp)
0x6ce <main+36>:    mov     -0x10(%rbp),%rcx
0x6d2 <main+40>:    mov     0x200938(%rip),%eax    # 0x201010 <x>  x的地址
0x6d8 <main+46>:    lea     0x131(%rip),%rdx      # 0x810 <z.2765>
0x6df <main+53>:    mov     %rcx,-0x18(%rbp)
0x6e3 <main+57>:    movsd   -0x18(%rbp),%xmm0
0x6e8 <main+62>:    mov     %eax,%esi
0x6ea <main+64>:    lea     0xdf(%rip),%rdi      # 0x7d0
0x6f1 <main+71>:    mov     $0x1,%eax
0x6f6 <main+76>:    callq   0x580 <printf@plt>
0x6fb <main+81>:    lea     -0x10(%rbp),%rax
0x6ff <main+85>:    lea     0x10a(%rip),%rcx      # 0x810 <z.2765>
0x706 <main+92>:    mov     %rax,%rdx
0x709 <main+95>:    lea     0x200900(%rip),%rsi    # 0x201010 <x>
0x710 <main+102>:   lea     0xc9(%rip),%rdi      # 0x7e0
0x717 <main+109>:   mov     $0x0,%eax
0x71c <main+114>:   callq   0x580 <printf@plt>
0x721 <main+119>:   mov     $0x0,%eax
0x726 <main+124>:   mov     -0x8(%rbp),%rdx
0x72a <main+128>:   xor     %fs:0x28,%rdx
0x733 <main+137>:   je      0x73a <main+144>
0x735 <main+139>:   callq   0x570 <__stack_chk_fail@plt>
0x73a <main+144>:   leaveq
0x73b <main+145>:   retq

```

反汇编查看 x、y、z 在代码段的表示形式。

```

6a5:  e9 66 ff ff ff      jmpq 610 <register_tm_clones>

00000000000006aa <main>:
6aa:  55                  push %rbp
6ab:  48 89 e5            mov %rsp,%rbp
6ae:  48 83 ec 20         sub $0x20,%rsp
6b2:  64 48 8b 04 25 28 00 mov %fs:0x28,%rax
6b9:  00 00
6bb:  48 89 45 f8         mov %rax,-0x8(%rbp)
6bf:  31 c0              xor %eax,%eax
6c1:  f2 0f 10 05 5f 01 00 movsd 0x15f(%rip),%xmm0      # 828 <z.2765+0x18>
6c8:  00
6c9:  f2 0f 11 45 f0         movsd %xmm0,-0x10(%rbp)
6ce:  48 8b 4d f0         mov -0x10(%rbp),%rcx
6d2:  8b 05 38 09 20 00     mov 0x200938(%rip),%eax      # 201010 <x>
6d8:  48 8d 15 31 01 00 00     lea 0x131(%rip),%rdx      # 810 <z.2765>
6df:  48 89 4d e8         mov %rcx,-0x18(%rbp)
6e3:  f2 0f 10 45 e8         movsd -0x18(%rbp),%xmm0
6e8:  89 c6              mov %eax,%esi
6ea:  48 8d 3d df 00 00 00     lea 0xdf(%rip),%rdi      # 7d0 <_IO_stdin_used+0x10>
6f1:  b8 01 00 00 00         mov $0x1,%eax
6f6:  e8 85 fe ff ff        callq 580 <printf@plt>
6fb:  48 8d 45 f0         lea -0x10(%rbp),%rax
6ff:  48 8d 0d 0a 01 00 00     lea 0x10a(%rip),%rcx      # 810 <z.2765>
706:  48 89 c2              mov %rax,%rdx
709:  48 8d 35 00 09 20 00     lea 0x200900(%rip),%rsi      # 201010 <x>
710:  48 8d 3d c9 00 00 00     lea 0xc9(%rip),%rdi      # 7e0 <_IO_stdin_used+0x20>
717:  b8 00 00 00 00         mov $0x0,%eax
71c:  e8 5f fe ff ff        callq 580 <printf@plt>
721:  b8 00 00 00 00         mov $0x0,%eax
726:  48 8b 55 f8         mov -0x8(%rbp),%rdx
72a:  64 48 33 14 25 28 00     xor %fs:0x28,%rdx
731:  00 00
733:  74 05              je 73a <main+0x90>
735:  e8 36 fe ff ff        callq 570 <__stack_chk_fail@plt>
73a:  c9                leaveq
73b:  c3                retq
73c:  0f 1f 40 00         nopl 0x0(%rax)

```

x 与 y 在汇编、链接阶段转换成补码与 ieee754 编码。

数值型常量与变量在存储空间上的区别是：常量在申请地址的就写入寄存器，变量在编译的时候才会写入寄存器。

字符串常量与变量在存储空间上的区别是：常量在申请地址的就写入寄存器，变量在编译的时候才会写入寄存器。但是字符串常量无法更改。

常量表达式在计算机中处理方法是：编译时直接计算结果。

3.3 main 的参数分析 (2 分)

反汇编查看 x、y、z 的地址，argc 的地址，argv 的地址与内容。

```

demerzel@demerzel-virtual-machine: ~/ccode/lab2/lab2_main
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_main$ ./a.out -1180300811 23
0601200006267123 1180300811-孙骁
./a.out
-1180300811
230601200006267123
1180300811-孙骁
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_main$

```

打印 x, y, z.

```

打开(O)  a.S  保存(S)
~/ccode/lab2/lab2_main

Disassembly of section .data:
000000000201000 <_data_start>:
000000000201000 <_data_start>:
...
000000000201008 <_dso_handle>:
201008: 08 10          or    %dl,(%rax)
20100a: 20 00          and    %al,(%rax)
20100c: 00 00          add    %al,(%rax)
...
000000000201010 <x>:
201010: f5             cmc
201011: 09 a6 b9      or    %esp,0x4cd0a4b9(%rsi)
...
000000000201014 <y>:
201014: a4             movsb %ds:(%rsi),%es:(%rdi)
201015: d0 4c 5c 00    rorb  0x0(%rsp,%rbx,2)
201019: 00 00          add    %al,(%rax)
20101b: 00 00          add    %al,(%rax)
20101d: 00 00          add    %al,(%rax)
...
000000000201020 <z>:
201020: 31 31          xor    %esi,(%rcx)
201022: 30 30          cmp    %dh,(%rax)
201024: 33 30          xor    (%rax),%esi
201026: 30 38          xor    %bh,(%rax)
201028: 31 31          xor    %esi,(%rcx)
20102a: 2d e5 ad 99 e9 sub    $0xe999ade5,%eax
20102f: aa             stos   %al,%es:(%rdi)
201030: 81             .byte 0x81
...

Disassembly of section .bss:
000000000201032 <_bss_start>:
...

```

查看 x, y, z 对应地址

3.4 指针与字符串的区别 (2 分)

cstr 的地址与内容截图，pstr 的内容与截图，截图 5

```

Disassembly of section .data:
000000000201000 <__data_start>:
...
000000000201008 <__dso_handle>:
201008: 08 10                or    %dl, (%rax)
20100a: 20 00                and    %al, (%rax)
...
000000000201020 <cstr>:
201020: 31 31                xor    %esi, (%rcx)
201022: 38 30                cmp    %dh, (%rax)
201024: 33 30                xor    (%rax), %esi
201026: 30 38                xor    %bh, (%rax)
201028: 31 31                xor    %esi, (%rcx)
20102a: 2d e5 ad 99 e9       sub    $0xe99ade5, %eax
20102f: aa                  stos   %al, %es: (%rdi)
201030: 81 00 00 00 00 00    addl   $0x0, (%rax)
...
000000000201088 <pstr>:
201088: 44 07                rex.R (bad)
20108a: 00 00                add    %al, (%rax)
20108c: 00 00                add    %al, (%rax)
...
Disassembly of section .bss:
000000000201090 <__bss_start>:
...
Disassembly of section .comment:
0000000000000000 <.comment>:
0: 47                rex.RXB
1: 43                rex.XB
2: 43 3a 20          rex.XB cmp (%r8), %spl
5: 28 55 62          sub    %dl, 0x62(%rbp)
8: 7c 6a             inc    %rcx

```

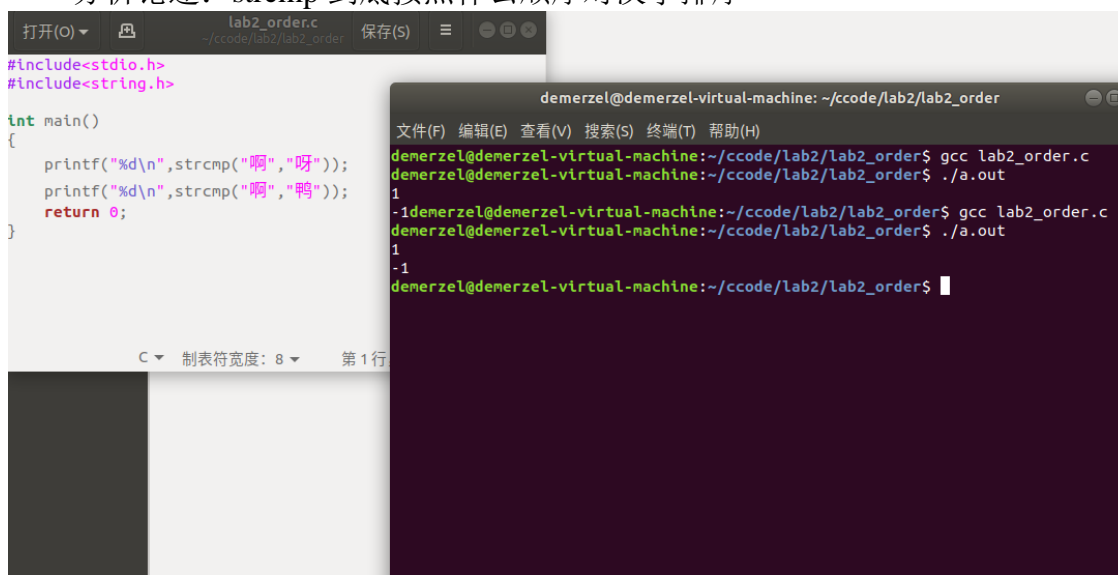
pstr 修改内容会出现什么问题 编译器报错：段错误（核心已转存），因为 pstr 的内容存储在.rodata 段，为只读，不可以更改。

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



The screenshot shows a code editor on the left and a terminal window on the right. The code editor displays the following C code:

```
#include<stdio.h>
#include<string.h>

int main()
{
    printf("%d\n",strcmp("啊","呀"));
    printf("%d\n",strcmp("啊","鸭"));
    return 0;
}
```

The terminal window shows the execution of the program:

```
demerzel@demerzel-virtual-machine: ~/cocode/lab2/lab2_order
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_order$ gcc lab2_order.c
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_order$ ./a.out
1
-1
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_order$ gcc lab2_order.c
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_order$ ./a.out
1
-1
demerzel@demerzel-virtual-machine:~/cocode/lab2/lab2_order$
```

Utf-8 并不是按照读音对汉字进行的排序,是按照 utf-8 的编码顺序对汉字排序进行比较。

4.3 讨论：按照姓氏笔画排序的方法实现

1、可以自己编写笔画序的编码对应文件,或者引用已有的编码文件对于姓名按照笔画排序。

2、构建结构体,结构体两个元素,一个是汉字的字符串,一个 int 类型的汉字笔画数,排序是比较两个 int 类型大小即可。

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：有要求，os 函数，例如 `printf`、`scanf` 是根据格式控制符的类型读取，数据根据格式控制符解释，即输入的数据类型必须依照格式控制符解释，否则读入时会有问题；同理，输出时是依照格式控制符对于数据解读。

第 6 章 整数表示与运算

6.1 提交 fib_dg.c

6.2 提交 fib_loop.c

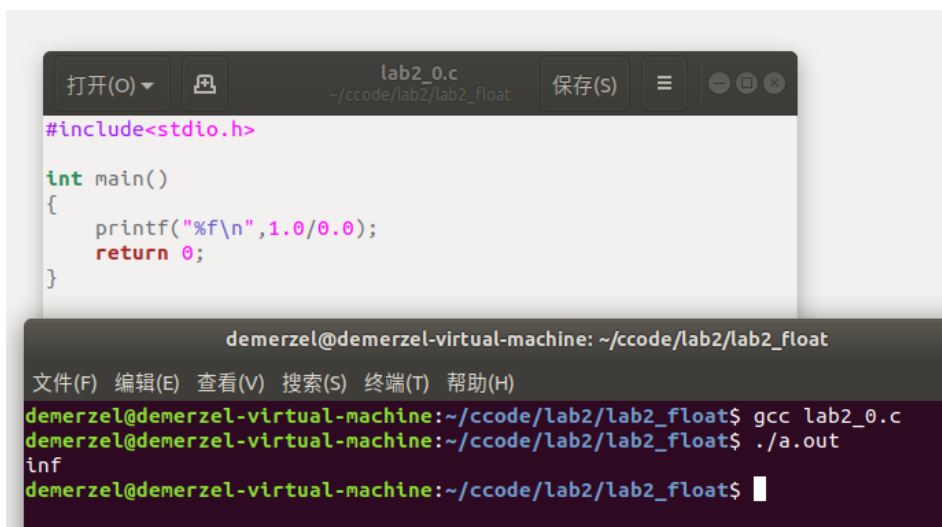
6.3 fib 溢出验证

int 时从 n=47 时溢出, long 时 n=93 时溢出。

unsigned int 时从 n=48 时溢出, unsigned long 时 n=94 时溢出。

6.4 除以 0 验证:

除以 0:



```
lab2_0.c
~/ccode/lab2/lab2_float 保存(S)

#include<stdio.h>

int main()
{
    printf("%f\n",1.0/0.0);
    return 0;
}

demerzel@demerzel-virtual-machine: ~/ccode/lab2/lab2_float
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$ gcc lab2_0.c
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$ ./a.out
inf
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$
```

除以极小浮点数

第 7 章 浮点数据的表示与运算

7.1 正数表示范围

写出 float/double 类型最小的正数、最大的正数（非无穷）

float 最小正数 0 0000 0000 000 0000 0000 0000 0001, 即 2^{-149} , 1.4E-45

float 最大正数 0 1111 1111 111 1111 1111 1111 1111, 即 3.4E38

double 最小正数 0 000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001, 即 2^{-1074} , 4.9E-324

double 最大正数 0 111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111, 即 1.8E308

7.2 浮点数的编码计算

(1) 按步骤写出 float 数 -1.1 的浮点编码计算过程, 写出该编码在内存中从低地址字节到高地址字节的 16 进制数值

1. 1.1 的二进制编码为 $1.0001100110011 \times 10^0$
2. 阶为 0, 即阶码 $E=127+0=0111\ 1111$
3. 尾数部分经过舍入是 000 1100 1100 1100 1100 1101
即 -1.1 的机器数表示为 1 0111 1111 000 1100 1100 1100 1100 1101
4. 转化为 16 进制为 0xBF8CCCCDH

(2) 验证: 编写程序, 输出值为 -1.1 的浮点变量其各内存单元的数值, 截图。

```
#include<stdio.h>

typedef unsigned char *pointer;

void show_bytes(pointer start, size_t len){
    size_t i;
    for (i=0; i<len; i++){
        printf("%.2x\t", start[i]);
    }
    printf("\n");
}

int main()
{
    float a=-1.1;
    show_bytes((pointer) &a, sizeof(float));
    return 0;
}
```

```
demerzel@demerzel-virtual-machine: ~/ccode/lab2/lab2_float
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$ gcc lab2_0.c
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$ ./a.out
cd    cc    8c    bf
demerzel@demerzel-virtual-machine:~/ccode/lab2/lab2_float$
```


1. 首先判断二者是否有 Nan，如果有，报告无法比较
2. 判断两数的正负情况，若符号位不同且两数不全为 0，则符号位为 0 的更大；若两数全为 0，则两数相等。
3. 此时两数的符号位一致，记为 a，从高位到低位比较二进制的每一个值。
若 a 为 0，则差异位为 1 的大；若 a 为 1，则差异位为 0 的更大。
4. 若两浮点数完全一致，则两数相等。

第 8 章 舍尾平衡的讨论

8.1 描述可能出现的问题

在对于报表数据的统计中，常常因为精度要求或者单位换算，对数据进行四舍五入的操作，这种操作常被称为舍尾处理。直接的舍尾处理可能会在最后的统计中带来较大的差距，原有的平衡可能被打破。

为保证数据表中数据关系的正确，需要舍尾调整数据，使得数据重新平衡，即舍尾平衡。

8.2 给出完美的解决方案

设有一组数据，用矩阵表示为 $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ ，设

$$sum_s = \sum_{i=1}^n a_{si} (s=1, 2, \cdots, m) \quad , \quad sum_q = \sum_{j=1}^m a_{jq} (q=1, 2, \cdots, n) \quad , \quad \text{称}$$

$Ps = sum_s - a_{si} (s=1, 2, \cdots, m)$ 为第 s 行 a_{si} 对此行的平衡差，

$Pq = sum_q - a_{jq} (q=1, 2, \cdots, n)$ 为第 q 列 a_{jq} 对此列的平衡差；和向列：

$a_{11}, a_{12}, \cdots, a_{1m}, a_{11}, a_{21}, \cdots, a_{n1}$ ，子项列 $a_{j1}, a_{j2}, \cdots, a_{jm}, a_{1q}, a_{2q}, \cdots, a_{nq}$ 。

则舍尾平衡条件为：

1. 两个不同方向的子项列的平衡差符号相同；
2. 两个相同方向的子项列的平衡差符号相异；
3. 和向列与不同方向的子项列的平衡差符号相异；
4. 和向列与相同方向的子项列的平衡差符号相同；
5. 两个和向列的平衡差符号相同。

以上给出的是舍尾平衡的一种方法，时间复杂度还有待优化。

第 9 章 总结

9.1 请总结本次实验的收获

通过使用 gdb 对于反汇编有了更深入的了解
对代码及数据在计算机内部的存储有了更深入地认识

9.2 请给出对本次实验内容的建议

希望以后实验的指导书（ppt）更详细一些，具体指出实验的要求，

注：本章为酌情加分项。

参考文献

- [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] 湛颖. 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>.