附录G

常用的ANSI C标准库函数

1. 数学函数

使用数学函数时，应在源文件中包含头文件math.h。数学函数如表1所示。

表 1　数学函数

函数名	函数原型	功能
acos	double acos(double x);	计算$\sin^{-1}(x)$的值，返回计算结果。注意，x的取值范围为−1～1
asin	double asin(double x);	计算$\cos^{-1}(x)$的值，返回计算结果。注意，x的取值范围为−1～1
atan	double atan(double x);	计算$\tan^{-1}(x)$的值，返回计算结果
atan2	double atan2(double x, double y);	计算$\tan^{-1}(x/y)$的值，返回计算结果
cos	double cos(double x);	计算cos(x)的值，返回计算结果。 注意，x为弧度
cosh	double cosh(double x);	计算cosh(x)的值，返回计算结果
exp	double exp(double x);	计算e^x的值，返回计算结果
fabs	double fabs(double x);	计算x的绝对值，返回计算结果
oor	double　oor(double x);	计算不大于x的最大整数，返回计算结果
fmod	double fmod(double x, double y);	计算x除以y的浮点余数，返回计算结果。$x=i\times y+f$，其中i为整数，f与x有相同的符号，且f的绝对值小于y的绝对值，当y=0时，返回NaN
frexp	double frexp(double val, int *eptr);	把双精度数val分解为小数部分（尾数）x和以2为底的指数n（阶码），即val=$x\times 2^n$，n存放在eptr指向的内存中，函数返回小数部分x，$0.5\leqslant x<1$
log	double log(double x);	计算log$e$$x$，即ln$x$，返回计算结果。注意，$x>0$
log10	double log10(double x);	计算lg$x$，返回计算结果。注意，$x>0$
modf	double modf(double val, double *iptr);	把双精度数val分解为整数部分和小数部分，把整数部分存到iptr指向的内存中。返回val的小数部分

续表

函数名	函数原型	功能
pow	double pow(double base, double exp);	计算base为底的exp次幂，即$base^{exp}$，返回计算结果。 当base等于0而exp小于0时或者base小于0而exp不为整数时，结果错误。该函数要求参数base和exp以及函数的返回值为double型，否则有可能出现数值溢出问题
sin	double sin(double x);	计算sin*x*的值，返回计算结果。 注意，*x*为弧度
sinh	double sinh(double x);	计算sinh(*x*)的值，返回计算结果
sqrt	double sqrt(double x);	计算$\sqrt{x}$的值，返回计算结果。 注意，$x \geqslant 0$
tanh	double tanh(double x);	计算tanh(*x*)的值，返回计算结果

2. 字符处理函数

使用字符处理函数时，应在源文件中包含头文件ctype.h。字符处理函数如表2所示。

表 2　字符处理函数

函数名	函数原型	功能
isalnum	int isalnum(int ch);	检查ch是否为字母（Alphabet）或数字（Numeric）。是，则返回1；不是，则返回0
isalpha	int isalpha(int ch);	检查ch是否为字母。是，则返回1；不是，则返回0
iscntrl	int iscntrl(int ch);	检查ch是否为控制字符（ASCII码取值范围为0～31）。是，则返回1；不是，则返回0
isdigit	int isdigit(int ch);	检查ch是否为数字（0～9）。是，则返回1；不是，则返回0
isgraph	int isgraph(int ch);	检查ch是否为可打印字符（ASCII码取值范围为33～126，不包括空格符）。是，则返回1；不是，则返回0
islower	int islower(int ch);	检查ch是否为小写字母（a～z）。是，则返回1；不是，则返回0
isprint	int isprint(int ch);	检查ch是否为可打印字符（ASCII码取值范围为32～126，包括空格符）。是，则返回1；不是，则返回0
ispunct	int ispunct(int ch);	检查ch是否为标点字符（不包括空格符），即除字母、数字和空格符以外的所有可打印字符。是，则返回1；不是，则返回0
isspace	int isspace(int ch);	检查ch是否为空格符、跳格符（制表符）或换行符。是，则返回1；不是，则返回0
isupper	int isupper(int ch);	检查ch是否为大写字母（A～Z）。是，则返回1；不是，则返回0
isxdigit	int isxdigit(int ch);	检查ch是否为一个十六进制数字字符（即0～9、A～F，或a～f）。是，则返回1；不是，则返回0
tolower	int tolower(int ch);	将ch字符转换为小写字母。返回ch对应的小写字母
toupper	int toupper(int ch);	将ch字符转换为大写字母。返回ch对应的大写字母

3. 字符串处理函数

使用字符串处理函数时，应在源文件中包含头文件string.h。字符串处理函数如表3所示。

表 3　字符串处理函数

函数名	函数原型	功能
memcmp	int memcmp(const void *buf1, const void *buf2, unsigned int count);	比较buf1和buf2指向的数组的前count个字符。若buf1<buf2，则返回负数。若buf1=buf2，则返回0。若buf1>buf2，则返回正数
memcpy	void *memcpy(void *to, const void *from, unsigned int count);	从from指向的数组向to指向的数组复制count个字符，如果两数组重叠，不定义该数组的行为。函数返回指向to的指针
memmove	void *memmove(void *to, const void *from, unsigned int count);	从from指向的数组向to指向的数组复制count个字符；如果两数组重叠，则复制仍进行，但把内容放入to后修改from的指向。函数返回指向to的指针
memset	void *memset(void *buf, int ch, unsigned int count);	把ch的低字节复制到buf指向的数组的前count个字节处，常用于把某个内存区域初始化为已知值。函数返回指向buf的指针
strcat	char *strcat(char *str1, const char *str2);	把str2指向的字符串连接到str1指向的字符串后面，在新形成的str1指向的字符串后面添加一个'\0'，原str1指向的字符串后面的'\0'被覆盖。因无边界检查，调用时应保证str1指向的字符串的空间足够大，能存放原始str1和str2指向的两个字符串的内容。函数返回指向str1的指针
strcmp	int strcmp(const char *str1, const char *str2);	按字典序比较str1和str2指向的字符串。若str1<str2，则返回负数。若str1=str2，则返回0。若str1>str2，则返回正数
strcpy	char *strcpy(char *str1, const char *str2);	把str2指向的字符串复制到str1中去，str2必须是结束标志为'\0'的字符串的指针。函数返回指向str1的指针
strlen	unsigned int strlen(const char *str);	统计str指向的字符串中实际字符的个数（不包括字符串结束标志'\0'）。函数返回str指向的字符串中实际字符的个数
strncat	char *strncat(char *str1, const char *str2, unsigned int count);	把str2指向的字符串中不多于count个字符连接到str1指向的字符串后面，并以'\0'终止该字符串，原str1指向的字符串后面的'\0'被str2指向的字符串的第一个字符覆盖。函数返回指向str1的指针
strncmp	int strcnmp(const char *str1, const char *str2, unsigned int count);	按字典序比较str1和str2指向的字符串的不多于count个字符。若str1<str2，则返回负数。若str1=str2，则返回0。若str1>str2，则返回正数
strstr	char *strstr(char *str1, char *str2);	找出str2指向的字符串在str1指向的字符串中第一次出现的位置（不包括str2指向的字符串的字符串结束标志）。函数返回该位置的指针。若找不到，则返回空指针
strncpy	char *strncpy(char *str1, const char *str2, unsigned int count);	把str2指向的字符串中的count个字符复制到str1中，str2必须是结束标志为'\0'的字符串的指针。如果str2指向的字符串少于count个字符，则将'\0'加到str1指向的字符串的尾部，直到满足count个字符为止。如果str2指向的字符串长度大于count个字符，则str1指向的字符串不用'\0'结尾。函数返回指向str1的指针

注：根据C语言标准，size_t代表无符号整数类型。在某些编译器中，size_t代表unsigned int；而在另一些编译器中，size_t代表unsigned long。该类型被推荐用于定义表示数组长度或下标的变量。size_t类型的定义包含在头文件stddef.h中，而该头文件又常常包含在其他头文件中（如stdio.h）。

4. 缓冲文件系统的输入输出函数

使用缓冲文件系统的输入输出函数时，应在源文件中包含头文件stdio.h。缓冲文件系统的输入输出函数如表4所示。

表 4　缓冲文件系统的输入输出函数

函数名	函数原型	功能
clearerr	void clearerr(FILE *fp);	清除文件指针错误指示。函数无返回值
fclose	int fclose(FILE *fp);	关闭fp指向的文件，释放文件缓冲存储区。成功返回0，否则返回非0值
feof	int feof(FILE *fp);	检查文件是否结束。若遇文件结束符，则返回非0值，否则返回0。注意，在读完最后一个字符后，feof()并不能探测到文件尾，直到再次调用fgetc()执行读操作，feof()才能探测到文件尾
ferror	int ferror(FILE *fp);	检查fp指向的文件中的错误。若无错，则返回0。若有错，则返回非0值
ush	int ush(FILE *fp);	如果fp指向输出流，即fp所指向的文件是“写打开”的，则将输出缓冲区中的内容实际写入文件。若函数调用成功，则返回0；若出现写错误，则返回EOF。若fp指向输入流，即fp所指向的文件是“读打开”的，则fflush()函数的行为是不确定的。某些编译器（如Visual C++ 6.0）支持用fflush(stdin)来清空输入缓冲区中的内容，fflush()操作输入流是对C语言标准的扩充。但是并非所有编译器都支持这个功能（Linux下的GCC就不支持），因此使用fflush(stdin) 来清空输入缓冲区会影响程序的可移植性
fgetc	int fgetc(FILE *fp);	从fp指向的文件中获取下一个字符。函数返回所得到的字符；若读入出错，则返回FOF
fgets	char *fgets(char *buf, int n, FILE *fp);	从fp指向的文件读取一个长度为$n-1$的字符串，存入起始地址为buf的存储空间。函数返回地址buf；若遇文件结束符或出错，返回NULL。注意，与gets()不同的是，fgets()从指定的流读字符串，读到换行符时将换行符也作为字符串的一部分读到字符串中
fopen	FILE *fopen(const char *filename, const char *mode);	以mode指定的方式打开名为filename的文件。若成功，则返回一个文件指针。若失败，则返回NULL，错误代码在全局变量errno中
freopen	FILE *freopen(const char *filename, const char *mode, FILE *stream);	用于重定向输入输出流，以指定模式将输入或输出重定向到另一个文件。该函数可在不改变代码原貌的情况下改变输入输出环境。filename指定需重定向到的文件名或文件路径。mode指定文件的访问方式。stream指定需被重定向的文件流。如果函数调用成功，则返回指向该输出流的文件指针；否则返回NULL

续表

函数名	函数原型	功能
fprintf	int fprintf(FILE *fp, const char *format, …);	把输出列表中的数据值以format指向的格式输出到fp所指向的文件中。函数返回实际输出的字符数
fputc	int fputc(int ch, FILE *fp);	将字符ch输出到fp指向的文件中（尽管ch为int型，但只写入低字节）。若成功，则返回该字符；否则返回EOF
fputs	int fputs(const char *str, FILE *fp);	将str指向的字符串输出到fp所指向的文件中。若成功，则返回0；若出错则返回非0值。注意，与puts()不同的是，fputs()不会在写入文件的字符串末尾加上换行符
fread	int fread(char *pt, unsigned int size, unsigned int n, FILE *fp);	从fp指向的文件中读取长度为size的*n*个数据项，存到pt所指向的内存单元。函数返回所读的数据项个数，若遇文件结束符或出错，则返回0
fscanf	int fscanf(FILE *fp, char format, …);	从fp指向的文件中按format指定的格式将输入数据送到地址列表中的指针所指向的内存单元。函数返回已输入的数据个数
fseek	int fseek(FILE *fp, long offset, int base);	将fp指向的文件的位置指针移到以base所指的位置为基准、以offset为偏移量的位置。若成功，函数返回当前位置；否则，返回-1
ftell	long ftell(FILE *fp);	返回fp指向的文件中的读写位置
fwrite	unsigned int fwrite(const char *ptr, unsigned int size, unsigned int n, FILE *fp);	把ptr指向的n*size字节输出到fp指向的文件中。函数返回写到fp文件中的数据项的个数
getc	int getc(FILE *fp);	从fp指向的文件中读入一个字符。函数返回所读的字符；若遇文件结束符或出错，返回EOF
getchar	int getchar();	从标准输入设备读取并返回下一个字符。函数返回所读字符；若遇文件结束符或出错，返回-1
gets	char *gets(char *str);	从标准输入设备读入字符串，放到str指向的字符数组中，一直读到换行符或EOF，换行符不作为读入字符串的内容，而是变成'\0'后作为该字符串的结束标志。若成功，则返回str指针；否则返回NULL
perror	void perror(const char *str);	向标准错误流输出str指向的字符串，并随后附上冒号以及全局变量errno代表的错误消息的文字说明。函数无返回值
printf	int printf(const char *format, …);	将输出列表中的数据值输出到标准输出设备。函数返回输出字符的个数；若出错，则返回负数

续表

函数名	函数原型	功能
putc	int putc(int ch, FILE *fp);	把一个字符ch输出到fp所指向的文件中。函数返回输出的字符ch；若出错，则返回EOF
putchar	int putchar(char ch);	把字符ch输出到标准输出设备。函数返回输出的字符ch；若出错，则返回EOF
puts	int puts(const char *str);	把str指向的字符串输出到标准输出设备，将'\0'转换为回车符。若成功，则返回非负数；若失败，则返回EOF
rename	int rename(const char *oldname, const char *newname);	把oldname指向的文件名改为newname指向的文件名。若成功，则返回0；若出错，则返回1
rewind	void rewind(FILE *fp);	将fp指向的文件中的位置指针置于文件开头，并清除文件结束符。函数无返回值
scanf	int scanf(const char *format, …);	从标准输入设备按format指向的字符串规定的格式，输入数据给地址列表中的指针所指向的单元。以“%s”输入字符串，遇到空白字符（包括空格符、回车符、制表符）时，系统认为读入结束（但在开始读之前遇到的空白字符会被系统自动跳过）。 函数返回读入并赋给参数的数据个数。若遇文件结束符，则返回EOF；若出错，则返回0

5. 动态内存分配函数

使用与动态内存分配相关的函数时，应在源文件中包含头文件stdlib.h（有的编译系统要求包含malloc.h）。动态内存分配函数如表5所示。

表 5 动态内存分配函数

函数名	函数原型	功能
calloc	void *calloc(unsigned int n, unsigned int size);	分配n个数据项的连续存储空间，每个数据项的大小为size字节，与malloc()不同的是，calloc()能自动将分配的内存初始化为0。如果分配成功，则返回所分配的内存的起始地址；如果内存不够导致分配不成功，则返回NULL
free	void free(void *p);	释放p所指向的存储空间。函数无返回值
malloc	void *malloc(unsigned int size);	分配size字节的存储空间。如果分配成功，则返回所分配的内存的起始地址；如果内存不够导致分配不成功，则返回NULL
realloc	void *realloc(void *p, unsigned int size);	将p所指向的已分配内存区域的大小改为size字节。size字节可比原来分配的空间大或小。函数返回指向该内存区域的指针

6. 其他常用函数

其他常用函数如表6所示。

表 6　其他常用函数

函数名	函数原型及其所在的头文件	功能
atof	#include <stdlib.h> double atof(const char *str);	把str指向的字符串转换成双精度浮点值，字符串中必须含合法的浮点数。函数返回转换后的双精度浮点值
atoi	#include <stdlib.h> int atoi(const char *str);	把str指向的字符串转换成整型值，字符串中必须含合法的整型数。函数返回转换后的整型值
atol	#include <stdlib.h> long int atol(const char *str);	把str指向的字符串转换成长整型值，字符串中必须含合法的整型数。函数返回转换后的长整型值
exit	#include <stdlib.h> void exit(int code);	该函数使程序立即终止，并清空和关闭任何打开的文件。程序正常退出状态由code等于0或EXIT_SUCCESS表示，非0值或EXIT_FAILURE表示定义实现错误。函数无返回值
rand	#include <stdlib.h> int rand(void);	产生伪随机数序列。函数返回0～RAND_MAX的随机整数，RAND_MAX至少是32767
srand	#include <stdlib.h> void srand(unsigned int seed);	为函数rand()生成的伪随机数序列设置起点种子。函数无返回值
time	#include <time.h> time_t time(time_t *time);	调用时可使用空指针，也可使用指向time_t类型变量的指针，若使用后者，则该变量可被赋予日历时间。函数返回系统的当前日历时间；如果系统丢失时间设置，则函数返回-1
ctime	#include <time.h> char *ctime(const time_t *time);	把日期和时间转换为由年、月、日、时、分、秒等时间分量构成的用"YYYY-MM-DD hh:mm:ss"格式表示的字符串
clock	#include <time.h> clock_t clock(void);	clock_t其实就是long型。该函数的返回值是硬件滴答数，要换算成秒或者毫秒，需要除以CLK_TCK或者CLOCKS_PER_SEC。例如，在Visual C++ 6.0下，这两个量的值都是1000，表示硬件滴答1000次是1s，因此一个进程的时间是用clock()除以1000来计算的。 注意：本函数仅能返回毫秒级的计时精度

Sleep	#include <stdlib.h>
	Sleep(unsigned long second);
		在C语言标准中和Linux下，函数的首字母不大写，但在Visual C++和Code::Blocks环境下函数首字母要大写。Sleep()函数的功能是将进程挂起一段时间，起延时作用。参数的单位是毫秒

system	#include <stdlib.h> int system(char *command);	发出一个DOS命令。例如，system("CLS")可以实现清屏操作
kbhit	#include <conio.h> int kbhit(void);	检查当前是否有键盘输入，若有，则返回一个非0值，否则返回0
getch	#include <conio.h> int getch(void);	无须用户按Enter键即可得到用户的输入，只要用户按一个键，就立刻返回用户输入字符对应的ASCII码值，但输入的字符不会回显在屏幕上，出错时返回-1。该函数在游戏中比较常用，玩家输入字符后无须按Enter键，字符也不会在屏幕上回显

7. 非缓冲文件系统的输入输出函数

使用非缓冲文件系统的输入输出函数时，应在源文件中包含头文件io.h和fcntl.h，这些函数是UNIX系统的成员，不是由ANSI C定义的。非缓冲文件系统的输入输出函数如表7所示。

表7 非缓冲文件系统的输入输出函数

函数名	函数原型	功能
close	int close(int handle);	关闭handle指定的文件。若关闭失败，返回-1，errno说明错误类型；否则返回0
creat	int creat(const char *pathname, unsigned int mode);	专门用来建立并打开新文件，相当于访问权限为O_CREAT|O_WRONLY|O_TRUNC的open()函数。若成功，则返回一个文件句柄；否则返回-1，全局变量errno说明错误类型
open	int open(const char *pathname, int access, unsigned int mode);	以access指定的存取方式打开名为pathname的文件，mode为文件类型及权限标志，仅在access包含O_CREAT时有效，一般使用常数0666。若成功，则返回一个文件句柄；否则返回-1，全局变量errno说明错误类型
read	int read(int handle, void *buf, unsigned int len);	从handle指定的文件中读取len字节的数据存放到buf指针指向的内存。返回实际读入的字节数。0表示读到文件末尾；-1表示出错，errno说明错误类型
lseek	long lseek(int handle, long offset, int fromwhere);	从handle指定的文件中的fromwhere开始，移动位置指针offset字节。offset为正，表示向文件末尾移动；为负，表示向文件头部移动。移动的字节数是offset的绝对值。返回移动后的指针位置。-1L表示出错，errno说明错误类型
write	int write(int handle, void *buf, unsigned int len);	把从buf指向的字符串开始的len字节写入handle指向的文件。返回实际写入的字节数。-1表示出错，errno说明错误类型