

HOPE

Turbo C++ 运行库函数 源程序与参考大全 下册

(共二册)



中国科学院希望高级电脑技术公司

73·87221072

1013358/3

Turbo C++ 运行库函数 源程序与参考大全(下册)

亦 鸥 等编



- MATH 库函数源程序
- EMU 库函数源程序
- 头文件和.ASI 文件
- 修改库函数源程序
- 使用批处理文件来建库



上册的内容为

☐ CLIB 库函数源程序

中国科学院希望高级电脑技术公司

一九九一年五月

Turbo C++运行库函数源程序与参考大全

《Turbo C++运行库函数源程序与参考大全》整理编译了 Turbo C++运行库中所有函数的源程序以及详细的函数使用说明。它不仅全部提供了《TurboC++参考手册》中提供的函数，而且还提供了实现这些函数的所有低层支持函数。这使得 Turbo C++运行库成为真正的开放式结构，用户可以从多方面高效地使用库函数。

不仅如此，有了库函数源程序，用户就可以对库函数进行单步断点等符号调试，可以领会 Turbo C++运行库以及整个软件的设计策略和技巧，为将来开发语言系统提供实例模型。Turbo C之所以能风靡当今 PC 世界，不仅因为它有良好的用户界面，强大的符号调试功能，还因为有设计精良、运行高速、结构优化的运行库函数。运行库函数源程序让用户能目睹其真正风采，领略其库设计策略、技巧、程序设计风格、函数依赖关系等等。

有了库函数源程序，就可以无限制地把 Turbo C 与程序设计语言揉合起来，综合 C 与其它程序设计语言的优点进行编程，使程序设计更上一层楼。

对于高级程序员来说，通过修改库函数源程序，可更高效地、多方位地使用 Turbo C++，对于一般的编程者和初学 C 的人，它又是一个很好的实例教材，是进行程序设计的良师益友。

齐全的运行库函数与低层支持函数，将给用户带来无穷无尽的新发现与高效率。

目 录

前言

第〇章 Turbo C++ 运行库概览.....1

第一章 CLIB 库函数源程序.....14

abs.cas	14
absread.cas	14
access.c	17
allocmem.cas	18
assert.c	19
atexit.c	20
atol.cas	20
bcd1.c	22
bcd2.cpp	27
bdos.c	27
bdosptr.cas	28
bioscom.cas	29
biosdisk.cas	30
biosequ.cas	32
bioskey.cas	33
biosprin.cas	34
brk.cas	35
bsearch.c	37
calloc.c	38
cgets.c	39
chdir.cas	39
chmod.c	41
chmoda.cas	42
chsize.cas	42
clearerr.c	45
clock.cas	45
close.c	46
closea.cas	47
closeall.c	47
clreol.c	48
clrscr.c	48

color.c	48
coreleft.cas	51
country.cas	52
cplx1.cpp	53
cplx2.cpp	54
cprintf.c	57
cputs.c	59
creat.cas	59
creata.cas	61
crtinit.cas	63
cscanf.c	66
ctime.c	67
ctrlbrk.c	72
ctype.c	73
cursor.c	74
cvtfak.asm	74
dbp.cpp	75
delay.cas	76
divt.cas	77
doscnd.c	78
dosenv.c	79
dosexr.cas	80
dostimu.cas	81
dup2.cas	83
eof.cas	84
exec.asm	87
execl.c	94
execle.c	95
execlp.c	95
execlpe.c	96
execv.c	96
execve.c	97
execvp.c	97
execvpe.c	97
exit.c	98
farheap.asm	98

fbrk.c	108
fcalloc.cas	110
fclose.c	111
corelft.c	112
fflush.c	112
fgetpos.c	113
fgets.c	114
feapchk.asm	114
filebuf.cpp	121
files.c	123
files2.c	124
findfirs.cas	124
flength.cas	126
flushall.c	127
fmode.c	128
fnmerge.c	128
fnsplit.c	129
fopen.c	132
format.cpp	135
fprintf.c	137
fpstklen.c	137
fputs.c	138
fread.c	138
freemem.cas	140
fscanf.c	140
fseek.c	141
fsetpos.c	143
fstat.cas	143
fstream.cpp	145
ftime.c	152
fwrite.c	153
f_lxmul.asm	154
f_pcmp.asm	155
getc.cas	155
getcbrk.c	158
getch.cas	159
getcurdi.cas	160
getcwd.c	161
getdate.c	162
getdta.cas	163

getenv.cas	163
getfat.cas	165
getftime.cas	167
getpass.c	168
getpsp.c	169
gets.c	169
getswit.c	170
getvect.cas	171
getveri.cas	172
getw.c	172
gotoxy.c	173
gfreemem.c	173
ggetmem.c	174
gotoxy.c	174
gptext.c	175
gregstr.c	176
harderr.cas	177
heaplen.c	179
h_ldiv.asm	179
h_llsh.asm	181
h_lrsh.asm	182
h_lursh.asm	183
h_pada.asm	183
h_padd.asm	185
h_pina.asm	186
h_psbp.asm	187
h_scopy.asm	188
h_spush.asm	189
inport.cas	190
insline.c	191
int86.cas	191
intdos.cas	193
intr.cas	195
ioctl.cas	197
ioerror.cas	198
ios.cpp	201
iostream.cpp	204
is.cas	206
isatty.cas	208
istream.cpp	208

istreamf.cpp	212	ostreamn.cpp	277
istreami.cpp	215	ostreamx.cpp	281
istreamn.cpp	218	ostrf.cpp	281
istreamx.cpp	223	outport.cas	281
istrf.cpp	224	parsfnm.cas	282
kbhit.cas	225	peek.c	283
keep.c	226	perror.c	284
labs.c	226	poke.c	285
ldivt.cas	227	printf.c	286
loadprog.c	228	pureerr.cpp	290
locale.c	230	putc.c	290
lock.cas	230	putch.c	293
lrotl.cas	232	putenv.cas	294
lrotr.cas	232	puts.c	297
lsearch.c	233	putw.c	297
lseek.cas	234	qsort.cas	298
ltoa.cas	235	rand.c	301
manip.cpp	237	randblk.cas	302
memccpy.cas	239	read.cas	303
memchr.cas	240	reada.cas	305
memcmp.cas	241	realcvt.asm	306
memcpy.cas	242	rename.cas	306
memicmp.c	243	rewind.c	307
memset.cas	243	rmdir.cas	307
mkdir.cas	244	rotl.cas	308
mktemp.c	245	rotr.cas	308
movedata.cas	246	scanf.c	309
movetext.c	247	scanner.cas	313
movmem.cas	247	scantod.asm	330
multbyte.c	249	scantol.cas	331
nearheap.asm	250	screen.c	336
newdel.cpp	257	scroll.c	338
nheapchk.asm	258	searchp.cas	340
n_lxmul.asm	263	segread.cas	343
n_pcmp.asm	264	setargv.asm	344
open.cas	264	setblock.cas	355
opena.cas	268	setbuf.c	355
ostream.cpp	269	setdate.cas	356
ostreamf.cpp	270	setdta.dta	357
ostreami.cpp	275	setenvp.asm	357

setenvp2.c	359	strlwr.c	403
setftime.cas	360	strmbfn.cpp	404
setjmp.cas	360	strncat.c	407
setmode.c	363	strncmp.cas	407
setupio.c	364	strncpy.cas	408
setvbuf.c	364	strnicmp.cas	409
sigdata.c	365	strnset.c	410
signal.c	365	strpbrk.c	411
siosync.cpp	371	strrchr.c	411
sleep.c	372	strrev.c	412
sound.cas	372	strset.c	413
spawn.asm	374	strspn.c	413
spawnl.c	376	strstr.cas	414
spawnle.c	378	strstrea.cpp	415
spawnlp.c	378	strtok.c	420
spawnlpe.c	379	ctrtol.c	421
spawnv.c	379	strtoul.c	422
spawnve.c	380	strupr.c	423
spawnvp.c	380	strxfrm.c	424
spawnvpe.c	381	swab.c	425
sprintf.c	381	system.c	425
sscanf.c	382	tell.c	427
stack.asm	384	textmode.c	427
stat.cas	384	timecvf.c	428
stdfile.cpp	387	tmpfile.c	430
stdiostl.cpp	387	tmpnam.c	430
stime.c	387	tolower.c	431
stklen.c	390	toupper.c	432
stpcpy.c	390	tzset.cas	432
strcat.c	391	umask.c	436
strchr.cas	395	ungetc.cas	436
strcmp.cas	396	unlink.cas	437
strcoll.c	397	vapp.cpp	438
strcpy.cas	397	vappv.cpp	438
strcspn.c	398	vdel.cpp	439
strdup.c	398	vfprintf.c	440
streambf.cpp	399	vfscanf.c	441
strerror.c	400	vidinfo.c	441
stricmp.cas	401	vnew.cpp	442
strlen.cas	402	vneww.cpp	443

vprinter.cas	444	ldexp.cas	495
vprintf.c	458	ldtrunc.cas	497
vram.cas	459	log.cas	499
vscanf.c	460	log10.cas	500
wherexy.c	461	matherr.c	501
wild.asm	462	modf.cas	503
wildargs.asm	462	poly.cas	504
window.c	462	pow.cas	505
write.c	462	pow10.cas	510
writea.cas	464	realcvt.cas	513
wscroll.c	465	scantod.cas	521
xclose.c	465	sin.cas	531
xfclose.c	466	sinh.cas	533
xfflush.c	466	sqrt.cas	534
zapctrlz.cas	466	stat87.cas	535
		strtod.c	536
		tan.cas	537
		tanh.cas	539
		xcvt.cas	540
		_matherr.c	547
第二章 MATH 库函数源程序	469		
acosasin.cas	469	第三章 EMU 库函数源程序	550
atan.cas	472	EMURULES.ASI	550
atan2.cas	473	FPINIT.ASM	559
atof.c	475		
ceil.cas	476	第四章 头文件和.ASI 文件	572
clear87.cas	477	ASMRULES.H	572
cos.cas	478	_DIR.H	577
cosh.cas	479	_FHEAP.H	578
ctrl87.cas	480	_GRAPH.H	578
difftime.c	482	_HEAP.H	584
efcvt.c	482	_IO.H	584
exp.cas	484	_MATH.H	585
fabs.cas	486	_PRINTF.H	588
flags87.asm	486	_PROCESS.H	589
floor.cas	487	_SCANF.H	590
fmod.cas	488	_STDIO.H	591
fperr.c	489	_VIDEO.H	592
fpreset.c	491	_HEAP.INC	593
frexp.cas	491	EMUVARS.ASI	593
ftol.asm	492		
gcvt.c	493		
hugeval.c	494		
hypot.cas	494		

EQUATES.ASI.....	596
RULES.ASI.....	597

第五章 修改库函数源程序.....607

第六章 使用批处理文件来建库.....	608
CLIB.BAT.....	609

CLIBREPL.BAT.....	612
CLIBRLIB.BAT.....	616
MATH.BAT.....	618
MATHREPL.BAT.....	621
MATHRLIB.BAT.....	626
CLIBRCMP.BAT.....	628
MATHRCMP.BAT.....	630

```

/*
#pragma inline
#include <stdlib.h>
/*
    rotr 将一个值向右循环移位
    用法 unsigned _rotr(unsigned value ,int count);
    原型在  stdlib.h
    说明 见 _rotl
*/

#undef _rotr /* not an intrinsic */
unsigned _rotr(unsigned val, int rotate_count)
{
asm      mov    ax, val
asm      mov    cx, rotate_count
asm      ror    ax, cl
      return    _AX;
}
*/

* 文件名 - scanf.c
*
* 函数
*      scanf - 从标准输入取格式化输入
*/

#include <stdarg.h>
#include <stdio.h>
#include <_stdio.h>
#include <_scanf.h>
#undef  ungetc /* remove the macro version */
/*

scanf 执行格式化输入
用法 int scanf(char *rformat[,argument,...]);
相关函数 int cscanf(char *format[,argument,...]);
用法 int fscanf(FILE *stream,char *format [,argument,...]);
      int sscanf(char *string,char *format[,argument,...]);
      int vfscanf(FILE *stream,char *format,va_list argp);
      int vscanf(char *format,va_list argp);
      int vsscanf(char *string,char *format,va_list argp);

原型在  stdio.h
说明  ...scanf 系列函数扫描输入字段，一次一个字符；再按一定格式转换。这些功能有：
    * 接受一格式字符串确定输入字段的解释方式（由用法中的 format 给定）
    * 为了格式输入，对数量可变的输入字段提供格式字符串
    * 将格式后的输入存储在格式字符串后的参数给出的地址（这些地址在用法中以"argument"
      和 va_list param 形式给出
    当...scanf 函数在格式字符串中遇到它的第一个格式指示符时，它根据该指示符扫描并转换第
    一个输入字段，然后把结果存在由第一个地址参数给定的位置中；接着转换并存储第二个输
    入字段，第三个...等等。
    ...scanf 中三个函数的输入流是隐含的
    scanf 和 vscanf 都从 stdin 接受输入
    cscanf 直接从控制台接受输入
    另四个...scanf 函数也使用另一个参数（参数表中的第一个参数），这个附加参数指定输入
    源：
    fscanf 和 vfscanf 从流（由 stream 所指）中接受输入
    sscanf 和 vsscanf 从内存字符串（由 stream 所指）中接受输入。
    ...scanf 的四个函数直接从函数调用中设置地址(scanf, cscanf, fscanf, sscanf)

```

其余三个函数(vscanf,vfscanf,vsscanf)从可变参数表中得到地址参数。v...scanf 函数为...scanf 函数的可选择入口点。

有关参数表的更详细信息，见 va...的定义。

以下对每一...scanf 函数作一总结：

- scanf 从stdin读数据，然后把它存在地址参数&arg1,...,&argn给定的位置中
- cscanf 直接从控制台读数据，然后把它存在地址参数&arg1,...,&argn给定的地址中
- fscanf 从指定名输入流中读数据，然后把它存在地址参数&arg1,...,&argn给定的地址中
- sscanf 读数据（存在字符串string中）到地址参数&arg1,...,&argn给定的位置，不改变源串
- vscanf 同scanf相似，只是从va_arg数组va_list param中接受地址参数
- vfscanf 同fscanf相似，只是它从va_arg数组中接受地址参数
- vsscanf 同sscanf相似，只是从va_arg数组va_list param中接受地址

1. 格式字符串：

在每一...scanf 函数调用中出现的格式字符串用于控制函数扫描、转换和存储输入字段的方式。对于给定的格式指示符，必须有足够的地址参数；否则的话，结果是不可预测的，也可能是灾难性的。多余的地址参数（多于格式所要求的）被忽略。

格式字符串是包含三种类型目标的字符串：空白字符，非空白字符和格式指示符

- 空白字符为空格()、制表(\t)或换行符(\n)。如果一个...scanf 函数在格式字符串中遇到一个空白字符，它将读但不存所有连续的空白字符直到输入中出现非空白字符
- 非空白字符是除了百分号(%)外的所有其它 ASCII 字符。如果一个...scanf 函数在格式字符串中遇到一个非空白字符，它将读但不存匹配的非空白字符
- 格式指示符控制...scanf 函数读入并转换输入字段中的字符为给定类型的值，然后把它们存在由地址参数给出的位置上

末尾的空格（包括换行符）不读，除非在格式字符串中显式匹配。

2. 格式指示：

...scanf 格式指示符有以下形式：

[%*] [width] [F/N] [h/l] type_character

每一格式指示以百分号(%)开始，接着是如下顺序：

- * 一个可选的赋值抑制字符 [*]
- * 一个可选的宽度指示符 [width]
- * 一个可选的指针大小指示符 [F/N]
- * 一个可选的参数类型修饰符 [h/l]
- * 类型字符

3. 可选的格式字符串成份：

以下是由...scanf 格式字符串中可选的字符和指示符所控制的输入格式的一般形式：

字符或指示符	控制或指示
*	控制下一个输入字段的赋值
width	可读入的最多字符数。如果...scanf函数遇到了空白字符或不可转换字符，读入字符将减少
size	覆盖地址参数的缺省大小（N=近指针，F=远指针）
类型参数	覆盖地址参数的缺省类型（h=指向短整型的指针,l=指向长整型的指针）

4. ...scanf 类型字符：

下表列出...scanf 类型字符每个要求的输入类型和输入的存储格式。

本表假定在格式指示符中没有包括可选的字符、指示符或修饰符(*,width,size)。如果想知道可选择成份是怎样影响...scanf 输入的，请看接下来的那张表。

类型字符	输入	参数类型
		数值
d	十进制数	指向整型的指针(int *arg)
D	十进制数	指向长整型的指针(long *arg)

o	八进制数	指向整型的指针(int *arg)
O	八进制数	指向长整型的指针(long *arg)
i	十、八或十六进制数	指向整型的指针(int *arg)
I	十、八或十六进制数	指向长整型的指针(long *arg)
u	无符号十进制数	指向无符号整型的指针(undigned int *arg)
U	无符号十进制数	指向无符号长整型的指针(unsigned long *arg)
x	十六进制数	指向整型的指针(int *arg)
X	十六进制数	指向长整型的指针(long *arg)
e	浮点数	指向浮点型的指针(float *arg)
E	浮点数	指向双精度型的指针(fouble *arg)
f	浮点数	指向浮点型的指针(float *arg)
g	浮点数	指向浮点型的指针(float *arg)
G	浮点数	指向双精度型的指针(double *arg)

		字符
s	字符串	指向字符数组的指针
c	字符	指向单个字符的指针, 如果同时给定字段宽(如%5c), 将为指向w维字符数组的指针
%	%字符	不作转换, 存字符%
		指针
n	无	指向整型的指针(int *arg), 成功读入的字符(到%n为止)数被存到此指针中
p	以YYYY:ZZZZ或ZZZZ形式表示的十六进制数	指向对象的指针(far* 或 near*) 转换缺省为存储模式的指针大小

5. 输入字段:

下列之任何一个为输入字段:

- * 下一空白字符前的所有字符(不包括空白字符)
- * 在当前格式指示下不能转换的第一个字符前的所有字符(如八进制格式下的8或9)
- * 到n个字符, 其中n为字段指示宽度

6. 约定

格式指示符中有些约定, 总结如下:

%c 转换:

这个指示符读下一个字符(包括空白字符), 为了跳过空白字符, 读下一个非空白字符, 可使用%1s。

%wc 转换(w=宽度指示):

地址参数是一字符数组指针, 该数组包含w个元素(char arg[w])

%s 转换:

地址参数是一指向字符数组的指针(char arg[])

数组大小至少为n+1个字节, 其中n为字符串s的长度。用空格或换行结束输入字段, 在数组的最后一个元素后自动加上一空字符终止符。

%[search_set]转换:

由方括号括起来的字符集合可用s类型字符替代。地址参数是一个指向字符数组的指针(char arg[])。

方括号内包含了一个可能字符的搜索集合组成的字符串(输入字段)。

如果括号内的第一个字符^, 则搜索集合为除去括号内字符的所有其它ASCII字符。

输入字段是不由空格分界的字符串。...scanf函数读相应的输入字符, 直到遇上不在搜索集合(或相反搜索集合)中的第一个字符为止。以下是这种转换类型的两个例子:

%[abcd] 搜索输入字段中所有的a, b, c, d字符

%[^abcd] 搜索输入字段中除a, b, c, d外的其它字符

在搜索集合中, 可以使用“范围设置”来定义一个字符(数字或字母)的范围。如为搜索所有的数字, 可以使用:

`%[0123456789]`

也可以使用:

`%[0-9]`

搜索字母或数字, 可使用:

`%[A-Z]` 搜索所有大写字母

`%[0-9A-Za-z]` 搜索所有数字, 字母(大小写)

`%[A-FT-Z]` 搜索从A到F, T到Z的大写字母

控制这些搜索集合范围的规则是很直观的:

- 短横线(-)前的字符必须小于后字符
- 短横线不能是最前或最后字符(否则就被认为是字符短横线,而不是范围指示符)
- 短横线两边字符必定是范围的起始和终止值

以下列出短横线仅是字符或是范围指示符的一些例子:

`%[-+*/]` 四个算术操作符

`%[z-a]` 字符'z','.', 'a'

`%[+0-9-A-F]` 字符'+','.',范围0到9,A到Z

`%[+0-9A-F]` 同上

`%[^0-9+A-F]` 除了'+','.',范围0到9,A到Z外的所有字符。

`%e,%E,%f,%g,%G`(浮点转换)

输入字段中的浮点数具有下列一般格式:

`[+/-]ddddddddd[.]ddd[E|e][+|-]ddd`

其中[]内为任选项,ddd 代表十进制,八进制或十六进制数字。

`%d,%i,%o,%x,%D,%I,%O,%X,%c,%n` 转换

在允许使用指向字符,整型或长整型的转换中,都可使用指向无符号字符、无符号整型或无符号长整型的指针。

7 赋值抑制字符:

赋值抑制字符为星号(*),不要把它与C的间接操作符(指针)相混淆(也为星号)

如果格式指示中有*号跟在%号后,下一输入字段被扫描但不赋给下一地址参数。被抑制的输入数据类型假定为跟在*号的类型字符所指定的类型。后继的文字匹配和赋值抑制不能直接决定。

8 宽度指示符:

宽度指示符(n)为一个十进制整数,它控制从当前输入字段中所读入的最多字符个数。

如果输入字段的字符个数少于n,...scanf 函数读字段中的所有字符,然后是下一字段和格式指示符

如果在读入给定宽度字符前遇到了空白字符或不可转换字符,已读的字符被转换、存储,接着函数处理下一格式指示符。

不可转换字符是按给定格式不能转换的字符(如八进制格式下的8或9,十六进制或十进制下的J或K等)

宽度指示符 所存输入宽度的影响

n 读,转换n个字符,并存到当前地址参数中

9.输入大小与参数类型修饰符

输入大小修饰符(N与F)和参数类型修饰符(h与l)影响...scanf 函数对相应地址参数arg的解释。

F和N覆盖arg缺省或声明的大小。

h和l指明下面的输入数据使用什么类型(h=short,l=long)。输入数据将被转换成指定的类型,输入数据的arg应指向对应大小的对象(%h为short,%l为long或double对象)

修饰符	转换影响
F	覆盖缺省或声明的大小 arg解释为远指针
N	覆盖缺省或声明的大小 arg解释为近指针
-	不能在巨型模式中使用
n	对d,i,o,u,x类型:转换输入为短整型,把它存在短整型对象中 对D,I,O,U,X类型:无影响

对e,f,c,s,n,p类型:无影响

对d,i,o,u,x类型:转换输入为长整型,存在长整型对象中

对e,f类型:转换输入为双精度型,存在双精度对象中

对D,I,O,U,X类型:无影响

对c,s,n,p类型:无影响

10. ...scanf 函数何时停止扫描

由于多种原因,...scanf 函数在遇到正常字段结束符(空白字符)前,可能停止扫描一特定字段或完全终止。

在下列情况下, ...scanf 函数停止扫描和存储当前字段,而进入对下一字段的处理:

- * 在格式指示符的百分号后出现赋值抑制字符(*),当前输入字段被扫描但不存储
- * 已读了 width 个字符(width 为宽度指示符,它是格式指示中的一个十进制正整数)
- * 在当前格式下不能转换下一读入字符(如在十进制格式下出现 A)。
- * 输入字段中的下一个字符没有在搜集集合中出现(或在相反搜索集合中出现)
- * 当出现以上情况, ...scanf 函数停止扫描当前输入字段的首字段时,下一个字符假定未读,被当作后一输入字段的首字符,或当作输入后继续读操作的首字符。

在下列情况下, ...scanf 终止扫描:

- * 输入字段中的下一个字符与格式字符串中的相应空白字符冲突
- * 输入字段中的下一字符为 EOF
- * 格式字符串用完

如果在格式字符串中出现一个不是格式指示部分的字符序列,它必须与输入字段的当前字符序列相匹配, ...scanf 函数扫描但不存储匹配字符。当出现冲突字符时,仍保留在输入字段中,就象没有读过一样。

返回值 所有的...scanf 函数都返回成功扫描、转换和存储的输入字段数,被扫描但不存储的字段不算在内。

如果这些函数试图在文件末尾读(对于 sscanf 和 vsscanf,为字符串尾),返回 EOF。

如果没有字段被存储,返回 0

可移植性 函数 scanf, fscanf, sscanf 和 cscanf 适用于 UNIX 系统,在 Kernighan 和 Richie 书中作了定义。

参见 atof,getc,printf

```
*/
int cdecl scanf(const char *fmt, ...)
{
    return( _scanner( (int near (*)(void *))_Nfgetc,
                      (void near (*)(int, void *))_Nungetc,
                      stdin,
                      fmt,
                      _va_ptr ) );
}

/*-----
* 文件名 - scanner.cas
*
* 函数
*   _scanner - 读格式化输入
*-----*/

#pragma inline
#include <asmrules.h>
#include <stdio.h>
#include <stdlib.h>
#include <scanf.h>
#define I asm
typedef enum
{
    _ZZ, /* terminator */
}
```

```

_ws,      /* space */
_dc,      /* dont care */
_pc,      /* percent */
_su,      /* suppress */
_nu,      /* numeral */
_ch,      /* character */
_de,      /* decimal */
_un,      /* unsigned decimal - same as decimal */
_in,      /* general int */
_fl,      /* float */
_ld,      /* long double */
_ha,      /* half */
_lo,      /* long */
_oc,      /* octal */
_st,      /* string */
_sc,      /* scanset */
_ct,      /* count of characters scanned */
_he,      /* hexadecimal */
_pt,      /* pointer */
_ne,      /* near */
_fa,      /* far */

```

charClass;

```

static const char scanCtype [128] =
{
/* NUL SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI */
_zz_dc_dc_dc_dc_dc_dc_dc_dc_ws_ws_ws_ws_ws_dc_dc,
/* DLE DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US */
_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc_dc,
/* SP ! " # $ % & ' ( ) * + , - . / */
_ws_dc_dc_dc_dc_pc_dc_dc_dc_dc_su_dc_dc_dc_dc_dc,
/* 0 1 2 3 4 5 6 7 8 9 : ; < = > ? */
_nu_nu_nu_nu_nu_nu_nu_nu_nu_dc_dc_dc_dc_dc_dc,
/* @ A B C D E F G H I J K L M N O */
_dc_dc_dc_dc_de_fl_fa_fl_ha_in_dc_dc_ld_dc_ne_oc,
/* P Q R S T U V W X Y Z [ \ ] ^ _ */
_dc_dc_dc_dc_dc_un_dc_dc_he_dc_dc_sc_dc_sc_dc_dc,
/* ` a b c d e f g h i j k l m n o */
_dc_dc_dc_ch_de_fl_fl_fl_ha_in_dc_dc_lo_dc_ct_oc,
/* p q r s t u v w x y z { | } ~ DEL */
_pt_dc_dc_st_dc_un_dc_dc_he_dc_dc_dc_dc_dc_dc_dc,
};

```

函数名 scanner - 读取格式化输入。

用法 int _scanner (int (*Get)(void *srcP),
void (*UnGet) (int ch, void *srcP),
void *srcP,
const char *formP,
va list varPP)

说明 所有 scanf 族函数的工作均由 _scanner 来做。

根据*formP 给出的格式扫描*srcP 并将结果放入*varPP。

Get 和 Unget 函数允许将源重新定义为文件或其他串行字符流。它们可能是 fget/ungetc 或其它等价对, 处理数据流/串 srcP。

格式串的语法是:

format ::= ([isspace] [literal | '%%' | '%' conversion])*

```

conversion ::= ['*'] [width] ['l' | 'h'] [type]
width ::= number;
type ::= 'd'|'D'|'u'|'U'|'o'|'O'|'x'|'X'|'i'|'I'|'n'|
        'e'|'E'|'f'|'F'|'g'|'G'|'p'|'N'|'F'|'s'|'c'|'l'

```

```
#pragma warn -use
```

```
#pragma warn -rvl
```

```

int near _scanner( int near (*Get) (void *srceP),
                  void near (*UnGet) (int ch, void *srceP),
                  void *srceP,
                  const char *formP,
                  va_list varPP )
{
    char flags; -
    int count = 0;
    int charCt = 0;
    int status;
    int width;
    char bitSet [32]; /* for scan sets */
    register SI, DI; /* prevent the compiler making its own usage */

    /*****
    'C' equivalent of inline code( for documentation )
    *****/

    #if 0
    REG char a, b;
    REG short wP;
    REG char *cP;
    long lRes;
    long double ldRes;
    #error /* The C text is algorithm commentary, not tested source ! */
    /* It is provided to clarify the intent of the assembler. */
    ssNEXT:
    if (\0 == (b = *(formP++)))
        return count; /* the normal end */
    if ((b != '%') || ('%' == (b = *(formP++)))
    {
        charCtt ++;
        if ((a = Get (srceP)) == EOF)
            goto ssNextEOF;
        if (!(b & 0x80) && (_ws == scanCtype [b])) /* white space ? */
        {
            while (!(a & 0x80) && (_ws == scanCtype [a]))
            {
                charCt ++;
                if ((a = Get (srceP)) == EOF)
                    goto ssNextEOF;
            }
            UnGet (a, srceP);
            charCt --;
        }
        else /* literal match required */
            if (a != b)
                goto ssEND;
        goto ssNEXT;
    }
}

```

```

/* if fall through to here then begin a conversion specification */
#if LDATA
    flags = isFarPtr;
#else
    flags = 0;
#endif
width = -1;
ssSwitch:
    switch ((b & 0x80) ? _de : scanCtype [b])
    {
        case (_su) :    flags |= isSuppressed;
                        b = *(formP++);
                        goto ssSwitch;
        case (_ha) :    flags |= isHalf;
                        b = *(formP++);
                        goto ssSwitch;
        case (_lo) :    flags |= isLong;
                        b = *(formP++);
                        goto ssSwitch;
        case (_ld) :    flags |= isLongDouble;
                        b = *(formP++);
                        goto ssSwitch;
        case (_nu) :    width = (width < 0) ? b - '0' :
                        10 * width + b - '0';
                        b = *(formP++);
                        goto ssSwitch;
        case (_ne) :    flags &= ~isFarPtr;
                        b = *(formP++);
                        goto ssSwitch;
        case (_fa) :    flags |= isFarPtr;
                        b = *(formP++);
                        goto ssSwitch;
        case (_pt) :    goto ssPTR;
        case (_de) :    base = 10;
                        goto ssINT;
        case (_oc) :    base = 8;
                        goto ssINT;
        case (_he) :    base = 16;
                        goto ssINT;
        case (_in) :    base = 0;
                        goto ssINT;
        case (_ct) :    lRes = charCt;
                        if ((flags & isSuppressed) == 0)
                            goto ssPUTINT;
                        b = *(formP++);
                        goto ssSwitch;
        case (_fl) :    goto ssFLOAT;
        case (_st) :    goto ssTOKEN;
        case (_ch) :    goto ssCHAR;
        case (_sc) :    goto ssSCANSET;
        case (_dc) :    goto ssEND;
        default: /* never occurs. */;
    }
ssINT:

```

```
lRes = _scantol (Get, UnGet, srceP, base, width & 0x7FFF,
```