

# 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     2021112845

班        级     2103601

学        生     张智雄

指 导 老 师     郑贵滨

实 验 地 点     G.709

实 验 日 期     2023.3.5

哈尔滨工业大学计算学部

# 目 录

<b>第 1 章 实验基本信息</b> .....	<b>- 4 -</b>
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 -
<b>第 2 章 实验环境建立</b> .....	<b>- 5 -</b>
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 -
<b>第 3 章 WINDOWS 软硬件系统观察分析</b> .....	<b>- 6 -</b>
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) .....	- 6 -
3.3 隐藏分区与虚拟内存之分页文件查看 (2 分) .....	- 6 -
3.4 任务管理与资源监视 (2 分) .....	- 7 -
3.5 CPUZ 下的计算机硬件详细信息 (2 分) .....	- 7 -
<b>第 4 章 LINUX 软硬件系统观察分析</b> .....	<b>- 8 -</b>
4.1 计算机硬件详细信息 (3 分) .....	- 8 -
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) .....	- 9 -
4.4 LINUX 下网络系统信息 (2 分) .....	- 9 -
<b>第 5 章 LINUX 下的 SHOWBYTE 程序</b> .....	<b>- 10 -</b>
5.1 源程序提交 (8 分) .....	- 10 -

5.2 运行结果比较 (2 分) .....	- 10 -
<b>第 6 章 程序的生成 CPP、GCC、AS、LD.....</b>	<b>- 11 -</b>
6.1 请提交每步生成的文件 (10 分) .....	- 11 -
<b>第 7 章 计算机数据类型的本质 .....</b>	<b>- 11 -</b>
7.1 运行 SIZEOF.C 填表 (5 分) .....	- 11 -
7.2 请提交源程序文件 SIZEOF.C (5 分) .....	- 11 -
<b>第 8 章 程序运行分析.....</b>	<b>- 12 -</b>
8.1 SUM 的分析 (10 分) .....	- 12 -
8.2 FLOAT 的分析 (10 分) .....	- 12 -
8.3 程序优化 (20 分) .....	- 13 -
<b>第 9 章 总结 .....</b>	<b>- 15 -</b>
9.1 请总结本次实验的收获.....	- 15 -
9.2 请给出对本次实验内容的建议.....	- 15 -
<b>参考文献.....</b>	<b>- 15 -</b>

## 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, 掌握程序的生成步骤

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

X64 CPU; 2.30GHz; 16G RAM; 1.5THD disk

#### 1.2.2 软件环境

Windows11 64 位; VMware Workstation 17 Pro; Ubuntu 22.10

#### 1.2.3 开发工具

Visual Studio 2019 64 位; CodeBlocks 64 位; vim+gcc

### 1.3 实验预习

上实验课前, 必须认真预习实验指导 PPT

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。

初步使用计算机管理、设备管理器、磁盘管理器、任务管理器、资源监视器、性能监视器、系统信息、系统配置、组件服务查看计算机的软硬件信息。

在 Windows、Linux 下分别编写 hello.c, 显示“Hello 1200300101-学霸”(可换成学生自己信息)

试着编写 showbyte.c 显示 hello.c 的内容: 如书 P2 页, 每行 16 个字符, 上一行为字符, 下一行为其对应的 10 进制形式。

试着编写 sizeof.c 打印输出 C 语言每一个数据类型(含指针)占用空间, 并在 Windows、Linux 的 32/64 模式分别运行, 并比较运行结果。

## 第 2 章 实验环境建立

### 2.1 Windows 下 hello 程序的编辑与运行 (5 分)

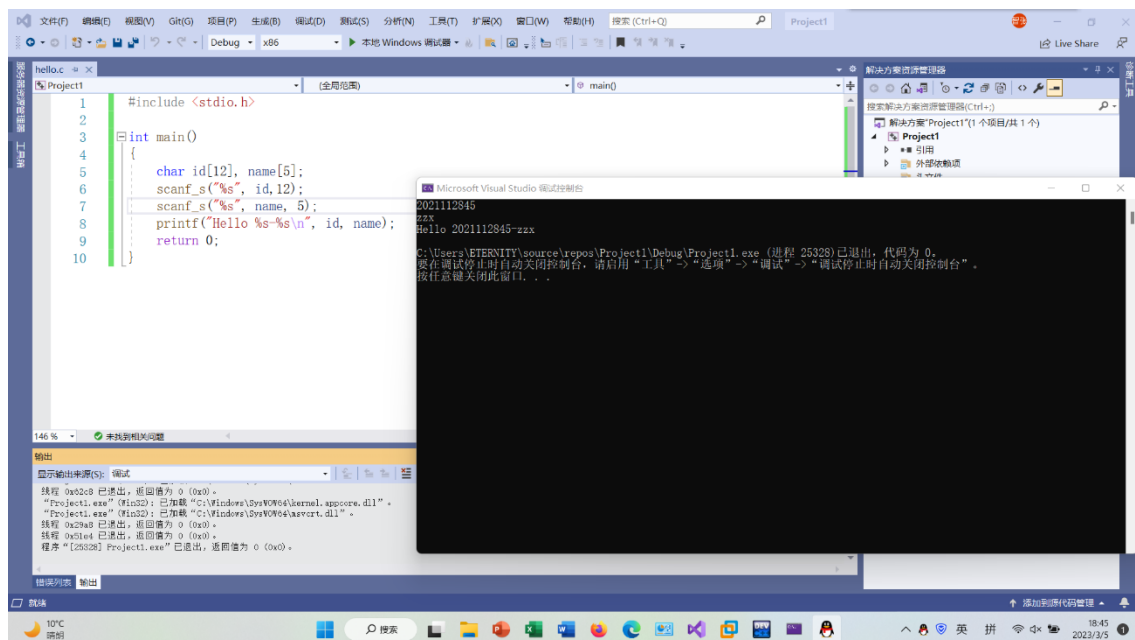


图 2-1 Windows 下 hello 运行截图

### 2.2 Linux 下 hello 程序的编辑与运行 (5 分)

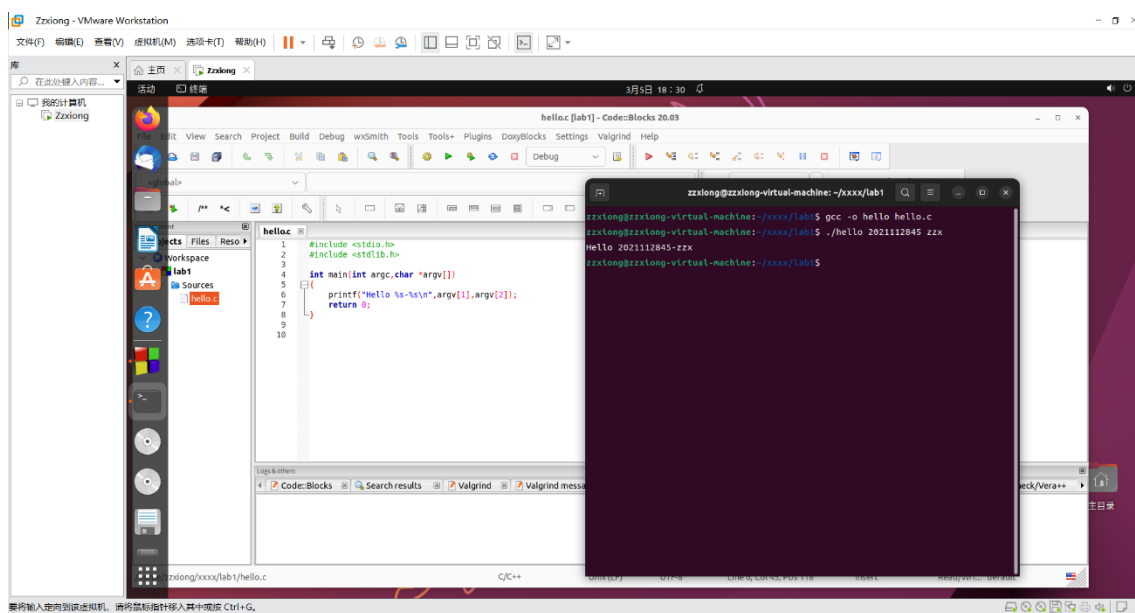


图 2-2 Linux 下 hello 运行截图

## 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）

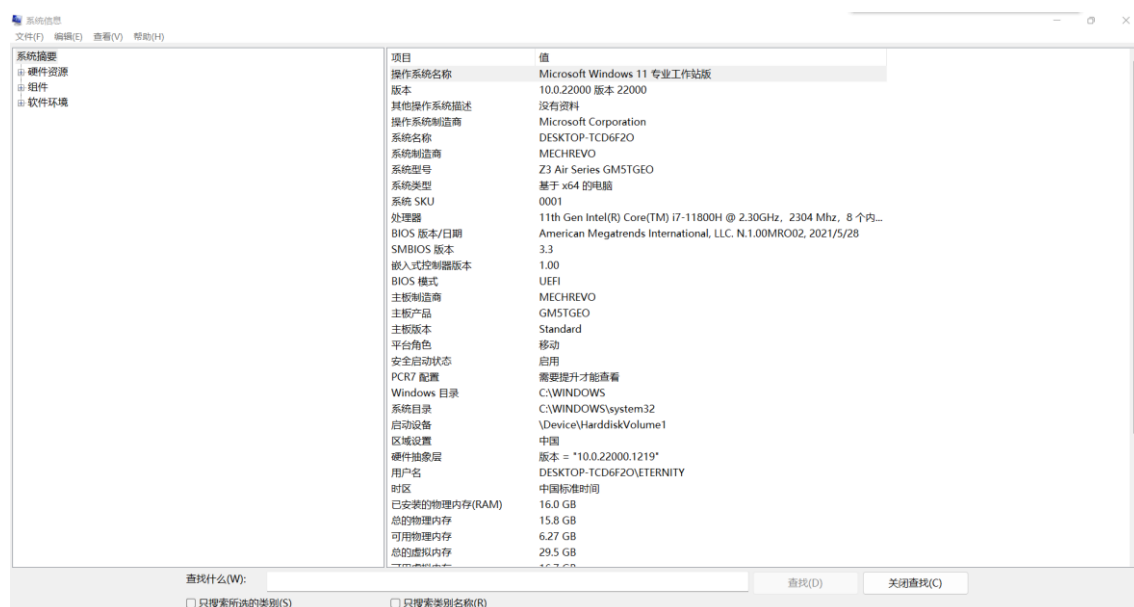


图 3-1 Windows 下计算机基本信息

### 3.2 设备管理器查看（2 分）

键盘：DESKTOP-TCDF6F2D/ 基于 ACPI x64 的电脑/Microsoft ACPI-Compliant System/PCI Express 根复合体/Intel(R) LPC Controller(HM570) -438B/PS/2 标准键盘

鼠标：DESKTOP-TCDF6F2D/ 基于 ACPI x64 的电脑/Microsoft ACPI-Compliant System/PCI Express 根复合体/ Intel(R) Serial IO I2C Host Controller -43E8/I2C HID 设备/HID-compliant mouse

### 3.3 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）：100MB、900MB

写出 pagefile.sys 的文件大小（Byte）：10,200,547,328Bytes

C 盘根目录下其他隐藏的系统文件名字为：\$Recycle.Bin、\$WinREAgent、Documents and Settings、System Volume Information、bootTel.dat、OneDriveTemp、ProgramData、Recovery、swapfile.sys、hiberfil.sys、bootmgr、BOOTNXT

### 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。

1.-; 系统中断; 延迟过程调用和中断服务例程

2.0; 系统空闲进程; 处理器空闲时间百分比

3.25596; msedge.exe; Microsoft Edge

### 3.5 CPUZ 下的计算机硬件详细信息 (2 分)

CPU 个数: 1 物理核数: 8 逻辑处理器个数: 16 L3 Cache 大小: 24MBytes



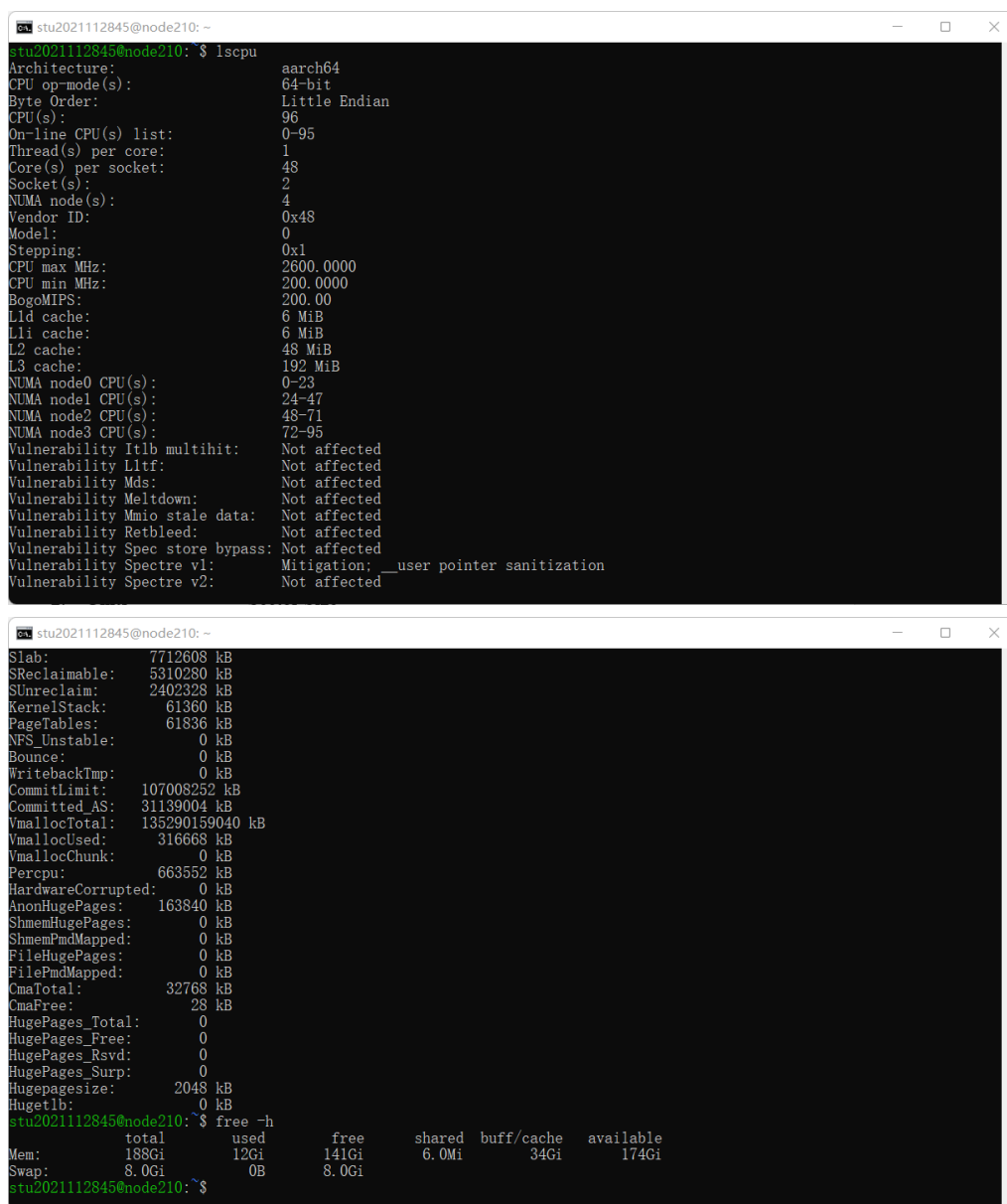
图 3-2 CPUZ 下 CPU 的基本信息

## 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: 188Gi Used: 12Gi Swap: 8.0Gi



```
stu2021112845@node210: ~  
stu2021112845@node210:~$ 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 Mmio stale data: Not affected  
Vulnerability Retbleed: Not affected  
Vulnerability Spec store bypass: Not affected  
Vulnerability Spectre v1: Mitigation; __user pointer sanitization  
Vulnerability Spectre v2: Not affected  
  
stu2021112845@node210:~$ free -h  
total        used        free        shared  buff/cache   available  
Mem:          188Gi       12Gi        141Gi         6.0Mi        34Gi        174Gi  
Swap:           8.0Gi          0B         8.0Gi
```

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: systemd
2. PID : “2”; Command: kthreadd

## 4.3 磁盘任务管理与资源监视 (3 分)

1. /dev/sda 设备的大小 1945.6 GB, 类型 AL15SEB120N
2. Units sectors of 1 \* 512 = 512 bytes  
Sector Size 512 bytes / 512 bytes(logical/physical)

## 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: 192.168.250.1

mac 地址: 02:42:d8:d2:43:5f

```
stu2021112845@node210:~$ ifconfig
br-d9571b531428: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.250.1 netmask 255.255.255.0 broadcast 192.168.250.255
    inet6 fe80::42:d8ff:fed2:435f prefixlen 64 scopeid 0x20<link>
    ether 02:42:d8:d2:43:5f txqueuelen 0 (Ethernet)
    RX packets 200037275 bytes 330552130366 (330.5 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 196651183 bytes 350627406220 (350.6 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

datapath: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1376
    inet6 fe80::5405:a4ff:feb3:4328 prefixlen 64 scopeid 0x20<link>
    ether 56:05:a4:b3:43:28 txqueuelen 1000 (Ethernet)
    RX packets 1872137 bytes 340224682 (340.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2985 bytes 264028 (264.0 KB)
    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::42:79ff:fe82:6ef4 prefixlen 64 scopeid 0x20<link>
    ether 02:42:79:82:6e:f4 txqueuelen 0 (Ethernet)
    RX packets 43585671 bytes 3737214031 (3.7 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43468406 bytes 289260080218 (289.2 GB)
    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>
    inet6 2406:280:1003:feee:6eeb:b6ff:fe15:932b prefixlen 64 scopeid 0x0<global>
    ether 6c:eb:b6:15:93:2b txqueuelen 1000 (Ethernet)
    RX packets 468813719 bytes 917868756228 (917.8 GB)
    RX errors 35 dropped 309 overruns 0 frame 0
    TX packets 380619972 bytes 893278426256 (893.2 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
```

图 4-2 Linux 下网络系统信息

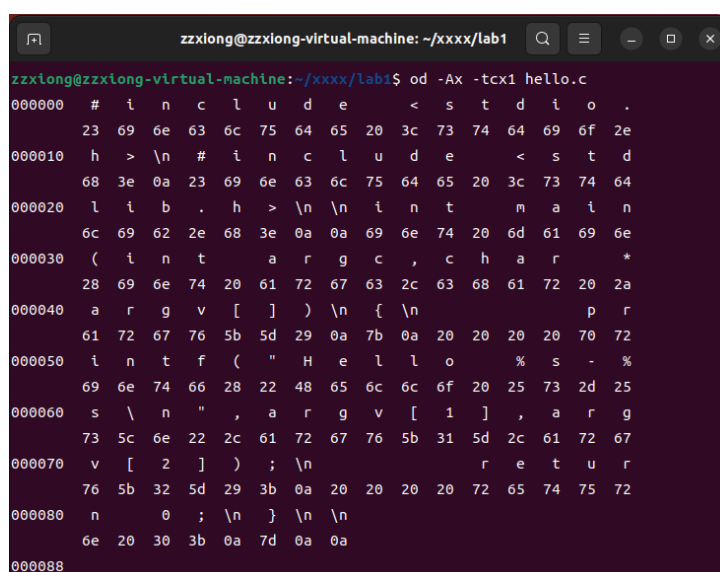
## 第 5 章 Linux 下的 showbyte 程序

### (10 分)

#### 5.1 源程序提交 (8 分)

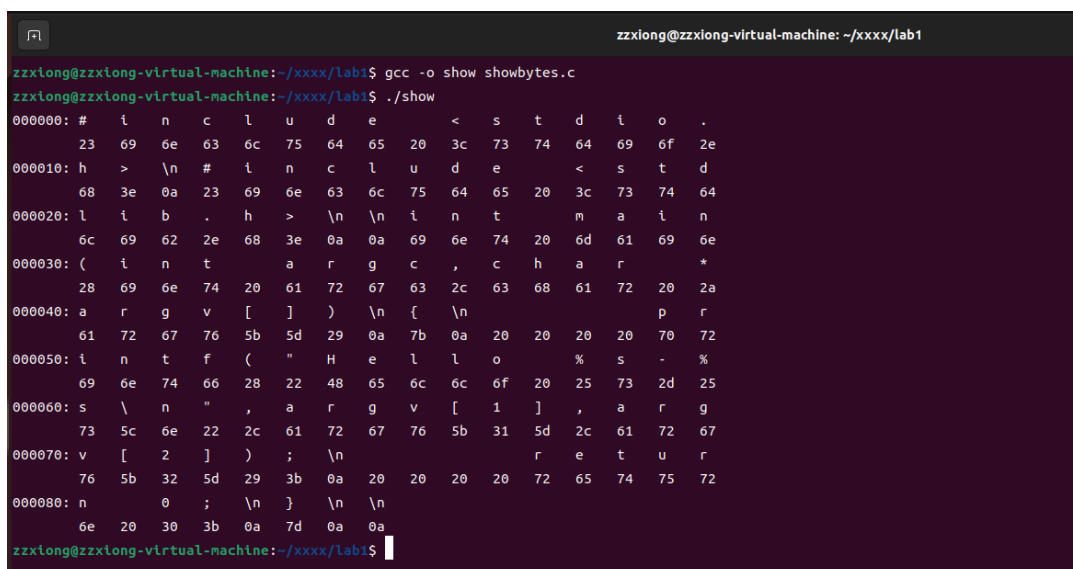
showbyte.c 与实验报告放在一个压缩包里

#### 5.2 运行结果比较 (2 分)



```
zzxiong@zzxiong-virtual-machine: ~/xxxx/lab1
zzxiong@zzxiong-virtual-machine:~/xxxx/lab1$ 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 # i n c l u d e < s t d
      68 3e 0a 23 69 6e 63 6c 75 64 65 20 3c 73 74 64
000020 l i b . h > \n \n i n t m a i n
      6c 69 62 2e 68 3e 0a 0a 69 6e 74 20 6d 61 69 6e
000030 ( i n t a r g c , c h a r *
      28 69 6e 74 20 61 72 67 63 2c 63 68 61 72 20 2a
000040 a r g v [ ] ) \n { \n p r
      61 72 67 76 5b 5d 29 0a 7b 0a 20 20 20 20 70 72
000050 i n t f ( " H e l l o % s - %
      69 6e 74 66 28 22 48 65 6c 6c 6f 20 25 73 2d 25
000060 s \ n " , a r g v [ 1 ] , a r g
      73 5c 6e 22 2c 61 72 67 76 5b 31 5d 2c 61 72 67
000070 v [ 2 ] ) ; \n r e t u r
      76 5b 32 5d 29 3b 0a 20 20 20 20 72 65 74 75 72
000080 n 0 ; \n } \n \n
      6e 20 30 3b 0a 7d 0a 0a
000088
```

图 5-1 OD 的输出结果



```
zzxiong@zzxiong-virtual-machine: ~/xxxx/lab1
zzxiong@zzxiong-virtual-machine:~/xxxx/lab1$ gcc -o show showbytes.c
zzxiong@zzxiong-virtual-machine:~/xxxx/lab1$ ./show
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 # i n c l u d e < s t d
      68 3e 0a 23 69 6e 63 6c 75 64 65 20 3c 73 74 64
000020: l i b . h > \n \n i n t m a i n
      6c 69 62 2e 68 3e 0a 0a 69 6e 74 20 6d 61 69 6e
000030: ( i n t a r g c , c h a r *
      28 69 6e 74 20 61 72 67 63 2c 63 68 61 72 20 2a
000040: a r g v [ ] ) \n { \n p r
      61 72 67 76 5b 5d 29 0a 7b 0a 20 20 20 20 70 72
000050: i n t f ( " H e l l o % s - %
      69 6e 74 66 28 22 48 65 6c 6c 6f 20 25 73 2d 25
000060: s \ n " , a r g v [ 1 ] , a r g
      73 5c 6e 22 2c 61 72 67 76 5b 31 5d 2c 61 72 67
000070: v [ 2 ] ) ; \n r e t u r
      76 5b 32 5d 29 3b 0a 20 20 20 20 72 65 74 75 72
000080: n 0 ; \n } \n \n
      6e 20 30 3b 0a 7d 0a 0a
zzxiong@zzxiong-virtual-machine:~/xxxx/lab1$
```

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	12	16	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。

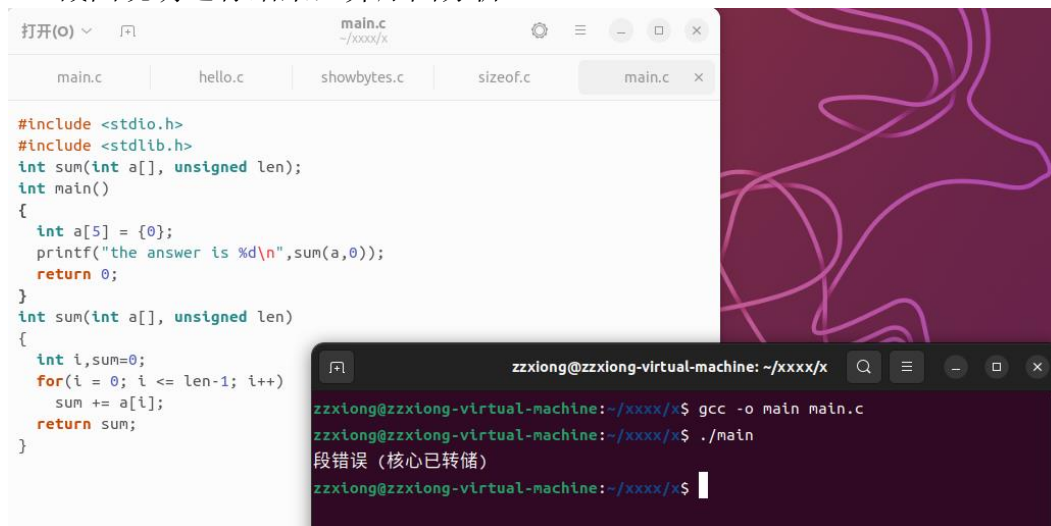


图 8-1 sum 的输出结果

结果：段错误（核心已转储）—> 内存访问越界。

原因：len 的数据类型为无符号数，而 i 是有符号数，当 i 和 len 进行大小比较时，会统一按以有符号数进行比较，当此时 len=0 时 len-1 的编码在有符号下的值为最大值 Umax，比较条件恒成立，从而产生死循环，数组访问时产生越界。

2.论述改进方法

改进方法：将 len 定义为有符号数，或者将循环控制条件改为  $i < len$ 。

### 8.2 float 的分析 (10 分)

1.运行结果截图，分析产生原因。

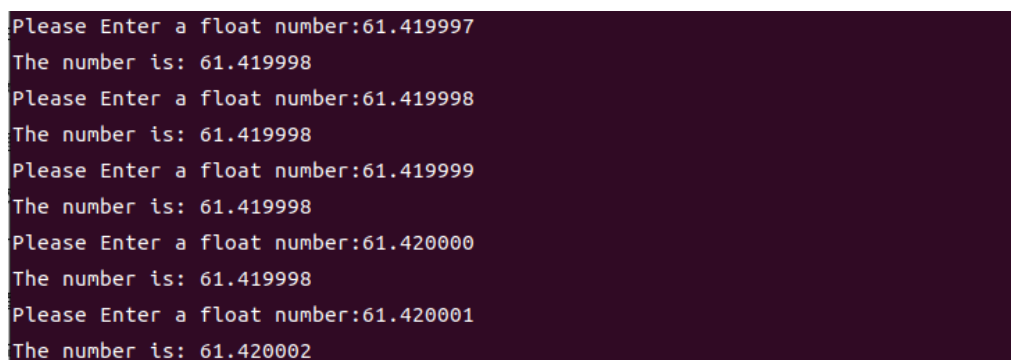


图 8-2 float 的输出结果 1

```
zxxiong@zxxiong-virtual-machine:~/xxx/x$ ./main
Please Enter a float number:10.186810
The number is: 10.186810
Please Enter a float number:10.186811
The number is: 10.186811
Please Enter a float number:10.186812
The number is: 10.186812
Please Enter a float number:10.186813
The number is: 10.186813
Please Enter a float number:10.186814
The number is: 10.186814
Please Enter a float number:10.186815
The number is: 10.186815
Please Enter a float number:0
The number is: 0.000000
```

图 8-3 float 的输出结果 2

原因: IEEE 754 规定下单精度浮点数的尾数仅有 23 位, 存在一定精度误差。

第一组数据: 二进制下, 第一组数据若要较为精确地表示所需位数远远大于 23 位, 而实际存储中尾数后面的数都会被截断并向偶数进行舍入, 因而有的数据经过舍入后结果发生一定程度的偏移。

第二组数据: 而第二组数据在 23 位尾数表示下已经较为精确, 在舍入时并不会产生数据的偏移, 因而最终的运行结果和数据输入的内容相同。

## 2. 论述编程中浮点数比较、汇总统计等应如何正确编程。

浮点数在计算机存储时大多是近似值, 无法精确表示准确数值。在比较中对阶、计算操作都有可能造成最终结果的误差。

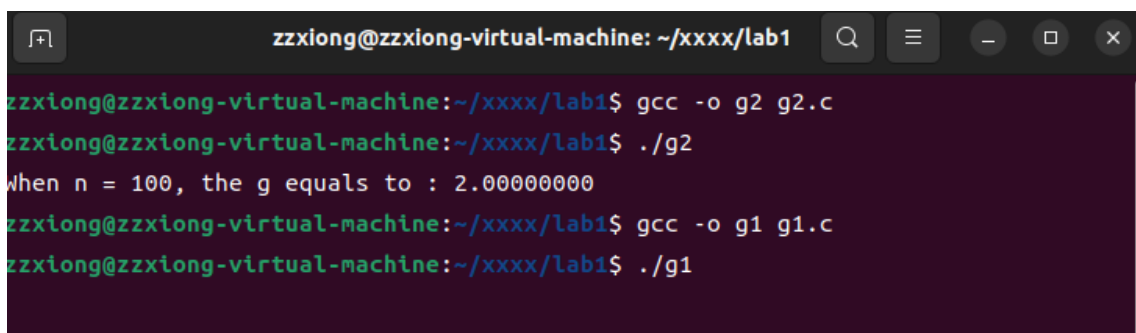
一些精度要求较高的工作不建议使用单精度浮点数的数据类型, 可以考虑选择使用 double 或 long double 等更高精度的浮点数类型或者用数组按位表示进行实际的运算。

## 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。

```
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ gcc -o g2 g2.c
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ ./g2
When n = 100, the g equals to : 0.61803399
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ gcc -o g1 g1.c
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ ./g1
```

图 8-4 黄金分割数的输出结果 (f 为 double 类型)



```
zxxiong@zxxiong-virtual-machine: ~/xxx/lab1
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ gcc -o g2 g2.c
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ ./g2
When n = 100, the g equals to : 2.00000000
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ gcc -o g1 g1.c
zxxiong@zxxiong-virtual-machine:~/xxx/lab1$ ./g1
```

图 8-4 黄金分割数的输出结果 (f 为 long long 类型)

结果分析:

对于使用递归的 g1.c, 由于递归重复调用的次数较多, 代码效率较低, 当 n 取较大的时候导致栈溢出。而 g2.c 利用循环采用动态规划的思想, 运算速度较快。

但同时通过修改 n、f、g 的数据类型, 发现 f 的数据类型对结果影响很大, 通过分析, 认为在斐波那契数列求解过程中数值大小超过了 long long 数据类型的最大取值范围( $-2^{32} + 1, 2^{32} - 1$ )导致溢出。

优化方法:

- 1). 将 f 的数据类型直接修改为可表示范围更广的 double 类型, 但由此每一次数的表示过程中都会产生一定的误差, 最终误差的叠加可能会导致较大偏移。
- 2). 利用数组表示 f, 最后将数组转换为 double 类型的大整数进行除法运算, 可以避免 double 数据类型在累加过程中产生的误差, 但仍有一定误差。
- 3). 利用数组对 f 进行二进制表示其原码, 加法运算和最终的除法运算均采用原码的运算规则, 这样可以基本避免运算过程中转换表示的误差, 最终结果保留位数仅由商的精度决定, 但耗费空间较大。

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

注: 最终提交的程序 g.c 采用的是上述第二种优化方法, 第一种直接在源程序 g2.c 或 g1.c 上修改即可。

## 第 9 章 总结

### 9.1 请总结本次实验的收获

1. 通过对计算机软硬件系统的观察，重新认识了现代计算机的结构。
2. 初步熟悉了 linux 环境下的简单编程以及指令控制。
3. 认识了程序从编写、编译和执行的步骤以及中间过程。
4. 初步熟悉了 linux 环境下使用 gdb 调试程序的过程。
5. 从编译器的角度认识了常见的数据类型及储存过程中的区别，能发现并修正、优化一些简单的程序错误或漏洞。

### 9.2 请给出对本次实验内容的建议

1. 实验 PPT 在一些实验内容上的讲解不是很清晰，如 showbytes.c 的具体要求、float 的分析中的数据类型的定义要求等。
2. 可以适当增加 linux 环境下指令的讲解以及实验，很多指令只是简单的复制粘贴，不能够理解其语句及各参数的含义。
3. 可以适当增加 vim 的使用讲解或者 gdb 调试的过程说明等。

注：本章为酌情加分项。

## 参考文献

[1] RANDALE.BRYANT, DAVIDR.O'HALLARON. 深入理解计算机系统[M]. 机械工业出版社, 2011.