哈爾濱工業大學

实验报告

实 验（一）

题　　目　计算机系统漫游

学　　号　2022110864

班　　级　22WL022

学　　生　宋雨桐

指 导 教 师　刘宏伟

实 验 地 点　G715

实 验 日 期　2024 年 3 月 20 日

哈尔滨工业大学计算学部

目 录

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

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 位以上。

1.2.3 开发工具

Visual Studio 2010 64 位以上；CodeBlocks 64 位；vi/vim/gedit+gcc。

1.3 实验预习

1. 填写上实验课前，必须认真预习实验指导 PPT。
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. 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。
4. 在 Windows、Linux 下分别编写 hello.c，显示“Hello-2022110864 宋雨桐”
5. 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 16 进制形式。

6. 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）

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

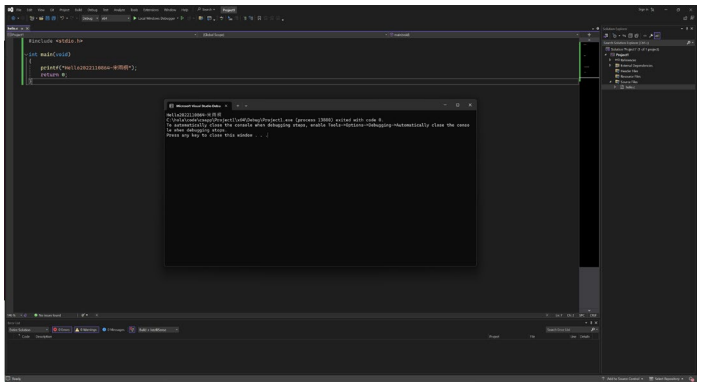

图 2-1 Windows 下 hello 运行截图

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。

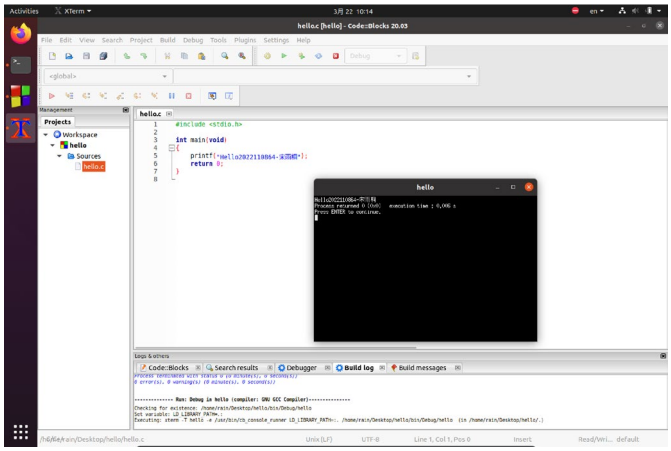

图 2-2 Linux 下 hello 运行截图

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

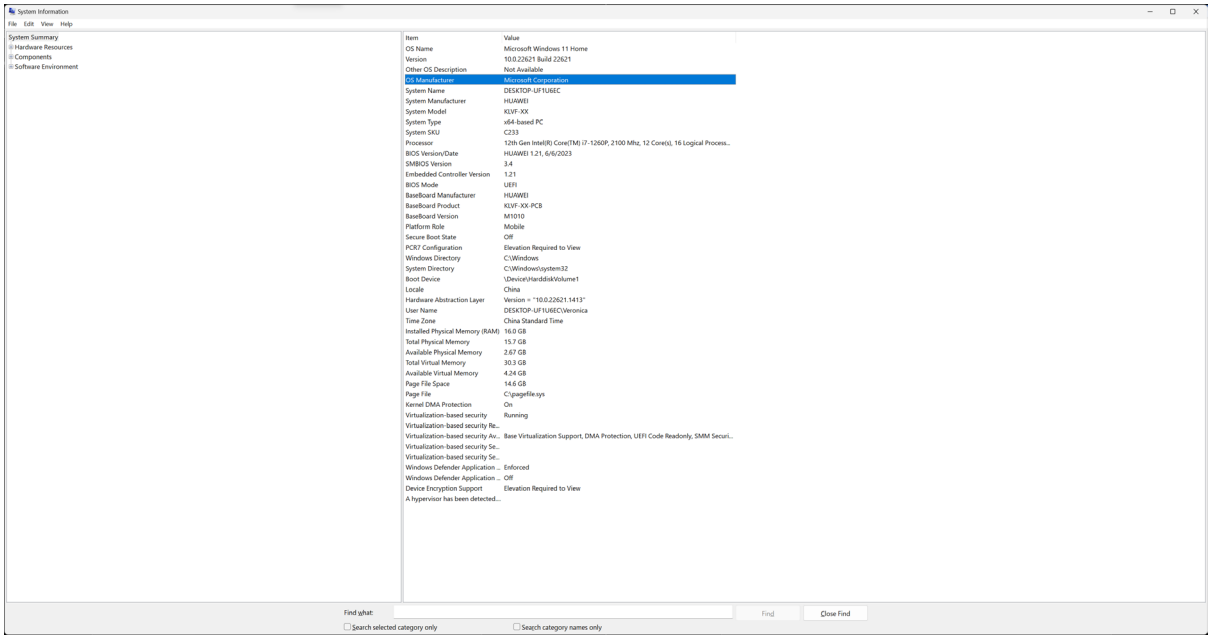

图 3-1 Windows 下计算机基本信息

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。

1. 键盘：DESKTOP-UF1U6EC//keyboards//Standard PS/2 Keyboard
2. 鼠标：DESKTOP-UF1U6EC/Mice and other pointing devices//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）：100, 719

写出 pagefile.sys 的文件大小（Byte）：11,274,289,152

C 根目录下其他隐藏的系统文件名字为：DumpStack.log.tmp hiberfil.sys pagefile.sys swapfile.sys

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. System interrupts: PID=-; Description: Deferred procedure calls and interrupt service routines

2. System Idle Process: PID=0; Description=Percentage of time the processor is idle

3.chrome.exe: PID=27600; Description=Google Chrome

3.5 CPUZ 下的计算机硬件详细信息（2 分）

CPU 个数：1 物理核数：12 逻辑处理器个数：16 L3 Cache 大小：18MB

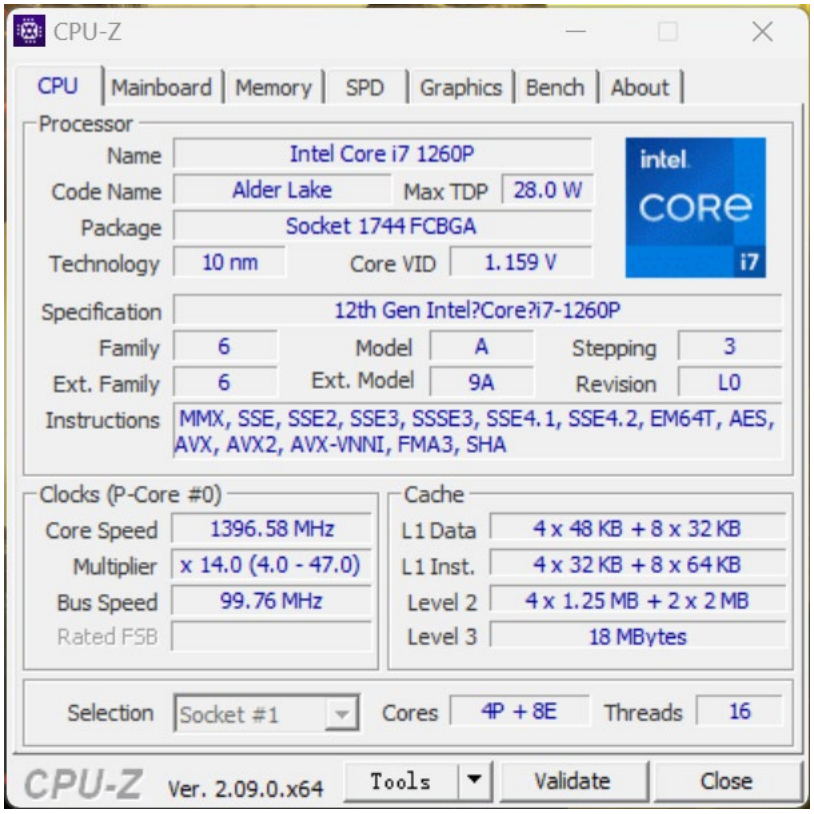

图 3-2 CPUZ 下 CPU 的基本信息

第 4 章 Linux 软硬件系统观察分析

（泰山服务器—个人电脑）

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：__9092184 kB__ Used：__846568kB__ Swap：__1190340kB__

```
CPU(s):                          2
On-line CPU(s) list:             0,1
Thread(s) per core:              1
```

```
processor       : 1
```

```
cpu cores       : 2
```

```
rain@ubuntu-20:~$ free
               total        used        free      shared  buff/cache   available
Mem:         9092184      846568     5787260       15020     2458356     7946788
Swap:        1190340           0     1190340
```

图 4-1 Linux 下计算机硬件详细信息

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=2; COMMAND=[kthreadd]

4.3 磁盘任务管理与资源监视（3 分）

/dev/sda 设备的大小__25__GB，类型__SSD__

Units:__512Bytes__ Sector Size:__512Bytes__

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：__10.0.2.15__

mac 地址：__08:00:27:63:79:a3_____

```
rain@ubuntu-20:~/csapp/test$ ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::fabc:49cc:b098:1c5d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:63:79:a3  txqueuelen 1000  (Ethernet)
        RX packets 15642  bytes 19996662 (19.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3765  bytes 422848 (422.8 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  (Local Loopback)
        RX packets 785  bytes 81627 (81.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 785  bytes 81627 (81.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 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）

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

```
rain@ubuntu-20:/media/sf_csapp$ 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   >  \r  \n  \r  \n   i   n   t       m   a   i   n   (   v
         68  3e  0d  0a  0d  0a  69  6e  74  20  6d  61  69  6e  28  76
000020   o   i   d   )  \r  \n   {  \r  \n                   p   r   i
         6f  69  64  29  0d  0a  7b  0d  0a  20  20  20  20  70  72  69
000030   n   t   f   (   "   H   e   l   l   o   2   0   2   2   1   1
         6e  74  66  28  22  48  65  6c  6c  6f  32  30  32  32  31  31
000040   0   8   6   4   - 345 256 213 351 233 250 346 241 220   "   )
         30  38  36  34  2d  e5  ae  8b  e9  9b  a8  e6  a1  90  22  29
000050   ;  \r  \n                   r   e   t   u   r   n       0   ;
         3b  0d  0a  20  20  20  20  72  65  74  75  72  6e  20  30  3b
000060  \r  \n   }  \r  \n
         0d  0a  7d  0d  0a
000065
```

图 5-1 OD 的输出结果

```
rain@ubuntu-20:/media/sf_csapp$ gcc -o showbyte showbyte.c
rain@ubuntu-20:/media/sf_csapp$ ./showbyte
#   i   n   c   l   u   d   e   <sp><   s   t   d   i   o   .
23  69  6e  63  6c  75  64  65  20  3c  73  74  64  69  6f  2e
h   >  \r  \n  \r  \n  i   n   t   <sp>m   a   i   n   (   v
68  3e  0d  0a  0d  0a  69  6e  74  20  6d  61  69  6e  28  76
o   i   d   )  \r  \n  {  \r  \n  <sp><sp><sp><sp>p   r   i
6f  69  64  29  0d  0a  7b  0d  0a  20  20  20  20  70  72  69
n   t   f   (   "   H   e   l   l   o   2   0   2   2   1   1
6e  74  66  28  22  48  65  6c  6c  6f  32  30  32  32  31  31
0   8   6   4   -   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   "   )
30  38  36  34  2d  e5  ae  8b  e9  9b  a8  e6  a1  90  22  29
;  \r  \n  <sp><sp><sp><sp>r   e   t   u   r   n   <sp>0   ;
3b  0d  0a  20  20  20  20  72  65  74  75  72  6e  20  30  3b
\r  \n  }  \r  \n
0d  0a  7d  0d  0a
```

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)

```c
#include <stdio.h>

int main(void)
{
    printf("Hello2022110864-宋雨桐");
    return 0;
}
```

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）

```
1.  #include <inttypes.h>
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.
5.  int main(void)
6.  {
7.      printf("sizeof(char) = %d\n", sizeof(char));
8.      printf("sizeof(short) = %d\n", sizeof(short));
9.      printf("sizeof(int) = %d\n", sizeof(int));
10.     printf("sizeof(long) = %d\n", sizeof(long));
11.     printf("sizeof(long long) = %d\n", sizeof(long long));
12.     printf("sizeof(int32_t) = %d\n", sizeof(int32_t));
13.     printf("sizeof(int64_t) = %d\n", sizeof(int64_t));
14.     printf("sizeof(float) = %d\n", sizeof(float));
15.     printf("sizeof(double) = %d\n", sizeof(double));
16.     printf("sizeof(long double) = %d\n", sizeof(long double));
17.     printf("sizeof(char *) = %d\n", sizeof(char *));
18.
19.     return EXIT_SUCCESS;
20. }
```

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。

```
rain@ubuntu-20:/media/sf_csapp$ gcc -o sum sum.c
rain@ubuntu-20:/media/sf_csapp$ ./sum
Segmentation fault (core dumped)
```

由于 len 的类型为 unsigned int，所以 len – 1 经过隐式类型转换为 unsigned int，得到的结果为一个很大的正数(UINT_MAX)。由于 main 函数中创建的数组大小小于这个很大的正数，导致指针**非法访问**到了数组以外的内存，出现了**分段错误**。

2.论述改进方法

将 len 的类型改为 signed int，同时在程序中应尽量**避免使用无符号数**。

8.2 float 的分析（10 分）

1.运行结果截图，分析产生原因。

```
rain@ubuntu-20:/media/sf_csapp$ gcc -o float float.c
rain@ubuntu-20:/media/sf_csapp$ ./float
Please input a float:61.419997
Value of this float is 61.419998
Please input a float:61.419998
Value of this float is 61.419998
Please input a float:61.419999
Value of this float is 61.419998
Please input a float:61.420000
Value of this float is 61.419998
Please input a float:61.420001
Value of this float is 61.420002
Please input a float:0
Value of this float is 0.000000
rain@ubuntu-20:/media/sf_csapp$
```

```
rain@ubuntu-20:/media/sf_csapp$ ./float
Please input a float:10.186810
Value of this float is 10.186810
Please input a float:10.186811
Value of this float is 10.186811
Please input a float:10.186812
Value of this float is 10.186812
Please input a float:10.186813
Value of this float is 10.186813
Please input a float:10.186814
Value of this float is 10.186814
Please input a float:10.186815
Value of this float is 10.186815
Please input a float:0
Value of this float is 0.000000
rain@ubuntu-20:/media/sf_csapp$
```

由于计算机存储本质上是离散的，所以不可能表示所有的实数，只能用近似的方法表示。这导致了不同的实数在计算机存储中的浮点数相同。

2. 论述编程中浮点数比较、汇总统计等应如何正确编程。

浮点数比较时，不可使用!=或==，而应该规定一个精确度，如

```
1.  if (fabs(price - p) < 0.000001)
```

汇总统计时，根据精度要求可选择 float 和 double 使用。若要求不损失精度，

可使用 Kahan Summation 算法，或者将数据扩大为若干倍转换成整数计算。

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。

```
rain@ubuntu-20:/media/sf_csapp$ gcc -o gc1 gc1.c
rain@ubuntu-20:/media/sf_csapp$ ./gc1
```

原因为，该递归不是尾递归，栈中储存了大量计算未完成的函数，导致栈的存储饱和，无法进行计算。

```
rain@ubuntu-20:/media/sf_csapp$ gcc -o gc2 gc2.c
rain@ubuntu-20:/media/sf_csapp$ ./gc2
2.877098
```

而在循环中，由于 fib(100)以上的大小均超过了 ULLONG_MAX，导致向上溢出，计算结果不正确。

2. 提交初始的 long/double 版本的 g1.c 与 g2.c。

```c
#include <stdio.h>
#include <stdlib.h>

long fib(int n);

int main(void)
{
    printf("%.8f", (double) fib(100) / (double) fib(101));

    return EXIT_SUCCESS;
}

long fib(int n)
{
    if (n == 1)
    {
        return 1;
    }
    if(n == 2)
    {
        return 1;
    }

    return fib(n - 1) + fib(n - 2);
}
```

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

```
#include <stdlib.h>

long fib(int n);

int main(void)
{
    printf("%lf\n", (double) fib(100) / (double) fib(101));
    return EXIT_SUCCESS;
}

long fib(int n)
{
    long a = 1, b = 1, result = 0;
    for (int i = 3; i <= n; i++)
    {
        result = a + b;
        a = b;
        b = result;
    }
    return result;
}
```

3. 提交最后优化后的程序 g.c

```
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>

#define SIZE 1000
#define PRECISE 8
#define MAXIMUM 1E17

typedef int64_t int64;
typedef struct num
{
    int64 digits[SIZE];
    int max_digits;
} num;

void fib(int n, num *a, num *b, num *result);
void num_add(num *result, num *a, num *b);
void num  init(num *a);
void num  printf(num *result);
void num  set(num *des, num *src);
void num  mul(num *mul  result, num *a, int times);
int num  sub(num *result, num *a, num *b);
void num  div(num *a, num *b);

int main(void)
{
    num *a = malloc(sizeof(num));
    num *b = malloc(sizeof(num));
    num *result  a = malloc(sizeof(num));
    num *result  b = malloc(sizeof(num));
    num *tmp = malloc(sizeof(num));

    num  init(a);
    num  init(b);
    num  init(result  a);
    num_init(result_b);
```

```
37.     num_init(tmp);
38.     a -> digits[0] = 1;
39.     b -> digits[0] = 1;
40.
41.     fib(100, a, b, result_a);
42.
43.     num_init(a);
44.     num_init(b);
45.     a -> digits[0] = 1;
46.     b -> digits[0] = 1;
47.     fib(101, a, b, result_b);
48.
49.     num_div(result_a, result_b);
50.
51.     free(a);
52.     free(b);
53.     free(result_a);
54.     free(result_b);
55.     free(tmp);
56.
57.     return EXIT_SUCCESS;
58. }
59.
60. void fib(int n, num *a, num *b, num * result)
61. {
62.     for (int i = 3; i <= n; i++)
63.     {
64.         num_add(result, a, b);
65.         num_set(a, b);
66.         num_set(b, result);
67.     }
68.     return;
69. }
70.
71. void num_init(num *a)
72. {
73.     a -> max_digits = 1;
74.     for (int i = 0; i < SIZE; i++)
75.     {
76.         a -> digits[i] = 0;
77.     }
78.
79.     return;
80. }
81.
82. void num_add(num *result, num *a, num *b)
83. {
84.     int max  digits = (a -> max  digits > b -> max  digits) ? a -> max  digits : b -> max  digits;
85.
86.     result -> max  digits = max  digits;
87.
88.     int carry = 0;
89.     for (int i = 0; i < max  digits; i++)
90.     {
91.         if (a -> digits[i] + b -> digits[i] + carry >= MAXIMUM)
92.         {
93.             result -> digits[i] = a -> digits[i] + b -> digits[i] + carry - (MAXIMUM - 1);
94.             carry = 1;
95.         }
96.         else
97.         {
98.             result -> digits[i] = a -> digits[i] + b -> digits[i] + carry;
```

```
99.             carry = 0;
100.            }
101.
102.        }
103.
104.        if (carry != 0)
105.        {
106.            result -> digits[max_digits] = 1;
107.            (result -> max_digits)++;
108.        }
109.
110.        return;
111.    }
112.
113.    void num_printf(num *result)
114.    {
115.        for (int i = result -> max_digits - 1; i >= 0; i--)
116.            printf("%" PRIu64 "", result -> digits[i]);
117.        printf("\n");
118.
119.        return;
120.    }
121.
122.    void num_set(num *des, num *src)
123.    {
124.        des -> max_digits = src -> max_digits;
125.        for (int i = 0; i < src -> max_digits; i++)
126.            des -> digits[i] = src -> digits[i];
127.
128.        return;
129.    }
130.
131.    void num_div(num *a, num *b)
132.    {
133.        printf("0.");
134.        num *quotient = malloc(sizeof(num));
135.        num *mod = malloc(sizeof(num));
136.        num *mul_result = malloc(sizeof(num));
137.        num *div_result = malloc(sizeof(num));
138.        num *sub_result = malloc(sizeof(num));
139.        num_init(quotient);
140.        num_init(mod);
141.        num_init(mul_result);
142.        num_init(div_result);
143.        num_init(sub_result);
144.
145.        num  set(mod, a);
146.
147.        int count = 0;
148.        int count  2 = 0;
149.        for (int i = 0; i <= PRECISE; i++)
150.        {
151.            count = 0;
152.            num  set(quotient, mod);
153.
154.            num  mul(mul  result, quotient, 10);
155.            num  set(quotient, mul  result);
156.
157.            while (num  sub(sub  result, quotient, b) == 1)
158.            {
159.                num  mul(mul  result, quotient, 10);
160.                num_set(quotient, mul_result);
```

```
161.             if (i != 0)
162.             {
163.                 printf("0");
164.                 i++;
165.             }
166.         }
167.
168.         num *quotient_tmp = malloc(sizeof(num));
169.         num_init(quotient_tmp);
170.         num_set(quotient_tmp, quotient);
171.
172.         while (num_sub(sub_result, quotient_tmp, b) == 0)
173.         {
174.             count++;
175.             num_set(quotient_tmp, sub_result);
176.         }
177.         if (i == PRECISE - 1)
178.         {
179.             count_2 = count;
180.             num_set(mod, quotient_tmp);
181.
182.             free(quotient_tmp);
183.
184.             continue;
185.         }
186.         if (i == PRECISE)
187.         {
188.             if (count >= 5)
189.             {
190.                 printf("%d", count_2 + 1);
191.             }
192.             free(quotient_tmp);
193.             break;
194.         }
195.         printf("%d", count);
196.
197.         num_set(mod, quotient_tmp);
198.
199.         free(quotient_tmp);
200.     }
201.     printf("\n");
202.
203.     free(quotient);
204.     free(mod);
205.     free(mul_result);
206.     free(div_result);
207.     free(sub  result);
208.
209.     return;
210. }
211.
212. void num  mul(num *mul  result, num *a, int times)
213. {
214.     num *mul  result  sum = malloc(sizeof(num));
215.     num  init(mul  result  sum);
216.
217.     for (int i = 0; i < times; i++)
218.     {
219.         num  add(mul  result, a, mul  result  sum);
220.         num  set(mul  result  sum, mul  result);
221.     }
222.
```

```
223.        free(mul_result_sum);
224.
225.        return;
226.    }
227.
228.    int num_sub(num *result, num *a_org, num *b_org)
229.    {
230.        int max_digits;
231.        int flag = 0;
232.        num *a = malloc(sizeof(num));
233.        num *b = malloc(sizeof(num));
234.        num_init(a);
235.        num_init(b);
236.        num_set(a, a_org);
237.        num_set(b, b_org);
238.        if (a -> max_digits == b -> max_digits)
239.        {
240.            max_digits = a -> max_digits;
241.            for (int i = max_digits - 1; i >= 0; i--)
242.            {
243.                if(a -> digits[i] > b -> digits[i])
244.                    break;
245.                else if(a -> digits[i] < b -> digits[i])
246.                {
247.                    num *tmp = malloc(sizeof(num));
248.                    num_init(tmp);
249.                    num_set(tmp, a);
250.                    num_set(a, b);
251.                    num_set(b, tmp);
252.                    free(tmp);
253.                    flag = 1;
254.                    break;
255.                }
256.            }
257.        }
258.        else if (a -> max_digits > b -> max_digits)
259.        {
260.            max_digits = a -> max_digits;
261.        }
262.        else
263.        {
264.            max_digits = b -> max_digits;
265.            num *tmp = malloc(sizeof(num));
266.            num_init(tmp);
267.            num_set(tmp, a);
268.            num_set(a, b);
269.            num  set(b, tmp);
270.            free(tmp);
271.            flag = 1;
272.
273.        }
274.
275.        result -> max  digits = max  digits;
276.
277.        int carry = 0;
278.        for (int i = 0; i < max  digits; i++)
279.        {
280.            if (a -> digits[i] - b -> digits[i] - carry < 0)
281.            {
282.                result -> digits[i] = a -> digits[i] - b -> digits[i] - carry + MAXIMUM;
283.                carry = 1;
284.            }
```

```
285.            else
286.            {
287.                result -> digits[i] = a -> digits[i] - b -> digits[i] - carry;
288.                carry = 0;
289.            }
290.        }
291.
292.        free(a);
293.        free(b);
294.        return flag;
295.    }
```

第 9 章 总结

9.1 请总结本次实验的收获

1. 硬件信息查询

在本次实验中，我深刻认识到了使用一些硬件检测软件（CPUZ，FansControl）的必要性，厘清了处理器与 CPU 的关系，以及如何将 csapp 这本书上的诸多术语与实际硬件一一对应。

2. 软件使用

本次实验，我重温了虚拟机的使用和系统的安装，发现了 VirtualBox 相对于 Vmware 的优势，同时研究了 Linux 系统下 vim 的键盘映射。

3. 可存任意大的数

在解决斐波那契数的存储问题的时候，我先后产生三个想法：

(1) 判断是否溢出，溢出则右移 5 位。

(2) 使用第三方库 GMP 来存储任意大的斐波那契数及黄金分割比数。

(3) 借鉴 GMP，用**数组存储任意大的斐波那契数**，并求出任意精度的黄金分割比。

上面的三个想法均已用递归和循环实现，命名为: gc1_shift.c, gc1_gmp.c, gc1_arr.c, gc2_shift.c, gc2_gmp.c, gc2_arr.c。

在解决递归函数栈开销过大的问题时，我引入了**尾递归**。虽然斐波那契数的尾递归与循环的形式相像，但是我仍觉这不失为一个好办法，尤其不同于一般递归的形式。

9.2 请给出对本次实验内容的建议

关于实验内容并无建议，我在自己完成实验的同时，也帮助了不少同学。

关于实验报告的一点薄见为，由于 Word 排版太过麻烦，经常出现“写作十分钟，排版半小时”的困扰，希望老师们能够同时给出 Word、Latex、Markdown 的模板，同学们可根据自身喜好，任选其一。虽然这会大大增加老师们的工作量，但是同学们能更投入到实验内容的学习，同时最后上传的报告也更赏心悦目。

最后，感谢各位老师的辛勤付出，此致敬礼！

参考文献

[1] Bryant, R. E., & O'Hallaron, D. R. (2016). Computer Systems: A programmer's perspective. Pearson.

[2] The GNU MP Bignum library. The GNU MP Bignum Library. (n.d.). https://gmplib.org/

[3] Introductory C programming. Coursera. (n.d.). https://www.coursera.org/specializations/c-programming