

哈尔滨工业大学

实验报告

实 验（一）

题 目 计算机系统漫游

专 业 计算机类

学 号 1170500913

班 级 1703002

学 生 熊健羽

指 导 教 师 史先俊

实 验 地 点 G721

实 验 日 期 2018 年 9 月 11 日

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	- 10 -
2.1 VMWARE 下中文 UBUNTU 安装（5 分）	- 10 -
2.2 UBUNTU 与 WINDOWS 目录共享（5 分）	- 11 -
第 3 章 WINDOWS 软硬件系统观察分析	- 13 -
3.1 查看计算机基本信息（2 分）	- 13 -
3.2 设备管理器查看（2 分）	- 14 -
3 隐藏分区与虚拟内存之分页文件查看（2 分）	- 14 -
3.4 任务管理与资源监视（2 分）	- 15 -
3.5 计算机硬件详细信息（2 分）	- 15 -
第 4 章 LINUX 软硬件系统观察分析	- 16 -
4.1 计算机硬件详细信息（3 分）	- 16 -
4.2 任务管理与资源监视（2 分）	- 17 -
4.3 共享目录的文件系统信息（3 分）	- 17 -
4.4 LINUX 下网络系统信息（2 分）	- 18 -
第 5 章 以 16 进制形式查看程序 HELLO.C	- 19 -
5.1 请查看 HELLOWIN.C 与 HELLOLINUX.C 的编码（3 分）	- 19 -
5.2 请查看 HELLOWIN.C 与 HELLOLINUX.C 的回车（3 分）	- 20 -
第 6 章 程序的生成 CPP、GCC、AS、LD	- 21 -
6.1 请提交每步生成的文件（4 分）	- 21 -
第 7 章 计算机系统的基本信息获取编程	- 22 -
7.1 请提交源程序文件（10 分）	- 22 -
第 8 章 计算机数据类型的本质	- 23 -
8.1 请提交源程序文件 DATATYPE.C（10 分）	- 23 -
第 9 章 程序运行分析	- 24 -

9.1 SUM 的分析（20 分）	- 24 -
9.2 FLOAT 的分析（20 分）	- 25 -
第 10 章 总结.....	- 27 -
10.1 请总结本次实验的收获.....	- 27 -
10.2 请给出对本次实验内容的建议.....	- 27 -
参考文献.....	- 28 -

第 1 章 实验基本信息

1.1 实验目的

运用现代工具进行计算机软硬件系统的观察与分析；
运用现代工具进行 Linux 下 C 语言的编程调试；
初步掌握计算机系统的基本知识与各种类型的数据表示。

1.2 实验环境与工具

1.2.1 硬件环境

CPU: Intel(R) Core(TM) i5-7200U @ 2.50GHz (64 位)

GPU: Intel(R) HD Graphics 620

Nvidia GeForce 940MX

物理内存: 8.00GB

磁盘: 1TB HDD

128GB SSD

1.2.2 软件环境

Windows10 64 位;

Vmware 14.11;

Ubuntu 18.04 64 位;

1.2.3 开发工具

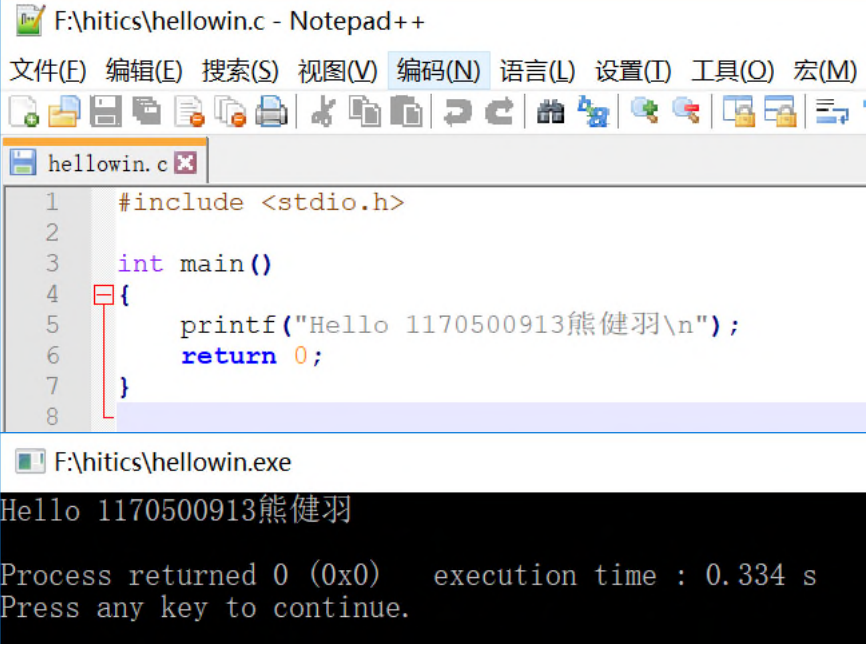
Visual Studio 2010 64 位;

Code::Blocks;

gedit, gcc, notepad++;

1.3 实验预习

1.windows 下编写 hellowin.c:

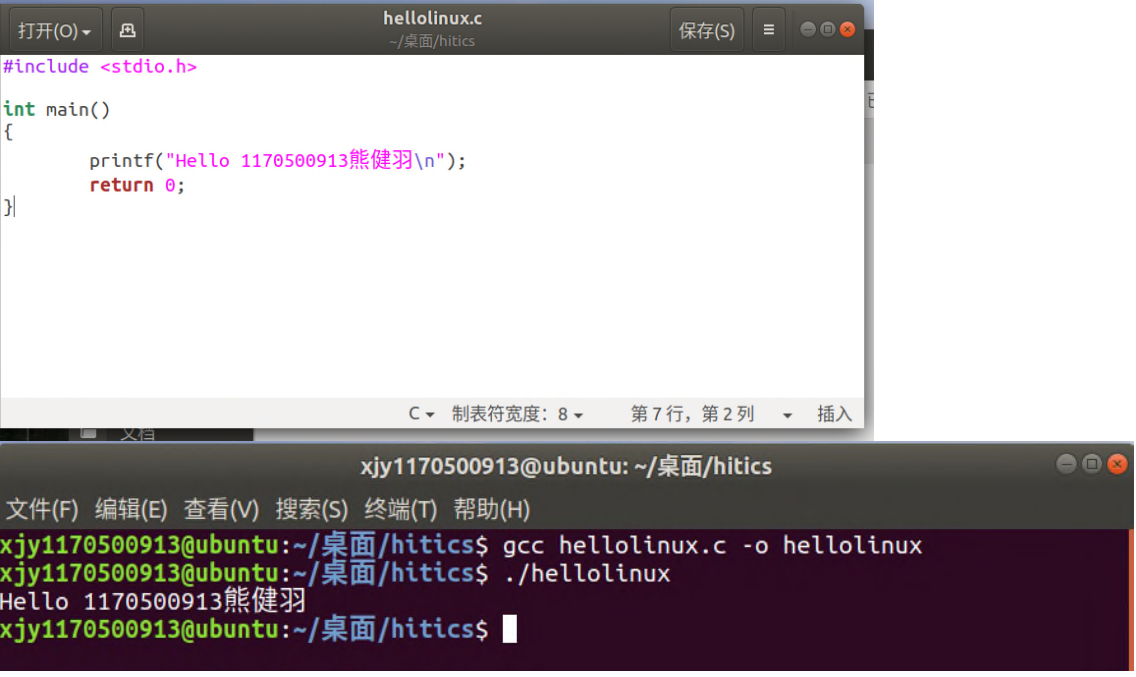


The screenshot shows a Notepad++ window titled "F:\hitics\hellowin.c - Notepad++". The menu bar includes "文件(E)", "编辑(E)", "搜索(S)", "视图(V)", "编码(N)", "语言(L)", "设置(I)", "工具(O)", and "宏(M)". The toolbar contains various icons for file operations. The code editor shows the following C code:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello 1170500913熊健羽\n");
6      return 0;
7  }
8
```

Below the code editor, the command prompt shows the execution of "F:\hitics\hellowin.exe". The output is "Hello 1170500913熊健羽". The process returned 0 (0x0) and the execution time was 0.334 s. The prompt asks to "Press any key to continue."

2.linux 下编写 hellolinux.c:



The screenshot shows a Linux terminal window titled "xjy1170500913@ubuntu: ~/桌面/hitics". The terminal displays the following commands and output:

```
#include <stdio.h>

int main()
{
    printf("Hello 1170500913熊健羽\n");
    return 0;
}

xjy1170500913@ubuntu:~/桌面/hitics$ gcc hellolinux.c -o hellolinux
xjy1170500913@ubuntu:~/桌面/hitics$ ./hellolinux
Hello 1170500913熊健羽
xjy1170500913@ubuntu:~/桌面/hitics$
```

3.

showbyte.c 代码如下：（由 syntax-highlight-word 生成代码高亮）

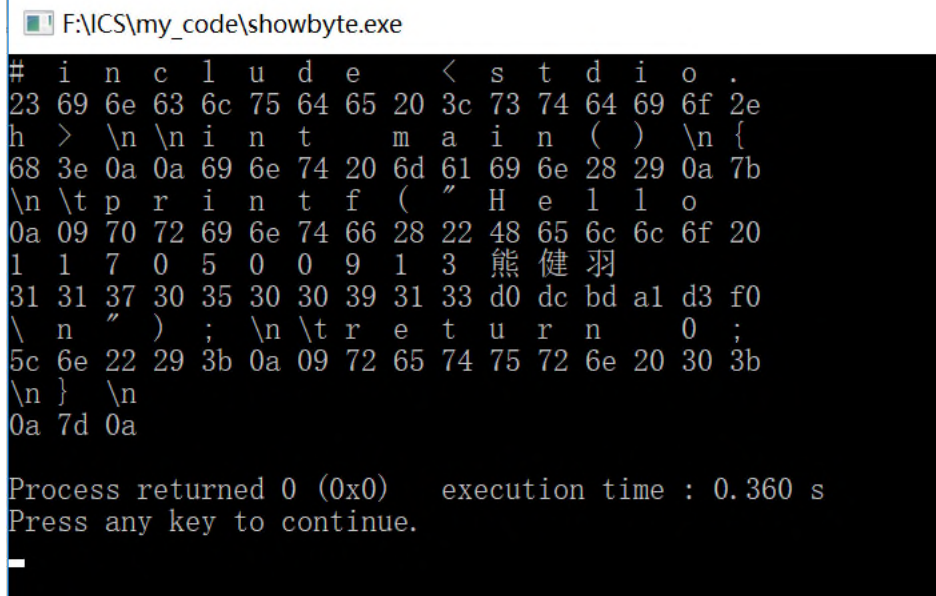
```
1.  /* showbyte.c
2.     文件指针读取文件 hellowin.c
3.     */
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  /*函数声明*/
7.  void ShowLine(char *s, int len);
8.
9.  int main()
10. {
11.     FILE *fp1;
12.     char s[60];
13.     char ch;
14.     int counter = 0;
15.     if((fp1 = fopen(".\\hellowin.c","r")) == NULL)
16.         //打开当前目录的 hellowin.c 文件
17.     {
18.         printf("failure to open!\n");
19.         exit(0);
20.     }
21.     while((ch = fgetc(fp1)) != EOF)
22.     {
23.         s[counter % 16] = ch;           //s 数组储存一行的字符串
24.         counter++;                     //counter 计数，控制换行
25.         if(counter % 16 == 0)          //一行完整 16 个字符
26.             ShowLine(s, 16);
27.     }
28.     if(counter % 16 != 0)              //若最后一行不满 16 个字符
29.         ShowLine(s, counter % 16);
30.     fclose(fp1);
31.     //getchar();
32.     return 0;
33. }
34.
35. void ShowLine(char *s, int len)        //输出一行（len 个）的字符和十六进制
36. {
37.     int i;
38.     for(i = 0; i < len; i++)
39.     {
40.
41.         if(s[i] == '\n')              //考虑制表符、回车符、换行符
42.             printf("\\n ");
43.         else if(s[i] == '\r')
44.             printf("\\n ");
45.         else if(s[i] == '\t')
46.             printf("\\t ");
47.
48.         else if(s[i] < 0)              //收到汉字字符
49.         {
50.             printf("%c%c ",s[i],s[i + 1]); //ASCII 中一个汉字为 2 个字节
51.             i++;
52.         }
53.         else                          //普通字符
54.             printf("%c ",s[i]);
55.     }
56.     printf("\\n");
```

```

57.     for(i = 0; i < len; i++)                //输出十六进制
58.         printf("%02x ",s[i] & 0xff);        //只取低 8 位
59.     printf("\n");
60. }

```

showbyte.c 编译运行结果截图如下:



```

F:\ICS\my_code\showbyte.exe
# i n c l u d e < s t d i o .
23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e
h > \n \n i n t m a i n ( ) \n {
68 3e 0a 0a 69 6e 74 20 6d 61 69 6e 28 29 0a 7b
\n \t p r i n t f ( " H e l l o
0a 09 70 72 69 6e 74 66 28 22 48 65 6c 6c 6f 20
l l 7 0 5 0 0 9 1 3 熊 健 羽
31 31 37 30 35 30 30 39 31 33 d0 dc bd a1 d3 f0
\n " ) ; \n \t r e t u r n 0 ;
5c 6e 22 29 3b 0a 09 72 65 74 75 72 6e 20 30 3b
\n } \n
0a 7d 0a

Process returned 0 (0x0)   execution time : 0.360 s
Press any key to continue.

```

4.

datatype.c 代码如下

```

1.  /*datatype.c*/
2.
3.  #include <stdio.h>
4.
5.  typedef unsigned char* byte_p;
6.
7.  int main();
8.  void showByte(byte_p p, int len);
9.
10. char ch = 'a';
11. int num = 1170500913;
12. int *p = #
13. int array[] = {0, 1, 2};
14. long lnum = 1170500913;
15. long long llnum = 1170500913;
16. float f = 1999.0;
17. double lf = 19991217.0;
18.
19. enum color
20. {
21.     red,
22.     green,
23.     yellow,
24.     blue
25. } sky = blue;
26.

```

```

27. union var
28. {
29.     int num;
30.     char ch;
31. } v = {66};
32. struct student
33. {
34.     int num;
35.     float score;
36.     char level;
37. } stu1 = {1, 98.5, 'A'};
38.
39. int (*p_printf)(const char *, ...) = printf;
40. int (*p_main)() = main;
41.
42. int main()
43. {
44.     printf("数据类型\t变量名\t变量内容\t变量地址\t变量对应的 16 进制内存字节\n");
45.     printf("DataType\tDataName\tContent\t\tAddress\t\tContent_HEX(Little_Endian)\n");
46.     printf("-----\n");
47.
48.     printf("char\t\tch\t\t%c\t\t0x%x\t", ch, (byte_p)&ch); //char
49.     showByte((byte_p)&ch, sizeof(char));
50.     printf("int\t\ttnum\t\t%d\t\t0x%x\t", num, (byte_p)&num); //int
51.     showByte((byte_p)&num, sizeof(int));
52.     printf("int*\t\ttp\t\t0x%x\t0x%x\t", p, (byte_p)&p); //int*
53.     showByte((byte_p)&p, sizeof(int*));
54.     printf("int[]\t\ttarray\t\tarray[0] = %d\t0x%x\t", array[0], (byte_p)array);
55.     //int[]
56.     showByte((byte_p)array, sizeof(int));
57.     printf("\t\t\t\tarray[1] = %d\t0x%x\t", array[1], (byte_p)(array + 1));
58.     showByte((byte_p)(array + 1), sizeof(int));
59.     printf("\t\t\t\tarray[2] = %d\t0x%x\t", array[2], (byte_p)(array + 2));
60.     showByte((byte_p)(array + 2), sizeof(int));
61.     printf("long\t\ttlnum\t\t%ld\t\t0x%x\t", lnum, (byte_p)&lnum); //long
62.     showByte((byte_p)&lnum, sizeof(long));
63.     printf("long long\t\ttllnum\t\t%I64d\t\t0x%x\t", llnum, (byte_p)&llnum); //long long
64.     showByte((byte_p)&llnum, sizeof(long long));
65.     printf("float\t\ttf\t\t%.2f\t\t0x%x\t", f, (byte_p)&f); //float
66.     showByte((byte_p)&f, sizeof(float));
67.     printf("double\t\ttlf\t\t%.2lf\t\t0x%x\t", lf, (byte_p)&lf); //double
68.     showByte((byte_p)&lf, sizeof(double));
69.     printf("enum color\t\ttsky\t\t%d\t\t0x%x\t", sky, (byte_p)&sky); //enum
70.     showByte((byte_p)&sky, sizeof(enum color));
71.     printf("union var\t\ttv\t\tv.num = %d\t0x%x\t", v.num, (byte_p)&(v.num)); //union
72.     showByte((byte_p)&(v.num), sizeof(v.num));
73.     printf("\t\t\t\ttv.ch = %c\t0x%x\t", v.ch, (byte_p)&(v.ch));
74.     showByte((byte_p)&(v.ch), sizeof(v.ch));
75.     printf("struct student\t\tstu1\t\tstu1.num = %d\t0x%x\t", stu1.num, (byte_p)&(stu1.num)); //struct
76.     showByte((byte_p)&(stu1.num), sizeof(stu1.num));
77.     printf("\t\t\t\tstu1.score=%.1f\t0x%x\t", stu1.score, (byte_p)&(stu1.score));
78.     showByte((byte_p)&(stu1.score), sizeof(stu1.score));
79.     printf("\t\t\t\tstu1.level = %c\t0x%x\t", stu1.level, (byte_p)&(stu1.level));
80.     showByte((byte_p)&(stu1.level), sizeof(stu1.level));
81. }

```



```

79. showByte((byte_p)&(stu1.level), sizeof(stu1.level));
80. printf("FunctionPointer\tp_main\t\t0x%x\t0x%x\t", p_main, (byte_p)&p_main)
    ; //main
81. showByte((byte_p)&p_main, sizeof(p_main));
82. printf("FunctionPointer\tp_printf\t0x%x\t0x%x\t", p_printf, (byte_p)&p_pri
ntf); //printf
83. showByte((byte_p)&p_printf, sizeof(p_printf));
84. printf("\n 注: p_main 与 p_printf 的内容即为 main 函数与 printf 函数的地址
    \n"); //tip
85. }
86.
87. void showByte(byte_p p, int len) //用于打印变量对应的 16 进制内存字节
88. {
89.     int i;
90.     printf("0x");
91.     for(i = 0; i < len; i++)
92.     {
93.         printf("%02x ", *p);
94.         p++;
95.     }
96.     printf("\n");
97. }

```

datatype.c 编译运行结果如下:

F:\ICS\my_code\datatype.exe

数据类型 DataType	变量名 DataName	变量内容 Content	变量地址 Address	变量对应的16进制内存字节 Content_HEX(Little_Endian)
char	ch	a	0x404010	0x61
int	num	1170500913	0x404014	0x31 6d c4 45
int*	p	0x404014	0x404018	0x14 40 40 00 00 00 00 00
int[]	array	array[0] = 0	0x404020	0x00 00 00 00
		array[1] = 1	0x404024	0x01 00 00 00
		array[2] = 2	0x404028	0x02 00 00 00
long	lnum	1170500913	0x40402c	0x31 6d c4 45
long long	llnum	1170500913	0x404030	0x31 6d c4 45 00 00 00 00
float	f	1999.00	0x404038	0x00 e0 f9 44
double	lf	19991217.00	0x404040	0x00 00 00 10 ab 10 73 41
enum color	sky	3	0x404048	0x03 00 00 00
union var	v	v.num = 66	0x40404c	0x42 00 00 00
		v.ch = B	0x40404c	0x42
struct student	stu1	stu1.num = 1	0x404050	0x01 00 00 00
		stu1.score=98.5	0x404054	0x00 00 c5 42
		stu1.level = A	0x404058	0x41
FunctionPointer	p_main	0x401560	0x404068	0x60 15 40 00 00 00 00 00
FunctionPointer	p_printf	0x402e70	0x404060	0x70 2e 40 00 00 00 00 00

注: p_main与p_printf的内容即为main函数与printf函数的地址

Process returned 0 (0x0) execution time : 0.298 s
Press any key to continue.

第 2 章 实验环境建立

2.1 Vmware 下中文 Ubuntu 安装（5 分）

安装 Ubuntu，安装中文输入法（搜狗输入法），用户名为学号！

打开终端 term，输入 Hello 1160300199 学霸（用真实学号姓名代替）。

截图：要求有 Windows 状态行，Vmware 窗口，Ubuntu 窗口，终端 term 窗口，输入的“Hello 1160300199 学霸”信息

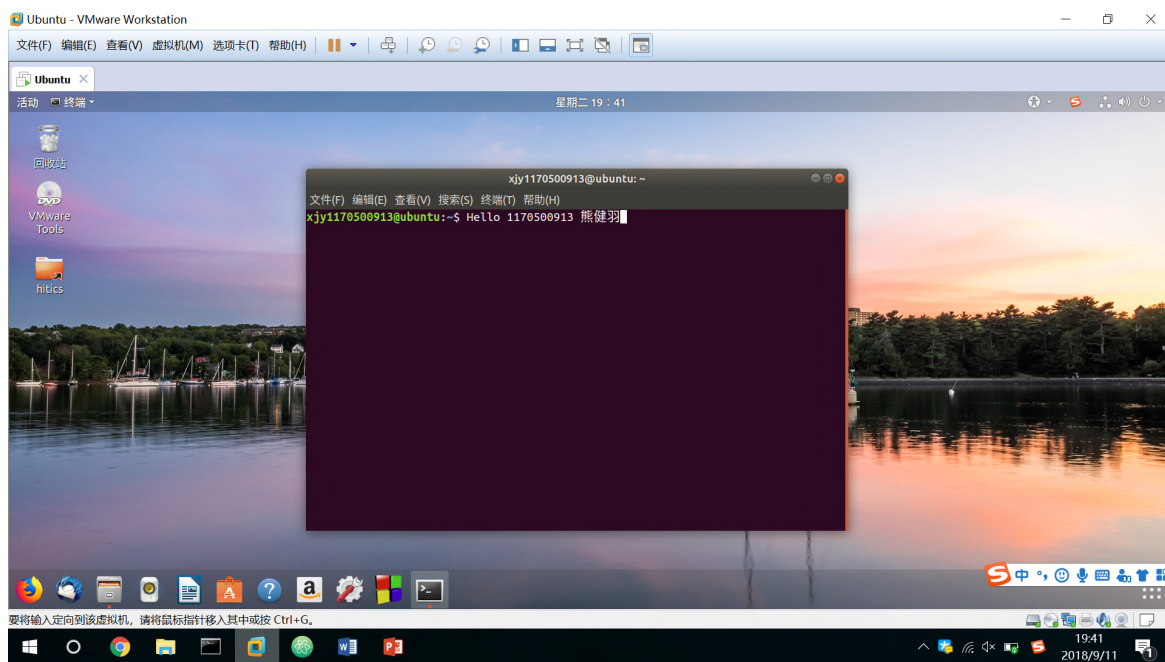


图 2-1 Vmware 下中文 Ubuntu 安装效果截图

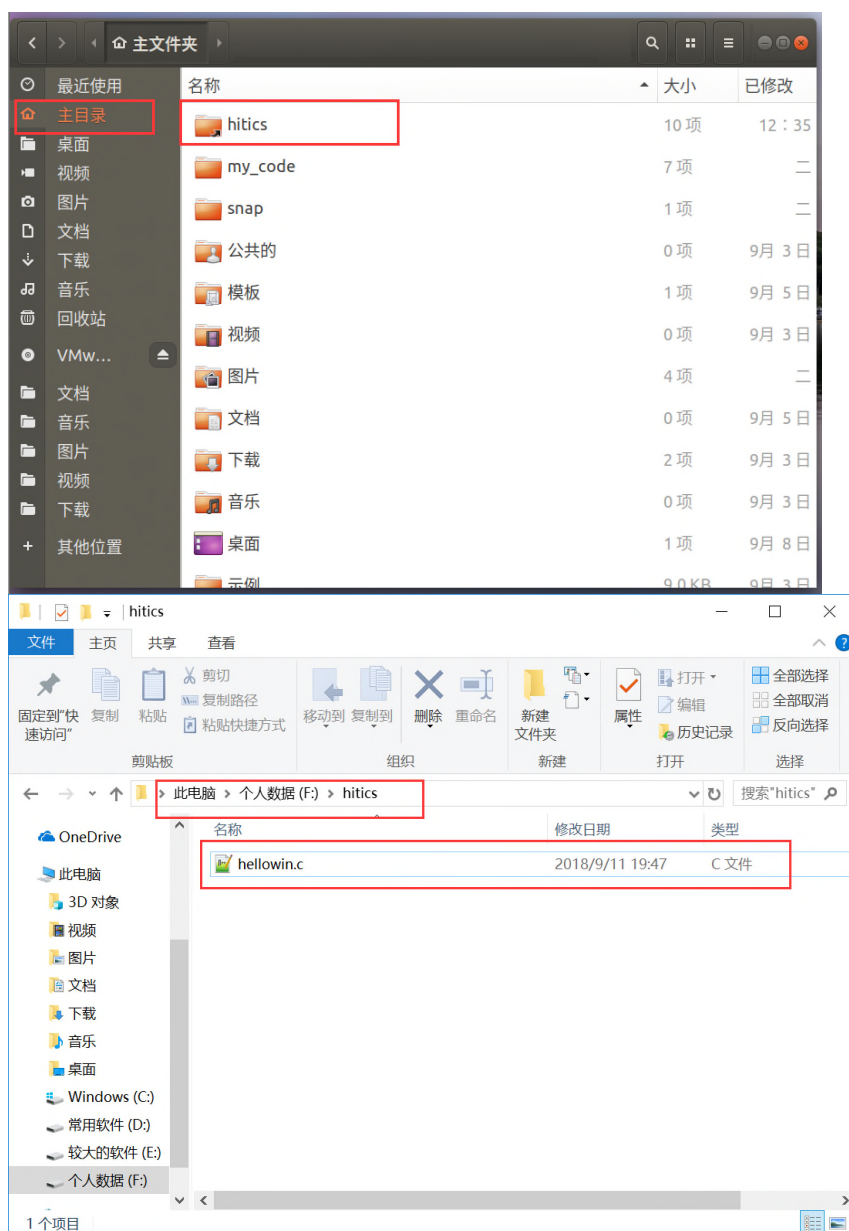
2.2 Ubuntu 与 Windows 目录共享 (5 分)

在 Windows 下建立一目录，将 helloworld.c 拷贝到此目录。在 vmware 下设置 Ubuntu 共享 hitics。

在 Ubuntu 下 Home/用户名中 建立一快捷链接 hitics 指向此共享目录,并在此目录建立 helloworld.c。

打开终端 term，进入此目录，输入 “ls -la” 指令。

截图：要求有 Ubuntu 的“文件”应用打开“Home/用户名”，能看到 hitics。term 窗口。



```
xjy1170500913@ubuntu: ~/hitics
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
xjy1170500913@ubuntu:~/hitics$ ls -la
总用量 5
drwxrwxrwx 1 root root  0 9月 11 19:48 .
dr-xr-xr-x 1 root root 4192 9月 11 19:49 ..
-rwxrwxrwx 1 root root  99 9月 11 19:47 hellowin.c
xjy1170500913@ubuntu:~/hitics$
```

图 2-2 Ubuntu 与 Windows 共享目录截图

第3章 Windows 软硬件系统观察分析

3.1 查看计算机基本信息 (2分)

截图： 控制面板->系统



(a)

命令行 systeminfo 执行结果(至少包含启动设备行)

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.17134.228]
(c) 2018 Microsoft Corporation. 保留所有权利。

C:\Users\xjy>systeminfo

主机名:          LENOVO7000
OS 名称:         Microsoft Windows 10 家庭中文版
OS 版本:         10.0.17134 阈值 Build 17134
OS 制造商:       Microsoft Corporation
OS 配置:         独立工作站
OS 构件类型:     Multiprocessor Free
注册的所有人:   Windows 用户
注册的组织:
产品 ID:         00342-30262-00002-AAOEM
初始安装日期:   2018/6/1, 18:58:00
系统启动时间:   2018/9/11, 16:00:19
系统制造商:     LENOVO
系统型号:       80X3
系统类型:       x64-based PC
处理器:         安装了 1 个处理器。
                 [01]: Intel64 Family 6 Model 142 Stepping 9 GenuineIntel ~400 Mhz
BIOS 版本:      LENOVO 4QCN25WW (V1.05), 2017/4/25
Windows 目录:   C:\WINDOWS
系统目录:       C:\WINDOWS\system32
启动设备:       \Device\HarddiskVolume4
系统区域设置:   zh-cn; 中文(中国)
输入法区域设置: zh-cn; 中文(中国)
时区:           (UTC+08:00) 北京, 重庆, 香港特别行政区, 乌鲁木齐
物理内存总量:   8,050 MB
可用的物理内存: 608 MB
虚拟内存: 最大值: 10,980 MB
虚拟内存: 可用: 1,566 MB
虚拟内存: 使用中: 9,414 MB
页面文件位置:   E:\pagefile.sys
域:             \LENOVO7000
登录服务器:     WORKGROUP
修补程序:       安装了 3 个修补程序。
                 [01]: KB4343669
                 [02]: KB4343902
                 [03]: KB4343909
网卡:           安装了 3 个 NIC.
```

```

网卡: 安装了 3 个 NIC。
      [01]: Intel(R) Dual Band Wireless-AC 3165
          连接名: WLAN
          启用 DHCP: 是
          DHCP 服务器: 172.20.0.1
          IP 地址
            [01]: 172.20.113.34
            [02]: fe80::f9f9:2e18:3788:349b
            [03]: 2001:250:fe01:130:c5cb:e857:c2d7:87bb
            [04]: 2001:250:fe01:130:f9f9:2e18:3788:349b
      [02]: VMware Virtual Ethernet Adapter for VMnet1
          连接名: VMware Network Adapter VMnet1
          启用 DHCP: 是
          DHCP 服务器: 192.168.253.254
          IP 地址
            [01]: 192.168.253.1
            [02]: fe80::5eb:dc8b:elbc:9f1
      [03]: VMware Virtual Ethernet Adapter for VMnet8
          连接名: VMware Network Adapter VMnet8
          启用 DHCP: 是
          DHCP 服务器: 192.168.217.254
          IP 地址
            [01]: 192.168.217.1
            [02]: fe80::546b:7c8c:538d:1450
Hyper-V 要求: 虚拟机监视器模式扩展: 是
               固件中已启用虚拟化: 是
               二级地址转换: 是
               数据执行保护可用: 是

```

(b)

图 3-1 Windows 下计算机基本信息

3.2 设备管理器查看 (2 分)

按链接列出设备，找出所有的键盘鼠标设备。写出每一个设备的从根到叶节点的路径。

键盘：Lenovo7000→基于 ACPI x64 的电脑→Microsoft ACPI-Compliant System→PCI Express 根复合体→ Mobile 7th Generation Intel(R) Processor Family I/O LPC Controller (U with iHDCP2.2 Premium) - 9D4E→PS/2 标准键盘

鼠标 1：Lenovo7000→基于 ACPI x64 的电脑→Microsoft ACPI-Compliant System→PCI Express 根复合体→Intel(R) USB 3.0 可扩展主机控制器 -1.0(Microsoft)→USB 根集线器(USB 3.0)→USB Composite Device→USB 输入设备→HID-compliant mouse

鼠标 2（若有）：无。

3 隐藏分区与虚拟内存之分页文件查看 (2 分)

写出计算机主硬盘的各隐藏分区的大小（MB）：

EFI 系统分区：260MB

OEM 分区：1000MB

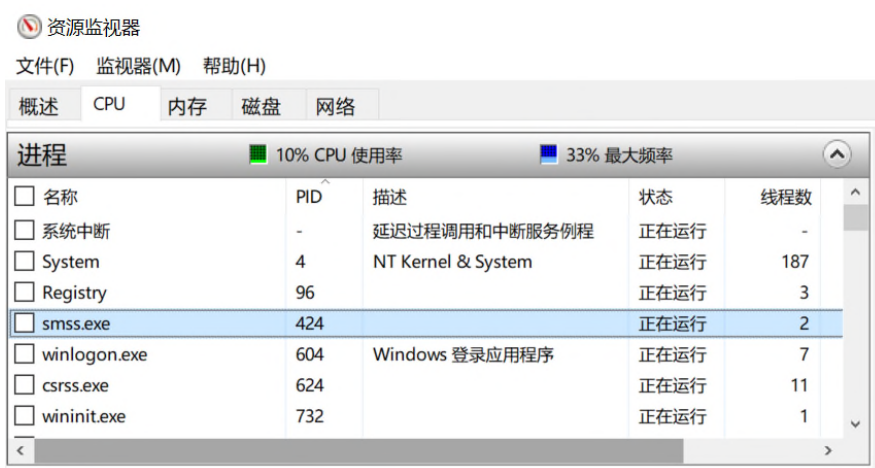
写出 pagefile.sys 的文件大小（Byte）：2929 MB = 3071279104 Byte

C 盘根目录下其他隐藏的系统文件名字为：

Intel MSOCCache ProgramData UserGuidePDF

3.4 任务管理与资源监视 (2 分)

写出你的计算机的 PID 最小的两个任务的名称、描述。



1. 系统中断 延迟过程调用和中断服务例程

2. System NT Kernel & System

3.5 计算机硬件详细信息 (2 分)

CPU 个数： 1 物理核数： 2 逻辑处理器个数： 4

L1 Cache 大小： 128 KB L2 Cache 大小： 512 KB L3 Cache 大小： 3.0 MB

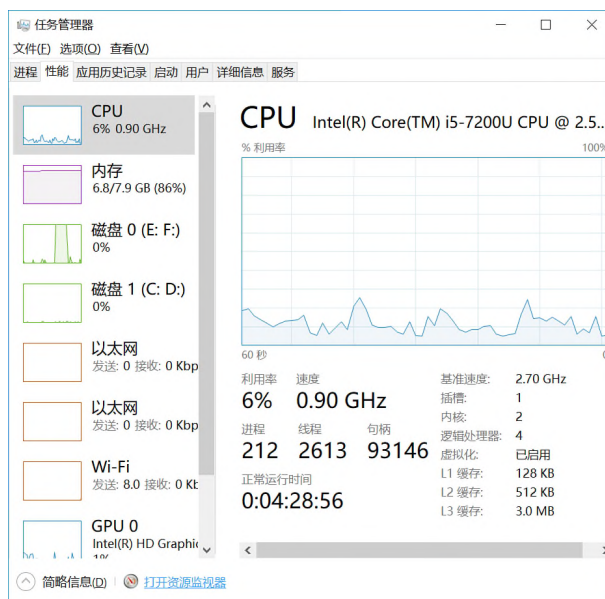


图 3-2 Windows 下计算机硬件详细信息

第4章 Linux 软硬件系统观察分析

4.1 计算机硬件详细信息 (3 分)

CPU 个数: 1 物理核数: 2 逻辑处理器个数: 2

MEM Total: 3.8 G Used: 1.5G Swap: 947 M

```
xjy1170500913@ubuntu:~$ lscpu
架构: x86_64
CPU 运行模式: 32-bit, 64-bit
字节序: Little Endian
CPU: 2
在线 CPU 列表: 0,1
每个核的线程数: 1
每个座的核数: 2
座: 1
NUMA 节点: 1
厂商 ID: GenuineIntel
CPU 系列: 6
型号: 142
型号名称: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
步进: 9
CPU MHz: 2711.998
BogoMIPS: 5423.99
超管理器厂商: VMware
虚拟化类型: 完全
L1d 缓存: 32K
L1i 缓存: 32K
L2 缓存: 256K
L3 缓存: 3072K
NUMA 节点0 CPU: 0,1
标记: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
mon nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq ssse3 fma cx16
r lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti fsgsbase tsc_adjust
```

```
xjy1170500913@ubuntu:~$ cat /proc/meminfo
MemTotal: 4015684 kB
MemFree: 732204 kB
MemAvailable: 2255316 kB
Buffers: 133192 kB
Cached: 1509328 kB
SwapCached: 0 kB
Active: 1790740 kB
Inactive: 785388 kB
Active(anon): 934864 kB
Inactive(anon): 18540 kB
Active(file): 855876 kB
Inactive(file): 766848 kB
Unevictable: 16 kB
Mlocked: 16 kB
SwapTotal: 969960 kB
SwapFree: 969960 kB
Dirty: 144 kB
Writeback: 0 kB
AnonPages: 933540 kB
Mapped: 313928 kB
Shmem: 19800 kB
Slab: 255016 kB
SReclaimable: 189792 kB
SUnreclaim: 65224 kB
KernelStack: 11600 kB
PageTables: 44792 kB
NFS_Unstable: 0 kB
Bounce: 0 kB
WritebackTmp: 0 kB
CommitLimit: 2977800 kB
Committed_AS: 4594692 kB
VmallocTotal: 34359738367 kB
VmallocUsed: 0 kB
VmallocChunk: 0 kB
HardwareCorrupted: 0 kB
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
CmaTotal: 0 kB
CmaFree: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 161600 kB
DirectMap2M: 4032512 kB
DirectMap1G: 2097152 kB
```

```
xjy1170500913@ubuntu:~$ free -h
              总计          已用          空闲          共享          缓冲/缓存          可用
内存:         3.8G         1.5G         645M           19M         1.7G         2.1G
交换:         947M           0B         947M
```


图 4-1 Linux 下计算机硬件详细信息截图

4.2 任务管理与资源监视 (2 分)

写出 Linux 下的 PID 最小的两个任务的 PID、名称 (Command)。

1.PID:1 Command: /sbin/init splash

2.PID:344 Command: /lib/systemd/systemd-journald

输入 htop 命令:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	156M	8928	6572	S	0.0	0.2	0:08.91	/sbin/init splash
344	root	19	-1	100M	22936	21808	S	0.0	0.6	0:06.47	/lib/systemd/systemd-journald
351	root	20	0	50144	8640	3156	S	0.0	0.2	0:03.73	/lib/systemd/systemd-udev
474	systemd-t	20	0	142M	3344	2788	S	0.0	0.1	0:00.36	/lib/systemd/systemd-timesyncd
475	systemd-r	20	0	70744	5296	4744	S	0.0	0.1	0:01.42	/lib/systemd/systemd-resolved

4.3 共享目录的文件系统信息 (3 分)

写出 Linux 下的 hitics 共享目录对应的文件系统的基本信息:

名称: vmhgfs-fuse 容量: 391G 挂载点: /mnt/hgfs

```
xjy1170500913@ubuntu:~$ df -h
```

文件系统	容量	已用	可用	已用%	挂载点
udev	1.9G	0	1.9G	0%	/dev
tmpfs	393M	2.1M	391M	1%	/run
/dev/sda1	20G	8.8G	9.8G	48%	/
tmpfs	2.0G	0	2.0G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	2.0G	0	2.0G	0%	/sys/fs/cgroup
/dev/loop0	13M	13M	0	100%	/snap/gnome-characters/117
/dev/loop7	13M	13M	0	100%	/snap/gnome-characters/103
/dev/loop3	15M	15M	0	100%	/snap/gnome-logs/37
/dev/loop1	15M	15M	0	100%	/snap/gnome-logs/40
/dev/loop2	87M	87M	0	100%	/snap/core/4917
/dev/loop6	35M	35M	0	100%	/snap/gtk-common-themes/319
/dev/loop8	3.8M	3.8M	0	100%	/snap/gnome-system-monitor/51
/dev/loop4	2.4M	2.4M	0	100%	/snap/gnome-calculator/180
/dev/loop5	141M	141M	0	100%	/snap/gnome-3-26-1604/70
/dev/loop11	3.8M	3.8M	0	100%	/snap/gnome-system-monitor/54
/dev/loop10	2.4M	2.4M	0	100%	/snap/gnome-calculator/199
/dev/loop9	2.3M	2.3M	0	100%	/snap/gnome-calculator/222
/dev/loop12	88M	88M	0	100%	/snap/core/5328
tmpfs	393M	16K	393M	1%	/run/user/121
vmhgfs-fuse	391G	137G	255G	35%	/mnt/hgfs

4.4 Linux 下网络系统信息 (2 分)

写出本虚拟机的 IPv4 地址: 192.168.217.130

mac 地址: 00:0c:29:70:1e:d8

```
xjy1170500913@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.217.130 netmask 255.255.255.0 broadcast 192.168.217.255
    inet6 fe80::603f:31c9:683b:b7f1 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:70:1e:d8 txqueuelen 1000 (以太网)
    RX packets 11322 bytes 8305481 (8.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2864 bytes 271666 (271.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 4453 bytes 306896 (306.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4453 bytes 306896 (306.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

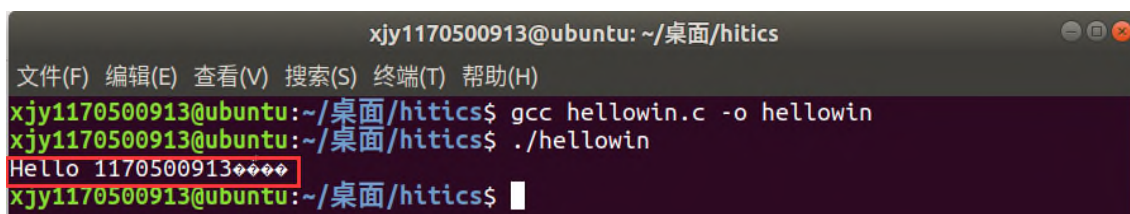
图 4-1 Linux 下网络系统信息

第5章 以16进制形式查看程序 Hello.c

5.1 请查看 HelloWin.c 与 HelloLinux.c 的编码 (3分)

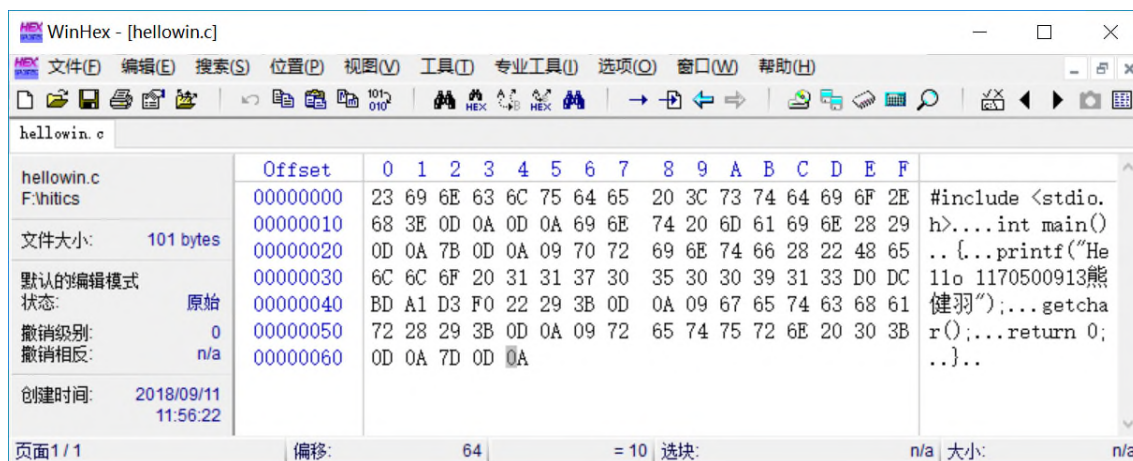
hellowin.c 采用 ASCII 编码, hellolinux.c 采用 UTF-8 编码, 你的姓名: 熊健羽 分别编码为: D0 DC BD A1 D3 F0 与 e7 86 8a e5 81 a5 e7 be bd。

hellowin.c 在 Linux 下用 gcc 缺省模式编译后运行结果为, 如图:



```
xjy1170500913@ubuntu: ~/桌面/hitcs
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
xjy1170500913@ubuntu:~/桌面/hitcs$ gcc hellowin.c -o hellowin
xjy1170500913@ubuntu:~/桌面/hitcs$ ./hellowin
Hello 1170500913◆◆◆◆
xjy1170500913@ubuntu:~/桌面/hitcs$
```

Windows 下用 winhex 查看 hellowin.c, 如下图:



Ubuntu 下用终端命令查看 hellolinux.c, 如下图:

```

xjy1170500913@ubuntu: ~/桌面/hitcs
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
xjy1170500913@ubuntu:~/桌面/hitcs$ od -Ax -tcx1 hellolinux.c
000000  #   i   n   c   l   u   d   e   <   s   t   d   i   o   .
23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e
000010  h   >   \n   \n   i   n   t   m   a   i   n   (   )   \n   {
68 3e 0a 0a 69 6e 74 20 6d 61 69 6e 28 29 0a 7b
000020  \n   \t   p   r   i   n   t   f   (   "   H   e   l   l   o
0a 09 70 72 69 6e 74 66 28 22 48 65 6c 6c 6f 20
000030  1   1   7   0   5   0   0   9   1   3   3   4   7   2   0   6   2   1   2   3   4   5   2   0   1   2   4   5
31 31 37 30 35 30 30 39 31 33 e7 86 8a e5 81 a5
000040 347 276 275   \   n   "   )   ;   \n   \t   r   e   t   u   r   n
e7 be bd 5c 6e 22 29 3b 0a 09 72 65 74 75 72 6e
000050  0   ;   \n   }   \n
20 30 3b 0a 7d 0a
000056
xjy1170500913@ubuntu:~/桌面/hitcs$

```

5.2 请查看 HelloWin.c 与 HelloLinux.c 的回车 (3 分)

Windows 下的回车编码为: 0D 0A，Linux 下的回车编码为: 0A。

交叉打开文件的效果是 windows 下 (ASCII 编码) 打开 hellolinux.c, 英文字符正常显示, 中文字符出现乱码 ; Linux 下 (UTF-8 编码) 打开 hellowin.c, 英文字符正常显示, 中文字符出现乱码。

```

F:\hitcs\hellolinux.c - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
hellolinux.c hello.s datatype.c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello 1170500913 谭姆们缇絳n");
6     return 0;
7 }

```

```

xjy1170500913@ubuntu: ~/桌面/hitcs
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello 1170500913 谭姆们缇絳n");
6     return 0;
7

```

第 6 章 程序的生成 Cpp、Gcc、As、ld

6.1 请提交每步生成的文件（4 分）

hello.i hello.s hello.o hello.out (附上 hellolinux.c)

链接如下（双击图标打开）：



[hello.i](#) [hello.s](#) [hello.o](#) [hello.out](#) [hellolinux.c](#)

第 7 章 计算机系统的基本信息获取编程

7.1 请提交源程序文件（10 分）

isLittleEndian.c

cpuWordSize.c

链接如下（双击图标打开）：



isLittleEndian.c

[isLittleEndian.c](#)



cpuWordSize.c

[cpuWordSize.c](#)

第 8 章 计算机数据类型的本质

8.1 请提交源程序文件 Datatype.c (10 分)

要求有 main 函数进行测试。

链接如下（双击图标打开）：



datatype.c

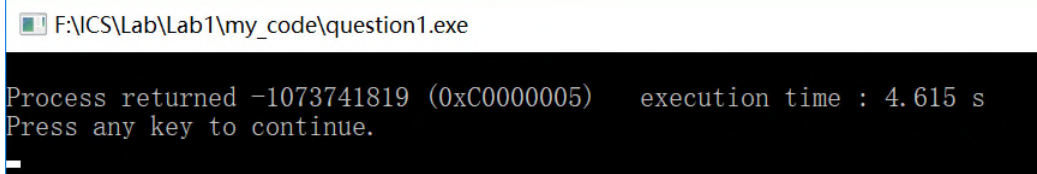
[datatype.c](#)

第 9 章 程序运行分析

9.1 sum 的分析 (20 分)

```
int sum(int a[], unsigned len)
{
    int i, sum = 0;
    for (i = 0; i <= len-1; i++)
        sum += a[i];
    return sum;
}
```

当用 `len = 0` 调用时，程序异常结束，如图：



Q:为什么程序这样的运行结果？

A: 变量 `len` 被定义为无符号整型。无符号类型与有符号类型同时参与运算时，有符号类型会转换为无符号类型进行计算。

因此，当 `len = 0` 时，`0 - 1` 运算将使用无符号加法：`-1` 对应的无符号数为 $2^{32} - 1 = \text{UMax}$ ，因此 `0 - 1` 的值为 `UMax`。由于任意的 `unsigned int` 型都是小于或等于 `UMax` 的，所以该循环控制表达式总为真，因此，程序将试图访问数组 `a` 的非法元素，导致程序异常结束。

Q:怎么改进程序？

A:有两种方法可以改正代码，其一是将 `len` 声明为 `int` 类型；其二是将 `for` 的循环控制表达式改为 `i < len`。

9.2 float 的分析 (20 分)

程序运行结果是：

```
F:\ICS\Lab\Lab1\my_code\question2
请输入一个浮点数: 61.419997
这个浮点数的值是: 61.419998
请输入一个浮点数: 61.419998
这个浮点数的值是: 61.419998
请输入一个浮点数: 61.419999
这个浮点数的值是: 61.419998
请输入一个浮点数: 61.420000
这个浮点数的值是: 61.419998
请输入一个浮点数: 61.420001
这个浮点数的值是: 61.420002
请输入一个浮点数: 0
这个浮点数的值是: 0.000000
```

```
F:\ICS\Lab\Lab1\my_code\questio
请输入一个浮点数: 10.186810
这个浮点数的值是: 10.186810
请输入一个浮点数: 10.186811
这个浮点数的值是: 10.186811
请输入一个浮点数: 10.186812
这个浮点数的值是: 10.186812
请输入一个浮点数: 10.186813
这个浮点数的值是: 10.186813
请输入一个浮点数: 10.186814
这个浮点数的值是: 10.186814
请输入一个浮点数: 10.186815
这个浮点数的值是: 10.186815
请输入一个浮点数: 0
这个浮点数的值是: 0.000000
```

Q: 为什么是这样的运行结果?

A: 根据 IEEE 标准, float 型以 1 位符号位, 8 位阶码, 23 位尾数的形式存储。23 位尾数能够表示的二进制最高精确有效数位为 24 位。

- ① 第一组数据, 阶码 $E = 5$, 这一段数轴两个相邻浮点数的距离为 $2^5 \times 2^{-23} \approx 3.8 \times 10^{-6}$, 即只能精确到小数点后 5 位, 第 6 位是不精确的, 因此多个小数点后第六位不同的小数, 转化为浮点数时发生舍入, 在内存中变成了同一个数据, 从而输出了同一个十进制数。

把第一组数先写成定点数的形式: (红色为无法精确表示的位)

61.419997: 111101.0110101111000010011101100011000110...

61.419998: 111101.0110101111000010011111101001010100...

61.419999: 111101.0110101111000010100001101111100010...

61.420000: 111101.0110101111000010100011110101100001...

61.420001: 111101.011010111100001010010111101111110...

根据舍入的法则, 写出 IEEE 754 表示:

第一组 (舍入之后):

61.419997:

01000010011101011010111000010100

61.419998:

01000010011101011010111000010100

61.419999:

01000010011101011010111000010100

61.420000:

01000010011101011010111000010100

61.420001:

01000010011101011010111000010101

可见输入 61.419997, 61.419998, 61.419999, 61.420000 时内存中的数据相同, 化成十进制为 61.4199981689453125, 舍入之后即为 61.419998; 输入 61.420001 时, 化成十进制为 61.420001983642578125, 舍入之后即为 61.420002。

- ② 第二组数据, 阶码 $E = 3$, 这一段数轴两个相邻浮点数的距离为 $2^3 \times 2^{-23} \approx 9.5 \times 10^{-7}$, 即能精确到小数点后 6 位, 第 7 位是不精确的, 因此第二组输入的数据均可精确表示, 输出正确的值。

Q: 使用浮点数时应注意什么?

A: 使用浮点数时应注意对精度的要求, 越靠近原点, 数越稠密, 表示的精度越高, 越远离原点, 数越稀疏, 表示的精度越低;

同时, 浮点数不能简单的作比较。比如从上述例子可以看出, 原始数值不相等的两个数, 存储为浮点数后可能相等。

第 10 章 总结

10.1 请总结本次实验的收获

1. 粗略地漫游了计算机系统,使用任务管理器、资源监视器、性能监视器、winhex、鲁大师等进行计算机软硬件系统的观察与分析,对计算机的硬件、操作系统等有了初步的认识;
2. 对我以前从未使用过的操作系统 Ubuntu 有了初步的了解,熟悉了其中的基本命令与操作,为之后的实验打下基础;
3. 对数据在计算机中的存储与表示有了更加深刻的认识,对字符的不同编码,整数的补码、无符号表示,浮点数的精度问题,浮点数的 IEEE 754 标准等有了更深刻的理解;
4. 通过编写代码,了解了数据在内存中的存储形式、顺序、不同数据类型的大小,理解了小端/大端的含义,以及 64 位编译与 32 位编译的具体区别;
5. 写实验报告的过程中,锻炼了 word 文档排版布局、语言构思的能力。

10.2 请给出对本次实验内容的建议

暂无。实验内容充实而丰富,我获益匪浅。

注:本章为酌情加分项。

参考文献

- [1] 大卫 R.奥哈拉伦，兰德尔 E.布莱恩特. 深入理解计算机系统[M]. 机械工业出版社.2018.4