

Tmax FDLリファレンスガイド

Tmax v6.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

Restricted Rights Legend

All TmaxSoft Software (Tmax®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(Tmax®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

Trademarks

Tmax®, Tmax WebtoB® and JEUS® are registered trademark of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

Tmax®, Tmax WebtoB®, JEUS® は、TmaxSoft Co., Ltd.の登録商標です。その他、記載されている会社名、製品名などは、各社の商標または登録商標です。

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : openssl-0.9.7.m, zlib-1.1.4, expat-2.0.0, net-snmp, DCE1.0, pthread, google-diff-match-patch, libevent, getopt

Detailed Information related to the license can be found in the following directory :
\${INSTALL_PATH}/license/oss_licenses

この製品の一部ファイルまたはモジュールは、openssl-0.9.7.m、zlib-1.1.4、expat-2.0.0、net-snmp、DCE1.0、pthread、google-diff-match-patch、libevent、getoptライセンスを遵守します。

詳細情報については、製品ディレクトリーの\${INSTALL_PATH}/license/oss_licensesに記載されている事項を参照してください。

文書情報

文書名: Tmax FDLリファレンスガイド

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	vii
第1章 紹介	1
1.1. 概要	1
1.2. C構造体とフィールド・バッファ	1
1.3. FDL表の作成	3
1.3.1. コマンド	3
1.3.2. 例	5
第2章 FDL関数	7
2.1. 概要	7
2.2. fballoc	10
2.3. fbbufop	12
2.4. fbbufop_proj	16
2.5. fbcalcsiz	18
2.6. fbchg_tu	19
2.7. fbchg_tut	21
2.8. fbdelall	22
2.9. fbdelall_tu	24
2.10. fbdelete	25
2.11. fbextread	27
2.12. fbfldcount	30
2.13. fbfpri	32
2.14. fbfree	33
2.15. fbftos	34
2.16. fbget	38
2.17. fbgetalloc_tu	39
2.18. fbgetalloc_tut	41
2.19. fbget_fbsiz	43
2.20. fbget_fldkey	45
2.21. fbget_fldname	46
2.22. fbget_fldno	47
2.23. fbget_fldtype	48
2.24. fbgetlast_tu	49
2.25. fbget_strfldtype	51
2.26. fbget_tu	52
2.27. fbget_tut	54
2.28. fbget_unused	56
2.29. fbget_used	57
2.30. fbgetf	58
2.31. fbgetlen	60
2.32. fbgetnth	61

2.33.	fbgetntht	62
2.34.	fbgetval	64
2.35.	fbgetvali	66
2.36.	fbgetval_last_tu	68
2.37.	fbgetvall_tu	70
2.38.	fbgetvals	71
2.39.	fbgetvals_tu	73
2.40.	fbgetvalt	74
2.41.	fbinit	76
2.42.	fbinsert	77
2.43.	fbisfbuf	79
2.44.	fbispres	80
2.45.	fbkeyoccur	82
2.46.	fbmake_fldkey	83
2.47.	fbnext_tu	84
2.48.	fbprint	86
2.49.	fbput	87
2.50.	fbputt	89
2.51.	fbrealloc	91
2.52.	fbread	92
2.53.	fbnull	93
2.54.	fbstelinit	95
2.55.	fbstinit	97
2.56.	fbstof	98
2.57.	fbsterror	100
2.58.	fbtypecvt	102
2.59.	fbupdate	103
2.60.	fbwrite	105
2.61.	getfberrno	106
2.62.	getfberror	107
索引	109

このガイドについて

対象読者

本書は、Tmax[®] (以下、Tmax)のFDL(Field Definition Language)関数を使用してプログラムを実装する開発者を対象としています。FDL関数は、フィールド・バッファを使用して、異なるプロセス間のデータを送受信するためにフィールドを操作(保存、修正、削除)する関数です。FDL関数の定義とサンプル・プログラムを参照してFDLのすべての機能を活用することにより、プログラムの実装における効率性を高めます。

前提知識

本書は、Tmaxシステムの概要とTmaxシステムが提供する各種機能や特性などを習得するための手引書です。

本書を理解するためには、以下の事項についての知識が必要です。

- ミドルウェアおよびUNIXシステムについての知識
- Tmaxの基本概念
- Java, Cプログラミングについての知識

制限事項

本書を読む前にTmaxの基本概念を熟知している必要があります。実務上の具体的な使用方法や管理・運用についての内容は、各製品ガイドを参照してください。

参考

Tmaxシステムの開発についての基本的な内容は、『Tmax 運用ガイド』および『Tmax アプリケーション開発ガイド』を、Tmaxが提供するコマンドとC APIについては、『Tmax リファレンスガイド』を参照してください。

本書の構成

本書は、計2章で構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 紹介

FDLの全般的な内容について説明します。

- 第2章: FDL関数

FDLで使用するAPI関数について説明します。

表記上の規則

表記	意味
<AaBbCc123>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlとCを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<div>参考</div>	参照/注意事項
[図1.1]	図の名称
[表1.1]	表の名称
AaBbCc123	コマンド、コマンド実行後の画面に出力された結果物、サンプル・コード
[]	オプション・パラメータ値
	選択パラメータ値

システム要件

	要求事項
プラットフォーム	IBM AIX 5.x / 6.1 / 7.1
	HP-UX 11.xx
	SunOS 5.7~5.9 / SunOS 5.10 / SunOS 5.11
ハードウェア	1GB以上のハードディスク空き容量
	512MB以上のメモリー空き容量
データベース	Oracle 9~12
	Tibero 4~5
	DB2
	Informix

関連文書

ガイド	説明
Tmax 運用ガイド	Tmaxを利用するための環境設定ファイルとシステム運用方法について説明しています
Tmax アプリケーション開発ガイド	Tmaxアプリケーション・プログラムの開発で使用するAPIの概念と使用方法および例について説明しています
Tmax リファレンスガイド	Tmaxアプリケーションの開発に使用するコマンドおよびクライアントとサーバーの接続、通信に使用する関数の使用方法と例について説明しています

お問合せ先

Korea

TmaxSoft Co., Ltd.
45, Jeongjail-ro, Bundang-gu,
Seongnam-si, Gyeonggi-do, 13613
South Korea
Tel: +82-31-8018-1000
Fax: +82-31-8018-1115
Email: info@tmax.co.kr
Web (Korean): <http://www.tmaxsoft.com>
TechNet: <http://technet.tmaxsoft.com>

USA

TmaxSoft Inc.
101 North Wacker Drive, Suite 2014,
Chicago, IL 60606
U.S.A
Tel: +1-312-525-8330
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/us_en/home

Japan

TmaxSoft Japan Co., Ltd.
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073
Japan
Tel: +81-3-5765-2550
Fax: +81-3-5765-2567
Email: info@tmaxsoft.co.jp
Web (Japanese): <http://www.tmaxsoft.co.jp>

China

Beijing TmaxSoft System Software Co., Ltd.
Room103, No.2 Huizhong Building, Seven Street Shangdi,
Haidian District, Beijing, 100085
P.R.China
Tel: +86-10-6298-8827
Email: info@tmaxsoft.com.cn
Web (Chinese): http://www.tmaxsoft.com/cn_en/home_cn_en

Brazil

Tmax Brasil Sistemas e Serviços Ltda.
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel
Alphaville Barueri, Sao Paulo, 06472-001
Brazil
Tel: +55-11-4191-3100
Fax: +55(11) 4191-3705 (extension#112)
Email: info.bra@tmaxsoft.com
Web (Portuguese): http://www.tmaxsoft.com/br_en/home_br_en

Russia

Tmax Rus L.L.C.
Leninsky prospekt, 113/1 (Park Place Moscow),
Office 318e, Moscow, 117198
Russia
Tel: +7(495)970-01-35
Email: info.rus@tmaxsoft.com
Web (Russian): http://www.tmaxsoft.com/ru_ru/home_ru_ru

Singapore

Tmax Singapore Pte. Ltd.
430 Lorong 6, Toa Payoh #10-02,
OrangeTee Building, 319402
Singapore
Tel: +65-6259-7223
Fax: +65-6258-7112
Email: info.sg@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/sg_en/home_sg_en

United Kingdom

TmaxSoft UK Ltd.
215 Knyvett House, Watermans Business Park,
The Causeway, Staines TW18 3BAB
United Kingdom
Tel: +44-1784-895005
Email: info.uk@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/gb_en/home_gb_en

Canada

TmaxSoft Canada, Inc.
2425 Matheson Blvd East, 8th floor,
Unit 824 Mississauga, ON, L4W 5K4
Canada
Tel: +1-905-361-2888
Email: info.canada@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/ca_en/home_ca_en

Australia

TmaxSoft Proprietary Limited
L32, 101 Miller Street, North Sydney 2060
Australia
Tel: +91-9845-330-704
Email: info.aus@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/au_en/home_au_en

India

TmaxSoft Technologies Private Limited
Sobha Alexander Plaza, 3rd Floor,
16/2 Commissariat Road, Bangalore-560025
India
Tel: +91-9845-330-704
Email: info.india@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/in_en/home_in_en

Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office
Windowist Tower. Eski Buyukdere Cad. No:26,
Maslak 34467 Istanbul
Turkey
Tel: +90-544-553-6045
Email: cslee@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/tr_en/home_tr_en

第1章 紹介

本章では、FDLの全般的な内容について説明します。

1.1. 概要

データ型とそのタイプに割り当てられるメモリーの量は、プラットフォームごとに異なります。このようなメモリーの割り当て方法の違いによって、同じデータ型を使用しても通信後のデータ値は予想と異なることがあります。一般的に、互いに異なるプラットフォーム間のデータ値を同じように認識するためには、データを文字列に変換して通信する方法を使用します。この方式を使用した場合、ネットワークの負荷が追加で発生することがあります。Tmaxはこのような通信上の問題を解決するために、FDL方式を提供します。

FDLは、フィールド・バッファという格納構造を定義し、操作する一連のC関数の集合です。

FDL方式は、識別子と識別子に対応するフィールド・バッファでのデータ値がペアで存在し、種類が異なるプロセス間のデータを相互交換できるように実装します。したがって、プロセス間のデータ交換は識別子(identifier)で行い、実際の値を処理する際はその識別子にマッピングされているフィールド・バッファのデータ値が呼び出されます。

以下は、本書で主に使われる用語についての説明です。

用語	説明
フィールドキー(fieldkey)	フィールド識別子FDLレコードやフィールド・バッファ内で独立したデータ項目のタグです
フィールド・バッファ(fieldbuffer)	データとデータを識別するフィールドキーを保存するバッファです
フィールド型(fieldtype)	標準C(ANSI C)で提供するshort型、integer型、long型、float型、double型、char型のいずれも可能です。また、Tmaxではstring型とcarray型をサポートします
フィールド順	フィールドを操作および処理する上で、フィールドキー・バッファ内で同じフィールドキーが複数回繰り返して現れる場合に指定する索引です

1.2. C構造体とフィールド・バッファ

レコードを表現する方法は多様です。その中で、C言語で提供するデータ構造体を使用してレコードを表現する方法と、フィールド・バッファを使用してレコードを表現する方法について説明します。

C 構造体

以下はC言語で作成されたプログラムで、学生のID、名前、住所、年齢、性別などの情報を含むために構造体を使用してフィールドを構成した例です。

```
struct student {  
    long stid;  
    char name[10];  
    char addr[40];  
    short age;  
    char sex;  
} ;
```

stidはlong型、nameとaddrはarray型、ageはshort型、sexはchar型の変数で宣言されています。

変数xを指定して構造体studentを示す場合、以下のように表現されます。これをフィールドのアドレスに使用できます。

```
x→stid,  
x→name,  
x→addr,  
x→age,  
x→sex
```

構造体を他のプロセスに渡すには、該当構造体のフィールドのうち使用していないフィールドを含めてすべてのフィールドのデータを渡し、固定長レコードとして通信する必要があります。また、構造体フィールドの名前が変われば、該当構造体のフィールドを使用するすべてのプログラムを再コンパイルする必要があります。

フィールド・バッファ

フィールド・バッファはC構造体とは異なり、レコードをフィールドに分割し、レコードの各フィールド間に関連性がある接続を提供するデータ構造です。フィールド名はフィールドのデータ型と一緒にデータが格納されているフィールドキーの値を示すもので、16文字まで使用できます。

フィールド・バッファの主なメリットは**データの独立性**です。該当フィールドを使用するすべてのプログラムを再コンパイルしなくても、フィールドのデータ型と長さを変更できるだけでなく、各フィールドの長さを可変的に使用できます。C構造体では、使用していないフィールドも一緒に全体構造体を送信する必要がありますが、フィールド・バッファ方式は、実際に使用するフィールドの値のみ送信できます。フィールド・バッファの各フィールドは、データ型と一意の識別番号の組み合わせで構成されます。整数型の識別番号をフィールドキーといい、フィールド・バッファはフィールドキーとデータをペアで格納します。

1.3. FDL表の作成

フィールド表は、Tmaxで提供するfdlcユーティリティを利用して一定形式に合わせた一般テキストファイルをコンパイルして作成します。

1.3.1. コマンド

以下は、フィールド型と一意の番号が定義されたdemo.fというユーザー定義ファイルの例です。このファイル形式のみTmaxで使用できます。

<demo.f>

```
#demo.f
#name    number  type    flags   comments
INPUT    101     string  -       -
OUTPUT   102     string  -       -
```

# name	number	type	flags	comments
*base	100			
INPUT	101	string	-	-
OUTPUT	102	string	-	-
SEQNO	201	string	-	-
CORPNO	202	string	-	-
COMPDATE	203	string	-	-
TOTMON	204	long	-	-
GUARAT	205	float	-	-
GUAMON	206	float	-	-
T_BITMAP	900	carray	-	-
FILENAME	901	string	-	-
NEWFILE	902	string	-	-
*base	1000			
data1	301	long	-	-
data2	302	long	-	-
data3	303	long	-	-
sumdata	304	long	-	-
data4	305	long	-	-

フィールド表の**nam**、**number**、**type**は必ず定義します。

フィールド値	説明
name	48文字以下で定義されたフィールド名です。fbget_fldkey()のパラメータで使用し、フィールド識別子を取得する機能を行います
number	<p>typeと組み合わせてフィールド識別子を作成するのに使用される番号です。</p> <p>numberとtypeを組み合わせてフィールド識別子を作成します。typeは一意になれないため、numberはすべてのフィールド表で一意の番号である必要があります。</p> <p>上のdemo.fの例で、「*base番号」はbaseが使用された後のnumberのbaseになるもので、実際にINPUTのnumberを201と指定したのと同じ役割をします。このbaseは他のbaseが出るまで有効です</p>
type	該当フィールドのフィールド型を設定します。使用できるフィールド型は、char、short、int、long、float、double、string、carrayの8つです

fdlc

フィールドキー表をコンパイルするコマンドです。フィールドキー方式は、クライアント/サーバー間のデータ通信で構造体を送信する方法のように全体構造体の項目は渡さず、必要な項目のみを渡します。項目を渡すには、それぞれの項目を区別できる一意キーが必要です。fdlcは、テキスト形式で定義されているフィールドキー表をコンパイルし、フィールドキーを作成するコマンドです。

● 使用方法

```
$ fdlc ( -c ) { - a | c | d | u } [-f] [-h ヘッダー・ファイル名] { -i フィールドキー表のファイル名 }
[-jc | ji] [-o 結果] [-u] [-p パッケージ名] [-x] [-V]
```

項目	説明
[-a]	<p>作成されたバイナリ形式のファイルに、テキストで作成されたフィールドキー表をコンパイルし、必要な部分を追加するオプションです。</p> <p>[-f]オプションを使用して対象ファイルを指定します。デフォルト・ファイルはtmax.fdlです。重複するフィールドの場合、新規値で置き換えられます</p>
[-c]	<p>テキストで作成されたフィールドキー表をコンパイルし、バイナリ形式のファイルを作成します。すでにバイナリ・ファイルが存在する場合、新しい内容で置き換えられます(デフォルト値)</p>
[-d]	<p>作成されたバイナリ形式のファイルに、テキストで作成されたフィールドキー表をコンパイルし、必要な部分を削除するオプションです。</p> <p>[-f]オプションを使用して対象ファイルを指定します。デフォルト・ファイルはtmax.fdlです</p>

項目	説明
[-u]	作成されたバイナリ形式のファイルに、テキストで作成されたフィールドキー表をコンパイルし、必要な部分を修正または追加するオプションです。 [-f] オプションを使用して対象ファイルを指定します。デフォルト・ファイルは tmax.fdl です
[-f]	テキストで作成されたフィールドキー表をコンパイルし、必要な部分を追加、修正、削除する場合に対象ファイルを指定するオプションです
[-h ヘッダー・ファイル名]	ヘッダー・ファイル名を変更する場合に使用するオプションです。 [-h] オプションなしでコンパイルした場合、fdlヘッダー・ファイル名は フィールドキー表名_fdl.h です
{ -i フィールドキー表のファイル名 }	クライアント・プログラムとサーバー・プログラムで使用されるフィールドキー表を定義したファイルを指定します。必須オプションで、パスと一緒に指定できます
[-jiljc]	WebTで使用されるオプションです。作成されるフィールド定義クラスの形式を指定します – ji : インターフェース形式のフィールド定義クラス・ファイルを作成します – jc : クラス形式のフィールドキー定義Javaファイルを作成します
[-o 結果]	コンパイル結果の名前を変更する際に使用するオプションです。 [-o] オプションなしでコンパイルした場合、fdlファイル名は tmax.fdl です
[-p パッケージ名]	WebTで使用されるオプションです。作成されたフィールド定義クラスのパッケージ名を与えられた値で設定します
[-x]	必ず [-a] オプションと一緒に使用します。このオプションが使用された場合、重複するフィールドについては元の値を維持します
[-V]	実行ファイルのバージョンを確認できます

1.3.2. 例

以下は、[-jiljc] オプションを設定して作成したJavaファイルの例です。

```
$> fdlc -c -i demo.f -ji -jc webtdemo
```

以下は、作成されたdemo_fdl.javaの内容です。

```
package webtdemo;
public interface demo_fdl {
```

```
public int INPUT = (469762149);    /* number: 101 type: string */
public int OUTPUT = (469762150); /* number: 102 type: string */
}
```

以下は、現在のディレクトリーにあるdemo.fフィールドキー表ファイルをfdlcユーティリティでコンパイルするコマンドです。コンパイル後、tmax.fdlとdemo_fdl.hファイルが作成されます。demo_fdl.hはフィールドキーが定義されているヘッダー・ファイルです。

```
$fdlc -c -i demo.f
```

fdlcユーティリティを使用してdemo.fをコンパイルすると、demo_fdl.hとtmax.fdlというファイルが作成されます。ここで作成されるファイル名を変更するには、以下のように[-o]オプションを使用し、希望するファイル名でバイナリ・ファイルを作成できます。

```
$fdlc -c -i demo.f -o filename.fdl
```

作成されたファイルは、クライアントとサーバーがフィールド・バッファーを使用して通信する場合に参照(プログラムと一緒にコンパイルされる)され、必要なデータ型のフィールドを使用できます。フィールド・バッファーを使用する際は、環境変数のFDLFILEにfilename.fdlの位置を必ず以下のように定義します。ここでのfilename.fdlとはfdlcユーティリティで作成されたバイナリ・ファイルを意味します。デフォルト値のtmax.fdlや[-o]オプションを指定し、作成したバイナリ・ファイルを入力します。

```
FDLFILE=/home/tmax/sample/fdl/filename.fdl
```

参考

1. フィールド表を使用する際、フィールドキーはドメイン内で一意である必要があります。
 2. fdlcユーティリティについての詳細は、『Tmax リファレンスガイド』を参照してください。
-

第2章 FDL関数

本章では、FDLで提供する関数の使用方法と例について説明します。

2.1. 概要

以下は、FDL関数の一覧です。

関数名	説明
fballoc	フィールド・バッファのサイズを計算し、メモリーを割り当てます
fbbufop	フィールド・バッファ内でのフィールド・データの移動、コピー、比較などを入力パラメータであるmodelに従って処理します
fbbufop_proj	特定フィールド・バッファのフィールド・データを指定したフィールド・バッファにコピーします
fbcalcsz	フィールド・バッファがメモリーに割り当てられる際に適切なメモリーが割り当てられるように、フィールド・バッファのサイズを計算します
fbchg_tu	フィールド・バッファで、入力パラメータで指定したフィールドキーのフィールド順のデータを変更します
fbchg_tut	フィールド・バッファで、入力パラメータで指定したフィールドキーのフィールド順のデータをユーザーが指定したタイプでフィールドキーのタイプに変換します
fbdelall	特定フィールドキーに格納されている全体データをフィールド・バッファから削除します
fbdelall_tu	フィールドキーの配列に指定されているすべてのフィールドキーに格納されているすべてのフィールド・データを削除します
fbdelete	メモリーに割り当てたフィールド・バッファから、指定したフィールドキーの指定したフィールド順に該当する特定フィールドのデータを削除します
fbextread	指定する特定ファイルからフィールドキーとそれに該当する値を読み込み、メモリーに割り当てたフィールド・バッファに格納します
fbfldcount	フィールド・バッファに現在格納されているすべてのフィールドの数を返します
fbfprint	フィールド・バッファに格納されているフィールド・データを指定したファイル・ストリームに出力します
fbfree	割り当てられたフィールド・バッファのメモリーを解除します
fbftos	フィールド・バッファに格納されているデータを、マッチングする構造体バッファに送信します

関数名	説明
fbget	フィールド・バッファで指定したフィールドキーに該当するフィールド・データを読み込みます
fbgetalloc_tu	指定したフィールドキーのフィールド順に該当するフィールド・データを読み込みます
fbgetalloc_tut	フィールドキーのフィールド順に該当するフィールド・データを、指定したタイプで必要な長さ分のバッファを新しく割り当てを受けて格納し、返します
fbget_fbsize	割り当てられたフィールド・バッファのサイズを返します
fbget_fldkey	フィールド表(FDLFILEに設定されたFDLファイル)と関連し、指定したフィールド名とマッチングする識別子を返します
fbget_fldname	フィールド表(FDLFILEに設定されたFDLファイル)と関連し、指定したフィールドキーとマッチングするフィールド名を返します
fbget_fldno	フィールド表を作成するために必要なデータのうち、指定したフィールドキーとマッチングするフィールド番号を返します
fbget_fldtype	指定したフィールドキーのフィールド型を整数で返します
fbgetlast_tu	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、順番が最後のデータを返します
fbget_strfldtype	指定したフィールドキーに格納されるフィールド・データのタイプを文字列に変換します
fbget_tu	フィールド・バッファで指定したフィールドキーの指定したフィールド順に該当するフィールドのデータを読み込みます
fbget_tut	指定したフィールドキーのフィールド順に該当するフィールド・データを指定したタイプに変換して返します
fbget_unused	フィールド・バッファのうち、まだ使用されていないフィールド・バッファのサイズを計算し、バイト単位で返します
fbget_used	フィールド・バッファのうち、現在使用中のフィールド・バッファのサイズを計算し、バイト単位で返します
fbgetf	フィールド・バッファに格納されているフィールド・データを順番に読み込みます
fbgetlen	フィールド・バッファに指定したフィールドキーのフィールド順に該当するフィールド・データの長さを返します
fbgetnth	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したデータが格納されているフィールドの数を返します
fbgetntht	指定したフィールドキーに格納されているフィールド・データのうち、指定したデータが格納されているフィールド順を返します
fbgetval	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したフィールド順に該当するデータを返します
fbgetvali	フィールド・バッファで指定したフィールドキーに格納されているデータのうち、指定したフィールド順に該当するフィールド・データを整数に変換して返します

関数名	説明
fbgetval_last_tu	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、順番が最後のデータを返します
fbgetvall_tu	フィールド・バッファで指定したフィールドキーのフィールド順のデータをlong型に変換して返します
fbgetvals	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したフィールド順のフィールド・データを、そのフィールド型に関係なくすべて文字列に変換して返します
fbgetvals_tu	フィールド・バッファでフィールドキーに格納されているフィールド・データを返します
fbgetvalt	フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したフィールド順に該当するフィールド・データを指定したタイプ(totype)に変換して返します
fbinit	メモリーに割り当てられたフィールド・バッファを初期化します
fbinsert	メモリーに割り当てられたフィールド・バッファに指定したフィールドキーのフィールド順に該当するデータを返します
fbisfbuf	指定したフィールド・バッファがメモリーに割り当てられた有効なバッファであるかを確認します
fbispres	フィールド・バッファに指定したフィールドキーのフィールド順にフィールド・データが存在するのを確認します
fbkeyoccur	フィールド・バッファで指定したフィールドキーに格納されているすべてのフィールドの数を返します
fbmake_fldkey	新しいフィールドキーを動的で作成します
fbnext_tu	フィールド・バッファで次のフィールドキーとフィールド・データを返します
fbprint	現在のフィールド・バッファの内容を決められた形式に従って標準出力します
fbput	メモリーに割り当てられたフィールド・バッファに新しいフィールドを追加します
fbputt	メモリーに割り当てられたフィールド・バッファに新しいフィールドを追加します
fbrealloc	メモリーに割り当てられたフィールド・バッファのサイズが不足した場合、メモリーのサイズを増やして、再度割り当てます
fbread	指定したファイル・ストリームからデータを読み込んで、メモリーに割り当てられたフィールド・バッファにフィールド・データでロードします
fbnull	フィールド・バッファで指定したフィールドキーのフィールド順とマッピングする構造体のメンバー変数がNULLであるかを確認します
fbstelinit	入力パラメータで指定した構造体のメンバー変数を個別にNULLに初期化します
fbstinit	入力パラメータで指定した構造体変数を初期化します
fbstof	C構造体バッファ(stname)に格納されているデータをマッチングするフィールド・バッファに送信します

関数名	説明
<code>fbsterror</code>	フィールド・バッファに関連するAPI関数を実行する際に発生するエラーコードに該当する内容を文字列で返します
<code>fbtypecv</code>	指定したデータ型のフィールド・データを指定した型に変換して返します
<code>fbupdate</code>	フィールド・バッファに、入力パラメータに指定したフィールドキーのフィールド順の位置に、指定したデータをすでに格納されているフィールド・データと変更します
<code>fbwrite</code>	メモリに割り当てられたフィールド・バッファの内容をiopに指定したファイル・ストリームに格納します
<code>getfberno</code>	フィールド・バッファに関連するAPI関数を実行中にエラーが発生した場合にエラーコードを返します
<code>getfberror</code>	フィールド・バッファに関連するAPI関数を実行中にエラーが発生した場合にエラーコードを返します

2.2. fballoc

フィールド・バッファのサイズを計算し、メモリを割り当てる関数です。

`f~()`、`t~()`、`te~()`を組み合わせた形式です。関数で使用するフィールドの数とデータの長さをパラメータに伝達すると、フィールド・バッファのサイズを計算し、計算されたサイズ分のフィールド・バッファをメモリに割り当てます。

割り当てられるメモリ・サイズ = `count * sizeof(int) + datalen + フィールドキーのヘッダーのサイズ`

パラメータに伝達された値で計算した結果が1024バイトより小さい場合、デフォルト値として1024バイトが割り当てられます。関数を使用すると、使用するサイズ分のみメモリが割り当てられるため、メモリの浪費を防ぐことができます。この関数で割り当てられたバッファを使用後は、`fbfree()`を使用して必ず解除します。

● プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
FBUF *fballoc(int count, FLDLEN datalen)
```

● パラメータ

パラメータ	説明
<code>count</code>	データを格納するフィールドの数を指定します
<code>datalen</code>	フィールド・バッファに格納されるデータ全体のサイズをバイト単位で指定します

- 戻り値

戻り値	説明
メモリー・ポインタ	関数の呼び出しに成功しました
NULL	関数の呼び出しに失敗しました (ferrorにエラーコードが設定されます)

- エラー

ferrnoには以下の値が設定されます。

エラーコード	説明
FBEMALLOC	システムエラーです。フィールド・バッファをメモリーに割り当てることに失敗しました

- 例

```
#include <fbuf.h>
#include <atmi.h>
...
FBUF *fbuf;
if ((fbuf = fballoc(100, 1000)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    ...
/* result allocated size : 1408 bytes */
```

フィールド・バッファをメモリーに割り当てるには、tpalloc()を使用する方法もあります。

以下のように、tpalloc()の3番目のパラメータに0を入力すると、デフォルト値の1024バイトが割り当てられます。パラメータに1024バイトより小さい数を入力しても、デフォルト値の1024バイトが割り当てられます。

```
. . .
FBUF *fbuf;
if ((fbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
/* result allocated size : 1024 bytes */
```

- 関連関数

fbisfbuf(), fbcalsize(), tpalloc(), fbinit()

2.3. fbbufop

フィールド・バッファ内でのフィールド・データの移動、コピー、比較などを入力パラメータであるmodeに従って処理する関数です。fbbufop()のパラメータのうちdestとsrcフィールド・バッファは同じバッファを使用できず、他のメモリーに割り当てられたバッファを必ず使用します。

● プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbbufop(FBUF *dest, FBUF *src, int mode)
```

● パラメータ

パラメータ	説明
dest	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです。srcにある内容をmodeに従ってdestに格納します
src	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです。変更される値を持つバッファです
mode	Process modeのうち1つを設定します

以下は、Process modeについての説明です。

Process mode	説明
FBMOVE	srcフィールド・バッファの内容をdestフィールド・バッファにコピーします。このモードではdestパラメータにstringまたはcarrayバッファを使用できます。 stringまたはcarrayバッファを使用するには、destパラメータを使用する際にフィールド・バッファ(FBUF *)に型キャストします
FBCOPY	srcフィールド・バッファの内容をdestフィールド・バッファにコピーします。このモードでは、dest、srcパラメータがすべてフィールド・バッファである必要があります
FBCOMP	以下は、設定値についての説明です – 0または1 : destとsrcフィールド・バッファの内容を比較し、マッチングする場合があります。0を返す場合は、2つのフィールド・バッファの値、構造、数、順序などが完全に一致します。1を返す場合は、構造、数、順序などがすべて一致しますが、値は一致しません – -1 : destとsrcフィールド・バッファの内容を比較し、マッチングしない場合です
FBCONCAT	srcフィールド・バッファの内容をdestフィールド・バッファに加えた場合です。destフィールド・バッファに既存データとsrcフィールド・バッファの内容が格納されます

Process mode	説明
FBJOIN	フィールドキーとフィールド順がマッチングするフィールドに限り、該当データをsrcフィールド・バッファからdestフィールド・バッファにコピーします。ただし、destフィールド・バッファに存在していたデータはすべて消えるため、注意してください
FBOJOIN	FBJOINモードと類似の動作をしますが、関数を実行した後にフィールドキーとフィールド順がマッチングしないフィールドのデータは、実行前の状態で保存されます
FBUPDATE	FBOJOINモードと類似の動作をします。フィールドキーとフィールド順がマッチングするフィールドのデータをsrcフィールド・バッファからdestフィールド・バッファにコピーすることは、FBOJOINモードと似ています。ただし、FBUPDATEモードはフィールドキーとフィールド順がマッチングしないsrcフィールド・バッファにあるフィールドのデータまでもdestフィールド・バッファにコピーされます

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEMALLOC	システムエラーです。フィールド・バッファをメモリーに割り当てることに失敗した場合、fberrorはFBEMALLOCに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
.....
main()
{
    FBUF *destbuf, *srcbuf;
    ...
    destbuf=fmalloc(5,5);
    ...
    fbput(destbuf, INPUT, "5555", 0);
    fbput(destbuf, INPUT, "7777", 0);
    fbput(destbuf, OUTPUT, "3333", 0);
}
```

```

fbput(destbuf, SEQNO, "8888", 0);

srcbuf=fballoc(5,5);
...
fbput(srcbuf, INPUT, "1111", 0);
fbput(srcbuf, OUTPUT, "2222", 0);
fbput(srcbuf, CORPNO, "3333", 0);

/* Modeを選択できるようにします。 */
...
printf("input : ");
scanf("%d", &a);

switch(a)
{
    /*FBMOVE mode */
    case 1:
        iRet=fbbufop(destbuf, srcbuf, FBMOVE);
        ...

        fbprint(destbuf);
        break;
    /* destbuf : "1111", "2222", "3333"*/

    /*FBCOMP mode */
    case 2:
        iRet=fbbufop(destbuf, srcbuf, FBCOMP);
        printf("ret = %d\n", iRet);
        if(iRet < 0) {
            printf("Do not match!!!\n");
            fbfree(destbuf);
            fbfree(srcbuf);
            exit(0);
        }
        printf("Match!!!\n");
        break;
    /* "Do not match!!! */

    /*FBCOPY mode */
    case 3:
        iRet=fbbufop(destbuf, srcbuf, FBCOPY);
        ...
        fbprint(destbuf);
        break;
    /* destbuf : "1111", "2222", "3333"*/

    /*FBCONCAT mode */

```

```

        case 4:
            iRet=fbbufop(destbuf, srcbuf, FBCONCAT);
            ...
            fbprint(destbuf);
            break;
        /* destbuf : "5555", "7777", "3333", "8888", "1111", "2222", "3333"*/

        /*FBJOIN mode */
        case 5:
            iRet=fbbufop(destbuf, srcbuf, FBJOIN);
            ...
            fbprint(destbuf);
            break;
        /* destbuf : "1111", "2222"*/

        /*FBOJOIN mode */
        case 6:
            iRet=fbbufop(destbuf, srcbuf, FBOJOIN);
            ...
            fbprint(destbuf);
            break;
        /* destbuf : "1111", "7777", "2222", "8888"*/

        /*FBUPDATE mode */
        case 7:
            iRet=fbbufop(destbuf, srcbuf, FBUPDATE);
            ...
            fbprint(destbuf);
            break;
        /* destbuf : "1111", "7777", "2222", "8888", "3333"*/

    }
    fbfree(destbuf);
    fbfree(srcbuf);
}

```

- 関連関数

fbbufop_proj()

2.4. fbbufop_proj

特定フィールド・バッファのフィールド・データを指定したフィールド・バッファにコピーする関数です。

srcフィールド・バッファの指定したフィールドのフィールド・データをdestフィールド・バッファにコピーします。関数のdestとsrcフィールド・バッファは同じバッファを使用できず、メモリーに別に割り当てられた他のバッファを使用する必要があります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbbufop_proj(FBUF *dest, FBUF *src, FLDKEY *fldkey)
```

- パラメータ

パラメータ	説明
dest	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです。srcにある内容をmodelに従ってdestに格納します
src	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです。変更される値を持つバッファです。srcをNULLに設定すると、destフィールド・バッファ内部で指定されたフィールドのデータがコピーされます
fldkey	コピーするフィールドを配列形式で指定できます。配列の最後には0を指定する必要があります。fldkeyポインタを配列に宣言した後、その配列にコピーするフィールドを2つ以上指定すると、srcフィールド・バッファからdestフィールド・バッファに、fldkeyに指定したフィールドのデータのみコピーされます

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBEMALLOC	システムエラーです。フィールド・バッファをメモリーに割り当てることに失敗した場合、fberrorはFBEMALLOCに設定されます

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLの場合、該当エラーが発生します

● 例

```
#include "demo_fdl.h"
...

int iRet, i;
FBUF *destbuf, *srcbuf;
FLDKEY fkey[4];

...
fbput(destbuf, INPUT, "aaa", 0);
fbput(destbuf, OUTPUT, "bbb", 0);
fbput(destbuf, SEQNO, "ccc", 0);
...
fbput(srcbuf, TEST1, "AAA", 0);
fbput(srcbuf, TEST2, "BBB", 0);
fbput(srcbuf, TEST3, "CCC", 0);

fkey[0]=fbget_fldkey("INPUT");
fkey[1]=fbget_fldkey("SEQNO");
fkey[2]=0; /* The last entry in the array must be 0 */

ret=fbbufop_proj(destbuf, NULL, fkey);
...
/* destbuf : aaa, ccc */
...
fkey[0]=fbget_fldkey("TEST3");
fkey[1]=fbget_fldkey("TEST1");
fkey[2]=fbget_fldkey("TEST2");

fkey[3]=0; /* The last entry in the array must be 0 */

ret=fbbufop_proj(destbuf, srcbuf, fkey);
...
/* destbuf : AAA, BBB, CCC */
/* fkey array is sorted if the entries are not in numeric order */
...
```

● 関連関数

fbbufop()

2.5. fbcalcsiz

fballoc()またはtpalloc()でフィールド・バッファがメモリーに割り当てられる際にフィールド・バッファのサイズを計算する関数です。

使用するフィールドの数と総データ長(バイト単位)をパラメータに伝達すると、フィールド・バッファのサイズを計算します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
long fbcalcsiz(int count, FLDLEN datalen)
```

- パラメータ

パラメータ	説明
count	データを格納するフィールドの数です
datalen	フィールド・バッファのデータの全体サイズをバイト単位で指定します

- 戻り値

与えられた入力パラメータ(count、datalen)に基づき、フィールド・バッファのサイズをバイト単位で返します。

- 例

```
#include "demo_fdl.h"
. . .

FBUF *fbuf;
long size;

size = fbcalcsiz(100, 1000);
printf("size : [%ld]\n", size); /* output : size : [1408] */
if ((fbuf = (FBUF *)tpalloc("FIELD", NULL, size)) == NULL) {
    printf("tpalloc failed, errno = %d\n", tperrno);
    . . . . .
}

/* fbuf is set in 1024 bytes by default if size is less than 1024*/
. . . . .
```

- 関連関数

fballoc(), tpalloc()

2.6. fbchg_tu

alloc()またはtpalloc()でメモリに割り当てられたフィールド・バッファで、入力パラメータで指定したフィールドキーのフィールド順のデータを変更する関数です。

入力パラメータで指定したフィールド順(nth)に既存のデータが存在しない場合は、フィールド順(nth)に与えられたデータが自動的にフィールドに追加されます。そのため、この関数を使用してフィールド・データを変更する際は関係ありませんが、フィールド順を追加する場合は処理速度が低下するため注意してください。

● プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbchg_tu(FBUF *fbuf, FLDKEY fldkey, int nth, char *value, FLDLEN fldlen)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで変更するフィールドのフィールドキーです
nth	fldkeyで指定したフィールドキーのデータのうち変更するフィールド順です
value	(char *)型のため、変更するフィールドキーが整数型や実数型の場合は必ず(char *)型に型キャストします。valueにNULLが設定された場合、該当フィールドキーを削除します
fldlen	一般的にはフィールドの長さを指定しなくても構いません。フィールドキーのタイプがFB_CARRAYの場合は必ずフィールド・データの長さをバイト単位で指定します

● 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

● エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
...
long rcvlen, ret;
char buffer[100];
FLDLEN fldlen;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", tperno);
    ...
}

fbput(fbuf, INPUT, "aaaa", 0); /* 0番目にはaaaaが存在する。 */
ret = fbchg_tu(fbuf, INPUT, 1, "bbbb ", 4); /* 1番目にbbbbを追加する。 */
if (ret < 0) {
    error processing...
}
ret = fbget_tu(fbuf, INPUT, 1, buffer, &fldlen);
/* 1番目にあるデータをbufferに入れる。 */
if (ret < 0) {
    error processing...
}
printf("Field Data: [%s]\n", buffer);
/* output : Field Data: [bbbb] */

fbprint(fbuf);
/* output : aaaa, bbbb */
.....
```

- 関連関数

fbbufop_proj()

2.7. fbchg_tut

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファで、入力パラメータで指定したフィールドキーのフィールド順のデータをユーザーが指定したタイプからフィールドキーのタイプに変換後、変更する関数です。fbchg_tu()と似ていますが、データ型が変更するフィールドキーのフィールド型と異なる場合、フィールド・データを変更する前に、指定するデータ型をフィールドキーのタイプに変換してから、フィールド・データを変更して格納できるという点が異なります。

入力パラメータで指定したフィールド順に既存のデータが存在しない場合は、フィールド順が与えられたデータに自動的にフィールドが追加されます。そのため、この関数を使用してフィールド順を追加する場合は、処理速度が低下するため注意してください。

● プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbchg_tut(FBUF *fbuf, FLDKEY fldkey, int nth, char *value, FLDLEN fldlen,
              int type)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで変更するフィールドのフィールドキーです
nth	fldkeyで指定したフィールドキーのデータのうち変更するフィールド順です
value	パラメータが(char*)型のため、変更するフィールドキーが整数型や実数型の場合は必ず(char*)型に型キャストします
fldlen	一般的にはフィールドの長さを指定しなくても構いません。しかし、フィールドキーのタイプがFB_CARRAYの場合は必ずフィールド・データの長さをバイト単位で指定します
type	フィールド・データを変換するデータ型で、設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT , FB_DOUBLE , FB_LONG , FB_STRING, FB_CHAR, FB_FLOAT , FB_INT

● 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使できない場合、fberrorはFBENOENTに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します (FB_CARRAYタイプの場合、データ長を指定しなければコードが設定されます)
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	正しくないフィールドが使用されました

- 例

```
#include "demo_fdl.h"
. . .
long ret, lVal=100000;
FBUF *fbuf;
. . .
fbputt(fbuf, INPUT, "aaaa", 0, FB_STRING);
fbchg_tut(fbuf, INPUT, 0, (char *)&lVal, 0, FB_LONG);
printf("1st occurrence of INPUT field is changed as follows : \n");
fbprint(fbuf);
. . . . .
/* fkey = 469762149, fname = INPUT, type = string, value = 100000 */
```

- 関連関数

fbput(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.8. fbdelall

特定フィールドキーに格納されている全体データをフィールド・バッファから削除する関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbdelall(FBUF *fbuf, FLDKEY fldkey)
```

- パラメータ

パラメータ	説明
fbuf	fballloc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから削除するフィールドキーです。fldkeyで指定するフィールドキーは、この関数を呼び出す前に指定したフィールド・バッファにフィールド順のデータが1つ以上格納されている必要があります。そうでない場合は-1を返し、ferrorはFBENOENTに設定されます

- 戻り値

戻り値	説明
削除したフィールドの数	関数の呼び出しに成功しました
-1	関数の実行中にフィールド・バッファに指定したフィールドキーが存在せずにエラーが発生しました（ferrorにエラーコードが設定されます）

- エラー

ferrorには以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE（環境変数：デフォルト値のtmax.fdl）に定義されず、フィールド・バッファで使用できない場合、ferrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;
if ((fbuf = fballloc(10, 100)) == NULL) {
    printf("fballloc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

ret = fbdelall(fbuf, INPUT);
```

```
printf("Delete Field Number: [%d]\n", ret);
/* output: Delete Field Number: [3] */
.....
```

- 関連関数

fbput(), fbget(), fbdelete(), fbdelall_tu()

2.9. fbdelall_tu

フィールドキーの配列に指定されているすべてのフィールドキーに格納されているすべてのフィールド・データを削除する関数です。

パラメータのうちfldkeyに設定されているデータを削除します。複数のフィールドキーを同時に削除する場合、fbdelall()を複数回呼び出すことよりfbdelall_tu()を1回呼び出す方が効率的です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbdelall_tu(FBUF *fbuf, FLDKEY *fldkey)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから削除するフィールドキーを2つ以上指定したフィールドキーの配列です。fieldkeyで指定するフィールドキー・ポインタは配列に宣言し、削除するフィールドキーを明示します。フィールドキーの配列に削除するフィールドキーを指定後、最後の索引には必ず0を指定します

- 戻り値

戻り値	説明
削除したフィールドの数	関数の呼び出しに成功しました
-1	フィールド・バッファが有効でない場合に発生するエラーです。入力パラメータがNULLの場合に発生します (fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	fldkey配列で最後に指定されたフィールドキーに該当するデータがない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey[3];

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, OUTPUT, "cccc", 0);

fkey[0] = INPUT;
fkey[1] = OUTPUT;
fkey[2] = 0; /* The last entry in the array must be 0 */

ret = fbdelall_tu(fbuf, fkey);
printf("The Number of deleted fields: [%d]\n", ret);

/* output: Number of deleted fields: [3] */
. . . . .
```

- 関連関数

fbput(), fbget(), fbdelete(), fbdelall()

2.10. fbdelete

fballoc()またはtpalloc()でメモリーに割り当てたフィールド・バッファから、指定したフィールドキーの指定したフィールド順に該当する特定フィールドのデータを削除する関数です。

fbchg_tu()は、指定したフィールド順に該当するフィールドが存在しない場合、自動的にフィールドを追加します。しかし、fbdelete()は指定したフィールド順に該当するフィールドがフィールド・バッファに存在しない場合にエラーを発生させ、fberrorはFBENOENTに設定されます。そのため、関数を実行前にフィールド・バッファに存在するフィールドをパラメータに指定する必要があります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbdelete(FBUF *fbuf, FLDKEY fldkey, int nth)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから削除するフィールドキーです
nth	削除する特定フィールド順です

● 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	パラメータに指定したフィールドキーの順に該当するフィールドがフィールド・バッファに存在しません(fberrorにエラーコードが設定されます)

● エラー

fberrnoでは、以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます。指定したフィールド順に該当するデータがフィールド・バッファに存在しない場合も、fberrorがFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

● 例

```
#include "demo_fdl.h"
. . .
char buffer[100];
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
```

```

fbput(fbuf, INPUT, "aaaa", 0); /* fbufの0番目のデータは"aaaa"です。 */
fbput(fbuf, INPUT, "bbbb", 0); /* fbufの1番目のデータは"bbbb"です。 */
fbput(fbuf, INPUT, "cccc", 0); /* fbufの2番目のデータは"cccc"です。 */
fbdelete(fbuf, INPUT, 1);      /* fbufの1番目のデータである"bbbb"を削除します。 */

ret=fbdelete(fbuf, INPUT, 3);
if(ret<0){
    printf("fbdelete error : %s\n", fbstrerror(fberror));
}
/* fbufの3番目にはデータが存在しないため
fbdelete error : FBENOENT (not found) が出力されます。 */

fbget(fbuf, INPUT, buffer, 0);
printf("INPUT: [%s]\n", buffer); /* output: INPUT: [aaaa] */
fbget(fbuf, INPUT, buffer, 0);
printf("INPUT: [%s]\n", buffer); /* output: INPUT: [cccc] */
.....

```

- 関連関数

fbput(), fbget(), fbdelall(), fbdelall_tu()

2.11. fbextread

iopで指定する特定ファイルからフィールドキーとそれに該当する値を読み込み、fballoc()またはtpalloc()でメモリーに割り当てたフィールド・バッファに格納する関数です。

内部でフラグも使用しない場合はfbput()を呼び出し、フラグが「+」の場合はfbchg_tu()を、「-」の場合はfbdelete()を、「=」の場合はfbget_tu()を呼び出した後、fbchg_tu()を呼び出します。したがって、このような関数で発生するエラーと同じエラーがfbextread()で発生することがあります。

フラグは4種類あり、demo.fファイルのflagsに設定されます。各フラグの内容は以下のとおりです。

フラグ	説明
+	フィールド・バッファでフィールドキーに該当する値を0番目に格納します
-	フィールド・バッファで該当するフィールドキーの0番目に該当するデータを削除します
=	フィールド・バッファで1番目のフィールド名(field_name1)に該当するデータのうち0番目のフィールド値を、2番目のフィールド名(field_name2)に該当するデータのうち0番目のフィールド値と入れ替えます
#	コメント処理されます
なし	フィールド・バッファに新規フィールドを追加します

フラグ別のフィールドキーと値が格納されているファイルの形式は以下のとおりです。

1. フラグが「+」の場合、あるいはフラグを使用しない場合

```
[flag] field_name or field_id (tab) field_value
```

2. フラグが「=」の場合

```
[flag] field_name1 or field_id1 (tab) field_name2 or field_id2
```

3. フラグが「-」の場合

```
[flag] field_name or field_id
```

● プロトタイプ

```
#include <stdio.h>
#include <usrinc/fbuf.h>

int fbextread(FBUF *fbuf, FILE *iop)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
iop	読み込むフィールドキーとそれに該当する値が格納されているファイルのポインタです

● 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

● エラー

fberrnoには以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合に該当エラーが発生します

● 例

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
```

```

#include "demo_fdl.h"

int main()
{
    long len;
    int ret;
    FILE *fp;
    FBUF *fbuf;
    FLDLEN fldlen;
    char buffer[100];

    fbuf = (FBUF *)tpalloc("FIELD", NULL, 0);
    if (fbuf == NULL) {
        printf("fbuf tpalloc fail[%d]\n", tperrn o);
    }
    ret = fbput(fbuf, INPUT, "user_input", 10);
    if (ret < 0) {
        printf("fbput fail[%d]\n", fberror);
        return -1;
    }
    ret = fbput(fbuf, OUTPUT, "user_output", 10);
    if (ret < 0) {
        printf("fbput fail[%d]\n", fberror);
        return -1;
    }
    ret = fbput(fbuf, STATLIN, "user_statlin", 10);
    if (ret < 0) {
        printf("fbput fail[%d]\n", fberror);
        return -1;
    }
    fp = fopen("file", "r");
    fbprint(fbuf);

    ret = fbextread(fbuf, fp);
    if (ret < 0) printf("Fextread fail[%s]\n", fberror);

    printf("\n\n<after Fextread>-----\n");
    fbprint(fbuf);
    tpfree((char *)fbuf);
}

```

– 「+」フラグを使用してSTSTLINフィールドキーの0番目の値を「statlin0_」に変更します。

```
+STATLIN      statlin0_
```

[結果]

```
user_input, user_output, statlin0_, user_statlin
```

- fbbufバッファにフィールド値を追加します。

```
OUTPUT  OUTPUT0_  
INPUT   INPUT0_  
INPUT   INPUT1_
```

[結果]

```
user_input, user_output, user_statlin, OUTPUT0_, INPUT0_, INPUT1_
```

- 「=」フラグを使用してINPUT値をOUTPUTに格納します。

```
=OUTPUT  INPUT
```

[結果]

```
user_input, user_input, user_statlin
```

- 「-」フラグを使用してINPUTフィールドキーを削除します。

```
-INPUT
```

[結果]

```
user_output, user_statlin
```

- 関連関数

fbput(), fbchg_tu(), fbdelete(), fbget_tu()

2.12. fbfldcount

fballoc()またはtpalloc()でメモリーに割り当てたフィールド・バッファに現在格納されているすべてのフィールドの数を返す関数です。fbfldcount()は、フィールド・バッファ内に格納されているフィールド・データの数を実算します。すでに実装されている複数のAPI関数を利用し、フィールド・バッファに実質的に格納されているフィールドの数も確認できます。1つのフィールドキーに順番を変えて複数のデータが保存されていても、fbfldcount()は別のフィールドと認識します。

- プロトタイプ

```
#include <fbuf.h>  
#include <atmi.h>  
int fbfldcount(FBUF *fbuf)
```

- パラメータ

パラメータ	説明
fbuf	fballloc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです

- 戻り値

戻り値	説明
フィールドの数	関数の呼び出しに成功しました(フィールド・バッファに現在格納されているフィールドの数を返します)
0	関数の呼び出しに失敗しました。あるいはフィールド・バッファに格納されているフィールドが存在しません(fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合、該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .

int cnt;
FBUF *fbuf;

if ((fbuf = fballloc(10, 100)) == NULL) {
    printf("fballloc failed, errno = %d\n", fberror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, OUTPUT, "cccc", 0);
cnt = fbfldcount(fbuf);
printf("Field Count: [%d]\n", cnt);

/* output: Field Count: [3] */
. . .
}
```

- 関連関数

fbgetnth(), fbkeyoccur()

2.13. fbfprint

フィールド・バッファに格納されているフィールド・データを指定したファイル・ストリームに出力する関数です。指定したファイルはテキスト・ファイルで、出力形式はfbprint()と同じです。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbfprint(FBUF *fbuf, FILE *iop)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
iop	出力データを格納するファイル・ストリームを意味します

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数の呼び出しに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoには以下の値のうち1つが設定されます。

エラーコード	説明
FBEBADFB	フィールド・バッファが有効ではありません。現在のフィールド・バッファ以外の、正常にメモリーが割り当てられたバッファを使用する必要があります
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
FILE *iop;
FBUF *fbuf;
```



```

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
}

iop = fopen("sample.text", "w");
fbput(fbuf, INPUT, "5555", 0);
fbput(fbuf, INPUT, "7777", 0);
fbfprint(fbuf, iop);
fclose(iop);

```

- 関連関数

fbprint()

2.14. fbfree

fballoc()またはtpalloc()で割り当てられたフィールド・バッファのメモリーを解除する関数です。関数は、以前に割り当てられたバッファをメモリーから解除する関数であるため、実行後に再度該当バッファを使用することはできません。バッファの使用が完全に終了後にfbfree()を呼び出す必要があります。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbfree(FBUF *fbuf)

```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()で割り当てられたフィールド・バッファです

- 戻り値

戻り値	説明
1	バッファを正常に解除しました
-1	フィールド・バッファのメモリーを解除できません (ferrorにエラーコードが設定されます)

- エラー

fbernoには以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
ret = fbfree(fbuf);
printf("fbfree, ret = %d\n", ret);
/* output : fbfree, ret = 1 */
fbput(fbuf, INPUT, "bbbb", 0);
fbprint(fbuf);
/* an error occurs since memory was not allocated. */
```

- 関連関数

fbcalcsz(), tpalloc(), fballoc(), fbinit()

2.15. fbftos

フィールド・バッファーに格納されているデータを、マッチングする構造体バッファーに送信する関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbftos(FBUF *fbuf, char *cstruct, char *stname)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーです
cstruct	使用する構造体タイプの変数を(char *)型に型キャストして使用します
stname	VIEWで定義した構造体の名前を文字列で指定します

- 戻り値

戻り値	説明
1	フィールド・バッファに格納されているデータをC構造体へ送信することに成功しました
-1	フィールド・バッファに格納されているデータをC構造体へ送信することに失敗しました (ferrorにエラーコードが設定されます)

- エラー

fbernoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl) に定義されておらず、フィールド・バッファで使用できない場合、ferrorはFBENOENTに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、ferrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、ferrorはFBEBADFLDに設定されます
FBEBADSTRUCT	有効でない構造体を使用されました。一般的に、Tmaxで認識できない形式の構造体を使用した場合や、構造体定義ファイルが正しくコンパイルされなかった場合に発生するエラーです

- 例

```
#include "demo_fdl.h"
#include "demo_sdl.h"
. . .
int ret, fdllen, val = 1000;
struct demo temp;
FBUF *fbuf;
Char *cstruct;

if(tpstart((TPSTART_T *)NULL) == -1) {
    printf("tpstart failed\n");
    . . . . .
}

if ((fbuf = fballocc(10, 100)) == NULL) {
    printf("fballocc failed, errno = %d\n", ferror);
}
```

```

.....
}

cstruct = (char *)&temp;
if(fbstinit((char *)&temp, "demo") < 1) {
    printf("fbstinit failed, errno = %d\n", ferror);
    .....
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INTDATA, (char *)&val, 0);

if(fbftos(fbuf, cstruct, "demo") < 0) {
    .....
}
.....

```

- 関連関数

fbstof(), fbstinit(), fbstelinit()

- 参照

fbftos()は、fballoc()またはtpalloc()を使用してメモリーに割り当てられたフィールド・バッファに格納されているフィールド・データを、マッチングするC構造体バッファ(stname)に送信する関数です。

一般構造体ファイルの***.sとは若干異なる構造体ファイルが事前に定義されている必要があります。構造体ファイル内のフィールドキーは、定義されている構造体のメンバー変数と1:1でマッピングされている必要があります。

以下は、フィールド・バッファとC構造体をマッピングさせる構造体ファイルの例です。

```

VIEW demo
#type      cname      Fldkey      count      flag      size      null
string     demodata    INPUT       5          -         20        ""
carray     carraydata  CARRAYDATA  1          -         20        "star"
char       chardata    CHARDATA    1          -         -         "c"
int         inum       INTDATA     5          -         -         0
short      snum       SHORTDATA   1          -         -         0
long       lnum       LONGDATA    1          -         -         1
float      fnum       FLOATDATA   1          -         -         1.0
double     dnum       DOUBLEDATA  1          -         -         1.0
END

```

項目	説明
type	構造体内のメンバー変数のデータ型を意味します

項目	説明
cname	メンバー変数の名前を意味します
Fldkey	決定したメンバー変数とマッピングされるフィールドキーを意味します(フィールドキーは「***.f」に定義されている必要があります)
count	構造体で格納できる最大のフィールド順を意味します
size	string型の場合の配列のサイズを意味します
null	初期化される際の値を示します。フィールド・バッファに値が格納されていない状態でフィールド・データをC構造体のバッファに送信する場合、基本的にC構造体にコピーされる値です。null項目に値を指定せず、フィールド・バッファにも値が格納されていない場合、各データ型に適切なデフォルト値がC構造体に格納されます。string型、carray型、char型のデフォルト値はNULLで、int型、short型、long型のデフォルト値は0です。そして、float型、double型のデフォルト値は0.0です

demo.sまたはdemo.vにVIEWを定義した後、クライアントではsdlcユーティリティを使用してVIEWタイプを以下のようにコンパイルします。

```
$TMAXDIR$/sample/sdl> sdlc -c -v demo.s -o tmax.sdl
$TMAXDIR$/sample/sdl> sdlc -c -v demo.v -o tmax.sdl
```

環境ファイル(コンソールを使用する場合は.profile)に、-oオプションを使用して、SDLFILEに指定された名前で.sdlファイルを作成します。このようにすることで、フィールドキーと構造体間の変換情報がSDLFILEに含まれます。上のVIEWファイルに定義されているフィールドキーは「***.f」に定義されている必要がある点に注意してください。

VIEWファイルをコンパイルすると、demo_sdl.hとtmax.sdlファイルが作成されます。クライアント・プログラムでは、demo_sdl.hファイルを含んで使用します。サーバーでは、sdlcユーティリティを使用して、VIEWファイルを以下のようにコンパイルします。

```
$TMAXDIR$/sample/sdl> sdlc -v demo.s
```

構造体ファイルをコンパイルすると、demo_sdl.cとdemo_sdl.hファイルが作成されます。フィールドキーと構造体間の変換情報がdemo_sdl.cファイルにあるため、この情報を使用する必要があるサーバー・プログラムをコンパイルする際、demo_sdl.cファイルと一緒にコンパイルされる必要があります。サーバー・プログラム内では、コンパイル時に必要な情報をすべて持っているため、tmax.sdlファイルが必要ありません。

フィールド・バッファで構造体にデータを送信する際に、フィールドキーと構造体のメンバー変数のデータとマッピングしない場合は、フィールドキーが無視されます。

2.16. fbget

フィールド・バッファで指定したフィールドキーに該当するフィールド・データを読み込む関数です。

データをフィールド・バッファから読み込む関数で、同じフィールドキーに格納されたデータは順番に読み込んでlocに格納します。つまり、手動で順番を増加させず、格納されたフィールド順だけこの関数を呼び出すと、関数内で自動的にフィールド順を増加させ、次のデータを読み込む準備がされます。そのため、該当フィールドキー内に格納されているすべてのフィールド・データを読み込むことができます。fbget()を使用する場合、1度読み込んだデータは再度読み込むことができないため、注意してください。そして、格納されたフィールド順より多くのfbget関数を呼び出すと、最後にフィールド順のフィールド・データが継続して返されます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbget(FBUF *fbuf, FLDKEY fldkey, char *loc, FLDLEN *fldlen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから読み込むフィールドキーを意味します
loc	順番に読み込んだフィールド・データを示すポインタです。フィールドキーが整数型または実数型の場合は、必ず各タイプに適切な変数を使用しますが、前で記述した関数のプロトタイプに適切に型キャストします
fldlen	読み込んだデータの長さを意味します

- 戻り値

戻り値	説明
1	指定したフィールドキーのフィールド・データが見つかりました
-1	指定したフィールドキーをフィールド・バッファで見つけることができませんでした (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します (CARRAY 型の場合、fldlen を指定していない場合も該当します)
FBETYPE	Tmax でサポートしていないタイプが使用されました

- 例

```
#include "demo_fdl.h"
. . .
long rcvlen, ret;
FLDLEN fldlen;
char buffer[100];
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

fbget(fbuf, INPUT, buffer, &fldlen);
printf("Field Data: [%s]\n", buffer);
/* output: Field Data: [aaaa] */
. . .
```

- 関連関数

fbput(), fbchg_tu(), fbget_tu(), fbchg_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.17. fbgetalloc_tu

指定したフィールドキーのフィールド順に該当するフィールド・データを読み込む関数です。

fldkey および nth パラメータに指定したフィールドキーのフィールド順に該当するフィールド・データを読み込み、extralen に指定した長さ分の新規バッファを割り当て、該当バッファにデータを格納し、ポインタを返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetalloc_tu(FBUF *fbuf, FLDKEY fldkey, int nth, int *extralen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから読み込むフィールドキーを意味します
nth	データを読み込むフィールドキーのフィールド順を意味します
extralen	新しく割り当てを受ける必要があるバッファのサイズを意味します。新しく割り当てられたバッファは、使用後にユーザーが必ずメモリーを解除します。extralenを0に指定すると、関数の実行後、新しく割り当てる必要があるバッファのサイズがフィールド・データの長さに設定されます。extralenは割り当てたバッファのサイズが設定されます

- 戻り値

戻り値	説明
バッファのポインタ	関数の呼び出しに成功しました
NULL	フィールド・バッファで指定したフィールドキーが見つかりませんでした。あるいは、関数の実行中に他のエラーが発生しました(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBEMALLOC	システムエラーです。フィールド・バッファをメモリーに割り当てられなかった場合、fberrorはFBEMALLOCに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました

- 例

```
#include "demo_fdl.h"
...
long ret;
int extralen;
char *buffer;
FBUF *fbuf;
```



```

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

extralen = 4;
buffer = fbgetalloc_tu(fbuf, INPUT, 0, &extralen);
printf("Field Data: [%s]\n", buffer);
/* output: Field Data: [aaaa], returned after adding 4 bytes to
the length of the field value. */
fbfree(fbuf);
...

```

- 関連関数

fbgetalloc_tut(), fbput(), fbchg_tu(), fbget_tu(), fbchg_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.18. fbgetalloc_tut

フィールド・バッファで指定したフィールドキーのフィールド順に該当するフィールド・データを、指定したタイプで必要な長さ分のバッファを新しく割り当てを受けて格納し、返す関数です。

この関数を使用すると、fldkeyおよびnthパラメータに指定したフィールドキーのフィールド順に該当するフィールド・データを読み込みます。フィールド・データのタイプとは関係なく、指定したタイプに変換し、extralenに指定した長さ分の新しいバッファを割り当て、そしてデータを格納し、ポインタを返します。

fbgetalloc_tut()はfbgetalloc_tu()とよく似ていますが、取得するフィールド・データのタイプを変換できるという点が異なります。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
char *fbgetalloc_tut(FBUF *fbuf, FLDKEY fldkey, int nth, int type, int *extralen)

```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから読み込むフィールドキーを意味します

パラメータ	説明
nth	データを読み込むフィールドキーのフィールド順を意味します
type	フィールド・データを変換するデータ型です。設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT , FB_DOUBLE , FB_LONG , FB_STRING, FB_CHAR, FB_FLOAT , FB_INT
extralen	新しく割り当てを受ける必要があるバッファのサイズをバイト単位で示します。割り当てられたバッファは、使用後にユーザーが必ずメモリーを解除します。extralenを0に指定すると、関数の実行後、新しく割り当てる必要があるバッファのサイズがフィールド・データの長さに設定されます

- 戻り値

戻り値	説明
バッファのポインタ	関数の呼び出しに成功した場合です。 フィールド・データを保存した新しく割り当てられたバッファのポインタを返します。戻り値は(char *)型で表現できない、int型、long型、double型、long型、float型の場合には状況に合わせて型キャストします
NULL	フィールド・バッファで指定したフィールドキーが見つからなかった場合です。あるいは、関数の実行中に他のエラーが発生した場合です (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl) に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBEMALLOC	システムエラーです。フィールド・バッファをメモリーに割り当てることに失敗した場合、fberrorはFBEMALLOCに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます
FBETYPE	Tmaxでサポートしていないタイプが使用されました

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに設定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
long ret;
int extralen;
char *buffer;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

extralen = 4;
buffer = fbgetalloc_tut(fbuf, INPUT, 0, FB_STRING, &extralen);
printf("Field Data: [%s]\n", buffer);
/* output: Field Data: [aaaa], returned after adding 4 bytes to the length of
the
field value. */

fbfree(fbuf);
. . .
```

- 関連関数

fbgetalloc_tu (), fbput(), fbget_tu(), fbchg_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.19. fbget_fbsize

fballoc()またはtpalloc()で割り当てられたフィールド・バッファのサイズを返す関数です。

fballoc()またはtpalloc()を使用して、メモリーに1024バイトより小さいフィールド・バッファを割り当てる場合、基本的にメモリーは1024バイトを割り当てます。このような場合、Tmax内部で使用されるヘッダー・サイズを加えて、fbget_fbsize()は1024 + 8 = 1032バイトを返します。これ以外の場合、fbget_fbsize()の戻り値はfbcalcsz()で計算した結果にヘッダー・サイズ(8バイト)を加えた値と同じです。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
long fbget_fbsize(FBUF *fbuf)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです

- 戻り値

戻り値	説明
フィールド・バッファのサイズ	関数の呼び出しに成功しました(メモリーに割り当てられたフィールド・バッファのサイズを返します)

- エラー

fberrnoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
long ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
fbput(fbuf, INPUT, "1111", 0);
ret = fbget_fbsize(fbuf );
printf("Field Buffer Size: [%ld]\n", ret);
/* output: Field Buffer Size: [1024] */
. . .
```

- 関連関数

fbcalcsz(), tpalloc(), fballoc(), fbfree(), fbinit()

2.20. fbget_fldkey

fdlcユーティリティを使用してコンパイルして作成されたフィールド表(FDLFILEに設定されたFDLファイル)と関連し、指定したフィールド名とマッチングする識別子を返す関数です。この識別子をフィールドキーといいます。フィールド・バッファに格納されるデータの形式はフィールドキーとフィールドデータの組み合わせであるため、フィールド・バッファにデータを格納するためには指定したフィールド名に該当するフィールドキーを知る必要があります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
FLDKEY fbget_fldkey(char *name)
```

- パラメータ

パラメータ	説明
name	フィールド表を作成するのに使用されるフィールド名です

- 戻り値

戻り値	説明
0	指定したフィールド名がフィールド表に存在しません
フィールド名とマッチングするフィールドキー	指定したフィールド名がフィールド表に存在します

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey;
FBUF *sndbuf;

fkey = fbget_fldkey("INPUT");
printf("Field Key : [%d]\n", fkey);

/* output : Field Key : [469762149] */
. . . . .
fbput(sndbuf, fkey, "1111", 0);
. . .
```

- 関連関数

fbget_fldname(), fbget_fldno(), fbget_fldtype(), fbget_strfldtype(), fbtypecvrt()

2.21. fbget_fldname

fbget_fldkey()とは反対に、fdlcユーティリティを使用してコンパイルして作成されたフィールド表(FDLFILEに設定されたFDLファイル)と関連し、指定したフィールドキーとマッチングするフィールド名を返す関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbget_fldname(FLDKEY fldkey)
```

- パラメータ

パラメータ	説明
fldkey	フィールド表を作成するために必要なデータを定義したファイル(***.f)を、fdlcユーティリティを使用してコンパイルして作成されたヘッダーファイル(***_fdl.h)あるいはfbget_fldkey()を使用して取得したフィールドキーです

- 戻り値

戻り値	説明
フィールド名とマッチングするフィールドキー	指定したフィールドキーのフィールド名がフィールド表に存在します
NULL	指定したフィールドキーのフィールド名がフィールド表に存在しません

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey;
char *name;

fkey = fbget_fldkey("INPUT");
printf("Field Key : [%d]\n", fkey);
/* output : Field Key : [469762149] */

name = fbget_fldname(fkey);
printf("Field Name : [%s]\n", name);
/* output : Field Name : [INPUT] */
. . .
```

- 関連関数

fbget_fldkey(), fbget_fldno(), fbget_fldtype(), fbget_strfldtype(), fbtypecvrt()

2.22. fbget_fldno

フィールド表を作成するために必要なデータのうち、指定したフィールドキーとマッチングするフィールド番号を返す関数です。フィールド表を作成するためには、フィールド型、フィールド名、フィールド番号などが必要で、これらの組み合わせで一意的なフィールドキーを作成できます。このようなフィールドキーを集めて管理するのがフィールド表です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbget_fldno(FLDKEY fldkey)
```

- パラメータ

パラメータ	説明
fldkey	フィールド表を作成するために必要なデータを定義したファイル(***.f)を、fdlcユーティリティを使用してコンパイルして作成されたヘッダー・ファイル(**_fdl.h)あるいはfbget_fldkey()を使用して取得したフィールドキーです

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファーで指定したフィールドキーが見つかりませんでした。あるいは、関数の実行中に他のエラーが発生しました(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファーで使用できない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	正しくないフィールドを使用しました

- 例

```
#include "demo_fdl.h"
. . .
```

```

long ret;
FLDKEY len;
int nth;
char buffer[100];
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

fbgetlast_tu(fbuf, INPUT, &nth, buffer, &len);
printf("Field Data: [%s], occurrence = %d, len = %d\n", buffer, nth, len);
/* output: Field Data: [cccc], occurrence = 2, len = 4 */
. . .

```

- 関連関数

fbput(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetalloc_tu()

2.23. fbget_fldtype

指定したフィールドキーのフィールド型を整数で返す関数です。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbget_fldtype(FLDKEY fldkey)

```

- パラメータ

パラメータ	説明
fldkey	フィールド表を作成するために必要なデータを定義したファイル(***.f)を、fdlcユーティリティーを使用してコンパイルして作成されたヘッダー・ファイル(**_fdl.h)あるいはfbget_fldkey()を使用して取得したフィールドキーです

- 戻り値

戻り値は1から8までのint値です。関数で返されるフィールド型とそれにマッチングする戻り値は以下のとおりです。

戻り値	フィールド型
1	character
2	short integer
3	integer
4	long integer
5	float
6	double
7	string
8	character array(carray)

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey;
int type;

fkey = fbget_fldkey("INPUT");
printf("Field Key : [%d]\n", fkey);/* output: Field Key : [469762149] */
type = fbget_fldtype(fkey);
printf("Field Type : [%d]\n", type);/* output : Field Type : [7] */
. . .
```

- 関連関数

fbget_fldname(), fbget_fldno(), fbget_fldkey(), fbget_strfldtype(), fbtypecvrt()

2.24. fbgetlast_tu

フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、順番が最後のデータを返す関数です。返されるデータがフィールド・データのうち何番目であるかと、フィールド・データの長さを、データと一緒に返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbgetlast_tu(FBUF *fbuf, FLDKEY fldkey, int *nth, char *loc, int *len)
```

- パラメータ

パラメータ	説明
fbuf	ballocc()またはtpallocc()でメモリーに割り当てられたフィールド・バッファーです
fldkey	フィールド・バッファーで読み込むフィールドキーを意味します
nth	指定したフィールドキーで格納されているフィールド・データのうち、順番が最後に格納されているポインタでデータを読み込むフィールドキーのフィールド順を格納します
loc	取得したフィールド・データを格納するポインタです
len	取得したフィールド・データの長さを格納するポインタです

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファーで指定したフィールドキーを見付けられませんでした。あるいは、関数の実行中に他のエラーが発生しました (ferrorにエラーコードが設定されます)

- エラー

fbernoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファーで使用できない場合、ferrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	正しくないフィールドを使用しました

- 例

```
#include "demo_fdl.h"
. . .
long ret;
FLDKEY len;
int nth;
char buffer[100];
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
```

```

    printf("fballoc failed, errno = %d\n", ferror);
    .....
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "cccc", 0);

fbgetlast_tu(fbuf, INPUT, &nth, buffer, &len);
printf("Field Data: [%s], occurrence = %d, len = %d\n", buffer, nth, len);
/* output: Field Data: [cccc], occurrence = 2, len = 4 */
. . .

```

- 関連関数

fbput(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetalloc_tu()

2.25. fbget_strfldtype

指定したフィールドキーに格納されるフィールド・データのタイプを文字列に変換する関数です。fbget_fldtype() は指定したフィールドキーのフィールド型を予め指定した整数で返しますが、この変数はフィールド型をそのまま文字列で返します。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
char *fbget_strfldtype(FLDKEY fldkey)

```

- パラメータ

パラメータ	説明
fldkey	フィールド表を作成するために必要なデータを定義したファイル(***.f)を、fdlcユーティリティを使用してコンパイルして作成されたヘッダーファイル(***_fdl.h)あるいはfbget_fldkey()を使用して取得したフィールドキーです

- 戻り値

戻り値	説明
空文字列("")	パラメータに指定したフィールドキーに該当するフィールド型が存在しません。あるいは、フィールドキーが正しく指定されませんでした
文字列(指定したフィールドキーのフィールド型)	パラメータに指定したフィールドキーに該当するフィールドキーが存在します

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey;
char *stype;
.....
fkey = fbget_fldkey("INPUT");
printf("Field Key : [%d]\n", fkey);
/* output : Field Key : [469762149] */

stype = fbget_strfldtype(fkey);
printf("Field Type : [%s]\n", stype);
/* output : Field type : [string] */
```

- 関連関数

fbget_fldname(), fbget_fldno(), fbget_fldtype(), fbtypecvrt()

2.26. fbget_tu

フィールド・バッファで指定したフィールドキーの指定したフィールド順に該当するフィールドのデータを読み込む関数です。fldkeyでは、指定したフィールドキーのvalueで指定したフィールド順に該当するフィールドのデータを読み込みます。

fbget_tu()と似ているfbget()は、読み込むフィールドキーのみを指定すると、そのフィールドキーに格納されているフィールド・データを最初から順番を増加させながら読み込みます。これに対し、fbget_tu()は指定したフィールドキーに格納されているデータのうちフィールド順までも指定できます。また、fbget()の場合は1回読み込んだデータにそれ以上アクセスできませんが、fbget_tu()はフィールド順まで指定できるため、1つのフィールド・データを複数回読み込むことができます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbget_tu(FBUF *fbuf, FLDKEY fldkey, int nth, char *value, FLDLEN *len)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーを意味します
nth	読み込むフィールド順を示します

パラメータ	説明
value	読み込んだフィールド・データを示すポインタです。フィールドキーが整数型または実数型の場合、必ず各タイプに適切な変数を使用しますが、前で記述した関数のプロトタイプに適切に型キャストします
len	読み込んだデータの長さを示します

- 戻り値

戻り値	説明
1	指定したフィールドキーがフィールド・バッファに存在します
-1	指定したフィールドキーがフィールド・バッファに存在しません (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl) に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	正しくないフィールドを使用しました

- 例

```
#include "demo_fdl.h"
. . .
char buffer[100];
long rcvlen;
FLDLLEN fldlen;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, OUTPUT, "cccc", 0);
```

```
fbget_tu(fbbuf, INPUT, 1, buffer, &fldlen);
printf("Data: [%s]\n", buffer) /* output: Data: [bbbb] */
.....
```

- 関連関数

fbput(), fbget(), fbget_tut(), fbgetval(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.27. fbget_tut

指定したフィールドキーのフィールド順に該当するフィールド・データを指定したタイプに変換して返す関数です。fldkeyで指定したフィールドキーのnthで指定したフィールド順に該当するフィールドのデータを読み込みます。

fbget_tu()とほぼ同じ役割をしますが、fbget_tu()は指定したフィールドキーのフィールド順に該当するフィールド・データをそのまま読み込み、fbget_tut()はフィールド・データを指定したタイプに変換して読み込むことができますという点が異なります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbget_tut(FBUF *fbbuf, FLDKEY fldkey, int nth, char *value, FLDLEN *len,
              int type)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーを意味します
nth	読み込むフィールド順を意味します
value	読み込んだフィールド・データを示すポインタです。フィールドキーが整数型または実数型の場合は、必ず各タイプに適切な変数を使用しますが、前で記述した関数のプロトタイプに適切に型キャストします
len	読み込んだデータの長さを意味します
type	読み込んだデータ型を変換する場合、データ型で設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_FLOAT, FB_INT

- 戻り値

戻り値	説明
1	指定したフィールドキーがフィールド・バッファに存在します
-1	指定したフィールドキーがフィールド・バッファに存在しません (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl) に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	正しくないフィールドを使用しました
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに設定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
FBUF *fbuf;
int ret, val=65;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

fbputt(fbuf, INPUT, (char *)&val, 0, FB_INT);
fbputt(fbuf, INTDATA, "1111", 0, FB_STRING);
fbprint(fbuf);

fbget_tut(fbuf, INTDATA, 0, (char *)&ret, &fldlen, FB_INT);
printf("Field Data: [%d]\n", ret);
/* output: Field Data: [1111] */
. . . . .
```

- 関連関数

fbput(), fbget(), fbget_tu(), fbgetval(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.28. fbget_unused

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーのうち、まだ使用されていないフィールド・バッファーのサイズを計算し、バイト単位で返す関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
long fbget_unused(FBUF *fbuf)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーです

- 戻り値

メモリーに割り当てられたフィールド・バッファーのうち、まだ使用されていないメモリーのサイズをバイト単位で返します。

- エラー

fbererrnoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
long ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "1111", 0);
ret = fbget_unused (fbuf);
printf("Unused Buffer Size: [%d]\n", ret);

/* output: Unused Buffer Size: [1000] */
. . .
```


- 関連関数

fbcalcsz(), fbget_fbsize(), fbget_used()

2.29. fbget_used

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファのうち、現在使用中のフィールド・バッファのサイズを計算し、バイト単位で返す関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
long fbget_used(FBUF *fbuf)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです

- 戻り値

メモリーに割り当てられたフィールド・バッファのうち、現在使用中のメモリーのサイズをバイト単位で返します。

- エラー

fbererrnoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータで指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
long ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "1111", 0);
```

```
ret = fbget_used(fbuf);
printf("Used Buffer Size: [%d]\n", ret);

/* output: Used Buffer Size: [24] */
. . .
```

- 関連関数

fbcalcsz(), fbget_fbsize(), fbget_unused()

2.30. fbgetf

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーに格納されているフィールド・データを順番に読み込む関数です。

fbgetf()はfbget()と同じ機能を行う関数ですが、内部的に以下のフィールドの位置を格納しているため、fbget()より実行速度が格段に速いです。Microsoft Visual Basicで20,000個のフィールド・データを受け取る場合、fbget()を使用すると400秒かかりますが、fbgetf()を使用すると2.5秒ですみます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbgetf(FBUF *fbuf, FLDKEY fldkey, char *loc, int *fldlen, int *pos)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーです
fldkey	データを取得するフィールドキーを意味します
loc	読み込んだデータが返すポインタです
fldlen	読み込んだデータの長さが返されます
pos	読み込む、次のデータの位置を返すポインタです。posはfbgetf()を使用するフィールドキー毎に別途指定する必要があり、fbgetf()を呼び出す前に必ず0に初期化します

- 戻り値

戻り値	説明
1	指定したフィールドキーのフィールド・データを正常に読み込みました
-1	指定したフィールドキーのフィールド・データを正常に読み込めませんでした (fberror にエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使えない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしないタイプが使用されました

- 例

```
#include "demo_fdl.h"
. . .
int ret, int0_pos, str0_pos, fldlen;
char str0[30];
FBUF *fbuf;

if ((fbuf = fballoc(20000, 1000000)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

int0_pos = str0_pos = 0;
while(1)
{
    ret = fbgetf(fbuf, STR0, (char *)str0, (int *)&fldlen, &str0_pos);
    if(ret < 0)
    {
        printf("FBERROR at STR0:%d\n", fberror);
        break;
    }
    ret = fbgetf(fbuf, INT0, (char *)&int0, (int *)&fldlen, &int0_pos);
    if(ret < 0)
    {
        . . . . .
    }
    . . . . .
}
```

- 関連関数

fbget(), fbgetval(), fbget_tu(), fbgetvals(), fbgetvali(), fbgetvalt(), fbgetvall_tu()

2.31. fbgetlen

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーに指定したフィールドキーのフィールド順に該当するフィールド・データの長さを返す関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbgetlen(FBUF *fbuf, FLDKEY fldkey, int nth)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーです
fldkey	データを取得するフィールドキーを意味します
nth	データの長さを確認しようとするフィールド順を意味します

- 戻り値

戻り値	説明
フィールド・データの長さ	フィールド・バッファーで指定したフィールドを見つけました
-1	フィールド・バッファーで指定したフィールドが見つかりませんでした (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数 : デフォルト値のtmax.fdl) で定義されず、フィールド・バッファーで使用できない場合、fberrorはFBENOENTに設定されます
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```

#include "demo_fdl.h"
. . .
int ret;
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
if(fbgetlen(fbuf, INPUT, 1) < 0)
{
    . . . . .
}
. . . . .

```

- 関連関数

fbget_fsize(), fbget_used(), fbget_unused(), fbcalcsz()

2.32. fbgetnth

フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したデータが格納されているフィールドの数を返す関数です。データが数回格納されている場合、該当フィールドの数のうち最も小さい値が返されます。

- プロトタイプ

```

#include <fbuff.h>
#include <atmi.h>
int fbgetnth(FBUF *fbuff, FLDKEY fldkey, char *value, FLDLEN fldlen)

```

- パラメータ

パラメータ	説明
fbuff	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファでデータを比較するフィールドキーです
value	フィールド・データと比較するデータを示すポインタです。フィールドキーが整数型または実数型の場合は、必ず各タイプに適切な変数を使用しますが、前で記述した関数のプロトタイプに適切に型キャストします
fldlen	データの長さを示すのに、指定したフィールドキーのタイプがFB_CARRAYではない場合は、0に指定しても構いません

- 戻り値

戻り値	説明
指定したデータのフィールド順	指定したフィールドキーのフィールド順がフィールド・バッファに存在します
-1	指定したフィールドキーのフィールド順がフィールド・バッファに存在しません (ferrorにエラーコードが設定されます)

- エラー

fbernoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
int occ;
FBUF *fbuf;
. . . .
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "dddd", 0);
fbput(fbuf, INPUT, "eeee", 0);
fbput(fbuf, OUTPUT, "cccc", 0);
occ = fbgetnth(fbuf, INPUT, "eeee", 0);
printf("Occurrence: [%d]\n", occ); /* output: Occurrence: [3] */
. . . .
```

- 関連関数

fbfldcount(), fbkeyoccur(), fbgetntht()

2.33. fbgetntht

指定したフィールドキーに格納されているフィールド・データのうち、指定したデータが格納されているフィールド順を返す関数です。fbgetntht()はfbgetnth()と非常に似ていて、fldkeyに指定したフィールドキーに格納されているフィールド・データのうちvalueに指定したデータが格納されているフィールド順を返します。

ユーザーがfromtypeで指定したデータ型とfldkeyで指定したフィールドキーのフィールド型が同じでない場合、valueのデータ型をフィールド型に変換後、value、fldlen、フィールド型が一致するフィールド順を検索して返す点がfbgetnth()と異なります。fbgetntht()はvalueとマッチングされる1番目のフィールド順を返します。

● プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbgetntht(FBUF *fbuf, FLDKEY fldkey, char *value, FLDLEN fldlen, int fromtype)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファでデータを比較するフィールドキーです
value	フィールド・データと比較するデータを示すポインタです。フィールドキーが整数型または実数型の場合は、必ず各タイプに適切な変数を使用しますが、前で記述した関数のプロトタイプに適切に型キャストします
fldlen	データの長さを示すのに指定したフィールドキーのタイプがFB_CARRAYでない場合は0に指定しても構いません
fromtype	ユーザーが指定するvalueのデータ型で、以下の値を設定できます – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_FLOAT, FB_INT

● 戻り値

戻り値	説明
指定したデータのフィールド順	指定したフィールドキーのフィールド順がフィールド・バッファに存在します
-1	指定したフィールドキーのフィールド順がフィールド・バッファに存在しません (fberrorにエラーコードが設定されます)

● エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE (環境変数: デフォルト値のtmax.fdl)に定義されていないため、フィールド・バッファで使えない場合、fberrorはFBENOENTに設定されます

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータで指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティーを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます

- 例

```
#include "demo_fdl.h"
. . .
int occ, val;
FBUF *fbuf;
. . . .
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

val = 1111;
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "dddd", 0);
fbput(fbuf, INPUT, "1111", 0);
fbput(fbuf, OUTPUT, "cccc", 0);

occ=fbgetntht(fbuf, INPUT, (char *)&val, 0, FB_INT);
printf("Occurrence: [%d]\n", occ); /* Occurrence: [2] */
. . .
```

- 関連関数

fbfldcount(), fbkeyoccur(), fbgetnth()

2.34. fbgetval

フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したフィールド順に該当するフィールド・データを返す関数です。fldkeyで指定したフィールドキーに格納されているフィールド・データのうち、nthパラメータで指定したフィールド順に該当するフィールド・データを返します。

fbgetval()は、fbget_tu()とほぼ同じ役割をする関数ですが、若干の違いがあります。fbget_tu()は、読み込むデータが関数のパラメータを使用して返しますが、fbgetval()は関数自体の戻り値で読み込むデータを返

します。関数が返すバッファ内のポインタは関連関数用でのみ使用可能で、バッファの内容の修正には使用できません。fbgetval()で返されるアドレス値は、該当バッファが修正されていない限り有効です。また、この値は、ユーザーが指定したタイプのパラメータに従わないこともあります。したがって、longやdouble型などについてキャッシュして該当値を使用する場合は正しいデータ値を保証できず、システムエラーが発生することがあります。

このような関連関数が必要であればfbgetval()を使用します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetval(FBUF *fbuf, FLDKEY fldkey, int nth, FLDLEN *fldlen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーです
nth	読み込むフィールド順を示します
fldlen	読み込んだデータの長さを意味します

- 戻り値

戻り値	説明
データに関するポインタ	指定したフィールドキーのフィールド順がフィールド・バッファに存在します
NULL	指定したフィールドキーのフィールド順がフィールド・バッファに存在しません (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されておらず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```

#include "demo_fdl.h"
. . .

char *ptr;
long rcvlen;
FLDLLEN fldlen;
FBUF *fbuf;

fbuf = fballloc(10, 100);
. . . . .

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);

ptr=fbgetval(fbuf, INPUT, 0, &fldlen);
printf("Data: [%s]\n", ptr);    /* output: Data: [aaaa] */
printf("len = %d\n", fldlen);  /* len = 4 */
. . . . .

```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.35. fbgetvali

フィールド・バッファで指定したフィールドキーに格納されているデータのうち、指定したフィールド順に該当するフィールド・データを整数に変換し、その整数値を返す関数です。

fldkeyで指定したフィールドキーに格納されているデータのうち、nthで指定したフィールド順に該当するフィールド・データを整数に変換して返します。指定したフィールドキーのタイプがどんなタイプであっても整数に変換します。

しかし、整数に変換できない場合、たとえばフィールド・データが「aaaa」の場合は0を返します。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbgetvali (FBUF *fbuf, FLDKEY fldkey, int nth)

```

- パラメータ

パラメータ	説明
fbuf	fballloc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーです

パラメータ	説明
nth	読み込むフィールド順です

- 戻り値

戻り値	説明
整数	関数の呼び出しに成功した場合です (fbgetvali()) は、読み込んだデータを整数に変換して返します)
0	指定したフィールドキーのフィールド順がフィールド・バッファに存在しません。あるいは、関数の実行中に他のエラーが発生しました (fberror にエラーコードが設定されます)

- エラー

fberrno に以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberror は FBENOSPACE に設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmax でサポートしていないタイプが使用されました
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーが fdlc ユーティリティーを使用してコンパイルされていないフィールドキーが使用された場合、fberror は FBEBADFLD に設定されます
FBENOENT	フィールドキーのフィールド順に該当するデータがフィールド・バッファに存在しない場合、fberror は FBENOENT に設定されます

- 例

```
#include "demo_fdl.h "
. . .

int cnt;
int vali;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

cnt=111;
```

```
fbput(fbuf, TOTMON, (char *)&cnt, 0);
fbput(fbuf, INPUT, "aaaa", 0);
vali=fbgetvali(fbuf, TOTMON, 0);
printf("Field Data: [%d]\n", vali); /* output: Field Data: [111] */

vali = fbgetvali(fbuf, INPUT, 0);
printf("Field Data : [%d]\n", vali); /* output: Field Data: [0] */

fbfree(fbuf);
.....
```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.36. fbgetval_last_tu

fldkeyで指定したフィールドキーに格納されているフィールド・データのうち、順番が最後のデータを返します。関数が返す際に返されるデータがフィールド・データのうち何番目であるのかと、フィールド・データの長さをデータと一緒に返します。

fbgetlast_tu()とほぼ同じ役割をする関数ですが、若干の違いはあります。fbgetlast_tu()は読み込んだデータが関数のパラメータを使用して返されますが、fbgetval_last_tu()は関数自体の戻り値として読み込むデータを返します。そのため、fbgetlast_tu()はプログラム内でローカル・バッファにデータを移して使用する必要がありますが、fbgetval_last_tu()は他の関数のパラメータにすぐに使用できます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetval_last_tu(FBUF *fbuf, FLDKEY fldkey, int *nth, FLDLEN *fldlen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから読み込むフィールドキーです
nth	指定したフィールドキーに格納されているフィールド・データのうち最後の順番が格納されているポインタです
fldlen	返されるデータの長さを示します

- 戻り値

戻り値	説明
データが格納されたポインタ	フィールド・バッファで指定したフィールドキーを見つけました(読み取ったデータが格納されているchar型ポインタを返します)
NULL	フィールド・バッファで指定したフィールドキーが見つかりませんでした(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数：デフォルト値のtmax.fdl)に定義されておらず、フィールド・バッファで使えない場合、fberrorはFBENOENTに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .

char      *ptr;
long      rcvlen;
int       i,occ ;
FLDLLEN   fldlen;
FBUF      *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "dddd", 0);
fbput(fbuf, INPUT, "eeee", 0);
fbput(fbuf, OUTPUT, "cccc", 0);

ptr = fbgetval_last_tu(fbuf, INPUT, &occ, &fldlen);
printf("Data: [%s], Occurrence: [%d]\n", ptr, occ);
/* output: Data: [eeee], Occurrence: [3] */
for (i = 0; i < occ; i++) {
    ptr = fbgetval(fbuf, INPUT, i, &fldlen);
    printf("Data: [%s]\n", ptr);
}
```

```
}  
.....
```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu(), fbgetlast_tu()

2.37. fbgetvall_tu

フィールド・バッファで指定したフィールドキーのフィールド順のデータをlong型に変換して返す関数です。fldkeyで指定したフィールドキーに格納されているフィールド・データのうち、nthで指定したフィールド順のフィールド・データのフィールド型がshort、int、longの場合、すべてlong型に変換してその値を返します。

fbgetval()と動作は似ていますが、fbgetval()はフィールド型そのままフィールド・データを返し、fbgetvall_tu()はlong型に変換して返します。指定したフィールドキーのフィールドデータがlong型に変換できない型の場合、fbgetvall_tu()は0を返します。

- プロトタイプ

```
#include <fbuf.h>  
#include <atmi.h>  
long fbgetvall_tu(FBUF *fbuf, FLDKEY fldkey, int nth)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファから読み込むフィールドキーを示します
nth	読み込むフィールド順を示します

- 戻り値

戻り値	説明
フィールド・データ	関数の呼び出しに成功しました(フィールド・データをlong型に変換して返します)
0	フィールド・バッファで指定したフィールドキーが見つかりませんでした。あるいは、指定したフィールドキーが、short、int、long型の中の1つではありません(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されておらず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBETYPE	Tmaxでサポートしていないタイプが使用されました。関数で、short、int、long型ではないフィールドがfldkeyで使用された場合が該当します
FBEINVAL	入力パラメータで指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
long lval;
short sval;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

sval = 111;
fbput(fbuf, INPUT, (char *)&sval, 0);
lval = fbgetvall_tu(fbuf, INPUT, 0);

printf("Field Data: [%ld]\n", lval); /* output: Field Data: [111] */
. . . . .
```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetval_last_tu()

2.38. fbgetvals

fldkeyで指定したフィールドキーに格納されているフィールド・データのうち、nthで指定したフィールド順のフィールド・データをそのフィールド型に関係なくすべて文字列に変換して返します。

fbgetval()と動作は似ていますが、fbgetval()はフィールド型そのままですべてフィールド・データを返し、fbgetvals()は文字列に変換して返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetvals (FBUF *fbuf, FLDKEY fldkey, int nth)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーを示します
nth	読み込むフィールド順を示します

- 戻り値

戻り値	説明
フィールド・データ	関数の呼び出しに成功しました(フィールド・データを文字列に変換して返します)
NULL	フィールド・バッファで指定したフィールドキーが見つかりませんでした。あるいは、変数の実行中に他のエラーが発生しました(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されておらず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティーを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます

- 例

```
#include "demo_fdl.h"
. . .
char *ptr;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
ptr = fbgetvals(fbuf, INPUT, 0);
```



```
printf("Field Data: [%s]\n", ptr);
/* output: Field Data: [aaaa] */
.....
```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.39. fbgetvals_tu

フィールド・バッファで指定したフィールドキーに格納されているフィールド・データのうち、指定したフィールド順のフィールド・データを、そのフィールド型がstringの場合に文字列を返します。

fbgetval()と動作は似ていますが、fbgetval()はフィールド型そのままフィールド・データを返し、fbgetvals_tu()は文字列を返します。また、fbgetvals()と動作は似ていますが、若干の違いがあります。

fbgetvals()は、フィールド・バッファで指定したフィールドキーのフィールド順に該当するフィールド・データを見つけられなかった場合はNULLを返します。しかし、fbgetvals_tu()はこのような場合に空文字列("")を返します。fbgetvals()はすべてのフィールド型で動作しますが、fbgetvals_tu()はフィールド型がstringの場合にのみ動作します。他のフィールド型のフィールドキーがパラメータに指定された場合、fbgetvals_tu()は空文字列("")を返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetvals_tu(FBUF *fbuf, FLDKEY fldkey, int nth)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーを示します
nth	読み込むフィールド順を示します

- 戻り値

戻り値	説明
フィールド・データ	関数の呼び出しに成功しました(フィールド・データを文字列に変換して返します)
空文字列("")	フィールド・バッファで指定したフィールドキーが見つかりませんでした。あるいは、関数の実行中に他のエラーが発生しました(fberrorにエラーコードが設定されます)

- エラー

ferrnoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

- 例

```
#include "demo_fdl.h"
. . .
char *ptr;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
ptr = fbgetvals_tu(fbuf, INPUT, 1);
printf("Field Data: [%s]\n", ptr); /* output: Field Data: [bbbb] */
. . . . .
```

- 関連関数

fbget(), fbgetval(), fbgetvalt(), fbgetvals(), fbgetval_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.40. fbgetvalt

fldkeyで指定したフィールドキーに格納されているフィールド・データのうち、nthパラメータで指定したフィールド順に該当するフィールド・データを指定したタイプ(totype)に変換して返します。

fbgetvalt()はfbget_tut()と動作が非常に似ていますが、若干の違いがあります。fbget_tut()は読み取むデータが関数のパラメータを使用して返されますが、fbgetvalt()は関数自体の戻り値として読み込むデータを返します。fbget_tut()は、プログラム内でローカル・バッファにデータを移して使用する必要がありますが、fbgetvalt()は他の関数のパラメータにすぐに使用できます。

また、fbgetval()とほぼ同じ役割をしますが、fbgetval()は指定したフィールドキーのフィールド順に該当するフィールド・データをそのまま読み込むのに対し、fbgetvalt()はフィールド・データを指定したタイプに変換して読み込むことができます。fbgetval()とは異なり、ユーザーが指定したタイプのパラメータに従うため、longやdouble型などについてキャッシュし、該当値を返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbgetvalt(FBUF *fbuf, FLDKEY fldkey, int nth, FLDLEN *len, int totype)
```

- パラメータ

パラメータ	説明
fbuf	fballloc()またはtpalloc()でメモリーに割り当てたフィールド・バッファです
fldkey	フィールド・バッファで読み込むフィールドキーを示します
nth	読み込むフィールド順を示します
len	読み込んだデータの長さを示します
totype	totypeに設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_INT, FB_FLOAT

- 戻り値

戻り値	説明
データ・ポイント	関数の呼び出しに成功しました(読み込んだデータが格納されているchar型ポインタを返します)
NULL	指定したフィールドキーのフィールド順がフィールド・バッファに存在しません(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます

- 例

```
#include "demo_fdl.h"
. . .

int *ptr;
```

```

long rcvlen;
FLDLEN fldlen;
FBUF *fbuf;
Int totype;
Int val = 111;

if ((fbuf = fballocc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, (char *)&val, 0, FB_INT);

ptr = fbgetvalt(fbuf, INPUT, 2, &fldlen, FB_INT);
printf("Data: [%d]\n", ptr); /* output: Data: [111] */
.....

```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu()

2.41. fbinit

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファを初期化する関数です。関数を使用する前に、fballoc()またはtpalloc()を使用してメモリーにフィールド・バッファを割り当てる必要があります。以前にフィールド・バッファに格納されていたすべてのデータは削除されるため、注意が必要です。

- プロトタイプ

```

#include <fbuff.h>
#include <atmi.h>
int fbinit(FBUF *fbuff, FLDLEN buflen)

```

- パラメータ

パラメータ	説明
fbuff	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
buflen	メモリーに割り当てられて、初期化が必要なフィールド・バッファのサイズを示します

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファを初期化するのに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	buflenにfbufを割り当てたサイズより設定値が大きい場合にこのエラーが発生します
FBEINVAL	入力パラメータで指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLの場合にこのエラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
FBUF *fbuf;
int ret;
long size;
size = fbcalcsz(10, 100);

if ((fbuf = (FBUF *)tpalloc("FIELD", NULL, size)) == NULL) {
    printf("tpalloc failed, errno = %d\n", tperrno);
    . . . . .
}
ret = fbinit(fbuf, size);
printf("fbinit, ret : [%d]\n", ret); /* fbinit, ret: [1] */
. . . . .
```

- 関連関数

fbcalcsz(), tpalloc(), fballoc()

2.42. fbinsert

fballoc()またはtpalloc()を使用して、メモリーに割り当てられたフィールド・バッファに指定したフィールドキーのフィールド順に該当するデータを返す関数です。fldkeyおよびnthに指定したフィールドキーのフィールド順に、valueに指定したデータを格納します。

fbinsert()は、パラメータにフィールドキーとフィールド順まで指定する必要があるため、フィールド・バッファ内の特定位置にデータを格納する際に有効に使用されます。しかし一般的な場合、フィールド・データを順番

に格納する際は、関数ユーザーがnthパラメータを手動で増加させる必要があるため注意してください。フィールドキー内に複数のフィールド・データが格納されている状況でフィールド順の中間にフィールド・データを挿入する場合は、指定したフィールド順にデータを格納でき、格納されていた既存のデータはそのフィールド順から自動的にフィールド順が1つずつ増加し、再度格納されます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbininsert(FBUF *fbuf, FLDKEY fldkey, int nth, char *value, int fldlen)
```

- パラメータ

パラメータ	説明
fbuf	fmalloc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファに格納するフィールドキーを示します
nth	格納するフィールド順を示します
value	格納するフィールド・データを示すポインタです。フィールドキーが整数型または実数型の場合は、必ず各タイプに適切な変数を使用し、前で記述した関数のプロトタイプに適切に型キャストします
fldlen	格納するデータの長さを示します。一般的に、フィールドキーのフィールド型がFB_CARRAYの場合は長さを指定する必要がありますが、そうでない場合はこのパラメータは0に指定しても構いません

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファにデータを格納するのに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します (現在のフィールド順より大きい数をnthに指定した場合、フィールドキーがcarray型であればこのエラーが設定されます)

エラーコード	説明
FBETYPE	Tmaxでサポートしていないタイプが使用されました

- 例

```
#include "demo_fdl.h"
. . .
long rcvlen, ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbinsert(fbuf, INPUT, 0, "aaaa", 0);
fbinsert (fbuf, INPUT, 1, "bbbb", 0);
fbinsert (fbuf, INPUT, 2, "cccc", 0);
. . . . .
```

- 関連関数

fbput(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.43. fbisfbuf

フィールド・バッファにデータを格納するためには、あらかじめfballoc()またはtpalloc()でメモリーにバッファの割り当てを受ける必要があります。fbisfbuf()は指定したフィールド・バッファがメモリーに割り当てを受けた有効なバッファであるかを確認する関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbisfbuf(FBUF *fbuf)
```

- パラメータ

パラメータ	説明
fbuf	チェックするフィールド・バッファのポインタです

- 戻り値

戻り値	説明
1	指定したフィールド・バッファが有効です

戻り値	説明
0	指定したフィールド・バッファが有効ではありません (ferrorにエラーコードが設定されます)

- エラー

ferrnoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
FBUF *fbuf;
int ret;

if ((fbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("tpalloc failed, errno = %d\n", tperrno);
    . . . . .
}
ret = fbisfbuf(fbuf);
printf("ret : [%d]\n", ret); /* output : ret : [1] */
. . .
```

- 関連関数

fballoc(), tpalloc()

2.44. fbispres

フィールド・バッファに指定したフィールドキーのフィールド順にフィールド・データが存在するのを確認する関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbispres(FBUF *fbuf, FLDKEY fldkey, int nth)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファでフィールド・データが存在するのを確認するフィールドキーです
nth	フィールド・データが存在するのを確認するフィールド順を意味します

- 戻り値

戻り値	説明
1	フィールド・バッファで指定したフィールドキーのフィールド順にフィールド・データが存在します
0	フィールド・バッファで指定したフィールドキーのフィールド順にフィールド・データが存在しません (ferrorにエラーコードが設定されます)

- エラー

fbernoに以下の値が設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合、該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
ret = fbispres(fbuf, INPUT, 1);
printf("Field Occ: [%d]\n", ret); /* output: Field Occ: [1] */
. . .
```

- 関連関数

fbgetnth(), fbfldcount(), fbkeyoccur()

2.45. fbkeyoccur

フィールド・バッファで指定したフィールドキーに格納されているすべてのフィールドの数を返す関数です。fbfldcount()は指定したフィールド・バッファに格納されているすべてのフィールドの数(1から始まるフィールド数)を返しますが、fbkeyoccur()は指定したフィールドキーの数を返します。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbkeyoccur(FBUF *fbuf, FLDKEY fldkey)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	数を確認するフィールドキーを示します

- 戻り値

戻り値	説明
フィールドの数	フィールド・バッファに指定したフィールドキーに格納されたフィールド・データが存在します
0	フィールド・バッファに指定したフィールドキーに格納されたフィールド・データが存在しません(fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合に該当エラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
int cnt;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
```

```

}
fbput(fbbuf, INPUT, "aaaa", 0);
fbput(fbbuf, INPUT, "bbbb", 0);
fbput(fbbuf, OUTPUT, "cccc", 0);
cnt = fbkeyoccur(fbbuf, INPUT);
printf("Field Count: [%d]\n", cnt);
/* output: Field Count: [2] */
.....

```

- 関連関数

fbgetnth(), fbfldcount()

2.46. fbmake_fldkey

関数はフィールド定義ファイル(***.f)に定義されてはいませんが、プログラム内で動的で新しいフィールドキーを作成する必要がある場合に使用する関数です。

フィールドキーを1つ作成する際は、フィールド型とTmaxで一意のフィールド番号が必要であるため、この2つをパラメータに指定すると、関数の実行後にフィールドキーが作成されて返されます。フィールド番号にすでに存在する番号を指定すると、従来存在するフィールドキーと同じフィールドキーを作成できます。作成されたフィールドキーにデータを格納して他のプロセスに渡すには、渡すプロセスにも内部的に新しく作成したフィールドキーと同じフィールドキー(typeとnoが同じである必要あり)が作成されている必要があります。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
FLDKEY fbmake_fldkey(int type, int no)

```

- パラメータ

パラメータ	説明
type	新しく作成されるフィールドキーのタイプです
no	フィールドキーを作成する際に必要なフィールド番号で、以下の値を設定できます – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_INT, FB_FLOAT

- 戻り値

指定されたtypeとnoを組み合わせで新しいフィールドキーを作成し、フィールドキーを返します。

- 例

```
#include "demo_fdl.h"
. . .
FLDKEY fkey;

fkey = fbmake_fldkey(FB_STRING, 101);
printf("Field Key : [%d]\n", fkey);
/* output : Field Key : [469762149] */
. . .
```

- 関連関数

fbget_fldkey()

2.47. fbnext_tu

フィールド・バッファで次のフィールドキーとフィールド・データを返す関数です。fldkeyに開始フィールドキーを指定すると、指定された開始フィールドキーからフィールド・バッファにあるすべてのフィールドキーおよびフィールド・データを順番に返します。フィールド・バッファに格納されている最初のフィールドキーから読み込むには、開始フィールドキーは必ずFIRSTFLDKEYである必要があります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbnext_tu(FBUF *fbuf, FLDKEY *fldkey, int *nth, char *value, int *len)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	次のフィールドキーを示すポインタです
nth	次のフィールドのフィールド順を示すポインタです
value	次のフィールドのフィールド・データを示すポインタです。valueがNULLでない場合、次のフィールドのフィールド・データがvalueにコピーされます。次のフィールドがそれ以上存在せずにvalueがNULLになる場合、valueはそれ以上値が変わらず、fldkeyとnthのみ変わります
len	次のフィールド・データの長さを示すポインタです。lenは、バッファがフィールド・データを格納するための空間が充分であるかを確認します。充分な場合、lenに次のフィールド・データが格納されているvalueの長さを格納します

- 戻り値

戻り値	説明
1	次のフィールドが見つかりました (fldkey、nth、value、lenなどの値をポインタで返します)
0	次のフィールドが見つかりませんでした
-1	関数が実行される際にエラーが発生しました (ferrorにエラーコードが設定されます)

● エラー

ferrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、ferrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します

● 例

```
#include "demo_fdl.h"
. . .
char data[100];
long rcvlen;
int i, occ, nth, fc, len;

FLDKEY fkey;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbput(fbuf, INPUT, "dddd", 0);
fbput(fbuf, INPUT, "eeee", 0);
fbput(fbuf, INPUT, "cccc", 0);

len = 4;
for(fkey = FIRSTFLDKEY; fc = fbnext_tu(fbuf, &fkey, &occ, data, &len) > 0;

len = sizeof(data))
printf("fkey = %ld Data: [%s], Occurrence: [%d] len [%d]\n", fkey, data, occ,
len);
```

```

/* output: fkey = 469762149 Data: [aaaa], Occurrence: [0] len [4]
   output: fkey = 469762149 Data: [bbbb], Occurrence: [0] len [4]
   output: fkey = 469762149 Data: [dddd], Occurrence: [0] len [4]
   output: fkey = 469762149 Data: [eeee], Occurrence: [0] len [4]
   output: fkey = 469762149 Data: [cccc], Occurrence: [0] len [4] */
. . .

```

- 関連関数

fbput(), fbget(), fbgetval(), fbgetvalt(), fbgetval_tu(), fbgetvals_tu(), fbgetvall_tu(), fbgetval_last_tu(), fbgetlast_tu()

2.48. fbprint

現在のフィールド・バッファの内容を決められた形式に従って標準出力する関数です。一般的に標準出力はモニターの画面であるため、フィールド・バッファの内容をモニターで確認する際に有効に使用されます。

出力形式は以下のとおりです。

```

fkey = 469762149, fname = INPUT, type = string, value = 5555 fkey = 469762149,
fname = INPUT, type = string, value = 7777 fkey = 469762149, fname = INPUT,
type = string, value = 3333 fkey = 469762149, fname = INPUT, type = string,
value = 8888

```

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbprint(FBUF *fbuf)

```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	関数が実行される際にエラーが発生しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEBADFB	フィールド・バッファが有効ではありません。そのため、現在のフィールド・バッファ以外で正常にメモリーに割り当てられたバッファを使用する必要があります

- 例

```
#include "demo_fdl.h"
. . .
FBUF *fbuf;
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbput(fbuf, INPUT, "5555", 0);
fbput(fbuf, INPUT, "7777", 0);
fbput(fbuf, INPUT, "3333", 0);
fbput(fbuf, INPUT, "8888", 0);
fbprint(fbuf);
. . . . .
```

- 関連関数

fbfprintf()

2.49. fbput

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファに新しいフィールドを追加する関数です。

関数のパラメータで指定したフィールドキーにフィールドを追加する際、指定したフィールドキーにフィールドが存在しなければ、valueは1番目にフィールドが作成されます。すでにフィールドが1つ以上存在する場合、valueは現在存在するフィールド順の中で最も高いフィールド順より1大きいフィールド順でフィールドが作成され、そのフィールドにフィールド・データとして格納されます。fbput()は、フィールド順が自動的に適切に増加するため、フィールド・バッファにデータを格納する際、fbinsert()のようにフィールド順まで考慮する必要はありません。

valueのデータ型がchar *型になっていますが、フィールドキーのフィールド型に従って、string型ではない場合はフィールド型に適した変数を宣言してデータを格納し、関数を呼び出す場合は適切に型キャストします。そのため、関数のプロトタイプが次のように定義されていても、FB_INT、FB_LONG、FB_FLOATなどのデータでフィールドキーにフィールドを追加して格納できます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbput(FBUF *fbuf, FLDKEY fldkey, char *value, FLDLEN len)
```

● パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	作成するフィールドのフィールドキーです
value	新しく作成されたフィールドに格納するデータです
len	valueの長さを示します。フィールド型がFB_CARRAYの場合は長さを明示する必要がありますが、そうでない場合は無視し、0に指定しても構いません

● 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファにフィールドを追加するのに失敗しました (fberrorにエラーコードが設定されます)

● エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました

● 例

```
#include "demo_fdl.h"
. . .
long rcvlen, ret;
FBUF *fbuf;
int ival = 1000;
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
```



```
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INTDATA, (char *)&ival, 0);
```

- 関連関数

fbinsert(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.50. fbputt

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーに新しいフィールドを追加する関数です。フィールドがフィールド・バッファーに追加される前に、格納するデータのvalueはユーザーが指定したデータ型で格納するフィールドキーのフィールド型に変換されます。

fbputt()はfbput()と非常に動作が似ていますが、fbput()は指定したデータを新しく作成されたフィールドにそのまま格納するのに対し、fbputt()は指定したデータ型と格納するフィールドキーのフィールド型が同じではない場合、指定したデータ型をフィールドキーのフィールド型に変換して格納できます。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbputt(FBUF *fbuf, FLDKEY fldkey, char *value, FLDLEN len, int type)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファーです
fldkey	作成するフィールドのフィールドキーです
value	新しく作成されたフィールドに格納するデータです
len	valueの長さを示します。フィールドのタイプがFB_CARRAYの場合は長さを明示する必要がありますが、そうでない場合は無視し、0に指定しても構いません
type	ユーザーが指定したデータ型で、データがフィールドに格納される前に指定したtypeからフィールドキーのタイプに変換されます。 typeに設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_INT, FB_FLOAT

- 戻り値

戻り値	説明
1	関数の呼び出しに成功しました
-1	フィールド・バッファにフィールドを追加するのに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティーを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます

- 例

```
#include "demo_fdl.h"
. . .
long rcvlen, ret;
FBUF *fbuf;
int ival;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
ival = 1111;
fbputt(fbuf, INPUT, "aaaa", 0, FB_STRING);
/* string型のINPUTフィールドキーにint型のival(1111)を入れるためにFB_INT型を指定。 */
fbputt(fbuf, INPUT, (char *)&ival, 0, FB_INT);
. . .
fbprint(fbuf);
fbfree(fbuf);
. . . . .
/* output :      fkey = 469762149, fname = NAME, type = string, value = aaaa
                  fkey = 469762149, fname = NAME, type = string, value = 1111 */
. . . . .
```

- 関連関数

fbinsert(), fbchg_tu(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.51. fbrealloc

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファのサイズが不足した場合、メモリーのサイズを増やして、再度割り当てる関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
FBUF *fbrealloc(FBUF *fbuf, int ncount, int nlen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
ncount	データを格納するフィールドの数を意味します
nlen	フィールド・バッファの全体サイズをバイト単位で指定します

- 戻り値

戻り値	説明
メモリーのポインタ	関数の呼び出しに成功しました
NULL	フィールド・バッファをメモリーに割り当てることに失敗しました (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEOS	OSまたはメモリーの割り当てに失敗、Tmaxへの接続失敗 (サービスの呼び出し、send、recvなど) のようなシステムエラーまたはネットワークの状態が不安定な場合などに現れるエラーです
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します。たとえば、fbufがNULLに指定された場合にエラーが発生します

- 例

```
#include "demo_fdl.h"
. . .
```

```

FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    .....
}
printf("Buffer Size: [%ld]\n", fbget_fsize(fbuf));
/* output: Buffer Size: [1024] */
if ((fbuf = (FBUF *)fbrealloc(fbuf, 1000, 150)) == NULL) {
    printf("fbrealloc failed, errno = %d\n", ferror);
    .....
}

printf("Buffer Size: [%ld]\n", fbget_fsize(fbuf));
/* output: Buffer Size: [4158] */

```

- 関連関数

fballoc(), tpalloc(), fbget_fsize(), fbget_used(), fbget_unused()

2.52. fbread

指定したファイル・ストリームからデータを読み込んで、fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファにフィールド・データでロードする関数です。iopで指定したファイル・ストリームからデータを読み込んで、fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファにフィールド・データでロードします。

fbread()で使用するファイル・ストリームは、以前にfbwrite()を呼び出して作成されたファイルを指定する必要があります。他のプロセスから渡されたデータを現在のプロセスのフィールド・バッファにロードする場合に使用します。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbread(FBUF *fbuf, FILE *iop)

```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
iop	データを読み込むファイル・ストリームを指定します

- 戻り値

戻り値	説明
1	指定されたファイル・ストリームからデータを正常にロードしました
-1	指定されたファイル・ストリームからデータをロードできませんでした (ferrorにエラーコードが設定されます)

- エラー

ferrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEOS	OSまたはメモリーの割り当てに失敗、Tmaxへの接続失敗 (サービスの呼び出し、send、recvなど) のようなシステムエラーまたはネットワークの状態が不安定な場合に現れるエラーです

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FILE *iop;
FBUF *fbuf;

if ((fbuf = fballocc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
iop = fopen("sample.dat", "r");
ret = fbread(fbuf, iop);

if (ret == 1)
    fbprint(fbuf);
fclose(iop)
. . . . .
```

- 関連関数

fbwrite()

2.53. fbsnull

フィールド・バッファーで指定したフィールドキーのフィールド順とマッピングする構造体のメンバー変数がNULLであるかを確認する関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbsnull(char *cstruct, char *cname, int nth, char *stname)
```

- パラメータ

パラメータ	説明
cstruct	使用する構造体タイプの変数を(char *)型に型キャストして使用します
cname	構造体のメンバー変数を文字列で指定できます
nth	フィールド順を示します
stname	VIEWで定義した構造体の名前を文字列で指定します

- 戻り値

戻り値	説明
1	指定したフィールド順のフィールド・データがNULLです
0	指定したフィールド順のフィールド・データがNULLではありません
-1	関数の実行中にエラーが発生しました (ferrorにエラーコードが設定されます)

- エラー

fbernoに以下の値が設定されます。

エラーコード	説明
FBEBADSTRUCT	有効でない構造体が使用されました。一般的に、Tmaxで認識できない形式の構造体を使用した場合や、構造体定義ファイルが正しくコンパイルされなかった場合に発生するエラーです

- 例

```
#include "demo_fdl.h"
#include "demo_sdl.h"
. . .
int val = 1000, ret;
struct demo temp;
char *cstruct;
FBUF *fbuf;

if ((fbuf = fballocc(10, 100)) == NULL) {
    printf("fballocc failed, errno = %d\n", ferror);
    . . . . .
```

```

}
if(tpstart((TPSTART_T *)NULL) == NULL){
    printf(tpstart error\n");
    .....
}
cstruct = (char *)&temp;
if(fbstinit(cstruct, "demo") < 0){
    .....
}
fbput(fbuf, INPUT, "aaaa", 0);
if(fbftos(fbuf, cstruct, "demo") < 0){
    .....
}
ret = fbsnull(cstruct, "demodata", 0, "demo");
if(ret < 0) {
    printf("fbsnull failed\n");
    .....
}
else if(ret == 0)
    printf("This occurrence is not null\n");
else
    printf("This occurrence is null\n");
.....

```

- 関連関数

fbgetlen(), fbstinit(), fbstelinit(), fbinit()

2.54. fbstelinit

入力パラメータで指定した構造体のメンバー数をそれぞれ適切にNULLに初期化する関数です。メンバー変数のデータ型が整数の場合は0に初期化し、文字配列の場合はNULLに初期化します。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbstelinit(char *cstruct, char *cname, char *stname)

```

- パラメータ

パラメータ	説明
cstruct	使用する構造体タイプの変数を(char *)型に型キャッシュして使用します
cname	構造体のメンバー変数を文字列で指定するか、(char *)型に型キャッシュして使用します

パラメータ	説明
stname	VIEWで定義した構造体の名前を文字列で指定します

- 戻り値

戻り値	説明
1	指定したメンバー変数を正常に初期化しました
-1	指定したメンバー変数を初期化できませんでした (fberror! にエラーコードが設定されます)

- エラー

fberrno に以下の値が設定されます。

エラーコード	説明
FBEBADSTRUCT	有効でない構造体が使用されました。 一般的に、Tmax で認識できない形式の構造体が使用された場合や、構造体定義ファイルが正しくコンパイルされなかった場合に発生するエラーです

- 例

```
#include "demo_fdl.h"
#include "demo_sdl.h"

. . .
cstruct demo temp;
char *cstruct;
if(tpstart((TPSTART_T *)NULL) == NULL) {
    printf("tpstart failed\n");
    . . . . .
}
cstruct = (char *)&temp;
if(fbstelinit(cstruct, "demodata", "demo") < 0) {
    . . . . .
}
. . . . .
```

- 関連関数

fbstinit(), fbinit(), fbsnull()

2.55. fbstinit

入力パラメータで指定した構造体変数を初期化する関数です。fbstelinit()は指定した構造体に対して各メンバー変数を適切に初期化しますが、fbstinit()は構造体全体を1度に初期化します。そのため、メンバー変数のうち1つを初期化する場合はfbstelinit()を使用し、構造体を初期化する場合はfbstinit()を使用するのが望ましいです。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbstinit(char *cstruct, char *stname)
```

- パラメータ

パラメータ	説明
cstruct	使用する構造体タイプの変数を(char *)型に型キャストして使用します
stname	VIEWで定義した構造体の名前を文字列で指定します

- 戻り値

戻り値	説明
1	指定した構造体を正常に初期化しました
-1	指定した構造体を正常に初期化できませんでした (fberrorにエラーコードが設定されます)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEBADSTRUCT	有効でない構造体が使用されました。一般的に、Tmaxで認識できない形式の構造体を使用した場合や、構造体定義ファイルが正しくコンパイルされなかった場合に発生するエラーです

- 例

```
#include "demo_fdl.h"
#include "demo_sdl.h"
. . .
cstruct demo temp;
char *cstruct;
if(tpstart((TPSTART_T *)NULL) == NULL) {
    printf("tpstart failed\n");
}
```

```

        .....
    }
    cstruct = (char *)&temp;
    if(fbstinit(cstruct, "demo") < 0) {
        .....
    }
    .....

```

- 関連関数

fbstelinit(), fbinit(), fbsnull()

2.56. fbstof

C構造体バッファ(stname)に格納されているデータをマッチングするフィールド・バッファーに送信する関数で、fbftos()と反対の機能を行います。一般構造体ファイルの***.sとは若干異なる構造体ファイル(VIEW定義ファイル)が事前に定義されている必要があります。構造体ファイル内のフィールドキーを定義される構造体のメンバー変数と1:1でマッチングされている必要があります。VIEW定義ファイルに関する内容は[fbftos\(\)](#)を参照してください。

fbstof()はfbftos()とは異なり、以下のような場合に、C構造体にある値がフィールド・バッファーにコピーされません。

- VIEWファイルのNULL項目に設定した値と同じ場合
- フィールド・バッファーにコピーする構造体フィールドにある値がデフォルト値と同じ場合

フィールド・バッファーに値を格納していない状態でfbftos()を使用してC構造体にコピーする場合は、VIEWファイルのNULL項目に定義された値がコピーされます。C構造体に値を格納していない状態でfbstof()を使用してフィールド・バッファーにコピーする場合、またはNULL項目に設定した値と同じ値で格納した後にコピーする場合は、フィールド・バッファーにコピーされません。

VIEWファイルのNULL項目に値を定義せず、フィールド・バッファーに値を格納していない状態でfbftos()を使用してC構造体でコピーする場合、fbftos()ではデフォルト値がコピーされます。C構造体に値を格納していない状態でfbstof()を利用してフィールド・バッファーにコピーする場合、またはデフォルト値と同じ値で格納した後にコピーする場合は、いかなる値もコピーされません。

- プロトタイプ

```

#include <fbuf.h>
#include <atmi.h>
int fbstof(FBUF *fbuf, char *cstruct, int mode, char *stname)

```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
cstruct	使用する構造体タイプの変数を(char *)型に型キャストして使用します
mode	<p>データをフィールド・バッファに送信する際の方式で、modeによって機能が若干異なります。modeはfbbufop()の各モードと完全に一致して動作するため、fbbufop()を参照してください。</p> <p>modeに設定可能な値は以下のとおりです</p> <ul style="list-style-type: none"> – FBUPDATE : FBUPDATEモードの場合、構造体のメンバー変数の値がNULLであれば、そのデータは無視され、フィールド・バッファに送信されません – FBJOIN – FBOJOIN – FBCONCAT
stname	VIEWで定義した構造体の名前を文字列で指定します

● 戻り値

戻り値	説明
1	構造体バッファに格納されているデータを正常にフィールド・バッファに送信しました
-1	構造体バッファに格納されているデータをフィールド・バッファに送信できませんでした (fberrorにエラーコードが設定されます)

● エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBETYPE	Tmaxでサポートしていないタイプが使用されました
FBEINVAL	入力パラメータで指定された値の中に、有効でないパラメータが存在します
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、fdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます
FBEBADSTRUCT	有効でない構造体を使用されました。一般的に、Tmaxで認識されていない形式の構造体を使用された場合や、構造体定義ファイルが正しくコンパイルされなかった場合に発生するエラーです

エラーコード	説明
FBEMALLOC	システムエラーです。フィールド・バッファをメモリに割り当てることに失敗した場合、fberrorはFBEMALLOCに設定されます

- 例

```
#include "demo_fdl.h"
#include "demo_sdl.h"
. . .
int ret, fdllen, val = 1000;
struct demo temp;
FBUF *fbuf;
Char *cstruct;

if(tpstart((TPSTART_T *)NULL) == -1) {
    printf("tpstart failed\n");
    . . . . .
}

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

cstruct = (char *)&temp;
if(fbstinit((char *)&temp, "demo") < 1) {
    printf("fbstinit failed, errno = %d\n", fberror);
    . . . . .
}

strcpy(temp.demodata, "aaaa");
temp.num = 1000;
if(fbstof(fbuf, cstruct, FBUPDATE, "demo") < 0) {
    . . . . .
}
. . . . .
```

- 関連関数

fbftof(), fbbufop(), fbstinit(), fbstelinit(), fbnull()

2.57. fbstrerror

フィールド・バッファと関連するAPI関数を実行する際に発生するエラーコードに該当する内容を文字列で返す関数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbstrerror(int ferror)
```

- パラメータ

パラメータ	説明
ferror	フィールド・バッファに関連するAPI関数を実行する際にエラーが発生すると、まずferrorという変数にエラーコードが設定されます。このferrorの値を関数の入力パラメータに指定し、このエラーに関する内容を文字列で確認できます。ここで発生し得るすべてのエラーは、FDLに関連するヘッダーファイルの<fbuf.h>に定義されています

- 戻り値

戻り値	説明
文字列	関数の呼び出しに成功しました(エラーコードに関する内容を文字列で返します)
NULL	関数の実行に失敗しました

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}

ret = fbget(fbuf, INPUT, "aaaa", 0);
if( ret < 0 )
    printf("ret = %d , ferror = %s[%d]\n", ret, fbstrerror(ferror) , ferror);
/* output : ret = -1, ferror = not found[6] */
. . .
```

- 関連関数

fberror()

2.58. fbttypecv

fromvalに指定したfromtypeデータ型のフィールド・データをtotypeで型キャッシュして返す変数です。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
char *fbtypecv(FLDLEN *tolen, int totype, char *fromval, int fromtype,
               FLDLEN fromlen)
```

- パラメータ

パラメータ	説明
tolen	変換されたデータの長さを返すポインタです
totype	データを変換するタイプで、設定可能な値は以下のとおりです – FB_CARRAY, FB_SHORT, FB_DOUBLE, FB_LONG, FB_STRING, FB_CHAR, FB_INT, FB_FLOAT
fromval	変換するデータを示すポインタです
fromtype	fromvalのデータ型です
fromlen	fromtypeに指定したタイプのデフォルトサイズと関連があります

- 戻り値

戻り値	説明
変換されたデータのポインタ	データを変換するのに成功しました
NULL	データを変換するのに失敗しました (fberrorにエラーコードを設定します)

- エラー

fberrnoに以下の値のうちの1つが設定されます。

エラーコード	説明
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足した場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します (FB_CARRAYタイプの場合、データの長さを指定しなければエラーが設定されます)
FBETYPE	Tmaxでサポートしていないタイプが使用されました

エラーコード	説明
FBEBADFLD	有効でないフィールドキーが使用されました。一般的に、フィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合、fberrorはFBEBADFLDに設定されます

- 例

```
#include "demo_fdl.h"

. . .
int tolen, totype, fromtype, fromlen;
long vall;
FBUF *fbuf;
char *ptr;

if ((fbuf = fballocc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    .....
}

vall = 1111;
fromlen = sizeof(long);

fbput(fbuf, TOTMON, (char *)&vall, 0);
ptr = fbtypecvrt(&tolen, FB_STRING, (char *)&vall, FB_LONG, fromlen);

printf("Converted string = %s\n", ptr);
/* output Converted string = 1111 */
.....
```

- 関連関数

fbget_fldname(), fbget_fldno(), fbget_fldtype(), fbget_strfldtype()

2.59. fbupdate

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファに、入力パラメータに指定したフィールドキーのフィールド順の位置に、valueに指定したデータをすでに格納されているフィールド・データと変更する関数です。

特定の位置にあるフィールド・データを変更するものなので、フィールドキーとフィールド順を必ず指定します。同じフィールドキーにフィールド順が重複してデータが置換されると、最後に変更したフィールド・データが有効です。

fbupdate()はfbchg_tu()と動作が似ています。ただし、fbchg_tu()は変更しようとして指定したフィールドキーのフィールド順がすでに存在しない場合、自動的に新しくフィールドを1つ追加してデータを格納しますが、

fbupdate()は指定したフィールドキーのフィールド順がすでに存在しない場合、エラーが発生して、fberrorにFBENOENTと設定する点異なります。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbupdate(FBUF *fbuf, FLDKEY fldkey, int nth, char *value, int fldlen)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
fldkey	フィールド・バッファに変更するフィールドのフィールドキーです
nth	fldkeyで指定したフィールドキーのデータのうち、変更するフィールド順を意味します。そして、valueパラメータが(char *)型であるため、変更するフィールドキーが整数型あるいは実数型の場合は必ず(char *)型に型キャストします
value	パラメータが(char *)型であるため、変更するフィールドキーが整数型あるいは実数型の場合は必ず(char *)型に型キャストします
fldlen	一般的には、フィールドの長さを指定しなくても構いません。フィールドキーがFB_CARRAY型の場合は、必ずフィールド・データの長さをバイト単位で指定します

- 戻り値

戻り値	説明
1	正常にフィールド・データを変更しました
-1	フィールド・データを変更できませんでした(fberrorにエラーコードを設定します)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBENOENT	指定したフィールドキーがFDLFILE(環境変数: デフォルト値のtmax.fdl)に定義されず、フィールド・バッファで使用できない場合、fberrorはFBENOENTに設定されます
FBENOSPACE	フィールド・バッファにデータを格納あるいはコピーする空間が不足する場合、fberrorはFBENOSPACEに設定されます
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBETYPE	Tmaxでサポートしていないタイプが使用されました

- 例

```
#include "demo_fdl.h"
. . .
long rcvlen, ret;
FBUF *fbuf;
char buf[20];
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", ferror);
    . . . . .
}
fbinsert(fbuf, INPUT, 0, "aaaa", 0);
ret = fbupdate(fbuf, INPUT, 0, "bbbb", 0);
fbget (fbuf, INPUT, buf, 0);
printf("ret = [%d] INPUT = %s \n", ret, buf);
/* output: "ret = 1 INPUT = bbbbbb" */
. . . . .
```

- 関連関数

fbinsert(), fbput(), fbchg_tu(), fbegt(), fbget_tu(), fbgetval(), fbgetval_last_tu(), fbgetvals_tu(), fbgetvall_tu()

2.60. fbwrite

fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファの内容をiopに指定したファイル・ストリームに格納する関数です。

ファイル・ストリームに格納されるデータはバイナリを含んだデータです。他のプロセスにデータを渡すために、ファイルを使用する場合に使用します。fbwrite()によって格納されたデータは、fbread()を使用してフィールド・バッファにロードできるため注意してください。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int fbwrite(FBUF *fbuf, FILE *iop)
```

- パラメータ

パラメータ	説明
fbuf	fballoc()またはtpalloc()でメモリーに割り当てられたフィールド・バッファです
iop	データを格納するファイル・ストリームを示すポインタです

- 戻り値

戻り値	説明
1	指定されたファイル・ストリームにデータを正常に格納しました
-1	指定されたファイル・ストリームにデータを格納できませんでした (fberrorにエラーコードを設定します)

- エラー

fberrnoに以下の値のうち1つが設定されます。

エラーコード	説明
FBEINVAL	入力パラメータに指定された値の中に、有効でないパラメータが存在します
FBEOS	OSまたはメモリーの割り当てに失敗、Tmaxへの接続失敗 (サービスの呼び出し、send、recvなど) のようなシステムエラーまたはネットワークの状態が不安定な場合などに現れるエラーです

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FILE *iop;
FBUF *fbuf;
if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
iop = fopen("sample.dat", "w");
fbput(fbuf, INPUT, "aaaa", 0);
fbput(fbuf, INPUT, "bbbb", 0);
fbwrite(fbuf, iop);
fclose(iop)
. . . . .
```

- 関連関数

fbread()

2.61. getfberrno

フィールド・バッファに関連するAPI関数を実行中にエラーが発生した場合にエラーコードを返す関数です。関数で返すエラーコードのエラーメッセージを知りたい場合は、この関数を入力パラメータにしてfbstrerror()を使用します。getfberrno()は入力パラメータがありません。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int getfberrno(void)
```

- 戻り値

現在fberrorに設定されている値を返します。

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}

ret = fbget(fbuf, INPUT, "aaaa", 0);
if( ret < 0 )
    printf("ret = %d , error = %s[%d]\n", ret, fbstrerror(getfberrno()) ,
          getfberrno());

/* output : ret = -1, error = not found[6] */
. . .
```

- 関連関数

fbstrerror(), getfberror()

2.62. getfberror

フィールド・バッファに関連するAPI関数を実行中にエラーが発生した場合にエラーコードを返す関数です。

関数で返すエラーコードのエラーメッセージを知りたい場合は、この関数を入力パラメータにしてfbstrerror()を使用します。getfberrno()は入力パラメータがありません。

- プロトタイプ

```
#include <fbuf.h>
#include <atmi.h>
int getfberror(void)
```

- 戻り値

現在fberrorに設定されている値を返します。

- 例

```
#include "demo_fdl.h"
. . .
int ret;
FBUF *fbuf;

if ((fbuf = fballoc(10, 100)) == NULL) {
    printf("fballoc failed, errno = %d\n", fberror);
    . . . . .
}
ret = fbget(fbuf, INPUT, "aaaa", 0);
if( ret < 0 )
    printf("ret = %d , error = %s[%d]\n", ret, fbstrerror(getfberror()) ,
          getfberror());
/* output : ret = -1, error = not found[6] */
. . .
```

- 関連関数

fbstrerror(), getfberror()

索引

C

C構造体, 2

F

FDL, 1

fdlc, 4

FDL関数

- fballoc, 10
- fbbufop, 12
- fbbufop_proj, 16
- fbcalcsiz, 18
- fbchg_tu, 19
- fbchg_tut, 21
- fbdelall, 22
- fbdelall_tu, 24
- fbdelete, 25
- fbextread, 27
- fbfldcount, 30
- fbfprint, 32
- fbfree, 33
- fbftos, 34
- fbget, 38
- fbget_fbsize, 43
- fbget_fldkey, 45
- fbget_fldname, 46
- fbget_fldno, 47
- fbget_fldtype, 48
- fbget_strfldtype, 51
- fbget_tu, 52
- fbget_tut, 54
- fbget_unused, 56
- fbget_used, 57
- fbgetalloc_tu, 39
- fbgetalloc_tut, 41
- fbgetf, 58
- fbgetlast_tu, 49

- fbgetlen, 60
- fbgetnth, 61
- fbgetntht, 62
- fbgetval, 64
- fbgetval_last_tu, 68
- fbgetvali, 66
- fbgetvall_tu, 70
- fbgetvals, 71
- fbgetvals_tu, 73
- fbgetvalt, 74
- fbinit, 76
- fbinsert, 77
- fbisfbuf, 79
- fbispres, 80
- fbkeyoccur, 82
- fbmake_fldkey, 83
- fbnext_tu, 84
- fbprint, 86
- fbput, 87
- fbputt, 89
- fbread, 92
- fbrealloc, 91
- fbnull, 93
- fbstelinit, 95
- fbstinit, 97
- fbstof, 98
- fbsterror, 100
- fbtypecvt, 102
- fbupdate, 103
- fbwrite, 105
- getfberrno, 106
- getfberror, 107

た

データの独立性, 2

は

- フィールドキー, 1
- フィールド・バッファ, 1, 2
- フィールド型, 1
- フィールド順, 1

