

## 实验二：C 编程环境

### 实验目的

1. 熟悉 Linux 下 C 程序设计的环境；
2. 对系统调用有初步了解。

### 实验时间

3 学时

### 实验内容

#### 1. Linux 下 C 语言程序的开发过程

a、在用户主目录下用 vi 编辑 C 语言源程序（源程序已附后），如：\$vi hello.c。

b、用 gcc 编译 C 语言源程序：\$gcc ./hello.c -o example

这里 gcc 是 Linux 下的 C 语言程序编译器（GNU C Compiler），./hello.c 表示待编译的源文件是当前工作目录下的 hello.c，-o example 表示编译后产生的目标代码文件名为 example。

c、若编译不正确，则进入 vi 修改源程序，否则，运行目标代码：\$./example 。

注意：

a、这只是 gcc 最基本的用法，其他常用选项有：-c, -S, -O, -O2, -g 等。

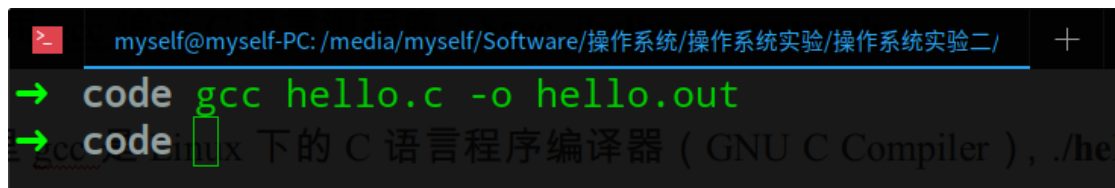
b、调试程序可以用 gdb（GNU debugger）。

答：a、使用 vi 编辑 C 语言源程序：



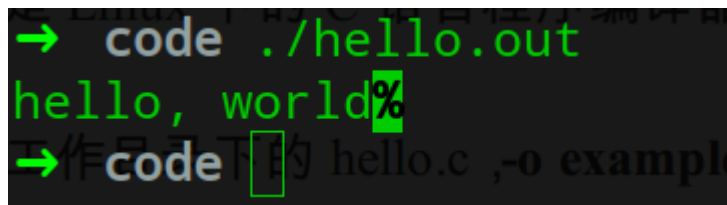
```
vi hellc × +
#include<stdio.h>
int main(void)
{
    printf("hello, world");
    return 0;
}
```

b、用 gcc 编译 C 语言源程序：



```
myself@myself-PC: /media/myself/Software/操作系统/操作系统实验/操作系统实验二/
→ code gcc hello.c -o hello.out
→ code gcc 下的 C 语言程序编译器 ( GNU C Compiler ), ./he
```

c、运行目标代码：



```
→ code ./hello.out
hello, world%
→ code 下的 hello.c , -o example
```

需要注意的是输出结果中的%是我使用的 zsh 用于标记输出内容结尾没有换行的符号，不是程序的输出内容。

2. 编辑、调试下面 c 语言程序，说明该程序的功能。

```
#include <stdio.h>
main() {
    int n,a[200],carry,temp,i,j,digit = 1;
    printf("Please input n:");
        scanf("%d",&n);
        a[0] = 1;
        for( i = 2; i <= n; ++i) {
            for( j = 1, carry = 0; j <= digit; ++j) {
                temp = a[j-1] * i + carry;
                a[j-1] = temp % 10;
                carry = temp / 10; }
            while(carry) { a[++digit-1] = carry % 10; carry /= 10;
                } }
        printf("Result is:\n%d ! = ",n);
        for( i = digit; i >=1; --i) { printf("%d",a[i-1]); } printf("\n"); }
```

答：使用 vi 编辑程序如下：

```

1 #include <stdio.h>
2 #include <stdio.h>
3 main()
4 {
5     int n, a[200], carry, temp, i, j, digit = 1;
6     printf("Please input n:");
7     scanf("%d", &n);
8     a[0] = 1;
9     for(i = 2; i <= n; ++i) {
10         for(j = 1, carry = 0; j <= digit; ++j) {
11             temp = a[j - 1] * i + carry;
12             a[j - 1] = temp % 10;
13             carry = temp / 10;
14         }
15         while(carry) {
16             a[++digit - 1] = carry % 10;
17             carry /= 10;
18         }
19     }
20     printf("Result is:\n%d != ", n);
21     for(i = digit; i >= 1; --i) { printf("%d", a[i - 1]); }
22     printf("\n");
23 }

```

:set nu 23,1 全部

使用-o 参数指定编译后的文件名，-g 参数用于产生用于 gdb 的调试信息，编译程序如下：

```

→ code gcc 2.c -o 2.out -g
2.c:3:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~

```

运行该程序，可以知道，这是一个求一个整数 n 的阶乘的程序，分析代码可知，该程序对数字的每一位存入数组 a 中，逐位进行乘法运算，可以实现结果不超过 200 位的任意数的阶乘。运行结果如下图所示：

```

→ code ./2.out
Please input n:5
Result is:
5 != 120
→ code ./2.out
Please input n:100
Result is:
100 != 9332621544394415268169923885626670049071596826438162146859296389
52175999932299156089414639761565182862536979208272237582511852109168640
00000000000000000000000000000000

```

使用 gdb 调试程序如下图所示：

```
→ code gdb 全文 标题 1 标题 2 标题 3 新样式 文字工具 查找替换 选择
GNU gdb (Debian 7.12-6+b2) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
> 运行该程序,可以知道,这是一个求一个整数 n 的阶乘的程序,分析代码可知,该
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details. 各位进行乘法运算,可以实现结果不超过 200 位的任意数
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) list
No symbol table is loaded. Use the "file" command.
(gdb) file ./2.out
Reading symbols from ./2.out...done.
(gdb) list
1      #include <stdio.h>
2
3      main()
4      {
5          int n, a[200], carry, temp, i, j, digit = 1;
6          printf("Please input n:");
7          scanf("%d", &n);
8          a[0] = 1;
9          for(i = 2; i <= n; ++i) {
10             for(j = 1, carry = 0; j <= digit; ++j) {
(gdb) □
```

### 3. 编写命令解释程序

#### (1) 内容:

利用 C 语言编写一个微型命令解释程序,接受并解释以下命令:

- |   |               |         |
|---|---------------|---------|
| 1 | dir           | //列当前目录 |
| 2 | cop 文件 1 文件 2 | //拷贝文件  |
| 3 | era 文件名       | //删除文件  |
| 4 | dis 字符串       | //显示字符串 |
| 5 | end           | //结束,退出 |

#### (2) 要求:

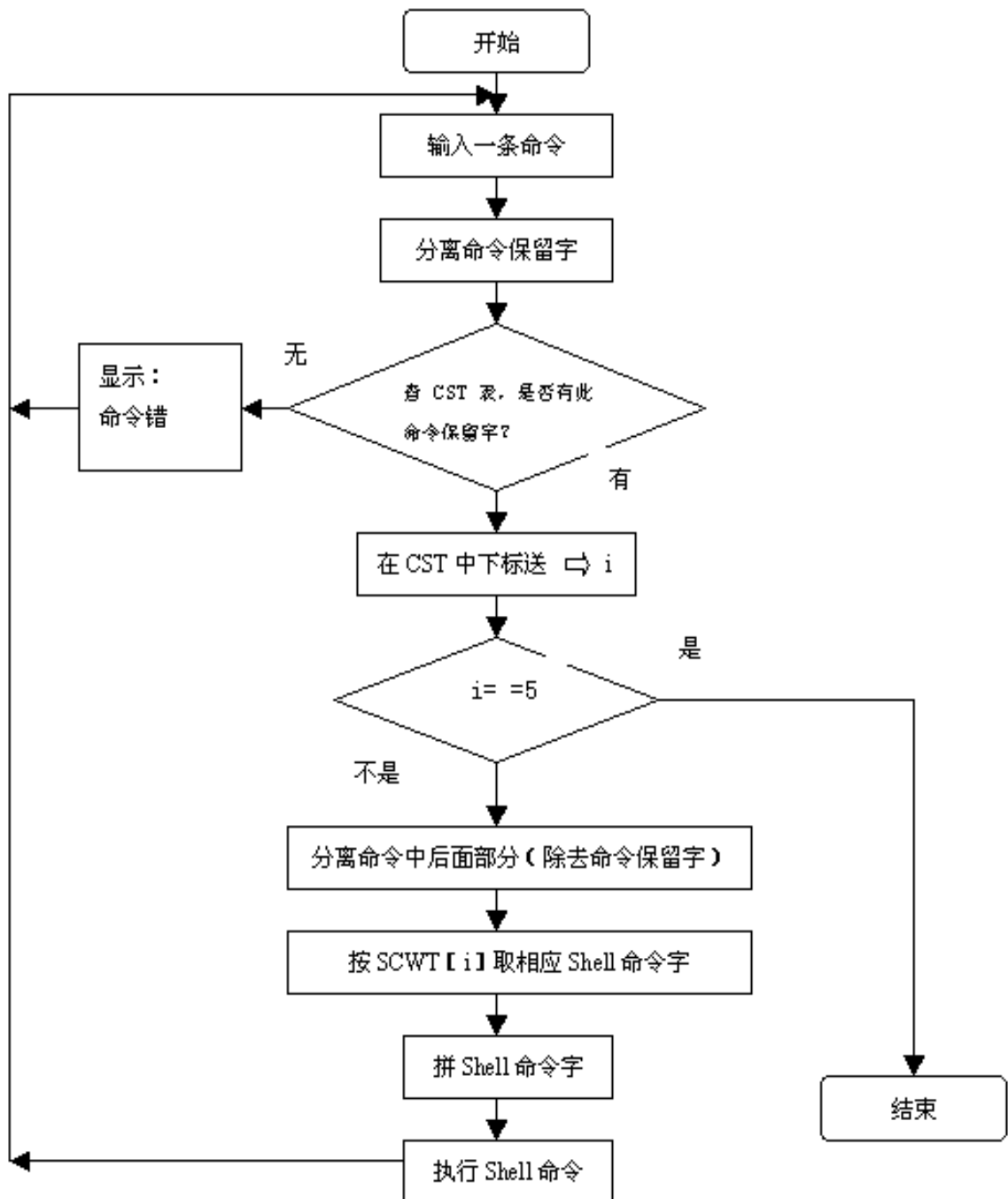
- 1 命令应该由空格隔开;
- 2 进行命令合法性检查,若不合法,显示出错信息,等待重新输入;
- 3 调用 shell 命令来完成各项功能。

#### (3) 思路: (不必拘泥于此,根据自己的理解和想法去编程。)

- 1 用静态指针数组或二维数组形式定义命令保留字表和 shell 命令字表。

静态数组形式如下:

- ```
static char * cst [ ]="dir"... "end"; static char * scwt [ ]="ls
-l"... "exit";
2 输入命令字 gets(string);
3 分离命令字 strtok();
4 比较命令字 strcmp();
```



5 执行 shell 命令：使用系统调用 `system()`。

(4) 命令解释程序模拟算法流程图示例

使用 vi 编辑的程序截图如下：

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #define CMD_MAX_LEN 1000
5
6 void split(char* command, char* cw, char* args);
7 void myGets(char* buff);
8
9 static char * cst[] = {"dir", "cop", "era", "dis", "end"}; //reserved words array
10 static char * scwt[] = {"ls -l", "copy", "rm", "echo", "exit"}; //shell command array
11
12 int main(void)
13 {
14     int i = 0;
15     char command[CMD_MAX_LEN]; //input command
16     char cw[CMD_MAX_LEN], args[CMD_MAX_LEN]; //command word, command args
17     char destCommand[CMD_MAX_LEN]; //shell command word & args
18     int len = sizeof(cst) / sizeof(char *); //number of cst elements
19     printf(">>> "); //prompt
20     myGets(command);
21     while(strcmp(command, "end"))
22     {
23         split(command, cw, args);
24         for(i = 0; i < len; i++)
25         {
26             if(!strcmp(cst[i], cw))
27             {
28                 destCommand[0] = '\0';
29                 strcpy(destCommand, scwt[i]);
30                 strcat(destCommand, " ");
31                 strcat(destCommand, args);
32                 system(destCommand);
33                 if(i == len - 1) break;
34                 printf("Please input n:");
35                 scanf("%d", &i);
36             }
37             if(i == len)
38                 printf("No command named \"%s\".\n", cw);
39             printf(">>> ");
40             myGets(command);
41         }
42     }
43     return 0;
44 }
45
46 void myGets(char* buff)
47 //filter out line break after using fgets
48 {
49     int i = 0;
50     fgets(buff, CMD_MAX_LEN, stdin);
51     //filter out line break
52     while(*(buff + i) != '\0' && *(buff + i) != '\n') i++;
53     *(buff + i) = '\0';
54 }
55
56 void split(char* command, char* cw, char* args)
57 //split command into command word and args
58 {
59     int i = 0, pos1, pos2, pos;
60     //skip leading blanks
61     while(*command == ' ' || *command == '\t') command++;
62     //find the first blank after the command word
63     pos1 = strcspn(command, " ");
64     pos2 = strcspn(command, "\t");
65     pos = pos1 > pos2 ? pos2 : pos1;
66     //save the result
67     for(i = 0; i < pos; i++)
68     {
69         *cw = *(command + i);
70         cw++;
71     }
72     *cw = '\0';
73     if(strlen(command) <= pos)
74         *args = '\0';
75     else
76     {
77         command += pos + 1;
78         strcpy(args, command);
79     }
80 }

```

程序代码如下：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #define CMD_MAX_LEN 1000
5
6  void split(char* command, char* cw, char* args);
7  void myGets(char* buff);
8
9  static char * cst[] = {"dir", "cop", "era", "dis",
10 "end"}; //reserved words array
11
12 static char * scwt[] = {"ls -l", "cp", "rm", "echo",
13 "exit"}; //shell command array
14
15 int main(void)
16 {
17     int i = 0;
18     char command[CMD_MAX_LEN]; //input command
19     char cw[CMD_MAX_LEN], args[CMD_MAX_LEN]; //command word,
20     command args
21     char destCommand[CMD_MAX_LEN]; //shell command word & args
22     int len = sizeof(cst) / sizeof(char *); //number of cst
23     elements
24     printf(">>> "); //prompt
25     myGets(command);
26     while(strcmp(command, "end"))
27     {
28         split(command, cw, args);
29         for(i = 0; i < len; i++)
30         {
31             if(!strcmp(cst[i], cw))
32             {
33                 destCommand[0] = '\0';
34                 strcpy(destCommand, scwt[i]);
35                 strcat(destCommand, " ");
36                 strcat(destCommand, args);
37                 system(destCommand);
38                 break;
39             }
40         }
41     }
42 }
```

```

34         }
35     }
36     if(i == len)
37         printf("No command named \"%s\".\n", cw);
38     printf(">>> ");
39     myGets(command);
40 }
41 return 0;
42 }
43
44 void myGets(char* buff)
45 //filter out line break after using fgets
46 {
47     int i = 0;
48     fgets(buff, CMD_MAX_LEN, stdin);
49     //filter out line break
50     while(*(buff + i) != '\0' && *(buff + i) != '\n') i++;
51     *(buff + i) = '\0';
52 }
53
54 void split(char* command, char* cw, char* args)
55 //split command into command word and args
56 {
57     int i = 0, pos1, pos2, pos;
58     //skip leading blanks
59     while(*command == ' ' || *command == '\t') command++;
60     //find the first blank after the command word
61     pos1 = strcspn(command, " ");
62     pos2 = strcspn(command, "\t");
63     pos = pos1 > pos2 ? pos2 : pos1;
64     //save the result
65     for(i = 0; i < pos; i++)
66     {
67         *cw = *(command + i);
68         cw++;
69     }
70     *cw = '\0';
71     if(strlen(command) <= pos)

```



```

72     *args = '\\0';
73     else
74     {
75         command += pos + 1;
76         strcpy(args, command);
77     }
78 }

```

上述程序中包含了 3 个函数，主函数 main 控制整个程序的输入，在输入 end 时退出，使用 myGets 函数得到一行内容，使用 myGets 的原因是 gets 函数已被弃用，myGets 函数内部通过调用 fgets 函数得到一行内容，fgets 得到的内容包含末尾的换行符，myGets 函数负责去掉换行符。然后再通过调用 split 函数将命令与参数列表分离，支持使用水平制表符或空格（1 个或多个）分割命令与参数。

执行效果演示：

```

myself@myself-PC: /media/myself/Software/操作系统/操作系统实验/操作系统实验二/
→ code gcc 3.c -o 3.out
→ code ./3.out
>>> cop 3.c test
>>> dir
总用量 65
-rwxrwxrwx 1 myself myself 465 4月 1 15:11 2.c
-rwxrwxrwx 1 myself myself 11216 4月 1 15:11 2.out
-rwxrwxrwx 1 myself myself 13128 4月 1 20:00 3
-rwxrwxrwx 1 myself myself 2087 4月 1 21:25 3.c
-rwxrwxrwx 1 myself myself 13160 4月 1 21:30 3.out
-rwxrwxrwx 1 myself myself 75 4月 1 14:56 hello.c
-rwxrwxrwx 1 myself myself 8416 4月 1 14:56 hello.out
-rwxrwxrwx 1 myself myself 2087 4月 1 21:30 test
>>> rm test
No command named "rm".
>>> era test
>>> dir
总用量 61
-rwxrwxrwx 1 myself myself 465 4月 1 15:11 2.c
-rwxrwxrwx 1 myself myself 11216 4月 1 15:11 2.out
-rwxrwxrwx 1 myself myself 13128 4月 1 20:00 3
-rwxrwxrwx 1 myself myself 2087 4月 1 21:25 3.c
-rwxrwxrwx 1 myself myself 13160 4月 1 21:30 3.out
-rwxrwxrwx 1 myself myself 75 4月 1 14:56 hello.c
-rwxrwxrwx 1 myself myself 8416 4月 1 14:56 hello.out
>>> dis "hello, world!"
hello, world!
>>> end
→ code █

```