SunOS リファレンスマニュアル 3 : ライブラリインタフェースおよびヘッダー

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–1215–10
2005 年 1 月

原典: *man pages section 3: Library Interfaces and Headers*

Part No: 816-5173-10

Revision A

050206@10536

目次

はじめに

概要

SunOS リファレンスマニュアルは、初めて SunOS を使用するユーザーやすでにある程度の知識を持っているユーザーのどちらでも対応できるように解説されています。このマニュアルを構成するマニュアルページは一般に参照マニュアルとして作られており、チュートリアルな要素は含んでいません。それぞれのコマンドを実行すると、どのような結果が得られるかについて、詳しく説明されています。なお、各マニュアルページの内容はオンラインでも参照することができます。

このマニュアルは、マニュアルページの内容によっていくつかのセクションに分かれています。各セクションについて以下に簡単に説明します。

- セクション 1 は、オペレーティングシステムで使えるコマンドを説明します。
- セクション 1M は、システム保守や管理用として主に使われるコマンドを説明します。
- セクション 2 は、すべてのシステムコールについて説明します。ほとんどのシステムコールに 1 つまたは複数のエラーがあります。エラーの場合、通常ありえない戻り値が返されます。
- セクション 3 は、さまざまなライブラリ中の関数について説明します。ただし、UNIX システムプリミティブを直接呼び出す関数については、セクション 2 で説明しています。
- セクション 4 は、各種ファイルの形式について説明します。また、ファイル形式を宣言する C 構造体を適用できる場合には随時説明しています。
- セクション 5 は、文字セットテーブルなど他のセクションには該当しないものについて説明します。
- セクション 7 は、特殊なハードウェア周辺装置またはデバイスドライバに関するさまざまな特殊ファイルについて説明します。STREAMS ソフトウェアドライバ、モジュール、またはシステムコールの STREAMS 汎用セットについても説明します。

- セクション 9 は、カーネル環境でデバイスドライバを記述するのに必要な参照情報を提供します。ここでは、デバイスドライバインタフェース (DDI) とドライバ/カーネルインタフェース (DKI) という 2 つのデバイスドライバインタフェース仕様について説明します。
- セクション 9F は、デバイスドライバが使用できるカーネル関数について説明します。

以下に、このマニュアルの項目を表記されている順に説明します。ほとんどのマニュアルページが下記の項目からなる共通の書式で書かれていますが、必要でない項目については省略されています。たとえば、記述すべきバグがコマンドにない場合などは、「使用上の留意点」という項目はありません。各マニュアルページの詳細は各セクションの `intro` を、マニュアルページの一般的な情報については `man`(1) を参照してください。

名前　コマンドや関数の名称と概略が示されています。

形式　コマンドや関数の構文が示されています。標準パスにコマンドやファイルが存在しない場合は、フルパス名が示されます。字体は、コマンド、オプションなどの定数にはボールド体 (bold) を、引数、パラメータ、置換文字などの変数にはイタリック体 (Italic) または <日本語訳> を使用しています。オプションと引数の順番は、アルファベット順です。特別な指定が必要な場合を除いて、1 文字の引数、引数のついたオプションの順に書かれています。

以下の文字がそれぞれの項目で使われています。

[ ]　このかっこに囲まれたオプションや引数は省略できます。このかっこが付いていない場合には、引数を必ず指定する必要があります。

. . .　省略符号。前の引数に変数を付けたり、引数を複数指定したりできることを意味します (例: ‘ `filename` . . .’)。

|　区切り文字 (セパレータ)。この文字で分割されている引数のうち 1 つだけを指定できます。

{ }　この大かっこに囲まれた複数のオプションや引数は省略できます。かっこ内を 1 組として扱います。

プロトコル　この項が使われているのは、プロトコルが記述されているファイルを示すサブセクション 3R だけです。パス名は常にボールド体 (bold) で示されています。

機能説明　コマンドの機能とその動作について説明します。実行時の詳細を説明していますが、オプションの説明や使用例はここでは示されていません。対話形式のコマンド、サブコマンド、リクエスト、マクロ、関数などに関しては「使用法」で説明します。

IOCTL　セクション 7 だけに使用される項です。`ioctl`(2) システムコールへのパラメータは `ioctl` と呼ばれ、適切なパラメータを持つデバイスクラスのマニュアルページだけに記載されています。特定の

デバイスに関する ioctl は、(そのデバイスのマニュアルページに) アルファベット順に記述されています。デバイスの特定のクラスに関する ioctl は、mtio(7I) のように io で終わる名前が付いているデバイスクラスのマニュアルページに記載されています。

オプション　各オプションがどのように実行されるかを説明しています。「形式」で示されている順に記述されています。オプションの引数はこの項目で説明され、必要な場合はデフォルト値を示します。

オペランド　コマンドのオペランドを一覧表示し、各オペランドがコマンドの動作にどのように影響を及ぼすかを説明しています。

出力　コマンドによって生成される出力 (標準出力、標準エラー、または出力ファイル) を説明しています。

戻り値　値を返す関数の場合、その値を示し、値が返される時の条件を説明しています。関数が 0 や -1 のような一定の値だけを返す場合は、値と説明の形で示され、その他の場合は各関数の戻り値について簡単に説明しています。void として宣言された関数はこの項では扱いません。

エラー　失敗の場合、ほとんどの関数はその理由を示すエラーコードを errno 変数の中に設定します。この項ではエラーコードをアルファベット順に記述し、各エラーの原因となる条件について説明します。同じエラーの原因となる条件が複数ある場合は、エラーコードの下にそれぞれの条件を別々のパラグラフで説明しています。

使用法　この項では、使用する際の手がかりとなる説明が示されています。特定の決まりや機能、詳しい説明の必要なコマンドなどが示されています。組み込み機能については、以下の小項目で説明しています。

コマンド
修飾子
変数
式
入力文法

使用例　コマンドや関数の使用例または使用方法を説明しています。できるだけ実際に入力するコマンド行とスクリーンに表示される内容を例にしています。例の中には必ず example% のプロンプトが出てきます。スーパーユーザーの場合は example# のプロンプトになります。例では、その説明、変数置換の方法、戻り値が示され、それらのほとんどが「形式」、「機能説明」、「オプション」、「使用法」の項からの実例となっています。

環境　コマンドや関数が影響を与える環境変数を記述し、その影響について簡単に説明しています。

終了ステータス	コマンドが呼び出しプログラムまたはシェルに返す値と、その状態を説明しています。通常、正常終了には 0 が返され、0 以外の値はそれぞれのエラー状態を示します。
ファイル	マニュアルページが参照するファイル、関連ファイル、およびコマンドが作成または必要とするファイルを示し、各ファイルについて簡単に説明しています。
属性	属性タイプとその対応する値を定義することにより、コマンド、ユーティリティ、およびデバイスドライバの特性を一覧しています。詳細は `attributes`(5) を参照してください。
関連項目	関連するマニュアルページ、当社のマニュアル、および一般の出版物が示されています。
診断	エラーの発生状況と診断メッセージが示されています。メッセージはボールド体 (bold) で、変数はイタリック体 (Italic) または <日本語訳> で示されており、C ロケール時の表示形式です。
警告	作業に支障を与えるような現象について説明しています。診断メッセージではありません。
注意事項	それぞれの項に該当しない追加情報が示されています。マニュアルページの内容とは直接関係のない事柄も参照用に扱っています。ここでは重要な情報については説明していません。
使用上の留意点	すでに発見されているバグについて説明しています。可能な場合は対処法も示しています。

序章

名前 | Intro – 関数とライブラリの紹介

機能説明 | このセクションでは、さまざまな Solaris ライブラリにある関数のうち、直接 UNIX システムのプリミティブを呼び出す関数 (このマニュアルページのセクション 2 で説明) 以外の関数について説明します。関数の宣言は、各ページで説明されている `#include` ファイルから取得できます。このセクションの説明はライブラリ別になっています。各ライブラリは、セクション番号の後ろに入っているライブラリ名 (ライブラリの省略名) で識別できます。関連のあるライブラリのコレクションは、下記に示すように大きく 5 つのボリュームに分かれており、6 番目のボリュームは、この 5 つのボリュームで説明されている関数、マクロ、外部変数が使用する共有ライブラリとヘッダーの内容説明になっています。ここでは、まず、この 6 番目のボリュームから説明します。

ライブラリインタフェースおよびヘッダー | このボリュームでは、他の 5 つのボリュームで説明されている関数、マクロ、外部変数が使用する共有ライブラリとヘッダーの内容について説明します。

(3LIB) | このセクションで説明するライブラリはどれも、共有オブジェクトとして実装されています。

共有オブジェクトの説明には、共有オブジェクトの共有インタフェースを定義する大域シンボルの定義が含まれていることがあります (例: `SUNW_1.1`)。このほかのインタフェースは、共有オブジェクトに含まれている可能性があります (例: `SUNW_private.1.1`)。共有インタフェースは、アプリケーション開発用に、安定度の高い専用のシンボルセットを提供します。専用インタフェースは内部使用のみであり、随時変更される可能性があります。

(3LIBUCB) | このセクションで説明する SunOS/BSD 互換ライブラリは、共有オブジェクトとして実装されています。上記の (3LIB) を参照してください。

(3HEAD) | このセクションで説明するヘッダーは、関数、マクロ、外部変数が使用するヘッダーです。ヘッダーには、関数のプロトタイプ、記号定数の定義、共通の構造体、プリプロセッサマクロ、定義済みの型が含まれています。残りの 5 つのボリュームで説明されている各関数は、この関数を使用するためにアプリケーションに含めなければならないヘッダーを指定します。多くの場合、必要なヘッダーは 1 つだけです。このようなヘッダーは、アプリケーション開発システムに配置されます。実行システムに配置する必要はありません。

基本ライブラリ関数 | このボリュームで説明する関数は、アプリケーション開発の基本となるコア C ライブラリ関数です。

(3C) | セクション 2 の関数とともに標準 C ライブラリ `libc` を構成する関数です。`libc` は、C コンパイルシステムにより自動的にリンクされます。標準 C ライブラリは共有オブジェクト `libc.so` として実装されています。`libc`(3LIB) を参照してください。『*ANSI C Programmer's Guide*』の「C Compilation System」の章にも参考

になる情報があります。規格に合致した環境では、動作が異なる関数もあります。こうした動作については、個々のマニュアルページで説明されています。standards(5) を参照してください。

libpthread ライブラリと libthread ライブラリは libc へのフィルタライブラリであり、マルチスレッドアプリケーションを構築するために使用します。libpthread は POSIX (standards(5) を参照) スレッドインタフェースを実装し、libthread は Solaris スレッドインタフェースを実装します。下記のマルチスレッドアプリケーションを参照してください。

(3C_DB) スレッドデバッグライブラリ libc_db を構成する関数です。このライブラリは共有オブジェクト libc_db.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。cc コマンド行で -lc_db を指定すると、このライブラリにリンクできます。libc_db(3LIB) を参照してください。

(3MALLOC) libmalloc、libbsdmalloc、libmapmalloc、libmtmalloc という複数のメモリー割り当てライブラリを構成する関数です。これらのライブラリはそれぞれ共有オブジェクト libmalloc.so、libbsdmalloc.so、libmapmalloc.so、libmtmalloc.so および libumem.so として実装されています。このライブラリ群は、C コンパイルシステムによって自動的にはリンクされません。libmalloc、libbsdmalloc、libmapmalloc、libmtmalloc とリンクするには、それぞれ -lmalloc、-lbsdmalloc、-lmapmalloc、-lmtmalloc の各オプションを指定します。libmalloc(3LIB)、libbsdmalloc(3LIB)、libmapmalloc(3LIB)、libmtmalloc(3LIB) を参照してください。

(3UCB) BSD 関数を含むソース互換ライブラリを構成する関数です。共有オブジェクト libucb.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリは /usr/ucb サブディレクトリにあるため、cc コマンド行で -lucb を指定することによってリンクできます。このライブラリのヘッダーは、/usr/ucbinclude に入っています。libucb(3LIBUCB) を参照してください。

ネットワーキングライブラリ関数

このボリュームで説明する関数は、さまざまなネットワーキングライブラリを構成します。

(3GSS) このライブラリ中の関数は、Generic Security Services API (GSS-API) ライブラリを構成するルーチンです。このライブラリは共有オブジェクト libgss.so として実装されています。ただし、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lgss を指定します。libgss(3LIB) を参照してください。

(3LDAP)	軽量ディレクトリアクセスプロトコルライブラリを構成する関数です。このライブラリは共有オブジェクト `libldap.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lldap` を指定します。`ldap`(3LDAP) を参照してください。
(3NSL)	ネットワークサービスライブラリ `libnsl` を構成する関数です。このライブラリは共有オブジェクト `libnsl.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lnsl` を指定します。`libnsl`(3LIB) を参照してください。 基本ネットワーキング関数の多くは、X/Open ネットワーキングインタフェースライブラリにも入っています。`libxnet` インタフェースの詳細については、下記のセクション (3XNET) を参照してください。
(3RAC)	リモート非同期呼び出しライブラリ `librac` を構成する関数です。このライブラリは共有オブジェクト `librac.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lrac` を指定します。`librac`(3LIB) を参照してください。
(3RESOLV)	リゾルバライブラリ `libresolv` を構成する関数です。このライブラリは共有オブジェクト `libresolv.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lresolv` を指定します。`libresolv`(3LIB) を参照してください。
(3RPC)	遠隔手続き呼び出しライブラリ `librpcsvc` と `librpcsoc` を構成する関数です。`librpcsoc` は互換性の提供のみを目的としたライブラリなので、これには新しいアプリケーションをリンクしないでください。この 2 つのライブラリは、それぞれ共有オブジェクト `librpcsvc.so` と `librpcsoc.so` として実装されています (`librt`(3LIB) 参照)。どちらのライブラリも、C コンパイルシステムによって自動的にはリンクされません。これらにリンクするには、`cc` コマンド行で `-lrpcsvc` か `-lrpcsoc` を指定します。`librpcsvc`(3LIB) および `librpcsoc`(3LIBUCB) を参照してください。
(3SLP)	SLP (サービスロケーションプロトコル, Service Location Protocol) ライブラリ `libslp` を構成する関数です。このライブラリは、共有オブジェクト `libslp.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。`libslp`(3LIB) を参照してください。
(3SASL)	シンプル認証とセキュリティ層 (SASL: Simple Authentication and Security Layer) ライブラリ `libsasl` を構成する関数です。SASL は、接続指向のネットワークアプリケーションが主に認証に使用

するセキュリティのフレームワークです。言い換えると、SASL は、ネットワークアプリケーションといくつかのセキュリティ機構を接合する接着剤の層です。したがって、アプリケーションはお互いに認証できるとともに、データの暗号化などの追加のセキュリティサービスも利用できます。接着剤の層として、SASL は、セキュリティ機構に固有なインタフェースをアプリケーションに見せないため、新しいセキュリティ機構を実装しても、柔軟かつ迅速に移植できます。

`libsasl` は、アプリケーションに対しては API を提供し、さまざまなプラグインに対しては SPI を提供します。このライブラリにリンクするには、`cc` コマンド行で `-lsasl` を指定します。`libsasl`(3LIB) を参照してください。

(3SOCKET) ソケットライブラリ `libsocket` を構成する関数です。このライブラリは、共有オブジェクト `libsocket.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lsocket` を指定します。`libsocket`(3LIB) を参照してください。

(3XNET) X/Open CAE 仕様『Networking Services』(Issue 4、1994 年 9 月) に準拠した X/Open ネットワーキングインタフェースを構成する関数です。このライブラリは共有オブジェクト `libxnet.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lxnet` を指定します。コンパイル情報については、`libxnet`(3LIB) および `standards`(5) を参照してください。

どのような状況でも、XTI API や TLI API ではなく、ソケット API を優先的に使用することをお勧めします。ただし、ほかの XPGV4v2 (`standards`(5) を参照) システムに移植する必要がある場合は、アプリケーションで `libxnet` インタフェースを使用する必要があります。移植の必要がない場合は、`libxnet` のインタフェースではなく、`libsocket` や `libnsl` のソケットインタフェースを使用する方法をお勧めします。XTI API、TLI API の中では、`libnsl` で使用できる TLI インタフェースよりも `libxnet` で使用できる XTI インタフェースのほうをお勧めします。

Curses ライブラリ関数

このボリュームで説明する関数は、グラフィックスと文字が表示された画面の更新機能を提供するライブラリを構成します。

(3CURSES) 次のライブラリを構成する関数です。

libcurses curses ライブラリ `libcurses` を構成する関数です。このライブラリは共有オブジェクト `libcurses.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lcurses` を指定します。`libcurses`(3LIB) を参照してください。

	libform	forms ライブラリ `libform` を構成する関数です。このライブラリは共有オブジェクト `libform.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lform` を指定します。`libform`(3LIB) を参照してください。
	libmenu	menus ライブラリ `libmenu` を構成する関数です。このライブラリは共有オブジェクト `libmenu.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lmenu` を指定します。`libmenu`(3LIB) を参照してください。
	libpanel	panels ライブラリ `libpanel` を構成する関数です。このライブラリは共有オブジェクト `libpanel.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lpanel` を指定します。`libpanel`(3LIB) を参照してください。
	(3PLOT)	グラフィックスライブラリ `libplot` を構成する関数です。このライブラリは共有オブジェクト `libplot.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-lplot` を指定します。`libplot`(3LIB) を参照してください。
	(3XCURSES)	`/usr/xpg4/lib/libcurses.so` にある X/Open Curses ライブラリを構成する関数です。このライブラリは、端末画面での入力と出力、その変更に使用する国際化された関数とマクロのセットを提供します。このライブラリには、ウィンドウの作成、テキストの強調表示、画面への書き込み、ユーザーの入力の読み取り、カーソルの移動などを実行するための関数が含まれています。X/Open Curses は、画面更新動作を最適化するように設計されています。X/Open Curses ライブラリは、X/Open Extended Curses 仕様 (Issue 4) に全面的に準拠していいます。
リアルタイムライブラリ関数	このボリュームで説明する関数は、リアルタイムライブラリを構成します。	
	(3AIO)	非同期入出力ライブラリ `liaio` を構成する関数です。このライブラリは共有オブジェクト `libaio.so` として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、`cc` コマンド行で `-laio` を指定します。`libaio`(3LIB) を参照してください。

(3DOOR) doors ライブラリ libdoor を構成する関数です。このライブラリは共有オブジェクト libdoor.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -ldoor を指定します。

(3RT) POSIX.4 リアルタイムライブラリ librt を構成する関数です。このライブラリは共有オブジェクト librt.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lrt を指定します。このライブラリは、以前は libposix4 という名前でした。下位互換のため、この libposix4 も維持されていますが、こちらの使用はできるだけ避けてください。librt(3LIB) を参照してください。

拡張ライブラリ関数

このボリュームで説明する関数は、特化されたさまざまなライブラリを構成します。構成されるライブラリは、次に示す限りではありません。

(3BSM) 基本セキュリティライブラリ libbsm を構成する関数です。このライブラリは共有オブジェクト libbsm.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lbsm を指定します。libbsm(3LIB) を参照してください。

(3CFGADM) 構成管理ライブラリ libcfgadm を構成する関数です。このライブラリは共有オブジェクト libcfgadm.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lcfgadm を指定します。libcfgadm(3LIB) を参照してください。

(3CONTRACT) 契約管理ライブラリ libcontract を構成する関数です。このライブラリは共有オブジェクト libcontract.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -lcontract を指定します。libcontract(3LIB) を参照してください。

(3CPC) CPU パフォーマンスカウンタライブラリ libcpc およびプロセスコンテキストライブラリ libpctx を構成する関数です。これらのライブラリはそれぞれ共有オブジェクト libcpc.so および libpctx.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lcpc を指定します。libcpc(3LIB) および libpctx(3LIB)を参照してください。

(3DAT) 直接アクセス転送ライブラリ libdevid を構成する関数です。このライブラリは共有オブジェクト libdat.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -ldat を指定します。libdat(3LIB) を参照してください。

(3DEVID) デバイス ID ライブラリ libdevid を構成する関数です。このライブラリは共有オブジェクト libdevid.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -ldevid を指定します。libdevid(3LIB) を参照してください。

(3DEVINFO) デバイス情報ライブラリ libbsm を構成する関数です。このライブラリは共有オブジェクト libdevinfo.so、アーカイブ libdevinfo.a として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -ldevinfo を指定します。libdevinfo(3LIB) を参照してください。

(3DMI) DMI ライブラリ libdmi、libdmici、libdmimi を構成する関数です。これらのライブラリはそれぞれ、共有オブジェクト libdmi.so、libdmici.so、libdmimi.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。これらにリンクするには、cc コマンド行で -ldmi、-ldmici、-ldmimi のいずれかを指定します。libdmi(3LIB)、libdmici(3LIB)、libdmimi(3LIB) を参照してください。

(3ELF) ELF (Extensible Linking Format) アクセスライブラリ libelf を構成する関数です。このライブラリは elf ファイルの作成と分析用のインタフェースを提供します。elf ファイルには、実行ファイル、オブジェクト、共有オブジェクトがあります。libelf は共有オブジェクト libelf.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lelf を指定します。libelf(3LIB) を参照してください。

(3EXACCT) 拡張アカウンティングアクセスライブラリ libexacct およびプロジェクトデータベースアクセスライブラリ libproject を構成する関数です。これらのライブラリはそれぞれ、共有オブジェクト libexacct.so および libproject.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。これらのライブラリにリンクするには、cc コマンド行で -lexacct または -lproject を指定します。libexacct(3LIB) および libproject(3LIB) を参照してください。

(3GEN) 文字列のパターンマッチングやパス名の操作を行うライブラリ libgen を構成する関数です。このライブラリは共有オブジェクト libgen.so として実装されていますが、C コンパイルシステ

ムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lgen を指定します。libgen(3LIB) を参照してください。

(3HBAAPI) 共通ファイバチャネル HBA 情報ライブラリ libhbaapi を構成する関数です。このライブラリは共有オブジェクト libhbaapi.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -lhbaapi を指定します。libhbaapi(3LIB) を参照してください。

(3KSTAT) カーネル統計情報ライブラリを構成する関数です。このライブラリは共有オブジェクト libkstat.so、アーカイブ libkstat.a として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lkstat を指定します。libkstat(3LIB) を参照してください。

(3KVM) カーネルの仮想メモリーライブラリへのアクセスを許可する関数です。このライブラリは共有オブジェクト libkvm.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lkvm を指定します。libkvm(3LIB) を参照してください。

(3LAYOUT) レイアウトサービスライブラリを構成する関数です。このライブラリは共有オブジェクト liblayout.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -llayout を指定します。liblayout(3LIB) を参照してください。

(3LGRP) ローカリティグループライブラリを構成する関数です。このライブラリは共有オブジェクト liblgrp.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -llgrp を指定します。liblgrp(3LIB) を参照してください。

(3M) 数学ライブラリ libm を構成する関数です。このライブラリは共有オブジェクト libm.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lm を指定します。

(3MAIL) ユーザーのメールボックス管理ライブラリ libmail を構成する関数です。このライブラリは共有オブジェクト libmail.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lmail を指定します。

(3MP)	整数型数学ライブラリ libmp を構成する関数です。このライブラリは共有オブジェクト libmp.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lmp を指定します。libmp(3LIB) を参照してください。
(3NVPAIR)	名前と値の組み合せのライブラリ libnvpair を構成する関数です。このライブラリは共有オブジェクト libnvpair.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lnvpair を指定します。libnvpair(3LIB) を参照してください。
(3PAM)	PAM (Pluggable Authentication Module) ライブラリ libpam を構成する関数です。このライブラリは共有オブジェクト libpam.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lpam を指定します。libpam(3LIB) を参照してください。
(3PICL)	PICL ライブラリ libpicl を構成する関数です。このライブラリは共有オブジェクト libpicl.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lpicl を指定します。libpicl(3LIB)libpicl(3LIB) および libpicl(3PICL) を参照してください。
(3PICLTREE)	PICL プラグインライブラリ libpicltree を構成する関数です。このライブラリは共有オブジェクト libpicltree.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lpicltree を指定します。libpicltree(3LIB)libpicltree(3LIB) および libpicltree(3PICLTREE) を参照してください。
(3RSM)	リモート共有メモリー librsm を構成する関数です。このライブラリは共有オブジェクト librsm.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lrsm を指定します。librsm(3LIB) を参照してください。
(3SCF)	オブジェクトキャッシュ用メモリー割り当てライブラリ libscf を構成する関数です。このライブラリは共有オブジェクト libscf.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -lscf を指定します。libscf(3LIB) を参照してください。

(3SEC) ファイルアクセス制御ライブラリ libsec を構成する関数です。このライブラリは共有オブジェクト libsec.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lsec を指定します。libsec(3LIB) を参照してください。

(3SECDB) セキュリティ属性データベースライブラリ libsecdb を構成する関数です。このライブラリは共有オブジェクト libsecdb.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lsecdb を指定します。libsecdb(3LIB) を参照してください。

(3SNMP) SNMP ライブラリ libssagent と libssasnmp を構成する関数です。この 2 つのライブラリは、それぞれ共有オブジェクト libssagent.so、libssasnmp.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。これらにリンクするには、cc コマンド行で -lssagent または -lssasnmp を指定します。libssagent(3LIB) を参照してください。

(3SYSEVENT) システムイベントライブラリ libsysevent を構成する関数です。このライブラリは共有オブジェクト libsysevent.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lsysevent を指定します。libsysevent(3LIB) を参照してください。

(3TECLA) 対話型コマンド行入力ライブラリ libtecla を構成する関数です。このライブラリは共有オブジェクト libtecla.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -ltecla を指定します。libtecla(3LIB) を参照してください。

(3TNF) TNF ライブラリ libtnf、libtnfctl、libtnfprobe を構成する関数です。この 3 つのライブラリは、それぞれ共有オブジェクト libtnf.so、libtnfctl.so、libtnfprobe.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。これらにリンクするには、cc コマンド行で -ltnf、-ltnfctl、-ltnfprobe のいずれかを指定します。libtnfctl(3TNF) および libtnfctl(3LIB) を参照してください。

(3UUID) 普遍一意な識別子ライブラリ libuuid を構成する関数です。このライブラリは共有オブジェクト libuuid.so として実装されていますが、C コンパイルシステムによっては自動的にリンクされません。このライブラリにリンクするには、cc コマンド行で -luuid を指定します。libuuid(3LIB) を参照してください。

(3VOLMGT) ボリューム管理ライブラリ libvolmgt を構成する関数です。このライブラリは共有オブジェクト libvolmgt.so として実装されていますが、C コンパイルシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lvolmgt を指定します。libvolmgt(3LIB)を参照してください。

(3WSREG) 製品インストールレジストリのライブラリ libwsreg を構成する関数です。このライブラリは、共有オブジェクト libwsreg.so として実装されていますが、C コンパイラシステムによって自動的にはリンクされません。このライブラリにリンクするには、cc コマンド行で -lwsreg オプションを指定してください。libwsreg(3LIB) を参照してください。

定義

マシンのバイトに適合するビットパターンを、1 文字と解釈することができます。ただし、言語によっては、1 文字に 1 バイト以上必要で、複数バイトで表現されるものもあります。

NULL 文字とは、値が 0 の文字のことです。C 言語では、便宜上この文字を \0 と表します。文字の並び (シーケンス) のことを文字配列と言います。最後の文字が NULL 文字になっている文字の並びのことを NULL 終了文字配列 (文字列) と言います。また、NULL 文字だけを含む文字配列のことを NULL 文字列と言います。0 をポインタにキャストすることによって得られる値は NULL ポインタと呼ばれています。C 言語では、この値は有効なポインタと一致しません。このため、ポインタを返す関数のほとんどは、NULL を返すことによってエラーを示します。NULL マクロは <stdio.h> で定義されています。size_t の型は適切なヘッダーの中に定義されています。

マルチスレッドアプリケーション

POSIX スレッドと Solaris スレッドは両方とも、同じアプリケーション内で使用できます。これらのスレッドはお互いに完全な互換性がありますが、ほかの POSIX 適合環境への移植性が保証されるのは POSIX スレッドだけです。

libpthread(3LIB) ライブラリと libthread(3LIB) ライブラリは、libc(3LIB) へのフィルタとして実装されます。

マルチスレッドアプリケーションをコンパイルするときには、コマンド行で -mt オプションを指定する必要があります。

マルチスレッドアプリケーションは -lthread とリンクする必要はありません。fork(2) に対する POSIX 意味論が必要でない限り、アプリケーションは -lpthread とリンクする必要はありません。 アプリケーションが -lpthread とリンクしたとき、fork() の呼び出しでは、すべてのスレッドをフォークするというデフォルトの動作ではなく、fork1(2)r の動作が想定されます。

POSIX 適合のアプリケーションをコンパイルするとき、_POSIX_C_SOURCE オプションまたは __POSIX_PTHREAD_SEMANTICS オプションのどちらかをコマンド行で指定する必要があります。POSIX.1c に適合するアプリケーションには、次のように、_POSIX_C_SOURCE オプションに 199506L 以上の値を定義します。

```
cc -mt [ flag... ] file... -D_POSIX_C_SOURCE=199506L -lpthread
```

Solaris の fork() と fork1() を区別する POSIX の動作の場合、次のようにコンパイルします。

```
cc -mt [ flag... ] file... -D_POSIX_PTHREAD_SEMANTICS
```

Solaris スレッドの動作の場合、次のようにコンパイルします。

```
cc -mt [ flag... ] file...
```

アプリケーションの安全性を保証するため、安全でないインタフェースはメインスレッドだけから呼び出すようにします。

MT 安全なインタフェースについては、各関数やライブラリのマニュアルページの「属性」の項目に記述されています (attributes(5) を参照)。インタフェースが MT 安全であることがマニュアルページに明示的に述べられていない場合、そのインタフェースは安全でないことを想定する必要があります。

リアルタイムアプリケーション

早期のバインディングを可能にするためは、環境変数 LD_BIND_NOW の値を非 NULL 値に設定してください。詳細については『リンカーとライブラリ』の再配置に関する説明を参照してください。

ファイル

INCDIR	通常は /usr/include
LIBDIR	通常は /lib または /usr/lib (32ビットの場合) あるいは、/lib/64 または /usr/lib/64 (64ビットの場合) です。
LIBDIR/*.so	共有ライブラリ

関連項目

ar(1), cc(1B), ld(1), fork(2), intro(3), stdio(3C), attributes (5), standards(5)

『リンカーとライブラリ』

『プロファイリングツール』

『*ANSI C Programmer's Guide*』

『診断』

浮動小数点の値を戻す関数の場合、エラーの処理方法はコンパイルモードによって変わります。-Xt オプション (デフォルト) を cc コマンド行で指定している場合、指定した引数に対して関数が定義されていないとき、あるいは、値が表現不可能なとき、これらの関数は、従来の 0、±HUGE、または NaN を戻します。ところが、-Xa オプションまたは -Xc オプションを指定している場合、これらの関数は、±HUGE ではなく、±HUGE_VAL を戻します。±HUGE_VAL と HUGE は math.h で定義されており、それぞれ、無限と単一精度の最大数を表します。

注意事項

ここで説明する関数、外部変数、マクロは、ユーザープログラムの中で再定義しないでください。他の名前を再定義してもこれらのライブラリの動作に影響を与えませんが、同じ名前を再定義すると、インクルードされているヘッダーの宣言と衝突してしまいます。

INCDIR の中にあるヘッダーは、このマニュアルページで説明するほとんどの関数のプロトタイプを提供します。関数のプロトタイプとは、引数の型を含む関数宣言のことです。関数のプロトタイプを使用すると、ユーザーのプログラムの中で使用されている関数の使用法が間違っていないか、コンパイラでチェックすることができます。コンパイラの代わりに `lint` プログラムチェッカーを使用することもできます。`lint` は、ヘッダーが `#include` 文でインクルードされていないことを検出すると、この矛盾を報告します。セクション 2、3C、3S の定義は自動的にチェックされます。その他の定義をチェックするには、`lint` に `-l` オプションを指定します。たとえば、`libm` の定義をチェックするには `-lm` を指定します。できるだけ `lint` を使用するようにしてください。詳細は、『プロファイリングツール』の `lint` に関する章を参照してください。

「STREAMS」と「ストリーム」の違いには十分気を付けてください。STREAMS とは、ネットワークサービスとデータ通信ドライバの開発をサポートするカーネル機構の一種であり、ユーティリティルーチン、カーネル機能、データ構造体で構成されています。一方、ストリーム とは、STREAMS に関連するバッファリングで使用されるファイルのことです。ストリーム は `<stdio.h>` の中で `FILE` 型へのポインタとして定義されています。

各要素の詳細な定義では、実装に固有のシンボリック名を参照しなければならない場合があります。ただし、必ずしもアプリケーションプログラムで使用できるようにする必要はありません。こうしたシンボリック名の多くは、境界の条件やシステムの制限を記述するものです。

このセクションでは、分かりやすくするために、このような実装固有の値をシンボリック名にしています。このようなシンボリック名は必ず中括弧 ({}) で囲まれ、ヘッダーを使ってアプリケーションからアクセスできる実装固有の定数のシンボリック名と区別されます。中括弧で囲まれているシンボリック名は、特定のシステムの文書中で定義されていることもありますが、ヘッダーを使用することによってアプリケーションプログラムから必ずしもアクセスできるわけではありません。

一般には、移植可能なアプリケーションプログラムコードの中で、このようなシンボリック名が参照されていてはなりません。たとえば、あるアプリケーションプログラムで、あるルーチンに提供された引数リストの長さが {`ARG_MAX`} より大きいかどうかをテストしてはなりません。

X/Open XFN 標準に基づいたフェデレーテッドネーミングサービスは、Solaris オペレーティングシステムの将来のリリースではサポートされない可能性があります。

インタフェースライブラリ

libcfgadm(3LIB)

名前 | libcfgadm – 構成管理のライブラリ

形式 |
```
cc [ flag . . . ] file . . . -lcfgadm -ldevinfo -ldl [ library . . . ]
#include <config_admin.h>
```

機能説明 | このライブラリに含まれるインタフェースは、構成管理のためのサービスを提供します。

インタフェース | 共有オブジェクト libcfgadm.so.1 は、以下に示す共通インターフェースを提供します。

共有オブジェクトのインタフェースについては、intro(3) を参照してください。

config_ap_id_cmp	config_change_state
config_help	config_list
config_list_ext	config_private_func
config_stat	config_strerror
config_test	config_unload_libs

ファイル |

/usr/lib/libcfgadm.so.1	共有オブジェクトファイル
/usr/lib/64/libcfgadm.so.1	64 ビット共有オブジェクトファイル

属性 | 以下の属性については、attributes(5) を参照してください。

属性の種類	属性の値
使用条件	SUNWcsl (32-bit) SUNWcslx (64-bit)
MT レベル	Mt-Safe

関連項目 | pvs(1), cfgadm(1M), intro(3), config_admin(3CFGADM), attributes(5)