几个疑惑

- `‘\0’==0`
- `’\0’==‘0’`
- `if(‘\0’)`

- `int a[10];`
- `a==&a`
- `a==&a[0]`
- `a[0]==*a`

School of Economics ang Management, Beihang University

Introduction to C Programming

Jichang Zhao

jichang@buaa.edu.cn

Character Strings

Objectives

- **String Fundamentals**
- **Library Functions**
- Case Study: Password Validation

String Fundamentals

- A **string literal** is any sequence of characters enclosed in **double quotes**
  - `"Good Morning!"`
  - Also called **string constant**, **string value**, **string**
  - A string is stored as an array of characters terminated by an end-of-string symbolic constant named NULL (`'\0'`)
- 补充：字符串常量的长度有限制，以前是509(C89)，新标准是4095(C99)
- 补充：就本质而言，C语言将字符串作为字符数组(长度+1）
- 补充：'\0'不是'0'，实际上0的ASCII编码是48
- 补充：存储于内存中的只读区域（Literals)，相同的字符串常量可能只存储一次

String Fundamentals

- 字符串的延续
  - 有可能超过屏幕的一行，为了提高可读性可以换行

1. printf(“I am now printing a very very very very \
        long string...");

2. printf("I am now printing a very very"
  "long string...");

String Fundamentals

- 字符串可以在声明时初始化
- char date[8]=“June 14”;
- char date[8]={‘J’,’u’,’n’,’e’,’ ’,’1’,’4’,’\0’};
- char date2[9]=”June 14”;
- char date2[9]={‘J’,’u’,’n’,’e’,’ ’,’1’,’4’,’\0’,’\0’}
- //编译器自动填充\0
- char date3[7]=“June 14”;
- char date3[7]={‘J’,’u’,’n’,’e’,’ ’,’1’,’4’};
- //这时编译器没在末尾中\0，即不是字符串，只是字符数组
- 数组长度一定要长于初始化字符串的长度（包括\0）
- char date[]=“June 14”;//编译器会自动计算长度，建议这种方式
- char *p=“abc”;
- char ch=“abc”[0];
- return ”0123456789abcdef”[digit];//这是什么函数？
- ***p=‘d’;//错误，会导致未定义错误，为什么？**

String Fundamentals

- 字符数组与字符指针
  - char date[]=“June 14”;
  - char *date=“June 14”;
  - 声明为数组时，可以修改date中的字符，但声明为指针是不可以修改
  - 声明为数组时，date是数组名；声明为指针时，date是普通的指针变量，可以指向其他字符串

  - char *p；**编译器并未给指向的字符串分配空间**
  - 1. char *p=malloc(...)；//动态分配
  - 2. char string[N+1], *p; p=string;

String Fundamentals

- 计算字符串中的空格
- int count_spaces(const char s[])
- {
  - int count=0,i;
  - for(i=0;s[i]!='\0';i++)
    - if(s[i]==' ')
      - count++;
  - return count;
- }
- int count_spaces(const char *s)
- {
  - int count=0;
  - while(*s)
  - {

```
        if(*s==' ')
          count++;
        s++;
      }
      return count;
```

- }

补充：使用数组或指针没有区别，一般使用指针更多一些。

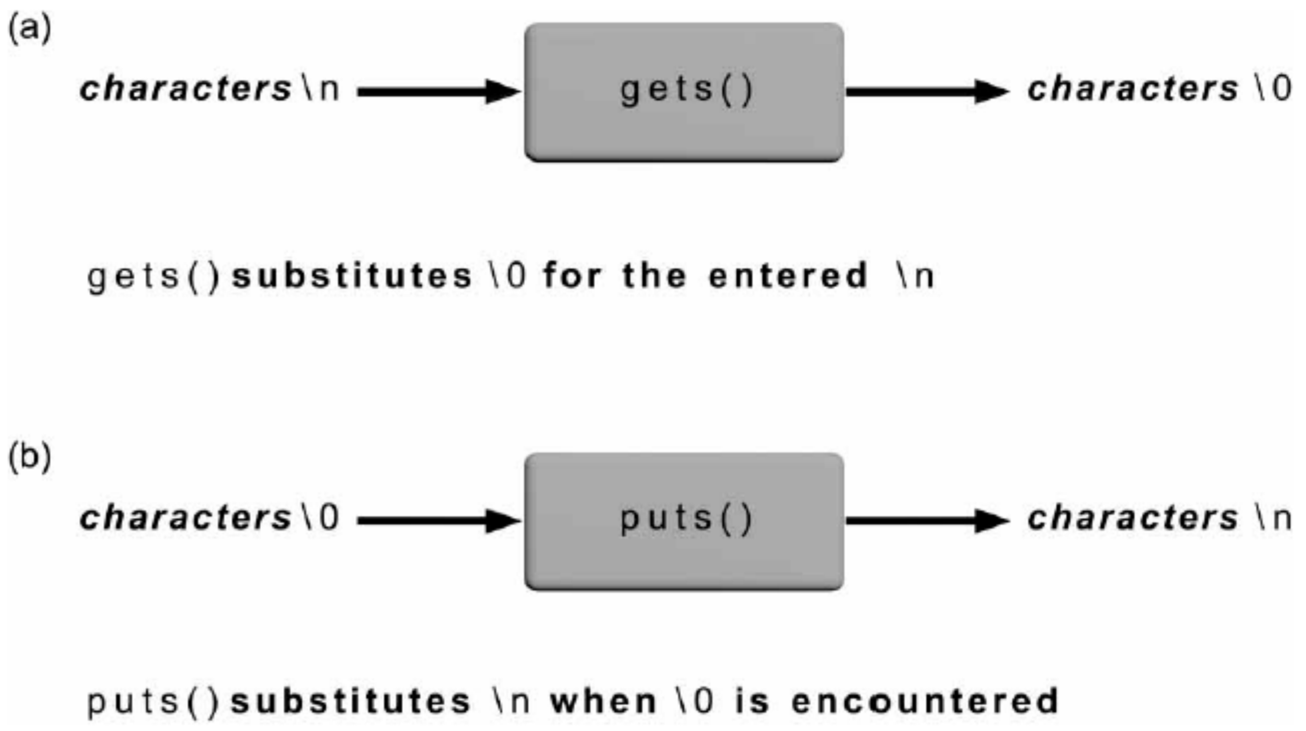
String Input and Output

- `gets()` accepts and stores the characters typed at the terminal into the character array
  - Pressing the Enter key generates a newline character, `\n`, which is interpreted by `gets()` as the ***end-of-character entry***
  - All the characters encountered by `gets()`, **except the newline character**, are stored in the message array
  - Automatically put '\0'

String Input and Output

- A `printf()` function call can be used in place of a `puts()` function call
  - `printf("%s\n",message);` ≡ `puts(message);`
- This correspondence between the output functions is not duplicated by the input functions `scanf()` and `gets()`
  - `gets()` stops accepting characters only ***when a newline is detected***
  - 补充：`scanf("%s"…)`不会读入空白符，会在末尾加`'\0'`
- **补充：gets不会跳过空白符，只在遇到\n时结束；scanf("%s"..)会跳过开始的空白字符，并在遇到空白字符处停止**
- **补充：gets和scanf都不会检查读取的长度是否超过写入数组的边界，因此如果超过则导致未定义行为**

String Input and Output



(a)
characters\n
gets()
characters \0
gets() substitutes \0 for the entered \n
(b)
characters\0
puts()
characters \n
puts() substitutes \n when \0 is encountered

Figure 9.2 (a) `gets()` substitutes `\0` for the entered `\n`
(b) `puts()` substitutes `\n` when `\0` is encountered

String Input and Output

- char ch=getchar();
  - 读入一个字符并将其返回
  - 不会跳过空白字符
  - 如果失败，会返回EOF(ctrl Z或D,可能需要输入两次)
  - 比scanf要快许多
  - while(getchar()!='\n');//skip rest of line
  - while((ch=getchar())==' ');//skip blanks
- putchar(ch);
  - 打印字符
  - 比printf要快许多
- 如何实现读取一行？

String Input and Output

```
#include<stdio.h>
int main(void)
{
      int length=0;
      while(getchar()!=‘\n’)
            length++;
      printf(“input string’s length is:%d\n”,length);
      return 0;
}
```

String Processing

```
#include<stdio.h>
#define LSIZE 81
int main()
{
      char message[LSIZE];
      char c;
      int i;
      i=0;
      while(i<(LSIZE-1)  && (c=getchar())!='\n'
                         && c!=EOF)
      {
            message[i++]=ch;
      }
      message[i]='\0';
}
```

补充：一种更好的风格时实际的字符数组长度总是大于**LSIZE**，即：

char message[LSIZE+1];

String Processing

```
int readline_asstr_array(char line[],int n)
{
        int i;
        char ch;
        while((ch=getchar())!='\n' && ch!=EOF)
        {
                if(i<n)
                {
                        line[i++]=ch;
                }
        }
        line[i]='\0';
        return i;
}//注意：根据上页的建议风格，字符数组的实际长度为n+1
```

Library Functions

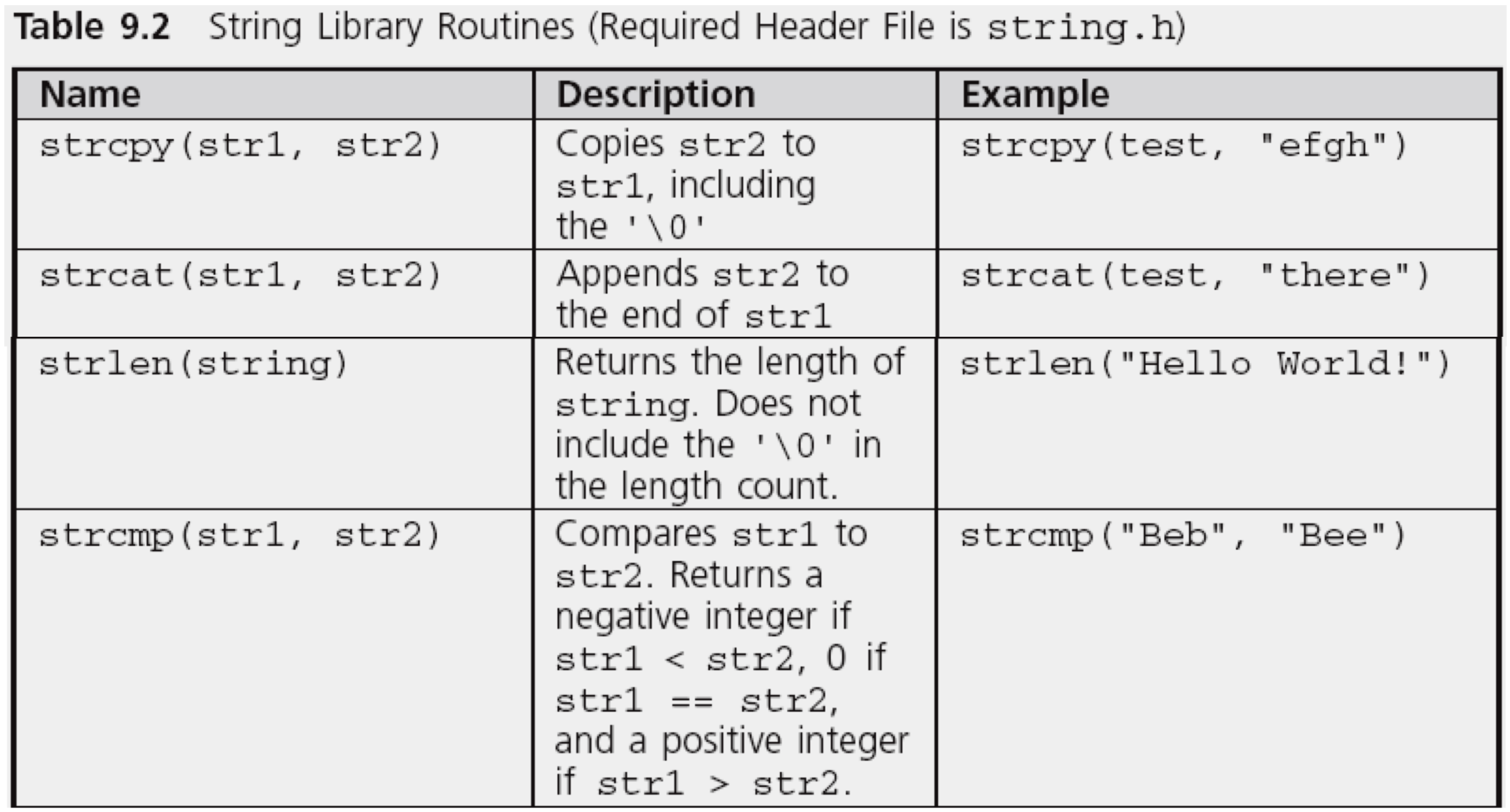


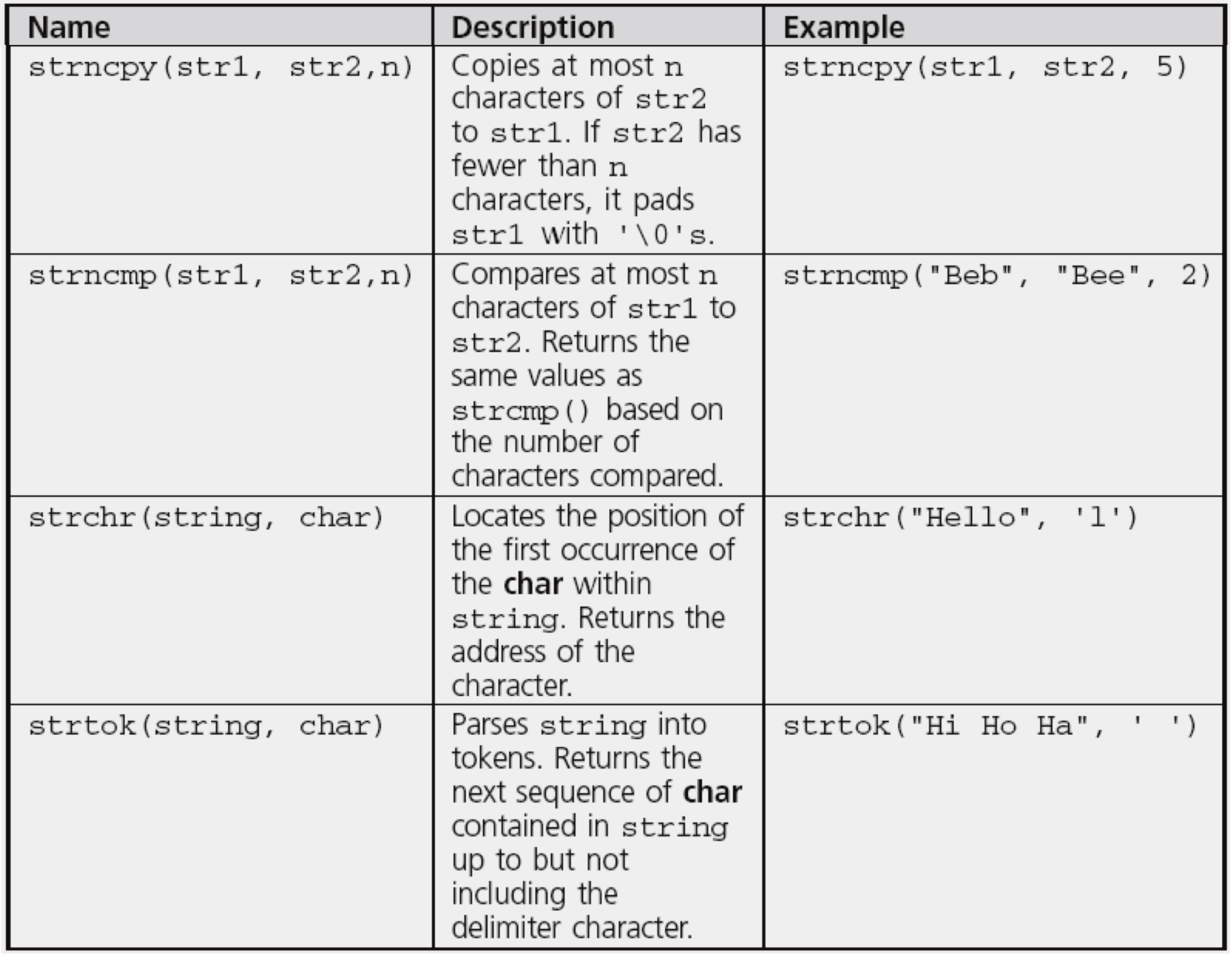
Table 9.2 String Library Routines (Required Header File is `string.h`)

Name	Description	Example
`strcpy(str1, str2)`	Copies `str2` to `str1`, including the `'\0'`	`strcpy(test, "efgh")`
`strcat(str1, str2)`	Appends `str2` to the end of `str1`	`strcat(test, "there")`
`strlen(string)`	Returns the length of `string`. Does not include the `'\0'` in the length count.	`strlen("Hello World!")`
`strcmp(str1, str2)`	Compares `str1` to `str2`. Returns a negative integer if `str1 < str2`, 0 if `str1 == str2`, and a positive integer if `str1 > str2`.	`strcmp("Beb", "Bee")`

Note: **Attempting to copy a larger string into a smaller string causes the copy to overflow the destination array beginning with the memory area immediately following the last array element.**

Library Functions

Table 9.2 String Library Routines (Required Header File is `string.h`) (continued)



Name	Description	Example
`strncpy(str1, str2,n)`	Copies at most `n` characters of `str2` to `str1`. If `str2` has fewer than `n` characters, it pads `str1` with `'\0'`s.	`strncpy(str1, str2, 5)`
`strncmp(str1, str2,n)`	Compares at most `n` characters of `str1` to `str2`. Returns the same values as `strcmp()` based on the number of characters compared.	`strncmp("Beb", "Bee", 2)`
`strchr(string, char)`	Locates the position of the first occurrence of the **char** within `string`. Returns the address of the character.	`strchr("Hello", 'l')`
`strtok(string, char)`	Parses `string` into tokens. Returns the next sequence of **char** contained in `string` up to but not including the delimiter character.	`strtok("Hi Ho Ha", ' ')`

Library Functions

- When comparing strings, their individual characters are evaluated in pairs; if a difference is found, the string with the first lower character is the smaller one
  - `"Good Bye"` is less than `"Hello"` because the first `'G'` in Good Bye is less than the first `'H'` in Hello
  - `"Hello"` is less than `"Hello "` because the `'\0'` terminating the first string is less than the `' '` in the second string
  - `"123"` is greater than `"122"` because `'3'` in `123` is greater than `'2'` in `122`
  - `"1237"` is greater than `"123"` because `'7'` in `1237` is greater than `'\0'` in `123`

Library Functions

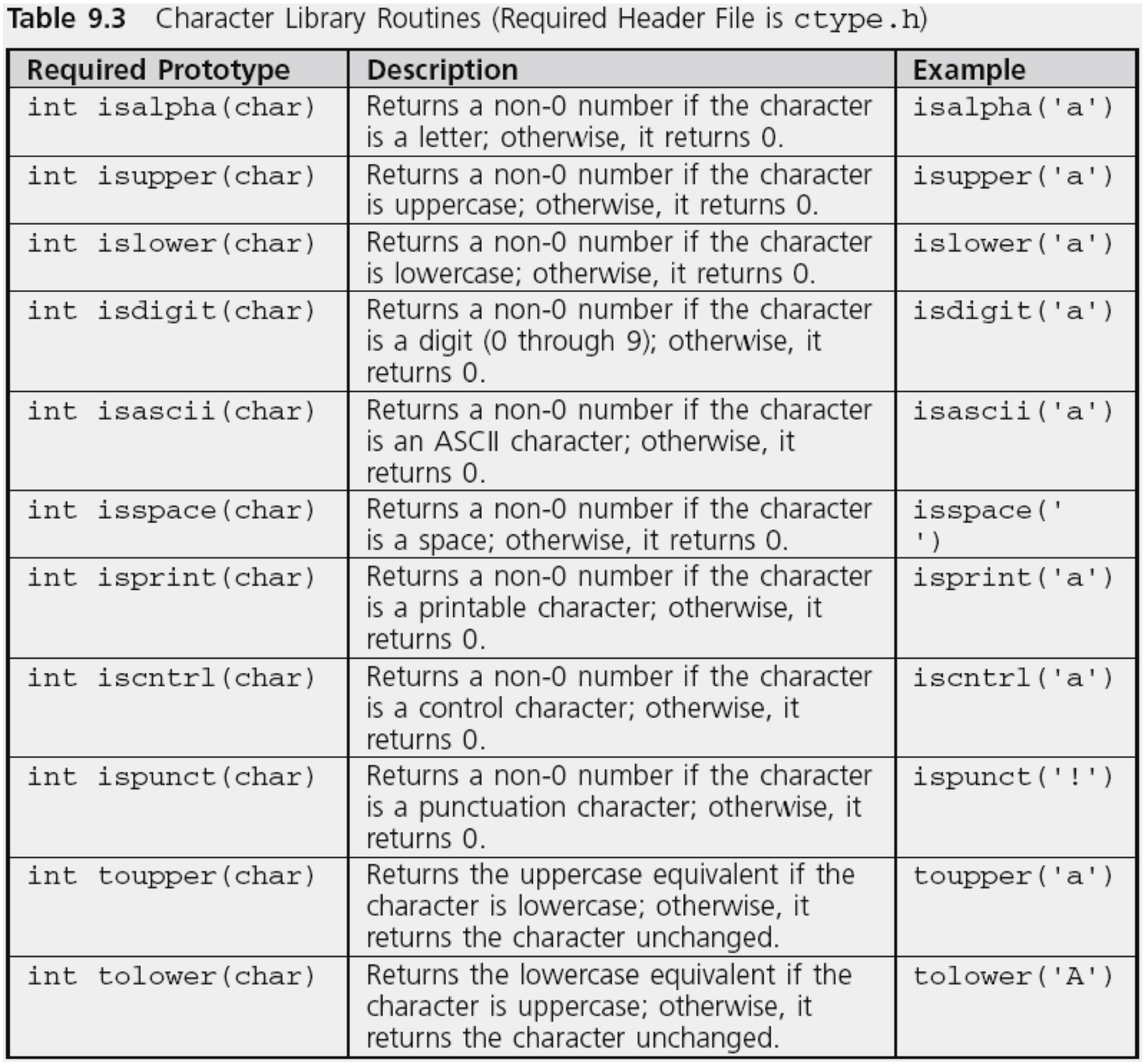


Table 9.3 Character Library Routines (Required Header File is `ctype.h`)

Required Prototype	Description	Example
`int isalpha(char)`	Returns a non-0 number if the character is a letter; otherwise, it returns 0.	`isalpha('a')`
`int isupper(char)`	Returns a non-0 number if the character is uppercase; otherwise, it returns 0.	`isupper('a')`
`int islower(char)`	Returns a non-0 number if the character is lowercase; otherwise, it returns 0.	`islower('a')`
`int isdigit(char)`	Returns a non-0 number if the character is a digit (0 through 9); otherwise, it returns 0.	`isdigit('a')`
`int isascii(char)`	Returns a non-0 number if the character is an ASCII character; otherwise, it returns 0.	`isascii('a')`
`int isspace(char)`	Returns a non-0 number if the character is a space; otherwise, it returns 0.	`isspace(' ')`
`int isprint(char)`	Returns a non-0 number if the character is a printable character; otherwise, it returns 0.	`isprint('a')`
`int iscntrl(char)`	Returns a non-0 number if the character is a control character; otherwise, it returns 0.	`iscntrl('a')`
`int ispunct(char)`	Returns a non-0 number if the character is a punctuation character; otherwise, it returns 0.	`ispunct('!')`
`int toupper(char)`	Returns the uppercase equivalent if the character is lowercase; otherwise, it returns the character unchanged.	`toupper('a')`
`int tolower(char)`	Returns the lowercase equivalent if the character is uppercase; otherwise, it returns the character unchanged.	`tolower('A')`

Conversion Routines

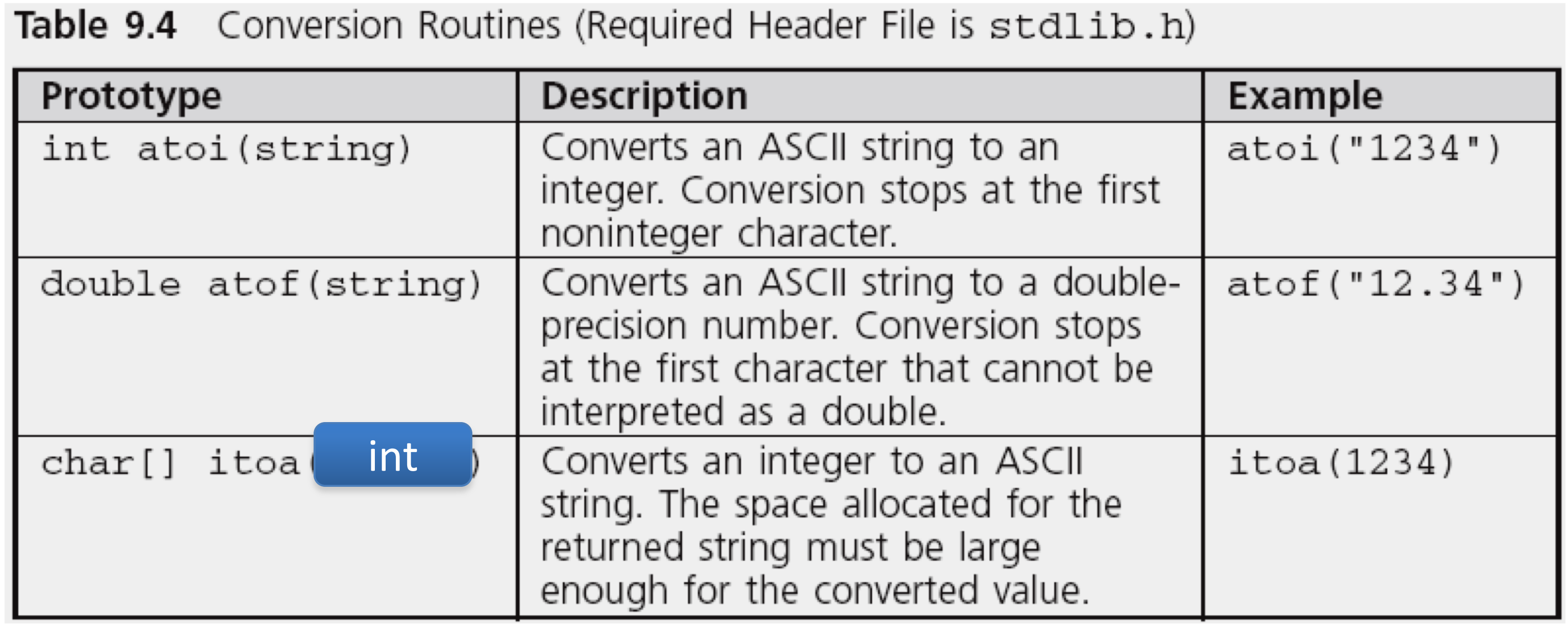


Table 9.4 Conversion Routines (Required Header File is `stdlib.h`)

Prototype	Description	Example
`int atoi(string)`	Converts an ASCII string to an integer. Conversion stops at the first noninteger character.	`atoi("1234")`
`double atof(string)`	Converts an ASCII string to a double-precision number. Conversion stops at the first character that cannot be interpreted as a double.	`atof("12.34")`
`char[] itoa(int)`	Converts an integer to an ASCII string. The space allocated for the returned string must be large enough for the converted value.	`itoa(1234)`

补充：一些习惯用法

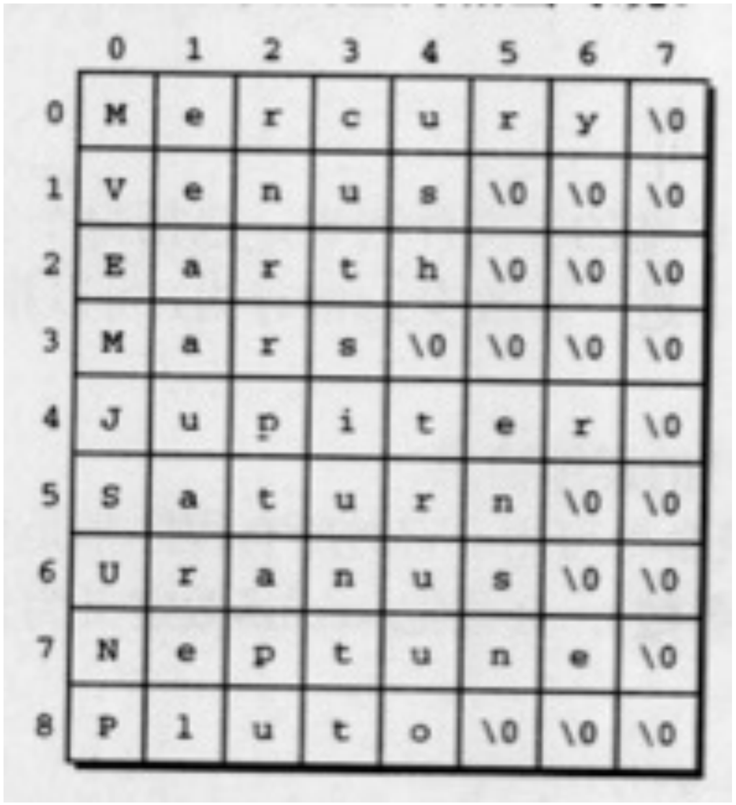
- 搜索字符串结尾
  - `while(*s++);`

  - `const char *p=s;`
  - `while(*s++);`
  - `return s-p;`

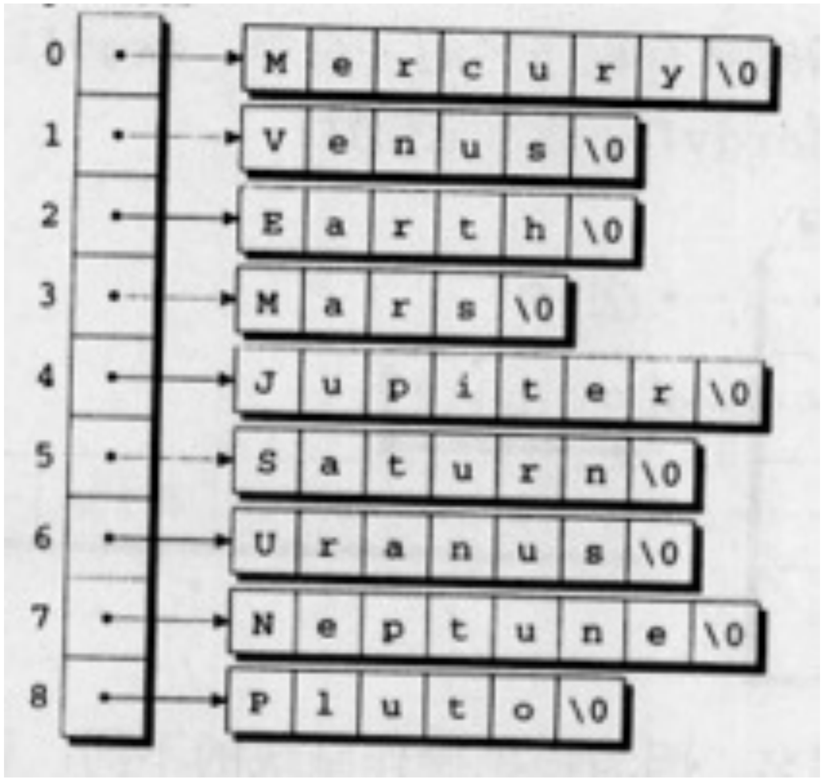
- 字符串复制
  - `while(*p++=*s++) ;`

补充：字符串数组

- 两种方式对应的存储方式有差别
  - char strings[][8]={ “Earth” , “Venus” …};
  - char *strings[]={“Earth”,”Venus”};
  - DEMO:sa.c



	0	1	2	3	4	5	6	7
0	M	e	r	c	u	r	y	\0
1	V	e	n	u	s	\0	\0	\0
2	E	a	r	t	h	\0	\0	\0
3	M	a	r	s	\0	\0	\0	\0
4	J	u	p	i	t	e	r	\0
5	S	a	t	u	r	n	\0	\0
6	U	r	a	n	u	s	\0	\0
7	N	e	p	t	u	n	e	\0
8	P	l	u	t	o	\0	\0	\0



补充：命令行参数

- 程序运行前需要提供的信息，如文件名或控制参数等
- C语言里也称为程序参数
  - main函数修改为`int main(int argc, char*argv[])`
  - argc是参数计数，包括程序名本身
  - argv参数向量，argv[0]是程序名
  - argv[argc]被设定为NULL，用来标记参数的结束
  - DEMO:arg.c

Case Study

- Ask the user to set up the password
- The password has to be a string with
  - No less than 6 characters
  - Capitalized letter(s)
  - Numbers
- If valid, accept it as new password;
- If not, ask the user to choose a new password.
- ***DEMO (checkpw.c)***
- ***如何精确地提示错误?***

Homework

- 均建议用指针实现，并注意测试的充分性
- 1, P351第4题
- 2, P351第8题
- 3, P372第6题
- 4, 《现代方法第2版》, P221, 12
- 5, 《现代方法第2版》, P221, 13
- 6, 《现代方法第2版》, P222, 1
- 7, 《现代方法第2版》, P223, 4