C库函数总结

C 库函数主要指那些由美国国家标准协会（ANSI）或国际标准化组织(ISO)发布的标准中规定的库函数，按照标准 C 的要求来进行 C 语言编程是很重要的，因为这样你的代码才有可能跨平台使用。

ctype.h

ctype.h 这个头文件主要定义了一批 C 语言字符分类函数，所有的函数都只有一个参数，且参数和返回值均为 `int` 类型。下面是简单的函数介绍：

函数名	说明	函数名	说明	函数名	说明
isalpha	是否为字母	isdigit	是否为数字	isalnum	是否为数字或字母
iscntrl	是否为控制字符	isgraph	是否为图形文字	isupper	是否为大写字母
islower	是否为小写字母	tolower	转换为小写字母	toupper	转换为大写字母
isprintf	是否为可打印字符	ispunct	是否为标点符号	isspace	是否为空白
isxdigit	是否为十六进制的数字				

stdio.h

stdio.h 这个头文件应该是大多数人接触 C 语言的时候第一个认识的 C 库函数。这个函数最常用，但是也很复杂。它定义了三种类型，一些宏和很多的输入输出函数。

1. 定义的三种类型：
    1. `size_t` 是由 `sizeof` 关键字产生的无符号整类型。
    2. `FILE` 是一个结构体类型，记录了控制流需要的所有信息，包括它的文件定位符、指向相关缓冲的指针、记录是否发生了读/写错误的错误提示符和记录文件是否结束的文件结束符。
    3. `fpos_t` 包含可以唯一指定文件中的每一个位置所需的所有信息。
2. 定义了一些常量：

1. NULL 空值
2. _IOFBF 表示完全缓冲 _IOLBF 表示线缓冲 _IONBF 表示无缓存
3. BUFSIZ 是 setbuf 函数所使用的缓冲区的大小
4. EOF 是负整数,该表达式由几个函数返回来说明文件的结束，即一个流输入结束了(END OF FILE)
5. FOPEN_MAX (20)同时打开的文件的最大数量
6. FILENAME_MAX 文件名的最大长度
7. L_tmpnam 整数，最大长度的临时文件名
8. SEEK_CUR 取得目前文件位置 SEEK_END 将读写位置移到文件尾时 SEEK_SET 将读写位置移到文件开头
9. TMP_MAX 表示 tmpnam 函数可以生成的单独文件名的最大数目
10. `stderr 标准错误流，默认为屏幕, 可输出到文件 stdin 标准输入流，默认为键盘 stdout 标准输出流，默认为屏幕

3. 一些相关的函数
    1. 文件操作函数 （4个）
       int remove(const char *filename); 删除文件
       int rename(const char *old, const char *new); 重命名文件
       FILE *tmpfile(void); 创建一个临时的二进制文件，并通过模式“wb+”打开。
       char *tmpname (char *s) ; 生成一个字符串，这个字符串是一个有效的文件名。
    2. 文件访问函数 (6个)
       FILE* fopen(const char *filename, const char *mode); 打开名字为 filename 指向的文件，并且把这个文件和一个流相关联。
       FILE* freopen(const char *filename, const char *mode, FILE *stream); 打开名字为 filename 指向的文件，并且把它和 stream 指向的流关联在一起。
       int fclose(FILE *stream); 使 stream 指向的流被清空，并且和流相关联的文件被关闭。
       int fflush(FILE *stream); 对 stream 指向的流执行清空行为。
       void setbuf(FILE *stream, char *buf); 除了没有返回值，等价于函数 setvbuf 。
       void setvbuf(FILE *stream, char *buf, int mode, size_t size); 设定

`stream` 缓冲的方式。

3. 格式化的输入输出函数 （9个）

```
int fprintf(FILE *stream, const char *format, ...);
int printf(const char *format, ...);
int sprintf(char *s, const char *format, ...);
```

返回传送的字符的数目。

```
int fscanf(FILE *stream, const char *format, ...);
int scanf(const char *format, ...);
int sscanf(char *s, const char *format, ...);
```

如果在任何转换之前发生了输入失败，返回 `EOF`；否则返回赋值的输入项的数目。

```
int vfprintf(FILE *stream, const char *format, va_list arg);
int vprintf(const char *format, va_list arg);
int vsprintf(char *s, const char *format, va_list arg);
```

等价于对应的 `printf` 函数，不过可变参数表用 `arg` 代替。

4. 字符输入/输出函数 （11个）

`int fgetc(FILE *stream);` 从 `stream` 指向的输入流中读取下一个字符。

`int getc(FILE *stream);` 等价于函数 `fgetc` 。

`int getchar(void);` 等价于用参数 `stdin` 调用函数 `getc` 。

`int fputc(int c, FILE *stream);` 把字符写到指向的输出流中指定的位置处。

`int putc(int c, FILE *stream);` 等价于 `fputc`。

`int putchar(int c);` 等价于把stdout作为第二个参数调用的 `putc`。

`char *fgets(char *s, int n, FILE *stream);` 从 `stream` 指向的流中读取字符。

`int fputs(const char *s, FILE *stream);` 把 `s` 指向的串写入 `stream` 指向的流中。

`char *gets(char *s);` 从 `stdin` 指向的输入流中读取若干个字符，并将其保留到 `s` 指向的数组中，直到遇到文件结束符或者读取一个换行符。

`int puts(const char *s);` 把 `s` 指向的串写到 `stdout` 指向的流中，并且在输出最后添加一个换行符。

`int ungetc(int c, FILE *stream);` 把 `c` 指定的字符（转换为 `unsigned char` 类型）退回到 `stream` 指向的输入流中。

5. 直接输入/输出函数 （2个）

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

从 `stream` 指向的流中读取最多 `nmemb` 个元素到 `ptr` 指向的数组中 。

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE
```

`*stream);`

从 `ptr` 指向的数组中读取最多 `nmemb` 个元素并将其写到 `stream` 指向的流中。

6. 文件定位函数 (5个)

   `int fgetpos(FILE *stream, fpos_t *pos);`

   把 `stream` 指向的流的文件定位符的当前值存储到 `pos` 指向的对象中。

   `int fsetpos(FILE *stream, const fpos_t *pos);`

   根据 `pos` 指向的对象的值来设置 `stream` 指向的流的文件定位符，

   `long int ftell(FILE *stream);`

   获得 `stream` 指向的流的文件定位符的当前值。

   `int fseek(FILE *stream, long int offset, int whence);`

   为 `stream` 指向的流设置文件定位符。

   `void rewind(FILE *stream);`

   把 `stream` 指向的流的文件定位符设置在文件的开始位置，等价于 `(void)fseek(stream, 0L, SEEK_SET);`，只不过流的错误指示符也被清零。

7. 错误处理函数 (4个)

   `void clearerr(FILE *stream);`

   清空 `stream` 指向的流的文件结束符和错误指示符。

   `int feof(FILE *stream);`

   测试 `stream` 指向的流的文件结束符。当且仅当 `stream` 流设置了文件结束符时函数返回一个非0值。

   `int ferror(FILE *stream);`

   测试 `stream` 指向的流的错误指示符。当且仅当 `stream` 流设置了错误指示符时函数返回一个非0值。

   `void perror(const char *s);`

   把整数表达式 `errno` 中的错误编号转换为一条错误消息。

stdlib.h

是标准工具库。包含了C语言的中最常用的系统函数

1. 定义了五个宏
   1. `NULL` 空
   2. `EXIT_FAILURE` 失败状态码
   3. `EXIT_SUCCESS` 成功状态码
   4. `RAND_MAX` `rand` 的最大返回值
   5. `MB_CUR_MAX` 多字节字符中的最大字节数

2. 定义了四个类型：
    1. `size_t` 是由 `sizeof` 关键字产生的无符号整类型。
    2. `wchar_t` 是一个整型，标识一个宽字节字符，例如 `L'x'` 的类型就是 `wchar_t` 。
    3. `div_t` 是结构体类型 作为 `div` 函数的返回类型
    4. `ldiv_t` 是一个结构类型，是函数 `ldiv` 的返回值类型。
3. 定义了六类函数：
    1. 字符串函数
       `double atof(const char *nptr);` 将字符串转换成浮点型数
       `int atoi(const char *nptr);` 将字符串转换成整型数
       `long int atol(const char *nptr);` 将字符串转换成长整型数
       `double strtod(const char *nptr, char **endptr);` 将字符串转换成浮点数
       `long int strtol(const char *nptr, char **endptr, int base);` 将字符串转换成长整型数
       `unsigned long int strtoul(const char *nptr, char **endptr, int base);` 将字符串转换成无符号长整型数
    2. 内存控制函数
       `void *calloc(size_t nmemb, size_t size);` 配置内存空间
       `void free(void *ptr);` 释放原先配置的内存
       `void *malloc(size_t size);` 配置内存空间
       `void *realloc(void *ptr, size_t size);` 重新分配主存
    3. 环境函数
       `int atexit(void (*func)(void));`
       注册func指向的函数，该函数在程序正常终止的时候被调用。
       `void exit(int status);`
       使程序正常终止。
       `void abort(void);`
       使程序异常终止，除非捕获了信号SIGABRT且信号处理程序没有返回。
       `char *getenv(const char *name);`
       搜索宿主环境提供的环境表，来查找一个能和name指向的串匹配的串。
       `int system(const char *string);`
       把sring指向的串传递给宿主环境，然后命令处理程序按照实现定义的方式执行它。
    4. 搜索和排序函数

`void bsearch(const void *key, const void *base, size_t nmemb, size_t size, int (*compare)(const void , const void *));`
搜索一个拥有 `nmemb` 个元素的数组，来查找与 `key` 指向的对象匹配的元素，
`void qsort(void base, size_t nmemb, size_t size, int (*compare)(const void , const void *));`
根据 `compare` 所指向的比较函数将数组内容排列成升序。

5. 数学函数
   `int abs(int j);` 计算整型数的绝对值
   `long int labs(long int j);` 将两个整数相除, 返回商和余数
   `div_t div(int number, int denom);` 取长整型绝对值
   `ldiv_t ldiv(long int number, long int denom);` 两个长整型数相除, 返回商和余数
   `int rand(void);` 随机数发生器
   `void srand(unsigned int seed);` 设置随机数种子
6. 多字节函数
   `int mblen(const char *s, size_t n);` 根据locale的设置确定字符的字节数
   `int mbtowc(wchar_t *pwc, const char *s, size_t n);` 把多字节字符转换为宽字符
   `int wctomb(char *s, wchar_t wchar);` 把宽字符转换为多字节字符
   `size_t mbstowcs(wchar_t *pwcs, const char *s, size_t n);` 把多字节字符串转换为宽字符串
   `size_t wcstombs(char *s, const wchar_t *pwcs, size_t n);` 把宽字符串转换为多字节字符串

string

包含了C语言的最常用的字符串操作函数

1. 定义了一个宏： `NULL` 空
2. 定义了一个变量： `size_t`
3. 定义了五类函数：
   1. 名字以mem开头的函数对任意的字符序列进行操作。其中一个参数指向字符串的起始位置，另一个参数对元素的个数尽行计数。
      `void *memmove(void *s1, const void *s2, size_t n);`
      `void *memcpy(void *s1, const void *s2, size_t n);`
      memmove和memcpy均从s2指向的对象中复制n个字符到s1指向的对象中，返回

s1的值。不同之处在于，当复制发生在两个重叠的对象中，memcpy对这种行为未定义，memmove则可以正确执行。

```
int memcmp(const void *s1, const void *s2, size_t n);
```

比较s1和s2指向的对象中前n个字符。

```
void *memchr(const void *s, int c, size_t n);
```

返回指向定位的字符的指针，如果没有返回空指针。

```
void *memset(void *s, int c, size_t n);
```

把c的值复制到s指向的对象的前n个字符的每个字符中，返回s的值。

2. 名字以strn开头的函数对非空字符序列进行操作。

```
char *strcpy(char *s1, const char *s2);
```

包括终止的空字符，返回s1的值

```
char *strcat(char *s1, const char *s2);
```

包括终止的空字符，返回s1的值。

```
int strcmp(const char *s1, const char *s2);
```

比较s1和s2指向的串。

```
char *strchr(const char *s, int c);
```

终止的空字符被认为串的一部分。

```
char *strrchr(cosnt char *s, int c);
```

确定c在s指向的串中最后一次出现的位置。r意为right一侧。

```
char *strstr(const char *s1, const char *s2);
```

s2指向的串的字符序列在s1指向的串中第一次出现的位置。

```
size_t strspn(const char *s1, const char *s2);
```

计算s1指向的字符串中完全由s2指向的字符串中的字符组成的最大初始段的长度。也就是说，它从s1的开头搜索一个和s2中的任意元素都不匹配的字符，返回此字符的下标。

```
size_t strcspn(const char *s1, const char *s2);
```

查找两个字符串第一个相同的字符在s1中的位置，也就是The first char both in s1 and s2，如果没有则返回终止的空字符的索引。

strcspn和strchr很相似，但它匹配的是任意一个字符集而不是一个字符。

```
char *strpbrk(const char *s1, const char *s2);
```

确定s2指向的串中的任意的字符在s1中第一次出现的位置。

```
char *strtok(char *s1, const char *s2);
```

分解字符串为一组字符串。s1为要分解的字符串，s2为分隔符字符串。

3. 名字以str开头的函数对空字符结尾的字符序列进行操作。

```
char *strncpy(char *s1, const char *s2, size_t n);
```

从s2指向的数组中复制最多n个字符（包括空字符），不会在s1的末尾自动添加空字符；如果达到n个字符之前遇到了空字符，则复制完空字符后停止，并在s1指向的数组后面添加空字符，直到写入了n个字符。

简单来说，复制，直到遇到空字符或达到n个字符，如果此时复制的字符数未达到n，就填充空字符，直到字符数达到n。

`char *strncat(char *s1, const char *s2, size_t n);`

从s2指向的数组中将最多n个字符（空字符及其后面的字符不添加）添加到s1指向的串的结尾。在最后的结果后面加上一个空字符。

简单来说，cat，直到遇到空字符或达到n个字符，最后的结果一定以空字符结尾。

`int strncmp(const char *s1, const char *s2, size_t n);`

对s1和s2指向的数组中的最多n个字符进行比较（空字符后面的字符不参加比较）。

4. 其他类

   `size_t strlen(const char *s);`

   计算s指向的串的长度，不包括结尾的空字符。

   `char *strerror(int errnum);`

   将errnum中的错误编号对应到一个错误信息串。

5. 与区域设置有关

   `int strcoll(const char *s1, const char *s2);`

   功能同strcmp，只是比较时串都被解释为适合当前区域设置的类别LC_COLLATE的形式。

   `size_t strxfrm(char *s1, const char *s2, size_t n);`

   如果区域选项是"POSIX"或者"C"，那么strxfrm()同用strncpy()来拷贝字符串是等价的。

熟练程序员的层次：

assert.h

这个头文件只定义了一个宏： `assert`

如果条件为假，assert将输出一些信息并调用abort函数退出程序。

一个可能的实现：

```C++
#define NDEBUG
#include <assert.h>
int main(int argc, char* argv[]) {
int a = 12;
int b = 24;
assert( a > b );
printf("a is larger than b!");
```

```
return 0;
}
```

上面的程序会发现程序中止， `printf` 并未执行，且有这样的输出： `main: Assertion 'a > b' failed.` 原因就是因为 `a` 其实小于 `b` ，导致断言失败， `assert` 输出错误信息，并调用 `abort()` 中止了程序执行。

limits.h

头文件定义了整型变量的一些极限值和设置。

`CHAR_BIT` 一个ASCII字符长度 `SCHAR_MIN` 字符型最小值
`SCHAR_MAX` 字符型最大值 `UCHAR_MAX` 无符号字符型最大值
`CHAR_MIN` / `CHAR_MAX` char字符的最大最小值，如果char字符正被表示有符号整数。它们的值就跟有符号整数一样。否则char字符的最小值就是0，最大值就是无符号char字符的最大值。
`MB_LEN_MAX` 一个字符所占最大字节数 `SHRT_MIN` 最小短整型
`SHRT_MAX` 最大短整形 `USHRT_MAX` 最大无符号短整型
`INT_MIN` 最小整型 `INT_MAX` 最大整形
`UINT_MAX` 最大无符号整型 `LONG_MIN` 最小长整型
`LONG_MAX` 最大长整型 `ULONG_MAX` 无符号长整型

stddef.h

此头文件定义了3个类型和2个宏，其中一些在其他头文件中也有定义。

- 定义了三个类型
  `ptrdiff_t` 两个指针相减的结果的类型，有符号整型，一般来说是int或long的typedef。
  `size_t` 是sizeof操作符的结果的类型，无符号整型。
  `wchar_t` 是一个整型，标识一个宽字节字符，例如L'x'的类型就是wchar_t。
- 定义了两个宏
  `NULL` 空指针常量。
  `offsetof(type, member-designator)` 展开为一个size_t类型的整值常量表达式。

time.h

作时间的函数。

1. 定义了两个宏
    1. NULL
    2. CLOCKS_PER_SEC 使 clock 函数的返回值 CLOCKS_PER_SEC 的单位是秒
2. 定义了四个类型
   声明的类型有size_t、clock_t、time_t和struct tm。
    1. size_t 在 stddef.h 中已介绍过。
    2. clock_t 是 clock 函数的返回值类型
    3. time_t 是 time 函数的返回值类型，
    4. struct tm 保存了一个日历时间的各组成部分，比如年月日时分秒等。
3. 定义了两类函数
    1. 时间获取函数
       `time_t time(time_t *timer);`
       取得目前的时间
       `clock_t clock(void);`
       确定处理器使用的时间。
    2. 时间转换函数
       `struct tm *gmtime(const time_t *timer);` 把日期和时间转换为(GMT)时间
       `struct tm *localtime(const time_t *timer);` 取得当地目前时间和日期
       `char *ctime(const time_t *timer);` 把日期和时间转换为字符串
       `time_t mktime(struct tm *timeptr);` 将时间结构数据转换成经过的秒数
       `char *asctime(const struct tm *timeptr);` 将时间和日期以字符串格式表示
       `size_t strftime(char *s, size_t maxsize, const char *format, const struct tm *timeptr);` 将时间格式化
    3. 其他函数
       `double difftime(time_t time1, time_t time0);`
       计算两个日历时间之差： `time1-time0,` 返回以秒为单位的差值。这个函数似乎简单到没有存在的必要，直接执行减法不就得了。

优秀程序员的层次：

float.h

float 头文件类似 limit 头文件，主要是浮点型数值的相关定义。

在所有实例里 `FLT` 指的是 `float`， `DBL` 是 `double`， `LDBL` 指的是 `long double`.

变量	定义
FLT_ROUNDS	定义浮点型数值四舍五入的方式，-1是不确定，0是向0，1是向最近，2是向正无穷大，3是负无穷大
FLT_RADIX 2	定义指数的基本表示（比如base-2是二进制，base-10是十进制表示法，16是十六进制）
FLT_MANT_DIG，DBL_MANT_DIG，LDBL_MANT_DIG	定义数值里数字的个数
FLT_DIG 6，DBL_DIG 10，LDBL_DIG 10	在四舍五入之后能不更改表示的最大小数位
FLT_MIN_EXP，DBL_MIN_EXP，LDBL_MIN_EXP	FLT_RADIX 的指数的最小负整数值
FLT_MIN_10_EXP -37，DBL_MIN_10_EXP -37，LDBL_MIN_10_EXP -37	10进制表示法的的指数的最小负整数值
FLT_MAX_EXP ,DBL_MAX_EXP ,LDBL_MAX_EXP	FLT_RADIX 的指数的最大整数值
FLT_MAX_10_EXP +37 ,DBL_MAX_10_EXP ,LDBL_MAX_10_EXP +37 +37	10进制表示法的的指数的最大整数值
FLT_MAX 1E+37，DBL_MAX 1E+37，LDBL_MAX 1E+37	浮点型的最大限
FLT_EPSILON 1E-5，DBL_EPSILON 1E-9，LDBL_EPSILON 1E-9	能表示的最小有符号数

math.h

math.h是 C 语言内的数学函数库。

函数	含义	函数	含义
三角函数			
double sin(double x);	正弦	double cos(double x);	余弦
double tan(double x);	正切	*cot三角函数，可以使用tan(PI/2-x)来实现。	

函数	含义	函数	含义
反三角函数			
double asin(double x);	结果介于[-PI/2, PI/2]	double acos(double x);	结果介于[0, PI]
double atan(double x);	反正切(主值), 结果介于[-PI/2, PI/2]	double atan2(double y,double);	反正切(整圆值), 结果介于[-PI, PI]
双曲三角函数			
double sinh(double x);	计算双曲正弦	double cosh(double x);	计算双曲余弦
double tanh(double x);	计算双曲正切		
指数与对数			
double exp(double x);	求取自然数e的幂	double sqrt(double x);	开平方
double log(double x);	以e为底的对数	double log10(double x);	以10为底的对数
double pow(double x, double y);	计算以x为底数的y次幂	float powf(float x, float y);	与pow一致，输入与输出皆为浮点数
取整			
double ceil(double);	取上整	double floor(double);	取下整
标准化浮点数			
double frexp(double f, int *p);	标准化浮点数, f = x * 2^p, 已知f求x, p (x介于[0.5, 1])	double ldexp(double x, int p);	与frexp相反, 已知x, p求f
取整与取余			
double modf(double, double*);	将参数的整数部分通过指针回传, 返回小数部分	double fmod(double, double);	返回两参数相除的余数