EFI Application Toolkit C Library (LIBC) Port Specification

Revision 0.08

January 25, 2000

ESG Server Software Technologies (SST)

Revision History

Date	Revision	Modifications
10/04/99	0.01	Initial version
11/16/99	0.02	Updated to reflect current implementation
11/23/99	0.03	Updated Appendix I to reflect current implementation
12/3/99	0.07	Released for customer review
1/25/00	0.08	Updated Appendix I to reflect current implementation

Disclaimers

Table Of Contents

Table Of Figures

This page intentionally left blank

1 Introduction

2 C Library – General

The following sections describe the execution requirements and operating environment of EFI-AT LIBC.

2.1 LIBC Operating Requirements

2.1.1 Operating Environments

EFI-AT LIBC supports several operating environments, including those listed below.

- EFI NT emulator environment
- EFI BIOS32 boot floppy environment
- EFI Sal64 IA-64 simulator environment

Support for these environments should translate directly into support for the corresponding IA-32 and IA-64 hardware platforms once they are available with EFI 1.0 compliant firmware installed.

2.2 LIBC Functionality

EFI-AT LIBC provides a subset of the standard C library, along with a subset of the BSD Unix system call API and some wide-character support specified in the draft ISO/ANSI 9x/2000 C Language standard. The major categories of standard functions supported are the assert, ctype, errno, locale, math, setjmp, stdio, stdlib, string and time functions. The other functions supported are wide-char ctype and wide-char string functions, and the system call API functions open, close, read, write, stat, fstat, lstat, ioctl, and isatty.

The full listing of all functions supported can be found in Appendix 1.

3 LIBC Integration Architecture

3.1 Overview

LIBC is based on source code from the FreeBSD source base, an open source Unix operating system. Thus, LIBC expects to call the Berkeley Unix system call API for system services. Much of the porting effort centers on developing an interface layer that supported a Berkeley Unix API that internally calls EFI services. In general, it is possible to support everything needed by LIBC using only the EFI services. LIBC was ported to have minimal dependency on the EFI Library included with the EFI reference code.

3.2 Architecture

The parts of LIBC that were most challenging to port were the input/output functions (stdio). These functions are expected to work with a variety of input/output devices, including (but not limited to) the console tty, the file system, and the serial interface. Read, write, and close should work for multiple I/O devices under EFI just as they do under an operating system.

To accomplish this it was necessary to abstract the device specific data and operations from the Unix API interface implementation. Thus, there are two layers to the Unix API interface: the device-independent layer and the device-specific layer.

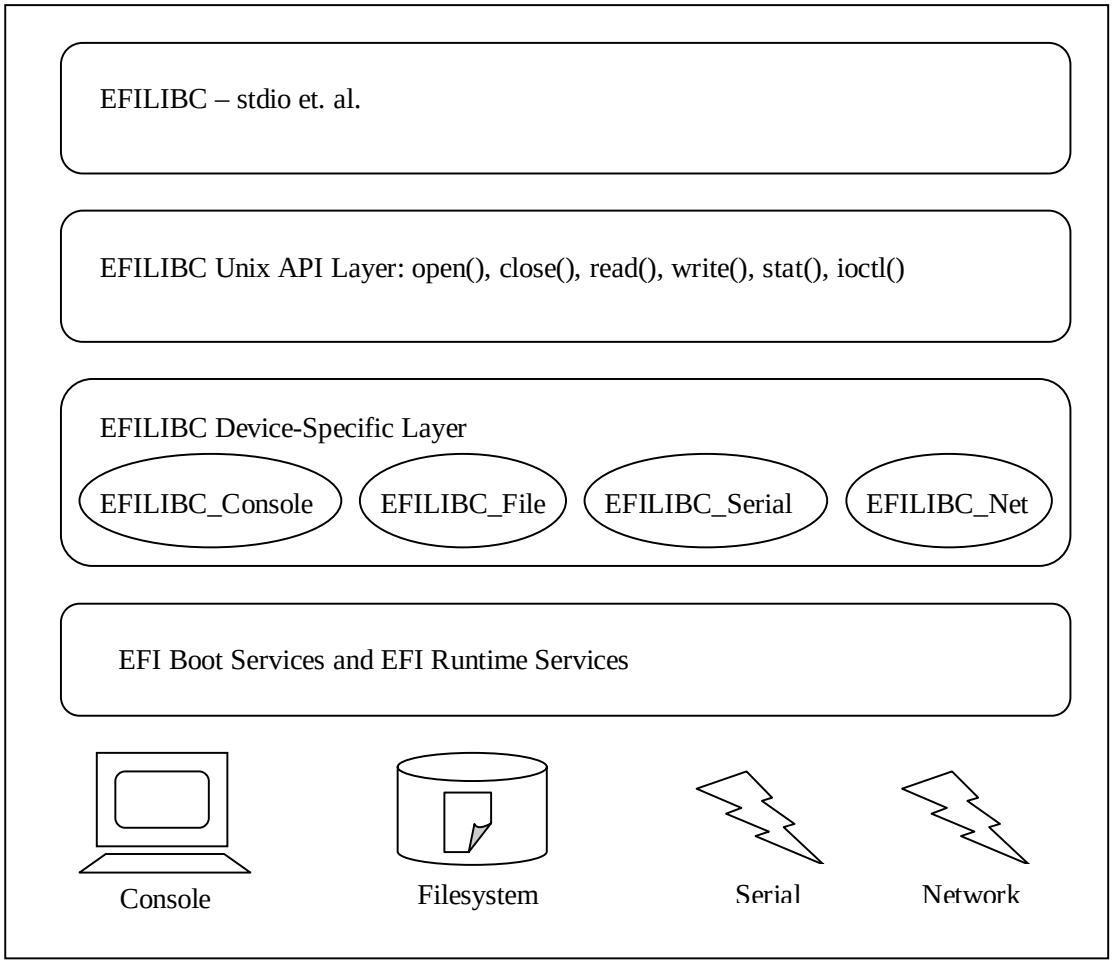

Figure 3-1: EFI LIBC Architecture

3.2.1 File Descriptors

At the heart of the abstraction of the device from the interface is the file descriptor. The file descriptor contains pointers to the device-specific I/O functions (open, close, read, write…), as well as device-specific information used by those functions. Examples of device specific information include the EFI_FILE_HANDLE for file system devices and the input buffer for **consolein:**. Device-specific information is pointed to by a VOID* pointer, so the device can define its own data structure for its own information.

File descriptors are not accessed directly. The file descriptor table is an opaque data structure, and a number of simple functions are provided to access file descriptor fields.

The Unix API layer of EFI LIBC uses the device-specific function pointers to perform the requested operation, thereby providing a single API that supports multiple different kinds of devices.

3.2.2 File Descriptor Extensibility / Symbolic Names

File descriptors point at device specific information needed to perform I/O on any particular device, and new devices can be registered in order to allow them to fully utilize the Unix API layer. The process of doing this consists of two steps.

In the first step, a protocol specific open() function is associated with an protocol interface GUID. This mapping is performed via the function _LIBC_MapProtocol(). The protocol specific open() function must provide LIBC with function pointers to all of the protocol specific functions associated with the protocol interface GUID. These function pointers are stored in the file descriptor table via the function _LIBC_AllocateNewFileDescriptor() when the protocol specific open() function is called.

In the second step, a symbolic name is associated with the protocol interface GUID and a specific device path. This interface is provided by the function _LIBC_MapDevice(). After the mappings are complete, the Unix API layer open() function looks up the symbolic name in the device table and then invokes the protocol specific open() call found in the protocol table. The protocol specific open() call sets up a new file descriptor containing all the other protocol specific function pointers.

In EFI LIBC environment mappings are not persistent, and applications must reestablish any desired mapping upon each invocation. EFI LIBC provides several default mappings for the console device including **consolein:**, **consoleout:**, and **consoleerr:** and a default mapping for the current file system **default:**. Note that the EFI LIBC does not support a mechanism to remove protocol or device mappings from the internal tables.

3.2.3 Console I/O

Formatted and string console I/O is converted from the native LIBC format (ASCII) to the console format (Unicode). For output this is done in the two functions __sprint() and __swbuf() using the ANSI standard function mbstowcs(). For input it is done in __srefill(). Unformatted console I/O (fread() and fwrite()) is *not* converted.

3.2.4 Application Startup & Initialization

Prior to using any LIBC function, an application must first initialize the library. Initialization of the library can be done explicitily or implictly depending on the type of application that is being developed.

3.2.4.1 Explicit Initialization

Applications may explicitly initialize LIBC by calling InitializeLIBC(). This function sets up and initializes all of the LIBC data structures needed to support the Unix API layer.

Application arguments argc or argv are not available unless the application is started from the shell. To access argc and argv the application must call InitializeShellApplication(). Note that InitializeShellApplication() requires both the EFI shell interface library and the EFI library to be linked to the application.

It may be possible to determine if an application was called from the shell by using the LoadedImage to determine the parent application's filename.

3.2.4.2 Implicit Initialization

Applications that are intended to run exclusively in the shell environment can implicitly initialize LIBC by defining _LIBC_Start_Shellapp_U() or _LIBC_Start_Shellapp_A() as the entry point to the application. These functions initialize LIBC by calling InitializeLIBC(). Additionally, each function also initializes the EFI shell interface library and the EFI library, and calls main(). Both functions call exit()after main() completes to insure that LIBC is terminated properly.

The functions differ in their handling of arguments retrieved from the shell interface library. When _LIBC_Start_Shellapp_U() is used, main() is called with Unicode arguments retrieved from the shell interface. When _LIBC_Start_Shellapp_A() is used, main()is called with ASCII arguments converted from the Unicode arguments retrieved from the command shell..

Note that the use of either _LIBC_Start_Shellapp_U() or _LIBC_Start_Shellapp_A() requires the EFI shell interface library and the EFI library to be linked to the application.

3.2.5 Application Cleanup & Termination

EFI does not provide any application exit event, so there is no way to invoke a callback on application exit. Therefore, the user is responsible for ensuring that exit() is called every time the application exits, in order to clean up the heap, flush I/O buffers, close files, etc. However, if the application entry point is defined as LIBC_StartShellapp_U() or LIBC_StartShellapp_A(), the funciton exit() is called automatically after main() completes execution.

Note that in EFI dynamic memory is not released by the system until ExitBootSevices() is called.

3.2.6 Internationalization

As implemented, LIBC is not well suited to support internationalization. Although there are some functions that would be useful in manipulating wide-character strings, this implementation of LIBC is an ASCII-character implementation and does not support anything more than the ANSI/ISO C specification circa 1997, which specified some locale support and wide-character conversion functions. This version provides wide-character string manipulation functions, but there is no support for wide-character formatted I/O, wide-character classification, etc.

3.2.7 Reentrancy

LIBC is not multithreaded and as such does not support reentrancy.

3.2.8 Critical Data Structures

Updates to certain critical data structures within LIBC are protected against interruption to prevent corruption. The mechanism used to provide protection is based upon the task privilege level (TPL) described in the EFI specification. Immediately before updating a critical data structure the current TPL is raised to TPL_NOTIFY. Following completion of the update, the TPL is restored. This mechanism provides some basic protection of the critical data structures.

Appendix 1: List of C Library Functions

This appendix lists all of the C Library functions of interest. The list of functions is organized by header file and includes for each function when the implementation is planned, the standard the function is referenced, and any additional remarks about the implementation.

Atk_Libc.h	Phase	Standard	Remarks
InitializeLibC	I	None	Implemented
_LIBC_AllocateNewFileDescriptor	I	None	Implemented
_LIBC_EfiExit	I	None	Implemented
_LIBC_GetOpenFileDevSpecific	I	None	Implemented
_LIBC_MapDevice	I	None	Implemented
_LIBC_MapProtocol	I	None	Implemented
_LIBC_Mount	TBD	None	
_LIBC_Start_Shellapp_A	I	None	Implemented
_LIBC_Start_Shellapp_U	I	None	Implemented

Assert.h	Phase	Standard	Remarks
Assert	I	ANSI	Implemented

Complex.h	Phase	Standard	Remarks
Cabs	-	ANSI	No FreeBSD implementation to port
Cabsf	-	ANSI	No FreeBSD implementation to port
Cabsl	-	ANSI	No FreeBSD implementation to port
Cacos	-	ANSI	No FreeBSD implementation to port
Cacosh	-	ANSI	No FreeBSD implementation to port
Cacoshf	-	ANSI	No FreeBSD implementation to port
Cacoshl	-	ANSI	No FreeBSD implementation to port
Carg	-	ANSI	No FreeBSD implementation to port
Cargf	-	ANSI	No FreeBSD implementation to port
Cargl	-	ANSI	No FreeBSD implementation to port
Casin	-	ANSI	No FreeBSD implementation to port
Casinf	-	ANSI	No FreeBSD implementation to port
Casinh	-	ANSI	No FreeBSD implementation to port
Casinhf	-	ANSI	No FreeBSD implementation to port
Casinhl	-	ANSI	No FreeBSD implementation to port
Casinl	-	ANSI	No FreeBSD implementation to port
Catan	-	ANSI	No FreeBSD implementation to port
Catanf	-	ANSI	No FreeBSD implementation to port
Catanh	-	ANSI	No FreeBSD implementation to port
Catanhf	-	ANSI	No FreeBSD implementation to port
Catanhl	-	ANSI	No FreeBSD implementation to port
Catanl	-	ANSI	No FreeBSD implementation to port
Ccos	-	ANSI	No FreeBSD implementation to port
Ccosf	-	ANSI	No FreeBSD implementation to port
Ccosh	-	ANSI	No FreeBSD implementation to port
Ccoshf	-	ANSI	No FreeBSD implementation to port
Ccoshl	-	ANSI	No FreeBSD implementation to port

Complex.h	Phase	Standard	Remarks
Ccosl	-	ANSI	No FreeBSD implementation to port
Cexp	-	ANSI	No FreeBSD implementation to port
Cexpf	-	ANSI	No FreeBSD implementation to port
Cexpl	-	ANSI	No FreeBSD implementation to port
Cimag	-	ANSI	No FreeBSD implementation to port
Cimagf	-	ANSI	No FreeBSD implementation to port
Cimagl	-	ANSI	No FreeBSD implementation to port
Clog	-	ANSI	No FreeBSD implementation to port
Clogf	-	ANSI	No FreeBSD implementation to port
Clogl	-	ANSI	No FreeBSD implementation to port
Conj	-	ANSI	No FreeBSD implementation to port
Conjf	-	ANSI	No FreeBSD implementation to port
Conjl	-	ANSI	No FreeBSD implementation to port
Cpow	-	ANSI	No FreeBSD implementation to port
Cpowf	-	ANSI	No FreeBSD implementation to port
Cpowl	-	ANSI	No FreeBSD implementation to port
Cproj	-	ANSI	No FreeBSD implementation to port
Cprojf	-	ANSI	No FreeBSD implementation to port
Cprojl	-	ANSI	No FreeBSD implementation to port
Creal	-	ANSI	No FreeBSD implementation to port
Crealf	-	ANSI	No FreeBSD implementation to port
Crealf	-	ANSI	No FreeBSD implementation to port
Csin	-	ANSI	No FreeBSD implementation to port
Csinf	-	ANSI	No FreeBSD implementation to port
Csinh	-	ANSI	No FreeBSD implementation to port
Csinhf	-	ANSI	No FreeBSD implementation to port
Csinhf	-	ANSI	No FreeBSD implementation to port
Csinhl	-	ANSI	No FreeBSD implementation to port
Csinhl	-	ANSI	No FreeBSD implementation to port
Csinl	-	ANSI	No FreeBSD implementation to port
Csqrt	-	ANSI	No FreeBSD implementation to port
Csqrtf	-	ANSI	No FreeBSD implementation to port
Csqrtl	-	ANSI	No FreeBSD implementation to port
Ctan	-	ANSI	No FreeBSD implementation to port
Ctanf	-	ANSI	No FreeBSD implementation to port
Ctanh	-	ANSI	No FreeBSD implementation to port
Ctanhf	-	ANSI	No FreeBSD implementation to port
Ctanhl	-	ANSI	No FreeBSD implementation to port
Ctanl	-	ANSI	No FreeBSD implementation to port

Ctype.h	Phase	Standard	Remarks
Digittoint	I	FreeBSD	Implemented
Isalnum	I	ANSI	Implemented
Isalpha	I	ANSI	Implemented
Isascii	I	FreeBSD	Implemented
Isblank	I	FreeBSD	Implemented
Iscntrl	I	ANSI	Implemented
Isdigit	I	ANSI	Implemented
Isgraph	I	ANSI	Implemented
Ishexnumber	I	FreeBSD	Implemented
Islower	I	ANSI	Implemented
Isprint	I	ANSI	Implemented
Ispunct	I	ANSI	Implemented
Isrune	I	FreeBSD	Implemented
Isspace	I	ANSI	Implemented

Ctype.h	Phase	Standard	Remarks
Isupper	I	ANSI	Implemented
Isxdigit	I	ANSI	Implemented
Toascii	I	FreeBSD	Implemented
Tolower	I	ANSI	Implemented
Toupper	I	ANSI	Implemented

Dirent.h	Phase	Standard	Remarks
Alphasort	I	FreeBSD	Implemented
Closedir	I	FreeBSD	Implemented
Getdirentries	I	FreeBSD	Implemented
Opendir	I	FreeBSD	Implemented
Readdir	I	FreeBSD	Implemented
Rewinddir	I	FreeBSD	Implemented
Scandir	I	FreeBSD	Implemented
Seekdir	I	FreeBSD	Implemented
Selldir	I	FreeBSD	Implemented
Teeldir	I	FreeBSD	Implemented

Err.h	Phase	Standard	Remarks
Err	I	FreeBSD	Implemented
Errc	I	FreeBSD	Implemented
Errx	I	FreeBSD	Implemented
Err_set_exit	I	FreeBSD	Implemented
Eff_set_file	I	FreeBSD	Implemented
Verr	I	FreeBSD	Implemented
Verrc	I	FreeBSD	Implemented
Verrx	I	FreeBSD	Implemented
Vwarn	I	FreeBSD	Implemented
Vwarnc	I	FreeBSD	Implemented
Vwarnx	I	FreeBSD	Implemented
Warn	I	FreeBSD	Implemented
Warnc	I	FreeBSD	Implemented
Warnx	I	FreeBSD	Implemented

Errno.h	Phase	Standard	Remarks
Errno	I	ANSI	Implemented

Fenv.h	Phase	Standard	Remarks
Feclearexcept	-	ANSI	No FreeBSD implementation to port
Fegetenv	-	ANSI	No FreeBSD implementation to port
Fegetexceptflag	-	ANSI	No FreeBSD implementation to port
Fegetround	-	ANSI	No FreeBSD implementation to port
Feholdexcept	-	ANSI	No FreeBSD implementation to port
Feraiseexcept	-	ANSI	No FreeBSD implementation to port
Fesetenv	-	ANSI	No FreeBSD implementation to port
Fesetexceptflag	-	ANSI	No FreeBSD implementation to port
Fesetround	-	ANSI	No FreeBSD implementation to port
Fetestexcept	-	ANSI	No FreeBSD implementation to port
Feupdateenv	-	ANSI	No FreeBSD implementation to port

Locale.h	Phase	Standard	Remarks

Locale.h	Phase	Standard	Remarks
Localeconv	II	ANSI	
Setlocale	II	ANSI	

Math.h	Phase	Standard	Remarks
			When implemented, these will not be tuned to take advantage of Intel architecture.
Acos	I	ANSI	Implemented
Acosf	-	ANSI	No FreeBSD implementation to port
Acosh	I	ANSI	Implemented
Acoshf	-	ANSI	No FreeBSD implementation to port
Acoshl	-	ANSI	No FreeBSD implementation to port
Acosl	-	ANSI	No FreeBSD implementation to port
Asin	I	ANSI	Implemented
Asinf	-	ANSI	No FreeBSD implementation to port
Asinh	I	ANSI	Implemented
Asinhf	-	ANSI	No FreeBSD implementation to port
Asinhl	-	ANSI	No FreeBSD implementation to port
Asinl	-	ANSI	No FreeBSD implementation to port
Atan	I	ANSI	Implemented
Atan2	I	ANSI	Implemented
Atan2f	-	ANSI	No FreeBSD implementation to port
Atan2l	-	ANSI	No FreeBSD implementation to port
Atanf	-	ANSI	No FreeBSD implementation to port
Atanh	I	ANSI	Implemented
Atanhf	-	ANSI	No FreeBSD implementation to port
Atanhl	-	ANSI	No FreeBSD implementation to port
Atanl	-	ANSI	No FreeBSD implementation to port
Cabs	I	ANSI	Implemented
Cbrt	I	ANSI	Implemented
Cbrtf	-	ANSI	No FreeBSD implementation to port
Ceil	I	ANSI	Implemented
Ceilf	-	ANSI	No FreeBSD implementation to port
Ceill	-	ANSI	No FreeBSD implementation to port
Cinhf	-	ANSI	No FreeBSD implementation to port
Cinhl	-	ANSI	No FreeBSD implementation to port
Copysign	I	ANSI	Implemented
Cos	I	ANSI	Implemented
Cosf	-	ANSI	No FreeBSD implementation to port
Cosh	I	ANSI	Implemented
Coshf	-	ANSI	No FreeBSD implementation to port
Coshl	-	ANSI	No FreeBSD implementation to port
Cosl	-	ANSI	No FreeBSD implementation to port
Crbtl	-	ANSI	No FreeBSD implementation to port
Drem	II	FreeBSD	
Erf	I	ANSI	Implemented
Erfc	I	ANSI	Implemented
Erfcf	-	ANSI	No FreeBSD implementation to port
Erfcl	-	ANSI	No FreeBSD implementation to port
Erff	-	ANSI	No FreeBSD implementation to port
Erfl	-	ANSI	No FreeBSD implementation to port
Exp	I	ANSI	Implemented
Exp2	-	ANSI	No FreeBSD implementation to port
Exp2f	-	ANSI	No FreeBSD implementation to port
Exp2l	-	ANSI	No FreeBSD implementation to port

Math.h	Phase	Standard	Remarks
Expf	-	ANSI	No FreeBSD implementation to port
Expl	-	ANSI	No FreeBSD implementation to port
Expm1	I	ANSI	Implemented
Expm1f	-	ANSI	No FreeBSD implementation to port
Expm1l	-	ANSI	No FreeBSD implementation to port
Fabs	I	ANSI	Implemented
Fabsf	-	ANSI	No FreeBSD implementation to port
Fasbsl	-	ANSI	No FreeBSD implementation to port
Fdim	-	ANSI	No FreeBSD implementation to port
Finite	I	FreeBSD	Implemented
Floor	I	ANSI	Implemented
Floorf	-	ANSI	No FreeBSD implementation to port
Floorl	-	ANSI	No FreeBSD implementation to port
Fma	-	ANSI	No FreeBSD implementation to port
Fmaf	-	ANSI	No FreeBSD implementation to port
Fmal	-	ANSI	No FreeBSD implementation to port
Fmax	-	ANSI	No FreeBSD implementation to port
Fmin	-	ANSI	No FreeBSD implementation to port
Fminf	-	ANSI	No FreeBSD implementation to port
Fminl	-	ANSI	No FreeBSD implementation to port
Fmod	I	ANSI	Implemented
Fpclassify	-	ANSI	No FreeBSD implementation to port
Frexp	I	ANSI	Implemented
Frexpf	-	ANSI	No FreeBSD implementation to port
Frexpl	-	ANSI	No FreeBSD implementation to port
Hypot	I	ANSI	Implemented
Hypotf	-	ANSI	No FreeBSD implementation to port
Hypotl	-	ANSI	No FreeBSD implementation to port
Ilogb	-	ANSI	No FreeBSD implementation to port
Ilogbf	-	ANSI	No FreeBSD implementation to port
Ilogbl	-	ANSI	No FreeBSD implementation to port
Infnan	II	FreeBSD	
Isfinite	-	ANSI	No FreeBSD implementation to port
Isgreater	-	ANSI	No FreeBSD implementation to port
Isgreaterequal	-	ANSI	No FreeBSD implementation to port
Isinf	I	ANSI	Implemented
Isless	-	ANSI	No FreeBSD implementation to port
Islessequal	-	ANSI	No FreeBSD implementation to port
Islessgreater	-	ANSI	No FreeBSD implementation to port
Isnan	I	ANSI	Implemented
Isnormal	-	ANSI	No FreeBSD implementation to port
Isunordered	-	ANSI	No FreeBSD implementation to port
J0	I	FreeBSD	Implemented
J1	I	FreeBSD	Implemented
Jn	I	FreeBSD	Implemented
Ldexp	I	ANSI	Implemented
Ldexpf	-	ANSI	No FreeBSD implementation to port
Ldexpl	-	ANSI	No FreeBSD implementation to port
Lgamma	I	ANSI	Implemented
Lgammaf	-	ANSI	No FreeBSD implementation to port
Lgammal	-	ANSI	No FreeBSD implementation to port
Llrint	-	ANSI	No FreeBSD implementation to port
Llround	-	ANSI	No FreeBSD implementation to port
Log	I	ANSI	Implemented
Log10	I	ANSI	Implemented
Log10f	-	ANSI	No FreeBSD implementation to port

Math.h	Phase	Standard	Remarks
Log10l	-	ANSI	No FreeBSD implementation to port
Log1p	I	ANSI	Implemented
Log1pf	-	ANSI	No FreeBSD implementation to port
Log1pl	-	ANSI	No FreeBSD implementation to port
Log2	-	ANSI	No FreeBSD implementation to port
Log2f	-	ANSI	No FreeBSD implementation to port
Log2l	-	ANSI	No FreeBSD implementation to port
Logb	I	ANSI	Implemented
Logbf	-	ANSI	No FreeBSD implementation to port
Logbl	-	ANSI	No FreeBSD implementation to port
Logf	-	ANSI	No FreeBSD implementation to port
Logl	-	ANSI	No FreeBSD implementation to port
Lrint	-	ANSI	No FreeBSD implementation to port
Lround	-	ANSI	No FreeBSD implementation to port
Modf	I	ANSI	Implemented
Modff	-	ANSI	No FreeBSD implementation to port
Modfl	-	ANSI	No FreeBSD implementation to port
Nan	-	ANSI	No FreeBSD implementation to port
Nearbyint	-	ANSI	No FreeBSD implementation to port
Nearbyintf	-	ANSI	No FreeBSD implementation to port
Nearbyintl	-	ANSI	No FreeBSD implementation to port
Nextafter	-	ANSI	No FreeBSD implementation to port
Nextafterx	-	ANSI	No FreeBSD implementation to port
Pow	I	ANSI	Implemented
Powf	-	ANSI	No FreeBSD implementation to port
Powl	-	ANSI	No FreeBSD implementation to port
Remainder	-	ANSI	No FreeBSD implementation to port
Rint	I	ANSI	Implemented
Rintf	-	ANSI	No FreeBSD implementation to port
Rintl	-	ANSI	No FreeBSD implementation to port
Round	-	ANSI	No FreeBSD implementation to port
Scalb	I	FreeBSD	Implemented
Scalbln	-	ANSI	No FreeBSD implementation to port
Scalblnf	-	ANSI	No FreeBSD implementation to port
Scalblnl	-	ANSI	No FreeBSD implementation to port
Scalbn	-	ANSI	No FreeBSD implementation to port
Scalbnf	-	ANSI	No FreeBSD implementation to port
Scalbnl	-	ANSI	No FreeBSD implementation to port
Signbit	-	ANSI	No FreeBSD implementation to port
Sin	I	ANSI	Implemented
Sinf	-	ANSI	No FreeBSD implementation to port
Sinh	I	ANSI	Implemented
Sinl	-	ANSI	No FreeBSD implementation to port
Sqrt	I	ANSI	Implemented
Sqrtf	-	ANSI	No FreeBSD implementation to port
Sqrtl	-	ANSI	No FreeBSD implementation to port
Tan	I	ANSI	Implemented
Tanf	-	ANSI	No FreeBSD implementation to port
Tanh	I	ANSI	Implemented
Tanhf	-	ANSI	No FreeBSD implementation to port
Tanhl	-	ANSI	No FreeBSD implementation to port
Tanl	-	ANSI	No FreeBSD implementation to port
Tgamma	-	ANSI	No FreeBSD implementation to port
Tgammaf	-	ANSI	No FreeBSD implementation to port
Tgammal	-	ANSI	No FreeBSD implementation to port
Trunc	-	ANSI	No FreeBSD implementation to port

Math.h	Phase	Standard	Remarks
Y0	I	FreeBSD	Implemented
Y1	I	FreeBSD	Implemented
Yn	I	FreeBSD	Implemented

Regex.h	Phase	Standard	Remarks
Regfree	I	FreeBSD	Implemented
Regexec	I	FreeBSD	Implemented
Regerror	I	FreeBSD	Implemented
Regcomp	I	FreeBSD	Implemented

Setjmp.h	Phase	Standard	Remarks
Longjmp	TBD	ANSI	
Setjmp	TBD	ANSI	

Signal.h	Phase	Standard	Remarks
			No good mapping from EFI events to Unix STDC signals. ASCII keyboard control characters not recognized by default EFI consoles.
Kill	-	FreeBSD	
Killpg	-	FreeBSD	
Psignal	-	FreeBSD	
Raise	-	ANSI	
Sigaction	-	FreeBSD	
Sigaddset	-	FreeBSD	
Sigaddset	-	FreeBSD	
Sigaltstack	-	FreeBSD	
Sigblock	-	FreeBSD	
Sigdelset	-	FreeBSD	
Sigdelset	-	FreeBSD	
Sigemptyset	-	FreeBSD	
Sigemptyset	-	FreeBSD	
Sigfillset	-	FreeBSD	
Sigfillset	-	FreeBSD	
Siginterrupt	-	FreeBSD	
Sigismember	-	FreeBSD	
Sigismember	-	FreeBSD	
Signal	-	ANSI	
Sigpause	-	FreeBSD	
Sigpending	-	FreeBSD	
Sigprocmask	-	FreeBSD	
Sigqueue	-	FreeBSD	
Sigreturn	-	FreeBSD	
Sigsetmask	-	FreeBSD	
Sigstack	-	FreeBSD	
Sigsuspend	-	FreeBSD	
Sigtimedwait	-	FreeBSD	
Sigvec	-	FreeBSD	
Sigwait	-	FreeBSD	
Sigwaitinfo	-	FreeBSD	

Stdarg.h	Phase	Standard	Remarks
Va_arg	I	ANSI	Implemented
Va_copy	-	ANSI	No FreeBSD implementation to port
Va_end	I	ANSI	Implemented
Va_start	I	ANSI	Implemented

Stdio.h	Phase	Standard	Remarks
Asprintf	I	FreeBSD	Implemented
Clearerr	I	ANSI	Implemented
Ctermid	I	FreeBSD	Implemented
Fclose	I	ANSI	Implemented
Fdopen	I	FreeBSD	Implemented
Feof	I	ANSI	Implemented
Ferror	I	ANSI	Implemented
Fflush	I	ANSI	Implemented
Fgetc	I	ANSI	Implemented
Fgetln	I	FreeBSD	Implemented
Fgetpos	I	ANSI	Implemented
Fgets	I	ANSI	Implemented
Fileno	I	FreeBSD	Implemented
Flockfile	I	FreeBSD	Implemented
Fopen	I	ANSI	Implemented
Fprintf	I	ANSI	Implemented
Fpurge	I	FreeBSD	Implemented
Fputc	I	ANSI	Implemented
Fputs	I	ANSI	Implemented
Fread	I	ANSI	Implemented
Fropen	-	FreeBSD	
Freopen	-	ANSI	Needs dup2, file handle reference counting
Fscanf	I	ANSI	Implemented
Fseek	I	ANSI	Implemented
Fseeko	I	FreeBSD	Implemented
Fsetpos	I	ANSI	Implemented
Ftell	I	ANSI	Implemented
Ftello	I	FreeBSD	Implemented
Ftrylockfile	I	FreeBSD	Implemented
Funlockfile	I	FreeBSD	Implemented
Funopen	I	FreeBSD	Implemented
Fwopen	I	FreeBSD	Implemented
Fwrite	I	ANSI	Implemented
Getc	I	ANSI	Implemented
Getchar	I	ANSI	Implemented
Gets	I	ANSI	Implemented
Getw	I	FreeBSD	Implemented
Perror	I	ANSI	Implemented
Printf	I	ANSI	Implemented
Putc	I	ANSI	Implemented
Putchar	I	ANSI	Implemented
Puts	I	ANSI	Implemented
Putw	I	FreeBSD	Implemented
Remove	I	ANSI	Implemented
Rename	I	ANSI	Implemented
Rewind	I	ANSI	Implemented
Scanf	I	ANSI	Implemented
Setbuf	I	ANSI	Implemented

Stdio.h	Phase	Standard	Remarks
Setbuffer	I	FreeBSD	Implemented
Setlinebuffer	I	FreeBSD	Implemented
Setvbuf	I	ANSI	Implemented
Snprintf	I	ANSI	Implemented
Sprintf	I	ANSI	Implemented
Sscanf	I	ANSI	Implemented
Sys_errlist	I	FreeBSD	Implemented
Sys_nerr	I	FreeBSD	Implemented
Tempname	I	FreeBSD	Implemented
Tmpfile	I	ANSI	Implemented
Tmpnam	I	ANSI	Implemented
Ungetc	I	ANSI	Implemented
Vasprintf	I	FreeBSD	Implemented
Vfprintf	I	ANSI	Implemented
Vfscanf	I	ANSI	Implemented
Vprintf	I	ANSI	Implemented
Vscanf	I	ANSI	Implemented
Vsnprintf	I	FreeBSD	Implemented
Vsprintf	I	ANSI	Implemented
Vsscanf	I	FreeBSD	Implemented

Stdlib.h	Phase	Standard	Remarks
Abort	I	ANSI	Implemented
Abs	I	ANSI	
Alloca	-	FreeBSD	Dynamic memory allocation on the stack
Arc4random	I	FreeBSD	Implemented
Arc4random_addr	I	FreeBSD	Implemented
Arc4random_stir	I	FreeBSD	Implemented
Atexit	I	ANSI	Implemented
Atof	II	ANSI	
Atoi	I	ANSI	Implemented
Atol	I	ANSI	Implemented
Atoll	-	ANSI	No FreeBSD implementation to port
Bsearch	TBD	ANSI	
Calloc	I	ANSI	Implemented
Cgetcap	-	FreeBSD	Capabilities database stuff
Cgetclose	-	FreeBSD	Capabilities database stuff
Cgetent	-	FreeBSD	Capabilities database stuff
Cgetfirst	-	FreeBSD	Capabilities database stuff
Cgetmatch	-	FreeBSD	Capabilities database stuff
Cgetnext	-	FreeBSD	Capabilities database stuff
Cgetnum	-	FreeBSD	Capabilities database stuff
Cgetset	-	FreeBSD	Capabilities database stuff
Cgetstr	-	FreeBSD	Capabilities database stuff
Cgetustr	-	FreeBSD	Capabilities database stuff
Daemon	-	FreeBSD	No EFI mapping
Devname	-	FreeBSD	No EFI mapping
Div	TBD	ANSI	
Drand48	TBD	FreeBSD	
Erand48	TBD	FreeBSD	
Exit	I	FreeBSD	Implemented
Free	I	ANSI	Implemented
Getbsize	TBD	FreeBSD	
Getenv	I	ANSI	Implemented

Stdlib.h	Phase	Standard	Remarks
Getloadavg	-	FreeBSD	No EFI mapping
Getopt	I	EFI	Implemented
Group_from_gid	-	FreeBSD	No EFI mapping
Heapsort	TBD	FreeBSD	
Initstate	I	FreeBSD	Implemented
Jrand48	TBD	FreeBSD	
Labs	I	ANSI	
Lcong48	TBD	FreeBSD	
Ldiv	TBD	ANSI	
Llabs	-	ANSI	No FreeBSD implementation to port
Lldiv	-	ANSI	No FreeBSD implementation to port
Lrand48	TBD	FreeBSD	
Malloc	I	ANSI	Implemented
Mblen	I	ANSI	Implemented
Mbstowcs	I	ANSI	Implemented
Mbtowc	I	ANSI	Implemented
Mergesort	TBD	FreeBSD	
Mrand48	TBD	FreeBSD	
Nrand48	TBD	FreeBSD	
Putenv	I	FreeBSD	Implemented
Qsort	I	ANSI	Implemented
Radixsort	TBD	FreeBSD	
Rand	I	ANSI	Implemented
Random	I	FreeBSD	Implemented
Realloc	I	ANSI	Implemented
Reallocf	I	FreeBSD	Implemented
Realpath	-	FreeBSD	No EFI mapping
Seed48	TBD	FreeBSD	
Setenv	I	FreeBSD	Implemented
Setstate	I	FreeBSD	Implemented
Sradixsort	TBD	FreeBSD	
Srand	I	ANSI	Implemented
Srand48	TBD	FreeBSD	
Srandom	I	FreeBSD	Implemented
Srandomdev	I	FreeBSD	Implemented
Strtod	I	ANSI	Implemented
Strtol	I	ANSI	Implemented
Strtof	-	ANSI	No FreeBSD implementation to port
Strtoimax	-	ANSI	No FreeBSD implementation to port
Strtold	-	ANSI	No FreeBSD implementation to port
Strtoull	-	ANSI	No FreeBSD implementation to port
Strtoq	I	FreeBSD	Implemented
Strtoul	I	ANSI	Implemented
Strtoumax	-	ANSI	No FreeBSD implementation to port
Strtouq	I	FreeBSD	Implemented
System	II	ANSI	
Unsetenv	I	FreeBSD	Implemented
User_from_uid	-	FreeBSD	No EFI mapping
Wcstombs	I	ANSI	Implemented
Wctomb	I	ANSI	Implemented

String.h	Phase	Standard	Remarks
Bcmp	I	FreeBSD	Implemented
Bcopy	I	FreeBSD	Implemented
Bzero	I	FreeBSD	Implemented

String.h	Phase	Standard	Remarks
Ffs	I	FreeBSD	Implemented
Index	I	FreeBSD	Implemented
Memccpy	I	FreeBSD	Implemented
Memchr	I	ANSI	Implemented
Memcmp	I	ANSI	Implemented
Memcpy	I	ANSI	Implemented
Memmove	I	ANSI	Implemented
Memset	I	ANSI	Implemented
Rindex	I	FreeBSD	Implemented
Strcasecmp	I	FreeBSD	Implemented
Strcat	I	ANSI	Implemented
Strchr	I	ANSI	Implemented
Strcmp	I	ANSI	Implemented
Strcoll	I	ANSI	Implemented
Strcpy	I	ANSI	Implemented
Strcspn	I	ANSI	Implemented
Strdup	I	FreeBSD	Implemented
Strerror	I	ANSI	Implemented
Strlen	I	ANSI	Implemented
Strmode	I	FreeBSD	Implemented
Strncasecmp	I	FreeBSD	Implemented
Strncat	I	ANSI	Implemented
Strncmp	I	ANSI	Implemented
Strncpy	I	ANSI	Implemented
Strpbrk	I	ANSI	Implemented
Strrchr	I	ANSI	Implemented
Strsep	I	FreeBSD	Implemented
Strspn	I	ANSI	Implemented
Strstr	I	ANSI	Implemented
Strtok	I	ANSI	Implemented
Strtok_r	I	FreeBSD	Implemented
Strxfrm	I	ANSI	Implemented
Swab	I	FreeBSD	Implemented

Time.h	Phase	Standard	Remarks
			Probably don't need to support all of these functions. The standard list is much shorter.
Asctime	I	ANSI	Implemented
Asctime_r	TBD	FreeBSD	
Clock	TBD	ANSI	
Clock_getres	TBD	FreeBSD	
Clock_gettime	TBD	FreeBSD	
Clock_settime	TBD	FreeBSD	
Ctime	I	ANSI	Implemented
Ctime_r	TBD	FreeBSD	
Difftime	I	ANSI	Implemented
Gettimeofday	I	FreeBSD	Implemented
Gmtime	I	ANSI	Implemented
Gmtime_r	TBD	FreeBSD	
Localtime	I	ANSI	Implemented
Mktime	I	ANSI	Implemented
Nanosleep	TBD	FreeBSD	
Posix2time	I	FreeBSD	Implemented
Strftime	I	ANSI	Implemented

Time.h	Phase	Standard	Remarks
Strfxtime	-	ANSI	No FreeBSD implementation to port
Strptime	I	FreeBSD	Implemented
Time	I	ANSI	Implemented
Time2posix	I	FreeBSD	Implemented
Timegm	I	FreeBSD	Implemented
Timelocal	TBD	FreeBSD	
Timezone	TBD	FreeBSD	
Tzset	I	FreeBSD	Implemented
Tzsetwall	I	FreeBSD	Implemented
Zonetime	TBD	ANSI	No FreeBSD implementation to port
Mkxtime	TBD	ANSI	

Unistd.h, Fcntl.h, Stat.h, Ioctl.h, Sysctl.h	Phase	Standard	Remarks
Close	I	FreeBSD	Implemented
Dup	I	FreeBSD	Implemented
Dup2	I	FreeBSD	Implemented
Exit	I	FreeBSD	Implemented
Fcntl	I	FreeBSD	Implemented
Fstat	I	FreeBSD	Implemented
Getdtablesize	I	FreeBSD	Implemented
Getpagesize	I	FreeBSD	Implemented
Getpid	I	FreeBSD	Implemented
Ioctl	I	FreeBSD	Implemented
Isatty	I	FreeBSD	Implemented
Lseek	I	FreeBSD	Implemented
Lstat	I	FreeBSD	Implemented
Mkdir	I	FreeBSD	Implemented
Mkdtemp	-	FreeBSD	Needs to be able to make directory
Mkstemp	I	FreeBSD	Implemented
Mktemp	I	FreeBSD	Implemented
Open	I	FreeBSD	Implemented
Read	I	FreeBSD	Implemented
Rmdir	I	FreeBSD	Implemented
Select	I	FreeBSD	Implemented
Sleep	I	FreeBSD	Implemented
Stat	I	FreeBSD	Implemented
Sysctl	I	FreeBSD	Implemented
Unlink	I	FreeBSD	Implemented
Usleep	I	FreeBSD	Implemented
Write	I	FreeBSD	Implemented
Writev	I	FreeBSD	Implemented

Wchar.h	Phase	Standard	Remarks
Btowc	II	ANSI	
Fgetwc	II	ANSI	
Fgetws	II	ANSI	
Fputwc	II	ANSI	
Fputws	II	ANSI	
Fwide	II	ANSI	
Fwprintf	II	ANSI	
Fwscanf	II	ANSI	
Getwc	II	ANSI	

Wchar.h	Phase	Standard	Remarks
Getwchar	II	ANSI	
Mbrlen	II	ANSI	
Mbrtowc	II	ANSI	
Mbsinit	II	ANSI	
Mbsrtowcs	II	ANSI	
Putwc	II	ANSI	
Putwchar	II	ANSI	
Swprintf	II	ANSI	
Swscanf	II	ANSI	
Ungetwc	II	ANSI	
Vfwprintf	II	ANSI	
Vfwscanf	II	ANSI	
Vswprintf	II	ANSI	
Vswscanf	II	ANSI	
Vwprintf	II	ANSI	
Vwscanf	II	ANSI	
Wcrtomb	II	ANSI	
Wcscat	I	ANSI	Implemented
Wcschr	I	ANSI	Implemented
Wcscmp	I	ANSI	Implemented
Wcscoll	II	ANSI	Requires locale support
Wcscpy	I	ANSI	Implemented
Wcscspn	I	ANSI	Implemented
Wcsftime	II	ANSI	
Wcsfxtime	II	ANSI	
Wcslen	I	ANSI	Implemented
Wcsncat	I	ANSI	Implemented
Wcsncmp	I	ANSI	Implemented
Wcsncpy	I	ANSI	Implemented
Wcspbrk	I	ANSI	Implemented
Wcsrchr	I	ANSI	Implemented
Wcsrtombs	II	ANSI	
Wcsspn	I	ANSI	Implemented
Wcsstr	I	ANSI	Implemented
Wcstod	II	ANSI	
Wcstof	II	ANSI	
Wcstok	I	ANSI	Implemented
Wcstol	II	ANSI	
Wcstold	II	ANSI	
Wcstoll	II	ANSI	
Wcstoul	II	ANSI	
Wcstoull	II	ANSI	
Wcsxfrm	II	ANSI	Requires locale support
Wctob	II	ANSI	
Wgetopt	I	EFI	Implemented
Wmemchr	I	ANSI	Implemented
Wmemcmp	I	ANSI	Implemented
Wmemcpy	I	ANSI	Implemented
Wmemmove	I	ANSI	Implemented
Wmemset	I	ANSI	Implemented
Wprintf	II	ANSI	
Wscanf	II	ANSI	

Wctype.h	Phase	Standard	Remarks
			Most functions have been implemented

Wctype.h	Phase	Standard	Remarks
			just for US English. To do them right would require locale support (see ctype.h implementation).
Iswalnum	I	ANSI	Implemented / US English only
Iswalpha	I	ANSI	Implemented / US English only
Iswcntrl	I	ANSI	Implemented / US English only
Iswctype	II	ANSI	
Iswdigit	I	ANSI	Implemented / US English only
Iswgraph	I	ANSI	Implemented / US English only
Iswlower	I	ANSI	Implemented / US English only
Iswprint	I	ANSI	Implemented / US English only
Iswpunct	I	ANSI	Implemented / US English only
Iswspace	I	ANSI	Implemented / US English only
Iswupper	I	ANSI	Implemented / US English only
Iswxdigit	I	ANSI	Implemented / US English only
Towctrans	II	ANSI	
Towlower	I	ANSI	Implemented / US English only
Towupper	I	ANSI	Implemented / US English only
Wctrans	II	ANSI	
Wctype	II	ANSI	

Appendix 2: Glossary

BIOS	Basic Input/Output System. The system BIOS is embedded software located on the system baseboard. The BIOS executes Power On Self Test (POST) to test and initialize system devices then bootstraps the operating system.
Device Path	An identifier used to locate and differentiate devices on the platform in the boot services envrionment.
EPS	External Product Specification. Document describing appearance, behavior, and programming interface.
GUID	Globally unique identifier. A 128-bit value used to differentiate services and structures in the boot services environment.
IA	Intel Architecture.
OS	Operating System. The server's main operating system.
Protocol Interface	A set of interface functions approriate to a particular type of device.
Unicode	A standard that provides a comprehensive character set and establishes rules for its use. The current Unicode character set includes alphabetic, syllabic, and ideographic characters.