A sequence of characters.

C 程 序 设 计
C Programming

11

字符串和字符串函数

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院
（国家示范性软件学院）
School of Informatics

黄 炜
博士·副教授
Dr. Wei Huang

知识框架

- 字符串的声明与赋值
- 字符串函数
- 排序算法

内容纲要

字符和字符串

- 字符（char类型）
  - 可打印字符（0x20-0x7E）
    - 字母、数字、运算符号、标点符号和其他符号
  - 控制字符（0x00-0x1F，0x7F）
    - 空格、回车、制表符、退格、警报等……

这里仅指ASCII码字符（char类型），现代程序包含汉字、日文、韩文等，则使用short类型，根据编码不同分为（Unicode、GB2312、Big5等码。）

- 字符串
  - 一个有格式的字符数组，以数值为0的元素为结束。
  - 物理意义：承载信息的文字

字符串的声明

- 字符串的声明

赋值方法	赋值语句	开辟空间	字符串常量开辟空间	复制数组
以数组形式声明	`char c[20];`	20B	0B	无初始化
在声明时赋值（含长度）	`char c[20] = "Hello world!";`	20B	13B	自c[0]赋为'H'至c[11]赋为'!'，c[12]赋为0，其余赋为0。
在声明时赋值（不含长度）	`char c[] = "Hello world!";`	13B	13B	
以指针变量形式赋值	`char *p = "Hello world!";`	4B/8B	13B	将字符串常量的内存首地址赋值给字符指针。

字符串的赋值

- 字符串的赋值

赋值方法	赋值语句
在声明时赋值	`char c[] = "Hello world!";`
通过字符串复制函数赋值	`strcpy(c, "Hello world!");`
通过字符串输入函数赋值	`scanf( "%s" , c );`

- 注意事项
  – 字符串数组名为常量，仅可在声明语句被字符串常量赋值
  – 通过字符串函数赋值前应开辟足够的目标空间

字符串常量的其它形式

- 字符串常量（字符串文字）
  - 一对双引号中包含任何字符，特殊字符转义
  - 字符串允许中间断字（断行）
- 字符串存储时
  - 不包括双引号，包括双引号内的字符，添加结束标志（\0）

```c
char greeting[50] = "Hello, and"" how are" " you"
" today!";
```

```c
char greeting[50] = "Hello, and how are you today!";
```

字符串：长度和数组空间

- 字符串长度：strlen函数（应包含<string.h>）
  - 从数组起始位置开始数，到第一个数字0的偏移量。
- 字符串空间：sizeof操作符
  - 数组声明时，所分配的空间的字节数。
  - 对字符串常量sizeof(name)比strlen(name)多1
  - 对字符串变量sizeof(name)和strlen(name)无必然联系

示例程序中为3

示例程序中为40

字符形式	W	e	i	\0	*	*	*	越界
数值形式	87	101	105	0	*	*	*	越界
标记	起始			strlen				sizeof
索引	0	1	2	3	4	…	39	40

内容纲要

字符串函数：标准输入输出流

- 从键盘接受输入：scanf("%s", ...)
  - 从缓冲区读入一个单词，以空白字符结束
  - 将空白字符除外的元素存入对应参数中
- 输出到屏幕：printf("%s", ...)
  - 将对应参数字符串输出到屏幕
- 字符串应先开辟足够空间再操作，先写后读

字符串函数：标准输入输出流

- 从键盘接受输入：gets(str)
  - 从缓冲区读入一行，以换行符结束
  - 将换行符替换为空字符（值为0），再存入第一个参数中
  - 用户输入可能超过声明字符串时开辟的空间，产生段错误
    - 推荐改用fgets函数
- 输出到屏幕：puts(str)
  - 将第一个参数字符串输出到屏幕
  - 并添加一个换行符

```c
//  strings1.c
#include <stdio.h>
#define MSG "I am a symbolic string constant."
#define MAXLENGTH 81
int main(void)
{
    char words[MAXLENGTH] = "I am a string in an array.";
    const char * pt1 = "Something is pointing at me.";
    puts("Here are some strings:");
    puts(MSG);
    puts(words);
    puts(pt1);
    words[8] = 'p';
    puts(words);

    return 0;
}
```

```
Here are some strings:
I am a symbolic string constant.
I am a string in an array.
Something is pointing at me.
I am a spring in an array.
```

字符串名称是其指针

- 数组名称的值是其指针（内存首地址）
- 字符串名称的值是其指针（内存首地址）
- 对指针可以间接寻址

表达式	类型	值
"Hello world!"	字符串，类比数组a	0x3000
*"Hello world!"	字符，类比数组元素*a，即a[0]	72
"Hello world!"[0]	字符，类比数组元素a[0]	72

- 字符串常量为指向字符的指针常量赋值
  - 如果为指针变量赋值，将来修改元素时，产生段错误

```c
/* strptr.c -- strings as pointers */
#include <stdio.h>
int main(void)
{
    printf("%s, %p, %c\n", "We", "are", *"space farers");

    return 0;
}
```

字符串是数组，其值为字符串首地址p，因此*p为字符串首个元素。

字符串在配合%s或puts输出时输出所有元素（直到0）。

```
We, 00C52010, s
```

```c
//  addresses.c  -- addresses of strings
#define MSG "I'm special."

#include <stdio.h>
int main()
{
    char ar[] = MSG;
    const char *pt = MSG;

    printf("address of \"I'm special\": %p \n", "I'm special");
    printf("                 address ar: %p\n", ar);
    printf("                 address pt: %p\n", pt);
    printf("             address of MSG: %p\n", MSG);
    printf("address of \"I'm special\": %p \n", "I'm special");

    return 0;
}
```

这里有一个英文句号，因而所存储的内存空间不同

字符串（类比于复合文字）的值为指针，且内容不可修改。
有经验的程序员在声明字符串指针时，常加注const，表示其指向的内容不可被修改。

字符串数组与指针的差别

- 运行结果

Visual C++	Ubuntu GCC
address of "I'm special": 00C26B40	address of "I'm special": 0x400705
address ar: 00DAFB9C	address ar: 0x7ffc145ecf30
address pt: 00C26B30	address pt: 0x4006f8
address of MSG: 00C26B30	address of MSG: 0x4006f8
address of "I'm special": 00C26B40	address of "I'm special": 0x400705

  – 加注const是必要的（不加注则产生运行错误）

```c
char * p1 = "Klingon";
p1[0] = 'F'; // ok?
printf("Klingon");
printf(": Beware the %ss!\n", "Klingon");
```

```
Program received signal SIGSEGV, Segmentation fault.
0x000000000040054a in main () at ./addresses.c:8
8         p1[0] = 'F'; // ok?
```

字符串数组

- 以指针数组定义字符串数组，每个字符串不定长
  - 声明语句
    ```
    char *mytalents[LIM]
    ```
  - 每个数组元素为字符串指针，其值没有必然联系
- 以多维数组定义字符串数组，每个字符串数组定长
  - 声明语句
    ```
    char yourtalents[LIM][SLEN]
    ```
  - 每个一级数组元素（如：yourtalents[1]）为字符串指针，相互之间距离为SLEN。

```c
//  arrchar.c -- array of pointers, array of strings
#include <stdio.h>
#define SLEN 40
#define LIM 5
int main(void)
{
    const char *mytalents[LIM] = {
        "Adding numbers swiftly",
        "Multiplying accurately", "Stashing data",
        "Following instructions to the letter",
        "Understanding the C language"
    };
    char yourtalents[LIM][SLEN] = {
        "Walking in a straight line",
        "Sleeping", "Watching television",
        "Mailing letters", "Reading email"
    };
    int i;
    puts("Let's compare talents.");
    printf ("%-36s  %-25s\n", "My Talents", "Your Talents");
    for (i = 0; i < LIM; i++)
        printf("%-36s  %-25s\n", mytalents[i], yourtalents[i]);
    printf("\nsizeof mytalents: %zd, sizeof yourtalents: %zd\n",
           sizeof(mytalents), sizeof(yourtalents));

    return 0;
}
```

```
Let's compare talents.
My Talents                            Your Talents
Adding numbers swiftly                Walking in a straight line
Multiplying accurately                Sleeping
Stashing data                         Watching television
Following instructions to the letter  Mailing letters
Understanding the C language          Reading email

sizeof mytalents: 20, sizeof yourtalents: 200
```

```c
/* p_and_s.c -- pointers and strings */
#include <stdio.h>
int main(void)
{
    const char * mesg = "Don't be a fool!";
    const char * copy;

    copy = mesg;
    printf("%s\n", copy);
    printf("mesg = %s; &mesg = %p; value = %p\n",
           mesg, &mesg, mesg);
    printf("copy = %s; &copy = %p; value = %p\n",
           copy, &copy, copy);

    return 0;
}
```

这里有一个赋值，因而二者的值相同，但二者毕竟是不同的变量，因此所在地址不同。

```
Don't be a fool!
mesg = Don't be a fool!; &mesg = 0xbfc84b58; value = 0x8048560
copy = Don't be a fool!; &copy = 0xbfc84b5c; value = 0x8048560
```

字符串函数：文件流

- 从文件流接受输入：fgets(str, len, fp)
  - 从缓冲区读入一行，以换行符结束，不超过指定长度
    - 有效的字符串和末尾空字符（0）不超过指定长度
  - 将包含换行符在内的元素存入参数中，随后设置空字符0
  - 允许指定最大长度，允许指定输入源fp（标准输入、文件）
- 输出到文件流：fputs(str, fp)
  - 将参数字符串输出到缓冲区，不添加换行符
  - 允许指定输出源fp（标准输出、文件）

```c
/*  getsputs.c  -- using gets() and puts() */
#include <stdio.h>
#define STLEN 81
int main(void)
{
    char words[STLEN];

    puts("Enter a string, please.");
    gets(words);  // typical use
    printf("Your string twice:\n");
    printf("%s\n", words);
    puts(words);
    puts("Done.");

    return 0;
}
```

```
Enter a string, please.
I'll go home.↲
Your string twice:
I'll go home.
I'll go home.
Done.
```

```c
/*  fgets1.c  -- using fgets() and fputs() */
#include <stdio.h>
#define STLEN 14
int main(void)
{
    char words[STLEN];
    puts("Enter a string, please.");
    fgets(words, STLEN, stdin);
    printf("Your string twice (puts(), then fputs()):\n");
    puts(words);
    fputs(words, stdout);
    puts("Enter another string, please.");
    fgets(words, STLEN, stdin);
    printf("Your string twice (puts(), then fputs()):\n");
    puts(words);
    fputs(words, stdout);
    puts("Done.");
    return 0;
}
```

```
I'll↲
Your string twice (puts(), then fputs()):
I'll

I'll
Enter another string, please.
go home↲
Your string twice (puts(), then fputs()):
go home

go home
Done.
```

```c
/*  fgets2.c  -- using fgets() and fputs() */
#include <stdio.h>
#define STLEN 10
int main(void)
{
    char words[STLEN];

    puts("Enter strings (empty line to quit):");
    while (fgets(words, STLEN, stdin) != NULL && words[0] !=
'\n')
        fputs(words, stdout);
    puts("Done.");

    return 0;
}
```

```
Enter strings (empty line to quit):
empty line↲
empty line
good↲
good
↲
Done.
```

```c
/*  fgets3.c  -- using fgets() */
#include <stdio.h>
#define STLEN 10
int main(void)
{
    char words[STLEN];
    int i;
    puts("Enter strings (empty line to quit):");
    while (fgets(words, STLEN, stdin) != NULL && words[0] != 
'\n')
    {
        i = 0;
        while (words[i] != '\n' && words[i] != '\0')
            i++;
        if (words[i] == '\n')
            words[i] = '\0';
        else // must have words[i] == '\0'
```

```
            while (getchar() != '\n')
                continue;
        puts(words);
    }
    puts("done");
    return 0;
}
```

```
empty line↲
empty lin
empty lines↲
empty lin
GOOD↲
GOOD
↲
done
```

```c
/* scan_str.c -- using scanf() */
#include <stdio.h>
int main(void)
{
    char name1[11], name2[11];
    int count;

    printf("Please enter 2 names.\n");
    count = scanf("%5s %10s",name1, name2);
    printf("I read the %d names %s and %s.\n",
           count, name1, name2);

    return 0;
}
```

```
Please enter 2 names.
Jack↲
Rose↲
I read the 2 names Jack and Rose.
```

字符串操作注意事项

- 先开辟空间，后操作
- 先赋值，后读取
- 应确保在开辟空间范围内有终止符0
  - 使用数组形式声明并赋值时
  - 对数组元素进行操作时

```c
/* put_out.c -- using puts() */
#include <stdio.h>
#define DEF "I am a #defined string."
int main(void)
{
    char str1[80] = "An array was initialized to me.";
    const char * str2 = "A pointer was initialized to me.";

    puts("I'm an argument to puts().");
    puts(DEF);
    puts(str1);
    puts(str2);
    puts(&str1[5]);
    puts(str2+4);

    return 0;
}
```

```
I'm an argument to puts().
I am a #defined string.
An array was initialized to me.
A pointer was initialized to me.
ray was initialized to me.
inter was initialized to me.
```

```c
/* nono.c -- no! */
#include <stdio.h>
int main(void)
{
    char side_a[] = "Side A";
    char dont[] = {'W', 'O', 'W', '!' };
    char side_b[] = "Side B";

    puts(dont);   /* dont is not a string */

    return 0;
}
```

```
WOW!Side B
```

字符串函数：字符串格式化

- 从字符串格式化输入：sscanf(str, fmt, ...)
  - 功能与scanf相同，只是输入为字符串
  - 返回值为正确赋值的参数个数
- 格式化输出到字符串：sprintf(str, fmt, ...)
  - 功能与printf相同，只是输出为字符串
  - 返回值为被实际赋值的字符串长度

自定义字符输入输出

- 允许基于getchar()和putchar()建立自己的函数

```c
//put_put.c -- user-defined output functions
#include <stdio.h>
void put1(const char *);
int put2(const char *);

int main(void)
{
    put1("If I'd as much money");
    put1(" as I could spend,\n");
    printf("I count %d characters.\n",
           put2("I never would cry old chairs to mend."));
    return 0;
}

void put1(const char * string)
{
    while (*string)  /* same as *string != '\0' */
        putchar(*string++);
}
```

```c
int put2(const char * string)
{
    int count = 0;
    while (*string)
    {
        putchar(*string++);
        count++;
    }
    putchar('\n');

    return(count);
}
```

```
If I'd as much money as I could spend,
I never would cry old chairs to mend.
I count 37 characters.
```

字符串函数

- 本节自习：注意输入、输出、作用
  - 长度：strlen()
  - 拼接：strcat(); strncat()
  - 比较：strcmp(); strncmp()
  - 复制：strcpy(); strncpy()
  - 打印：sprintf()
  - 其它：strchr(); strpbrk(); strrchr(); strstr()

```c
/* test_fit.c -- try the string-shrinking function */
#include <stdio.h>
#include <string.h> /* contains string function prototypes */
void fit(char *, unsigned int);
int main(void)
{
    char mesg[] = "Things should be as simple as possible,"
    " but not simpler.";

    puts(mesg);
    fit(mesg,38);
    puts(mesg);
    puts("Let's look at some more of the string.");
    puts(mesg + 39);

    return 0;
}
```

```c
void fit(char *string, unsigned int size)
{
    if (strlen(string) > size)
        string[size] = '\0';
}
```

```
Things should be as simple as possible, but not simpler.
Things should be as simple as possible
Let's look at some more of the string.
 but not simpler.
```

```c
/* str_cat.c -- joins two strings */
#include <stdio.h>
#include <string.h>  /* declares the strcat() function */
#define SIZE 80
char * s_gets(char * st, int n);
int main(void)
{
    char flower[SIZE];
    char addon[] = "s smell like old shoes.";
    puts("What is your favorite flower?");
    if (s_gets(flower, SIZE))
    {
        strcat(flower, addon);
        puts(flower);
        puts(addon);
    }
    else
        puts("End of file encountered!");
    puts("bye");
    return 0;
}
char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
What is your favorite flower?
Lily↲
Lilys smell like old shoes.
s smell like old shoes.
bye
```

```c
/* join_chk.c -- joins two strings, check size first */
#include <stdio.h>
#include <string.h>
#define SIZE 30
#define BUGSIZE 13
char * s_gets(char * st, int n);
int main(void)
{
    char flower[SIZE];
    char addon[] = "s smell like old shoes.";
    char bug[BUGSIZE];
    int available;

    puts("What is your favorite flower?");
    s_gets(flower, SIZE);
    if ((strlen(addon) + strlen(flower) + 1) <= SIZE)
        strcat(flower, addon);
    puts(flower);
    puts("What is your favorite bug?");
    s_gets(bug, BUGSIZE);
    available = BUGSIZE - strlen(bug) - 1;
    strncat(bug, addon, available);
    puts(bug);
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
What is your favorite flower?
Rose↵
Roses smell like old shoes.
What is your favorite bug?
bee↵
bees smell l
```

```c
/* nogo.c -- will this work? */
#include <stdio.h>
#define ANSWER "Grant"
#define SIZE 40
char * s_gets(char * st, int n);

int main(void)
{
    char try[SIZE];
    puts("Who is buried in Grant's tomb?");
    s_gets(try, SIZE);
    while (try != ANSWER)
    {
        puts("No, that's wrong. Try again.");
        s_gets(try, SIZE);
    }
    puts("That's right!");
    return 0;
}
```

```c
char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Who is buried in Grant's tomb?
Grant↲
No, that's wrong. Try again.
Grants↲
No, that's wrong. Try again.
Ctrl+C
```

```c
/* compare.c -- this will work */
#include <stdio.h>
#include <string.h>   // declares strcmp()
#define ANSWER "Grant"
#define SIZE 40
char * s_gets(char * st, int n);
int main(void)
{
    char try[SIZE];
    puts("Who is buried in Grant's tomb?");
    s_gets(try, SIZE);
    while (strcmp(try,ANSWER) != 0)
    {
        puts("No, that's wrong. Try again.");
        s_gets(try, SIZE);
    }
    puts("That's right!");
    return 0;
}
```

```c
char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Who is buried in Grant's tomb?
Grant↲
That's right!
```

```c
/* compback.c -- strcmp returns */
#include <stdio.h>
#include <string.h>
int main(void)
{
    printf("strcmp(\"A\", \"A\") i
    printf("%d\n", strcmp("A", "A"));
    printf("strcmp(\"A\", \"B\") is ");
    printf("%d\n", strcmp("A", "B"));
    printf("strcmp(\"B\", \"A\") is ");
    printf("%d\n", strcmp("B", "A"));
    printf("strcmp(\"C\", \"A\") is ");
    printf("%d\n", strcmp("C", "A"));
    printf("strcmp(\"Z\", \"a\") is ");
    printf("%d\n", strcmp("Z", "a"));
    printf("strcmp(\"apples\", \"apple\") is ");
    printf("%d\n", strcmp("apples", "apple"));
    return 0;
}
```

```
strcmp("A", "A") is 0
strcmp("A", "B") is -1
strcmp("B", "A") is 1
strcmp("C", "A") is 1
strcmp("Z", "a") is -1
strcmp("apples", "apple") is 1
```

```c
/* quit_chk.c -- beginning of some program */
#include <stdio.h>
#include <string.h>
#define SIZE 80
#define LIM 10
#define STOP "quit"
char * s_gets(char * st, int n);
int main(void)
{
    char input[LIM][SIZE];
    int ct = 0;
    printf("Enter up to %d lines (type quit to quit):\n", LIM);
    while (ct < LIM && s_gets(input[ct], SIZE) != NULL &&
           strcmp(input[ct],STOP) != 0) {
        ct++;
    }
    printf("%d strings entered\n", ct);
    return 0;
}
```

```c
char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue
    }
    return ret_val;
}
```

```
Enter up to 10 lines (type quit to quit):
begin↲
quite↲
quit↲
2 strings entered
```

```c
/* starsrch.c -- use strncmp() */
#include <stdio.h>
#include <string.h>
#define LISTSIZE 6
int main()
{
    const char * list[LISTSIZE] = {
        "astronomy", "astounding", "astrophysics",
"ostracize",
        "asterism", "astrophobia"
    };
    int count = 0;
    int i;
    for (i = 0; i < LISTSIZE; i++)
        if (strncmp(list[i],"astro", 5) == 0)
        {
            printf("Found: %s\n", list[i]);
            count++;
        }
    printf("The list contained %d words beginning"
           " with astro.\n", count);
    return 0;
}
```

```
Found: astronomy
Found: astrophysics
Found: astrophobia
The list contained 3 words beginning with astro.
```

```c
/* copy1.c -- strcpy() demo */
#include <stdio.h>
#include <string.h>  // declares strcpy()
#define SIZE 40
#define LIM 5
char * s_gets(char * st, int n);
int main(void)
{
    char qwords[LIM][SIZE];
    char temp[SIZE];
    int i = 0;
    printf("Enter %d words beginning with q:\n", LIM);
    while (i < LIM && s_gets(temp, SIZE))
    {
        if (temp[0] != 'q')
            printf("%s doesn't begin with q!\n", temp);
        else
        {
            strcpy(qwords[i], temp);
```

```c
                i++;
            }
    }
    puts("Here are the words accepted:");
    for (i = 0; i < LIM; i++)
        puts(qwords[i]);

    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Enter 5 words beginning with q:
quite nice↲
quick brown fox↲
fox↲
fox doesn't begin with q!
q↲
qqq↲
Q↲
Q doesn't begin with q!
q↲
Here are the words accepted:
quite nice
quick brown fox
q
qqq
q
```

```c
/* copy2.c -- strcpy() demo */
#include <stdio.h>
#include <string.h>    // declares strcpy()
#define WORDS  "beast"
#define SIZE 40
int main(void)
{
    const char * orig = WORDS;
    char copy[SIZE] = "Be the best that you can be.";
    char * ps;
    puts(orig);
    puts(copy);
    ps = strcpy(copy + 7, orig);
    puts(copy);
    puts(ps);
    return 0;
}
```

```
beast
Be the best that you can be.
Be the beast
beast
```

```c
/* copy3.c -- strncpy() demo */
#include <stdio.h>
#include <string.h>  /* declares strncpy() */
#define SIZE 40
#define TARGSIZE 7
#define LIM 5
char * s_gets(char * st, int n);
int main(void)
{
    char qwords[LIM][TARGSIZE];
    char temp[SIZE];
    int i = 0;
    printf("Enter %d words beginning with q:\n", LIM);
    while (i < LIM && s_gets(temp, SIZE))
    {
        if (temp[0] != 'q')
            printf("%s doesn't begin with q!\n", temp);
        else
        {
                strncpy(qwords[i], temp, TARGSIZE - 1);
                qwords[i][TARGSIZE - 1] = '\0';
                i++;
            }
        }
    puts("Here are the words accepted:");
    for (i = 0; i < LIM; i++)
        puts(qwords[i]);
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Enter 5 words beginning with q:
quite↲
o↲
o doesn't begin with q!
quick↲
quit↲
queen↲
quadrangle↲
Here are the words accepted:
quite
quick
quit
queen
quadra
```

```c
/* format.c -- format a string */
#include <stdio.h>
#define MAX 20
char * s_gets(char * st, int n);
int main(void) {
    char first[MAX];
    char last[MAX];
    char formal[2 * MAX + 10];
    double prize;
    puts("Enter your first name:");
    s_gets(first, MAX);
    puts("Enter your last name:");
    s_gets(last, MAX);
    puts("Enter your prize money:");
    scanf("%lf", &prize);
    sprintf(formal, "%s, %-19s: $%6.2f\n", last, first, prize);
    puts(formal);
    return 0;
}
```

```c
char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val) {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Enter your first name:
Wei↲
Enter your last name:
Huang↲
Enter your prize money:
10000↲
Huang, Wei                      : $10000.00
```

ctype.h字符函数和字符串

- 基于ctype.h字符函数，可用于编写
  - 字符串的转换大写、小写、首字母大写、句首大写、统计空白符个数等函数。
- 使用字符函数前应包含相应文件头
- 由于上述函数较为简单，方便自行编写，不需要特地记忆。

```c
/* mod_str.c -- modifies a string */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define LIMIT 81
void ToUpper(char *);
int PunctCount(const char *);

int main(void)
{
    char line[LIMIT];
    char * find;
    puts("Please enter a line:");
    fgets(line, LIMIT, stdin);
    find = strchr(line, '\n');// look for newline
    if (find)                 // if the address is not NULL,
        *find = '\0';         // place a null character there
    ToUpper(line);
    puts(line);
    printf("That line has %d punctuation characters.\n",
PunctCount(line));
    return 0;
}

void ToUpper(char * str)
{
    while (*str)
    {
        *str = toupper(*str);
        str++;
    }
}
```

```
int PunctCount(const char * str)
{
    int ct = 0;
    while (*str)
    {
        if (ispunct(*str))
            ct++;
        str++;
    }

    return ct;
}
```

```
Please enter a line:
Not yet.↲
NOT YET.
That line has 1 punctuation characters.
```

字符串把转换为数字

- 本节自习：注意输入、输出、作用
  - 格式化输入：sscanf()
  - 其它：atoi(); atof()
  - 其它： strtod(); strtol(); strtoul()

```c
/* strcnvt.c -- try strtol()  */
#include <stdio.h>
#include <stdlib.h>
#define LIM 30
char * s_gets(char * st, int n);

int main()
{
    char number[LIM];
    char * end;
    long value;
    puts("Enter a number (empty line to quit):");
    while(s_gets(number, LIM) && number[0] != '\0')
    {
        value = strtol(number, &end, 10);  /* base 10 */
        printf("base 10 input, base 10 output: %ld, stopped
at %s (%d)\n",
               value, end, *end);
        value = strtol(number, &end, 16);  /* base 16 */
        printf("base 16 input, base 10 output: %ld, stopped at %s (%d)\n",
               value, end, *end);
        puts("Next number:");
    }
    puts("Bye!\n");
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
```

```c
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_v
}
```

```
Enter a number (empty line to quit):
53↲
base 10 input, base 10 output: 53, stopped at  (0)
base 16 input, base 10 output: 83, stopped at  (0)
Next number:
67↲
base 10 input, base 10 output: 67, stopped at  (0)
base 16 input, base 10 output: 103, stopped at  (0)
Next number:
↲
Bye!
```

内容纲要

命令行参数的起因

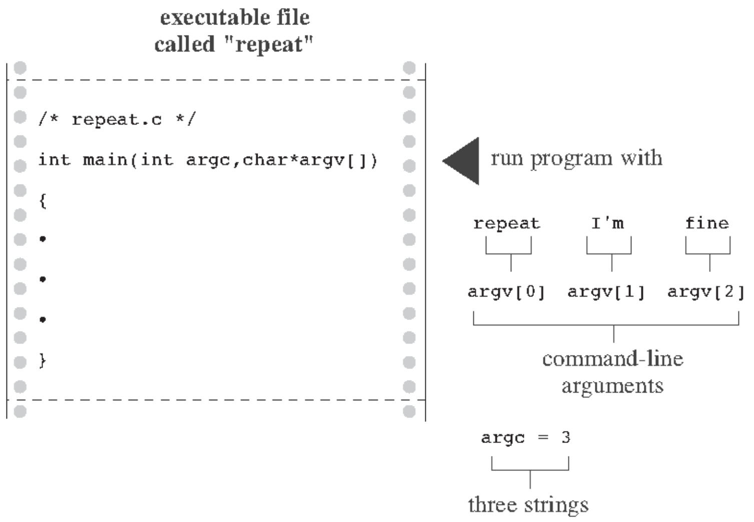
- 很多服务器都是无人值守的
  - 虽然启动程序可以靠鼠标双击，服务器重启后一些程序无法靠鼠标自动启动
  - 通过命令行启动程序，才能在服务器重启时自动运行
- 为了避免重复生成多个程序，命令行也应有参数

命令行参数

- 如GCC一般，程序可以读取命令行参数为自己所用

```
gcc main.c -o main -Wall -std=c99
```

  - main函数的第一个参量是命令行参数个数
  - main函数的第二个参量是命令行参数字符数组



```
/* repeat.c */
int main(int argc,char*argv[])
{
.
.
.
}
```

```c
/* repeat.c -- main() with arguments */
#include <stdio.h>
int main(int argc, char *argv[])
{
    int count;

    printf("The command line has %d arguments:\n", argc - 1);
    for (count = 1; count < argc; count++)
        printf("%d: %s\n", count, argv[count]);
    printf("\n");

    return 0;
}
```

```
D:\***\Ch11>repeat I'll be here "I'll be here"↲
The command line has 4 arguments:
1: I'll
2: be
3: here
4: I'll be here
```

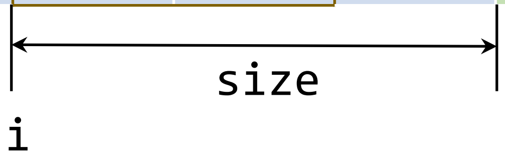
内容纲要

冒泡排序法（Bubble Sort）

- 两层循环
  - 内层循环逐个比较，将最大值（泡泡）浮至顶部
  - 外层循环控制内层循环的范围，已经浮出的元素不需再比较

size	i	a[0]	a[1]	a[2]	a[3]
4	1	4	3	2	1
4	2	3	4	2	1
4	3	3	2	4	1
3	1	3	2	1	4
3	2	2	3	1	4
2	1	2	1	3	4
-	-	1	2	3	4

size	i	a[0]	a[1]	a[2]	a[3]
3	1	3	2	1	4



```
for (size = len; size > 1; size--)
    for (i = 1; i < size; i++)
        swap(arr[i - 1], arr[i]);
```

```
void bubble_sort_1(double arr[], const unsigned int len)
{
    unsigned int size, i;
    for (size = len; size > 1; size--) // 排序的范围从全数组慢慢收缩至停止
    {
        for (i = 1; i < size; i++) // 单趟排序
        {
            if (arr[i - 1] > arr[i]) // 需要发生交换的条件
            {
                double temp; // temp的唯一正确用法
                temp = arr[i - 1];
                arr[i - 1] = arr[i];
                arr[i] = temp;
            }
        }
    }
}
```

```
void bubble_sort_2(double arr[], const unsigned int len)
{
    unsigned int sorted, i; // sorted是排好序的下标，i是索引
    for (sorted = 0; sorted < len; sorted++) // 排序的范围从全数组慢慢收缩至停止，虽然此处是递增的，但是在下面使用时有减号
    {
        for (i = 1; i < len - sorted; i++) // 单趟排序
        {
            if (arr[i - 1] > arr[i]) // 需要发生交换的条件
            {
                double temp; // temp的唯一正确用法
                temp = arr[i - 1];
                arr[i - 1] = arr[i];
                arr[i] = temp;
            }
        }
    }
}
```

冒泡排序法（加强版）

- 增加一个标记量
  - 如果某次内层排序没有实际发生排序，则认定为已经排好序，循环终止。

size	i	a[0]	a[1]	a[2]	a[3]
4	1	4	1	2	3
4	2	1	4	2	3
4	3	1	2	4	3
3	1	1	2	3	4
3	2	1	2	3	4
2	1	1	2	3	4
-	-	1	2	3	4

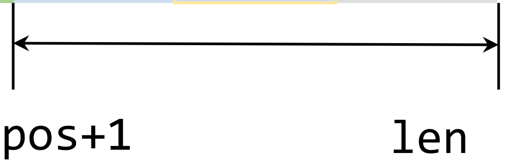
```
void bubble_sort_3(double arr[], const unsigned int len)
{
    unsigned int size, i; // size是排序的范围，i是索引
    for (size = len; size > 1; size--) // 排序范围从全数组收缩至停止
    {
        int is_swapped = 0; // 标记是否交换
        for (i = 1; i < size; i++) // 单趟排序
        {
            if (arr[i - 1] > arr[i]) // 需要发生交换的条件
            {
                double temp; // temp的唯一正确用法
                temp = arr[i - 1];
                arr[i - 1] = arr[i];
                arr[i] = temp;
                is_swapped = 1; // 标记已经经过交换
            }
        }
        if (!is_swapped) // 如果未经交换，说明已排好序，可退出不必再排
            break;
    }
}
```

选择排序法（ Select Sort ）

- 两层循环
  - 内层循环逐个比较，将最大值与所应在位置置换
  - 外层循环控制内层循环的范围，已经安排的位置不再比较

pos	min_i	a[0]	a[1]	a[2]	a[3]
0	2	4	3	5	1
1	2	5	3	4	1
2	2	5	4	3	1
3	-	5	4	3	1

pos	j	a[0]	a[1]	a[2]	a[3]
0	1	4	3	5	1



```
for (pos = 0; pos < len - 1; pos++)
    for (j = pos + 1; j < len; j++)
        swap(arr[j - 1], arr[j]);
```

```c
void select_sort(double arr[], const unsigned int len) {
    for (unsigned int i = 0; i < len - 1; i++) {
        int min_i = i;
        for (unsigned int j = i + 1; j < len; j++)
            if (arr[j] < arr[min_i])
                min_i = j;
        if (min_i != i) {
            double temp = arr[i];
            arr[i] = arr[min_i];
            arr[min_i] = temp;
        }
    }
}
```

快速排序法（ Quick Sort ）

- 选择元素，将小放到左边，大放到右边，再分别排序

low	hig	i	j	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
0	8	1	5	4	7	2	9	3	1	5	6	8
				x	从左向右找出大于x的数				从右向左找出小于x的数			
		3	4	1	*	2	9	3	7	5	6	8
		3	3	1	3	2	*	9	7	5	6	8
lo	hi	lo	hi	1	3	2	4	9	7	5	6	8
0	0	8	8	1	3	2	4	8	7	5	6	9
2	2	7	7	1	2	3	4	6	7	5	8	9
		5	6	1	2	3	4	5	6	7	8	9
		5	5	1	2	3	4	5	6	7	8	9

```
void quick_sort(double arr[], const int left, const int right)
{
    if (left < right)
    {
        double key = arr[left];
        int low = left;
        int high = right;
        while (low < high)
        {
            while (low < high && arr[high] > key)
                high--;
            arr[low] = arr[high];
            while (low < high && arr[low] < key)
                low++;
            arr[high] = arr[low];
        }
        arr[low] = key;
        quick_sort(arr, left, low - 1);
        quick_sort(arr, low + 1, right);
    }
}
```

例题：字符串排序

- 在字符串排序时，尽量使用指针而不是字符串复制
  - 字符串的复制比指针赋值耗费的时间更长
  - 将指针或下标存入数组也需要耗费时间

字符串复制	指针/下标复制

`for(i=0; i<STR_COUNT; i++) {`
`  for(j=0; j<STR_COUNT-1-i; j++)`
`    if(strcmp(str[j],str[j+1])>0) {`
`      strcpy(s,str[j]);`
`      strcpy(str[j],str[j+1]);`
`      strcpy(str[j+1],s);`
`    }`
`}`
	`for(i=0; i<STR_COUNT; i++)`
	`  for(j=0; j<STR_COUNT-1-i; j++)`
	`      if(strcmp(str[index[j]],`
	`str[index[j+1]])>0) {`
	`        temp=index[j];`
	`        index[j]=index[j+1];`
	`        index[j+1]=temp;`
	`      }`

```
/* sort_str.c -- reads in strings and sorts them */
#include <stdio.h>
#include <string.h>
#define SIZE 81        /* string length limit, including \0  */
#define LIM 20         /* maximum number of lines to be read */
#define HALT ""        /* null string to stop input          */

void stsrt(char *strings[], int num);  /* string-sort function */
char * s_gets(char * st, int n);

int main(void)
{
    char input[LIM][SIZE];     /* array to store input       */
    char *ptstr[LIM];          /* array of pointer variables */
    int ct = 0;                /* input count                */
    int k;                     /* output count               */

    printf("Input up to %d lines, and I will sort them.\n",LIM);
    printf("To stop, press the Enter key at a line's start.\n");
    while (ct < LIM && s_gets(input[ct], SIZE) != NULL
           && input[ct][0] != '\0')
    {
        ptstr[ct] = input[ct];  /* set ptrs to strings        */
        ct++;
    }
    stsrt(ptstr, ct);          /* string sorter              */
    puts("\nHere's the sorted list:\n");
    for (k = 0; k < ct; k++)
        puts(ptstr[k]);        /* sorted pointers            */
    return 0;
}

/* string-pointer-sorting function */
void stsrt(char *strings[], int num)
{
    char *temp;
    int top, seek;
    for (top = 0; top < num-1; top++)
        for (seek = top + 1; seek < num; seek++)
            if (strcmp(strings[top],strings[seek]) > 0)
            {
                temp = strings[top];
                strings[top] = strings[seek];
                strings[seek] = temp;
            }
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    int i = 0;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        while (st[i] != '\n' && st[i] != '\0')
```

```c
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else // must have words[i] == '\0'
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

```
Input up to 20 lines, and I will sort them.
To stop, press the Enter key at a line's start.
notice↲
Be careful↲
note 4↲
!!↲
~~↲
↲

Here's the sorted list:

!!
Be careful
note 4
notice
~~
```

内容纲要

C 程 序 设 计
C Programming

11

谢谢观看

理
论
课
程

厦門大學
XIAMEN UNIVERSITY

信 息 学 院
（国家示范性软件学院）
School of Informatics

黄 炜
博士·副教授
Dr. Wei Huang