BEA Tuxedo®

BEA Tuxedo FML リファレンス

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA Tuxedo FML リファレンス

Part Number	Date	Software Version
N/A	January 2003	BEA Tuxedo リリース 8.1

目次

このマニュアルについて

このマニュアルでは、BEA Tuxedo ATMI 環境で使用されるフィールド操作言語 (FML) 関数について説明します。FML は、フィールド化バッファを定義し操作するための C 言語関数のセットです。リファレンス・ページは関数名のアルファベット順になっています。

対象読者

このマニュアルは、次のような読者を対象としています。

- BEA Tuxedo 環境でアプリケーションをコンフィギュレーションおよび管理するシステム管理者
- BEA Tuxedo 環境でアプリケーションをプログラミングするアプリケーション開発者

このマニュアルは、BEA Tuxedo プラットフォームおよび C 言語のプログラミングの知識があることを前提としています。

e-docs Web サイト

BEA 製品のマニュアルは BEA 社の Web サイト上で参照することができます。BEA ホーム・ページの [製品のドキュメント] をクリックするか、または http://edocs.beasys.co.jp/e-docs/index.html に直接アクセスしてください。

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。

このマニュアルの PDF 版は、e-docs Web サイトの BEA Tuxedo マニュアル・ページから入手できます。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo Documents ページの [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader をお持ちではない場合は、Adobe 社の Web サイト (http://www.adobe.co.jp/) から無償で入手できます。

関連情報

関連情報は、各リファレンス・ページの「関連項目」に一覧表示されています。

サポート情報

皆様の BEA Tuxedo マニュアルに対するフィードバックをお待ちしています。ご意見やご質問がありましたら、電子メールで docsupport-jp@bea.com までお送りください。お寄せいただきましたご意見は、BEA Tuxedo マニュアルの作成および改訂を担当する BEA 社のスタッフが直接検討いたします。

電子メール メッセージには、BEA Tuxedo 8.1 リリースのマニュアルを使用していることを明記してください。

BEA Tuxedo に関するご質問、または BEA Tuxedo のインストールや使用に際して問題が発生した場合は、http://www.bea.com の BEA WebSupport を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されている カスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

- お客様のお名前、電子メール・アドレス、電話番号、Fax 番号
- お客様の会社名と会社の住所
- ご使用のマシンの機種と認証コード
- ご使用の製品名とバージョン
- 問題の説明と関連するエラー・メッセージの内容

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

規則	項目
太字	用語集に定義されている用語を示します。
Ctrl + Tab	2 つ以上のキーを同時に押す操作を示します。
イタリック体	強調またはマニュアルのタイトルを示します。

等幅テキスト	コード・サンプル、コマンドとオプション、データ構造とメンバ、データ型、ディレクトリ、およびファイル名と拡張子を示します。また、キーボードから入力する文字も示します。
	例 :
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`等幅太字`	コード内の重要な単語を示します。 例 : `void **commit** ( )`
`等幅イタリック体`	コード内の変数を示します。 例 : `String *expr*`

規則	項目
大文字	デバイス名、環境変数、および論理演算子を示します。 例 : LPT1 SIGNON OR
{ }	構文の行で選択肢を示します。かっこは入力しません。
[ ]	構文の行で省略可能な項目を示します。かっこは入力しません。 例 : `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
|	構文の行で、相互に排他的な選択肢を分離します。記号は入力しません。
...	コマンド行で次のいずれかを意味します。 ■ コマンド行で同じ引数を繰り返し指定できること ■ 省略可能な引数が文で省略されていること ■ 追加のパラメータ、値、その他の情報を入力できること 省略符号は入力しません。 例 : `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
. . .	コード例または構文の行で、項目が省略されていることを示します。省略符号は入力しません。

セクション 3fml - FML 関数

表 1BEA Tuxedo ATMI FML **関数**

名前	機能説明
FML 関数の紹介	FML 関数について紹介する
CFadd、CFadd32(3fml)	フィールドの変換および追加
CFchg、CFchg32(3fml)	フィールドの変換および変更
CFfind、CFfind32(3fml)	検索、変換してポインタを返す
CFfindocc、CFfindocc32(3fml)	変換された値のオカレンスの検索
CFget、CFget32(3fml)	フィールドの取得および変換
CFgetalloc、CFgetalloc32(3fml)	フィールドの取得、領域の割り当て、および変換
F_error、F_error32(3fml)	最終エラーのエラー・メッセージを出力
F32to16、F16to32(3fml)	16 ビット FML バッファと 32 ビット FML バッファの間の変換
Fadd、Fadd32(3fml)	新しいフィールド・オカレンスの追加
Fadds、Fadds32(3fml)	FLD_STRING 型の値を変換し、バッファに追加
Falloc、Falloc32(3fml)	フィールド化バッファの割り当ておよび初期化
Fappend、Fappend32(3fml)	新しいフィールド・オカレンスの追加

表 1BEA Tuxedo ATMI FML **関数（続き）**

名前	機能説明
Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)	式をコンパイルし評価ツリーを返す
Fboolev、Fboolev32、Fvboolev、Fvboolev32(3fml)	評価ツリーからバッファ内容を評価
Fboolpr、Fboolpr32、Fvboolpr、Fvboolpr32(3fml)	解析された論理式を出力
Fchg、Fchg32(3fml)	フィールド・オカレンス値を変更
Fchgs、Fchgs32(3fml)	フィールド・オカレンスの変更－呼び出し側は、文字列を提供
Fchksum、Fchksum32(3fml)	フィールド化バッファのチェックサムの計算
Fcmp、Fcmp32(3fml)	2 つのフィールド化バッファの比較
Fconcat、Fconcat32(3fml)	ソース・バッファと宛先バッファの連結
Fcpy、Fcpy32(3fml)	ソース・バッファを宛先バッファにコピー
Fdel、Fdel32(3fml)	バッファからフィールド・オカレンスを削除
Fdelall、Fdelall32(3fml)	バッファからすべてのフィールド・オカレンスを削除
Fdelete、Fdelete32(3fml)	バッファからフィールドのリストを削除
Fextread、Fextread32(3fml)	フォーマットされた出力からフィールド化バッファを作成
Ffind、Ffind32(3fml)	バッファにあるフィールド・オカレンスの検索
Ffindlast、Ffindlast32(3fml)	バッファにあるフィールドの最後のオカレンスの値を検索
Ffindocc、Ffindocc32(3fml)	フィールド値のオカレンスの検索
Ffinds、Ffinds32(3fml)	文字列表現を指すポインタを返す
Ffloatev、Ffloatev32、Fvfloatev、Fvfloatev32(3fml)	式の値を double 型で返す

表 1BEA Tuxedo ATMI FML **関数（続き）**

名前	機能説明
Ffprint、Ffprint32(3fml)	フィールド化バッファを指定されたストリームに出力
Ffree、Ffree32(3fml)	フィールド化バッファ用に割り当てられた領域の解放
Fget、Fget32(3fml)	フィールド・オカレンスのコピーおよび長さの取得
Fgetalloc、Fgetalloc32(3fml)	領域の割り当て、およびフィールド・オカレンスのコピーの取得
Fgetlast、Fgetlast32(3fml)	最後のオカレンスのコピーの取得
Fgets、Fgets32(3fml)	値を文字列に変換
Fgetsa、Fgetsa32(3fml)	malloc() で領域を割り当て、変換された値を取得する
Fidnm_unload、Fidnm_unload32(3fml)	*id->nm* マッピング・テーブルから領域を取り戻す
Fidxused、Fidxused32(3fml)	使用されている領域の大きさを返す
Fielded、Fielded32(3fml)	バッファがフィールド化されている場合は、true を返す
Findex、Findex32(3fml)	フィールド化バッファのインデックスを行う
Finit、Finit32(3fml)	フィールド化バッファを初期化
Fjoin、Fjoin32(3fml)	ソース・バッファを宛先バッファに組み込む
Fldid、Fldid32(3fml)	フィールド名をフィールド識別子にマップする
Fldno、Fldno32(3fml)	フィールド識別子をフィールド番号にマップする
Fldtype、Fldtype32(3fml)	フィールド識別子をフィールド・タイプにマップする
Flen、Flen32(3fml)	バッファ中のフィールド・オカレンスの len を返す
Fmbpack32(3fml)	符号化名およびマルチバイト・データ情報を準備する

表 1BEA Tuxedo ATMI FML **関数（続き）**

名前	機能説明
Fmbunpack32(3fml)	符号化名およびマルチバイト・データ情報を抽出する
Fmkfldid、Fmkfldid32(3fml)	フィールド識別子を作成
Fmove、Fmove32(3fml)	フィールド化バッファを宛先バッファに移動
Fname、Fname32(3fml)	フィールド識別子をフィールド名にマップする
Fneeded、Fneeded32(3fml)	バッファに必要な大きさの計算
Fnext、Fnext32(3fml)	次のフィールド・オカレンスの取得
Fnmid_unload、Fnmid_unload32(3fml)	*nm->id* マッピング・テーブルから領域を取り戻す
Fnum、Fnum32(3fml)	バッファにあるすべてのオカレンスの数を返す
Foccur、Foccur32(3fml)	バッファにあるフィールド・オカレンスの数を返す
Fojoin、Fojoin32(3fml)	ソース・バッファを宛先バッファに組み込む
Fpres、Fpres32(3fml)	フィールド・オカレンスがバッファにある場合は true を返す
Fprint、Fprint32(3fml)	標準出力にバッファを出力する
Fproj、Fproj32(3fml)	バッファのプロジェクションを提供
Fprojcpy、Fprojcpy32(3fml)	バッファのプロジェクションおよびコピーを提供
Fread、Fread32(3fml)	フィールド化バッファの読み取り
Frealloc、Frealloc32(3fml)	フィールド化バッファを再割り当て
Frstrindex、Frstrindex32(3fml)	バッファにあるインデックスをリストアする
Fsizeof、Fsizeof32(3fml)	フィールド化バッファのサイズを返す
Fstrerror、Fstrerror32(3fml)	FML エラーのエラー・メッセージ文字列の獲得
Ftypcvt、Ftypcvt32(3fml)	フィールド・タイプを別のフィールド・タイプに変換

表 1BEA Tuxedo ATMI FML **関数（続き）**

名前	機能説明
Ftype、Ftype32(3fml)	フィールド・タイプを指すポインタを返す
Funindex、Funindex32(3fml)	フィールド化バッファのインデックスの破棄
Funused、Funused32(3fml)	フィールド化バッファの未使用分のバイト数を返す
Fupdate、Fupdate32(3fml)	ソースを持つ宛先バッファの更新
Fused、Fused32(3fml)	フィールド化バッファで使用しているバイト数を返す
Fvall、Fvall32(3fml)	フィールド・オカレンスを long 型で返す
Fvals、Fvals32(3fml)	フィールド・オカレンスの文字列の値を返す
Fvftos、Fvftos32(3fml)	フィールド化バッファから C 構造体にコピー
Fvneeded、Fvneeded32(3fml)	VIEW バッファに必要な大きさの計算
Fvnull、Fvnull32(3fml)	構造体の要素が NULL かどうかのチェック
Fvopt、Fvopt32(3fml)	マッピング・エントリのフラグ・オプションの変更
Fvrefresh、Fvrefresh32(3fml)	C 構造体からフィールド化バッファにコピーする
Fvselinit、Fvselinit32(3fml)	構造体の要素を NULL 値で初期化する
Fvsinit、Fvsinit32(3fml)	C 構造体を NULL 値で初期化する
Fvstof、Fvstof32(3fml)	C 構造体からフィールド化バッファにコピーする
Fvstot、Fvttos(3fml)	C 構造体からターゲットのレコード・タイプに変換、およびその逆の変換
Fwrite、Fwrite32(3fml)	フィールド化バッファを書き込む
tpconvfmb32(3fml)	ソースの符号化からターゲットの符号化にマルチバイト文字を変換する

FML 関数の紹介

形式

```
“#include <fml.h>”
“#include <fml32.h>”
```

機能説明

FML は、フィールド化バッファと呼ばれる記録構造を定義、操作する一連の C 言語関数で、フィールド化バッファにはフィールドと呼ばれる属性と値の対が含まれます。属性はフィールドの識別子であり、対応する値はフィールドのデータ内容を表します。

フィールド化バッファを使用すると、関連したフィールドのセットを名前でアクセスできるので、協調動作するプロセス間でパラメータ化されたデータを受け渡す場合に便利です。他のプロセスとコミュニケートする必要のあるプログラムは、FML ソフトウェアを使用し、フィールドを含む構造体を意識せずにフィールドへのアクセスが可能です。

FML はまた、`VIEW` と呼ばれる、フィールド化バッファを C 構造体にマップする (その逆も可能) 機能も提供します。`VIEW` はフィールド化バッファではなく構造体間で、大量のデータのやり取りを行います。データを構造体に転送し操作すると、アプリケーションはより高速に実行します。`VIEW` を使用すると、フィールド化バッファのデータ独立性と従来のレコード構造の効率性、簡便性を共に享受することができます。

FML16 と FML32

FML には 2 つのサイズがあります。オリジナルの FML インターフェイスは、フィールド長に 16 ビットの値を使用して、フィールドを識別する情報を格納します。このマニュアル・ページでは、これらを FML16 と呼びます。FML16 では、一意のフィールド数は 8191、個々のフィールド長は最大 64K バイト、フィールド化バッファの総容量は 64K に制限されます。このインターフェイスの定義、型、および関数のプロトタイプは `fml.h` に定義され、FML16 インターフェイスを使用するアプリケーション・プログラムは、このファイルをインクルードする必要があります。各関数は、`-lfml` にあります。FML32 は、フィールド長と識別子に 32 ビットの値を使用します。FML32 では、約 3000 万個のフィールドを含むことができ、フィールド長およびバッファ長は約 20 GB まで使用できます。FML32 の定義、型および関数プロトタイプは、`fml32.h` に定義されます。各関数は `-lfml32` にありま

す。FML32 のすべての定義、型、関数名には、接尾辞 “32” が付きます (MAXFBLEN32、Fchg32 など)。また、環境変数にも接頭辞 “32” が付きます (FLDTBLDIR32、FIELDTBLS32、VIEWFILES32、VIEWDIR32 など)。

FML バッファ

フィールド化バッファは、フィールド識別子とフィールド値の対 (long, short など固定長フィールドの場合)、またはフィールド識別子、フィールド長、フィールド値の三揃い (可変長フィールドの場合) で構成されます。

フィールド識別子は、フィールド化バッファ内の個々のデータ項目に対するタグで、フィールド番号とフィールドのデータ型で構成されます。FML16 では、フィールド識別子は 1 から 8191 の範囲で、その型定義は FLDID です。FML32 では、1 から 33,554,431 の範囲で、型定義は FLDID32 です。BEA Tuxedo ATMI システムは、フィールドに関する以下の規則に従っています。

FML16 **のフィールド番号**		FML32 **のフィールド番号**	
予約済み	**使用可能**	**予約済み**	**使用可能**
1 ～ 100	101 ～ 8191	1 ～ 10,000、30,000,001 ～ 33,554,431	10,001 ～ 30,000,000

BEA Tuxedo ATMI システムでは強制的に使用不可にはなっていませんが、アプリケーションが予約番号を使用しないようにしてください。

フィールドの型は、標準 C 言語の型のいずれでもかまいません (short、long、float、double、char)。ほかに、string (NULL 文字で終了する文字列)、carray(文字配列)、mbstring (マルチバイト文字配列 —BEA リリース 8.1 またはそれ以降で使用可能)、ptr (バッファへのポインタ)、fml32 (埋め込み型 FML32 バッファ)、および view32 (埋め込み型 VIEW32 バッファ) がサポートされています (ptr 型、fml32 型、および view32 型は、FML32 インターフェイスでのみサポート)。fml.h では、サポートされているフィールド型は、FLD_SHORT、FLD_LONG、FLD_FLOAT、FLD_DOUBLE、FLD_CHAR、FLD_STRING、および FLD_CARRAY として定義されています。fml32.h では、サポートされているフィールド型は、FLD_SHORT、FLD_LONG、FLD_FLOAT、FLD_DOUBLE、FLD_CHAR、FLD_STRING、FLD_CARRAY、FLD_MBSTRING、FLD_PTR、FLD_FML32、および FLD_VIEW32 として定義されています。

FML16 では、フィールド化バッファのポインタは、FBFR * 型、フィールド長は FLDLEN 型、フィールドのオカレンス数は FLDOCC 型です。FML32 では、フィールド化バッファのポインタは FBFR32 * 型、フィールド長は FLDLEN32 型、フィールドのオカレンス数は FLDOCC32 型です。

フィールドは、FML インターフェイスのフィールド識別子により参照されます。ただし通常は、アプリケーション・プログラマにはフィールド名を覚える方が簡単です。フィールド名をフィールド識別子にマップするには、2 つの方法があります。

フィールド名と識別子のマッピングは、field_tables(5) に記述されるフィールド・テーブル・ファイルにより、実行時に FML プログラムで行うことができます。FML16 インターフェイスでは、環境変数 FLDTBLDIR を使用し、フィールド・テーブルを検索するディレクトリのリストを指定します。また FIELDTBLS により、使用するテーブル・ディレクトリ内のファイルのリストを指定します。FML32 インターフェイスでは、FLDTBLDIR32 と FIELDTBLS32 を使用します。アプリケーション・プログラムでは、FML 関数 Fldid() と Fldid32() が、実行時にフィールド名をフィールド識別子に変換します。また Fname() と Fname32() は、フィールド識別子をフィールド名に変換します。

コンパイル時のフィールド名と識別子のマッピングは、フィールド名のマクロ定義を含むフィールド・ヘッダ・ファイルを使用して行われます。mkfldhdr() と mkfldhdr32() は、フィールド・テーブル・ファイルからヘッダ・ファイルを作成するための関数です (mkfldhdr、mkfldhdr32(1) を参照)。これらのヘッダ・ファイルは C プログラムの #include で指定されるファイルで、コンパイル時にフィールド名をフィールド識別子にマップすることができます。

フィールド化バッファ内のフィールドは、複数回出現する場合があります。FML 関数の多くが、検索または変更の対象にするフィールド・オカレンスを指定する引数を取ります。フィールドが複数回出現する場合、オカレンスは、最初のオカレンスを 0 として、順次番号付けされます。すべてのオカレンスが集まると、論理的なシーケンスを構成しますが、オカレンス番号に対応するオーバヘッドはありません (つまり、オカレンス番号はフィールド化バッファに格納されません)。フィールド・オカレンスを追加すると、そのオカレンスは、オカレンスの集まりの最後に追加され、一番高い番号より 1 つ高い番号のオカレンスとして参照されます。一番高い番号のオカレンス以

外のオカレンスを削除すると、削除されたオカレンスより高い番号のオカレンスはすべて、1 つだけ下位にシフトします (たとえば、オカレンス 6 がオカレンス 5 になり、オカレンス 5 がオカレンス 4 になります)。

フィールド化バッファに多数のフィールドが含まれる場合、内部インデックスを使用すると、FML でのアクセスが促進されます。通常、ユーザはこのインデックスの存在を意識することはありません。フィールド化バッファをディスクに格納したり、プロセス間またはコンピュータ間で転送する場合は、まず `Funindex()` または `Funindex32()` を使用してこのインデックスを削除すると、ディスク領域や転送時間を節約できます。後でインデックスが必要になれば、`Findex()` または `Findex32()` を使用して再び作成します。

FML16 から FML32 への変換

適切に書かれた既存の FML16 アプリケーションは、容易に FML32 インターフェイスを使用するように変更できます。FML 関数の呼び出しに使用するすべての変数には、適切な typedef (`FLDID`、`FLDLEN`、および `FLDOCC`) を使用する必要があります。FML 型付きバッファのための `tpalloc()` 呼び出しには、FML ではなく `FMLTYPE` 定義を使用します。アプリケーションのソース・コードに、`fml.h` の代わりに `fml32.h` を指定し、`fml1632.h` を組み込むことで 32 ビットの関数を使用できるようになります。`fml1632.h` には、すべての 16 ビットの型定義を 32 ビット版に変換したり、16 ビットの関数やマクロを 32 ビット版に変換するマクロが含まれています。

VIEWS

`VIEWS` は、フィールド操作言語の一部です。この機能を使用すると、フィールドと C 構造体のメンバとのマッピングの指定によって、フィールド化バッファと C 言語プログラムの C 構造体の間でデータを受け渡しすることができます。フィールド化バッファ情報を大量に操作する場合は、データを C 構造体内に転送するとパフォーマンスが向上します。`VIEWS` 関数でフィールド化バッファ内の情報をバッファ内のフィールドから抽出して C 構造体に入れ、C 構造体内でデータを操作した後、再度 `VIEWS` 関数を使用して、更新した値をフィールド化バッファに戻すことができます。

型付きバッファは、フィールド化バッファという FML の概念に基づいた ATMI 環境の機能の 1 つです。ATMI 環境と共に納入される 2 種類の標準バッファは、FML 型付きバッファと `VIEW` 型付きバッファです。このバッファがさらに異なる点は、`VIEW` バッファは FML フィールド化バッファとの関連性がまったくなくてもよいということです。また `FML32` と `VIEW32` のバッファ型が使用される点でも異なります。

VIEW 記述は、viewfile(5) で解説されるようにソース viewfile に作成、格納されます。VIEW 記述は、フィールド化バッファ内のフィールドを C 構造体内のメンバにマップします。ソース VIEW 記述は、viewc() または viewc32() でコンパイルされ、VIEW オブジェクト・ファイルが生成されます。これは、フィールド化バッファと C プログラムの C 構造体の間で転送されるデータのマッピングに使用されます (viewc、viewc32(1) を参照)。また、VIEW コンパイラはアプリケーション・プログラムにインクルードする C 言語のヘッダ・ファイルを作成し、VIEW 記述で記述される構造体を定義します。VIEW 逆アセンブラである viewdis() または viewdis32() は、オブジェクト VIEW 記述を読み取り可能な形式に変換する (つまり、ソース VIEW 記述に戻す) ためのものです。逆アセンブラの出力は、VIEW コンパイラに再入力できます (viewdis、viewdis32(1) を参照)。

オブジェクト・ファイルは実行時に使用され、VIEWFILES と VIEWDIR 環境変数を用いて VIEW 構造体を操作します。VIEWFILES には、アプリケーションのためのコンマで区切ったオブジェクト viewfile のリストが含まれている必要があります。絶対パス名で指定されたファイルはそのまま使用されます。相対パス名でリストされたファイルは VIEWDIR 変数により指定されたディレクトリのリストにあるかどうかが検索されます (下記参照)。VIEWDIR は、相対ファイル名を持つ VIEW オブジェクト・ファイルの検索に使用する、コンマで区切られたディレクトリ・リストを指定します。VIEW32 構造では、VIEWFILES32 および VIEWDIR32 が使用されます。

大半の FML 関数によってサポートされているデータ型以外に、VIEWS はソース VIEW 記述内で int 型を間接的にサポートします。VIEW 記述がコンパイルされると、VIEW コンパイラは、使用中のマシンの種類によって、すべての int 型を short 型か long 型に自動的に変換します。

VIEWS は、10 進数データ型もサポートしています。これは dec_t 型のフィールドとして定義され、パック 10 進数の値は総バイト数と小数点以下のバイト数として表されます。このようなフィールドは、FML では直接サポートされていませんが、このフィールドと、FML でサポートされているほかのフィールドとの変換は自動的に行われます。パック 10 進数は、COBOL 環境では 2 桁の 10 進数が 1 バイトにパックされ、下位バイトに符号が格納されます。C 環境では、データ型は 10 進指数、符号、桁、およびパック 10 進数の値を含む dec_t 型定義で定義されます。

FML バッファは、Fvftos() または Fvftos32() を使用して、VIEW に変換できます。view は、Fvstof() または Fvstof32() を使用して、フィールド化バッファに変換できます。フィールド化バッファと構造体の間でデータ転送を行うと、ソース・データは自動的に宛て先データの型に変換されます。フィールドの複数オカレンスがサポートされています。これは構造体中の配列として扱われます。NULL 値は構造体の空のメンバを表すのに使用し、viewfile の構造体の各メンバに対して指定できます。メンバに NULL 値を指定しないと、省略時の NULL 値が使用されます。また C 構造体のメンバとフィールド化バッファ内のフィールドの間にマッピングが存在していても、それらの間でのデータの転送を禁止することができます。

VIEW は、目標レコード形式からの変換、または目標レコード形式への変換もできます。デフォルトの形式は、IBM System/370 の COBOL レコードです。Fvstot() 関数がバイトの位、浮動小数点や 10 進数形式、および文字セットの変換を処理し (ASCII から EBCDIC へ)、Fvttos() が本来の形式に戻します。これらの関数には、32 ビット・バージョンもあります。Fcodeset() 関数は、代替 ASCII/EBCDIC トランザクション・テーブルの指定に使用できます。

エラー処理

ほとんどの FML 関数は、1 つまたは複数の戻り値を返します。エラーの条件は、エラーの他には考えられない戻り値で示されます。通常、エラーであれば -1、不正なフィールド識別子 (BADFLDID) またはアドレスであれば 0 です。またエラー・タイプが、外部整数 Ferror (FML16 の場合)、あるいは Ferror32 (FML32 の場合) として得られます。Ferror と Ferror32 は、正常呼び出しではクリアされないので、エラーが示された後でのみテストします。

F_error() と F_error32() 関数は、標準エラー出力にメッセージを出力します。この関数は、パラメータを 1 つ (文字列) 取り、コロンと空白を付加してその引数文字列を出力します。次に、エラー・メッセージとその後に続く復帰改行文字を出力します。表示されるエラー・メッセージは、エラー発生時に設定された Ferror と Ferror32 内の現在のエラー番号に対して定義されているメッセージです。

メッセージ・カタログからエラー・メッセージのテキストを検索するには、Fstrerror(3fml) を使用することができます。これらは、userlog(3c) への引数として使用できるポインタを返します。

FML 関数で生成されるエラー・コードは、それぞれの FML マニュアル・ページに記載されています。

関連項目 CFadd、CFadd32(3fml)、CFchg、CFchg32(3fml)、CFfind、CFfind32(3fml)、CFfindocc、CFfindocc32(3fml)、CFget、CFget32(3fml)、CFgetalloc、CFgetalloc32(3fml)、F_error、F_error32(3fml)、Fadd、Fadd32(3fml)、Fadds、Fadds32(3fml)、Falloc、Falloc32(3fml)、Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)、Fboolev、Fboolev32、Fvboolev、Fvboolev32(3fml)、Fboolpr、Fboolpr32、Fvboolpr、Fvboolpr32(3fml)、Fchg、Fchg32(3fml)、Fchgs、Fchgs32(3fml)、Fchksum、Fchksum32(3fml)、Fcmp、Fcmp32(3fml)、Fconcat、Fconcat32(3fml)、Fcpy、Fcpy32(3fml)、Fdel、Fdel32(3fml)、Fdelall、Fdelall32(3fml)、Fdelete、Fdelete32(3fml)、Fextread、Fextread32(3fml)、Ffind、Ffind32(3fml)、Ffindlast、Ffindlast32(3fml)、Ffindocc、Ffindocc32(3fml)、Ffinds、Ffinds32(3fml)、Ffloatev、Ffloatev32、Fvfloatev、Fvfloatev32(3fml)、Ffprint、Ffprint32(3fml)、Ffree、Ffree32(3fml)、Fget、Fget32(3fml)、Fgetalloc、Fgetalloc32(3fml)、Fgetlast、Fgetlast32(3fml)、Fgets、Fgets32(3fml)、Fgetsa、Fgetsa32(3fml)、Fidnm_unload、Fidnm_unload32(3fml)、Fidxused、Fidxused32(3fml)、Fielded、Fielded32(3fml)、Findex、Findex32(3fml)、Finit、Finit32(3fml)、Fjoin、Fjoin32(3fml)、Fldid、Fldid32(3fml)、Fldno、Fldno32(3fml)、Fldtype、Fldtype32(3fml)、Flen、Flen32(3fml)、Fmkfldid、Fmkfldid32(3fml)、Fmove、Fmove32(3fml)、Fname、Fname32(3fml)、Fneeded、Fneeded32(3fml)、Fnext、Fnext32(3fml)、Fnmid_unload、Fnmid_unload32(3fml)、Fnum、Fnum32(3fml)、Foccur、Foccur32(3fml)、Fojoin、Fojoin32(3fml)、Fpres、Fpres32(3fml)、Fprint、Fprint32(3fml)、Fproj、Fproj32(3fml)、Fprojcpy、Fprojcpy32(3fml)、Fread、Fread32(3fml)、Frealloc、Frealloc32(3fml)、Frstrindex、Frstrindex32(3fml)、Fsizeof、Fsizeof32(3fml)、Fstrerror、Fstrerror32(3fml)、Ftypcvt、Ftypcvt32(3fml)、Ftype、Ftype32(3fml)、Funindex、Funindex32(3fml)、Funused、Funused32(3fml)、Fupdate、Fupdate32(3fml)、Fused、Fused32(3fml)、Fvall、Fvall32(3fml)、Fvals、Fvals32(3fml)、Fvftos、Fvftos32(3fml)、Fneeded、Fneeded32(3fml)、Fvnull、Fvnull32(3fml)、Fvopt、Fvopt32(3fml)、Fvselinit、Fvselinit32(3fml)、Fvsinit、Fvsinit32(3fml)、Fvstof、Fvstof32(3fml)、Fwrite、Fwrite32(3fml)、field_tables(5)、viewfile(5)

『『FML を使用した BEA Tuxedo アプリケーションのプログラミング』』

CFadd、CFadd32(3fml)

名前 CFadd()、CFadd32() ―フィールドの変換および追加

形式

```
#include <stdio.h>
#include "fml.h"
int CFadd(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len, int
type)
#include fml32.h>
int
CFadd32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32 len,
int type)
```

機能説明 CFadd() は、Fadd() に似た動きをしますが、CFadd は、フィールド化バッファにフィールドが追加される場合、*value* をユーザ指定の型から *fieldid* 型に変換します。*fbfr* は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。*value* は、追加される値を指すポインタです。*len* は、追加される値の長さです。len は、FLD_CARRAY 型の場合のみ必要です。*type* は、*value* 内のフィールドのデータ型です。

バッファにフィールドが追加される前に、データ項目の型は、ユーザが提供した型から *fieldid* で指定された型に変換されます。ソース・タイプが FLD_CARRY (任意の文字配列) である場合、*len* 引数は、配列の長さに設定される必要があります。その他の場合、長さは無視されます。C 言語が、12345L のような構成を許していないために、変換され、追加されるフィールドの値は、最初に、変数 *value* に入れられる必要があります。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFadd() または CFadd32() が使用されているとき、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFadd32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行している場合でも、CFadd() または CFadd32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、CFadd() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、*value* パラメータに NULL が指定された場合)。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目　「FML 関数の紹介」、Fadd、Fadd32(3fml)

CFchg、CFchg32(3fml)

名前 CFchg()、CFchg32() ―フィールドの変換および変更

形式

```
#include <stdio.h>
#include "fml.h"
int CFchg(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *value,
       FLDLEN len, int type)
#include "fml32.h"
int CFchg32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc,
       char *value,
FLDLEN32 len, int type)
```

機能説明 CFchg() は、Fchg() に似た働きをしますが、フィールドがフィールド化バッファにおいて変更されるために、最初に、ユーザ指定の *type* から *fieldid* 型に *value* を変換します。*fbfr* は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。*oc* はフィールドのオカレンス番号です。*value* は、新しい値を指すポインタです。*len* は、変更される値の長さです。len は、型が FLD_CARRAY の場合のみ必要です。type は、value のデータ・タイプです。

存在しないフィールド・オカレンスが指定された場合、必要とされる値が追加されるまで、存在しないオカレンスに NULL 値が追加されます (たとえば、バッファに存在しないフィールドのフィールド・オカレンスを 4 に変更すると、3 個の NULL 値の後に指定されたフィールド値が追加されます)。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFchg() または CFchg32() が使用されている場合に、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFchg32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、CFchg() または CFchg32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、CFchg() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、value パラメータに NULL が指定された場合)。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目 「FML 関数の紹介」、CFadd、CFadd32(3fml)、Fchg、Fchg32(3fml)

CFfind、CFfind32(3fml)

名前 CFfind()、CFfind32() ―検索、変換してポインタを返す

形式

```
#include <stdio.h>
#include "fml.h"
char * CFfind(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *len,
  int type)
#include "fml32.h"
char *
CFfind32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32 *len,
  int type)
```

機能説明 CFfind() は、バッファ内の指定されたフィールドを検索し、それを変換してからその変換した値を指すポインタを返します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。len は、出力時に使用されるもので変換された値の長さを指すポインタです。type は、ユーザが変換したいフィールドの変換後のデータ・タイプです。

Ffind() と同じように、関数が返すポインタは、読み取り専用です。変換された値は 1 つのプライベート・バッファに保持されているので、CFfind() が返すポインタの妥当性は、次のバッファ操作までしか保証されません (たとえその操作が非破壊的である場合でも)。これは、Ffins() が返す値と違う点です。つまり、バッファを次に変更するまで保証されます。Ffind() とは異なり、CFfind() は、呼び出し側が直接使用できるように変換した値を整列させます。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFfind() または CFfind32() が使用されている場合に、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFfind32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、CFfind() または CFfind32() を呼び出すことができます。

戻り値 上記の「形式」の項では、CFfind() の戻り値のデータ型は、char ポインタ (C の char **) として記述されています。実際、返されるポインタは、格納済みのフィールドの型と同じ型を持つオブジェクトを指しています。

この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、CFfind() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目　「FML 関数の紹介」、Ffind、Ffind32(3fml)

CFfindocc、CFfindocc32(3fml)

名前 CFfindocc()、CFfindocc32() ―変換された値のオカレンスの検索

形式

```
#include <stdio.h>
#include "fml.h"
FLDOCC
CFfindocc(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len, int
  type)
#include "fml32.h"
FLDOCC32
CFfindocc32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32
  len, int type)
```

機能説明 CFfindocc() は、Ffindocc() に似た働きをしますが、最初に、ユーザ指定の型 value を fieldid 型に変換します。CFfindocc() は、ユーザが提供する値、長さ、および型と一致するバッファにある指定されたフィールドのオカレンスを検索します。CFfindocc() は、最初に一致したフィールドのオカレンス番号を返します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。value は、検索される値を指すポインタです。型が FLD_CARRAY の場合、len は、入力値と比較する値の長さです。type は、value 内のフィールドのデータ型です。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFfindocc() または CFfindocc32() が使用されている場合に、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFfindocc32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、CFfindocc() または CFfindocc32() を呼び出すことができます。

戻り値 フィールド値が見つからない場合、または他のエラーが見つかった場合、-1 を返し、CFfindocc() は Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、CFfindocc() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、value パラメータに NULL が指定された場合)。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目 「FML 関数の紹介」、Ffindocc、Ffindocc32(3fml)

CFget、CFget32(3fml)

名前 CFget()、CFget32() ―フィールドの取得および変換

形式

```
#include <stdio.h>
#include "fml.h"
int
CFget(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *buf, FLDLEN *len,
 int type)
#include "fml32.h"
int
CFget32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *buf,
 FLDLEN32 *len, int type)
```

機能説明 CFget() は、Fget() に似た変換関数です。主な違いとして、CFget() は、ユーザが提供したバッファに変換した値を複写します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。buf は、プライベート・データ領域を指すポインタです。入力時において len は、プライベート・データ領域の長さを指すポインタです。戻り時において len は、返された値の長さを指すポインタです。入力時において len パラメータが NULL の場合は、フィールド値を入れるためのバッファが十分大きいために、値の長さが返されないものと想定されます。buf パラメータが NULL の場合、フィールド値は返されません。type は、ユーザが変換したい戻り値の変換後のデータ・タイプです。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFget() または CFget32() が使用されているとき、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFget32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、CFget() または CFget32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、CFget() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FNOSPACE]
"no space in fielded buffer"
len で指定されたデータ領域の大きさは、フィールド値を保持できるほど大きくありません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目　「FML 関数の紹介」、Fget、Fget32(3fml)

CFgetalloc、CFgetalloc32(3fml)

名前 CFgetalloc()、CFgetalloc32() ―フィールドの取得、領域の割り当て、変換

形式

```
#include <stdio.h>
#include "fml.h"
char *
CFgetalloc(FBFR *fbfr, FLDID fieldid, FLDOCC oc, int type, FLDLEN
 *extralen)
#include "fml32.h"
char *
CFgetalloc32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, int type,
 FLDLEN32 *extralen)
```

機能説明 CFgetalloc() は、バッファから指定されたフィールドを獲得し、領域を割り当てます。次に、そのフィールドをユーザ指定のタイプに変換し、その位置を指すポインタを返します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。type は、ユーザが変換したいフィールドの変換後のデータ・タイプです。呼び出し時において extralen は、値を受け取るために割り当てられる追加領域の長さを指すポインタです。戻り時において extralen は、実際に使用された領域の大きさを指すポインタです。extralen が NULL の場合、割り当てる追加領域はなく、実際の長さは返されません。ユーザには、(変換された) 戻り値を解放する責任があります。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。CFgetalloc() または CFgetalloc32() が使用されている場合に、これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

CFgetalloc32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、CFgetalloc() または CFgetalloc32() を呼び出すことができます。

戻り値　正常終了時には、CFgetalloc() は、変換された値を指すポインタを返します。エラー時には、NULL を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、CFgetalloc() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

関連項目　「FML 関数の紹介」、Fgetalloc、Fgetalloc32(3fml)

F_error、F_error32(3fml)

名前 F_error()、F_error32() ―最終エラーのエラー・メッセージを出力

形式

```
#include stdio.h>
#include "fml.h"
extern int Ferror;
void
F_error(char *msg)
#include "fml32.h"
extern int Ferror32;
void
F_error32(char *msg)
```

機能説明 この関数 F_error()} は、UNIX システム・エラーを出力する perror() に似た働きをします。つまり、標準エラー出力 (ファイル記述子 2) にエラー・メッセージを作成し、システム・コールあるいはライブラリ関数呼び出し時に発生した最後のエラーを記述します。文字列引数 msg は、最初に出力されます。次にコロン、空白が出力され、メッセージおよび改行が出力されます。msg が NULL ポインタあるいは NULL 文字列を指している場合、コロンは出力されません。さまざまな利用形態を考慮して、この引数文字列には、エラーを起こしたプログラム名を入れます。エラー番号は、外部変数 Ferror から獲得されます。Ferror は、エラーが起こったときに設定されますが、エラーを起こさない呼び出しが行われた場合はクリアされません。MS-DOS および OS/2 環境では、Ferror は、FMLerror に再定義されます。

エラー・メッセージをすばやく出力するために、F_error() は、他の FML 関数がエラーを返すときに呼び出されます。エラー・メッセージが FEUNIX の場合、Uunix_err() が呼び出されます。

F_error32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、F_error() または F_error32() を呼び出すことができます。

戻り値 F_error() は、void 宣言なので戻り値はありません。

関連項目 「FML 関数の紹介」

UNIX システムのリファレンス・ページの perror(3)、Uunix_err(3)

F32to16、F16to32(3fml)

名前 F32to16()、F16to32() — 16 ビット FML バッファと 32 ビット FML バッファの間の変換

形式

```
#include <stdio.h>
#include "fml.h"
#include "fml32.h"
int
F32to16(FBFR *dest, FBFR32 *src)
int
F16to32(FBFR32 *dest, FBFR *src)
```

機能説明 F32to16() は、32 ビットの FML バッファを 16 ビットの FML バッファに変換します。これは、フィールド対フィールドのバッファの変換を行い、フィールド化バッファのインデックスを作成することによって行ないます。FLDID32 から FLDID を生成し、フィールド値 (文字列、carray、および mbstring フィールドの長さ) をコピーすることによって、フィールドを変換します。dest は、宛先バッファのポインタです。src は、もとのフィールド化バッファを指すポインタです。ソース・バッファは変更されません。

これらの関数は、領域の不足のために異常終了する場合があります。処理を完了するために十分な追加のメモリを割り当てた後で、これらの関数を再び呼び出すことができます。

F16to32() は、16 ビットの FML バッファを 32 ビットの FML バッファに変換します。この関数は、fml32 ライブラリまたは共有オブジェクトの中にあり、エラー時には Ferror32 を設定します。

F32to16() は、FML ライブラリまたは共有オブジェクトの中にあり、エラー時には Ferror を設定します。これらの関数を使用するためには、fml.h および fml32.h の両方をインクルードする必要があることに注意してください。同じファイルに、fml1632.h をインクルードする必要はありません。

F32to16() は、FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、FBADFLD を設定して異常終了します。これらのフィールド・タイプに対して F16to32() を実行しても、何の影響もありません。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、F32to16() または F16to32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、F32to16() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、変換先のフィールド化バッファにコピーされますが、バッファには十分な領域が残っていません。このエラーは、32 ビットの FML フィールドが長すぎて、16 ビットの FML フィールドに収まらない場合にも返されます。このエラーが返された場合は、変換先のバッファにフィールドはできません。

[FBADFLD]
"invalid field number or type"
このエラーは、F32to16() 関数でのみ発生します。ソース・バッファは、16 ビットの FML がサポートしている 8 つのフィールド・タイプ以外のフィールド識別子を持っているか、フィールド番号が 8191 を超えています。

関連項目 「FML 関数の紹介」

Fadd、Fadd32(3fml)

名前 Fadd()、Fadd32() ―新しいフィールド・オカレンスの追加

形式

```
#include stdio.h>
#include "fml.h"
int Fadd(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len)
#include "fml32.h"
int Fadd32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32
len)
```

機能説明 Fadd() は、指定されたフィールド値を指定されたバッファに追加します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。value は、新しい値を指すポインタです。したがって、ポインタの型は、追加される値と同じフィールド識別子の型である必要があります。len は、追加される値の長さです。len は、FLD_CARRAY 型または FLD_MBSTRING 型の場合のみ必要です。

追加する値は、value パラメータで指定する場所にあります。既に 1 つまたは複数のフィールド・オカレンスがある場合、新しいフィールド・オカレンスとして値が追加され、現在指定されているオカレンスの最大値より 1 つ大きいオカレンス番号が割り当てられます (指定されたオカレンスを追加するには、Fchg() を使用する必要があります)。

上記の「形式」の項では、Fadd() の引数 value のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、これは、単に、Fadd() に渡せる値のある特定の型を記述しています。実際、引数 *value* の型は、追加されているフィールドのフィールド化バッファ表現と同じ型のオブジェクトのポインタである必要があります。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、*value* の型は、long のポインタ (C の long *) である必要があります。同様に、FLD_SHORT 型でバッファに格納される場合、*value* は、short 型のポインタ (C の short *) である必要があります。大事なことは、Fadd() は、*value* が指すオブジェクトが、追加される格納済みのフィールドの型と同じ型を持っていることを想定しているということです。

FLD_PTR 型の値の場合、Fadd32() はポインタの値を格納します。FLD_PTR フィールドで指定されるバッファは、tpalloc() を呼び出して割り当てる必要があります。FLD_FML32 型の値の場合、Fadd32() はインデックスを除いた FLD_FML32 フィールド全体の値を格納します。FLD_VIEW32 型の値の場合、Fadd() は FVIEWFLD 型の構造体へのポインタを格納します。これには、*vflags* (現在未使用で 0 に設定された flags フィールド)、*vname* (VIEW 名を含む文字配列)、および *data* (C 構造体として格納される VIEW データへのポインタ) が含まれます。アプリケーションは、*vname* と *data* を Fadd32() に提供します。

FLD_MBSTRING 型の値の場合、値は Fmbpack32() 関数のパッキングされた出力引数であり、引数 *len* は Fmbpack32() の出力引数 *size* の値の長さです。

FLD_CARRAY 型の値の場合、引数 *len* は値の長さです。FLD_CARRAY または FLD_MBSTRING 以外のすべての型では、*value* が参照するオブジェクトの長さは、その型から導かれ (たとえば、FLD_FLOAT の場合、長さは sizeof(float) になる)、*len* の内容は無視されます。

Fadd32 は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fadd() または Fadd32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fadd() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、Fadd() の value パラメータに NULL を指定した場合)。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファに追加されますが、バッファには十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド番号は無効です。

関連項目 「FML 関数の紹介」、CFadd、CFadd32(3fml)、Fadds、Fadds32(3fml)、Fchg、Fchg32(3fml)

Fadds、Fadds32(3fml)

名前 Fadds()、Fadds32() ― FLD_STRING 型の値を変換し、バッファに追加

形式

```
#include <stdio.h>
#include "fml.h"
int
Fadds(FBFR *fbfr, FLDID fieldid, char *value)
#include "fml32.h"
int
Fadds32(FBFR32 *fbfr, FLDID32 fieldid, char *value)
```

機能説明 Fadds() は、ユーザ型 FLD_STRING からフィールド型 (*fieldid*) への変換を処理し、それをフィールド化バッファに追加するために提供されています。*fbfr* は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。value は、追加される値を指すポインタです。

この関数は、type (FLD_STRING) を提供し、len が 0 である CFadd() を呼び出します。

Fadds32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fadds() または Fadds32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fadds() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファに追加されますが、バッファには十分な領域が残っていません。

[FTYPERR]
"invalid field type"
指定されたフィールド型は無効です。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、Fadds() の value パラメータに NULL を指定した場合)。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、CFchg、CFchg32(3fml)、CFfind、CFfind32(3fml)、CFget、CFget32(3fml)、Falloc、Falloc32(3fml)、Fchgs、Fchgs32(3fml)、Ffinds、Ffinds32(3fml)、Fgets、Fgets32(3fml)、Fgetsa、Fgetsa32(3fml)

Falloc、Falloc32(3fml)

名前 Falloc()、Falloc32() ―フィールド化バッファの割り当ておよび初期化

形式

```
#include <stdio.h>
#include "fml.h"
FBFR *
Falloc(FLDOCC F, FLDLEN V)
#include "fml32.h"
FBFR32 *
Falloc32(FLDOCC32 F, FLDLEN32 V)
```

機能説明 Falloc() は、フィールド化バッファ用に、malloc() を使用して領域を動的に割り当て、Finit() を呼び出してそれを初期化します。パラメータには、バッファに格納されているすべてのフィールドについて、フィールド数 (F) および値の領域のバイト数 (V) があります。

Falloc32() は、より多くのフィールドを持つ大きなバッファのために使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Falloc() または Falloc32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Falloc() は異常終了し、Ferror を次の値に設定します。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、フィールド数が 0 より小さい、V が 0、または合計の大きさが 65534 より大きい)。

関連項目　「FML 関数の紹介」、Ffree、Ffree32(3fml)、Fielded、Fielded32(3fml)、Finit、Finit32(3fml)、Fneeded、Fneeded32(3fml)、Frealloc、Frealloc32(3fml)、Fsizeof、Fsizeof32(3fml)、Funused、Funused32(3fml)

UNIX システムのリファレンス・ページの malloc(3)

Fappend、Fappend32(3fml)

名前 Fappend()、Fappend32() ―新しいフィールド・オカレンスの追加

形式

```
#include <stdio.h>
#include "fml.h"
int
Fappend(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len)
#include "fml32.h"
int
Fappend32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32 len)
```

機能説明 Fappend() は、指定されたフィールド値を指定されたバッファの終わりに追加します。Fappend() は、汎用目的の FML アクセスに必要な内部構造体および順序付けを保持しない、大規模なバッファを作成する際に役に立ちます。この最適化の影響として、Fappend() への呼び出しの後には、Fappend() への追加の呼び出し、FML インデックス・ルーチン Findex() および Funindex() への呼び出し、または Free()、Fused()、Funused()、Fsizeof() への呼び出ししか行なえません。Findex() または Funindex() を呼び出す前に他の FML ルーチンへの呼び出しを行うと、Ferror が FNOTFLD に設定されてエラーが発生します。

fbfr は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。value は、新しい値を指すポインタです。したがって、ポインタの型は、追加される値と同じフィールド識別子の型である必要があります。len は、追加される値の長さです。len は、FLD_CARRAY 型または FLD_MBSTRING 型の場合のみ必要です。

追加する値は、value パラメータで指される場所にあります。既に 1 つまたは複数のフィールド・オカレンスがある場合、新しいフィールド・オカレンスとして値が追加され、現在指定されているオカレンスの最大値より 1 つ大きいオカレンス番号が割り当てられます (指定されたオカレンスを追加するには、Fchg() を使用する必要があります)。

上記の「形式」の項では、Fappend() の引数 value のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、これは、単に、Fappend() に渡せる値の 1 つの特定の型を記述しています。実際、引数 value の型は、追加されているフィールドのフィールド化バッファ表現と同

じ型のオブジェクトへのポインタにします。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、value の型は、long のポインタ (C の long *) である必要があります。同様に、FLD_SHORT 型でバッファに格納される場合、value は、short 型のポインタ (C の short *) である必要があります。重要なことは、Fappend() は、value が指すオブジェクトが、追加される格納済みのフィールドと同じ型を持っていると想定しているということです。

FLD_MBSTRING 型の値の場合、値は Fmbpack32() 関数のパッキングされた出力引数であり、引数 *len* は Fmbpack32() の出力引数 *size* の値の長さです。

FLD_CARRAY 型の値の場合、引数 *len* は値の長さです。FLD_CARRAY または FLD_MBSTRING 以外のすべての型では、*value* が参照するオブジェクトの長さは、その型から導かれ (たとえば、FLD_FLOAT の場合、長さは、sizeof(float) になる)、*len* の内容は無視されます。

Fappend32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fappend() または Fappend32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fappend() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、Fappend() の value パラメータに NULL を指定した場合)。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファに追加されますが、バッファには十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド番号は無効です。

関連項目 「FML 関数の紹介」、Fadd、Fadd32(3fml)、Ffree、Ffree32(3fml)、Findex、Findex32(3fml)、Fsizeof、Fsizeof32(3fml)、Funindex、Funindex32(3fml)、Funused、Funused32(3fml)、Fused、Fused32(3fml)

Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)

名前 Fboolco()、Fboolco32()、Fvboolco()、Fvboolco32() 一式をコンパイルし評価ツリーを返す

形式

```
#include <stdio.h>
#include "fml.h"
char *
Fboolco(char *expression)
char *
Fvboolco(char *expression, char *viewname)
#include "fml32.h"
char *
Fboolco32(char *expression)
char *
Fvboolco32(char *expression, char *viewname)
```

機能説明 Fboolco() は、expression が指す論理表現をコンパイルし、評価ツリーのポインタを返します。認識される表現は、C で認識される表現に近いものです。文法の説明は、『FML を使用した BEA Tuxedo アプリケーションのプログラミング』に記載されています。

Fboolco() が作成する評価ツリーは、下記の「関連項目」の項でリストされている他の論理関数に使用されます。したがって、式を再びコンパイルする必要はありません。

Fboolco32() は 32 ビット FML で使用されます。

Fvboolco() および Fvboolco32() は、同じ VIEW 機能を提供します。viewname パラメータは、フィールド・オフセットを取り出す VIEW を示します。

FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると、この関数は異常終了します。これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

Warkstation プラットフォームでは、これらの関数はサポートされていません。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、ここで記述する関数、Fboolco()、Fboolco32()、Fvboolco()、または Fvboolco32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fboolco() は異常終了し、Ferror を次の値に設定します。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FSYNTAX]
"bad syntax in Boolean expression"
Fboolco() は、認識できないフィールド名以外に、論理表現に文法エラーを見つけました。

[FBADNAME]
"unknown field name"
フィールド・テーブルまたはビュー・ファイルにないフィールド名が指定されました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効でした (たとえば、expression が NULL)。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに viewname が見つかりません。

[FVFOPEN]
"cannot find or open viewfile"
viewname 検索中にプログラムで VIEWDIR または VIEWFILES で指定したファイルの 1 つが見つかりませんでした。

[EUNIX]
"operating system error"
viewname 検索中にプログラムは VIEWDIR または VIEWFILES で指定したファイルの 1 つを読み込み用に開けませんでした。

[FVFSYNTAX]
"bad viewfile"
viewname 検索中に VIEWDIR または VIEWFILES で指定したファイルの 1 つが破壊されていたか、VIEW ファイルではありませんでした。

[FMALLOC]
"malloc failed"
viewname 検索中に malloc() が VIEW 情報を格納するための領域の割り当てに失敗しました。

[FEBADOP]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、および FLD_VIEW32 など)。

使用例

```
#include "stdio.h"
#include "fml.h"
extern char *Fboolco(\|);
char *tree;
...
if((tree=Fboolco("FIRSTNAME %% 'J.*n' & SEX = 'M'")) == NULL)
 F_error("pgm_name");
```

この例では、'J' で始まり 'n' で終わる (たとえば、John、Jean、Jurgen など) FIRSTNAME フィールドがバッファにあるか、SEX フィールドが 'M' に等しいかを検査する論理表現をコンパイルします。

tree 配列の最初の文字は、最下位バイトを形成します。次の文字は、最上位バイトを形成します。それぞれ、全体の配列の長さをバイト数で指定する unsigned 型 (16 ビット) です。この値は、複写あるいは配列の操作に役立ちます。

関連項目 Fboolev、Fboolev32、Fvboolev、Fvboolev32(3fml)、Fboolpr、Fboolpr32、Fvboolpr、Fvboolpr32(3fml)、Fldid、Fldid32(3fml)

Fboolev、Fboolev32、Fvboolev、Fvboolev32(3fml)

名前 Fboolev()、Fboolev32()、Fvboolev()、Fvboolev32() ―評価ツリーに対するバッファの評価

形式

```
#include stdio.h>
#include "fml.h"
int
Fboolev(FBFR *fbfr, char *tree)
int
Fvboolev(char *cstruct, char *tree, char *viewname)
#include "fml32.h"
int
Fboolev32(FBFR32 *fbfr, char *tree)
int
Fvboolev32(char *cstruct, char *tree, char *viewname)
```

機能説明 Fboolev() は、フィールド化バッファを指すポインタ (fbfr)、Fboolco() が返す評価ツリーのポインタ (tree) を利用して、フィールド化バッファが指定された論理条件と一致する場合は、true (1) を返し、そうでない場合は false (0) を返します。この関数は、フィールド化バッファあるいは評価ツリーのいずれも変更しません。評価ツリーは、Fboolco() によって以前コンパイルされたものです。

Fboolev32() は 32 ビット FML で使用されます。

Fvboolev() および Fvboolev32() は、同じ VIEW 機能を提供します。viewname パラメータは、フィールド・オフセットを取り出す VIEW を示し、Fvboolco() または Fvboolco32() で指定された VIEW と同じにする必要があります。

Warkstation プラットフォームでは、これらの関数はサポートされていません。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、ここで記述する関数、Fboolev()、Fboolev32()、Fvboolev()、または Fvboolev32() を呼び出すことができます。

戻り値 Fboolev() は、バッファ内の表現と評価ツリーが一致している場合、1 を返します。表現が評価ツリーと不一致の場合、0 を返します。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fboolev() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
fbfr バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
fbfr バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効でした (たとえば、Fboolev の tree パラメータに NULL が指定されている場合)。

[FSYNTAX]
"bad syntax in Boolean expression"
認識できないフィールド名以外に、Fboolev() によって、論理表現に文法エラーが見つかりました。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに viewname が見つかりません。

[FVFOPEN]
"cannot find or open viewfile"
viewname 検索中にプログラムで VIEWDIR または VIEWFILES で指定したファイルの 1 つが見つかりませんでした。

[EUNIX]
"operating system error"
viewname 検索中にプログラムは VIEWDIR または VIEWFILES で指定したファイルの 1 つを読み込み用に開けませんでした。

[FVFSYNTAX]
"bad viewfile"
viewname 検索中に VIEWDIR または VIEWFILES で指定したファイルの 1 つが破壊されていたか、VIEW ファイルではありませんでした。

[FMALLOC]
"malloc failed"
viewname 検索中に malloc() が VIEW 情報を格納するための領域の割り当てに失敗しました。

使用例 Fboolco() の使用例でコンパイルした評価ツリーを使用します。

```
#include stdio.h>
#include "fml.h"
#include "fld.tbl.h"
FBFR *fbfr;
...
Fchg(fbfr,FIRSTNAME,0,"John",0);
Fchg(fbfr,SEX,0,"M",0);
if(Fboolev(fbfr,tree) > 0)
 fprintf(stderr,"Buffer selected\\\\n");
else
 fprintf(stderr,"Buffer not selected\\\\n");
```

Buffer selected を出力します。

関連項目 「FML 関数の紹介」、Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)、Fboolpr、Fboolpr32、Fvboolpr、Fvboolpr32(3fml)

Fboolpr、Fboolpr32、Fvboolpr、Fvboolpr32(3fml)

名前 Fboolpr()、Fboolpr32()、Fvboolpr()、Fvboolpr32() ―パースされた論理式を出力

形式

```
#include <stdio.h>
#include "fml.h"
void
Fboolpr(char *tree, FILE *iop)
int
Fvboolpr(char *tree, FILE *iop, char *viewname)
#include "fml32.h"
void
Fboolpr32(char *tree, FILE *iop)
int
Fvboolpr32(char *tree, FILE *iop, char *viewname)
```

機能説明 Fboolpr() は、コンパイル済みの表現を指定された出力ストリームに出力します。評価ツリー (tree) は、Fboolco() で以前作成されたものです。iop は、出力ストリームを指す FILE 型のポインタです。出力内容のすべてがかっこ内に入れられ、評価ツリーで示されるように解析されます。この関数はデバッグに役立ちます。

Fboolpr32() は 32 ビット FML で使用されます。

Fvboolpr() および Fvboolpr32() は、同じ VIEW 機能を提供します。viewname パラメータは、フィールド・オフセットを取り出す VIEW を示し、Fvboolco() または Fvboolco32() で指定された VIEW と同じにする必要があります。

Warkstation プラットフォームでは、これらの関数はサポートされていません。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、ここで記述する関数、Fboolpr()、Fboolpr32()、Fvboolpr()、または Fvboolpr32() を呼び出すことができます。

戻り値 Fboolpr() は、void を返すものとして宣言されているので、戻り値はありません。Fvboolpr() は、VIEW 名が無効な場合は -1 を返します。

エラー 次の条件の場合、Fvboolpr() は失敗し、Ferror を次の値に設定します。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに viewname が見つかりません。

[FVFOPEN]
"cannot find or open viewfile"
viewname 検索中にプログラムで VIEWDIR または VIEWFILES で指定したファイルの 1 つが見つかりませんでした。

[EUNIX]
"operating system error"
viewname 検索中にプログラムは VIEWDIR または VIEWFILES で指定したファイルの 1 つを読み込み用に開けませんでした。

[FVFSYNTAX]
"bad viewfile"
viewname 検索中に VIEWDIR または VIEWFILES で指定したファイルの 1 つが破壊されていたか、VIEW ファイルではありませんでした。

[FMALLOC]
"malloc failed"
viewname 検索中に malloc() が VIEW 情報を格納するための領域の割り当てに失敗しました。

移植性 この関数は、Windows 用の BEA Tuxedo System Workstation DLL では使用できません。

関連項目 「FML 関数の紹介」、Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)

Fchg、Fchg32(3fml)

名前 Fchg()、Fchg32() ―フィールド・オカレンス値を変更

形式

```
#include <stdio.h>
#include "fml.h"
int
Fchg(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *value, FLDLEN len)
#include "fml32.h"
int
Fchg32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *value,
 FLDLEN32 len)
```

機能説明 Fchg() は、バッファ内のフィールド値を変更します。*fbfr* は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。*value* は、新しい値を指すポインタです。したがって、その型は、変更される値と同じ型である必要があります (下記参照)。len は、変更される値の長さです。len は、FLD_CARRAY 型または FLD_MBSTRING 型の場合のみ必要です。

-1 のオカレンスが指定される場合、フィールド値は、新しいオカレンスとしてバッファに追加されます。指定されたフィールド・オカレンスが見つかった場合、フィールド値は、指定された値に変更されます。存在しないフィールド・オカレンスが指定された場合、必要とするオカレンスが追加されるまで、見つからないオカレンス用には NULL 値が追加されます (たとえば、バッファに存在しないフィールド用にフィールド・オカレンスを 4 に変更すると、3 個の NULL 値の後に指定されたフィールド値が追加されます)。文字列および文字の値の NULL 値は、NULL 文字列 (長さは 1 バイト) で構成されています。long および short フィールド用の NULL 値は、0 で構成されています。float および double 値用の NULL 値は、0.0 で構成されています。文字配列用の NULL 値は、長さゼロの文字列で構成されています。新しい、または変更した値は、value に含まれます。文字配列の場合 (他の場合は考慮しない)、value の長さは len で指定されます。value が NULL の場合、フィールド・オカレンスは削除されます。削除する値が見つからない場合、エラーと見なされます。

上記の「形式」の項では、Fchg() の引数 *value* のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、Fchg() に渡されるある 1 つの特別な種類の値のみ記述しています。実際、引数 *value* の型は、変更するフィールドのフィールド化バッファの表現の型と同じ型のオブジェクトを指すポインタでなければなりません。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、*value* の型は、long のポインタ (C の long *) である必要があります。同様に、FLD_SHORT 型でバッファに格納される場合、value は、short 型のポインタ (C の short **) である必要があります。大事なことは、Fchg() は、*value* が指すオブジェクトが、変更される格納済みのフィールドの型と同じ型を持っていると想定していることです。

FLD_PTR 型の値の場合、Fchg32() はポインタの値を格納します。FLD_PTR フィールドが指すバッファを割り当てるには、tpalloc() を呼び出します。FLD_FML32 型の値の場合、Fchg32() はインデックスを除いた FLD_FML32 フィールド全体の値を格納します。FLD_VIEW32 型の値の場合、Fchg() は FVIEWFLD 型の構造体を指すポインタを格納します。これには、*vflags* (現在未使用で 0 に設定された flags フィールド)、*vname* (VIEW 名を含む文字配列)、および *data* (C 構造体として格納される VIEW データを指すポインタ) が含まれます。アプリケーションは、*vname* と *data* を Fchg32() に提供します。

FLD_MBSTRING 型の値の場合、値は Fmbpack32() 関数のパッキングされた出力引数であり、引数 *len* は Fmbpack32() の出力引数 *size* の値の長さです。

FLD_CARRAY 型の値の場合、引数 *len* は値の長さです。FLD_CARRAY または FLD_MBSTRING 以外のすべての型では、*value* が参照するオブジェクトの長さは、その型から導かれ (たとえば、FLD_FLOAT の場合、長さは、sizeof(float) になる)、*len* の内容は無視されます。

Fchg32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fchg() または Fchg32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Fchg() は失敗し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスの削除が要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目　「FML 関数の紹介」、CFchg、CFchg32(3fml)、Fadd、Fadd32(3fml)、Fcmp、Fcmp32(3fml)、Fdel、Fdel32(3fml)

Fchgs、Fchgs32(3fml)

名前 Fchgs()、Fchgs32() ―フィールド・オカレンスの変更 - 呼び出し側は、文字列を提供

形式

```
#include <stdio.h>
#include "fml.h"
int
Fchgs(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *value)
#include "fml32.h"
int
Fchgs32(FBFR32 *fbfr, FLDID32 fieldid, int oc, char *value)
```

機能説明 Fchgs() は、ユーザ型 FLD_STRING の変換を処理のために提供されています。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。value は、追加される文字列を指すポインタです。この関数は、文字列を fieldid のフィールド・タイプに変換するための FLD_STRING 型および len (値が 0) を設定して非文字列関数 CFchg() を呼び出します。

Fchgs32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fchgs() または Fchgs32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fchgs() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、CFchg、CFchg32(3fml)、Fchg、Fchg32(3fml)

Fchksum、Fchksum32(3fml)

名前 Fchksum()、Fchksum32() ―フィールド化バッファのチェックサムの計算

形式

```
#include <stdio.h>
#include "fml.h"
long
Fchksum(FBFR *fbfr)
#include "fml32.h"
long
Fchksum32(FBFR32 *fbfr)
```

機能説明 高信頼性の I/O では、チェックサムは、Fchksum() を使用して計算され、書き込まれているフィールド化バッファに格納されます。*fbfr* は、フィールド化バッファを指すポインタです。格納されたチェックサムは、すべてのバッファを受信したことを確かめるために、受信プロセスによって検査されます。

FLD_PTR 型の値の場合、ポインタまたはポインタによって参照されるデータの代わりに、ポインタのフィールドの名前がチェックサム計算に含まれます。

Fchksum32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fchksum() または Fchksum32() を呼び出すことができます。

戻り値 正常終了の場合、Fchksum() はチェックサムを返します。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fchksum() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Fread、Fread32(3fml)、Fwrite、Fwrite32(3fml)

Fcmp、Fcmp32(3fml)

名前 Fcmp()、Fcmp32() ― 2 つのフィールド化バッファの比較

形式

```
#include <stdio.h>
#include "fml.h"
int
Fcmp(FBFR *fbfr1, FBFR *fbfr2)
#include "fml32.h"
int
Fcmp32(FBFR32 *fbfr1, FBFR32 *fbfr2)
```

機能説明 Fcmp() は、フィールド識別子を比較してから、2 つの FML バッファのフィールド値を比較します。fbfr1 および fbfr2 は、比較するフィールド化バッファを指すポインタです。

FLD_PTR 型の値の場合、ポインタの値 (アドレス) が等しければ、2 つのポインタ・フィールドは等しいと見なされます。FLD_FML32 型の値の場合、すべてのフィールド・オカレンスおよび値が等しければ、2 つのフィールドは等しいと見なされます。FLD_VIEW32 型の値の場合、VIEW 名が同じとき、およびすべての構造体メンバのオカレンスと値が等しいとき、2 つのフィールドは等しいと見なされます。

Fcmp32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fcmp() または Fcmp32() を呼び出すことができます。

戻り値 この関数は、2 つのバッファが等しい場合には 0 を返します。次に示す条件の場合 -1 を返します。

- fbfr1 フィールドのフィールド識別子が、比較する fbfr2 フィールドのフィールド識別子より小さい。
- fbfr1 のフィールド値が、比較する fbfr2 フィールドのフィールド値より小さい。
- fbfr1 が fbfr2 より短い。

Fcmp() は、上記のいずれかの条件の逆が true であると、1 を返します。たとえば、fbfr1 フィールドのフィールド識別子が fbfr2 フィールドの対応するフィールド識別子より大きい場合です。この場合、バッファの実際の大きさ (つまり、Falloc() に渡された大きさ) は認識されず、バッファのデータのみが認識されます。この関数は、エラー発生時に -2 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fcmp() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml)

Fconcat、Fconcat32(3fml)

名前 Fconcat()、Fconcat32() ―ソース・バッファと宛先バッファの連結

形式

```
#include <stdio.h>
#include "fml.h"
int
Fconcat(FBFR *dest, FBFR *src)
#include "fml32.h"
int
Fconcat32(FBFR32 *dest, FBFR32 *src)
```

機能説明 Fconcat() は、ソース・バッファのフィールドを、すでにある宛先バッファにあるフィールドに追加します。dest は、宛先バッファのポインタです。src は、元のフィールド化バッファを指すポインタです。宛先バッファにあるオカレンスは維持され、ソース・バッファの新しいオカレンスは、フィールドに対して比較的大きなオカレンス番号で追加されます。

Fconcat32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fconcat() または Fconcat32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fconcat() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファに追加されますが、バッファには十分な領域が残っていません。

関連項目 「FML 関数の紹介」、Fjoin、Fjoin32(3fml)、Fupdate、Fupdate32(3fml)

Fcpy、Fcpy32(3fml)

名前 Fcpy()、Fcpy32() ―ソース・バッファを宛先バッファにコピー

形式

```
#include <stdio.h>
#include "fml.h"
int
Fcpy(FBFR *dest, FBFR *src)
#include "fml32.h"
int
Fcpy32(FBFR32 *dest, FBFR32 *src)
```

機能説明 Fcpy() は、あるフィールド化バッファの内容を他のフィールド化バッファに複写するために使用されます。dest は、宛先バッファのポインタです。src は、元のフィールド化バッファを指すポインタです。Fcpy() は、宛先がフィールド化バッファであることを想定しています。このようにすると、宛先バッファに、ソース・バッファのデータを入れるための十分な大きさがあるかを検査できます。

FLD_PTR 型の値の場合、Fcpy32() はバッファのポインタをコピーします。アプリケーション・プログラマは、対応するポインタがコピーされたときのバッファの再割り当ておよび解放を管理する必要があります。

Fcpy32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fcpy() または Fcpy32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fcpy() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
宛先バッファは、ソース・バッファを保持できる十分な大きさではありません。

関連項目 「FML 関数の紹介」、Fmove、Fmove32(3fml)

Fdel、Fdel32(3fml)

名前 Fdel()、Fdel32() ―バッファからフィールド・オカレンスを削除

形式

```
#include stdio.h>
#include "fml.h"
int
Fdel(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"
int
Fdel32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Fdel() は、指定されたフィールド・オカレンスをバッファから削除します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。

フィールド化バッファに、1 つのフィールドに対し複数のオカレンスがあり、最後のオカレンスでないフィールド・オカレンスが削除された場合、バッファにある高位のオカレンスは、1 つずつ繰り下がります。すべてのオカレンスのオカレンス番号を同じままにするためには、Fchg() を使用してフィールド・オカレンス値を NULL に設定します。

FLD_PTR 型の値の場合、Fdel32() によって FLD_PTR フィールド・オカレンスが削除されても、参照されるバッファは変更されないか、またはポインタは解放されません。データ・バッファは、オペークなポインタとして扱われます。

Fdel32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fdel() または Fdel32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fdel() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml)、Fdelall、Fdelall32(3fml)、Fdelete、Fdelete32(3fml)

Fdelall、Fdelall32(3fml)

名前 Fdelall()、Fdelall32() ―バッファからすべてのフィールド・オカレンスを削除

形式

```
#include <stdio.h>
#include "fml.h"
int
Fdelall(FBFR *fbfr, FLDID fieldid)
#include "fml32.h"
int
Fdelall32(FBFR32 *fbfr, FLDID32 fieldid)
```

機能説明 Fdelall() は、バッファにある指定されたフィールドのすべてのオカレンスを削除します。*fbfr* は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。フィールドのオカレンスがない場合は、エラーと見なされます。

FLD_PTR 型の値の場合、Fdelall32() によって FLD_PTR フィールド・オカレンスが削除されても、参照されるバッファは変更されないか、またはポインタは解放されません。データ・バッファは、オペークなポインタとして扱われます。

Fdelall32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fdelall() または Fdelall32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fdelall() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールドが要求されましたが、指定されたフィールドは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fdel、Fdel32(3fml)、Fdelete、Fdelete32(3fml)

Fdelete、Fdelete32(3fml)

名前 Fdelete()、Fdelete32() ―バッファからフィールドのリストを削除

形式

```
#include <stdio.h>
#include "fml.h"
int
Fdelete(FBFR *fbfr, FLDID *fieldid)
#include "fml32.h"
int
Fdelete32(FBFR32 *fbfr, FLDID32 *fieldid)
```

機能説明 Fdelete() は、フィールド識別子の配列 (fieldid[]) にリストされているすべてのフィールドのすべてのオカレンスを削除します。配列の最後のエントリは、BADFLDID である必要があります。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子の配列を指すポインタです。これは、バッファからいくつものフィールドを削除する場合、Fdelall() を数回呼び出すより有効な方法です。更新は、その場所で行われます。フィールド識別子の配列は、Fdelete() によって再配列されます (それらがソートされていない場合はソートされ、番号順になります)。

FLD_PTR 型の値の場合、Fdelete32() によって FLD_PTR フィールド・オカレンスが削除されても、参照されるバッファは変更されないか、またはポインタは解放されません。データ・バッファは、オペークなポインタとして扱われます。

Fdelete() は、削除するフィールドがフィールド化バッファにない場合も正常終了で返ります。

Fdelete32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fdelete() または Fdelete32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fdelete() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fdel、Fdel32(3fml)、Fdelall、Fdelall32(3fml)

Fextread、Fextread32(3fml)

名前 Fextread()、Fextread32() ―フォーマットされた出力からフィールド化バッファを作成

形式

```
#include <stdio.h>
#include "fml.h"
int
Fextread(FBFR *fbfr, FILE *iop)
#include "fml32.h"
int
Fextread32(FBFR32 *fbfr, FILE *iop)
```

機能説明 Fextread() は、フォーマットされた出力 (つまり、Fprint() の出力) からフィールド化バッファを作成するために使用されます。パラメータは、フィールド化バッファを指すポインタ (fbfr) とファイル・ストリームを指すポインタ (iop) です。入力のファイル形式は、基本的に Fprint() の出力形式と同じです。次にそれを示します。

```
[flag] fldname or fldid tab> fldval (or fldname, if flag is ``='')
```

選択フラグおよびそれらの意味を次に示します。

+
: フィールド化バッファにあるフィールドのオカレンス 0 は、指定された値に変更されます。

-
: 指定されたフィールドのオカレンス 0 は、フィールド化バッファから削除されます。タブ文字が必要です。すべてのフィールド値は、無視されます。

=
: この場合、入力行の最後のフィールドは、フィールド化バッファ内のフィールド名です。そのフィールドのオカレンス 0 の値は、入力行の最初のフィールドで指定されたオカレンス 0 に割り当てられます。

#
: この行はコメントとして扱われ、無視されます。

フラグが指定されない場合、値 (*fldval*) を持った *fldname* で指定されているフィールドの新しいオカレンスが、フィールド化バッファに追加されます。後続の改行 (-) は、入力が済んだ各バッファの後に置く必要があります。

FLD_FML32 および FLD_VIEW32 型の値の場合、Fextread32() はそれぞれ、入れ子になった FML32 バッファと VIEW32 フィールドを生成します。この関数は、FLD_PTR フィールド・タイプを無視します。関数に FLD_PTR 型の値が与えられても、エラーは返されません。

Fextread32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fextread() または Fextread32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fextread() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド番号は無効です。

[FEUNIX]
"UNIX system call error"
UNIX システム・コール・エラーが起こりました。外部の整数型変数 errno は、システム・コールによるエラーを示すために設定され、外

部の整数型変数 Uunixerr (Uunix.h で定義されている値) は、エラーを返したシステム・コールに設定されます。

[FBADNAME]
"unknown field name"
フィールド・テーブルにないフィールド名が指定されました。

[FSYNTAX]
"bad syntax in format"
外部バッファの形式に文法エラーがありました。発生する可能性のあるエラーとしては、不正な EOF 指示子、次の形式にない入力行 (fieldid または name tab > value)、2 つの制御文字、1000 文字より大きいフィールド値、または無効な 16 進のエスケープ・シーケンスなどです。

[FNOTPRES]
"field not present"
削除するフィールドが、フィールド化バッファにありません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid parameter"
iop の値が NULL です。

移植性 この関数は、Windows 用の BEA Tuxedo System Workstation DLL では使用できません。

関連項目 「FML 関数の紹介」、Fprint、Fprint32(3fml)

Ffind、Ffind32(3fml)

名前 Ffind()、Ffind32() ―バッファにあるフィールド・オカレンスの検索

形式

```
#include <stdio.h>
#include "fml.h"
char *
Ffind(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *len)
#include "fml32.h"
char *
Ffind32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32 *len)
```

機能説明 Ffind() は、バッファ内の指定されたフィールド・オカレンスの値を検索します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。フィールドがある場合、その長さは、*len に設定され、その位置は、関数の値として返されます。len の値が NULL の場合は、フィールドの長さは返されません。Ffind() は、フィールドへの読み取り専用アクセスを得る場合に役立ちます。どのような場合でも、Ffind() が返す値は、バッファの変更に使用してはなりません。

一般に、FLD_LONG、FLD_FLOAT、FLD_DOUBLE、FLD_PTR、FLD_FML32、および FLD_VIEW32 型の場合、それぞれの値がバッファ内で適切に配置されているとは限らないため、格納されているままの型を直接使用するのには向いていません。これらの値は、最初に、適宜に配列されたメモリ位置に複写されなければなりません。変換関数 CFfind() を使用してこのようなフィールドにアクセスする場合、検索される変換済みの値は正しく配列されることが保証されます。バッファの変更は、関数 Fadd() または Fchg() によってのみ行われる必要があります。Ffind() および Ffindlast() が返す値は、バッファが変更されない限り有効です。

Ffind32() は、FLD_FML32 および FLD_VIEW32 フィールド・タイプで提供される、埋め込み型のバッファ内の指定フィールドのオカレンスをチェックしません。

FLD_MBSTRING 型の指定されたフィールド識別子の場合、Ffind32() の戻り値は Fmbunpack(32) 関数を使用して解析できます。

Ffind32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffind() または Ffind32() を呼び出すことができます。

戻り値　上記の「形式」の項では、Ffind() の戻り値のデータ型は、char のポインタ (C の char *) として記述されています。実際、返されるポインタは、格納済みのフィールドの型と同じ型を持つオブジェクトを指しています。

この関数は、エラー時には NULL のポインタを返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Ffind() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目　「FML 関数の紹介」、Ffindlast、Ffindlast32(3fml)、Ffindocc、Ffindocc32(3fml)、Ffinds、Ffinds32(3fml)

Ffindlast、Ffindlast32(3fml)

名前 Ffindlast()、Ffindlast32() ―バッファにあるフィールドの最後のオカレンスの値を検索

形式

```
#include <stdio.h>
#include "fml.h"
char *
Ffindlast(FBFR *fbfr, FLDID fieldid, FLDOCC *oc, FLDLEN *len)
#include "fml32.h"
char *
Ffindlast32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 *oc, FLDLEN32
*len)
```

機能説明 Ffindlast() は、バッファにあるフィールドの最後のオカレンスの値を検索します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc は、フィールドのオカレンス番号を受け取るために使用する整数型のポインタです。len は、値の長さです。バッファにフィールドのオカレンスがない場合、NULL が返されます。一般に、Ffindlast() は Ffind() に似た動きをします。Ffindlast を使用すると、ユーザは、フィールド・オカレンスを提供しないという点が主な違いです。その代わり、フィールドの最後のオカレンスの値およびオカレンス番号の両方が返されます。最後のフィールドのオカレンス番号を返すために、Ffindlast() のオカレンス引数 oc は、整数型のポインタであり、Ffind() の場合のように整数ではありません。oc が NULL で指定されている場合、最後のオカレンスのオカレンス番号は返されません。len の値が NULL の場合は、フィールドの長さは返されません。

一般に、FLD_LONG、FLD_FLOAT、FLD_DOUBLE、FLD_PTR、FLD_FML32、および FLD_VIEW32 型の場合、それぞれの値がバッファ内で適切に配置されているとは限らないため、格納されているままの型を直接使用するのには向いていません。これらの値は、最初に、適宜に配列されたメモリ位置に複写される必要があります。変換関数 CFfind() を使用してこのようなフィールドにアクセスする場合、検索される変換済みの値は正しく配列されることが保証されます。バッファの変更は、関数 Fadd() または Fchg() によってのみ行われる必要があります。Ffind() および Ffindlast() が返す値は、バッファが変更されない限り有効です。

Ffindlast32() は、FLD_FML32 および FLD_VIEW32 フィールド・タイプで提供される、埋め込み型のバッファ内の指定フィールドのオカレンスをチェックしません。

FLD_MBSTRING 型の指定されたフィールド識別子の場合、Ffindlast32() の戻り値は Fmbunpack(32) 関数を使用して解析できます。

Ffindlast32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffindlast() または Ffindlast32() を呼び出すことができます。

戻り値 上記の「形式」の項では、Ffindlast() の戻り値のデータ型は、char のポインタ (C の char *) として記述されています。実際、返されるポインタは、格納済みのフィールドの型と同じ型を持つオブジェクトを指しています。

この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ffindlast() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールドが要求されましたが、指定されたフィールドは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目　「FML 関数の紹介」、CFfind、CFfind32(3fml)、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml)、Ffind、Ffind32(3fml)、Ffindocc、Ffindocc32(3fml)、Ffinds、Ffinds32(3fml)

Ffindocc、Ffindocc32(3fml)

名前 Ffindocc()、Ffindocc32() ―フィールド値のオカレンスの検索

形式

```
#include <stdio.h>
#include "fml.h"
FLDOCC
Ffindocc(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len)
#include "fml32.h"
FLDOCC32
Ffindocc32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32
len)
```

機能説明 Ffindocc() は、バッファ内の指定されたフィールドのオカレンスを参照し、ユーザ指定のフィールド値と一致する最初のフィールド・オカレンスのオカレンス番号を返します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。検索される値は、value パラメータが指す場所にあります。len は、その型が FLD_CARRAY または FLD_MBSTRING の場合、値の長さです。fieldid がフィールド型 FLD_STRING で、len が 0 でない場合、文字列に対してパターン整合が行われます。サポートされているパターン整合は、regcmp(3) (UNIX System V プログラマ・リファレンス・マニュアル) に記述されているパターンと同じです。さらに、正規表現の代替がサポートされています。たとえば、"A|B" は、"A" または "B" と一致しています。パターンは、全体のフィールド値と一致する必要があります。すなわち、パターン "value" は、暗黙的に "^value$" として扱われます。MS-DOS および OS/2 環境用に提供される Ffindocc() のバージョンは、FLD_STRING フィールドに対する regcmp(3) のパターン整合機能をサポートしていませんので、strcmp(3) (UNIX System V プログラマ・リファレンス・マニュアル) を使用します。

上記の「形式」の項では、Ffindocc() の引数 value のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、これは、Ffindocc() に渡すことのできる値のある特定の値のみを記述しています。実際、value 引数の型は、検索されるフィールドのフィールド化バッファ表現の型と同じ型のオブジェクトのポインタである必要があります。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、value の型は、long のポインタ (C の long *) である必要があります。同様に、

FLD_SHORT 型でバッファに格納される場合、value は、short 型のポインタ (C の short **) である必要があります。大事な点は、Ffindocc() は、value で指されるオブジェクトが、格納済みの検索されるフィールドの型と同じ型を持っていることを想定していることです。

FLD_PTR 型の値の場合、Ffindocc32() は指定されたポインタ値と一致するフィールドのオカレンスを検索します。FLD_FML32 型の値の場合、すべてのフィールド・オカレンスおよび値が等しければ、2 つのフィールドは等しいと見なされます。FLD_VIEW32 型の値の場合、VIEW 名が同じとき、およびすべての構造体メンバのオカレンスおよび値が等しいとき、2 つのフィールドは等しいと見なされます。

FLD_MBSTRING 型の値の場合、値は Fmbpack32() 関数のパッキングされた出力引数であり、引数 *len* は Fmbpack32() の出力引数 *size* の値の長さです。

Ffindocc32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffindocc() または Ffindocc32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ffindocc() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド値が要求されましたが、指定されたフィールドおよび値は、フィールド化バッファにありませんでした。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが、無効でした (たとえば、NULL の value パラメータを Ffindocc() に渡した、あるいは無効な文字列パターンを指定した)。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Ffind、Ffind32(3fml)、Ffindlast、Ffindlast32(3fml)、Ffinds、Ffinds32(3fml)、UNIX システムのリファレンス・マニュアルの regcmp(3)

Ffinds、Ffinds32(3fml)

名前 Ffinds()、Ffinds32() ー文字列表現を指すポインタを返す

形式

```
#include <stdio.h>
#include "fml.h"
char *
Ffinds(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"
char *
Ffinds32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Ffinds() は、ユーザ型 FLD_STRING に変換する処理のために提供されます。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。指定されたフィールド・オカレンスは、検索されると、バッファにあるその型から NULL 終了の文字列に変換されます。基本的に、このマクロは変換関数 CFfind() を呼び出します。CFfind() は、utype (FLD_STRING) および ulen (値が 0) を提供します。Ffinds() が返すポインタが妥当である期間は、CFfind() で記述されているものと同じです。

Ffinds32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffinds() または Ffinds32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ffinds() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド型は無効です。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、`malloc()` を使用しての領域の動的な割り当てが失敗しました。

関連項目 「FML 関数の紹介」、`CFfind、CFfind32(3fml)`、`Ffind、Ffind32(3fml)`

Ffloatev、Ffloatev32、Fvfloatev、Fvfloatev32(3fml)

名前 Ffloatev()、Ffloatev32()、Fvfloatev()、Fvfloatev32() 一式の値を double 型で返す

形式

```
#include <stdio.h>
#include "fml.h"
double
Ffloatev(FBFR *fbfr, char *tree)
double
Fvfloatev(char *cstruct, char *tree, char *viewname)
#include "fml32.h"
double
Ffloatev32(FBFR32 *fbfr, char *tree)
double
Fvfloatev32(char *cstruct, char *tree, char *viewname)
```

機能説明 Ffloatev() は、フィールド化バッファのポインタ (fbfr) および Fboolco() が返す評価ツリーのポインタ (tree) を利用して、算術式の値 (ツリーによって表される) を double 型で返します。この関数は、フィールド化バッファあるいは評価ツリーのいずれも変更しません。

Ffloatev32() は 32 ビット FML で使用されます。

Fvfloatev() および Fvfloatev32() は、同じ VIEW 機能を提供します。viewname パラメータは、フィールド・オフセットを取り出す VIEW を示し、Fvboolco() または Fvboolco32() で指定した VIEW と同じである必要があります。

Warkstation プラットフォームでは、これらの関数はサポートされていません。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、ここで記述する関数、Ffloatev()、Ffloatev32()、Fvfloatev()、または Fvfloatev32() を呼び出すことができます。

戻り値 正常終了時には、Ffloatev() は、表現の値を double で返します。

この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Ffloatev() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

[FSYNTAX]
"bad syntax in Boolean expression"
論理表現の tree に文法エラーがありました。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに viewname が見つかりません。

[FVFOPEN]
"cannot find or open viewfile"
viewname 検索中にプログラムで VIEWDIR または VIEWFILES で指定したファイルの 1 つが見つかりませんでした。

[EUNIX]
"operating system error"
viewname 検索中にプログラムは VIEWDIR または VIEWFILES で指定したファイルの 1 つを読み込み用に開けませんでした。

[FVFSYNTAX]
"bad viewfile"
viewname 検索中に VIEWDIR または VIEWFILES で指定したファイルの 1 つが破壊されていたか、VIEW ファイルではありませんでした。

[FMALLOC]
"malloc failed"
viewname 検索中に malloc() が VIEW 情報を格納するための領域の割り当てに失敗しました。

関連項目 「FML 関数の紹介」、Fboolco、Fboolco32、Fvboolco、Fvboolco32(3fml)、Fboolev、Fboolev32、Fvboolev、Fvboolev32(3fml)

Ffprint、Ffprint32(3fml)

名前 Ffprint()、Ffprint32() ―フィールド化バッファを指定されたストリームに出力

形式

```
#include <stdio.h>
#include "fml.h"
int
Ffprint(FBFR *fbfr, FILE *iop)
#include "fml32.h"
int
Ffprint32(FBFR32 *fbfr, FILE *iop)
```

機能説明 Ffprint() は、テキストが指定された出力ストリームに出力されることを除いて、Fprint() に類似しています。fbfr は、フィールド化バッファを指すポインタです。iop は、出力ストリームを指す FILE 型のポインタです。

バッファの各フィールドでは、フィールド名およびタブ区切りのフィールド値を出力します。Fname() は、フィールド名を決めるために使用されます。フィールド名を決めることができない場合は、フィールド識別子が出力されます。文字列内の出力できない文字および文字配列のフィールド値は、2 文字の 16 進数 (バックスラッシュの後) で表されます。改行は、出力されるバッファの後に出力されます。

FLD_PTR 型の値の場合、Ffprint32() はフィールド名またはフィールド識別子、およびポインタ値を 16 進数で出力します。この関数はポインタ情報を出力しますが、Fextread32() 関数は FLD_PTR フィールド・タイプを無視します。

FLD_FML32 型の値の場合、Ffprint32() は各入れ子レベルに先行タブを追加することにより、再帰的に FML32 バッファを出力します。FLD_VIEW32 型の値の場合、Ffprint32() は VIEW32 フィールド名、および構造体メンバの名前と値のペアを出力します。

Ffprint32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffprint() または Ffprint32() を呼び出すことができます。

戻り値　この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Ffprint() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

移植性　この関数は、Windows 用の BEA Tuxedo System Workstation DLL では使用できません。

関連項目　「FML 関数の紹介」、Fprint、Fprint32(3fml)

Ffree、Ffree32(3fml)

名前 Ffree()、Ffree32() ―フィールド化バッファ用に割り当てられた領域の解放

形式

```
#include <stdio.h>
#include "fml.h"
int
Ffree(FBFR *fbfr)
#include "fml32.h"
int
Ffree32(FBFR32 *fbfr)
```

機能説明 Ffree() は、その引数であるフィールド化バッファ用に割り当てられた領域を取り戻します。*fbfr* は、フィールド化バッファを指すポインタです。フィールド化バッファは、無効になります。すなわち、非フィールド化となり、解放されます。Ffree32() は、FLD_PTR フィールドのポインタによって参照されるメモリ領域を解放しません。

Ffree() は、(UNIX システムのリファレンス・ページの) free() よりも有効です。Ffree() は、フィールド化バッファを無効にしますが、free() は無効にしません。フィールド化バッファを無効にすることは重要です。それは、(UNIX システムのリファレンス・ページの) malloc() が、解放されたメモリをクリアせずに再使用するためです。このため、free() が使用されると、malloc() は、有効なフィールド化バッファのようであるが実際はそうではないメモリを返す恐れがあります。

Ffree32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ffree() または Ffree32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ffree() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Falloc、Falloc32(3fml)、Frealloc、Frealloc32(3fml)

UNIX システムのリファレンス・ページの free(3)、malloc(3)

Fget、Fget32(3fml)

名前 Fget()、Fget32() ―フィールド・オカレンスのコピーおよび長さの取得

形式

```
#include <stdio.h>
#include "fml.h"
int
Fget(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *loc, FLDLEN
  *maxlen)
#include "fml32.h"
int
Fget32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *loc,
  FLDLEN32 *maxlen)
```

機能説明 Fget() は、値が変更されたときに、フィールド化バッファのフィールドを検索するために使用されます。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。呼び出し側は、プライベート・データ領域を指すポインタ (loc)、データ領域の長さ (*maxlen) を Fget() に渡します。フィールドの長さは、*maxlen に返されます。関数が呼び出された時に、maxlen が NULL の場合は、そのフィールドの値 (loc) のデータ領域がフィールド値を入れるのに十分大きいために、値が返されないと想定されます。loc が NULL の場合は、値は検索されません。このように、この関数を呼び出すことにより、フィールドが存在するかどうかを確かめることができます。

上記の「形式」の項では、Fget() の引数 value のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、これは、Fget() に渡すことのできる値のある特定の値のみを記述しています。実際、value 引数の型は、検索されるフィールドのフィールド化バッファ表現の型と同じ型のオブジェクトのポインタである必要があります。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、value の型は、long のポインタ (C の long *) である必要があります。同様に、FLD_SHORT 型でバッファに格納される場合、value は、short 型のポインタ (C の short **) である必要があります。大事なことは、Fget() は value で指されるオブジェクトが、検索される格納済みのフィールドの型と同じ型を持っていることを想定していることです。

FLD_MBSTRING 型の指定されたフィールド識別子の場合、Fget32() の戻り値は Fmbunpack(32) 関数を使用して解析できます。

Fget32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fget() または Fget32() を呼び出すことができます。

戻り値 Fget32() が FLD_VIEW32 フィールド・タイプで使用されると、FVIEWFLD 構造体を指すポインタが返されます。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fget() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space"
maxlen に指定されているデータ領域の大きさは、フィールド値を保持するのに十分な大きさではありません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目　「FML 関数の紹介」、CFget、CFget32(3fml)、Fgetalloc、Fgetalloc32(3fml)、Fgetlast、Fgetlast32(3fml)、Fgets、Fgets32(3fml)、Fgetsa、Fgetsa32(3fml)

Fgetalloc、Fgetalloc32(3fml)

名前 Fgetalloc()、Fgetalloc32() ー領域の割り当て、およびフィールド・オカレンスのコピーの取得

形式

```
#include <stdio.h>
#include "fml.h"
char *
Fgetalloc(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *extralen)
#include "fml32.h"
char *
Fgetalloc32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32
  *extralen)
```

機能説明 Fget() 同様、Fgetalloc() は、バッファ・フィールドのコピーの検索および作成を行いますが、malloc() (UNIX System V『プログラマ・リファレンス・マニュアル』) を介してフィールド用の領域を取得します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。Fgetalloc() の最後の引数 extralen は、フィールド値の大きさに加えて取得される予備の領域の大きさを提供します。extralen は、検索される値が、フィールド化バッファに再挿入される前に拡張される場合に使用されます。extralen が NULL の場合、割り当てる追加領域はなく、実際の長さは返されません。Fgetalloc() で取得した領域を free() で解放するのは呼び出し側の責任です。バッファは、あらゆるフィールドの型に対して正しく並べられます。

FLD_MBSTRING 型の指定されたフィールド識別子の場合、Fgetalloc32() の戻り値は Fmbunpack(32) 関数を使用して解析できます。

Fgetalloc32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fgetalloc() または Fgetalloc32() を呼び出すことができます。

戻り値 上記の「形式」の項では、Fgetalloc() の戻り値のデータ型は、char のポインタ (C の char *) として記述されています。実際、返されるポインタは、格納済みのフィールドの型と同じ型を持つオブジェクトを指しています。

Fgetalloc32() が FLD_VIEW32 フィールド・タイプで使用されると、FVIEWFLD 構造体を指すポインタが返されます。この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Fgetalloc() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FMALLOC]
“malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目　「FML 関数の紹介」、CFget、CFget32(3fml)、Fget、Fget32(3fml)、Fgetlast、Fgetlast32(3fml)、Fgets、Fgets32(3fml)、Fgetsa、Fgetsa32(3fml)

UNIX システムのリファレンス・ページの free(3)、malloc(3)

Fgetlast、Fgetlast32(3fml)

名前 Fgetlast()、Fgetlast32() ―最後のオカレンスのコピーの取得

形式

```
#include <stdio.h>
#include "fml.h"
int
Fgetlast(FBFR *fbfr, FLDID fieldid, FLDOCC *oc, char *value, FLDLEN
  *maxlen)
#include "fml32.h"
int
Fgetlast32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 *oc, char
  *value, FLDLEN32 *maxlen)
```

機能説明 Fgetlast() は、フィールド識別子 (fieldid) で識別される最後のオカレンスの値とオカレンス番号の両方を検索するために使用されます。fbfr は、フィールド化バッファを指すポインタです。最後のフィールドのオカレンス番号を返すため、オカレンスの引数 (oc) は、整数型を指すポインタであり、整数ではありません。

呼び出し側は、プライベート・バッファを指すポインタ (loc)、バッファの長さ (*maxlen) を Fgetlast() に提供します。フィールドの長さは、*maxlen に返されます。関数が呼び出された時に、maxlen が NULL の場合は、そのフィールド値のバッファがフィールド値を入れるのには十分に大きいために、値が返されないと想定されます。loc が NULL の場合、値は返されません。oc が NULL の場合は、オカレンスは、返されません。

上記の「形式」の項では、Fgetlast() の引数 value のデータ型は、char のポインタ (C の char *) として記述されています。技術的には、これは、Fgetlast() に渡すある特別な値のみを記述します。実際、value 引数の型は、検索されるフィールドのフィールド化バッファ表現の型と同じ型のオブジェクトのポインタである必要があります。たとえば、フィールドが、FLD_LONG 型でバッファに格納されている場合は、value の型は、long 型のポインタ (C の long *) である必要があります。同様に、FLD_SHORT 型でバッファに格納される場合、value は、short 型のポインタ (C の short *) である必要があります。Fgetlast() は value で指されるオブジェクトが、検索される格納済みのフィールドの型と同じ型を持っていることを想定していることが重要です。

FLD_MBSTRING 型の指定されたフィールド識別子の場合、Fgetlast32() の戻り値は Fmbunpack(32) 関数を使用して解析できます。

Fgetlast32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fgetlast() または Fgetlast32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fgetlast() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space"
maxlen に指定されているデータ領域の大きさは、フィールド値を保持するのに十分な大きさではありません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fget、Fget32(3fml)、Fgetalloc、Fgetalloc32(3fml)、Fgets、Fgets32(3fml)、Fgetsa、Fgetsa32(3fml)

Fgets、Fgets32(3fml)

名前 Fgets()、Fgets32() ー値を文字列に変換

形式

```
#include <stdio.h>
#include "fml.h"
int
Fgets(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *buf)
#include "fml32.h"
int
Fgets32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *buf)
```

機能説明 Fgets() は、フィールド化バッファからフィールド・オカレンスを検索し、値をユーザ型 FLD_STRING に変換します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。Fgets() の呼び出し側は、プライベート・バッファを指すポインタ (buf) を提供します。プライベート・バッファは、検索されるフィールド値のために使用されます。buf は、値を保持するために十分に大きいと想定されています。基本的に、Fgets() は、想定されている utype (FLD_STRING) および ulen (値が 0) を利用して CFget() を呼び出します。

Fgets32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fgets() または Fgets32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fgets() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目 「FML 関数の紹介」、CFget、CFget32(3fml)、Fget、Fget32(3fml)、Fgetalloc、Fgetalloc32(3fml)、Fgetlast、Fgetlast32(3fml)、Fgetsa、Fgetsa32(3fml)

Fgetsa、Fgetsa32(3fml)

名前 Fgetsa()、Fgetsa32() ― malloc() で領域を割り当て、変換された値を取得する

形式

```
#include <stdio.h>
#include "fml.h"
char *
Fgetsa(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *extra)
#include "fml32.h"
char *
Fgetsa32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32
  *extra)
```

機能説明 Fgetsa() は、CFgetalloc() を呼び出すマクロです。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。この関数は、UNIX System V『プログラマ・リファレンス・マニュアル』の malloc() を使用して、文字列に変換されている検索対象のフィールド値に領域を割り当てます。extra が NULL でない場合、extra には、フィールド値の大きさに加えて割り当てる余分な領域を指定します。全体の大きさは、extra に返ります。

malloc() で割り当てた領域を (UNIX システムのリファレンス・ページの) free() を使用して解放するのは、ユーザの責任です。

Fgetsa32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fgetsa() または Fgetsa32() を呼び出すことができます。

戻り値 正常終了の場合、割り当てられたバッファを指すポインタを返します。

この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fgetsa() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目 「FML 関数の紹介」、CFget、CFget32(3fml)、Fget、Fget32(3fml)、Fgetlast、Fgetlast32(3fml)、Fgets、Fgets32(3fml)

UNIX システムのリファレンス・ページの free(3)、malloc(3)

Fidnm_unload、Fidnm_unload32(3fml)

名前 Fidnm_unload()、Fidnm_unload32() ― *id->nm* マッピング・テーブルから領域を取り戻す

形式

```
#include <stdio.h>
#include "fml.h"
void
Fidnm_unload(void);
#include "fml32.h"
void
Fidnm_unload32(void);
```

機能説明 Fidnm_unload() は、Fname() によってフィールド識別子用に割り当てられた領域を、フィールド名のマッピング・テーブル用に取り戻します。

Fidnm_unload32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fidnm_unload() または Fidnm_unload32() を呼び出すことができます。

戻り値 この関数は、void 宣言のため何も返しません。

関連項目 「FML 関数の紹介」、Fname、Fname32(3fml)、Fnmid_unload、Fnmid_unload32(3fml)

Fidxused、Fidxused32(3fml)

名前 Fidxused()、Fidxused32() ー使用されている領域の大きさを返す

形式

```
#include <stdio.h>
#include "fml.h"
long
Fidxused(FBFR *fbfr)
#include "fml32.h"
long
Fidxused32(FBFR32 *fbfr)
```

機能説明 Fidxused() は、バッファのインデックスで使用している現在の領域の大きさを示します。fbfr は、フィールド化バッファを指すポインタです。

Fidxused32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fidxused() または Fidxused32() を呼び出すことができます。

戻り値 正常終了の場合、インデックスで使用しているバッファ内の領域の大きさを返します。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fidxused() は、異常終了して Ferror を次のように設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Findex、Findex32(3fml)、Frstrindex、Frstrindex32(3fml)、Funused、Funused32(3fml)、Fused、Fused32(3fml)

Fielded、Fielded32(3fml)

名前 Fielded()、Fielded32() ―バッファがフィールド化されている場合は、true を返す

形式

```
#include stdio.h>
#include "fml.h"
int
Fielded(FBFR *fbfr)
#include "fml32.h"
int
Fielded32(FBFR32 *fbfr)
```

機能説明 Fielded() は、指定されたバッファがフィールド化されているかを調べるのに使用されます。fbfr は、フィールド化バッファを指すポインタです。

Fielded32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fielded() または Fielded32() を呼び出すことができます。

戻り値 Fielded() は、バッファがフィールド化されている場合には true を返します。また、バッファがフィールド化されていない場合は、false を返しますが、この場合は Ferror を設定しません。

関連項目 「FML 関数の紹介」、Finit、Finit32(3fml)、Fneeded、Fneeded32(3fml)、Fsizeof、Fsizeof32(3fml)

Findex、Findex32(3fml)

名前 Findex()、Findex32() ―フィールド化バッファのインデックスを行う

形式

```
#include <stdio.h>
#include "fml.h"
int
Findex(FBFR *fbfr, FLDOCC intvl)
#include "fml32.h"
int
Findex32(FBFR32 *fbfr, FLDOCC32 intvl)
```

機能説明 Findex() は、フィールド化バッファのインデックスを行うために明示的に呼び出されます。fbfr は、フィールド化バッファを指すポインタです。2 番目のパラメータ intvl には、インデックスの間隔を指定します。すなわち、インデックスするフィールドを理想的に分離します。intvl の値が 0 の場合は、現在のバッファのインデックス値が使用されます。現在の値が 0 の場合は、FSTDXINTVL (省略時値は 16) が使用されます。インデックス値に 1 を使用すると、必ず、バッファにあるすべてのフィールドがインデックスされます。インデックス間隔の大きさとバッファのインデックスに割り当てられる領域の大きさは、反比例します。したがって、間隔が小さいとより多くのフィールドがインデックスされ、より多くの領域が使用されます。

Findex32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Findex() または Findex32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Findex() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
エントリは、インデックスに追加されますが、バッファには十分な領域が残っていません。

関連項目 「FML 関数の紹介」、Fidxused、Fidxused32(3fml)、Frstrindex、Frstrindex32(3fml)、Funindex、Funindex32(3fml)

Finit、Finit32(3fml)

名前 Finit()、Finit32() ―フィールド化バッファを初期化

形式

```
#include <stdio.h>
#include "fml.h"
int
Finit(FBFR *fbfr, FLDLEN buflen)
#include "fml32.h"
int
Finit32(FBFR32 *fbfr, FLDLEN32 buflen)
```

機能説明 Finit() は、フィールド化バッファを静的に初期化するために呼び出されます。fbfr は、フィールド化バッファを指すポインタです。buflen は、バッファの長さです。Finit() は、バッファを指すポインタおよびバッファの長さを利用して、フィールドを持たないバッファ用に内部的な構造体を設定します。また、Finit() は以前使用したバッファを再初期化する場合にも使用します。

Finit32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Finit() または Finit32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Finit() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファのポインタが NULL です。

[FNOSPACE]
"no space in fielded buffer"
フィールド化バッファ用に指定されたバッファの大きさが小さすぎます。

使用例 フィールドを持たないようバッファを再初期化する、正しい方法を次に示します。`Finit(fbfr, (FLDLEN)Fsizeof(fbfr));`

関連項目 「FML 関数の紹介」、`Falloc`、`Falloc32(3fml)`、`Fneeded`、`Fneeded32(3fml)`、`Frealloc`、`Frealloc32(3fml)`

Fjoin、Fjoin32(3fml)

名前 Fjoin()、Fjoin32() ―ソース・バッファを宛先バッファに組み込む

形式

```
#include stdio.h>
#include "fml.h"
int
Fjoin(FBFR *dest, FBFR *src)
#include "fml32.h"
int
Fjoin32(FBFR32 *dest, FBFR32 *src)
```

機能説明 Fjoin() は、2 つのバッファをフィールド識別子 / オカレンスの整合に基づいて結合するために使用されます。dest は、宛先バッファを指すポインタです。src は、元のフィールド化バッファを指すポインタです。フィールド ID/ オカレンスと一致するフィールドの場合は、フィールド値はソース・バッファの値を使って宛先バッファで更新されます。宛先バッファのフィールドに、ソース・バッファ内の対応するフィールド識別子 / オカレンスがない場合、そのフィールドは削除されます。バッファを組み込むことにより FLD_PTR フィールドが削除されても、ポインタによって参照されるメモリ領域は変更または解放されません。

この関数は、新しい値が古い値より大きい場合に領域不足によって異常終了します。この場合、宛先バッファは変更されます。しかし、これが発生した場合でも、宛先バッファは、Frealloc() および Fjoin() を繰り返し使用することによって再割り当てできます。宛先バッファのある一部分が更新されている場合でも、Fjoin() 関数を繰り返し使用することにより正しい結果になります。

Fjoin32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fjoin() または Fjoin32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fjoin() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

使用例　次に使用例を示します。

```
FBFR *src, *dest;  ...  if(Fjoin(dest,src)  0)  F_error("pgm_name");
```

dest がフィールド A、B および C オカレンスを 2 個持っており、src が、フィールド A、C および D を持っている場合、結果的に、dest は、元のフィールド値 A および元のフィールド値 C を持つことになります。

関連項目　「FML 関数の紹介」、Fconcat、Fconcat32(3fml)、Fojoin、Fojoin32(3fml)、Fproj、Fproj32(3fml)、Fprojcpy、Fprojcpy32(3fml)、Frealloc、Frealloc32(3fml)

Fldid、Fldid32(3fml)

名前 Fldid()、Fldid32() ―フィールド名をフィールド識別子にマップする

形式

```
#include <stdio.h>
#include "fml.h"

FLDID
Fldid(char *name)

#include "fml32.h"

FLDID32
Fldid32(char *name)
```

機能説明 Fldid() は、フィールド名からフィールド識別子を導くランタイムな変換機能を提供し、そのフィールドの name パラメータに対応した FLDID を返します。最初の呼び出しでは、フィールド・テーブル用の領域が動的に割り当てられ、テーブルがロードされます。Fldid() でロードしたフィールド・テーブルが占有しているデータ領域を回復するには、Fnmid_unload() 関数を呼び出してファイルをアンロードする必要があります。

Fldid32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fldid() または Fldid32() を呼び出すことができます。

戻り値 この関数は、エラー時には BADFLDID を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fldid() は異常終了し、Ferror を次の値に設定します。

[FBADNAME]
"unknown field name"
フィールド・テーブルにないフィールド名が指定されました。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目　「FML 関数の紹介」、Fldno、Fldno32(3fml)、Fname、Fname32(3fml)、Fnmid_unload、Fnmid_unload32(3fml)

UNIX システムのリファレンス・ページの malloc(3)

Fldno、Fldno32(3fml)

名前 Fldno()、Fldno32() ―フィールド識別子をフィールド番号にマップする

```
#include <stdio.h>
#include "fml.h"

int
Fldno(FLDID fieldid)

#include "fml32.h"

long
Fldno32(FLDID32 fieldid)
```

機能説明 Fldno() は、パラメータとしてフィールド識別子 (fieldid) を受け取り、識別子にあるフィールド番号を返します。

Fldno32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fldno() または Fldno32() を呼び出すことができます。

戻り値 この関数は、フィールド番号を返します。エラーは返しません。

関連項目 「FML 関数の紹介」、Fldid、Fldid32(3fml)、Fldtype、Fldtype32(3fml)

Fldtype、Fldtype32(3fml)

名前 Fldtype()、Fldtype32() ―フィールド識別子をフィールド・タイプにマップする

形式

```
#include <stdio.h>
#include "fml.h"

int
Fldtype(FLDID fieldid)

#include "fml32.h"

int
Fldtype32(FLDID32 fieldid)
```

機能説明 Fldtype() は、フィールド識別子 (fieldid) を受け取り、fml.h で定義されているように、識別子 (整数) にあるフィールド・タイプを返します。

Fldtype32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fldtype() または Fldtype32() を呼び出すことができます。

戻り値 この関数は、フィールド・タイプを返します。

関連項目 「FML 関数の紹介」、Fldid、Fldid32(3fml)、Fldno、Fldno32(3fml)

Flen、Flen32(3fml)

名前 Flen()、Flen32() ―バッファ中のフィールド・オカレンスの長さを返す

形式

```
#include <stdio.h>
#include "fml.h"

int
Flen(FBFR *fbfr, FLDID fieldid, FLDOCC oc)

#include "fml32.h"

long
Flen32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Flen() は、バッファ中の指定したフィールド・オカレンスの値を見つけ出し、そのフィールド長を返します。*fbfr* は、フィールド化バッファを指すポインタです。*fieldid* は、フィールド識別子です。*oc* はフィールドのオカレンス番号です。

FLD_PTR 型の値の場合、Flen32() は sizeof(char*) に基づくポインタ・フィールドの固定長を返します。FLD_FML32 型の値の場合、Flen32() は入れ子になったバッファの長さに対する Fused32() の値を返します。FLD_VIEW32 型の値の場合、Flen32() は VIEW データの長さと VIEW 名の長さを返します。

FLD_MBSTRING 型の値の場合、Flen32() は Fmbpack32() で作成され、パッキングされた出力の長さを返します。

Flen32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Flen() または Flen32() を呼び出すことができます。

戻り値 Flen() は、正しく終了するとフィールド長を返します。

この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件が発生すると、Flen() は異常終了し、Ferror を次のように設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOTPRES]
"field not present"
フィールド・オカレンスが要求されましたが、指定されたフィールドとオカレンスの両方、あるいはどちらかは、フィールド化バッファにありませんでした。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目　「FML 関数の紹介」、Fnum、Fnum32(3fml)、Fpres、Fpres32(3fml)

Fmbpack32(3fml)

名前 Fmbpack32() ―符号化名およびマルチバイト・データ情報を準備する

形式

```
#include "fml32.h"

int
Fmbpack32 (char *enc,void *ind,FLDLEN32 indlen,void
*packed,FLDLEN32 *size,long flags)
```

機能説明 Fmbpack32() は、FML32 型付きバッファに入力された FLD_MBSTRING フィールドの符号化名およびマルチバイト・データ情報を準備します。Fmbpack32() は、FLD_MBSTRING フィールドが FML32 API で FML32 バッファに追加される前に使用します。

enc は、NULL でない場合、コード・セットのマルチバイト・データ (*ind*) に対するコード・セットの符号化名を保持する ASCII 文字列 (NULL で終了) です。*enc* が NULL、引数 *flags* が 0 の場合、出力 *packed* に含められる符号化名は、プロセス TPMBENC 環境変数から取得されます。引数 *flags* が FBUFENC の場合、*enc* は無視されます。

ind は、コード・セットのマルチバイト・データです。

indlen は、*ind* のバイト数です。

packed は、Fmbpack32() の出力を指すポインタです。これは、FML32 バッファに FLD_MBSTRING フィールドを含める FML32 API の入力値として使用されます。パック領域は、FLDLEN32(TM32U) バウンダリに配置する必要があります。

入力での *size* は、*packed* が指すメモリ・サイズです。サイズが Fmbpack32() の結果の処理には不十分な場合、FNOSPACE が返され、*size* は *packed* のバイト数にリセットされます。Fmbpack32() が正常に実行された後、*size* は実際に使用されたバイト数にリセットされます。

flags は、0 または FBUFENC です。*flags* を FBUFENC に設定すると、Fmbpack32() は引数 *enc* を無視し、FBUFENC を *packed* の入力データと共に含めます。符号化名を指定しない場合、出力 *packed* は、FML32 API によって強制的に FLD_MBSTRING フィールドで符号化名を FML32 バッファから取得

するよう処理します。したがって、FBUFENC を使用する場合、FML32 バッファの符号化名を設定するために、アプリケーション開発者は tpsetmbenc() も使用する必要があります。

戻り値 Fmbpack32() は、正しく終了すると正の値を返します。Fmbpack32() は、エラー発生時に -1 を返し、Ferror32 を設定してエラー条件を示します。

エラー 次の条件の場合、Fmbpack32() は異常終了し、Ferror32 を次の値に設定します。

[FEINVAL]
ind、*packed*、または *size* が NULL です。*enc* または *indlen* が無効です。

[FNOSPACE]
packed のサイズが、Fmbpack32() の結果を処理するには不十分です。

関連項目 Fmbunpack32(3fml)、tpconvfmb32(3fml)、tpsetmbenc(3c)、tuxgetmbenc(3c)、tuxsetmbenc(3c)

Fmbunpack32(3fml)

名前 Fmbunpack32() ―符号化名およびマルチバイト・データ情報を抽出する

形式

```
#include "fml32.h"

int
Fmbunpack32 (void *packed,FLDLEN32 ilen,char *enc,void
*outd,FLDLEN32 *olen,long flags)
```

機能説明 Fmbunpack32() は、FML32 型付きバッファ内の FLD_MBSTRING フィールドから符号化名およびマルチバイト・データ情報を抽出します。Fmbunpack32() は、FLD_MBSTRING フィールドが FML32 API (Ffind32()、Fget32()、など) で FML32 バッファから抽出された後に使用します。

packed は、FML32 API で出力された FLD_MBSTRING フィールドのデータを指すポインタです。

ilen は、*packed* のバイト数です。

enc は、FLD_MBSTRING フィールドのコード・セットの符号化名が、パッキングされた情報 (*packed*) の一部である場合に、符号化名を保持する *packed* 内の ASCII 文字列 (NULL で終了) です。FBUFENC フラグを設定して Fmbpack32() で FLD_MBSTRING フィールドを作成した場合、*enc* は NULL に設定します。後者の場合、アプリケーション開発者は、FML32 バッファで tpgetmbenc() を使用して、FLD_MBSTRING フィールドの符号化名を取得する必要があります。

outd には、Fmbunpack32() の正常終了時に *packed* から抽出されるマルチバイト・データが入ります。

入力での *olen* は、*outd* が指すメモリ・サイズです。サイズが Fmbunpack32() の結果の処理には不十分な場合、FNOSPACE が返され、*olen* は *outd* のバイト数にリセットされます。Fmbunpack32() が正常に実行された後、*olen* は実際に使用されたバイト数にリセットされます。

flags は現在使用されていないので、0 に設定します。

戻り値 Fmbunpack32() は、正しく終了すると正の値を返します。Fmbunpack32() は、エラー発生時に -1 を返し、Ferror32 を設定してエラー条件を示します。

エラー 次の条件の場合、Fmbunpack32() は異常終了し、Ferror32 を次の値に設定します。

[FEINVAL]
outd、*olen*、または *packed* が NULL です。*packed* または *ilen* が無効です。

[FNOSPACE]
outd のサイズが、Fmbunpack32() の結果を処理するには不十分です。

関連項目 Fmbpack32(3fml)、tpconvfmb32(3fml)、tpgetmbenc(3c)、tuxgetmbenc(3c)、tuxsetmbenc(3c)

Fmkfldid、Fmkfldid32(3fml)

名前 Fmkfldid()、Fmkfldid32() ―フィールド識別子を作成

```
#include <stdio.h>
#include "fml.h"

FLDID
Fmkfldid(int type, FLDID num)

#include "fml.h"

FLDID32
Fmkfldid32(int type, FLDID32 num)
```

機能説明 Fmkfldid() によって、有効な型 (fml.h で定義されている) から有効なフィールド識別子を作成できます。フィールド番号を順次選択するアプリケーション・ジェネレータをコーディングする場合、あるいはフィールド識別子を再作成する場合に役立ちます。

type は、有効なフィールド・タイプです (整数。Fldtype、Fldtype32(3fml) を参照)。num は、フィールド番号です (現在あるフィールドとの混同をさけるために未使用のフィールド番号でなければならない)。

Fmkfldid32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fmkfldid() または Fmkfldid32() を呼び出すことができます。

戻り値 この関数は、エラー時には BADFLDID を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fmkfldid() は異常終了し、Ferror を次の値に設定します。

[FBADFLD]
"unknown field number or type"
指定されたフィールド番号は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド・タイプは、無効です (fml.h で定義されている)。

関連項目 「FML 関数の紹介」、Fldtype、Fldtype32(3fml)

Fmove、Fmove32(3fml)

名前 Fmove()、Fmove32() ―フィールド化バッファを宛先バッファに移動

形式

```
#include <stdio.h>
#include "fml.h"

int
Fmove(char *dest, FBFR *src)

#include "fml32.h"

int
Fmove32(char *dest, FBFR32 *src)
```

機能説明 Fmove() は、フィールド化バッファから任意の型のバッファに複写するときに使用されます。dest は、宛先バッファを指すポインタです。src は、元のフィールド化バッファを指すポインタです。

Fmove() と Fcpy() の違いとして、Fcpy() は、宛先バッファをフィールド化バッファにする点です。これにより、フィールド化バッファが、元のバッファからのデータを入れるために十分な大きさになります。Fmove() は、このような検査は行いません。何の検査もせずに、ソース・フィールド化バッファの Fsizeof() バイトのデータをターゲット・バッファに移動します。宛先バッファは、short バウンダリで並んでいる必要があります。

FLD_PTR 型の値の場合、Fmove32() はバッファ・ポインタを渡します。アプリケーション・プログラマは、対応するポインタの移動に応じてバッファの再割り当て、および解放を管理する必要があります。

Fmove32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fmove() または Fmove32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fmove() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファがフィールド化バッファでないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Fcpy、Fcpy32(3fml)、Fsizeof、Fsizeof32(3fml)

Fname、Fname32(3fml)

名前 Fname()、Fname32() ―フィールド識別子をフィールド名にマップする

形式

```
#include <stdio.h>
#include "fml.h"

char *
Fname(FLDID fieldid)

#include "fml32.h"

char *
Fname32(FLDID32 fieldid)
```

機能説明 Fname() は、フィールド識別子 (fieldid) からフィールド名を導くランタイムな変換機能を提供し、引数に対応した名前を持つ文字列を指すポインタを返します。最初の呼び出しでは、フィールド・テーブル用の領域が動的に割り当てられ、テーブルがロードされます。Fname() で作成したマッピング・テーブルによって使用されているテーブル領域は、Fidnm_unload() 関数を呼び出して回復することができます。

Fname32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fname() または Fname32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fname() は異常終了し、Ferror を次の値に設定します。

[FBADFLD]
"unknown field number or type"
フィールド名が不明または無効な場合、フィールド番号が指定されます。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目　「FML 関数の紹介」、Ffprint、Ffprint32(3fml)、Fidnm_unload、Fidnm_unload32(3fml)、Fldid、Fldid32(3fml)、Fprint、Fprint32(3fml)

Fneeded、Fneeded32(3fml)

名前 Fneeded()、Fneeded32() ―バッファに必要な大きさの計算

形式

```
#include <stdio.h>
#include "fml.h"

long
Fneeded(FLDOCC F, FLDLEN V)

#include "fml32.h"

long
Fneeded32(FLDOCC32 F, FLDLEN32 V)
```

機能説明 Fneeded() は、フィールド化バッファに割り当てなければならない領域を決める場合に使用されます。引数 F はフィールドの数、引数 V はすべてのフィールド値に必要な領域のバイト数です。

Fneeded32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fneeded() または Fneeded32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fneeded() は異常終了し、Ferror を次の値に設定します。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、フィールド数が 0 より小さい、V が 0、または合計の大きさが 65534 より大きい)。

関連項目 「FML 関数の紹介」、Falloc、Falloc32(3fml)、Fielded、Fielded32(3fml)、Finit、Finit32(3fml)、Fsizeof、Fsizeof32(3fml)、Funused、Funused32(3fml)、Fused、Fused32(3fml)

Fnext、Fnext32(3fml)

名前 Fnext()、Fnext32() ―次のフィールド・オカレンスの取得

形式

```
#include <stdio.h>
#include "fml.h"

int
Fnext(FBFR *fbfr, FLDID *fieldid, FLDOCC *oc, char *value, FLDLEN
*len)

#include "fml32.h"

int
Fnext32(FBFR32 *fbfr, FLDID32 *fieldid, FLDOCC32 *oc, char *value,
FLDLEN32 *len)
```

機能説明 Fnext() は、指定されたフィールド・オカレンスの後、バッファ内の次のフィールドを検索します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子を指すポインタです。oc は、フィールドのオカレンス番号を指すポインタです。*value* は、次のフィールドの値を指すポインタです。len は、次の値の長さです。

フィールド識別子 FIRSTFLDID は、バッファにある最初のフィールドを取得するために指定する必要があります (たとえば、Fnext() の最初の呼び出しにおいて)。value が NULL でない場合、次のフィールド値は、value に複写されます。*len は、値を入れるために割り当てられる十分な領域がバッファにあるかを判断するために使用されます。値の長さは、*len に返されます。関数が呼び出されたときに len が NULL の場合は、十分な領域があるために、新しい値の長さが返されないと想定されます。value が NULL の場合、値は検索されず、fieldid および oc のみ更新されます。*fieldid パラメータは、次に検索されるフィールドに設定されます。*oc パラメータは、次のオカレンスに設定されます。検索するフィールドがこれ以上ない場合、0 (バッファの終わり) が返されます。*fieldid、*oc、および *value は変更されません。フィールドは、フィールド識別子の順に返されます。

値の型は、char * ですが、返される値は、検索される次のフィールドと同じ型になります。

検索されるフィールドのタイプが FLD_VIEW32 である場合、*value* パラメータは FVIEWFLD 構造体を指します。Fnext() 関数を使用して、構造体の *vname* フィールドと *data* フィールドを設定します。

Fnext32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fnext() または Fnext32() を呼び出すことができます。

戻り値 Fnext() は、次のオカレンスを正常に検索できた場合、1 を返します。バッファの終わりに達した場合は、0 を返します。

この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fnext() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space"
len で指定されている値の大きさは、フィールド値を保持する十分な大きさではありません。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、fieldid または *oc* に NULL を指定している)。

関連項目 「FML 関数の紹介」、Fget、Fget32(3fml)、Fnum、Fnum32(3fml)

Fnmid_unload、Fnmid_unload32(3fml)

名前　Fnmid_unload()、Fnmid_unload32() — *nm->id* マッピング・テーブルから領域を取り戻す

形式

```
#include <stdio.h>
#include "fml.h"
void Fnmid_unload(void)
#include "fml32.h"
void Fnmid_unload32(void)
```

機能説明

Fldid() でロードしたフィールド・テーブルが占有しているデータ領域を回復するには、Fnmid_unload() 関数を呼び出してファイルをアンロードする必要があります。

Fnmid_unload32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fnmid_unload() または Fnmid_unload32() を呼び出すことができます。

戻り値　この関数は、void 宣言のため何も返しません。

関連項目　「FML 関数の紹介」、Fidnm_unload、Fidnm_unload32(3fml)、Fldid、Fldid32(3fml)

Fnum、Fnum32(3fml)

名前 Fnum()、Fnum32() ―バッファにあるすべてのオカレンスの数を返す

形式

```
#include <stdio.h>
#include "fml.h"

FLDOCC
Fnum(FBFR *fbfr)

#include "fml32.h"

FLDOCC32
Fnum32(FBFR *fbfr)
```

機能説明 Fnum() は、指定されたバッファにあるフィールドの数を返します。fbfr は、フィールド化バッファを指すポインタです。FLD_FML32 フィールドおよび FLD_VIEW32 フィールドは、含まれるフィールドの数に関係なく、単一のフィールドとしてカウントされます。

Fnum32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fnum() または Fnum32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fnum() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Foccur、Foccur32(3fml)、Fpres、Fpres32(3fml)

Foccur、Foccur32(3fml)

名前 Foccur()、Foccur32() ―バッファにあるフィールド・オカレンスの数を返す

形式

```
#include <stdio.h>
#include "fml.h"

FLDOCC
Foccur(FBFR *fbfr, FLDID fieldid)

#include "fml32.h"

FLDOCC32 Foccur32(FBFR32 *fbfr, FLDID32 fieldid)
```

機能説明 Foccur() は、*fbfr* で参照されるバッファの *fieldid* で指定されるフィールド・オカレンスの数を決めるために使用されます。FLD_FML32 フィールド・タイプであるため、埋め込み型の FML32 バッファ内のフィールドのオカレンスはカウントされません。

Foccur32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Foccur() または Foccur32() を呼び出すことができます。

戻り値 正常終了の場合、Foccur() は、オカレンスの数を返します。オカレンスがない場合は、0 を返します。

この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Foccur() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fnum、Fnum32(3fml)、Fpres、Fpres32(3fml)

Fojoin、Fojoin32(3fml)

名前 Fojoin()、Fojoin32() ―ソース・バッファを宛先バッファに組み込む

```
#include <stdio.h>
#include "fml.h"

int
Fojoin(FBFR *dest, FBFR *src)

#include "fml32.h"

int
Fojoin32(FBFR32 *dest, FBFR32 *src)
```

機能説明 Fojoin() は、Fjoin() に似ていますが、宛先バッファ dest のフィールドに、ソース・バッファ src に対応するフィールド識別子 / オカレンスがない場合でも、そのフィールドは削除されません。対応するフィールド識別子 / オカレンスを宛先バッファに持たないソース・バッファにあるフィールドは、宛先バッファに追加されません。バッファを組み込むことにより FLD_PTR フィールドが削除されても、ポインタによって参照されるメモリ領域は変更または解放されません。

Fjoin() と同様、この関数は領域不足により異常終了し、さらに領域を割り当てた後、操作を完了するために再起動されます。

Fojoin32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fojoin() または Fojoin32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fojoin() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

使用例 次に使用例を示します。

```
if(Fojoin(dest,src) 0)
  F_error("pgm_name");
```

dest が、フィールド A、B および C のオカレンスを 2 個持っており、src が、フィールド A、C および D を持っている場合、結果的に、dest は、ソース・フィールド値 A、宛先フィールド値 B、ソース・フィールド値 C、および 2 番目の宛先フィールド値 C を含んでいます。

関連項目 「FML 関数の紹介」、Fconcat、Fconcat32(3fml)、Fjoin、Fjoin32(3fml)、Fproj、Fproj32(3fml)

Fpres、Fpres32(3fml)

名前 Fpres()、Fpres32() ―フィールド・オカレンスがバッファにある場合は true を返す

```
#include <stdio.h>

#include "fml.h"

int
Fpres(FBFR *fbfr, FLDID fieldid, FLDOCC oc)

#include "fml32.h"

int
Fpres32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Fpres() は、指定されたフィールド (*fieldid*) のオカレンス (*oc*) が、*fbfr* で参照されるバッファにあるかを検出するために使用されます。Fpres32() は、指定されたフィールドのオカレンスが、埋め込み型のバッファにあるかチェックしません。埋め込み型のバッファ内のフィールドは FLD_FML32 フィールド・タイプです。

Fpres32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fpres() または Fpres32() を呼び出すことができます。

戻り値 Fpres() は、指定されたオカレンスがある場合は true を、そうでない場合は、false を返します。

関連項目 「FML 関数の紹介」、Ffind、Ffind32(3fml)、Fnum、Fnum32(3fml)、Foccur、Foccur32(3fml)

Fprint、Fprint32(3fml)

名前 Fprint()、Fprint32() ―標準出力にバッファを出力する

形式

```
#include <stdio.h>
#include "fml.h"

int
Fprint(FBFR *fbfr)

#include "fml32.h"

int
Fprint32(FBFR32 *fbfr)
```

機能説明 Fprint() は、標準出力に指定されたバッファを出力します。fbfr は、フィールド化バッファを指すポインタです。バッファの各フィールドでは、フィールド名およびタブ区切りのフィールド値を出力します。Fname() が、フィールド名を決めるために使用されます。フィールド名を決めることができない場合は、フィールド識別子が出力されます。文字列内の出力できない文字および文字配列のフィールド値は、2 文字の 16 進数 (バックスラッシュの後) で表されます。改行は、出力されるバッファの後に出力されます。

FLD_PTR 型の値の場合、Fprint32() はフィールド名またはフィールド識別子、およびポインタ値を 16 進数で出力します。この関数はポインタ情報を出力しますが、Fextread32() 関数は FLD_PTR フィールド・タイプを無視します。

FLD_FML32 型の値の場合、Fprint32() は各入れ子レベルに先行タブを追加して、再帰的に FML32 バッファを出力します。FLD_VIEW32 型の値の場合、Fprint32() は VIEW32 フィールド名、および構造体メンバの名前と値のペアを出力します。

Fprint32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fprint() または Fprint32() を呼び出すことができます。

戻り値　この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Fprint() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FMALLOC]
"malloc failed"
malloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目　「FML 関数の紹介」、Fextread、Fextread32(3fml)、Ffprint、Ffprint32(3fml)、Fname、Fname32(3fml)

Fproj、Fproj32(3fml)

名前 Fproj()、Fproj32() ―バッファのプロジェクション

形式

```
#include <stdio.h>
#include "fml.h"

int
Fproj(FBFR *fbfr, FLDID *fieldid)

#include "fml32.h"

int
Fproj32(FBFR32 *fbfr, FLDID32 *fieldid)
```

機能説明 Fproj() は、必要なフィールドのみを維持する目的から、バッファを更新するために使用されます。fbfr は、フィールド化バッファを指すポインタです。必要なフィールドは、fieldid で指されるフィールド識別子の配列で指定されます。配列の最後のエントリは、BADFLDID でなければなりません。更新は、その場所で行われます。したがって、投影の結果にないフィールドは、フィールド化バッファから削除されます。フィールド識別子の配列は、再配列される場合があります。それらが番号順でない場合、ソートされます。バッファを更新することにより FLD_PTR フィールドが削除されても、ポインタによって参照されるメモリ領域は変更または解放されません。

Fproj32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fproj() または Fproj32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fproj() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

使用例

```
#include "fld.tbl.h"
FBFR *fbfr;
FLDID fieldid[20];
...
fieldid[0] = A;          /* フィールド A のフィールド識別子 */
fieldid[1] = D;          /* フィールド D のフィールド識別子 */
fieldid[2] = BADFLDID;   /* 標識値 */
...
if(Fproj(fbfr, fieldid)  0)
    F_error("pgm_name");
```

バッファが、フィールド A、B、C および D を持っている場合、使用例の結果、バッファは、フィールド A および D のオカレンスのみ持つようになります。フィールド識別子の配列のエントリは、特定の順番になっている必要はありませんが、フィールド識別子の配列の最後の値は、フィールド識別子 0 (BADFLDID) である必要があります。

関連項目

「FML 関数の紹介」、Fjoin、Fjoin32(3fml)、Fojoin、Fojoin32(3fml)、Fprojcpy、Fprojcpy32(3fml)

Fprojcpy、Fprojcpy32(3fml)

名前 Fprojcpy()、Fprojcpy32() ―バッファのプロジェクションおよびコピー

形式

```
#include <stdio.h>
#include "fml.h"

int
Fprojcpy(FBFR *dest, FBFR *src, FLDID *fieldid)

#include "fml32.h"

int
Fprojcpy32(FBFR32 *dest, FBFR32 *src, FLDID32 *fieldid)
```

機能説明 Fprojcpy() は、Fproj() と類似していますが、プロジェクションはその場所で行われず、宛先バッファで行われます。dest は宛先バッファのポインタです。src は、元のフィールド化バッファを指すポインタです。fieldid は、フィールド識別子の配列を指すポインタです。宛先バッファのすべてのフィールドは、最初に削除され、ソース・バッファに投影した結果は宛先バッファに組み込まれます。ソース・バッファは変更されません。フィールド識別子の配列は、再配列される場合があります。それらが番号順でない場合、ソートされます。バッファを更新することにより FLD_PTR フィールドが削除されても、ポインタによって参照されるメモリ領域は変更または解放されません。

この関数は、領域が不足した場合に異常終了します。この関数は、操作を完了するために付加領域を十分に割り当てた後、再起動されます。

Fprojcpy32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fprojcpy() または Fprojcpy32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fprojcpy() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファのどちらかがフィールド化バッファでないか、または Finit() によって初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、変換先のフィールド化バッファにコピーされますが、バッファには十分な領域が残っていません。

関連項目 「FML 関数の紹介」、Fjoin、Fjoin32(3fml)、Fojoin、Fojoin32(3fml)、Fproj、Fproj32(3fml)

Fread、Fread32(3fml)

名前 Fread()、Fread32() ―フィールド化バッファの読み取り

形式

```
#include <stdio.h>
#include "fml.h"

int
Fread(FBFR *fbfr, FILE *iop)

#include "fml32.h"

int
Fread32(FBFR32 *fbfr, FILE32 *iop)
```

機能説明 フィールド化バッファは、Fread() を使用して、ファイル・ストリームから読み取られます。fbfr は、フィールド化バッファを指すポインタです。iop は、FILE 型の入力ストリームを指すポインタです。ストリームについては、UNIX System V『プログラマ・リファレンス・マニュアル』の stdio(3S) を参照してください。Fread() は、フィールド化バッファをストリームから *fbfr* に読み取り、以前に格納されているバッファのデータをクリアしてバッファのインデックスを再作成します。FLD_PTR フィールド・タイプは、Fread32() では無視されます。関数にタイプ FLD_PTR の値が与えられても、エラーは返されません。

Fread32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fread() または Fread32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fread() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。また、このエラーは、読み取られたデータがフィールド化バッファでない場合に返されます。

[FNOSPACE]
"no space in fielded buffer"
ストリームから読み取られるフィールド化バッファを保持するための十分な領域が、バッファにありません。

[FEUNIX]
"UNIX system call error"
read() システム・コールが異常終了しました。外部変数の整数型データ errno は、システム・コールによるエラーを示すために設定される必要があります。

移植性 この関数は、Windows 用の BEA Tuxedo System Workstation DLL では使用できません。

関連項目 「FML 関数の紹介」、Findex、Findex32(3fml)、Fwrite、Fwrite32(3fml)

UNIX システムのリファレンス・ページの stdio(3S)

Frealloc、Frealloc32(3fml)

名前 Frealloc()、Frealloc32() ―フィールド化バッファの再割り当て

形式

```
#include <stdio.h>
#include "fml.h"

FBFR *
Frealloc(FBFR *fbfr, FLDOCC nf, FLDLEN nv)

#include "fml32.h"

FBFR32 *
Frealloc32(FBFR32 *fbfr, FLDOCC32 nf, FLDLEN32 nv)
```

機能説明 Frealloc() は、フィールド化バッファを拡大するための領域の再割り当てに使用されます。fbfr は、フィールド化バッファを指すポインタです。2 番目のパラメータ nf は、新しいフィールドの数です。3 番目のパラメータ nv は、新しい値の領域のバイト数です。これらは、増分ではありません。

Frealloc32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Frealloc() または Frealloc32() を呼び出すことができます。

戻り値 正常終了の場合、Frealloc() は、再割り当てされた FBFR を指すポインタを返します。

この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Frealloc() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、フィールド数が 0 より小さい、v が 0、または合計の大きさが 65534 より大きい)。

[FMALLOC]
"malloc failed"
新しい大きさが、現在バッファにあるものより小さいか、または realloc() を使用しての領域の動的な割り当てが失敗しました。

関連項目 「FML 関数の紹介」、Falloc、Falloc32(3fml)、Ffree、Ffree32(3fml)

Frstrindex、Frstrindex32(3fml)

名前 Frstrindex()、Frstrindex32() ―バッファにあるインデックスをリストアする

形式

```
#include <stdio.h>
#include "fml.h"

int
Frstrindex(FBFR *fbfr, FLDOCC numidx)

#include "fml32.h"

int
Frstrindex32(FBFR32 *fbfr, FLDOCC32 numidx)
```

機能説明 インデックス解除されたフィールド化バッファは、Findex() または Frstrindx() を呼び出して再インデックスされます。fbfr は、フィールド化バッファを指すポインタです。Findex() は、バッファのすべてのインデックスを計算するため、かなり効率が悪くなります (バッファをすべて走査することが必要になる)。この関数は、インデックスされていないバッファが変更された場合、またはバッファの前の状態がわからないとき (たとえば、バッファがあるプロセスから他のプロセスへインデックスなしで送られた場合) に使用する必要があります。Frstrindex() は、非常に高速ですが、バッファが、前のインデックス解除操作以降、変更されていない場合のみ使用されます。Frstrindx() の 2 番目の引数 numidx は、Funindex() 関数の戻り値です。

Frstrindex32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Frstrindex() または Frstrindex32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Frstrindex() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

使用例 バッファをインデックスなしで送信するためには、次のような文が実行されます。

```
save = Funindex(fbfr);
num_to_send = Fused(fbfr);
transmit(fbfr,num_to_send);         /* 関数プロトタイプ */
Frstrindx(fbfr,save);
```

これら 4 つの文は、次のことを実行します。

```
1. - /* インデックスを解除し、Frstrindx を保存する */
2. - /* 送信するバイト数を決める */
3. - /* インデックスなしで fbfr を送信する */
4. - /* インデックスをリストアする */
```

この場合、transmit() には、メモリ・ポインタおよび長さが渡されます。送信されるデータは、メモリ・ポインタで始まり、num_to_send バイトです。いったんバッファが送信されると、そのインデックスは、Frstrindex() を使用してリストアされます (どのような場合でも、transmit() が、インデックスを変更しないことが前提となっている)。送信の終わりを受信すると、フィールド化バッファを受け取ったプロセスは、Findex() を使用してそのバッファを次に示す方法でインデックスします。

```
receive(fbfr); /* 任意の場所から fbfr を取得する */
Findex(fbfr);  /* fbfr をインデックスする */
```

受信プロセスは、次の理由により Frstrindx() を呼び出すことができません。

1. 受信プロセスは、Funindex() を呼び出していません。したがって、Frstrindex() の引数 numidx の値がわかりません。
2. インデックス自体、送信されないため利用できません。

解決策としては、Findex() を明示的に呼び出すことです。もちろんユーザは、フィールド化バッファのインデックスのバージョンを自由に送信する (つまり、Fsizeof(*fbfr) バイトを送信する) ことができ、受信側の Findex() のコストを回避します。

関連項目 「FML 関数の紹介」、Findex、Findex32(3fml)、Fsizeof、Fsizeof32(3fml)、Funindex、Funindex32(3fml)

Fsizeof、Fsizeof32(3fml)

名前 Fsizeof()、Fsizeof32() ―フィールド化バッファのサイズを返す

形式

```
#include <stdio.h>
#include "fml.h"

long
Fsizeof(FBFR *fbfr)

#include "fml32.h"

long
Fsizeof32(FBFR32 *fbfr)
```

機能説明 Fsizeof() は、フィールド化バッファのサイズをバイト数で返します。fbfr は、フィールド化バッファを指すポインタです。

Fsizeof32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fsizeof() または Fsizeof32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fsizeof() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Fidxused、Fidxused32(3fml)、Fused、Fused32(3fml)、Funused、Funused32(3fml)

Fstrerror、Fstrerror32(3fml)

名前 Fstrerror()、Fstrerror32() ― FML エラーのエラー・メッセージ文字列の取得

形式

```
#include <fml.h>

char *
Fstrerror(int err)

#include <fml32.h>

char *
Fstrerror32(int err)
```

機能説明 Fstrerror() は、LIBFML_CAT からエラー・メッセージのテキストを検索するために使用します。err は、FML 関数呼び出しが -1 またはその他のエラー値を返したときに F_error に設定されるエラー・コードです。

Fstrerror() が返したポインタを、userlog() または F_error への引数として使用できます。

Fstrerror32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fstrerror() または Fstrerror32() を呼び出すことができます。

戻り値 err が無効なエラー・コードの場合、Fstrerror() は、NULL を返します。正常終了した場合は、この関数は、エラー・メッセージのテキストを持つ文字列を指すポインタを返します。

エラー Fstrerror() は、エラーが発生すると NULL を返します。この場合は、F_error の設定をしません。

関連項目 「FML 関数の紹介」、tpstrerror(3c)、userlog(3c)、F_error、F_error32(3fml)

Ftypcvt、Ftypcvt32(3fml)

名前 Ftypcvt()、Ftypcvt32() ―フィールド・タイプを別のフィールド・タイプに変換

形式

```
#include <stdio.h>
#include "fml.h"

char *
Ftypcvt(FLDLEN *tolen, int totype, char *fromval, int fromtype,
  FLDLEN fromlen)

#include "fml32.h"

char *
Ftypcvt32(FLDLEN32 *tolen, int totype, char *fromval, int fromtype,
  FLDLEN32 fromlen)
```

機能説明 Ftypcvt() は、値 *fromval を型 totype に変換します。値 *fromval は、型 fromtype および長さ fromlen を持ちます (fromtype が FLD_CARRAY の場合。それ以外の場合は fromlen が fromtype から推定される)。Ftypcvt() は、正常終了時、変換した値にポインタを返し、*tolen を変換された長さに設定します。異常終了時は Ftypcvt() は NULL を返します。

Ftypcvt32() は、FLD_PTR、FLD_MBSTRING、FLD_FML32、または FLD_VIEW32 のフィールド・タイプが使用されると異常終了します。これらのフィールド・タイプの 1 つが指定されると、Ferror に FEBADOP が設定されます。

Ftypcvt32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Ftypcvt() または Ftypcvt32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ftypcvt() は異常終了し、Ferror を次の値に設定します。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) から文字列に変換するときに、malloc() を使用しての領域の動的な割り当てが失敗しました。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、*tolen* または *fromval* パラメータに NULL が指定された場合)。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
指定されたフィールド・タイプは無効です (FLD_PTR、FLD_FML32、または FLD_VIEW32 など)。

関連項目 「FML 関数の紹介」、CFadd、CFadd32(3fml)、CFchg、CFchg32(3fml)、CFfind、CFfind32(3fml)、CFget、CFget32(3fml)、CFgetalloc、CFgetalloc32(3fml)

Ftype、Ftype32(3fml)

名前 Ftype()、Ftype32() ―フィールド・タイプを指すポインタを返す

形式

```
#include <stdio.h>
#include "fml.h"

char *
Ftype(FLDID fieldid)

#include "fml32.h"

char *
Ftype32(FLDID32 fieldid)
```

機能説明 Ftype() は、フィールド識別子 fieldid を受け取り、フィールドの型名を持つ文字列を指すポインタを返します。たとえば、Ftype() に short 型の FLDID が与えられた場合は、文字列 “short” を指すポインタが返されます。このデータ領域は読み取り専用です。

Ftype32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Ftype() または Ftype32() を呼び出すことができます。

戻り値 Ftype() は、正常終了時にフィールド・タイプを識別する文字列を指すポインタを返します。

この関数は、エラー発生時に NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Ftype() は異常終了し、Ferror を次の値に設定します。

[FTYPERR]
"invalid field type"
指定されたフィールド識別子は無効です。

関連項目 「FML 関数の紹介」、Fldid、Fldid32(3fml)、Fldno、Fldno32(3fml)

Funindex、Funindex32(3fml)

名前 Funindex()、Funindex32() ―フィールド化バッファのインデックスの破棄

形式

```
#include <stdio.h>
#include "fml.h"

FLDOCC
Funindex(FBFR *fbfr)

#include "fml32.h"

FLDOCC32
Funindex32(FBFR32 *fbfr)
```

機能説明 Funindex() は、フィールド化バッファのインデックスを破棄します。fbfr はフィールド化バッファを指すポインタです。この関数が正常終了して戻る場合は、バッファはインデックスなしになります。結果として、インデックスにバッファ領域は割り当てられず、ユーザ・フィールドとして利用可能な領域が増加します。ただし、アクセス時間が遅くなる可能性があります。ディスクに格納する場合、または別の場所に転送する場合は、バッファをインデックスなしにすることが有効です。ディスク格納の場合、ディスク領域が節約され、転送を行う場合、転送コストを減少させることができます。

バッファがインデックスなしになった後、バッファ開始からの上位バイトの数は関数呼び出し Fused(fbfr) によって決定されます。

Funindex32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Funindex() または Funindex32() を呼び出すことができます。

戻り値 Funindex() は、インデックスが取り除かれる前にバッファが持つインデックスの要素数を返します。

この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Funindex() は、異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目 「FML 関数の紹介」、Findex、Findex32(3fml)、Frstrindex、Frstrindex32(3fml)、Fsizeof、Fsizeof32(3fml)、Funused、Funused32(3fml)

Funused、Funused32(3fml)

名前 Funused()、Funused32() ―フィールド化バッファの未使用分のバイト数を返す

形式

```
#include <stdio.h>
#include "fml.h"

long
Funused(FBFR *fbfr)

#include "fml32.h"

long
Funused32(FBFR32 *fbfr)
```

機能説明 Funused() は、現在使用していないバッファ領域の量を返します。ユーザ・データまたはヘッダやインデックスといった、オーバヘッド・データを含まない領域は使用されていません。

fbfr は、フィールド化バッファを指すポインタです。

Funused32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Funused() または Funused32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Funused() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目　「FML 関数の紹介」、Fidxused、Fidxused32(3fml)、Fused、Fused32(3fml)

Fupdate、Fupdate32(3fml)

名前 Fupdate()、Fupdate32() ―ソース・バッファで宛先バッファを更新する

形式

```
#include <stdio.h>
#include "fml.h"

int
Fupdate(FBFR *dest, FBFR *src)

#include "fml32.h"

int
Fupdate32(FBFR32 *dest, FBFR32 *src)
```

機能説明 Fupdate() は、ソース・バッファのフィールド値で宛先バッファを更新します。dest および src は、フィールド化バッファを指すポインタです。フィールド ID/ オカレンスと一致するフィールドの場合は、フィールド値はソース・バッファの値を使って宛先バッファで更新されます。ソース・バッファに、一致するフィールドを持たない宛先バッファのフィールドはそのままです。宛先バッファに、一致するフィールドを持たないソース・バッファのフィールドが、宛先バッファに追加されます。

FLD_PTR 型の値の場合、Fupdate32() はポインタの値を格納します。FLD_PTR フィールドが指すバッファを割り当てるには、tpalloc() を呼び出します。FLD_FML32 型の値の場合、Fupdate32() はインデックスを除いた FLD_FML32 フィールド全体の値を格納します。FLD_VIEW32 型の値の場合、Fupdate32() は FVIEWFLD 型の構造体を指すポインタを格納します。これには、*vflags* (現在未使用で 0 に設定された flags フィールド)、*vname* (VIEW 名を含む文字配列)、および *data* (C 構造体として格納される VIEW データを指すポインタ) が含まれます。アプリケーションは、*vname* と *data* を Fupdate32() に提供します。

Fupdate32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fupdate() または Fupdate32() を呼び出すことができます。

戻り値　この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Fupdate() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
ソース・バッファまたは宛先バッファのどちらかが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
ソース・バッファまたは宛先バッファがフィールド化バッファでないか、または Finit() で初期化されていません。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、宛先バッファで追加または変更されますが、バッファ領域の残りが十分ではありません。

関連項目　「FML 関数の紹介」、Fjoin、Fjoin32(3fml)、Fojoin、Fojoin32(3fml)、Fproj、Fproj32(3fml)、Fprojcpy、Fprojcpy32(3fml)

Fused、Fused32(3fml)

名前 Fused()、Fused32() ―フィールド化バッファで使用しているバイト数を返す

形式

```
#include <stdio.h>
#include "fml.h"

long
Fused(FBFR *fbfr)

#include "fml32.h"

long
Fused32(FBFR32 *fbfr)
```

機能説明 Fused() は、ユーザ・データとヘッダの両方を含む (インデックスは、いつでもドロップできるため含まない)、フィールド化バッファで使用している領域のサイズをバイト数で返します。fbfr は、フィールド化バッファを指すポインタです。

Fused32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fused() または Fused32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fused() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

関連項目　「FML 関数の紹介」、Fidxused、Fidxused32(3fml)、Funused、Funused32(3fml)

Fvall、Fvall32(3fml)

名前 Fvall()、Fvall32() ―フィールド・オカレンスを long 型で返す

```
#include <stdio.h>
#include "fml.h"

long
 Fvall(FBFR *fbfr, FLDID fieldid, FLDOCC oc)

#include "fml32.h"

long
Fvall32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Fvall() は、long 型および short 型の値の場合に Ffind() と同様に動作しますが、値を指すポインタではなく、フィールドの実際の値を返します。fbfr は、フィールド化バッファを指すポインタです。fieldid はフィールド識別子です。oc はフィールドのオカレンス番号です。

指定したフィールド・オカレンスが発見されない場合は、0 が返されます。この関数は、戻り値をチェックせずにフィールドの値を別の関数に渡すのに役立ちます。この関数は、型 FLD_LONG または FLD_SHORT のフィールドに対してのみ有効です。

Fvall32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvall() または Fvall32() を呼び出すことができます。

戻り値 Fvall() は、FLD_LONG および FLD_SHORT 以外のフィールド・タイプに対して 0 を返し、Ferror を FTYPERR に設定します。

この関数は、他のエラーが発生すると 0 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvall() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
fieldid が誤っている、またはフィールド・タイプが FLD_SHORT でも FLD_LONG でもありません。

関連項目 「FML 関数の紹介」、Ffind、Ffind32(3fml)、Fvals、Fvals32(3fml)

Fvals、Fvals32(3fml)

名前 Fvals()、Fvals32() ―フィールド・オカレンスの文字列の値を返す

形式

```
#include <stdio.h>
#include "fml.h"

char *
Fvals(FBFR *fbfr, FLDID fieldid, FLDOCC oc)

#include "fml32.h"

char *
Fvals32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

機能説明 Fvals() は、文字列の値の場合に Ffind() と同様に動作しますが、値が返されることを保証します。fbfr は、フィールド化バッファを指すポインタです。fieldid は、フィールド識別子です。oc はフィールドのオカレンス番号です。

指定したフィールド・オカレンスが見つからない場合は、NULL が返されます。この関数は、戻り値をチェックせずにフィールドの値を別の関数に渡すのに役立ちます。この関数は、型 FLD_STRING のフィールドに対してのみ有効です。ほかのフィールド・タイプの場合は、自動的に NULL 文字列を返します (つまり、変換は行われません)。

Fvals32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvals() または Fvals32() を呼び出すことができます。

戻り値 この関数は、エラーが発生すると NULL を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvals() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FBADFLD]
"unknown field number or type"
指定されたフィールド識別子は無効です。

[FTYPERR]
"invalid field type"
fieldid が誤っている、またはフィールド・タイプが FLD_STRING ではありません。

関連項目 「FML 関数の紹介」、CFfind、CFfind32(3fml)、Ffind、Ffind32(3fml)、Fvall、Fvall32(3fml)

Fvftos、Fvftos32(3fml)

名前 Fvftos()、Fvftos32() ―フィールド化バッファから C 構造体にコピー

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvftos(FBFR *fbfr, char *cstruct, char *view)

#include "fml32.h"

int
Fvftos32(FBFR32 *fbfr, char *cstruct, char *view)
```

機能説明 Fvftos() 関数は、フィールド化バッファから C 構造体へデータを転送します。fbfr は、フィールド化バッファを指すポインタです。cstruct は、C 構造体を指すポインタです。view は、コンパイルした VIEW 記述子の名前を指すポインタです。

フィールドは、フィールド化バッファから view のメンバ記述子に基づく構造体に複写されます。フィールド化バッファのフィールドが、一致するバッファを C 構造体に持たない場合は、無視されます。C 構造体で指定されたメンバがフィールド化バッファに一致するフィールドを持たない場合は、メンバに NULL 値が複写されます。使用した NULL 値が VIEW 記述子の各メンバに対して定義可能です。

C 構造体で複数のオカレンスを格納するには、構造体のメンバは配列である必要があります (たとえば、int zip[4] は zip の 4 つのオカレンスを格納できます)。バッファに存在するオカレンスの数が、配列にある要素数よりも少ない場合は、余分の要素スロットには NULL 値が割り当てられます。一方、バッファの配列にある要素数よりもオカレンスの数の方が多い場合は、余分なオカレンスは無視されます。

フィールド識別子およびメンバに対してマッピング・エントリが存在する場合でも、マッピングを禁止する VIEW 記述子のオプションがあります。これらのオプションは、初めは VIEW ファイルで指定されていますが、実行時に Fvopt() を使用して変更できます。

Fvftos32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvftos() または Fvftos32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvftos() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEINVAL]
"invalid argument to function"
呼び出された関数への引数の 1 つが無効です (たとえば、Fvftos の *cstruct* パラメータに NULL を指定している)。

[FBADACM]
"ACM contains negative value"
構造体からフィールド化バッファにデータを移動させるときには、連想カウント・メンバ (ACM) は負の値であってはなりません。

[FBADVIEW]
"cannot find or get view"
指定した VIEW 記述子が NULL であるか、あるいは VIEWDIR または VIEWFILES で指定されたファイルにこの記述子がありませんでした。

関連項目 「FML 関数の紹介」、Fvopt、Fvopt32(3fml)、viewfile(5)

Fvneeded、Fvneeded32(3fml)

名前 Fvneeded()、Fvneeded32() ― VIEW バッファに必要な大きさの計算

形式

```
#include <stdio.h>
#include "fml.h"

long
Fvneeded(char *subtype)

#include "fml32.h"

long
Fvneeded32(char *subtype)
```

機能説明 Fvneeded() は、VIEW C 構造体の大きさを返します。*subtype* は VIEW の名前です。Fvneeded() を呼び出すと、割り当てる VIEW バッファの大きさを決定できます。Fvneeded32() は 32 ビット VIEW で使用されます。

戻り値 Fvneeded() は、VIEW のサイズをバイト数で返します。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvneeded() は異常終了し、Ferror を次の値に設定します。

[FEINVAL]
"invalid argument to function"
VIEWDIR 環境変数および VIEWFILES 環境変数で指定された VIEW ファイルで、要求された VIEW が見つかりませんでした。

関連項目 「FML 関数の紹介」、viewfile(5)

Fvnull、Fvnull32(3fml)

名前 Fvnull()、Fvnull32() ー構造体の要素が NULL かどうかのチェック

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvnull(char *cstruct, char *cname, FLDOCC oc, char *view)

#include "fml32.h"

int
Fvnull32(char *cstruct, char *cname, FLDOCC32 oc, char *view)
```

機能説明 Fvnull() は、構造体要素のオカレンスが NULL であるかどうかを判断するのに使用します。cstruct は、C 構造体を指すポインタです。*cname* は、cstruct 内の要素名を指すポインタです。oc は要素のオカレンス番号です。view は、コンパイルした VIEW 記述子の名前を指すポインタです。

Fvopt() のオプションは、この関数に影響しません。

Fvnull32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのコンテキスト状態で実行している場合でも、Fvnull() または Fvnull32() を呼び出すことができます。

戻り値 Fvnull() は、C 構造体で指定した cname が NULL の場合は 1 を返し、NULL でない場合は 0 を返します。この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvnull() は異常終了し、Ferror を次の値に設定します。

[FBADVIEW]
"cannot find or get view"
指定した VIEW 記述子が VIEWDIR または VIEWFILES で指定されたファイル内にありませんでした。

[FNOCNAME]
"cname not found"
C 構造体フィールド名が、VIEW 記述子で見つかりません。

関連項目 「FML 関数の紹介」、Fvopt、Fvopt32(3fml)、viewfile(5)

Fvopt、Fvopt32(3fml)

名前 Fvopt()、Fvopt32() ―マッピング・エントリのフラグ・オプションの変更

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvopt(char *cname, int option, char *view)

#include "fml32.h"

int
Fvopt32(char *cname, int option, char *view)
```

機能説明 Fvopt() は、実行時にバッファと構造体のマッピング・オプションの指定を可能にします。cname は VIEW 記述子、view での要素名を指すポインタです。option は、マッピング・オプションのために必要な設定を指定します。有効なオプションおよびその意味は次の通りです。

F_FTOS
: フィールド化バッファから構造体への一方向のマッピング、VIEW 記述子内のフラグ S

F_STOF
: 構造体からフィールド化バッファへの一方向のマッピング、VIEW 記述子内のフラグ F

F_OFF
: フィールド化バッファと構造体の間でのマッピングを行わない、VIEW 記述子内のフラグ N

F_BOTH
: フィールド化バッファと構造体の間の双方向のマッピング、VIEW 記述子内のフラグ S および F

Fvopt32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvopt() または Fvopt32() を呼び出すことができます。

戻り値　この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー　次の条件の場合、Fvopt() は異常終了し、Ferror を次の値に設定します。

[FEINVAL]
"invalid argument to function"
呼び出された関数への引数の 1 つが無効でした (たとえば、cname パラメータまたは view パラメタに NULL を指定する、あるいは無効な option を指定する)。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに VIEW が見つかりませんでした。

[FNOCNAME]
"cname not found"
C 構造体フィールド名が、VIEW 記述子で見つかりません。

関連項目　「FML 関数の紹介」、viewfile(5)

Fvrefresh、Fvrefresh32(3fml)

名前 Fvrefresh()、Fvrefresh32() ― C 構造体からフィールド化バッファにコピー

形式

```
#include <stdio.h>
#include "fml.h"

void
Fvrefresh()

#include "fml32.h"

void
Fvrefresh32()
```

機能説明 Fvrefresh() は、ビュー構造マッピングの内部キャッシュ・メモリをクリアし、初期化し直します。この操作が必要になるのは、頻繁に使用するビューが動的に更新される場合に限られます。

Fvrefresh32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvrefresh() または Fvrefresh32() を呼び出すことができます。

戻り値 このルーチンは void 関数であり、値を返しません。

エラー このルーチンは void 関数であり、エラー・コードはセットされません。

関連項目 「FML 関数の紹介」

Fvselinit、Fvselinit32(3fml)

名前 Fvselinit()、Fvselinit32() ―構造体の要素を NULL 値で初期化する

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvselinit(char *cstruct, char *cname, char *view)

#include "fml32.h"

int
Fvselinit32(char *cstruct, char *cname, char *view)
```

機能説明 Fvselinit() は、C 構造体の個々の要素を、適切な NULL 値で初期化します。cstruct は、C 構造体を指すポインタです。cname は、cstruct の要素名を指すポインタです。view は、コンパイルした VIEW 記述子の名前を指すポインタです。

Fvselinit() は、VIEW がコンパイルされたときに C フラグが使用された場合は要素の関連するカウント・メンバを 0 に設定します。また、L フラグが VIEW ファイルで使用された場合は、関連する長さのメンバを関連する NULL 値の長さに設定します。

Fvselinit32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvselinit() または Fvselinit32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvselinit() は異常終了し、Ferror を次の値に設定します。

[FEINVAL]
"invalid argument to function"
呼び出された関数への引数の 1 つが無効です (たとえば、Fvselinit に無効な NULL cstruct パラメータを指定)。

[FBADVIEW]
"cannot find or get view"
指定した VIEW 記述子が NULL であるか、あるいは VIEWDIR または VIEWFILES で指定されたファイルにこの記述子がありませんでした。

[FNOCNAME]
"cname not found"
C 構造体フィールド名が、VIEW 記述子で見つかりません。

関連項目 「FML 関数の紹介」、Fvsinit、Fvsinit32(3fml)、viewfile(5)

Fvsinit、Fvsinit32(3fml)

名前 Fvsinit()、Fvsinit32() — C 構造体を NULL 値で初期化

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvsinit(char *cstruct, char *view)

#include "fml32.h"

int
Fvsinit32(char *cstruct, char *view)
```

機能説明 Fvsinit() は、C 構造体のすべてのメンバを、VIEW 記述子、view で指定された NULL 値に初期化します。cstruct は、C 構造体を指すポインタです。view は、コンパイルされた VIEW 記述子を指すポインタです。

Fvsinit() は、VIEW がコンパイルされたときに C フラグが使用された場合は要素の関連するカウント・メンバを 0 に設定します。また、L フラグが VIEW ファイルで使用された場合は、関連する長さのメンバを関連する NULL 値の長さに設定します。

Fvsinit32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvsinit() または Fvsinit32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvsinit() は異常終了し、Ferror を次の値に設定します。

[FEINVAL]
"invalid argument to function"
呼び出された関数への引数の 1 つが無効です (たとえば、Fvsinit() に無効な NULL cstruct パラメータが指定されている)。

[FBADVIEW]
"cannot find or get view"
指定した VIEW 記述子が NULL であるか、あるいは VIEWDIR または VIEWFILES で指定されたファイルにこの記述子がありませんでした。

関連項目 「FML 関数の紹介」、Fvselinit、Fvselinit32(3fml)、viewfile(5)

Fvstof、Fvstof32(3fml)

名前 Fvstof()、Fvstof32() ― C 構造体からフィールド化バッファにコピー

形式

```
#include <stdio.h>
#include "fml.h"

int
Fvstof(FBFR *fbfr, char *cstruct, int mode, char *view)

#include "fml32.h"

int
Fvstof32(FBFR32 *fbfr, char *cstruct, int mode, char *view)
```

機能説明 Fvstof() は、C 構造体からフィールド化バッファへデータを転送します。fbfr は、フィールド化バッファを指すポインタです。cstruct は、C 構造体を指すポインタです。mode は、転送が行われる方法を指定します。view は、コンパイルされた VIEW 記述子を指すポインタです。mode は、次の 4 つのいずれかの値です。

- FUPDATE
- Fojoin()
- Fjoin()
- Fconcat()

これらのモードが行う動作は、Fupdate()、Fojoin()、Fjoin()、および Fconcat() で記述しているものと同じです。ソース・バッファを指定する場所を除いて、Fvstof() をこれらの関数と同じように考えることができます。Fvstof() は C 構造体を指定します。FUPDATE は、NULL 値を持つ構造体の要素を移動しないことに注意してください。

Fvstof32() は、viewc32 で定義されたビューや、フィールド数の多い大規模ビューのための VIEW32 型付きバッファに対して使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvstof() または Fvstof32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fvstof() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効でした (たとえば、Fvstof() に NULL cstruct パラメータまたは無効な mode が指定されている場合)。

[FNOSPACE]
"no space in fielded buffer"
フィールド値は、フィールド化バッファで追加あるいは変更されますが、バッファには、十分な領域が残っていません。

[FBADACM]
"ACM contains negative value"
構造体からフィールド化バッファにデータを移動させるときには、連想カウント・メンバ (ACM) は負の値であってはなりません。

[FMALLOC]
"malloc failed"
CARRAY (または MBSTRING) あるいは文字列の値から変換するときに、malloc() を使用した領域の動的な割り当てが失敗しました。

関連項目 「FML 関数の紹介」、Fconcat、Fconcat32(3fml)、Fjoin、Fjoin32(3fml)、Fojoin、Fojoin32(3fml)、Fupdate、Fupdate32(3fml)、Fvftos、Fvftos32(3fml)

Fvstot、Fvttos(3fml)

名前 Fvstot()、Fvttos() — C 構造体からターゲットのレコード・タイプに変換、およびその逆の変換

形式

```
#include <stdio.h>
#include "fml.h"

long
Fvstot(char *cstruct, char *trecord, long treclen, char *viewname)

long
Fvttos(char *cstruct, char *trecord, char *viewname)

#include "fml32.h"

int
Fvstot32(char *cstruct, char *trecord, long treclen, char
*viewname)

int
Fvttos32(char *cstruct, char *trecord, char *viewname)

int Fcodeset(char *translation_table)
```

機能説明 Fvstot() は、C の構造体からターゲットのレコード・タイプにデータを転送します。Fvttos() は、ターゲットのレコードから C の構造体にデータを転送します。trecord は、ターゲットのレコードを指すポインタです。cstruct は、C 構造体を指すポインタです。viewname は、コンパイルされたビューの記述名を指すポインタです。コンパイルされたビュー記述のあるディレクトリとファイルを探すために、VIEWDIR と VIEWFILES が使用されます。

Fvttos32() と Fvstot32() は、32 ビットの VIEW に対して使用されます。

FML バッファからターゲット・レコードに変換するには、まず Fvftos() を呼び出して FML バッファから C 構造体に変換し、次に Fvstot() を呼び出してターゲット・レコードに変換します。ターゲット・レコードから FML バッファに変換するには、まず Fvttos() を呼び出して C 構造体に変換し、次に Fvstof() を呼び出してそのデータ構造体を FML バッファに変換します。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fvstot() または Fvttos() を呼び出すことができます。

デフォルト変換 - IBM/370

下に示したデフォルト・ターゲットは、IBM/370 の COBOL レコードです。デフォルトのデータ変換は、下の表に基づいて行われます。

表 2 デフォルトのデータ変換

データ構造	レコード
float	COMP-1
double	COMP-2
long	S9(9) COMP
short	S9(4) COMP
int	S9(9) COMP または S9(4) COMP
dec_t(*m*, *n*)	S9(2**m*-(*n*+1))V9(*n*)COMP-3
ASCII 文字	EBCDIC 文字
ASCII 文字列	EBCDIC 文字列
carray	文字配列
mbstring	マルチバイト文字配列

IBM/370 のレコードでは、フィールド間のフィルタ・バイトはありません。ビューに対応するデータ構造の一部をなすデータ・アイテムに対しては、COBOL SYNC 節は指定することはできません。

整数フィールドは、変換を実行するマシン上の整数のサイズに応じて 4 バイトまたは 2 バイトの整数に変換されます。

IBM/370 フォーマットへの変換、あるいは IBM/370 フォーマットからの変換を行う場合は、ビューの文字列フィールドは NULL で終了している必要があります。

carray フィールドまたは mbstring フィールドのデータは変更されずに渡されます。データの変換は行われません。

パック 10 進数は、IBM/370 環境では 1 バイトにパッキングされた 2 桁の 10 進数として存在し、下位の 1/2 バイトは符号を格納するために使用されます。パッキングされた 10 進数の長さは 1 ～ 16 バイトで、1 ～ 31 桁の数字と 1 つの符号の格納領域があります。

パック 10 進数は、C の構造体では dec_t というフィールド・タイプを利用することによってサポートされます。dec_t フィールドは、コンマで区切られた 2 つの数字で構成されたサイズに定義されています。コンマの左側の数字は、10 進数が占有するバイトのトータル長を示します。右側の数値は、小数点以下の桁数です。変換の公式を以下に示します。

```
dec_t(m, n) => S9(2*m-(n+1))V9(n)COMP-3
```

10 進数の値は、decimal() で説明した関数を利用することにより、他のデータ型に変換したり、他のデータ型から変換することができます (int、long、string、double、float など)。

下の表は、ASCII (左側) と EBCDIC (右側) の間でのデフォルトの文字変換における対応関係を示したものです。

```
| 00 00 |01 01 |02 02 |03 03 |04 37 |05 2d |06 2e |07 2f |
| 08 16 |09 05 |0a 25 |0b 0b |0c 0c |0d 0d |0e 0e |0f 0f |
| 10 10 |11 11 |12 12 |13 13 |14 3c |15 3d |16 32 |17 26 |
| 18 18 |19 19 |1a 3f |1b 27 |1c 1c |1d 1d |1e 1e |1f 1f |
| 20 40 |21 5a |22 7f |23 7b |24 5b |25 6c |26 50 |27 7d |
| 28 4d |29 5d |2a 5c |2b 4e |2c 6b |2d 60 |2e 4b |2f 61 |
| 30 f0 |31 f1 |32 f2 |33 f3 |34 f4 |35 f5 |36 f6 |37 f7 |
| 38 f8 |39 f9 |3a 7a |3b 5e |3c 4c |3d 7e |3e 6e |3f 6f |
| 40 7c |41 c1 |42 c2 |43 c3 |44 c4 |45 c5 |46 c6 |47 c7 |
| 48 c8 |49 c9 |4a d1 |4b d2 |4c d3 |4d d4 |4e d5 |4f d6 |
| 50 d7 |51 d8 |52 d9 |53 e2 |54 e3 |55 e4 |56 e5 |57 e6 |
| 58 e7 |59 e8 |5a e9 |5b ad |5c e0 |5d bd |5e 5f |5f 6d |
| 60 79 |61 81 |62 82 |63 83 |64 84 |65 85 |66 86 |67 87 |
| 68 88 |69 89 |6a 91 |6b 92 |6c 93 |6d 94 |6e 95 |6f 96 |
| 70 97 |71 98 |72 99 |73 a2 |74 a3 |75 a4 |76 a5 |77 a6 |
| 78 a7 |79 a8 |7a a9 |7b c0 |7c 6a |7d d0 |7e a1 |7f 07 |
| 80 20 |81 21 |82 22 |83 23 |84 24 |85 15 |86 06 |87 17 |
| 88 28 |89 29 |8a 2a |8b 2b |8c 2c |8d 09 |8e 0a |8f 1b |
| 90 30 |91 31 |92 1a |93 33 |94 34 |95 35 |96 36 |97 08 |
| 98 38 |99 39 |9a 3a |9b 3b |9c 04 |9d 14 |9e 3e |9f e1 |
| a0 41 |a1 42 |a2 43 |a3 44 |a4 45 |a5 46 |a6 47 |a7 48 |
| a8 49 |a9 51 |aa 52 |ab 53 |ac 54 |ad 55 |ae 56 |af 57 |
| b0 58 |b1 59 |b2 62 |b3 63 |b4 64 |b5 65 |b6 66 |b7 67 |
| b8 68 |b9 69 |ba 70 |bb 71 |bc 72 |bd 73 |be 74 |bf 75 |
| c0 76 |c1 77 |c2 78 |c3 80 |c4 8a |c5 8b |c6 8c |c7 8d |
| c8 8e |c9 8f |ca 90 |cb 9a |cc 9b |cd 9c |ce 9d |cf 9e |
| d0 9f |d1 a0 |d2 aa |d3 ab |d4 ac |d5 4a |d6 ae |d7 af |
| d8 b0 |d9 b1 |da b2 |db b3 |dc b4 |dd b5 |de b6 |df b7 |
| e0 b8 |e1 b9 |e2 ba |e3 bb |e4 bc |e5 4f |e6 be |e7 bf |
| e8 ca |e9 cb |ea cc |eb cd |ec ce |ed cf |ee da |ef db |
```

```
| f0 dc |f1 dd |f2 de |f3 df |f4 ea |f5 eb |f6 ec |f7 ed |
| f8 ee |f9 ef |fa fa |fb fb |fc fc |fd fd |fe fe |ff ff |
```

実行時には、Fcodeset() を呼び出すことにより、別の文字変換表を使用することができます。translation_table は、512 バイトのバイナリ・データを指している必要があります。最初の 256 バイトのデータは、ASCII から EBCDIC への変換テーブルとして解釈されます。残りの 256 バイトのデータは EBCDIC から ASCII への変換テーブルとして解釈されます。512 バイトより後ろのデータは無視されます。ポインタが NULL のときは、デフォルトの変換テーブルが使用されます。

戻り値 正常終了すると、Fvstot() はターゲット・レコードのデータ長を返し、Fvttos() は C 構造体のデータ長を返します。

これら 2 つの関数は、エラーが発生した場合は -1 という値を返し、Ferror をエラー条件を示す値にセットします。

エラー 次の条件が発生すると、Fvttos() は異常終了し、Ferror を次のように設定します。

[FEINVAL]
"invalid argument to function"
呼び出された関数の引数の 1 つが無効です (たとえば、Fvttos() に対して NULL の trecord または cstruct パラメータを指定した場合)。また、このエラーは、変換元または変換先のターゲット・レコードの値が範囲を超えている場合にも返されます。

[FBADACM]
"ACM contains negative value"
連想カウント・メンバ (Associated Count Member) は負の値にセットできません。

[FBADVIEW]
"cannot find or get view"
VIEWDIR または VIEWFILES で指定したファイルに viewname が見つかりません。

[FNOSPACE]
"no space in buffer"
変換したデータ構造のサイズがターゲット・レコードのサイズを上回っています。

[FVFOPEN]
"cannot find or open viewfile"
viewname 検索中にプログラムで VIEWDIR または VIEWFILES で指定したファイルの 1 つが見つかりませんでした。

[FEUNIX]
"operating system error"
viewname 検索中にプログラムは VIEWDIR または VIEWFILES で指定したファイルの 1 つを読み込み用に開けませんでした。

[FVFSYNTAX]
"bad viewfile"
viewname 検索中に VIEWDIR または VIEWFILES で指定したファイルの 1 つが破壊されていたか、VIEW ファイルではありませんでした。

[FMALLOC]
"malloc failed"
viewname 検索中に malloc() が VIEW 情報を格納するための領域の割り当てに失敗しました。

使用例 VIEW test.v

```
VIEW test
#type   cname   fbname  count   flag    size    null
float   float1  FLOAT1  1       -       -       0.0
double  double1 DOUBLE1 1       -       -       0.0
long    long1   LONG1   1       -       -       0
short   short1  SHORT1  1       -       -       0
int     int1    INT1    1       -       -       0
dec_t   dec1    DEC1    1       -       4,2     0
char    char1   CHAR1   1       -       -       ''
string  string1 STRING1 1       -       20      ''
carray  carray1 CARRAY1 1       -       20      ''
END
```

同等の COBOL レコード

```
02 OUTPUT-REC.
        05 FLOAT1                          USAGE IS COMP-1.
        05 DOUBLE1                         USAGE IS COMP-2.
        05 LONG1                         PIC S9(9) USAGE IS COMP.
        05 SHORT1                        PIC S9(4) USAGE IS COMP.
        05 INT1                          PIC S9(9) USAGE IS COMP.
        05 DEC1                          PIC S9(5)V9(2) COMP-3.
        05 CHAR1                          PIC X(01).
```

```
        05 STRING1                          PIC X(20).
        05 CARRAY1                          PIC X(20).
```

C プログラム

```
#include "test.h"
#include "decimal.h"

main()
{

        struct test s1;
        char data[100];

        s1.float1 = 1.0;
        s1.double1 = 2.0;
        s1.long1 = 3;
        s1.short1 = 4;
        s1.int1 = 5;
        deccvdbl(6.0,s1.dec1);
        s1.char1 = '7';
        (void) strcpy(s1.string1, "eight");
        (void) strcpy(s1.carray1, "nine");

        if (Fvstot((char *)&s1, data, reclen, "test") == -1) {
          printf("Fvstot failed: %sn", Fstrerror(Ferror));
          exit(0);
        }
        /* ターゲット・マシンに送信して応答を得る */
        ...

        /* 逆方向に変換する */
        if (Fvttos(data, (char *)&s1, "test") == -1) {
          printf("Fvttos failed:%sn", Fstrerror(Ferror));
          exit(0);
        }

        /* 構造体を使用する */
        .....
        exit(0);
}
```

関連項目 「FML 関数の紹介」、Fvftos、Fvftos32(3fml)、Fvstof、Fvstof32(3fml)、viewfile(5)

UNIX システムのリファレンス・ページの decimal(3)

Fwrite、Fwrite32(3fml)

名前 Fwrite()、Fwrite32() ―フィールド化バッファを書き込む

形式

```
#include <stdio.h>
#include "fml.h"

int
Fwrite(FBFR *fbfr, FILE *iop)

#include "fml32.h"

int
Fwrite32(FBFR32 *fbfr, FILE *iop)
```

機能説明 Fwrite() は、フィールド化バッファをストリームに書き込みます (ストリームについては、UNIX System V『プログラマ・リファレンス・マニュアル』の stdio(3S) を参照)。Fwrite() は、バッファのインデックスを破棄します。

fbfr は、フィールド化バッファを指すポインタです。iop は、出力ストリームを指す FILE 型のポインタです。

FLD_PTR フィールド・タイプの場合、ポインタによって指示されているデータではなく、ポインタだけが出力ストリームに書き込まれます。FLD_VIEW32 フィールド・タイプの場合、VIEW32 バッファ内のデータではなく、FVIEWFLD 構造体だけが出力ストリームに書き込まれます。

Fwrite32() は 32 ビット FML で使用されます。

マルチスレッドのアプリケーション内のスレッドは、TPINVALIDCONTEXT を含め、どのようなコンテキスト状態で実行していても、Fwrite() または Fwrite32() を呼び出すことができます。

戻り値 この関数は、エラー発生時に -1 を返し、Ferror を設定してエラー条件を示します。

エラー 次の条件の場合、Fwrite() は異常終了し、Ferror を次の値に設定します。

[FALIGNERR]
"fielded buffer not aligned"
バッファが適切なバウンダリで開始していません。

[FNOTFLD]
"buffer not fielded"
バッファがフィールド化されていないか、または Finit() で初期化されていません。

[FEUNIX]
"UNIX system call error"
write システム・コールが異常終了しました。外部の整数型変数 errno は、システム・コールによるエラーを示すために設定され、外部の整数型変数 Uunixerr (Uunix.h で定義されている値) は、エラーを返したシステム・コールに設定されます。

移植性 この関数は、Windows 用の BEA Tuxedo System Workstation DLL では使用できません。

関連項目 「FML 関数の紹介」、Findex、Findex32(3fml)、Fread、Fread32(3fml)

UNIX システムのリファレンス・ページの stdio(3S)

tpconvfmb32(3fml)

名前 tpconvfmb32() ―ソースの符号化からターゲットの符号化にマルチバイト文字を変換する

形式

```
#include <atmi.h>
#include "fml32.h"

extern int tperrno;

int
tpconvfmb32 (FBFR32 **bufp, FLDID32 *ids, char *target_encoding,
long flags)
```

機能説明 tpconvfmb32() は、FML32 型付きバッファにある FLD_MBSTRING フィールドのマルチバイト文字を、ターゲットの名前付きの符号化に変換します。具体的には、tpconvfmb32() は、FLD_MBSTRING フィールドで指定されたソースの符号化名と *target_encoding* で定義されたターゲットの符号化名を比較し、符号化名が異なる場合に tpconvfmb32() は、FLD_MBSTRING フィールドのデータをターゲットの符号化に変換します。

システムによる符号化変換に代わる方法として、tpconvfmb32() があります。プロセス TPMBACONV 環境変数を NULL 以外の値に設定した場合、システムによる FLD_MBSTRING フィールド・データの符号化変換が自動的に行われます。

bufp は、FML32 型付きバッファを指すポインタです。ポインタに関連付けられたサイズが FML32 バッファの変換済み出力データの処理に不十分な場合、bufp は内部的に再割り当てされます。*bufp* は、Falloc() 関数ではなく tpalloc() 関数を使用して定義する必要があります。*bufp* に FLD_FML32 フィールドを指定した場合、FLD_MBSTRING フィールドに対して再帰的にチェックされます。*bufp* に FLD_PTR フィールドを指定した場合、これらのフィールドは省略されます。

ids には、変換するフィールド識別子の配列を指すポインタが入ります。*ids* が NULL の場合、*bufp* にあるすべての FLD_MBSTRING は、必要に応じてターゲットの符号化に変換されます。配列を使用する場合は、0 (BADFLDID) で終了する必要があります。

target_encoding は、*bufp* メッセージ内にある FLD_MBSTRING フィールドの変換に使用する、ターゲットのコード・セットの符号化名です。*target_encoding* が NULL の場合、tpconvfmb32() は、プロセス TPMBENC 環境変数で定義された符号化名を使用します。

flags は、tpconvfmb32() では使用されません。これは、ユーザ定義の変換関数用のバッファ・タイプ・スイッチ関数に渡されます。

戻り値 正常終了の場合、tpconvfmb32() は 0 を返します。エラーが発生した場合、tpconvfmb32() は -1 を返し、tperrno を設定してエラー条件を示します。

エラー 次の条件の場合、tpconvfmb32() は異常終了し、tperrno を次の値に設定します。

[TPEPROTO]
bufp は、バッファ・タイプ・スイッチ変換関数がない Tuxedo バッファに変換します。

[TPESYSTEM]
Tuxedo システムのエラーが発生しました (*bufp* が有効な Tuxedo バッファに対応していない場合など)。

[TPEINVAL]
target_encoding または *bufp* が NULL です。

[TPEOS]
オペレーティング・システムのエラーが発生しました。外部整数 Uunixerr (Uunix.h で定義されている値) が、そのエラーを返したシステム・コールに設定されます。

関連項目 Fmbpack32(3fml)、Fmbunpack32(3fml)、tpalloc(3c)、tpsetmbenc(3c)、tuxgetmbaconv(3c)、tuxgetmbenc(3c)、tuxsetmbaconv(3c)、tuxsetmbenc(3c)