B1WW-7051-01Z0(00)

Microsoft® Windows® 98
Microsoft® Windows® Me
Microsoft® Windows NT®
Microsoft® Windows® 2000
Microsoft® Windows® XP
Microsoft® Windows Server™ 2003

NetCOBOL for Windows V8.0
COBOLファイルアクセスルーチン

使用手引書

NetCOBOL

FUJITSU

まえがき

COBOLファイルアクセスルーチンは、COBOLファイルを操作するためのC言語用のAPI(Application Program Interface)関数群です。
この関数群を利用することによって、COBOLファイルをアクセスするアプリケーションの開発/運用が行えます。以下に示すシステムの32ビットモードで動作します。

- Microsoft(R) Windows(R) 98 operating system
- Microsoft(R) Windows(R) Millennium Edition
- Microsoft(R) Windows NT(R) Workstation operating system Version 4.0
- Microsoft(R) Windows NT(R) Server Network operating system Version 4.0
- Microsoft(R) Windows NT(R) Server Network operating system Version 4.0, Terminal Server Edition
- Microsoft(R) Windows NT(R) Server Network operating system, Enterprise Edition Version 4.0
- Microsoft(R) Windows(R) 2000 Professional operating system
- Microsoft(R) Windows(R) 2000 Server operating system
- Microsoft(R) Windows(R) 2000 Advanced Server operating system
- Microsoft(R) Windows(R) XP Professional operating system
- Microsoft(R) Windows(R) XP Home Edition operating system
- Microsoft(R) Windows Server(TM) 2003, Standard Edition
- Microsoft(R) Windows Server(TM) 2003, Enterprise Edition

製品の呼び名について

本書に記載されている製品の名称を、以下のように略して表記します。

- 「Microsoft(R) Windows(R) 98 operating system」
  → 「Windows(R) 98」
- 「Microsoft(R) Windows(R) Millennium Edition」
  →「Windows(R) Me」
- 「Microsoft(R) Windows NT(R) Workstation operating system Version 4.0」
  → 「Windows NT(R)」または、「Windows NT(R) 4.0」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0」
  → 「Windows NT(R)」または、「Windows NT(R) 4.0」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0, Terminal Server Edition」
  → 「Windows NT(R)」、「Windows NT(R) 4.0」または、「Windows NT(R) 4.0 T.S.E」
- 「Microsoft(R) Windows NT(R) Server Network operating system, Enterprise Edition Version 4.0」
  → 「Windows NT(R)」、「Windows NT(R) 4.0」または、「Windows NT(R) 4.0 E.E」
- 「Microsoft(R) Windows(R) 2000 Professional operating system」
  → 「Windows(R) 2000」または、「Windows(R) 2000 Professional」
- 「Microsoft(R) Windows(R) 2000 Server operating system」
  → 「Windows(R) 2000」または、「Windows(R) 2000 Server」
- 「Microsoft(R) Windows(R) 2000 Advanced Server operating system」
  → 「Windows(R) 2000」または、「Windows(R) 2000 Advanced Server」
- 「Microsoft(R) Windows(R) XP Professional operating system」
  → 「Windows(R) XP」または、「Windows(R) XP Professional」
- 「Microsoft(R) Windows(R) XP Home Edition operating system」
  → 「Windows(R) XP」または、「Windows(R) XP Home Edition」

- 「Microsoft(R) Windows Server(TM) 2003, Standard Edition」
  →「Windows Server (TM) 2003」
- 「Microsoft(R) Windows Server(TM) 2003, Enterprise Edition」
  →「Windows Server (TM) 2003」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0」
  →「Windows NT(R) Server」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0, Terminal Server Edition」
  →「Windows NT(R) Server」
- 「Microsoft(R) Windows NT(R) Server Network operating system, Enterprise Edition Version 4.0」
  →「Windows NT(R) Server」
- 「Windows(R) 98」、「Windows NT(R)」、「Windows(R) 2000」、「Windows(R) XP」および「Windows Server(TM) 2003」
  →「Windows(R)」
- 「Microsoft(R) Visual C++(R) development system」
  →「Visual C++(R)」

本書の目的

本書はCOBOLのファイル入出力機能の知識と、C言語を用いたプログラミングについての知識がある方を対象としています。
本書ではCOBOLファイルアクセスルーチンを利用したCソースプログラムの作成と、プログラムのリンク、実行方法について説明しています。COBOLのファイル入出力機能については、以下のマニュアルを参照してください。

- COBOL文法書
- NetCOBOL 使用手引書

登録商標について

本書に記載している登録商標を、以下に示します。
Microsoft、Windows、Windows NT、Windows Server、Visual C++は、米国Microsoft Corporationの米国およびその他の国における商標または登録商標です。

2005年6月

目次

第1章 COBOLファイルアクセスルーチンを利用する前に

ここでは、COBOLファイルアクセスルーチンの紹介と、準備するもの、環境設定について説明します。

1.1 COBOLファイルアクセスルーチンとは

COBOLファイルアクセスルーチンは、COBOLファイルを操作するためのC言語用のAPI(Application Program Interface)関数群です。これらの関数は、COBOLランタイムシステムを呼び出すことによって、ファイルの操作を行います。

COBOLファイルアクセスルーチンを使用することにより、以下の操作が実現できます。

- COBOLアプリケーションで作成したファイルの読込み/書換えなどの既存資産への入出力。
- COBOLで扱う以下の編成のファイルの創成。
  - 行順ファイル
  - レコード順ファイル
  - 相対ファイル
  - 索引ファイル
- COBOLアプリケーションとのファイル/レコードの排他/共用。
- 既存の索引ファイルのファイル属性/レコードキー構成の解析。

1.2 準備するもの

開発言語として32ビット対応Cコンパイラを用意します。Cコンパイラのインストールと、その動作環境の設定を行っておいてください。

1.3 環境設定

NetCOBOLをインストールすると、COBOLファイルアクセスルーチンはNetCOBOLと同じフォルダに格納されます。

NetCOBOLをインストールした後、環境変数PATHと環境変数LIBにNetCOBOLのインストールフォルダが設定されていることを確認してください。設定されていなければ、これらの環境変数にNetCOBOLのインストールフォルダを追加してください。環境変数の設定については、“NetCOBOL使用手引書”の“1.2.1 環境変数の設定”を参照してください。

第2章 使い方

ここでは、Cソースプログラムの作成、翻訳、オブジェクトファイルのリンク、プログラムの実行について説明します。

2.1 Cソースプログラムの作成

COBOLファイルアクセスルーチンを用いたCソースプログラムをテキストエディタなどで作成します。当アクセスルーチンを使う上での注意事項については、“第6章 注意事項”を参照してください。
Cソースプログラムには以下の記述を入れ、ヘッダファイルをインクルードすることを明示します。

- #include "f3bifcfa.h"

2.2 Cソースプログラムの翻訳

Cソースプログラムを翻訳します。
Cコンパイラに、インクルードファイルを検索するパスを指定する翻訳オプションを指定してください。この翻訳オプションに、NetCOBOLをインストールしたフォルダを指定します。
以下に、Visual C++(R)での翻訳オプションを示します。

- /I *directory*

*directory*には、NetCOBOLをインストールしたフォルダを指定します。

例

“/I C:¥COBOL”

2.3 オブジェクトファイルのリンク

オブジェクトファイルをリンクし、実行可能プログラムを作成します。
オブジェクトファイルをリンクするときは、NetCOBOLをインストールしたフォルダの中にある以下のファイルを結合します。

- F3BIFCFA.lib

2.4 プログラムの実行

作成したアプリケーションプログラムを実行します。
このとき、特に考慮すべきことはありません。プログラムは、COBOLアプリケーションとファイル/レコードを排他/共用できます。

第3章 API関数と構造体

ここでは、以下について説明します。

3.1 ファイルのオープン

ここでは入出力機能のAPI関数のうち、ファイルをオープンするAPI関数について説明します。

3.1.1 cobfa_open()

ファイルをオープンします。

```
------------------------------------------------------------------------
long cobfa_open (
       const char              *fname,     /* ファイル名          */
       long                     openflgs,  /* オープン属性        */
       const struct fa_keylist *keylist,   /* レコードキーリスト */
       long                     reclen     /* レコード長          */
     );
------------------------------------------------------------------------
```

説明

ファイル名*fname*が指すファイルを、オープン属性*openflgs*、レコード長*reclen*、レコードキーリスト*keylist*の情報をもとにオープンします。
オープン属性*openflgs*の指定値には以下のa.からi.までの9つのカテゴリがあり、これらをビットの論理和で結合して指定します。カテゴリb.からi.までは省略可能です。(*は省略値)

a. オープンモード

	記号定数	意味	対応するCOBOL構文
	FA_INPUT	INPUTモード	OPEN INPUT
	FA_OUTPUT	OUTPUTモード	OPEN OUTPUT
	FA_INOUT	I-Oモード	OPEN I-O
	FA_EXTEND	EXTENDモード	OPEN EXTEND

― ファイル編成が行順ファイル(FA_LSEQFILE)の場合、オープンモードにI-Oモード(FA_INOUT)を指定することはできません。

b. ファイル編成

	記号定数	意味	対応するCOBOL構文
*	FA_SEQFILE	レコード順ファイル	ORGANIZATION IS SEQUENTIAL
	FA_LSEQFILE	行順ファイル	ORGANIZATION IS LINE SEQUENTIAL
	FA_RELFILE	相対ファイル	ORGANIZATION IS RELATIVE
	FA_IDXFILE	索引ファイル	ORGANIZATION IS INDEXED

c. レコード形式

	記号定数	意味	対応するCOBOL構文
*	FA_FIXLEN	固定長形式	RECORD CONTAINS integer CHARACTERS
	FA_VARLEN	可変長形式	RECORD IS VARYING IN SIZE

d. 呼出し法

	記号定数	意味	対応するCOBOL構文
*	FA_SEQACC	順呼出し	ACCESS MODE IS SEQUENTIAL
	FA_RNDACC	乱呼出し	ACCESS MODE IS RANDOM
	FA_DYNACC	動的呼出し	ACCESS MODE IS DYNAMIC

— ファイル編成が以下のどちらかの場合、呼出し法に順呼出し(FA_SEQACC)以外を指定することはできません。
  - 行順ファイル(FA_LSEQFILE)、または
  - レコード順ファイル(FA_SEQFILE)

e. ロックモード

	記号定数	意味	対応するCOBOL構文
	FA_AUTOLOCK	自動ロック	LOCK MODE IS AUTOMATIC
	FA_MANULOCK	手動ロック	LOCK MODE IS MANUAL
	FA_EXCLLOCK	排他ロック	LOCK MODE IS EXCLUSIVE　または、OPEN WITH LOCK

— オープンモードがOUTPUTモード(FA_OUTPUT)の場合、ロックモードに排他ロック(FA_EXCLLOCK)を指定したものとして扱います。
— オープンモードがINPUTモード(FA_INPUT)の場合、ロックモードに自動ロック(FA_AUTOLOCK)または手動ロック(FA_MANULOCK)を指定することはできません。
— オープンモードがINPUTモード(FA_INPUT)の場合、ロックモードのデフォルト値として、共用モードでファイルをオープンします。読込み時は、レコードロックの指定が無効になります。
— オープンモードがINPUTモード(FA_INPUT)以外の場合、ロックモードのデフォルト値は排他ロック(FA_EXCLLOCK)になります。
— ファイル編成が以下のどちらかの場合、ロックモードに手動ロック(FA_MANULOCK)を指定することはできません。
  - 行順ファイル(FA_LSEQFILE)、または
  - レコード順ファイル(FA_SEQFILE)

f. 不定ファイル

	記号定数	意味	対応するCOBOL構文
*	FA_NOTOPT	不定ファイルでない	SELECT filename
	FA_OPTIONAL	不定ファイル	SELECT OPTIONAL filename

g. 動作コード系

	記号定数	意味
*	FA_ASCII	ファイルはSJISまたはJIS8でエンコードした文字データを持つ
	FA_UCS2	ファイルはUCS-2でエンコードした文字データを持つ
	FA_UTF8	ファイルはUTF-8でエンコードした文字データを持つ

— 動作コード系は、行順ファイルが持つ文字データのエンコード種別を指定するもの

です。行順ファイルは、ファイルの構造がエンコード種別により異なるため、この指定が必要になります。
- 動作コード系は、ファイル編成が行順ファイル(FA_LSEQFILE)のときだけ指定することができます。その他のファイル編成は、ファイルの構造がエンコード種別に依存しないため、動作コード系を指定する必要はありません。
- 動作モードがUnicodeのCOBOLアプリケーションで扱う行順ファイルには、FA_UCS2またはFA_UTF8のいずれかを指定します。レコードのデータ項目が日本語項目の場合には前者を、それ以外の場合は後者を指定します。
- 動作モードがUnicodeでないCOBOLアプリケーションで扱う行順ファイルにはFA_ASCIIを指定します。

h. キーパートフラグ使用指定

	記号定数	意味
	FA_USEKPFLAGS	struct fa_keypart型の構造体のkp_flagsメンバの指定値を有効とする

- キーパートフラグ使用指定は、レコードキーリスト*keylist*が包含するstruct fa_keypart型の構造体のメンバkp_flagsの指定値を有効とするときに設定します。struct fa_keypart型については、“3.15.1 struct fa_keydesc”を参照してください。
- キーパートフラグ使用指定は、ファイル編成が索引ファイル(FA_IDXFILE)のときだけ指定することができます。
- 当指定は、動作モードがUnicodeのCOBOLアプリケーションで扱う索引ファイルを使用する場合に必要です。
- 当指定の省略時は、メンバkp_flagsの指定値を無視します。

i. BSAM指定(高速処理およびファイルの最大サイズ拡張)

	記号定数	意味
	FA_BSAM	BSAM（ファイルの高速処理）指定で操作する

- COBOLアプリケーションによる「ファイルの高速処理」で作成されたCOBOLファイルを参照する場合や、COBOLアプリケーションによる「ファイルの高速処理」で参照されるファイルを作成する場合に指定します。
- ファイル編成がレコード順ファイル(FA_SEQFILE)または、行順ファイル(FA_LSEQFILE)の場合にのみ、利用できます。
- FA_BSAMを指定することにより、「ファイルの高速処理」と同じファイル最大サイズの拡張、制限、注意事項が発生します。詳しくは、“NetCOBOL 使用手引書”の“7.7.4 ファイルの高速処理”を参照してください。

レコード形式が固定長形式(FA_FIXLEN)の場合、レコード長*reclen*を固定レコード長として扱います。レコード形式が可変長形式(FA_VARLEN)の場合、レコード長*reclen*を最大レコード長として扱います。レコード長はFA_NRECSIZE(32760)を超えてはいけません。
なお、当アクセスルーチンでは可変長形式のときにユーザが最小レコード長を与えるインタフェースがありません。したがって、最小レコード長未満のレコードの書出し/更新でエラーが発生することはありません。
レコードキーリスト*keylist*は、ファイル編成が索引ファイル(FA_IDXFILE)である場合にだけ意味を持ちます。この場合、*keylist*はオープンするファイルの主レコードキー、副レコードキーの構成として有効になります。struct fa_keylist型については“3.15.2 struct fa_keylist”を参照してください。

索引ファイルのオープンで、*keylist*にNULLポインタを指定した場合、当関数は既存のファイルの索引構成とレコード形式、レコード長を認識してオープンします。このとき、レコード形式の指定とレコード長の指定は無効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	○
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	○
	I-Oモード	○
	EXTENDモード	○
呼出し法	順呼出し	○
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。

－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
1以上	成功	当関数の実行が成功しました。入出力状態が、状況を示すコードを保持していることがあります。 この復帰値は、オープンに成功したファイルのファイルディスクリプタの値です。ただし、ファイルディスクリプタはファイルのオープン時にＯＳが返却したファイルハンドルの値ではありませんので、注意してください。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。
FA_ENOERR	5	不定ファイルが存在しなかったため、仮想的にファイルをオープンしました。または、オープンモードがINPUTモード(FA_INPUT)以外である場合、ファイルを新規に創成しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EBADACC	90	指定したファイル編成、呼出し法、オープンモードの組み合わせでは実行することができません。
FA_EFNAME	35	ファイルが存在しません。
FA_EFLOCKED	93	ファイルはすでに排他でオープンされています。
FA_EFNAME	90	ファイル名が正しくありませんでした。または、ファイルへのアクセスが失敗しました。
FA_EFNAME	91	ファイル名を指定していません。
FA_EBADFLAG	39	指定したファイル編成やレコード形式などの属性と、既存ファイルの構成が異なっています。
FA_EBADKEY	39	指定したレコードキーの情報と、既存の索引ファイルのキーの構成が異なっています。
FA_EBADKEY	90	指定したレコードキーの情報が正しくありません。
FA_EBADLENG	39	指定したレコード長と、既存ファイルのレコード長が異なっています。
FA_EBADFILE	90	指定した索引ファイルは、内部情報が破壊しています。または、ファイル編成が索引ファイルではありません。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.2 ファイルのクローズ

ここでは入出力機能のAPI関数のうち、ファイルをクローズするAPI関数について説明します。

3.2.1 cobfa_close()

ファイルをクローズします。

```
------------------------------------------------------------------------
long cobfa_close (
       long fd  /* ファイルディスクリプタ */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が指すファイルをクローズします。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	○
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	○
	I-Oモード	○
	EXTENDモード	○
呼出し法	順呼出し	○
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。

－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	42	不正なファイルディスクリプタを指定しています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.3 レコードの読込み

ここでは入出力機能のAPI関数のうち、ファイルが持つレコードを読み込むAPI関数について説明します。

3.3.1 cobfa_rdkey()

任意のレコードキーの値で示すレコードを読み込みます。(乱読込み)

```
------------------------------------------------------------------------
long cobfa_rdkey (
        long                        fd,         /* ファイルディスクリプタ */
        long                        readflgs,   /* 読込み属性             */
        char                       *recarea,    /* レコード域             */
        const struct fa_keydesc *keydesc,    /* レコードキー構成指定   */
        long                        keynum      /* レコードキー番号指定   */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が指すファイルに対して、レコード域*recarea*の任意のレコードキーの値でレコードを指定します。指定されたレコードを読み込み、レコード域*recarea*に格納します。

読込み属性*readflgs*の指定値には以下の2つのカテゴリがあり、これらをビットの論理和で結合して指定します。これらは省略可能です。(*は省略値)

a. 読込みモード

	記号定数	意味
*	FA_EQUAL	指定したレコードキーの値に該当するレコードを読み込む

b. レコードロックフラグ

	記号定数	意味	対応するCOBOL構文
	FA_LOCK	ロックありで読み込む	READ WITH LOCK
	FA_NOLOCK	ロックなしで読み込む	READ WITH NO LOCK

— レコードロックフラグのデフォルト値は、オープン時のロックモードの指定により異なります。ロックモードが自動ロック(FA_AUTOLOCK)である場合はデフォルト値がロックあり(FA_LOCK)となり、それ以外の場合のデフォルト値はロックなし(FA_NOLOCK)となります。

任意のレコードキーの指定は、レコードキー構成指定*keydesc*で行います。struct fa_keydesc型については、“3.15.1 struct fa_keydesc”を参照してください。

このレコードキー構成指定にNULLを指定した場合は、任意のレコードキーの指定としてレコード

キー番号指定*keynum*が有効になります。主レコードキーを指定するには、レコードキー番号指定に1を指定します。副レコードキーを指定するには、レコードキー番号指定に2以上の値を指定します。この値は索引ファイルを創成したときの副レコードキーを宣言したときの並びの順番に対応しています。最初の副レコードキーなら2を、2番目の副レコードキーなら3を、それ以降もこれらと同様に指定します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	－
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。

－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。入出力状態が、状況を示すコードを保持していることがあります。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。
FA_ENOERR	2	読み込んだレコードの参照キーの値が、次に続くレコードの参照キーの値と同じです。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	47	指定したファイルは、INPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。

エラー番号	入出力状態	説明
FA_EBADFLAG	90	当関数を実行することができない読込みモードを指定しています。または、その他のフラグの指定が正しくありません。
FA_ENOREC	23	任意のレコードキーの値が示すレコードが存在しません。
FA_EBADKEY	90	指定したレコードキー構成またはレコードキー番号は存在しません。または、正しくありません。
FA_ELOCKED	99	主レコードキーの値で指定したレコードはロックされています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.3.2 cobfa_rdnext()

レコードを順に読み込みます。(順読込み)

```
------------------------------------------------------------------------
long cobfa_rdnext (
        long   fd,          /* ファイルディスクリプタ */
        long   readflgs,    /* 読込み属性             */
        char *recarea       /* レコード域             */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、位置付けられているレコードの次(または前)のレコードを読み込み、レコード域*recarea*に格納します。
読込み属性*readflgs*の指定値には以下の2つのカテゴリがあり、これらをビットの論理和で結合して指定します。これらは省略可能です。(*は省略値)

a. 読込みモード

	記号定数	意味	対応するCOBOL構文
*	FA_NEXT	次のレコードを読み込む	READ NEXT RECORD
	FA_PREV	前のレコードを読み込む	READ PREVIOUS RECORD

― ファイル編成が以下のどちらかの場合、読込みモードに前のレコード(FA_PREV)を指定することはできません。
  - 行順ファイル(FA_LSEQFILE)、または
  - レコード順ファイル(FA_SEQFILE)

b. レコードロックフラグ

	記号定数	意味	対応するCOBOL構文
	FA_LOCK	ロックありで読み込む	READ WITH LOCK

	記号定数	意味	対応するCOBOL構文
	FA_NOLOCK	ロックなしで読み込む	READ WITH NO LOCK

― レコードロックフラグのデフォルト値は、オープン時のロックモードの指定により異なります。ロックモードが自動ロック(FA_AUTOLOCK)の場合は、デフォルト値がロックあり(FA_LOCK)となり、それ以外の場合のデフォルト値はロックなし(FA_NOLOCK)となります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	○
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	○
	乱呼出し	－
	動的呼出し	○

○：当関数の実行が可能です。ただし、ファイルの位置付けが不定でないことが 必須条件となります。

－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。入出力状態が、状況を示すコードを保持していることがあります。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。
FA_ENOERR	2	読み込んだレコードの参照キーの値が、次に続くレコードの参照キーの値と同じです。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	47	指定したファイルは、INPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができない呼出し法で、ファイルがオープンしてあります。
FA_EENDFILE	10	ファイル終了条件が発生しました。
FA_EBADFLAG	90	当関数を実行することができない読込みモードを指定しています。または、その他のフラグの指定が正しくありません。
FA_ENOCURR	46	レコードへの位置付けが不定でした。
FA_ELOCKED	99	順読込みによって位置付けようとしたレコードはすでにロックされています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.3.3 cobfa_rdrec()

相対レコード番号が示すレコードを読み込みます。(乱読込み)

```
------------------------------------------------------------------------
long cobfa_rdrec (
        long            fd,         /* ファイルディスクリプタ */
        long            readflgs,   /* 読込み属性             */
        char           *recarea,    /* レコード域             */
        unsigned long   recnum      /* 相対レコード番号       */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、相対レコード番号*recnum*が指すレコードを読み込み、レコード域*recarea*に格納します。
読込み属性*readflgs*の指定値には、以下の2つのカテゴリがあり、これらをビットの論理和で結合して指定します。これらは省略可能です。(*は省略値)

a. 読込みモード

	記号定数	意味
*	FA_EQUAL	指定した相対レコード番号のレコードを読み込む

b. レコードロックフラグ

	記号定数	意味	対応するCOBOL構文
	FA_LOCK	ロックありで読み込む	READ WITH LOCK
	FA_NOLOCK	ロックなしで読み込む	READ WITH NO LOCK

― レコードロックフラグのデフォルト値は、オープン時のロックモードの指定により異なります。ロックモードが自動ロック(FA_AUTOLOCK)の場合は、デフォルト値がロックあり(FA_LOCK)となり、それ以外の場合のデフォルト値はロックなし(FA_NOLOCK)となります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	―
	レコード順ファイル	―
	相対ファイル	○
	索引ファイル	―
オープンモード	INPUTモード	○
	OUTPUTモード	―
	I-Oモード	○
	EXTENDモード	―
呼出し法	順呼出し	―
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。

―：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	47	指定したファイルは、INPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。

エラー番号	入出力状態	説明
FA_EBADACC	90	当関数を実行することができないファイル編成または呼出し法で、ファイルがオープンしてあります。
FA_EBADFLAG	90	当関数を実行することができない読込みモードを指定しています。または、その他のフラグの指定が正しくありません。
FA_ENOREC	23	相対レコード番号が示すレコードが存在しません。
FA_ELOCKED	99	相対レコード番号で指定したレコードはロックされています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.4 レコードの書出し

ここでは入出力機能のAPI関数のうち、ファイルにレコードを書き出すAPI関数について説明します。

3.4.1 cobfa_wrkey()

主レコードキーの値で指定したレコードを書き出します。(乱書出し)

```
------------------------------------------------------------------------
long cobfa_wrkey (
        long        fd,       /* ファイルディスクリプタ */
        const char *recarea,  /* レコード域             */
        long        reclen    /* 書出しレコード長       */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、レコード域*recarea*が持つ主レコードキーの値が指すレコードを、レコード域が持つ内容で書き出します。
ファイルが可変長形式であるときだけ、書出しレコード長*reclen*が有効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	－
	索引ファイル	○
オープンモード	INPUTモード	－
	OUTPUTモード	○
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EDUPL	22	書出ししようとしたレコードの主レコードキーまたは副レコードキーの値が、すでにファイル中に存在しています。しかし、主レコードキーまたは副レコードキーは重複を許可していません。
FA_EBADLENG	44	指定した書出しレコード長の値が指定可能な範囲を超えています。
FA_ENOTOPEN	48	指定したファイルは、OUTPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。

注

これらは代表的なステータスです。これ以外のステータスについては“4.1 エラー番号”と、“4.2 入出力状態”を参照してください。

3.4.2 cobfa_wrnext()

レコードを順に書き出します。(順書出し)

```
------------------------------------------------------------------------
long cobfa_wrnext (
       long        fd,       /* ファイルディスクリプタ */
       const char *recarea,  /* レコード域             */
       long        reclen    /* 書出しレコード長       */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、レコード域*recarea*が持つ内容で、レコードを順に書き出します。
ファイルが可変長形式の場合だけ、書出しレコード長*reclen*が有効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	○
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○

オープン属性	種別	実行の可否
オープンモード	INPUTモード	－
	OUTPUTモード	○
	I-Oモード	－
	EXTENDモード	○
呼出し法	順呼出し	○
	乱呼出し	－
	動的呼出し	－

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EDUPL	22	書き出そうとしたレコードの主レコードキーまたは副レコードキーの値がすでにファイル中に存在しています。しかし、主レコードキーまたは副レコードキーは重複を許可していません。
FA_EBADLENG	44	指定した書出しレコード長の値が指定可能な範囲を超えています。
FA_ENOTOPEN	48	指定したファイルは、OUTPUTモード以外、かつ、EXTENDモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.4.3 cobfa_wrrec()

相対レコード番号で指定したレコードを書き出します。(乱書出し)

```
------------------------------------------------------------------------
long cobfa_wrrec (
        long            fd,         /* ファイルディスクリプタ */
        const char     *recarea,    /* レコード域             */
        long            reclen,     /* 書出しレコード長       */
```

```
    unsigned long  recnum    /* 相対レコード番号        */
  );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタfdが示すファイルで、相対レコード番号*recnum*が指すレコードを、レコード域*recarea*が持つ内容で書き出します。
ファイルが可変長形式の場合だけ、書出しレコード長*reclen*が有効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	○
	索引ファイル	－
オープンモード	INPUTモード	－
	OUTPUTモード	○
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EDUPL	22	すでに存在する相対レコード番号を使って書出しを行おうとしました。
FA_EBADLENG	44	指定した書出しレコード長の値が指定可能な範囲を超えています。
FA_ENOTOPEN	48	指定したファイルは、OUTPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.5 レコードの削除

ここでは入出力機能のAPI関数のうち、ファイルが持つレコードを削除するAPI関数について説明します。

3.5.1 cobfa_delcurr()

順読込みしたレコードを削除します。(順削除)

```
------------------------------------------------------------------------
long cobfa_delcurr (
       long  fd  /* ファイルディスクリプタ */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、順読込みによって位置付けられているレコード(カレントレコード)を削除します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	―
	レコード順ファイル	―
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	―
	OUTPUTモード	―
	I-Oモード	○
	EXTENDモード	―
呼出し法	順呼出し	○
	乱呼出し	―
	動的呼出し	―

○：当関数の実行が可能です。ただし、レコードが順読込みによって位置付けら れていることが必須条件となります。
―：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	49	指定したファイルは、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。
FA_ELOCKED	99	順読込みによって位置付けられているレコードはロックされています。
FA_ENOCURR	43	指定したファイルでの順読込みが成功していませんでした。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.5.2 cobfa_delkey()

主レコードキーの値が示すレコードを削除します。(乱削除)

```
------------------------------------------------------------------------
long cobfa_delkey (
       long        fd,       /* ファイルディスクリプタ */
       const char *recarea   /* レコード域             */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、レコード域*recarea*の内容が持つ主レコードキーの値が指すレコードを削除します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	－
	索引ファイル	○
オープンモード	INPUTモード	－
	OUTPUTモード	－
	I-Oモード	○

オープン属性	種別	実行の可否
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	49	指定したファイルは、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。
FA_ELOCKED	99	主レコードキーの値で指定したレコードはロックされています。
FA_ENOREC	23	主レコードキーの値で指定したレコードは存在しません。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.5.3 cobfa_delrec()

相対レコード番号が示すレコードを削除します。(乱削除)

```
------------------------------------------------------------------------
long cobfa_delrec (
       long          fd,      /* ファイルディスクリプタ */
       unsigned long recnum   /* 相対レコード番号       */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、相対レコード番号*recnum*が指すレコードを削除します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	○
	索引ファイル	－
オープンモード	INPUTモード	－
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	49	指定したファイルは、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。
FA_ELOCKED	99	相対レコード番号で指定したレコードはロックされています。
FA_ENOREC	23	相対レコード番号で指定したレコードは存在しません。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.6 レコードの書換え

ここでは入出力機能のAPI関数のうち、ファイルが持つレコードを書き換えるAPI関数について説明します。

3.6.1 cobfa_rewcurr()

順読込みしたレコードを書き換えます。(順更新)

```
------------------------------------------------------------------------
long cobfa_rewcurr (
        long         fd,        /* ファイルディスクリプタ */
        const char *recarea,    /* レコード域             */
        long         reclen     /* 書換えレコード長       */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、順読込みによって位置付けられているレコード(カレントレコード)を、レコード域*recarea*の内容で書き換えます。
ファイルが可変長形式であるときだけ、書換えレコード長*reclen*の指定が有効になります。ファイル編成がレコード順ファイル(FA_SEQFILE)のときは元のレコード長を変えることはできません。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	－
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	○
	乱呼出し	－
	動的呼出し	－

○：当関数の実行が可能です。ただし、レコードが順読込みによって位置付けら れていることが必須条件となります。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。

復帰値	状態	説明
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EDUPL	22	書き換えようとしたレコードの主レコードキーまたは、副レコードキーの値がすでにファイル中に存在しています。しかし、主レコードキーまたは副レコードキーは重複を許可していません。
FA_EBADLENG	44	指定した書換えレコード長の値が指定可能な範囲を超えています。

注

この表にあるステータス以外に発生するものは、“3.5.1 cobfa_delcurr()”の発生するステータスを参照してください。

3.6.2 cobfa_rewkey()

主レコードキーの値が示すレコードを書き換えます。(乱更新)

```
------------------------------------------------------------------------
long cobfa_rewkey (
       long        fd,       /* ファイルディスクリプタ */
       const char *recarea,  /* レコード域             */
       long        reclen    /* 書換えレコード長       */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルの、レコード域*recarea*の中に持つ主レコードキーの値が指すレコードを、レコード域*recarea*の内容で書き換えます。
ファイルが可変長形式であるときだけ、書換えレコード長*reclen*の指定が有効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	－
	索引ファイル	○
オープンモード	INPUTモード	－

オープン属性	種別	実行の可否
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EDUPL	22	重複を許可していないレコードキーに対して、すでに存在するレコードキーの値を持つレコード内容で書き換えようとしました。
FA_EBADLENG	44	指定した書換えレコード長の値が指定可能な範囲を超えています。

注

この表にあるステータス以外に発生するものは、“3.5.2 cobfa_delkey()”の発生するステータスを参照してください。

3.6.3 cobfa_rewrec()

相対レコード番号が示すレコードを書き換えます。(乱更新)

```
------------------------------------------------------------------------
long cobfa_rewrec (
       long           fd,        /* ファイルディスクリプタ */
       const char    *recarea,   /* レコード域             */
       long           reclen,    /* 書換えレコード長       */
       unsigned long  recnum     /* 相対レコード番号       */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルの、相対レコード番号*recnum*のレコードを、レコード域*recarea*の内容で書き換えます。
ファイルが可変長形式であるときだけ、書換えレコード長*reclen*の指定が有効になります。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	○
	索引ファイル	－
オープンモード	INPUTモード	－
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	－
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_EBADLENG	44	指定した書換えレコード長の値が指定可能な範囲を超えています。

注

この表にあるステータス以外に発生するものは、“3.5.3 cobfa_delrec()”の発生するステータスを参照してください。

3.7 レコードの位置決め

ここでは入出力機能のAPI関数のうち、ファイルが持つレコードをアクセスする位置を決めるAPI関数について説明します。

3.7.1 cobfa_stkey()

任意のレコードキーの値が示すレコードに位置付けます。

```
------------------------------------------------------------------------
long cobfa_stkey (
       long                        fd,       /* ファイルディスクリプタ */
       long                        stflgs,   /* 位置付け属性           */
       const char                  *recarea, /* レコード域             */
       const struct fa_keydesc *keydesc,     /* レコードキー構成指定   */
       long                        keynum,   /* レコードキー番号指定   */
       long                        keyleng   /* 有効キー長             */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、レコード域*recarea*の中に持つ任意のレコードキーの値に関連するレコードに位置付けを行います。また、指定した任意のレコードキーを、以降の順読込みでの参照キーとして宣言します。

位置付け属性*stflgs*の指定値には以下の2つのカテゴリがあり、これらをビットの論理和で結合して指定します。これらは省略可能です。(*は省略値)

a. 位置付けモード

	記号定数	意味	対応するCOBOL構文
	FA_FIRST	先頭レコード	START FIRST RECORD
*	FA_EQUAL	レコードキーの値と等しいレコード	START KEY IS =
	FA_GREAT	レコードキーの値を超えるレコード	START KEY IS >
	FA_GTEQ	レコードキーの値以上のレコード	START KEY IS >=
	FA_LESS	レコードキーの値より小さいレコード	START KEY IS <
	FA_LTEQ	レコードキーの値以下のレコード	START KEY IS <=

b. 逆順読込みフラグ

	記号定数	意味	対応するCOBOL構文
*	なし	順読込み時に論理的に順方向に読み込む	−

	記号定数	意味	対応するCOBOL構文
	FA_REVORD	順読込み時に論理的に逆方向に読み込む	START WITH REVERSED ORDER

— 逆順読込みフラグのデフォルト値は、正順読込みになります。
— 位置付けモードがレコードキーの値を超過(FA_GREAT)またはレコードキーの値以上(FA_GTEQ)の場合、逆順読込みフラグ(FA_REVORD)を指定することはできません。
— 逆順読込みフラグの指定は、乱読込み、位置付け、ファイルのクローズを行ったり、ファイル終了条件が発生したりすることで無効になります。

任意のレコードキーの指定は、レコードキー構成指定*keydesc*で行います。struct fa_keydesc型については“3.15.1 struct fa_keydesc”を参照してください。
このレコードキー構成指定にNULLを指定した場合は、任意のレコードキーの指定としてレコードキー番号指定*keynum*が有効になります。主レコードキーを指定するには、レコードキー番号指定に1を指定します。副レコードキーを指定するには、レコードキー番号指定に2以上の値を指定します。この値は索引ファイルを創成したときの副レコードキーを宣言したときの並びの順番に対応しています。最初の副レコードキーなら2を、2番目の副レコードキーなら3を、それ以上もこれらと同様に指定します。
有効キー長*keyleng*は、有効な参照キーの長さを短くするために使います。有効な参照キーの長さを短くしない通常の場合は、0を指定します。この場合、任意のレコードキーの全体が参照キーとなります。有効な参照キーの長さを短くする場合は、1以上の値を指定します。この値は、キーパートの並びを連続した先頭からのキーの長さをバイト単位で指定します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	－
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	○
	乱呼出し	－
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	47	指定したファイルは、INPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。
FA_EBADFLAG	90	当関数を実行することができない位置付けモードを指定しています。または、その他のフラグの指定が正しくありません。
FA_ENOREC	23	指定した条件に該当するレコードが存在しませんでした。
FA_EBADKEY	90	指定したレコードキー構成またはレコードキー番号は存在しません。または、正しくありません。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.7.2 cobfa_strec()

相対レコード番号が示すレコードに位置付けます。

```
------------------------------------------------------------------------
long cobfa_strec (
       long          fd,       /* ファイルディスクリプタ */
       long          stflgs,   /* 位置付け属性           */
       unsigned long recnum    /* 相対レコード番号       */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、相対レコード番号*recnum*の値に関連するレコードに位置付けを行います。

位置付け属性*stflgs*の指定値には以下の1つのカテゴリがあります。これは省略可能です。(*は省略値)

a. 位置付けモード

	記号定数	意味	対応するCOBOL構文
*	FA_EQUAL	recnumと等しいレコード	START KEY IS =
	FA_GREAT	recnumを超えるレコード	START KEY IS >
	FA_GTEQ	recnum以上のレコード	START KEY IS >=

	記号定数	意味	対応するCOBOL構文
	FA_LESS	recnumより小さいレコード	START KEY IS <
	FA_LTEQ	recnum以下のレコード	START KEY IS <=

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	－
	相対ファイル	○
	索引ファイル	－
オープンモード	INPUTモード	○
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	○
	乱呼出し	－
	動的呼出し	○

○：当関数の実行が可能です。

－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	47	指定したファイルは、INPUTモード以外、かつ、I-Oモード以外でオープンしてあります。または、不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	当関数を実行することができないファイル編成または、呼出し法でファイルがオープンしてあります。
FA_EBADFLAG	90	当関数を実行することができない位置付けモードを指定しています。または、その他のフラグの指定が正しくありません。

エラー番号	入出力状態	説明
FA_ENOREC	23	指定した条件に該当するレコードが存在しませんでした。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.8 レコードロックの解除

ここでは入出力機能のAPI関数のうち、ファイルが持つロック中のレコードをロック解除するAPI関数について説明します。

3.8.1 cobfa_release()

指定するファイルのすべてのレコードロックを解除します。

```
------------------------------------------------------------------------
long cobfa_release (
       long  fd  /* ファイルディスクリプタ */
    );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示すファイルで、すべてのレコードロックを解除します。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	－
	レコード順ファイル	○
	相対ファイル	○
	索引ファイル	○
オープンモード	INPUTモード	－
	OUTPUTモード	－
	I-Oモード	○
	EXTENDモード	－
呼出し法	順呼出し	○
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
－：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時(注)

エラー番号	入出力状態	説明
FA_ENOTOPEN	90	ファイルのオープン時に取得したものでない不正なファイルディスクリプタを指定しています。

注

これらは代表的なステータスです。これ以外のステータスについては、“4.1 エラー番号”と“4.2 入出力状態”を参照してください。

3.9 ファイル情報の取得

ここでは、ファイルの情報を取得する機能を持つAPI関数について説明します。

3.9.1 cobfa_indexinfo()

索引ファイルの属性またはレコードキーの構成を取得します。

```
------------------------------------------------------------------------
long cobfa_indexinfo (
       long                fd,        /* ファイルディスクリプタ */
       struct fa_keydesc *buffer,     /* 取得結果の格納域       */
       long                funccode   /* 機能コード             */
     );
------------------------------------------------------------------------
```

説明

ファイルディスクリプタ*fd*が示す索引ファイルに関する情報を、取得結果の格納域*buffer*内に設定します。
取得する情報の種類は機能コード*funccode*への指定で選択します。
機能コードに0を指定すると、取得結果の格納域にstruct fa_dictinfo型でファイルの属性を格納します。struct fa_dictinfo型については、“3.15.3 struct fa_dictinfo”を参照してください。
機能コードに1以上を指定すると、取得結果の格納域にstruct fa_keydesc型で主レコードキーまたは副レコードキーの構成を格納します。機能コードの値に1を指定するとき主レコードキーの構成を取得し、2以上では副レコードキーの構成を取得します。
この値は、索引ファイルを創成したときの副レコードキーを宣言したときの並びの順番に対応しています。最初の副レコードキーなら2を、2番目の副レコードキーなら3を、これ以降もこれらと同様に指定します。
struct fa_keydesc型については、“3.15.1 struct fa_keydesc”を参照してください。

実行可能な条件

当関数の機能を実行することができるファイル編成、オープンモードおよび呼出し法は、それぞれ下表のとおりです。

オープン属性	種別	実行の可否
ファイル編成	行順ファイル	―
	レコード順ファイル	―
	相対ファイル	―
	索引ファイル	○
オープンモード	INPUTモード	○
	OUTPUTモード	○
	I-Oモード	○
	EXTENDモード	○
呼出し法	順呼出し	○
	乱呼出し	○
	動的呼出し	○

○：当関数の実行が可能です。
―：当関数の実行はできません。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	当関数の実行が成功しました。
-1	失敗	当関数の実行が失敗しました。エラー番号と入出力状態が、状況を示すコードを保持しています。

発生するステータス

当関数の呼出しによって発生するステータスは、下表のようになります。

関数の実行の成功時

エラー番号	入出力状態	説明
FA_ENOERR	0	関数の実行は成功しました。

関数の実行の失敗時

エラー番号	入出力状態	説明
FA_ENOTOPEN	90	不正なファイルディスクリプタを指定しています。
FA_EBADACC	90	指定したファイルは索引ファイルでありません。

3.10 エラー番号の取得

ここでは、入出力の状況を取得する機能のAPIのうち、エラー番号の取得を行うAPI関数について説明します。

3.10.1 cobfa_errno()

エラー番号を返却します。

```
------------------------------------------------------------------------
long cobfa_errno (
       void  /* 引数なし */
    );
------------------------------------------------------------------------
```

説明

入出力のAPI関数またはファイル情報取得のAPI関数を実行した結果、起こったエラーを識別する番号を返却します。

実行可能な条件

つねに呼出し可能。

復帰値

当関数の復帰値は、下表のようになります。

復帰値	状態	説明
FA_ENOERR	成功	入出力機能の実行またはファイル情報取得の実行が成功したことを意味します。
FA_ENOERR 以外	失敗	入出力機能の実行またはファイル情報取得の実行が失敗したことを意味します。(注１)

注1

復帰値の詳細については、“4.1 エラー番号”を参照してください。

3.11 入出力状態の取得

ここでは、入出力の状況を取得する機能のAPIのうち、入出力状態の取得を行うAPI関数について説明します。

3.11.1 cobfa_stat()

入出力状態を返却します。

```
--------------------------------------------------------------------------
long cobfa_stat (
       void  /* 引数なし */
     );
--------------------------------------------------------------------------
```

説明

入出力のAPI関数またはファイル情報取得のAPI関数を実行した結果の入出力状態を返却します。

実行可能な条件

つねに呼出し可能。

復帰値

入出力状態を返却します。入出力状態の種類とその詳細については、“NetCOBOL 使用手引書”の“付録B 入出力状態一覧”を参照してください。

3.12 読込みレコード長の取得

ここでは、入出力の状況を取得する機能のAPIのうち、読込みレコード長の取得を行うAPI関数について説明します。

3.12.1 cobfa_reclen()

レコード長を返却します。

```
------------------------------------------------------------------------
long cobfa_reclen (
       void  /* 引数なし */
     );
------------------------------------------------------------------------
```

説明

レコード形式が可変長であるファイルを扱うとき、入出力のAPI関数のうち、以下の関数の実行が成功したあと、読み込んだレコードの実際の長さを返却します。読込み失敗や他の入出力機能の実行後は、不定な値を返却します。

- cobfa_rdnext()
- cobfa_rdrec()
- cobfa_rdkey()

実行可能な条件

つねに呼出し可能。

復帰値

読み込んだレコードの長さ(バイト数)。

3.13 相対レコード番号の取得

ここでは、入出力の状況を取得する機能のAPIのうち、相対レコード番号の取得を行うAPI関数について説明します。

3.13.1 cobfa_recnum()

相対レコード番号を返却します。

```
------------------------------------------------------------------------
unsigned long cobfa_recnum (
                void  /* 引数なし */
              );
------------------------------------------------------------------------
```

説明

相対ファイルで、入出力のAPI関数のうち、以下の関数の実行が成功したあと、現在位置付けられているレコードの相対レコード番号を返却します。

- cobfa_rdnext()
- cobfa_rdrec()

読込みが失敗した後や、相対ファイル以外のファイル編成での入出力機能の実行後には不定な値を返却します。

実行可能な条件

つねに呼出し可能。

復帰値

相対ファイルで、現在位置付けられている相対レコード番号。

3.14 マルチスレッド環境下での排他制御

マルチスレッド環境下では、複数のスレッドが同時にCOBOLランタイムシステムによるファイルアクセスを行わないように、排他制御する必要があります。ここでは、その排他制御を行うための以下の関数について説明します。

3.14.1 LOCK_cobfa()

マルチスレッド環境下でのCOBOLファイルアクセスに排他ロックをかけます。

```
------------------------------------------------------------------------
long LOCK_cobfa (
       const unsigned long *timeout,  /* 待ち時間     */
       unsigned long       *errcode   /* エラーコード */
     );
------------------------------------------------------------------------
```

説明

他のスレッドが同時にCOBOLファイルへのアクセスを行わないように、COBOLランタイムシステムが行うファイルアクセスに対して排他ロックをかけます。マルチスレッド環境下でファイルへのアクセスを行う場合、競合による問題を発生させないために、この関数を呼び出す必要があります。

待ち時間*timeout*にはミリ秒単位の値を持つ長整数型へのポインタを指定します。もし他のスレッドが先に排他ロックをかけているとき、当スレッドでの関数の呼出しから*timeout*までの間に排他ロックが解除されないと、関数の実行は失敗します。*timeout*にNULLを指定した場合、待ち時間は無制限となります。

当関数の復帰値が-1であるとき、エラーコード**errcode*にWindowsシステムエラーコードを設定します。ただし、*errcode*にNULLを設定した場合には、エラーコードを設定しません。当関数の復帰値が-1でない場合の値は不定です。

実行可能な条件

つねに呼出し可能。

復帰値

関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	COBOLファイルアクセスの排他ロックが成功しました。
1		COBOLファイルアクセスの排他ロックが成功しました。排他ロックをしていた他のスレッドがロックを解除しないで終了したため、代わりに当スレッドがロック制御権を得ました。他のスレッドが異常な終了をしている可能性がありますので、処理を継続する際には十分な注意が必要です。
-1	失敗	システムエラーが発生しました。この場合、エラーコード**errcode*にWindowsシステムエラーコードが設定されます。
-2		待ち時間*timeout*を超えました。排他ロックをかけませんでした。

使い方

当関数は、入出力機能を持つAPI関数か、ファイルの情報を取得するAPI関数の前に呼びます。API関数呼出しに続けて、状況を取得するAPI関数を必要に応じて呼んだ後、“3.14.2 UNLOCK_cobfa()”を呼びます。
状況を取得する関数の呼出しは、入出力かファイル情報の取得のAPI関数呼出しとともに、必ず一回の排他ロック中に行ってください。数回の排他制御に分けて状況を取得すると、状況を示す値が他のスレッドで上書きされ、正しくない可能性があります。
以下に1つの入出力ごとに排他制御を行う例を示します。

```
------------------------------------------------------------------------
long ret, fd, eno, stat, recnum;

ret = LOCK_cobfa ( NULL, NULL );  /* ファイルアクセスを排他ロックする */
if ( ret < 0 ) { ....  /* エラー処理 */ }
fd = cobfa_open ( "file.rel", FA_INPUT | FA_RELFILE | FA_FIXED, 10 );
stat = cobfa_stat ( );
eno = cobfa_errno ( );
UNLOCK_cobfa ( NULL );  /* ファイルアクセスの排他ロックを解除する */
if ( eno == FA_EFNAME ) {
  printf ( "errno: %d, stat: %d¥n", eno, stat );
  ...
}

....  /* ファイル入出力に関係のないさまざまな処理を行う */

ret = LOCK_cobfa ( NULL, NULL );  /* ファイルアクセスを排他ロックする */
if ( ret < 0 ) { ....  /* エラー処理 */ }
cobfa_rdnext ( fd, FA_NEXT, buff );
recnum = cobfa_recnum ( );
UNLOCK_cobfa ( NULL );  /* ファイルアクセスの排他ロックを解除する */
num = recnum;

....  /* ファイル入出力に関係のないさまざまな処理を行う */

ret = LOCK_cobfa ( NULL, NULL );  /* ファイルアクセスを排他ロックする */
if ( ret < 0 ) { ....  /* エラー処理 */ }
cobfa_close ( fd );
UNLOCK_cobfa ( NULL );  /* ファイルアクセスの排他ロックを解除する */
------------------------------------------------------------------------
```

また、最初にCOBOLファイルへのアクセスを始めてから最後にアクセスが終わるまでの間、排他制御をすることもできます。このように、排他制御を行う範囲は任意に広げることができます。
以下の図に、まとめて排他制御を行う例を示します。

```
------------------------------------------------------------------------
long ret, fd;

ret = LOCK_cobfa ( NULL, NULL );  /* ファイルアクセスを排他ロックする */
if ( ret < 0 ) { ....  /* エラー処理 */ }
fd = cobfa_open ( "file.seq", FA_OUTPUT | FA_SEQFILE | FA_FIXED, 10 );
```

```
cobfa_wrnext ( fd, buffer, 10 );
....  /* その他、ファイルに対する入出力を行う */

cobfa_close ( fd );
UNLOCK_cobfa ( NULL );  /* ファイルアクセスの排他ロックを解除する */
-------------------------------------------------------------------------
```

3.14.2 UNLOCK_cobfa()

マルチスレッド環境下でのファイルアクセスに排他ロックを解除します。

```
-------------------------------------------------------------------------
long UNLOCK_cobfa (
       unsigned long *errcode  /* エラーコード */
     );
-------------------------------------------------------------------------
```

説明

COBOLランタイムシステムのファイルアクセスにかけていた排他ロックを解除します。
当関数の復帰値が-1であるとき、エラーコード**errcode*にWindowsシステムエラーコードを設定します。ただし、*errcode*にNULLを設定した場合、エラーコードを設定しません。当関数の復帰値が-1でない場合の値は不定です。

実行可能な条件

つねに呼び出し可能。

復帰値

関数の復帰値は、下表のようになります。

復帰値	状態	説明
0	成功	COBOLファイルアクセスの排他ロックの解除が成功しました。
-1	失敗	システムエラーが発生しました。この場合、エラーコード**errcode*にWindowsシステムエラーコードが設定されます。

3.15 使用する構造体

ここでは、API関数が使用する構造体について説明します。

3.15.1 struct fa_keydesc

cobfa_rdkey()関数とcobfa_stkey()関数では、任意のレコードキーの選択をstruct fa_keydesc型で指定できます。
cobfa_indexinfo()関数は、オープンしてある索引ファイルの任意のレコードキーの構成を知ることができます。
ここでは、レコードキーの構成を与えるstruct fa_keydesc型のメンバと、設定する/格納される値について説明します。

```
------------------------------------------------------------------------
#define FA_NPARTS 254u                          /* max number of key parts     */

    struct fa_keydesc {
      long  k_flags;                            /* flags (duplicatable or not) */
      long  k_nparts;                           /* number of parts in key      */
      struct fa_keypart k_part [FA_NPARTS];     /* each key part               */
    };
------------------------------------------------------------------------
```

レコードキーの属性を示すk_flagsには以下の値が入ります。

FA_DUPS:
　このレコードキーは重複可能
FA_NODUPS:
　このレコードキーは重複を許可しない

k_npartsには、レコードキーの中のパート(キーパート)の数が入ります。キーパート数の最小値は1で、最大値はFA_NPARTS(254)です。

個々のキーパートの情報を持つk_partは、struct fa_keypart型の配列で宣言しています。以下で説明します。

```
------------------------------------------------------------------------
#define FA_NRECSIZE 32760u  /* max number of bytes in a record */
#define FA_NKEYSIZE 254u    /* max number of bytes in a key    */

struct fa_keypart {
    short kp_start;         /* starting byte of key part       */
    short kp_leng;          /* length in bytes                 */
    long  kp_flags;         /* flags (UCS-2 key part or not)   */
};
------------------------------------------------------------------------
```

kp_startには、レコードの先頭位置を0とするバイト単位の変位を設定します。この変位の最大値はFA_NRECSIZE-1(32759)です。

kp_lengには、キーパートの長さを設定します。この長さの最小値は1で、上位制限値は、FA_NKEYSIZE(254)です。変位と長さの和がFA_NRECSIZE(32760)を超えてはいけません。
kp_flagsには、以下の1つのカテゴリの情報を設定します。このメンバへの設定値は、ファイルのオープン(cobfa_open()関数)のオープン属性引数にキーパートフラグ使用指定(FA_USEKPFLAGS)を指定したときだけ有効になります。

キーパートのコード系種別

記号定数	説明
FA_UCS2KPCODE	キーパートのコード系種別はUCS-2である
FA_ANYKPCODE	キーパートのコード系種別は、上記以外である

動作モードがUnicodeのCOBOLアプリケーションで扱う索引ファイルで、キーパートが日本語項目であるときに、FA_UCS2KPCODEを設定します。キーパートが日本語項目でないときや動作モードがUnicodeでないときには、FA_ANYKPCODEを指定します。
レコードキーの構成の取得(cobfa_indexinfo()関数)でkp_flagsを参照するときには、マスク値FA_KPCODEMASKを用いてキーパートのコード系種別を取り出してください。

FA_KPCODEMASKの使用例

```
#include "f3bifcfa.h"

#define GET_PRIM_KEY  1
struct fa_keydesc keydesc1;
  :
ret = cobfa_indexinfo ( fd, &keydesc1, GET_PRIM_KEY );
for ( i = 0; i < keydesc1.k_nparts; i ++ ) {
    :
  switch ( keydesc1.k_part[i].kp_flags & FA_KPCODEMASK ) {
    case FA_UCS2KPCODE:
       :
      break;
    case FA_ANYKPCODE:
       :
      break;
  }
    :
}
```

設定例

具体的な設定例としてa.とb.の2つを挙げます。

a. 具体的な設定例
   - 主レコードキーは重複可能。キーパートは2つ。
   - 最初のキーパートは先頭から5バイト目にあり、長さは3。
   - 次のキーパートは先頭から11バイト目にあり、長さは5。

```
-------------------------------------------------------------------------
バイト目| 1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
変位    |0--|1--|2--|3--|4--|5--|6--|7--|8--|9--|10-|11-|12-|13-|14-|15-|
        |=========================================================....
                        |---+---+---|           |---+---+---+---+---|
                        最初のパート             次のパート
                         part-1                  part-2
-------------------------------------------------------------------------
```

COBOLで記述した場合の参考例:

```
-------------------------------------------------------------------------
   :
000100 ENVIRONMENT DIVISION.
000200 CONFIGURATION SECTION.
000300 INPUT-OUTPUT SECTION.
000400 FILE-CONTROL.
000500     SELECT FILENAME-1 ASSIGN TO SYS006
000600     ORGANIZATION IS INDEXED
000700     RECORD KEY IS PART-1 PART-2 WITH DUPLICATES.
000800 DATA DIVISION.
000900 FILE SECTION.
001000 FD FILENAME-1.
001100   01 RECORD-1.
001200     02 FILLER PIC X(4).
001300     02 PART-1 PIC X(3).
001400     02 FILLER PIC X(3).
001500     02 PART-2 PIC X(5).
   :
-------------------------------------------------------------------------
```

Cソースプログラム例:

```
-------------------------------------------------------------------------
#include "f3bifcfa.h"

struct fa_keydesc keydesc1;

keydesc1.k_flags = FA_DUPS;
keydesc1.k_nparts = 2;  /* number of key parts: 2 */
keydesc1.k_part[0].kp_start = 4;  /* part_1: 5 - 1 == 4 */
keydesc1.k_part[0].kp_leng = 3;
keydesc1.k_part[0].kp_flags = FA_ANYKPCODE;
keydesc1.k_part[1].kp_start = 10; /* part_2: 11 - 1 == 10 */
keydesc1.k_part[1].kp_leng = 5;
keydesc1.k_part[1].kp_flags = FA_ANYKPCODE;
-------------------------------------------------------------------------
```

b. 具体的な設定例
   - 主レコードキーは重複を許可しない。キーパートは3つ。
   - 1番目のキーパートは先頭から1バイト目にあり、長さは3。
   - 2番目のキーパートは先頭から9バイト目にあり、長さは2。
   - 3番目のキーパートは先頭から5バイト目にあり、長さは4。

```
--------------------------------------------------------------------------------
バイト目| 1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
変位    |0--|1--|2--|3--|4--|5--|6--|7--|8--|9--|10-|11-|12-|13-|14-|15-|
        |=================================================....
        |---+---+---|   |---+---+---+---|---+---|
         1番目のパート   3番目のパート   2番目のパート
         part-1         part-3         part-2
--------------------------------------------------------------------------------
```

COBOLで記述した場合の参考例:

```
--------------------------------------------------------------------------------
   :
000100 ENVIRONMENT DIVISION.
000200 CONFIGURATION SECTION.
000300 INPUT-OUTPUT SECTION.
000400 FILE-CONTROL.
000500     SELECT FILENAME-1 ASSIGN TO SYS006
000600     ORGANIZATION IS INDEXED
000700     RECORD KEY IS PART-1 PART-2 PART-3.
000800 DATA DIVISION.
000900 FILE SECTION.
001000 FD FILENAME-1.
001100   01 RECORD-1.
001200     02 PART-1 PIC X(3).
001300     02 FILLER PIC X(1).
001400     02 PART-3 PIC X(4).
001500     02 PART-2 PIC X(2).
   :
--------------------------------------------------------------------------------
```

Cソースプログラム例:

```
--------------------------------------------------------------------------------
#include "f3bifcfa.h"

struct fa_keydesc keydesc2;

keydesc2.k_flags = FA_NODUPS;
keydesc2.k_nparts = 3;  /* number of key parts: 3 */
keydesc2.k_part[0].kp_start = 0; /* part_1: 1 - 1 == 0 */
keydesc2.k_part[0].kp_leng = 3;
keydesc2.k_part[0].kp_flags = FA_ANYKPCODE;
keydesc2.k_part[1].kp_start = 8; /* part_2: 9 - 1 == 8 */
keydesc2.k_part[1].kp_leng = 2;
keydesc2.k_part[1].kp_flags = FA_ANYKPCODE;
keydesc2.k_part[2].kp_start = 4; /* part_3: 5 - 1 == 4 */
keydesc2.k_part[2].kp_leng = 4;
keydesc2.k_part[2].kp_flags = FA_ANYKPCODE;
--------------------------------------------------------------------------------
```

3.15.2 struct fa_keylist

cobfa_open()関数では、オープンする索引ファイルのすべてのレコードキーの構成をstruct fa_keylist型で指定します。
ここでは、レコードキー全体の構成を指定するstruct fa_keylist型のメンバに設定する値について説明します。

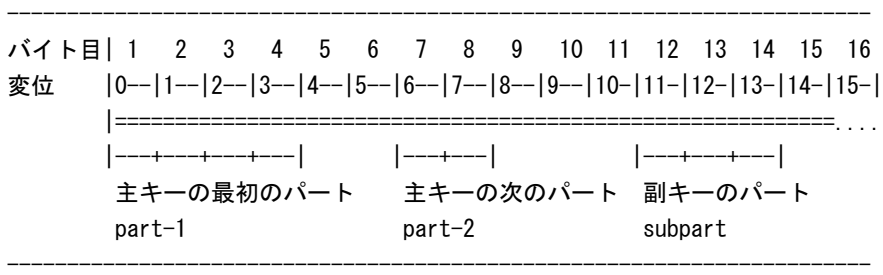
```
------------------------------------------------------------------------
#define FA_NKEYS 126u                  /* max number of all keys        */

struct fa_keylist {
  long kl_nkeys;                        /* number of keydesc           */
  struct fa_keydesc *kl_key [FA_NKEYS]; /* keydesc address of each key */
};
------------------------------------------------------------------------
```

レコードキーの総数を示すkl_nkeysには、レコードキーの総数を設定します。索引ファイルは主レコードキーを必ず含むので、レコードキーの総数は必ず1以上になります。なお、レコードキーの総数の最大値は、FA_NKEYS(126)です。
個々のレコードキーの情報を持つkl_keyは、struct fa_keydesc型のポインタの配列で宣言しています。struct fa_keydesc型については、“3.15.1 struct fa_keydesc”を参照してください。
なお、すべてのレコードキーが持つキーパート数の合計は、FA_NALLPARTS(255)を超えてはいけません。また、すべてのレコードキーが持つ各キーパートの長さの合計がFA_NALLKEYSIZE(255)を超えてはいけません。

設定例

具体的な設定例としてa.を挙げます。

a. 具体的な設定例
   - 索引ファイルのレコードキーの構成は、主レコードキーと副レコードキーが1つずつ、合計2個ある。
   - 主レコードキーは重複を許可しない。
   - 主レコードキーのキーパートは2つ。
   - 最初のキーパートは先頭から1バイト目にあり、長さは4。
   - 次のキーパートは先頭から7バイト目にあり、長さは2。
   - 副レコードキーは重複可能。
   - 副レコードキーのキーパートは1つ。
   - キーパートは先頭から12バイト目にあり、長さは3。



COBOLで記述した場合の参考例:

```
   :
000100 ENVIRONMENT DIVISION.
000200 CONFIGURATION SECTION.
000300 INPUT-OUTPUT SECTION.
000400 FILE-CONTROL.
000500     SELECT FILENAME-1 ASSIGN TO SYS006
000600     ORGANIZATION IS INDEXED
000700     RECORD KEY IS PART-1 PART-2
000800     ALTERNATE RECORD KEY IS SUBPART WITH DUPLICATES.
000900 DATA DIVISION.
001000 FILE SECTION.
001100 FD FILENAME-1.
001200   01 RECORD-1.
001300     02 PART-1  PIC X(4).
001400     02 FILLER  PIC X(2).
001500     02 PART-2  PIC X(2).
001600     02 FILLER  PIC X(3).
001700     02 SUBPART PIC X(3).
   :
```

Cソースプログラム例:

```
#include "f3bifcfa.h"

struct fa_keylist keylist;   /* for all keys structure   */
struct fa_keydesc keydesc1;  /* for prime record key     */
struct fa_keydesc keydesc2;  /* for alternate record key */

keylist.kl_nkeys = 2;          /* number of keys: 2 (prim & alt) */
keylist.kl_key[0] = &keydesc1; /* prime key address              */
keylist.kl_key[1] = &keydesc2; /* alternate key address          */
keydesc1.k_flags  = FA_NODUPS;

keydesc1.k_nparts = 2;            /* number of key parts: 2 */
keydesc1.k_part[0].kp_start = 0;  /*  1 - 1 ==  0 */
keydesc1.k_part[0].kp_leng = 4;
keydesc1.k_part[0].kp_flags = FA_ANYKPCODE;
keydesc1.k_part[1].kp_start = 6;  /* 7 - 1 == 6 */
keydesc1.k_part[1].kp_leng = 2;
keydesc1.k_part[1].kp_flags = FA_ANYKPCODE;

keydesc2.k_flags = FA_DUPS;
keydesc2.k_nparts = 1;             /* number of key parts: 1 */
keydesc2.k_part[0].kp_start = 11;  /* 12 - 1 == 11 */
keydesc2.k_part[0].kp_leng = 3;
keydesc2.k_part[0].kp_flags = FA_ANYKPCODE;
```

3.15.3 struct fa_dictinfo

cobfa_indexinfo()関数は、オープンしてある索引ファイルの情報をstruct fa_dictinfo型で返却することができます。
ここでは、索引ファイルの情報を取得するstruct fa_dictinfo型のメンバに格納される値について説明します。

```
------------------------------------------------------------------------
struct fa_dictinfo {
    long di_nkeys;    /* number of keys defined          */
    long di_recsize;  /* max or fixed data record size   */
    long di_idxsize;  /* size of indexes                 */
    long di_flags;    /* other flags (fixed or variable) */
};
------------------------------------------------------------------------
```

di_nkeysには、索引ファイル中のレコードキーの総数が設定されます。
ファイルのレコード長を示すdi_recsizeには、固定レコード長または最大レコード長が設定されます。なお、当アクセスルーチンでは、最小レコード長を取得することはできません。
すべてのレコードキーの長さの合計を示すdi_idxsizeには、レコードキーの総長が設定されます。
ファイルの属性を示すdi_flagsは、以下の1つのカテゴリの情報を持ちます。

レコード形式

記号定数	説明
FA_FIXLEN	レコード形式が固定長であることを示す。
FA_VARLEN	レコード形式が可変長であることを示す。

指定した索引ファイルが上記のどの属性に該当するかは、di_flagsの値と上記のどちらかの値との論理積を求めることによって知ることができます。

Cソースプログラム例:

```
------------------------------------------------------------------------
#include "f3bifcfa.h"
struct fa_dictinfo di;
long fd, ret;
  :
ret = cobfa_indexinfo(fd, &di, 0);  /* to get indexed file info */
if (di.di_flags & FA_FIXLEN) {
  /* process for fixed length record type */
    :
}
------------------------------------------------------------------------
```

第4章　エラー番号と入出力状態

ここでは、エラー番号と入出力状態について説明します。

4.1 エラー番号

エラー番号はAPI関数のエラーの種別を返却します。これには入出力状態だけでは表現できないCOBOLの翻訳エラーに相当する情報も含まれます。詳細については、“3.10.1 cobfa_errno()”を参照してください。
以下にエラー番号とその意味について説明します。

エラー番号	説明
FA_ENOERR	入出力機能の実行またはファイル情報取得の実行が成功したことを意味します。
FA_ENOSPC	ディスク容量が不足しています。
FA_EDUPL	キーの重複に関するエラーです。以下のどちらかの状態です。 ● レコードキーに指定した値を持つレコードが、すでにファイルに存在します。レコードキーは重複を許していません。 ● 相対レコード番号が既存のものと重複します。
FA_ENOTOPEN	ファイルのオープンに関するエラーです。以下のどちらかの状態です。 ● まだオープンしていないファイルです。 ● ファイルは、この機能を実行することができないオープンモード でオープンしています。
FA_EBADARG	引数に関するエラーです。以下のいずれかの状態です。 ● レコード域引数がNULLポインタです。 ● ファイル情報の取得関数で、機能番号が範囲外です。 ● ファイル情報の取得関数で、構造体ポインタがNULLです。
FA_EBADKEY	索引ファイルのレコードキー指定に関するエラーです。以下のいずれかの状態です。 ● 与えたレコードキー構成リストに矛盾があります。 ● 与えたレコードキー構成はファイルのキー構成と一致するものがありません。 ● 与えたレコードキー番号はファイルが持つキーの数を超えています。
FA_ETOOMANY	OSまたは当アクセスルーチンの制限値を超える数のファイルのオープンを行おうとしました。
FA_EBADFILE	ファイルの内部構成に関するエラーです。以下のいずれかの状態です。 ● ファイルの内部情報が正しくないか、破壊されています。 ● 正しいファイル編成を指定していません。 ● 動作コード系の指定と、行順ファイルのエンコード形式(シフトJIS、UCS-2、UTF-8) が一致していません。
FA_ELOCKED	レコードはすでにロックされています。
FA_EENDFILE	ファイル終了条件が発生しました。
FA_ENOREC	指定したレコードは存在しません。
FA_ENOCURR	レコードへの位置付けが不定です。
FA_EFLOCKED	ファイルはすでに排他オープンされています。
FA_EFNAME	オープン時に与えたファイル名に関するエラーです。以下のいずれかの状態です。 ● ファイルが存在しません。 ● ファイルにはアクセスすることができません。 ● ファイル名がNULLポインタまたは空文字列です。 ● ファイル名の構成が正しくありません。
FA_EBADMEM	機能を実行するために必要なメモリの獲得が失敗しました。
FA_EKEYSEQ	キーの順序誤りまたは変更誤りです。以下のどちらかの状態です。 ● 順書出しで、主レコードキー値が昇順でありません。 ● 順書換えで、主レコードキー値を変更しようとしました。

エラー番号	説明
FA_EBADACC	実行不可能な組合わせが発生しました。以下のいずれかの状態です。 ● 呼出し法に違反する機能の実行を要求しました。 ● このファイル編成では実行できない機能です。 ● オープン時のフラグの組合せが正しくありません。
FA_EBADFLAG	フラグの指定値が正しくありません。以下のどちらかの状態です。 ● オープンモード、読込みモード、位置付けモードに使用できないモードを指定しています。 ● その他、受け入れることができない値をフラグに指定しています。
FA_EBADLENG	長さに関するエラーです。以下のどちらかの状態です。 ● レコード長がファイルの定量制限を超えています。“留意事項”の“ファイル機能全般”を参照してください。 ● 位置付け時の有効キー長がファイルが持つキーパートの長さの合計を超えています。
FA_EOTHER	上記以外のエラーが発生しました。この場合、cobfa_stat()関数の復帰値を取得して状況を判断してください。cobfa_stat()関数については“cobfa_stat()”を参照してください。

4.2 入出力状態

当アクセスルーチンでは、入出力状態を取得するとき、cobfa_stat()関数を呼び出します。
cobfa_stat()関数については“3.11.1 cobfa_stat()”を参照してください。
入出力状態の種類とその詳細については、“NetCOBOL 使用手引書”の“付録B 入出力状態一覧”を参照してください。

第5章 サンプルプログラム

ここでは、提供する例題プログラムについて説明します。

5.1 行順ファイルの読込み

概要

このサンプルプログラムは、指定したファイルを行順ファイルとしてINPUTモードでオープンし、読み込んだレコードの内容を表示します。

提供プログラム

- fcfa01.c（Cソースプログラム）
- fcfa01.mak（MAKEファイル）
- fcfa01.txt（プログラム説明書）

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_rdnext()関数
- cobfa_stat()関数
- cobfa_errno()関数
- cobfa_reclen()関数
- cobfa_close()関数

プログラムの翻訳とリンク

MAKEファイルの“CDIR =”と書かれている行の右側の内容が、Cコンパイラをインストールしたフォルダ名となるように修正してください。また“COBDIR =”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたフォルダ名となるように修正してください。
MAKEファイルを修正した後、以下のコマンドを入力します。

```
> nmake -f fcfa01.mak
```

プログラムの実行

適当なテキストファイルをコマンドライン引数にしてプログラムを実行します。ここではfcfa01自身のソースプログラムを入力します。

```
> fcfa01 fcfa01.c
```

格納フォルダ

- *C:¥COBOL*¥SAMPLES¥FCFA01
  （*C:¥COBOL*は、NetCOBOLをインストールしたフォルダです。）

5.2 行順ファイルの読込みと索引ファイルの書出し

概要

このサンプルプログラムは、特定の行順ファイル(fcfa02.inp)をINPUTモードでオープンし、そのレコードの内容を索引ファイル(fcfa02.idx)のレコードとして書き出します。
最後に、その索引ファイルをINPUTモードでオープンし、主キーの順で画面に表示します。

提供プログラム

- fcfa02.c (Cソースプログラム)
- fcfa02.inp (入力用行順ファイル)
- fcfa02.mak (MAKEファイル)
- fcfa02.txt (プログラム説明書)

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_rdnext()関数
- cobfa_wrkey()関数
- cobfa_stkey()関数
- cobfa_close()関数

プログラムの翻訳とリンク

MAKEファイルの“CDIR =”と書かれている行の右側の内容が、Cコンパイラをインストールしたフォルダ名となるように修正してください。また“COBDIR =”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたフォルダ名となるように修正してください。
MAKEファイルを修正した後、以下のコマンドを入力します。

```
> nmake -f fcfa02.mak
```

プログラムの実行

コマンドライン引数を付けずに実行します。

```
> fcfa02
```

格納フォルダ

- *C:¥COBOL*¥SAMPLES¥FCFA02
  (*C:¥COBOL*は、NetCOBOLをインストールしたフォルダです。)

5.3 索引ファイルの情報の取得

概要

このサンプルプログラムは、指定したファイルを索引ファイルとしてINPUTモードでオープンし、ファイル自体の属性と、レコードキーの各構成を表示します。

提供プログラム

- fcfa03.c（Cソースプログラム）
- fcfa03.mak（MAKEファイル）
- fcfa03.txt（プログラム説明書）

使用しているCOBOLファイルアクセスルーチンの関数

- cobfa_open()関数
- cobfa_indexinfo()関数
- cobfa_stat()関数
- cobfa_errno()関数
- cobfa_close()関数

プログラムの翻訳とリンク

MAKEファイルの“CDIR =”と書かれている行の右側の内容が、Cコンパイラをインストールしたフォルダ名となるように修正してください。また“COBDIR =”と書かれている行の右側の内容が、COBOLコンパイラをインストールしたフォルダ名となるように修正してください。
MAKEファイルを修正した後、以下のコマンドを入力します。

```
> nmake -f fcfa03.mak
```

プログラムの実行

適当な索引ファイルをコマンドライン引数にしてプログラムを実行します。ここではfcfa02を実行して生成した索引ファイルを指定します。

```
> fcfa03 ..¥fcfa02¥fcfa02.idx
```

格納フォルダ

- *C:¥COBOL*¥SAMPLES¥FCFA03
  （*C:¥COBOL*は、NetCOBOLをインストールしたフォルダです。）

第6章 注意事項

ここでは、COBOLファイルアクセスルーチンの制限事項と留意事項について説明します。

6.1 制限事項

索引ファイル

- 動的呼出し法(FA_DYNACC)でオープンしたファイルが、以下のどちらかの条件のとき、順読込みで位置付けられているレコードに対して、順書換え/順削除を実行する方法はありません。
  - cobfa_stkey()関数で、逆順読込み(FA_REVORD)を指定した場合。
  - 主レコードキーの属性が重複可能(FA_DUPS)である場合。

6.2 留意事項

ファイル機能全般

- アプリケーションプログラムでオープンしたファイルは、必ず、すべてをクローズしてから処理を終了してください。この操作を行わないとファイルの内容の破壊、システムリソースのリークなどの問題が発生します。
- 同一プロセス内で、同一ファイルをオープンしても二重オープンエラー(入出力状態:41)は発生しません。この場合、別のファイルディスクリプタが割り当てられます。
- API関数の実行時にエラーが複数重なる場合には、COBOLアプリケーションとは異なった入出力状態を返却する場合があります。
- COBOLでは同じ入出力文を使っても異なる呼出し法を扱えますが、APIによる入出力では、呼出し法によって使用する関数が異なります。適切な関数を使用してください。
- 入出力の状況(cobfa_errno()、cobfa_stat()、cobfa_reclen()、cobfa_recnum()の各関数の復帰値)はファイルごとに情報を保持しません。これらの値はプロセス単位で保持するので、実行した入出力機能の状況値が必要な場合は、次の入出力機能を実行する前までに値を変数などに保存してください。以前の入出力の状況の値は、次の入出力機能の実行によって上書きされます。
- レコードの読込み/書出し/書換え/位置付けを行う場合、データ受渡し用のレコード域は、ファイルをオープンするときの最大/固定レコード長以上の大きさを持つ領域をあらかじめ確保しておく必要があります。
- COBOLでは翻訳時にエラーが検出できる場合でも、API関数では実行時にしかエラーを検出できないので注意してください。具体的には、呼出し法の不一致や正しくない索引キー指定などがこれに該当します。
- ファイルに関する定量制限は、“NetCOBOL 使用手引書”の“7.1.1 ファイルの種類と特徴”および“COBOL文法書”の“付録B.4 順ファイル”、“付録B.5 相対ファイル”、“付録B.6 索引ファイル”を参照してください。
- COBOLのデータ型とC言語のデータ型の対応については、“NetCOBOL 使用手引書”の“10.3.3 データ型の対応”を参照してください。また、システム2進と規格2進の扱いに注意してください。

行順ファイル

- 0バイトの長さのレコードを書き出すことはできません。

レコード順ファイル

- 印刷ファイルを扱えません。したがって、以下の仕様となります。
  - COBOL構文のLINAGE句に相当するものはありません。
  - COBOL構文のWRITE文の改行制御/ページ制御に相当するものはありません。

相対ファイル

- 以下の関数では、相対レコード番号を引数で明に指定する必要があります。
  - cobfa_delrec()関数（乱削除）
  - cobfa_rewrec()関数（乱書換え）
  - cobfa_rdrec()関数（乱読込み）
  - cobfa_strec()関数（位置決め）
- 以下の関数でアクセスしたレコードの相対レコード番号は、cobfa_recnum()関数で取得することができます。
  - cobfa_rdnext()関数（順読込み）
  - cobfa_wrnext()関数（順書出し）

索引ファイル

- 既存ファイルのオープン時に、レコードキー構成を指定しないでオープンすることができます。この場合、ファイルアクセスルーチンは既存ファイル内部のレコードキー構成を調

査してオープンします。

- オープン中の索引ファイルの属性(レコード長、レコード形式)とレコードキー構成を調べる機能(cobfa_indexinfo()関数)があります。

マルチスレッド

- マルチスレッド機能は、サーバ向け運用環境製品固有の機能です。
- COBOLファイルアクセスの排他制御(排他ロック/排他ロック解除)には、必ず以下の関数を使ってください。詳細については、各API関数の説明を参照してください。
  - 3.14.1 LOCK_cobfa()
  - 3.14.2 UNLOCK_cobfa()
- マルチスレッド環境下では、COBOLファイルをアクセスする際に必ず排他制御を行うようにしてください。排他制御を行わないと、競合によって以下のような問題が発生する可能性があります。
  - ファイル入出力の不正な実行結果
  - スレッドの異常終了
  - ファイル内容の破壊

Unicode

V60L10以降では、Unicodeを扱うために以下の追加/変更があります。

- struct fa_keypart型の構造体にメンバkp_flagsを追加しました。struct fa_keypart型については、“3.15.1 struct fa_keydesc”を参照してください。
- ファイルのオープン(cobfa_open()関数)のオープン属性引数が次の2つのカテゴリを認識するようになりました。
  - 動作コード系
  - キーパートフラグ使用指定
- ファイルのオープンのオープン属性引数に与える索引ファイル(FA_IDXFILE)の定義値を変更しました。

Unicodeを扱う場合の留意事項を以下にあげます。

- Unicodeを扱うアプリケーションは、Windows NT(R) 4.0、Windows(R) 2000、Windows(R) XPおよび、Windows Server(TM) 2003で使用することができます。
- 動作モードがUnicodeのCOBOLアプリケーションで扱う索引ファイルのキーパート長kp_lengは、バイト数を設定してください。
- 動作モードがUnicodeのCOBOLアプリケーションで扱う行順ファイルのレコード長には、バイト数を指定してください。

その他

- V50L10以前のファイルアクセスルーチンを使って作成したアプリケーションやDLLなどは、正しく動作します。しかし、V60L10でヘッダファイルの内容を一部変更しているため、V50L10以前のファイルアクセスルーチン用に作成したものを修正する場合は、必ず関連するソースプログラムのすべてを翻訳しなおすようにしてください。

付録A リファレンス

A.1 API関数

当アクセスルーチンは、以下の4つのカテゴリに分かれ、合計24個のAPI関数を持ちます。

a. 入出力

関数名	機能の概要	補足(*1)
3.2.1 cobfa_close()	ファイルをクローズする	
3.5.1 cobfa_delcurr()	順読込みしたレコードを削除する	S
3.5.2 cobfa_delkey()	主レコードキーの値によって示されるレコードを削除する	RD
3.5.3 cobfa_delrec()	相対レコード番号によって示されるレコードを削除する	RD
3.1.1 cobfa_open()	ファイルをオープンする	
3.8.1 cobfa_release()	指定したファイルのレコードロックをすべて解除する	
3.6.1 cobfa_rewcurr()	順読込みしたレコードを書き換える	SD
3.6.2 cobfa_rewkey()	主レコードキーの値によって示されるレコードを書き換える	RD
3.6.3 cobfa_rewrec()	相対レコード番号によって示されるレコードを書き換える	RD
3.3.1 cobfa_rdkey()	任意のレコードキーの値によって示されるレコードを読み込む	RD
3.3.2 cobfa_rdnext()	レコードを順に読み込む	S
3.3.3 cobfa_rdrec()	相対レコード番号によって示されるレコードを読み込む	RD
3.7.1 cobfa_stkey()	任意のレコードキーの値によって示されるレコードに位置付ける	SD
3.7.2 cobfa_strec()	相対レコード番号によって示されるレコードに位置付ける	SD
3.4.1 cobfa_wrkey()	指定した主レコードキーの値を持つレコードを書き出す	RD
3.4.2 cobfa_wrnext()	レコードを順に書き出す	S
3.4.3 cobfa_wrrec()	指定した相対レコード番号を持つレコードを書き出す	RD

*1：関数の使用範囲を記号で示しています。各記号の意味は以下のとおりです。

記号	説明
RD	オープンモードが乱呼出し、または動的呼出しで使用できる関数
SD	オープンモードが順呼出し、または動的呼出しで使用できる関数

記号	説明
S	オープンモードが順呼出しのときにだけ使用できる関数

b. ファイルの情報を取得

関数名	機能の概要
3.9.1 cobfa_indexinfo()	索引ファイルが持つファイルの属性、またはレコードキーの構成を取得する

c. 入出力の状況を取得

関数名	機能の概要
3.10.1 cobfa_errno()	エラー番号を取得する
3.12.1 cobfa_reclen()	読み込んだレコードの長さを取得する
3.13.1 cobfa_recnum()	位置付けられたレコードの相対レコード番号を取得する
3.11.1 cobfa_stat()	入出力状態を取得する

d. マルチスレッド環境下での排他制御

関数名	機能の概要
3.14.1 LOCK_cobfa()	他のスレッドのCOBOLファイルへのアクセスに対して排他ロックをかける
3.14.2 UNLOCK_cobfa()	他のスレッドのCOBOLファイルへのアクセスに対して排他ロックを解除する

A.2 API関数で使用する構造体

いくつかのAPI関数は、構造体型へのポインタ型を用います。これらの構造体は合計3つで以下のものがあります。

構造体名	機能の概要
3.15.3 struct fa_dictinfo	索引ファイルの属性取得のための構造体
3.15.1 struct fa_keydesc	任意のレコードキーの指定のための構造体。指定した任意のレコードキーの構造を取得するためにも使用する
3.15.2 struct fa_keylist	すべてのレコードキーを指定するための構造体

索引