

Peanut CLib Extension

Module 1.1.0

User Manual v1

Mar. 2024

Contents

Introduction	4
Namespace	4
Argument	4
Extension constants in namespace <code>c</code>	5
Extension Functions in namespace <code>c</code>	6
printf()	6
sprintf()	7
fopen()	8
fclose()	9
fprintf()	10
fread()	11
fflush()	12
fseek()	13
gets()	14
atoi()	15
malloc()	16
free()	17
rand()	18
srand()	19
system()	20
isalnum()	21
isalpha()	22
isblank()	23
isprint()	24
time()	25
sqrt	26
strcat()	27
strcmp()	28
strlen()	29
Extension Functions in namespace <code>c : util</code>	30
getMemAddr()	31
logPrintf()	32

pruneTailBlanks().....	33
locateSubstr()	34
charAt()	35

Introduction

This Peanut extension module is to provide functions whose interfaces follow same-name functions in the C-language standard library. This extension module does not support all functions implemented in the C-language standard library but those might be frequently used only.

Namespace

The namespace `c` is created under the global namespace by this extension module. All the extension functions registered to Peanut can be only called with namespace resolution `c::` preceded. Besides, the namespace is created under namespace `c`. The following lists the namespaces created by this module.

<code>c::util::</code>	For extension functions that are frequency used but no corresponding functions are found in the C-language standard library.
<code>c::</code>	For extension functions which are found in the C-language standard library.

Argument

In a C program, there exists pointer-type parameters; and we know that callee functions use the passed-in pointer (memory address) to update caller function's variable in a C program. In a Peanut script, pointer-type argument is not supported; instead, a call-by-reference approach is used for the callee function to update passed-in variable from the caller function.

Users who use this extension module should note that this extension module uses call-by-reference parameters to emulate standard C-library functions' pointer-type parameters if the pointer-type parameters will be updated by the C-library functions. Also note that a variable should be passed for a call-by-reference parameter as the argument.

Provided extension functions do not pay many efforts to check arguments. For some cases, improper arguments may cause the whole interpreter crash.

In the following paragraph, we do not intend to describe the functions with details, but list provided functions and figure out what the users should note.

Extension constants in namespace **c**

Name	Description
SEEK_SET	A reference position in a file; beginning of the file.
SEEK_CUR	A reference position in a file; current position of the file pointer.
SEEK_END	A reference position in a file; end of the file.

Extension Functions in namespace **c**

printf()

Prototype	printf(format, ...)	
Description	Output formatted message.	
Return Value	The total number of characters of the message outputted.	
Parameter	format	A string with message and format specifiers. The following specifiers are supported: %c %d %x %X %f %F %p %P %s The %p and %P specifiers are used to output a handle value.
	...	Optional parameters. The number of optional parameters is 0 to 7. Values of them will be outputted depending on their corresponding format specifiers existing in the first arguments.
Note	The programmer should pay effort on maintaining the consistency between the specifier and value.	
See Also		

```
str = "Hello string!\n";  
c::printf("Hello World 123!\n%s", str);
```

sprintf()

Prototype	sprintf(variable, format, ...)	
Description	Set formatted message to a variable.	
Return Value	The total number of characters of the string set to the variable.	
Parameter	variable	In C, this is a pointer-type parameter. In this extension, this parameter is a call-by-reference parameter. Users should pass a variable as the argument for this parameter when this function is called.
	format	A string with message to be outputted and format specifiers. The following specifiers are supported: %c %d %x %X %f %F %p %P %s
	...	Optional parameters. The number of optional parameters is 0 to 6. Values of them will be outputted depending on their corresponding format specifiers existing in the second arguments.
Note	The programmer should pay effort on maintaining the consistency between the specifier and value.	
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var str;

    c::sprintf(str, "Hello World %s!", "ABC");
    c::printf("Hello World 123!\n%s", str);
}
```

fopen()

Prototype	fopen(filename, mode)	
Description	Open a file for read/write contents to/from that file.	
Return Value	The handle of the file.	
Parameter	filename	The specified file name.
	mode	The open mode of the file. Valid values are "r", "w", "w+", and "rb".
Note		
See Also		

```
fh = c::fopen("log.txt", "w");  
c::util::logPrintf(fh, "start log\n");  
  
frh = c::fopen("image.bin", "rb");
```


fclose()

Prototype	fclose(fh)	
Description	Close the file.	
Return Value		
Parameter	fh	The file handle returned by fopen()
See Also		

```
fh = c::fopen("log.txt", "w");  
c::util::logPrintf(fh, "start log\n");  
...  
c::fclose(fh);
```

fprintf()

Prototype	fprintf(fhandle, format, ...)	
Description	Write formatted data to a file.	
Return Value	Number of characters written to the file.	
Parameter	fhandle	The file handle returned by fopen()
	format	A string with message to be outputted and format specifiers. The following specifiers are supported: %c %d %x %X %f %F %p %P %s
	...	Optional parameters. Number of optional parameters is up to 6.
Note	The programmer should pay effort on maintaining the consistency between the specifier and value.	
See Also		

```
fh = c::fopen("log.txt", "w");  
c::util::logPrintf(fh, "do a log by logPrintf()\n");  
c::fprintf(fh, "do a log by fprintf()\n");  
...  
c::fclose(fh);
```

fread()

Prototype	fread(memHdle, isize, inum, fhandle)	
Description	Read file content into memory.	
Return Value	0	
Parameter	memHdle	A memory handle returned by malloc()
	isize	An integer indicate the size of an item to read.
	inum	An integer indicate the number of items to read.
	fhandle	A file handle returned by fopen()
See Also		

```
memHandle = c::malloc(nFWUpdateGranu);
if (memHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

fwImgFileHandle = c::fopen(fw_filename, "rb");
if (fwImgFileHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

c::fseek(fwImgFileHandle, 0, c::SEEK_SET);
c::fread(memHandle, 1, nThisReadSize, fwImgFileHandle);
```

fflush()

Prototype	fflush(fhandle)	
Description	Flush the output buffer to the file on the disk.	
Return Value	0	
Parameter	fhandle	The file handle returned by fopen()
See Also		

```
fh = c::fopen("log.txt", "w");  
c::util::logPrintf(fh, "do a log by logPrintf()\n");  
c::fprintf(fh, "do a log by fprintf()\n");  
c::fflush(fh);
```

fseek()

Prototype	fseek(fhandle, offset, whence)	
Description	Flush the output buffer to the file on the disk.	
Return Value	0	
Parameter	fhandle	The file handle returned by fopen()
	offset	Number of bytes to offset from whence.
	whence	A reference position. Valid values are SEEK_SET, SEEK_CUR, and SEEK_END.
See Also		

```
fwImgFileHandle = c::fopen(fw_filename, "rb");
if (fwImgFileHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

c::fseek(fwImgFileHandle, 0, c::SEEK_SET);
c::fread(memHandle, 1, nThisReadSize, fwImgFileHandle);
```

gets()

Prototype	gets (&v)	
Description	Get a string from stdin.	
Return Value	0	
Parameter	v	A call-by-reference parameter. The argument should be an elementary variable for storing the string from stdin.
Note	Maximum size of the input string is 1024.	
See Also		

```
while (1)
{
    c::printf("\nSelection [1~5]: ");
    c::gets(gets_str);
    if (c::strlen(gets_str) <= 0)
        continue;
    fmtidx = c::atoi(gets_str);
    if (fmtidx >= 1 && fmtidx <= 5)
        break;
}
```

atoi()

Prototype	atoi(str)	
Description	Interpreting the str content as an integer.	
Return Value	The integer that the string str is interpreted.	
Parameter	str	A string content
See Also		

```
while (1)
{
    c::printf("\nSelection [1~5]: ");
    c::gets(gets_str);
    if (c::strlen(gets_str) <= 0)
        continue;
    fmtidx = c::atoi(gets_str);
    if (fmtidx >= 1 && fmtidx <= 5)
        break;
}
```

malloc()

Prototype	malloc(msize)	
Description	Allocate memory block.	
Return Value	A memory handle.	
Parameter	msize	Size of the memory block; msize is in bytes.
Note		
See Also	free()	

```
memHandle = c::malloc(nFWUpdateGranu);
if (memHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

fwImgFileHandle = c::fopen(fw_filename, "rb");
if (fwImgFileHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

c::fseek(fwImgFileHandle, 0, c::SEEK_SET);
c::fread(memHandle, 1, nThisReadSize, fwImgFileHandle);
```


free()

Prototype	free (mhdle)	
Description	Deallocate memory block.	
Return Value		
Parameter	mhdle	A memory handle returned by malloc().
Note		
See Also	malloc()	

```
nSize = 0x10000;  
memHandle = c::malloc(nSize);  
c::fread(memHandle, 1, nSize, fwImgFileHandle);  
...  
c::free(memHandle);
```

rand()

Prototype	rand ()	
Description	Generate a pseudo random number, an integer.	
Return Value	A random integer. The size of the number depends on the value returned from Peanut built-in function intsize().	
Parameter	N/A	This function requires no parameter.
Note		
See Also	srand()	

srand()

Prototype	srand (seed)	
Description	Set random seed.	
Return Value	0	
Parameter	seed	The random seed, which is an integer.
Note		
See Also	rand()	

```
c::srand(c::time());
```

system()

Prototype	system(cmdstr)	
Description	Execute system command.	
Return Value	A integer returned by the command processor.	
Parameter	cmdstr	A command string.
Note		
See Also		

```
rv = c::system("if [ -f Makefile_lnx ] ; then echo \"yes\" ; else echo  
\"no\" ; fi");
```

isalnum()

Prototype	isalnum(c)	
Description	Check whether c is a decimal digit or an lowercase/uppercase letter. Note that the definition a decimal digit or lower/uppsercase letter is based on ASCII definition.	
Return Value	0 or 1. 0 stands for false; 1 stands for true.	
Parameter	c	A character to be checked.
See Also		

isalpha()

Prototype	isalpha (c)	
Description	Check whether c is an alphabetic letter.	
Return Value	0 or 1. 0 stands for false; 1 stands for true.	
Parameter	c	A character to be checked.
See Also		

isblank()

Prototype	isblank(c)	
Description	Check whether c is a blank character.	
Return Value	0 or 1. 0 stands for false; 1 stands for true.	
Parameter	c	A character value to be checked.
See Also		

isprint()

Prototype	<code>isprint(c)</code>	
Description	Check whether c is a printable character.	
Return Value	0 or 1. 0 stands for false; 1 stands for true.	
Parameter	c	A character value to be checked.
See Also		

time()

Prototype	time()	
Description	Return current time; measured in seconds.	
Return Value	Current time in seconds.	
Parameter	N/A	This function requires no parameter.
See Also		

```
stime = ctime = c::time();  
while (ctime - stime < 10)  
{  
    ...  
    ctime = c::time();  
}
```

sqrt

Prototype	sqrt (x)	
Description	Compute square root.	
Return Value	The square root of the input value.	
Parameter	x	A floating number
Note		
See Also		

```
c::printf("square root of 3.0 = %f\n", c::sqrt(3.0));
```

strcat()

Prototype	strcat(dest, source)	
Description	Concatenate two strings into one. Append source string to dest string.	
Return Value	The new string.	
Parameter	dest	This parameter is a call-by-value parameter. The destination string, to which the source string will be appended.
	source	A string to be appended.
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var fs0 = "0";
    var fs1 = "1";
    var fsn, n;

    c::printf("fs[0] = %s\n", fs0);
    c::printf("fs[1] = %s\n", fs1);
    for (n = 2; n < 10; ++n)
    {
        fsn = "";
        c::strcat(fsn, fs0);
        c::strcat(fsn, fs1);
        c::printf("fs[%d] = %s\n", n, fsn);

        fs0 = fs1;
        fs1 = fsn;
    }
}
```

strcmp()

Prototype	strcmp(str1, str2)	
Description	Compare str1 string to str2 string.	
Return Value	<0 : that first character that does not match has a lower value in str1 than str2. 0 : both strings are equal >0 : that first character that does not match has a greater value in str1 than str2.	
Parameter	str1	The first string to be compared
	str2	The second string to be compared
See Also		

strlen()

Prototype	strlen(str)	
Description	Get string str length.	
Return Value	The length of string str.	
Parameter	str	A string
See Also		

Extension Functions in namespace `c::util`

In the following paragraph, the extension functions are created in the namespace `util`, where `util` namespace is created in namespace `c`. To call the extension functions, the function names should be prefixed with the namespace resolution `c::util::`.

getMemAddr()

Prototype	getMemAddr (mHdle)	
Description	Get the address of a memory block.	
Return Value	An integer representing the memory block address.	
Parameter	mHdle	The memory handle of the memory block.
See Also		

```
memHandle = c::malloc(nFWUpdateGranu);
if (memHandle == 0)
{
    c::util::logPrintf(logfd, "Fail to open %s!\n", fw_filename);
    goto Return;
}

addr = c::util::getMemAddr(mHandle);
```

logPrintf()

Prototype	logPrintf(fhandle, format, ...)	
Description	Save the formatted message into a file and print it on the screen at the same time. If fhandle is 0, the formatted message is printed on the screen only.	
Return Value	The number of characters printed.	
Parameter	fhandle	The file handle returned by fopen().
	format	A string with message to be outputted and format specifiers. The following specifiers are supported: %c %d %x %X %f %F %p %P %s
	...	Optional parameters. Number of optional parameters is up to 6.
Note	The programmer should pay effort on maintaining the consistency between the specifier and value.	
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var fhandle = fopen("log.txt", "w");

    c::util::logPrintf(fhandle, "Script start\n");
    c::util::logPrintf(fhandle, "Current time: %s\n",
        c::util::getCurLocalTimeStr());

    fclose(fhandle);
}
```


pruneTailBlanks()

Prototype	pruneTailBlanks (&v)	
Description	Remove blank characters from the string tail. The function stop the operation once the last character is not a blank character.	
Return Value	0 or 1. 0 stands for failure. 1 stands for success.	
Parameter	v	A call-by-reference parameter. The argument should be an elementary variable whose value is a string.
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var str = "abc 123    ";
    c::util::pruneTailBlanks(str);
    print(str, "\n");
}
```

locateSubstr()

Prototype	locateSubstr(string, substr)	
Description	Check whether a sub-string is contained within a string and return the offset of the sub-string.	
Return Value	-1 : not found or input error otherwise : the offset of the sub-string in the string	
Parameter	string	A string in which to search for the sub-string specified by substr
	substr	The sub-string to be searched
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var aStr = "abcdefghijklmnopqrstuvwxyz";
    var loc = c::util::locateSubstr(aStr, "opqrs");

    print(loc < 0 ? "No." : "Yes.");
}
```

charAt()

Prototype	charAt(str, index)	
Description	Get the indexed character value in a string.	
Return Value	The character value.	
Parameter	str	A string
	index	The index of the character in the string. Say the length of the string str is n, the valid index ranges from 0 to n-1.
See Also		

```
#load <pxm_clib.pxm>

main()
{
    var aStr = "abcdefghijklmnopqrstuvwxyz";
    var rStr;
    var nLen, nIdx, temp;

    nLen = c::strlen(aStr);
    rStr = "";
    for (nIdx = nLen - 1; nIdx >= 0; --nIdx)
    {
        c::sprintf(temp, "%c", c::util::charAt(aStr, nIdx));
        rStr += temp;
    }
    print("aStr = ", aStr, "\n");
    print("rStr = ", rStr);
}
```