

哈尔滨工业大学

实验报告

实 验（一）

题 目 计算机系统漫游

学 号 120L020701

班 级 2003005

学 生 董琦

指 导 教 师 吴锐

实 验 地 点 G712

实 验 日 期 2022/3/18 56 节

哈尔滨工业大学计算学部

目 录

第 1 章 实验基本信息	- 4 -
1.1 实验目的	- 4 -
1、运用现代工具进行计算机软硬件系统的观察与分析。	- 4 -
2、运用现代工具进行 LINUX 下 C 语言的编程调试, 掌握程序的生成步骤。 ..	- 4 -
3、初步掌握计算机系统的基本知识与各种类型的数据表示。	- 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 -
第 2 章 实验环境建立	- 5 -
2.1 WINDOWS 下 HELLO 程序的编辑与运行 (5 分)	- 5 -
2.2 LINUX 下 HELLO 程序的编辑与运行 (5 分)	- 5 -
第 3 章 WINDOWS 软硬件系统观察分析	- 6 -
3.1 查看计算机基本信息 (2 分)	- 6 -
3.2 设备管理器查看 (2 分)	- 7 -
3 隐藏分区与虚拟内存之分页文件查看 (2 分)	- 7 -
3.4 任务管理与资源监视 (2 分)	- 8 -
3.5 CPUZ 下的计算机硬件详细信息 (2 分)	- 8 -
第 4 章 LINUX 软硬件系统观察分析	- 9 -
4.1 计算机硬件详细信息 (3 分)	- 9 -
4.2 任务管理与资源监视 (2 分)	- 9 -
4.3 磁盘任务管理与资源监视 (3 分)	- 10 -
4.4 LINUX 下网络系统信息 (2 分)	- 10 -
第 5 章 LINUX 下的 SHOWBYTE 程序	- 11 -
5.1 源程序提交 (8 分)	- 11 -
5.2 运行结果比较 (2 分)	- 11 -
第 6 章 程序的生成 CPP、GCC、AS、LD	- 12 -
6.1 请提交每步生成的文件 (10 分)	- 12 -
第 7 章 计算机数据类型的本质	- 13 -
7.1 运行 SIZEOF.C 填表 (5 分)	- 13 -
7.2 请提交源程序文件 SIZEOF.C (5 分)	- 13 -

第 8 章 程序运行分析	- 14 -
8.1 SUM 的分析（10 分）	- 14 -
8.2 FLOAT 的分析（10 分）	- 14 -
8.3 程序优化（20 分）	- 15 -
第 9 章 总结	- 17 -
9.1 请总结本次实验的收获	- 17 -
9.2 请给出对本次实验内容的建议	- 17 -
参考文献	- 18 -

第 1 章 实验基本信息

1.1 实验目的

- 1、运用现代工具进行计算机软硬件系统的观察与分析。
- 2、运用现代工具进行 Linux 下 C 语言的编程调试，掌握程序的生成步骤。
- 3、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

处理器 Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz 8 核
机带 RAM 16.0 GB (15.8 GB 可用)
系统类型 64 位操作系统, 基于 x64 的处理器

1.2.2 软件环境

Windows:

版本 Windows 11 家庭中文版

版本 21H2

安装日期 2022/2/4

操作系统版本 22000.556

体验 Windows 功能体验包 1000.22000.556.0

Ubuntu:

版本 Ubuntu 20.04.3 LTS

类型 64 位

1.2.3 开发工具

Windows: Visual Studio 2019

Ubuntu: VScode gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04)

第 2 章 实验环境建立

2.1 Windows 下 hello 程序的编辑与运行 (5 分)

截图：要求有 Windows 状态行，Visual Studio 界面，源程序界面，运行结果界面。

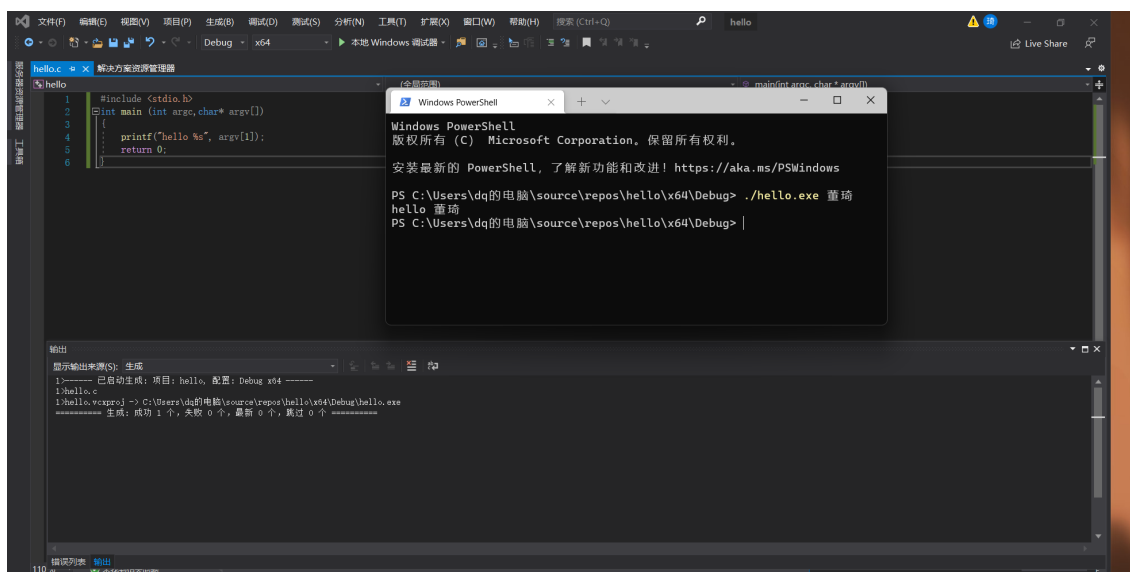


图 2-1 Windows 下 hello 运行截图

2.2 Linux 下 hello 程序的编辑与运行 (5 分)

截图：要求有 Ubuntu 的 OS 窗口，Codeblocks 界面，源程序界面，运行结果界面。

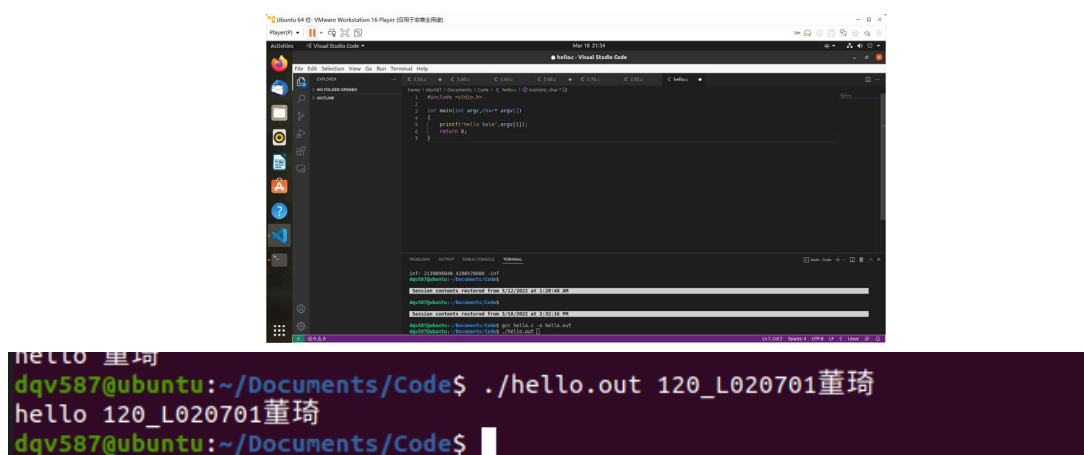
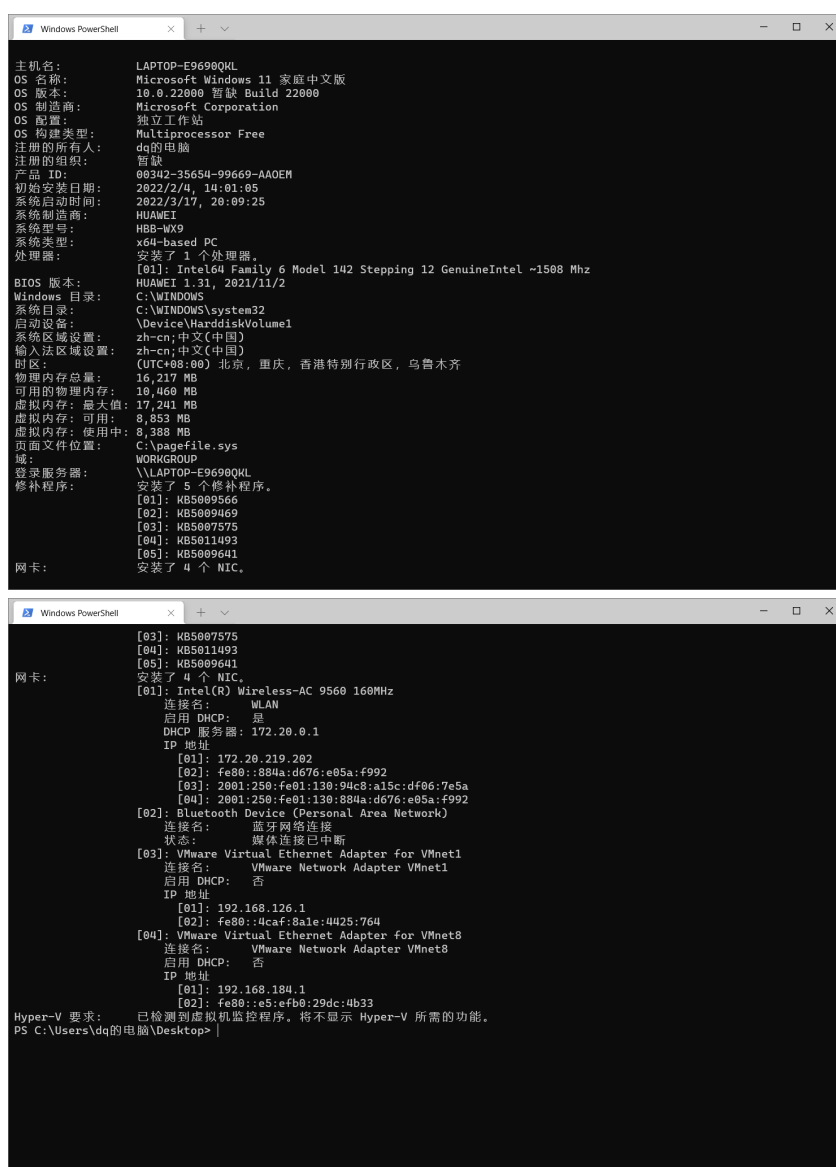


图 2-2 Linux 下 hello 运行截图

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）

运行 Windows 管理工具中的“系统信息”程序，查看 CPU、物理内存、系统目录、启动设备、页面文件等信息，并截图



```
Windows PowerShell
主机名: LAPTOP-E9690QKL
OS 名称: Microsoft Windows 11 家庭中文版
OS 版本: 10.0.22000 预览 Build 22000
OS 制造商: Microsoft Corporation
OS 配置: 独立工作站
OS 构建类型: Multiprocessor Free
注册的所有人: dq的电脑
注册的组织: 智缺
产品 ID: 00342-35654-99669-AAOEM
初始安装日期: 2022/2/4, 14:01:05
系统启动时间: 2022/3/17, 20:09:25
系统制造商: HUAWEI
系统型号: H88-WX9
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 12 GenuineIntel ~1508 Mhz
BIOS 版本: HUAWEI 1.31, 2021/11/2
Windows 目录: C:\WINDOWS
系统目录: C:\WINDOWS\system32
启动设备: \Device\HarddiskVolume1
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: 16,217 MB
可用的物理内存: 10,460 MB
虚拟内存: 最大值: 17,241 MB
虚拟内存: 可用: 8,853 MB
虚拟内存: 使用中: 8,388 MB
页面文件位置: C:\pagefile.sys
域: WORKGROUP
登录服务器: \\LAPTOP-E9690QKL
修补程序: 安装了 5 个修补程序。
[01]: KB5009566
[02]: KB5009469
[03]: KB5007575
[04]: KB5011493
[05]: KB5009641
网卡: 安装了 4 个 NIC。
[01]: Intel(R) Wireless-AC 9560 160MHz
连接名: WLAN
启用 DHCP: 是
DHCP 服务器: 172.20.0.1
IP 地址: [01]: 172.20.219.202
[02]: fe80::884a:d676:e05a:f992
[03]: 2001:250:fe01:130:94c8:a15c:df06:7e5a
[04]: 2001:250:fe01:130:884a:d676:e05a:f992
[02]: Bluetooth Device (Personal Area Network)
连接名: 蓝牙网络连接
状态: 媒体连接已中断
[03]: VMware Virtual Ethernet Adapter for VMnet1
连接名: VMware Network Adapter VMnet1
启用 DHCP: 否
IP 地址: [01]: 192.168.126.1
[02]: fe80::4caf:8a1e:4025:764
[04]: VMware Virtual Ethernet Adapter for VMnet8
连接名: VMware Network Adapter VMnet8
启用 DHCP: 否
IP 地址: [01]: 192.168.184.1
[02]: fe80::e5:efb0:29dc:4b33
Hyper-V 要求: 已检测到虚拟机监控程序, 将不显示 Hyper-V 所需的功能。
PS C:\Users\dq\电脑\Desktop>
```

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：

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---Intel® USB 3.1 可扩展主机控制器-1.10 (Microsoft)---USB 根集线器 (USB3.0)---USB Composite Device---USB 输入设备---HID keyboard Device

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---Intel® USB 3.1 可扩展主机控制器-1.10 (Microsoft)---USB 根集线器 (USB3.0)---英特尔®无线 Bluetooth®---Microsoft 蓝牙 LE 枚举器---MX Master3---符合蓝牙低功耗 GATT 的 HID 设备---HID Keyboard Device

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---I/O LPC Controller - 0284 for Intel(R) 400 Series Chipset Family On-Package Platform Controller Hub---PS/2 标准键盘

鼠标：

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---Intel® USB 3.1 可扩展主机控制器-1.10 (Microsoft)---USB 根集线器 (USB3.0)---USB Composite Device---USB 输入设备---HID-compliant mouse

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---Intel®串行 IO I2C 主机控制器-02E8---I2C HID 设备---HID-compliant mouse

LAPTOP---基于 ACPI x64 的电脑---Microsoft ACPI-Compliant System---PCI Express 根复合体---Intel® USB 3.1 可扩展主机控制器-1.10 (Microsoft)---USB 根集线器 (USB3.0)---英特尔®无线 Bluetooth®---Microsoft 蓝牙 LE 枚举器---MX Master3---符合蓝牙低功耗 GATT 的 HID 设备---HID-compliant mouse

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)：

卷	布局	类型	文件系统	状态	容量	可用空间	% 可用
(磁盘 0 磁盘分区 1)	简单	基本		状态良好 (EFI 系统分区)	100 MB	100 MB	100 %
(磁盘 0 磁盘分区 5)	简单	基本		状态良好 (恢复分区)	512 MB	512 MB	100 %
(磁盘 0 磁盘分区 6)	简单	基本		状态良好 (恢复分区)	12.00 GB	12.00 GB	100 %
(磁盘 0 磁盘分区 7)	简单	基本		状态良好 (恢复分区)	1.00 GB	1.00 GB	100 %

写出 pagefile.sys 的文件大小 (Byte): 1048576K

C 盘根目录下其他隐藏的系统文件名字为:

\$WinREAgent	2022/3/10 1:21	文件夹
Intel	2022/3/17 20:09	文件夹
OneDriveTemp	2020/7/10 16:10	文件夹

3.4 任务管理与资源监视 (2 分)

写出你的计算机的 PID 为 “-”、最小与最大的 3 个任务的 PID、名称、描述。

- 系统中断 延迟过程调用和中断服务例程
- 4 System NT Kernel\System
- 21488 usysdialg.exe Huorong Sysdiag Helper

3.5 CPUZ 下的计算机硬件详细信息 (2 分)

CPU 个数: 1 物理核数: 4 逻辑处理器个数: 8 L3 Cache 大小: 6MB

图 3-2 CPUZ 下 CPU 的基本信息



第 4 章 Linux 软硬件系统观察分析

(泰山服务器)

4.1 计算机硬件详细信息 (3 分)

CPU 个数: 2 物理核数: 96 逻辑处理器个数: 96
MEM Total: 192616 Used: 21862 Swap: 8191

```
stu_120L020701@node210:~/Desktop$ lscpu
Architecture:          aarch64
CPU op-mode(s):        64-bit
Byte Order:             Little Endian
CPU(s):                 96
On-line CPU(s) list:    0-95
Thread(s) per core:     1
Core(s) per socket:     48
Socket(s):              2
NUMA node(s):          4
Vendor ID:              0x48
Model:                  0
Stepping:               0x1
CPU max MHz:            2600.0000
CPU min MHz:            200.0000
BogoMIPS:               200.00
L1d cache:              6 MiB
L1i cache:              6 MiB
L2 cache:               48 MiB
L3 cache:               192 MiB
NUMA node0 CPU(s):      0-23
NUMA node1 CPU(s):      24-47
NUMA node2 CPU(s):      48-71
NUMA node3 CPU(s):      72-95
Vulnerability Itlb multihit: Not affected
Vulnerability L1tf:      Not affected
Vulnerability Mds:       Not affected
Vulnerability Meltdown:  Not affected
Vulnerability Spec store bypass: Not affected
Vulnerability Spectre v1: Mitigation; __user pointer sanitization
Vulnerability Spectre v2: Not affected
Vulnerability Srbds:     Not affected
Vulnerability Tsx async abort: Not affected
Flags:                   fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrrn jscv
stu_120L020701@node210:~/Desktop$ free -m
              total        used        free      shared  buff/cache   available
Mem:         192616        21862       153681         297       17072       169178
Swap:          8191           31         8160
```

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 systemd
- 2 kthreadd

4.3 磁盘任务管理与资源监视 (3 分)

1. /dev/sda 设备的大小 1946 GB, 类型 disk
2. Units sectors of 1 * 512 = 512 bytes Sector Size 512 bytes / 512 bytes

```
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9f6dde6a
```

4.4 Linux 下网络系统信息 (2 分)

写出机器正联网用的网卡 IPv4 地址: 10.42.0.1

mac 地址: fe80::a0f9:f2ff:fe29:dc67

```
stu_1201020701@node210:~/Desktop$ ifconfig -a
cni0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450
    inet 10.42.0.1 netmask 255.255.255.0 broadcast 10.42.0.255
    inet6 fe80::a0f9:f2ff:fe29:dc67 prefixlen 64 scopeid 0x20<link>
    ether a2:f9:f2:29:dc:67 txqueuelen 1000 (Ethernet)
    RX packets 6666600 bytes 2301692876 (2.3 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6940031 bytes 6542235466 (6.5 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:fcff:fe89:ce88 prefixlen 64 scopeid 0x20<link>
    ether 02:42:fc:89:ce:88 txqueuelen 0 (Ethernet)
    RX packets 424618 bytes 78253728 (78.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 581900 bytes 222125191 (222.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp125s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.11.210 netmask 255.255.255.0 broadcast 192.168.11.255
    inet6 fe80::6eeb:b6ff:fe15:932b prefixlen 64 scopeid 0x20<link>
    ether 6c:eb:b6:15:93:2b txqueuelen 1000 (Ethernet)
    RX packets 11733902 bytes 7667458997 (7.6 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 9820332 bytes 7678026641 (7.6 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp125s0f1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 6c:eb:b6:15:93:2c txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp125s0f2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 6c:eb:b6:15:93:2d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp125s0f3: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
```

图 4-2 Linux 下网络系统信息

第 5 章 Linux 下的 showbyte 程序

(10 分)

5.1 源程序提交 (8 分)

showbyte.c 与实验报告放在一个压缩包里

5.2 运行结果比较 (2 分)

格式稍微有点不一样，但是结果一样的。

运行 `od -Ax -tcx1 hello.c` 以及 `showbyte.c`，结果截图。

```
r:72  n:6e  :20  0:30  ;:3b  \n:a  }:7d  dqv587@ubuntu:~/Documents/Code$ od -Ax -tcx1 hello.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 ( i n t
68 3e 0a 0a 69 6e 74 20 6d 61 69 6e 28 69 6e 74
000020  a r g c , c h a r * a r g v
20 61 72 67 63 2c 63 68 61 72 2a 20 61 72 67 76
000030  [ ] ) \n { \n p r i n t f
5b 5d 29 0a 7b 0a 20 20 20 20 70 72 69 6e 74 66
000040  ( " h e l l o % s \n " , a r
28 22 68 65 6c 6c 6f 20 25 73 5c 6e 22 2c 61 72
000050  g v [ 1 ] ) ; \n r e t u
67 76 5b 31 5d 29 3b 0a 20 20 20 20 72 65 74 75
000060  r n \0 ; \n }
72 6e 20 30 3b 0a 7d
000067
```

图 5-1 OD 的输出结果

```
dqv587@ubuntu:~/Documents/Code$ ./show_bytes hello.c
#:23  l:69  n:6e  c:63  l:6c  u:75  d:64  e:65  :20  <:3c  s:73  t:74  d:64  l:69  o:6f  .:2e
h:68  >:3e  \n:a  \n:a  l:69  n:6e  t:74  :20  m:6d  a:61  l:69  n:6e  (:28  l:69  n:6e  t:74
:20  a:61  r:72  g:67  c:63  ,:2c  c:63  h:68  a:61  r:72  *:2a  :20  a:61  r:72  g:67  v:76
[:5b  ]:5d  ):29  \n:a  {:7b  \n:a  :20  :20  :20  :20  p:70  r:72  i:69  n:6e  t:74  f:66
(:28  ":22  h:68  e:65  l:6c  l:6c  o:6f  :20  %:25  s:73  \:5c  n:6e  ":22  ,:2c  a:61  r:72
g:67  v:76  [:5b  1:31  ]:5d  ):29  ;:3b  \n:a  :20  :20  :20  :20  r:72  e:65  t:74  u:75
r:72  n:6e  :20  0:30  ;:3b  \n:a  }:7d  dqv587@ubuntu:~/Documents/Code$
```

图 5-2 showbyte 的输出结果

第 6 章 程序的生成 Cpp、Gcc、As、ld

6.1 请提交每步生成的文件 (10 分)

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

第 7 章 计算机数据类型的本质

7.1 运行 sizeof.c 填表 (5 分)

	Win/VS/x86	Win/VS/x64	Linux/M32	Linux/M64
char	1	1	1	1
short	2	2	2	2
int	4	4	4	4
long	4	4	4	8
long long	8	8	8	8
float	4	4	4	4
double	8	8	8	8
long double	8	8	12	16
指针	4	8	4	8

7.2 请提交源程序文件 sizeof.c (5 分)

第 8 章 程序运行分析

8.1 sum 的分析 (10 分)

1.截图说明运行结果，并原因分析。

```
dqv587@ubuntu:~/Documents/Code$ gcc sum.c -o sum.out
dqv587@ubuntu:~/Documents/Code$ ./sum.out
Sum:45
dqv587@ubuntu:~/Documents/Code$ gcc sum.c -o sum.out
dqv587@ubuntu:~/Documents/Code$ ./sum.out
Segmentation fault (core dumped)
```

结果描述：第一个结果是 len=10，第二个结果是 len=0.

问题描述：可以看到当 len 等于 0 时程序运行错误，报错 Segmentation fault (core dumped)

分析原因：当 len 等于 0 时，由于传入参数是 unsigned 类型，减 1 操作后发生下溢出，然后访问数组发生越界，程序报错。

2.论述改进方法

方法 1：将 len 类型定义为 int，这样 len 为 0 时，减 1 后就不会发生下溢出。

方法 2：将 $i \leq \text{len}-1$ 改为 $i+1 \leq \text{len}$ ，这样避免了对 len 的计算，也不会发生溢出情况了。

8.2 float 的分析 (10 分)

1.运行结果截图，分析产生原因。

```
dqv587@ubuntu:~/Documents/Code$ ./float.c
请输入1个浮点数:10.186810
这个浮点数的值是:10.186810
请输入1个浮点数:10.186811
这个浮点数的值是:10.186811
请输入1个浮点数:10.186812
这个浮点数的值是:10.186812
请输入1个浮点数:10.186813
这个浮点数的值是:10.186813
请输入1个浮点数:10.186814
这个浮点数的值是:10.186814
请输入1个浮点数:10.186815
这个浮点数的值是:10.186815
请输入1个浮点数:0
这个浮点数的值是:0.000000
请输入1个浮点数:61.419997
这个浮点数的值是:61.419998
请输入1个浮点数:61.419998
这个浮点数的值是:61.419998
请输入1个浮点数:61.419999
这个浮点数的值是:61.419998
请输入1个浮点数:61.420000
这个浮点数的值是:61.419998
请输入1个浮点数:61.420001
这个浮点数的值是:61.420002
请输入1个浮点数:0
这个浮点数的值是:0.000000
```

产生原因：规格化浮点数的编码是不均匀的，绝对值小的浮点数密度大，而绝对值大的浮点数密度小。当浮点数在 10.18681 附近时，浮点数可以准确编码这几个值，故显示出来的数与输入的数相同；而浮点数在 61.419998 附近时，浮点数的编码不能将这几个值全部表示出来，故会在二进制编码的基础上发生舍入，因此产生了输入输出数不一致的情况。另外，浮点数舍入遵从向偶数舍入的规则。

2. 论述编程中浮点数比较、汇总统计等应如何正确编程。

浮点数比较：因为浮点数在储存时无法精确表示，故在运算时会产生误差，因此在比较相等时，不应该直接用`==`判断，而应该计算俩数的绝对值，来看这个绝对值大小是否在误差范围之内。

另外，有时候当俩个数不同时，但是由于浮点数编码将这两个数编码为同一个浮点数，这样系统会误判这两个数相等。这种情况下，应该选用精度更高的浮点数类型，从而将这两个数在编码上区分开来。

汇总统计：浮点数运算不满足结合性和分配性，当俩数数量级相差很大时，会发生精度丢失，因此当汇总统计时，应该注意计算顺序，按数量级从小到大进行汇总统计。如果还是不能得到准确结果，那么应该将数据按照数量级分类，使之汇总统计计算不会发生精度丢失，然后再进行分别求和。

故当浮点数比较时要考虑浮点数编码时不能准确表示的情况，汇总统计时要考虑数据间的数量级差异。

8.3 程序优化 (20 分)

1. 截图说明运行结果，分析问题产生原因。

递归：

```

0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 41
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 42
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 43
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 44
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 45
-1.387202
dqv587@ubuntu:~/Documents/Code$ ./g1 46
-2.582630
dqv587@ubuntu:~/Documents/Code$ ./g1 47
-0.631859
dqv587@ubuntu:~/Documents/Code$

dqv587@ubuntu:~/Documents/Code$ gcc g1.c -
dqv587@ubuntu:~/Documents/Code$ ./g1 44
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 45
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 46
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 47
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 48
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 49
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g1 50
0.618034
dqv587@ubuntu:~/Documents/Code$

```

由于斐波那契数列的递归求法时间复杂度为 $O(2^n)$ ，求到 $n=50$ 时所需时间就很大了， $n=100$ 时计算机无法完成，故只展示到 $n=50$ 。

上图右为 `int-float` 组合，可以看到 $n \leq 44$ 时，所得结果近似为 0.618，结果较好，而当 n 大于 44 时，得到结果为负数。原因为当 $n > 44$ 时，`fib(n)` 的值已经超过了 `int` 能表示的范围，发生了上溢出，其中一个数变为负数，故而结果得到一个负数。

上图左为 `long-float` 组合，可以看到 n 在 45 到 50 的范围内时，得到结果均为 0.618，可见此时的 `fib(n)` 的值并没有超过 `long` 的编码范围，但是由于递归的时间复杂度原因，无法验证 $n=100$ 时是否不会溢出，故更大的 n 值在循环方式中分析。

循环:

```
dqv587@ubuntu:~/Documents/Code$ ./g2 100
-1.293602
dqv587@ubuntu:~/Documents/Code$ ./g2 45
-1.387202
dqv587@ubuntu:~/Documents/Code$ ./g2 44
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2 451
-0.429825
dqv587@ubuntu:~/Documents/Code$ ./g2 51
-0.792746
dqv587@ubuntu:~/Documents/Code$ ./g2 52
4.824990
dqv587@ubuntu:~/Documents/Code$ ./g2 53
0.171674
dqv587@ubuntu:~/Documents/Code$ ./g2 54
-1.203765
dqv587@ubuntu:~/Documents/Code$ ./g2 100
0.257925
dqv587@ubuntu:~/Documents/Code$ ./g2 50
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2 76
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2 88
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2 94
1.353470
dqv587@ubuntu:~/Documents/Code$ ./g2 91
-1.207078
dqv587@ubuntu:~/Documents/Code$ ./g2 90
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2
```

斐波那契数列循环求法时间复杂度为 $O(n)$, 故可以求到 100.

上左图为 int/float 版本, 可以看到当 n 为 45 时, 发生了上溢出, 与递归情况一样, 上右图为 long/float 版本, 可以看到 n 为 92 时, 仍然会发生溢出, 可见 long 型整数仍无法容纳 fib(100) 的值。

```
dqv587@ubuntu:~/Documents/Code$ gcc g2.c -o g2
dqv587@ubuntu:~/Documents/Code$ ./g2 100
0.257925
dqv587@ubuntu:~/Documents/Code$ ./g2 91
-1.207078
dqv587@ubuntu:~/Documents/Code$ ./g2 90
0.618034
dqv587@ubuntu:~/Documents/Code$
```

上图为 long/double 版本, 可以看到与 long/float 版本没有什么区别。

```
dqv587@ubuntu:~/Documents/Code$ gcc g2.c -o g2
dqv587@ubuntu:~/Documents/Code$ ./g2 100
0.257925
dqv587@ubuntu:~/Documents/Code$ ./g2 90
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g2 91
-1.207078
dqv587@ubuntu:~/Documents/Code$
```

上图为 long long/double 版本, 可以看到仍然会发生溢出, 原因是 linux 下, long 和 long long 都是 8 字节编码, 编码范围一样, 故结果一样。

2. 提交初始的 long/double 版本的 g1.c 与 g2.c。

3. 提交最后优化后的程序 g.c

优化后的程序, 考虑了时间复杂度和储存值范围的问题, 选择了循环计算斐波那契数列和使用 double 类型储存 fib(n) 的值, 这样既不会发生溢出, 也不会发生精度丢失。

```
dqv587@ubuntu:~/Documents/Code$ ./g 100
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g 1474
0.618034
dqv587@ubuntu:~/Documents/Code$ ./g 1475
0.000000
dqv587@ubuntu:~/Documents/Code$
```

如上图运行结果, 可以看到 $n=1475$ 时才会发生溢出。

第 9 章 总结

9.1 请总结本次实验的收获

经过本次实验，我对计算机的组成有了大略的认识，熟悉了计算机各种配置查看、资源管理、运行进程等信息，更加了解了计算机的组成。

然后通过几个程序的编写，对计算机如何编码字符、整数、浮点数有了清晰的认知，体会到了其编码的优越性和局限性，同时明白了计算机编码何时会发生溢出，对以后编程时规避类似问题起到了指导作用。

9.2 请给出对本次实验内容的建议

虽然这次实验带领我们漫游了计算机系统，但是我仍然对计算机组成理解不甚明了，如果能对这部分再详细讲解一些就好了。

参考文献

- [1] Kernighan B W, Ritchie D M. The C Programming Language[M]. 2. Dennis Ritchie & Bell Labs, / June 2018.
- [2] Bryant R E, David R. O'Hallaron. Computer Systems a Programmer's Perspective[M]. 3rd. Carnegie Mellon University, 2016.