
# Net Express

# ファイル処理

Micro Focus NetExpress™

# ファイル処理

Micro Focus®

第 2 版
1998 年 10 月

# 序文

このプログラミング ガイドでは、NetExpress が持つファイル処理機能について説明します。

## 対象読者

このガイドは、NetExpress でファイルにアクセスする COBOL アプリケーションを作成する方を対象にしています。

## このガイドの使用方法

「第 1 章 はじめに」では、NetExpress で使用可能なファイル処理機能と、Micro Focus ファイル ハンドラの概要について説明します。

「第 2 章 ファイル編成」では、さまざまなファイル編成、COBOL ファイル編成の構造について説明します。

「第 3 章 ファイル名」では、使用するファイル命名規則、物理ファイルに対する論理ファイル名の割り当て方法、ファイル名マッピングの概念について説明します。

「第 4 章 ファイル状態」では、ファイル状態コードとその使用方法について説明します。

「第 5 章 ファイルの共有」では、NetExpress で使用可能なファイルとレコードのロック機能について説明します。

「第 6 章 ファイル ハンドラの構成」では、構成可能な COBOL ファイル処理機能について詳細に説明します。

「第 7 章 ファイル ハンドラ API」では、ファイル ハンドラ API と、これを使用してファイル ハンドラを直接呼び出す方法について説明します。

「第 8 章 Btrieve」では、Btrieve ファイルへアクセスするための NetExpress の機能について説明します。

「第 9 章 ファイルのソート」では、mfsort コマンド行ユーティリティと呼び出し可能なソート モジュールについて説明します。

「第 10 章 REBUILD」では、破損した索引ファイルを修理するための REBUILD ユーティリティについて説明します。

「第 11 章 GUI データツール」では、PC でアプリケーションのデータ ファイルを作成し、維持するための NetExpress の機能について説明します。

## 表記規則

このプログラミング ガイドでは、次の書体や規則を使用しています。

- 入力するテキストは、次のように表示されます。

  ```
  cat script_name | more
  ```

斜体テキストは、コマンドの一部として入力する変数を表します。

- コマンド行やコード例でオプション入力するテキストは、角かっこ ([]) で囲みます。次の式では、オプションで入力する単語 `NOT` がない場合、*`column_name`* は `pattern_value` のように設定され、`NOT` がある場合、*`column_name`* は *`pattern_value`* 以外であるように設定されます。。

  *`column_name`* `[NOT] LIKE` *`pattern_value`*

- 特定のモデルやオペレーティング システムだけに適用する項や段落については、段落のすぐ前に太字斜体の項目名が記載されています。例えば、次のようになります。

*UNIX*
この段落は、UNIX システムだけに適用されます。

# 目次

# 第1章 はじめに

## 1.1 概要

プログラミング言語として COBOL が持つ主要な機能の 1 つに、組み込みのファイル処理機能があります。順ファイル、相対ファイル、索引ファイルなどのファイル編成は、簡単な COBOL 構文で処理することができます。さらに、Micro Focus COBOL では、COBOL のファイル処理機能を次のように拡張しています。

- ファイル編成に追加された「行順ファイル」を使用すると、COBOL プログラムから ASCII テキスト ファイルへアクセスすることができます。
- COBOL システム ライブラリ ルーチンを使用すると、ファイルとディレクトリを操作することができます。
- COBOL システム ライブラリ ルーチンを使用すると、バイト レベルでファイルを処理することができます。
- 変換モジュールを使用すると、Btrieve ファイルに対する読み書きが可能になります。
- Fileshare を使用すると、プログラムからネットワーク サーバーのファイル グループにアクセスすることができます。

## 1.2 Micro Focus ファイル ハンドラ

Micro Focus ファイル ハンドラのデフォルト設定では、すべての標準 COBOL ファイル編成 (順ファイル、相対ファイル、および索引ファイル) に対して、すべての COBOL 入出力操作を実行することができます。さらに、ファイル ハンドラ API では、COBOL プログラムから直接、ファイル ハンドラを呼び出すことができます。

## 1.3 ファイル処理 マニュアル

### 1.3.1 オンライン マニュアル

NetExpress には、多くの HTML マニュアルがあり、Web ブラウザを使用して表示することができます。

- *ファイル処理* (このマニュアル)

  主要なファイル処理機能とその処理方法の詳細を、使用例と合わせて説明します。記載されている機能は、次のとおりです。

  - ファイル編成

    さまざまなファイル編成と、COBOL ファイル編成の構造について説明します。

- ファイル名

  使用するファイル命名規則、物理ファイルに対する論理ファイル名の割り当て方法、ファイル名マッピングの概念を説明します。

- ファイル状態

  ファイル状態コードとその使用方法について説明します。

- ファイルの共有

  NetExpress で使用可能なファイルとレコードのロック機能について説明します。

- 構成

  構成可能な COBOL ファイル処理機能について詳細に説明します。

- ファイル ハンドラ API

  ファイル ハンドラ API と、これを使用してファイル ハンドラを直接呼び出す方法について説明します。

- Btrieve

  Btrieve ファイルへアクセスするための NetExpress の機能について説明します。

- ファイルのソート

  mfsort コマンド行ユーティリティと呼び出し可能なソート モジュールについて説明します。

- REBUILD

  破損した索引ファイルを修復するための REBUILD ユーティリティについて説明します。

- GUI データ ツール

  データ ファイル エディタ、データ ファイル変換、およびレコード レイアウト エディタについて説明します。ファイル索引の修正については、REBUILD の章を参照してください。

- *Fileshare ユーザー ガイド* (Fspubb02.htm)

  このユーザー ガイドは、NetExpress で Fileshare の機能を使用してアプリケーションを作成する COBOL プログラマを対象としています。

  - ネットワークを通じたデータ ファイルへの効率的な同時アクセス
  - トランザクション処理

    - ロールフォーワード回復

- *データベースへのアクセス* (Dbpubb02.htm)

  データがリレーショナル データベースに保存されている場合、NetExpress で埋め込み SQL を使用してデータへアクセスする COBOL アプリケーションを作成することができます。 リレーショナル データベースへのアクセス方法に関する詳細は、このデータベース アクセス マニュアルで説明されています。

### 1.3.2 オンライン ヘルプ

ファイル処理の特定の面に精通していて、簡単な参照や例が必要な場合、NetExpress のオンライン ヘルプ ファイルで次の項目を選択してください。 (ヘルプの [トピック] ボタンをクリックし、[目次] タブがアクティブになっていることを確認してから、[プログラミング]、[ファイル処理] を選択し、次の項目を参照してください。)

- 方法

  操作の実行方法について、簡単な手順を説明します。

- 例

  例（通常はコード）とその例で使用されるファイル処理技法を説明します。

または、 [目次] から [リファレンス] 、[ファイル処理] の順に選択し、リスト、テーブル、データ構造体、エラー メッセージ、コンパイラ指令、ランタイム スイッチなどの参照情報を表示してください。

# 第2章 ファイル編成

ファイルはデータの集合で、通常はディスクに保存されています。論理的なファイルでは、データを意味のあるグループに分割することができます。例えば、あるファイルに会社の製品情報をすべて保存し、別のファイルにすべての社員情報を保存することができます。物理的なファイルについては、ファイル編成の観点から考える必要があります。

## 2.1 ファイル編成

「ファイル編成」という用語は、ファイルにデータを格納する方法を意味します。また、データへのアクセス方法をも意味します。この COBOL システムでは、順ファイル、相対ファイル、索引ファイルの 3 つのファイル編成をサポートしています。

### 2.1.1 順ファイル

順ファイルでは、個々のレコードへ順番にアクセスすることになります。つまり、最初にファイルに書き込まれた順番と同じ順番でレコードにアクセスします。新しいレコードは常にファイルの末尾に追加されます。

この COBOL システムでは、3 種類の順ファイルがサポートされています。

- レコード順
- 行順
- プリンタ順

#### 2.1.1.1 レコード順ファイル

デフォルトでは、ファイルを作成し、ファイル編成を順ファイルに指定すると、レコード順ファイルが作成されるため、レコード順ファイルは、ほとんどの場合「順ファイル」と呼ばれます。

ファイルをレコード順ファイルとして定義するには、COBOL プログラムでファイルに対して SELECT 文と ORGANIZATION IS RECORD SEQUENTIAL を指定します。例えば、次のように指定します。

```
select recseq assign to "recseq.dat"
   organization is record sequential.
```

デフォルトでは、順ファイルを指定するとレコード順ファイルが設定されるため、必ずしも ORGANIZATION IS RECORD SEQUENTIAL を指定する必要はありません。単に ORGANIZATION IS SEQUENTIAL と指定するだけでかまいません (ただし、コンパイラ指令 SEQUENTIAL が設定されていない場合)。

#### 2.1.1.2 行順ファイル

行順ファイル (「テキスト ファイル」または「ASCII ファイル」とも呼ばれる) は、主に、表示専用データに使用

します。「メモ帳」などのほとんどの PC エディタでは行順ファイルが作成されます。

行順ファイルでは、ファイルの各レコードの間は、レコード区切り文字で区切られます。 レコード区切り文字は、キャリッジ リターン コード (x"0D") とライン フィード コード (x"0A") で構成され、各レコードの最後にある空白文字以外の文字の後に挿入されます。WRITE 文は、データ レコードから後続の空白文字を削除し、レコード区切り文字を追加します。 READ 文は、レコード区切り文字を削除し、必要に応じて、データ レコードを後続の空白文字で埋め、データを読み込むプログラムが定義するレコード サイズに合わせます。

ファイルを行順ファイルとして定義するには、COBOL プログラムでファイルに対して SELECT 文と ORGANIZATION IS LINE SEQUENTIAL を指定します。例えば、次のように指定します。

```
select lineseq assign to "lineseq.dat"
   organization is line sequential.
```

#### 2.1.1.3 プリンタ順ファイル

プリンタ順ファイルは、直接、または、いったんディスク ファイルにスプールして、プリンタに送信するファイルを指します。このファイルは、連続した印刷レコードから構成されており、レコード間に縦方向の位置を表す文字 (ライン フィードなど) が入る場合があります。 印刷レコードは、印刷可能な文字で構成され、キャリッジ リターン (x"0D") で終了します。ただし、印刷可能な文字がない場合もあります。

プリンタ順ファイルで OPEN 文を指定すると、ファイルに x"0D" を書き込み、最初の印刷レコードを印刷する前にプリンタを最初の文字位置に移動させます。WRITE 文は、印刷レコードと末尾のキャリッジ リターン コード (x"0D") をプリンタに書き込む前に、印刷レコードから後続空白を削除します。 WRITE 文で BEFORE 句または AFTER 句を指定すると、印刷レコードの書き込み前または後に、プリンタに対して 1 つ以上のライン フィード コード (x"0A")、改ページ コード (x"0C")、または縦方向のタブ コード (x"0B") を送信することができます。

プリンタ順ファイルを、入力ファイル (INPUT) または入出力ファイル (I/O) として開かないでください。

ファイルをプリンタ順ファイルとして定義するには、SELECT 文で ASSIGN TO LINE ADVANCING FILE または ASSIGN TO PRINTER と指定します。例えば、次のように指定します。

```
select printseq
   assign to line advancing file "printseq.dat".
```

### 2.1.2 相対ファイル

相対ファイルは、各レコードをファイル内の順位 (レコード 1、レコード 2 のように) によって識別するファイルです。つまり、レコードに対して、順番にアクセスすることも、ランダムにアクセスすることもできます。順アクセスでファイルの次のレコードにアクセスするには、READ 文または WRITE 文を実行するだけです。ランダム アクセスでは、相対キーとしてデータ項目を定義し、データ項目の中で、READ または WRITE の対象とするレコードの序数を指定する必要があります。

相対ファイルの場合、レコードへランダムにアクセスできるため、アクセスが高速化します。ただし、ディスク領域を節約する必要がある場合は、相対ファイルを使用しないでください。相対ファイルに対して可変長レコードを宣言することは可能ですが、ファイルに対して WRITE 文を指定すると、システムは最大レコード長までの書き込みを処理し、使用しない文字位置まで埋めてしまうためです。このような処理を行う理由は、COBOL ファイル処理ルーチンが、ファイルでレコード番号を指定されたレコードの物理的な位置をすばやく計算するためです。

相対ファイルには、常に固定長レコードが含まれるため、データ圧縮を指定しても領域を節約することはできません。Micro Focus ファイル ハンドラでは、相対ファイルにデータ圧縮を指定しても無効になります。

相対ファイルの各レコードの後には 2 バイトのレコード マーカーがあり、レコードの現在の状態を示します。レコード マーカーが示す状態は、次のとおりです。

x"0D0A" - レコードが存在する

x"0D00" - レコードが削除されたか、書き込まれていない

相対ファイルからレコードを削除すると、レコードのレコードマーカーが更新され、削除済みの記録が表示されます。ただし、削除されたレコードの内容は、物理的には新しいレコードが書き込まれるまでそのまま残ります。セキュリティの観点から実際のデータをファイルに残さないようにするには、削除する前に REWRITE を使用してレコードを上書きする必要があります (例えば、空白文字などを使用する)。

相対ファイルを定義するには、COBOL プログラムでファイルに対して SELECT 文と ORGANIZATION IS RELATIVE を指定します。

レコードにランダムにアクセスする場合は、次のように定義する必要があります。

- ファイルに対して SELECT 文と ACCESS MODE IS RANDOM または ACCESS MODE IS DYNAMIC を指定する。
- プログラムの作業場所節で相対キーを定義する。

例えば、次のように指定します。

```
select relfil assign to "relfil.dat"
   organization is relative
   access mode is random
   relative key is relfil-key.
...
working-storage section.
01 relfil-key pic 9(8) comp-x.
```

上記のコード例では、相対ファイルを定義しています。アクセス モードはランダムなので、相対キー (relfil-key) を

定義します。ランダム アクセスの場合、常に、ファイルからレコードを読み込もうとする前に、相対キーにレコード番号を指定することが必要です。

ACCESS MODE IS DYNAMIC を指定した場合、順アクセスとランダム アクセスの両方でファイルへアクセスすることができます。

### 2.1.3 索引ファイル

索引ファイルでは、各レコードが主キーを持っています。各レコードを互いに区別するために、主キーの値は各レコードに対して一意でなければなりません。この条件が満たされている場合、レコードの主キーの値を指定すると、レコードへランダムにアクセスできます。索引ファイルのレコードへも順アクセスすることが可能です。索引ファイルでは、主キーの他に、副キーと呼ばれる追加キーを指定することができます。レコードの副キーの値は一意である必要はありません。

ファイルを索引ファイルとして定義するには、COBOL プログラムでファイルに対して SELECT 文と ORGANIZATION IS INDEXED を指定します。また、RECORD KEY 句を使用して主キーも指定する必要があります。

```
select idxfile assign to "idx.dat"
   organization is indexed
   record key is idxfile-record-key.
```

実際には、ほとんどの索引ファイルは、データ ファイル (レコード データを含む) と索引ファイル (索引構造体を含む) という 2 つの別個のファイルで構成されています。その場合、COBOL プログラムで指定した名前はデータ ファイル名になります。関連付けられた索引ファイルの名前はデータ ファイル名に .idx という拡張子を追加して作成します。他のコンテキストでは拡張子 .idx を使用しないでください。

索引はレコードが追加されると大きくなる逆ツリー構造として構築されます。

索引ファイルでは、ランダムに選択したレコードを検索するためのディスク アクセス回数は、主にファイルのレコード数とレコード キーの長さによって決まります。ファイルの入出力は、ファイルを順に読み込んでいく場合よりも高速になります。

全種類のファイルを定期的にバックアップすることをお勧めしますが、索引ファイルでは 2 つのファイルのどちらかが使用できなくなる場合 (例えば、媒体の破損など) があります。このような状況で索引ファイルを損失した場合、Rebuild ユーティリティを使用して、データ ファイルから索引を回復し、障害復帰に必要な時間を短縮することが可能です。

#### 2.1.3.1 主キー

索引ファイルの主キーを定義するには、SELECT 文と RECORD KEY IS 句を使用します。

```
select idxfile assign to "idx.dat"
   organization is indexed
```

```
   record key is idxfile-record-key.
```

#### 2.1.3.2 副キー

各レコードには、主キーの他に、副キーと呼ばれる追加のキーを必要な数だけ指定できます。副キーは、SELECT 文と ALTERNATE RECORD KEY IS 句を使用して定義します。

```
select idxfile assign to "idx.dat"
   organization is indexed
   record key is idxfile-record-key
   alternate record key is idxfile-alt-key.
```

#### 2.1.3.3 重複キー

重複した値を持つキーを定義することができます。ただし、レコードの主キーの値は一意でなければならないため、主キーは重複させることができません。

重複キーを使用する場合、各キーに同じ値を指定できる回数が制限されることに注意してください。重複キーに同じ値を指定するたびに、キーの出現番号が 1 ずつ増えます。個々のキーで値を重複させられる最大回数は、索引ファイルの種類によって異なります (索引ファイルの種類とその特徴の詳細リストについては、オンライン ヘルプの目次で [*索引ファイル*]、[*種類*] の順に選択し、参照してください)。出現番号は、重複キー レコードを作成順に読み込むために使用します。そのため、削除したレコードの出現番号を再使用することはできません。つまり、一部のキーをすでに削除した場合でも、重複値の最大出現回数に到達する可能性があることを意味します。

索引ファイルの種類によっては、データ ファイルに重複レコードを持つものがあります (索引ファイルの種類とその特徴の詳細なリストについては、オンライン ヘルプ ファイルで [*索引ファイル*]、[*種類*] の順に選択し、参照してください)。このようなファイルでは、データ ファイルの各レコードの後に、そのレコードの重複キーに対するキーの出現番号を示すシステム レコードが続きます。この番号は、ファイルの履歴で、あるキー値が使用された回数を示すカウンタに過ぎません。重複レコードが存在すると、レコードに対する REWRITE 操作と DELETE 操作がより高速になりますが、このようなファイルのデータ レコードは標準的なファイルのデータ レコードよりも大きくなります。

副キーに重複した値を指定するには、SELECT 文の ALTERNATE RECORD KEY 句で WITH DUPLICATES を使用します。

```
file-control.
select idxfile assign to "idx.dat"
   organization is indexed
   record key is idxfile-record-key
   alternate record key is idxfile-alt-key with duplicates.
```

### 2.1.3.4 スパース (疎) キー

スパース キーを設定すると、指定した値に対する索引エントリがファイルに格納されません。例えば、すべて空白文字であるキーをスパース キーとして定義した場合、空白文字しかないデータ項目を持つレコードが、そのキーの索引エントリとしてファイルに格納されることはありません。

スパース キーとして指定できるのは、副キーだけです。

この機能を使用すると、索引ファイルを小さくできます。キーが大きくなり、副キーに指定した値を持つレコードの数が増えた場合に、多くのディスク容量を節約できます。

スパース キーを有効化するには、SELECT 文の ALTERNATE RECORD KEY 句で SUPPRESS WHEN ALL を使用します。

```
file-control.
select idxfile assign to "idx.dat"
   organization is indexed
   record key is idxfile-record-key
   alternate record key is idxfile-alt-key with duplicates suppress when all "A".
```

この例では、副キーの値がすべて A であるレコードが書き込まれた場合、実際のキーの値は索引ファイルに格納されません。

### 2.1.3.5 索引ファイルへのアクセス

主キーと副キーの両方を使用して、直接 (ランダム アクセス)、またはキーの順番にしたがって (順アクセス)、索引ファイルからレコードを読み込むことができます。アクセス モードは、次のとおりです。

- SEQUENTIAL

  デフォルトでは、SEQUENTIAL が設定されています。レコードへは、レコード キーの値の昇順 (または 降順) でアクセスします。

- RANDOM

  レコード キーの値でアクセスするレコードを指定します。

- DYNAMIC

  適切な形式の入出力文を使用することにより、プログラムで順アクセスとランダム アクセスを切り替えることができます。

索引ファイルへのアクセス方法は、SELECT 文で ACCESS MODE IS 句を使用して定義します。次に例を示します。

```
file-control.
```

```
select idxfile assign to "idx.dat"
   organization is indexed
   access mode is dynamic
   record key is idxfile-record-key
   alternate record key is idxfile-alt-key.
```

## 2.2 固定長レコードと可変長レコード

ファイルには、固定長レコード (すべてのレコードが全く同じ長さである場合) か、可変長レコード (各レコードの長さが異なる場合) を使用します。 可変長レコードを使用すると、ディスク領域を節約することができます。例えば、アプリケーションが生成するレコードのうち、短いレコードが多く、長いレコードが少ない場合、固定長レコードを使用すると、固定長を最長のレコード長に合わせる必要があります。固定長レコードは、このように大量のディスク領域を無駄にするため、可変長レコードを使用した方が大きな利点があります。

レコードの種類は、次のように指定します。

- RECORDING MODE IS V 句を指定すると、ファイルのレコードは可変長レコードになります。RECORDING MODE IS F 句を指定すると、ファイルのレコードは固定長レコードになります。

上記以外の方法

- RECORD IS VARYING 句を指定すると、ファイルのレコードは可変長レコードになります。RECORD CONTAINS n CHARACTERS 句を指定すると、ファイルのレコードは固定長レコードになります。

上記以外の方法

- RECMODE"V" コンパイラ指令を指定すると、ファイルのレコードは可変長レコードになります。RECMODE"F" を指定すると、ファイルのレコードは固定長レコードになります。

上記以外の方法

- RECMODE"OSVS" コンパイラ指令と、次のどちらかを指定すると、ファイルのレコードは可変長レコードになります。
    - RECORD CONTAINS n TO m CHARACTERS を指定した場合
    - 複数のレコード領域が指定され、それぞれの長さが異なる場合

## 2.3 ファイル ヘッダー

ファイル ヘッダーは、ファイルの先頭にある 128 バイトのブロックです。索引ファイル、可変長レコードをもつレコード順ファイル、可変長レコードをもつ相対ファイルは、すべてファイル ヘッダーを持っています。さらに、これらのファイルでは、各レコードの先頭に 2 バイトまたは 4 バイトのレコード ヘッダーが付きます。

ファイル ヘッダー、レコード ヘッダー、およびヘッダーをもつファイルの構造の詳細については、オンライン ヘルプ ファイルを参照してください。ヘルプ ファイルの索引で [*構造*]、[*ヘッダー付きファイル*] を参照してください。

# 第3章 ファイル名

この章では、次の事項について説明します。

- ファイル命名規則
- ファイル名の割り当て
- ファイル名マッピング
- 動的に割り当てられたファイル、または、外部で割り当てられたファイルを検索するためにランタイム システムが採用する検索順序

## 3.1 ファイル命名規則

この COBOL システムと、この COBOL システムを使って作成したアプリケーションでは、標準的な DOS のファイル命名規則を採用しています。

また、New Technology File System (NTFS) のファイル命名規則もサポートしています。NTFS では、ファイル名やディレクトリ名に最大 254 文字まで使用できます。この中には、空白文字や必要な数だけのピリオド (.) を含めることができます。 この COBOL システムでは、最後のピリオドの後のテキストを拡張子と見なし (または最後がピリオドの場合、空白文字を拡張子と見なす)、COBOL システムが名前を作成するときにはこれを削除することもあります。

## 3.2 ファイル名の割り当て

物理的なファイル名、または、実行時に物理的な名前にマップできる論理的なファイル名を指定するには、SELECT 文の ASSIGN 句を使用します。

ファイル名の割り当てには、次のような 3 種類の方法があります。

- STATIC

  SELECT 文の ASSIGN 句でファイル名をリテラルとして指定します。

- DYNAMIC

  SELECT 文の ASSIGN 句でファイル名をデータ項目として指定するので、実行時にプログラムで変更することができます。

- EXTERNAL

  SELECT 文の ASSIGN 句でファイル名を EXTERNAL として指定し、実行時に決定します。

これら 3 種類すべてのファイル名の割り当て方法でファイル名の実行時マッピングを使用することができます。

### 3.2.1 ファイル名の静的割り当て

ファイル名の静的割り当てでは、ファイル名を SELECT 文でリテラルとして指定します。

```
select filename
   assign to literal.
```

次の例では、stockfile を開くと、b:ドライブの現在のディレクトリにある warehs.buy ファイルが開きます。

```
select stockfile
   assign to "b:warehs.buy".
```

次の例では、WRITE 文を使用すると、最初のパラレル プリンタである prn: にデータが出力されます。

```
select printfile
   assign to "prn:".
```

次の例では、input-file を開くと、現在のディレクトリに相対的な data ディレクトリの prog ファイルが開きます。

```
select input-file
   assign to "data¥prog".
```

### 3.2.2 ファイル名の動的割り当て

ファイル名の動的割り当てでは、ファイル名を SELECT 文で COBOL データ項目として指定します。

```
select filename
   assign to dynamic data-item
```

*data-item* COBOL データ項目の名前です。プログラムでデータ項目が明示的に宣言されていない場合、ピクチャ句 PIC X(255) を使用して、コンパイラがデータ項目を自動的に作成します。ファイルに対して OPEN 文を実行する前に、プログラムでデータ項目の値を指定する必要があります。

ASSIGN"DYNAMIC" コンパイラ指令を使用する場合には、ASSIGN 句から DYNAMIC という単語を省略することができます。

次の例では、input.dat というファイルが現在のディレクトリに作成されます。

```
...
select fd-in-name
   assign to dynamic ws-in-file.
...
```

```
working-storage section.
01 ws-in-file    pic x(30).
...
move "input.dat" to ws-in-file.
...
open output fd-in-name.
```

### 3.2.3 ファイル名の外部割り当て

ファイル名の外部割り当てでは、ファイル名を SELECT 文で次のように指定します。

```
select filename
   assign to external external-file-reference
```

*external-file-reference* 外部環境に指定されたファイルでさらにマッピングが可能かどうかを識別する COBOL 語です。 *external-file-reference* に 1 つまたは複数のハイフン (-) が含まれる場合、最後のハイフンまでのすべての文字が無視されます (最後のハイフンも無視されます)。

ランタイム ファイル名マッピングの詳細については、「*ファイル名マッピング*」の項を参照してください。

## 3.3 ファイル名マッピング

この COBOL システムには、プログラムで ASSIGN 句を使用して指定したファイル名を別の名前にマップする方法が複数あるので、実行時に柔軟に対応できます。

次の説明で使用する「環境変数」には、外部ファイル マッパーを通して有効化された Micro Focus の拡張環境変数も含まれます (この章の「*外部ファイル マッパー*」の項を参照してください)。

ファイル ハンドラにファイル名 (リテラル、データ項目の内容、または、ASSIGN TO EXTERNAL 構文を使用した場合は外部リファレンスとしてのファイル名) が指定されている場合、その名前の最初の要素を分離します。つまり、バック スラッシュ文字 (¥) がある場合、最初のバック スラッシュ文字より前にあるすべてのテキストを分離します。バック スラッシュ文字がファイル名に含まれない場合、テキスト全体を分離します。また、ファイル名がバック スラッシュ文字で始まる場合には分離させません。これらの処理の後で、次の処理を行います。

- 先頭にドル記号 ($) が存在する場合、これを削除し、最初の要素に "dd_" という文字を追加して、この名前をもつ環境変数を検索します。
- ASSIGN EXTERNAL 構文が使用されている場合、または、ファイル名の最初の要素がドル文字で始まる場合に、この環境変数が見つからないと、最初の要素 (ドル記号がある場合には先頭のドル記号は除く) と同じ名前をもつ環境変数を検索します。
- 元のファイル名がドル文字で始まり、少なくとも 1 個のバック スラッシュ文字を含む場合以外は、検索に

失敗すると元のファイル名は変更されません。この場合、最初の要素全体と最初のバック スラッシュが名前から削除されます。

この処理結果を物理ファイルのファイル名にします。

例

| ASSIGN 句のファイル名 | 検索対象の環境変数 | 環境変数の内容 | 物理ファイルのファイル名 |
|---|---|---|---|
| dir¥file1 | dd_dir | d2 | d2¥file1 |
| dir¥file1 | dd_dir | d2¥d4 | d2¥d4¥file1 |
| dir1¥dir2¥file1 | dd_dir1 | d4 | d4¥dir2¥file1 |
| dir1¥dir2¥file1 | dd_dir1 | d2¥d4 | d2¥d4¥dir2¥file1 |
| file1 | dd_file1 | d2 | d2 |

### 3.3.1 複数パス

ファイル名のマッピングに使用する環境変数は、複数のパス名を指します。そのため、環境変数が指定した最初のパスに対して「ファイルが見つからない」という状態が返されると、システムは次のファイルを検索します。

次のような dd_dir という名前の環境変数があると仮定します。

```
dd_dir=¥a¥b;¥c¥d;
```

この場合、¥a¥b に対して「ファイルが見つからない」という状態が返されると、システムは、割り当てられたファイルを ¥c¥d で検索します。

---

注記

- I/O、OUTPUT または EXTEND として開かれたファイルについて、パスのどの部分でもファイルが見つからない場合、このファイルは、検索順序の最初にあるディレクトリに作成されます。
- dd_path リストの各パスの長さは、パスの許容最大文字数以下でなければなりません。

---

## 3.4 オペレーティング システム デバイス名の割り当て

プリンタに直接レポートを送信するか、通信ポートを通してデータを転送するための COBOL プログラムを作成することができます。そのためには、COBOL ファイル名にデバイス名を割り当てる必要があります。

次に示すデバイス名は、ファイル名の静的割り当て、動的割り当て、または、外部割り当てを使用して指定することができます。

CON　　コンソール キーボードまたは画面

PRN　　最初のパラレル プリンタ

LPT1　　最初のパラレル プリンタ

LPT2　　2 番目のパラレル プリンタ

LPT3　　3 番目のパラレル プリンタ

COM1　　最初の非同期通信ポート

COM2　　2 番目の非同期通信ポート

これらのデバイス名を指定するときに、オプションで末尾にコロン (:) を指定することができます。

次の例では、 `fd-name` へ読み書き操作を行うと、コンソール画面でデータの読み書きが行われます。

```
select fd-name
   assign to "con".
```

この例では、`fd-name` へ書き込み操作を行うと、データが最初のパラレル プリンタである `lpt1:` に出力されます。

```
select fd-name
   assign to dynamic ws-filename.
   ...
   move "lpt1:" to ws-filename.
```

## 3.4.1 パイプのセットアップ

COBOL ファイル構文を使用して、別のプロセス (dir コマンドなど) を起動し、そのプロセスの標準入力にデータを書き込んだり、そのプロセスの標準出力からのデータを読み込んだりすることができます。この場合、COBOL ファイル編成は、LINE SEQUENTIAL または RECORD SEQUENTIAL のどちらかである必要があります。

### 3.4.1.1 出力パイプ

プロセスを起動し、データを標準入力に書き込むためには、ファイル名の > 記号の後にコマンド名を続ける必要があります。ファイルは、出力ファイルとして開く必要があります。

例

```
select output-file
   assign to "cmd /c print"
   organization is line sequential.
...
open output output-file
write output-file-record from "Hello world".
```

この例では、プログラムは文字 "Hello world" を印刷プロセスの標準入力に渡します。

#### 3.4.1.2 入力パイプ

プロセスを起動し、標準出力からデータを読み込むには、ファイル名の "lt" シンボルの後にコマンド名を続ける必要があります。ファイルは、入力ファイルとして開く必要があります。

例

```
select input-file
   assign to "<cmd /c dir"
   organization is line sequential.
...
open input input-file
read input-file
```

この例では、プログラムは dir プロセスを起動し、そのプロセスが標準出力に書き込む最初の行を読み込みます。

#### 3.4.1.3 双方向パイプ

双方向パイプは、入力パイプと出力パイプの機能を組み合わせたものです。双方向パイプを使用するには、ファイル名のパイプ記号 (|) の後にコマンド名を続ける必要があります。ファイルは、入出力ファイルとして開く必要があります。

例

```
select i-o-file
   assign to "| cmd /c sort"
   organization is line sequential.
...
   open i-o i-o-file
```

```
write i-o-file-record from "Hello world"
read i-o-file
```

この例では、プログラムは sort プロセスを起動し、"Hello world" という行を標準入力に渡します。次に、sort プロセスの標準出力から 1 レコードを読み込みます。

## 3.5 外部ファイル マッパー (Mfextmap)

外部ファイル マッパー (Mfextmap) を使用すると、外部でファイル名を割り当てることができます。 この方法では、ファイル名のマッピングをテキスト ファイル (マッパー ファイル) 中で処理できるため、COBOL プログラムで使用するファイル名を物理的なファイル名に柔軟にマップできます。ファイル名のマッピングは、ファイルを編集するだけで変更できます。

```
select filename
   assign to external assigned-name
```

*filename* と *assigned-name* の内容は次のとおりです。

*filename* プログラムで使用する論理 (内部) ファイル名。

*assigned-name* マッパー ファイルのエントリ。

外部ファイル マッパーを使用すると、オペレーティング システムの環境変数でファイル名の割り当てを決定する場合より、環境領域に必要なメモリ量を削減することができます。

### 3.5.1 マッパー ファイル構造

外部ファイル マッパーを使用するには、通常のテキスト ファイルを作成し (メモ帳などのテキスト エディタを使用して作成する)、mfextmap.dat という名前を付けておく必要があります。

このファイルの各行には、次のような形式でファイル名が割り当られています。

*assigned-name physical-name*

これらのパラメータの内容は、次のとおりです。

*assigned-name* プログラムで使用する、割り当て済みのファイル名。

*physical-name* 物理ファイルの実際の名前。パス名を含む。

- 1 行に書き込める割り当ては 1 つだけです。
- 割り当てられたファイル名と物理的なファイル名の間は、少なくとも 1 つの空白文字で区切る必要があります。

- 割り当てられたファイル名の前と物理的なファイル名の後には、無制限に空白文字を入れることができます。

プログラムの実行中にマッパー ファイルの内容を変更することができます。使用されるマッピング情報は、常に、現在のバージョンのマッパー ファイルに保存された情報です。

### 3.5.2 マッパー ファイルの場所

マッパー ファイルを作成する場合、このファイルを mfextmap.dat と命名し、次のどれかに格納する必要があります。

- 現在のディレクトリ (プログラムが実行されているディレクトリ )
- COBOL システム パス上のディレクトリ (環境変数 COBDIR により示されます)
- 環境変数 MFEXTMAP により指定されたパス上のディレクトリ

MFEXTMAP と COBDIR は、どちらも複数のパスを定義することができます。

設定された MFEXTMAP が指定するディレクトリにマッパー ファイルが存在しない場合、システムは、プログラムで使用する割り当て済みのファイル名と同じ名前をもつ環境変数を検索し、ファイル名を決定しようとします。

MFEXTMAP が設定されていない場合、システムは、まず現在のディレクトリでマッパー ファイルを検索します。ここで見つからないと、次に COBDIR パスにしたがって検索します。マッパー ファイルが見つからない場合、システムはプログラムで使用する割り当て済みのファイル名と同じ名前をもつ環境変数を検索し、ファイル名を決定しようとします。

### 3.5.3 外部ファイル マッパーのアクティブ化

外部ファイル マッパーを使用するには、次の手順にしたがいます。

- 共有ランタイム システムとプログラムをリンクする必要があります。
- 有効なマッパー ファイルを作成しておく必要があります。

次に、外部ファイル マッパーをアクティブ化するために、実行時チューナ environment_mapper を TRUE に設定します。この設定を行うには、$COBDIR¥cobopt.cfg 構成ファイル、または、COBCONFIG 環境変数が指定する構成ファイルに、次の行を追加します。

```
set environment_mapper=TRUE
```

### 3.5.4 外部ファイル マッパーの無効化

外部ファイル マッパーを無効化すると、ディスクにアクセスして現在のバージョンのマッパー ファイルを検索する必要がなくなるため、システムの処理速度が向上します。

外部ファイル マッパーを無効化するには、実行時チューナ environment_mapper を FALSE に設定してください。

# 第4章 ファイル状態

ファイル状態コードは、この COBOL システムでプログラムが実行したファイル操作の成否を示すために使用します。

ファイル状態コードの完全なリストは、オンライン ヘルプ ファイルに掲載されています。 ヘルプ ファイルの索引で [ファイル状態] を参照してください。

## 4.1 ファイル状態の意味

ファイル状態は、ファイル操作結果 (成功、または、エラーの発生など) を示す 2 バイトのコードです。エラーが発生した場合、ファイル状態にはエラー原因も表示されます。

ファイルにファイル状態のデータ項目が定義されている場合、ファイルに入出力操作 (OPEN、CLOSE、READ、WRITE、REWRITE、START、および DELETE) を行うたびに、ランタイム システムがファイル状態のデータ項目を更新し、操作結果を示します。

ファイル状態のデータ項目を定義するかどうかは選択できます。ファイル状態のデータ項目が宣言されていない場合に、重大なファイル エラーが発生すると、COBOL ランタイム システムはエラー メッセージを表示し、プログラムを中断します。

入出力操作が行われるたびに、ファイル状態のデータ項目を調べ、操作が成功したかどうかを確認してください。例えば、プログラムがディスクに書き込みを行っているときには、WRITE 操作を完了できるだけの十分なディスク領域がない場合もあります。ファイル状態のデータ項目を定義していない場合に、ディスク容量が不足すると、ランタイム システムはエラー番号を表示し、プログラムを中断します。(書き込み中のファイルについて) ファイル状態のデータ項目を定義してある場合、ファイル状態のデータ項目は更新され、プログラムの実行は継続されます。プログラムは後でファイル状態のデータ項目を調べ、ディスクが一杯であると判断すると、適切な措置を講じます。

## 4.2 ファイル状態データ項目の定義

ファイルに対して SELECT 文と FILE STATUS IS を指定し、作業場所節でファイル状態のデータ項目を定義すると、プログラムの各ファイルに、関連付けられたファイル状態のデータ項目を設定することができます。

```
select in-file
   assign to "user.dat"
   file status is ws-file-status.
   .
   .
   .
```

```
working-storage section.
01 ws-file-status       pic xx.
```

各ファイルで別個のデータ項目を使用することも、複数のファイルで 1 つのデータ項目を使用することも可能です。

ファイル状態は、2 バイトのコードです。拡張ファイル状態コード以外のファイル状態コードの表記規則では、データ項目は英数字 2 文字 (PIC XX) で指定されます。拡張ファイル状態コードの場合、2 番目のバイトにはバイナリの数字が格納されます。

ファイル状態のデータ項目に格納された最初のバイトは、状態キー 1 と呼ばれ、入出力操作の結果として次のどれかの状態を示します。

| 値 | 状態 |
|---|---|
| 0 | 操作の成功 |
| 1 | AT END (ファイルの終わり) |
| 2 | 無効なキー |
| 3 | 永続的なエラー |
| 4 | 論理エラー (入出力操作の順序が不適切) |
| 9 | COBOLランタイム システム エラー メッセージ |

ファイル状態データ項目の最初のバイトが 9 以外である場合、2 番目のバイト (状態キー 2 と呼ばれる) は英数字になります。

ファイル状態データ項目の最初のバイトが 9 である場合、2 番目のバイトは、Micro Focus が定義した RTS エラーコードを含むバイナリ フィールドになります。

次のコード例は、ファイル状態の確認方法を示します。まず最初のバイト (状態キー 1 ) を調べ、より詳細な情報が必要な場合に 2 番目のバイト (状態キー 2 ) を確認します。

```
select recseq
   assign to "recseq.dat"
   file status is ws-file-status
   organization is record sequential.
...
file section.
fd recseq
record contains 80 characters.
```

```
01 recseq-fd-record                          pic x(80).
...
working-storage section.
01 ws-file-status.
   05 status-key-1                           pic x.
   05 status-key-2                           pic x.
   05 binary-status redefines status-key-2  pic 99 comp-x.
...
procedure division.
...
perform check-status.
...
check-status.
   evaluate status-key-1
      when "0" next sentence
      when "1" display "ファイルの終わりに到達"
         perform check-eof-status
      when "2" display "無効なキー"
         perform check-inv-key-status
      when "3" display "永続的エラー"
        perform check-perm-err-status
      when "4" display "論理エラー"
      when "9" display "ランタイム システム エラー"
         perform check-mf-error-message
   end-evaluate.
...
check-eof-status.
   if status-key-2 = "0"
     display "論理レコードが残っていません。"
   end-if.
...
```

```
check-inv-key-status.
   evaluate status-key-2
     when "2" display "重複キーを作成しようとしています。"
     when "3" display "レコードが見つかりません。"
  end-evaluate.
...
check-perm-err-status.
   if status-key-2 = "5"
     display "ファイルが見つかりません。"
   end-if.
...
check-mf-error-message.
   evaluate binary-status
      when 002 display "ファイルを開けません。"
      when 007 display "ディスク領域が不足しています。"
      when 013 display "ファイルが見つかりません。"
      when 024 display "ディスク エラー"
      when 065 display "ファイルがロックされています。"
      when 068 display "レコードがロックされています。"
      when 039 display "レコードが一致しません。"
      when 146 display "現在のレコードがありません。"
      when 180 display "ファイル形式が正しくありません。"
      when 208 display "ネットワーク エラー"
      when 213 display "ロックが多すぎます。"
      when other display "エラー状態ではありません。"
      display binary-status
  end-evaluate.
```

## 4.3 ファイル状態の規則

この COBOL システムでは、ファイル状態の規則を数多くサポートしています。各規則では、コードを個別に定義していますが、規則の間で重複するものもあります。

サポートされている規則を次に示します。

- ANSI'85 ファイル状態
- ANSI'74 ファイル状態
- Microsoft COBOL V2 ファイル状態

### 4.3.1 ANSI'85 ファイル状態

ANSI'85 の操作用に読み込まれた標準システムを使用している場合、デフォルトでは、ANSI'85 ファイル状態コードが作成されます。

### 4.3.2 ANSI'74 ファイル状態

コンパイラ指令 NOANS85 でプログラムをコンパイルする場合、ANSI'74 ファイル状態コードが作成されます。

ANSI'74ファイル状態コードが作成されている場合に ANSI'85 構文を使用するには、NOANS85 コンパイラ指令を ANS85"SYNTAX" に置き換えてください。

### 4.3.3 Microsoft COBOL V2 ファイル状態

MS"2" コンパイラ指令と NOANS85 コンパイラ指令を使用してプログラムをコンパイルすると、Microsoft COBOL V2 ファイル状態コードが作成されます。コンパイル時に NOANS85 を指定していない場合は、ANSI'85 ファイル状態コードが作成されます。

Microsoft COBOL V2 ファイル状態コードが作成されている場合に ANSI'85 構文を使用するには、NOANS85 コンパイラ指令を ANS85"SYNTAX" に置き換えてください。

Microsoft COBOL V2 ファイル状態コードの完全なリストは、オンライン ヘルプに掲載されています。ヘルプ ファイルの索引で [ファイル状態] を参照してください。

## 4.4 拡張ファイル状態コード

ANSI'74 と ANSI'85 のファイル状態規則は、拡張ファイル状態コードで拡張されます。 拡張ファイル状態コードでは、ファイル状態の最初のバイトが "9" になります。2 番目のバイトはバイナリ (COMP-X) の数字で、ランタイム エラー番号と同じになります。

拡張ファイル状態コードは、9/*nnn* のように表示されます。*nnn* は 2 番目のバイトのバイナリの数字を指します。

例えば、ディスクにファイルを書き込んでいるときにディスク容量が足りなくなった場合、ANSI'74 ファイル状態は "30" になります。これは、次に示すエラー メッセージに変換されます。

**永続的エラー。これ以外の情報はありません。**

このエラー メッセージは非常に一般的です。"永続的エラー" は、ディスクに障害があるか、ディスク ドライブのふたが開いていることを示します。このCOBOLシステムでは、包括的なファイル状態を返すのではなく、拡張ファイル状態 9/007 を返します。最初のバイトの文字 "9" と 2 番目のバイトのバイナリ値 7 を合わせると、「ディスクがいっぱいです。」という意味になります。

ANSI'74 または ANSI'85 のファイル状態コードを使用している場合に、より具体的な拡張ファイル状態があると、ランタイム システムは拡張ファイル状態コードを返します。

拡張ファイル状態コードの完全なリストは、オンライン ヘルプに掲載されています。ヘルプ ファイルの索引で [ファイル状態] を参照してください。

次のコード例は、標準のファイル状態を拡張ファイル状態として使用できるように再定義する方法を示します。この例では、入力ファイルが存在しないことが想定されています。そのため、OPEN INPUT 文を実行すると、9/013 ("ファイルが見つかりません。") というファイル状態が返されます。

```
select in-file
   assign to "user.dat".
   file status is file-status.
…
working-storage section.
01 file-status.
   05 status-key-1           pic x.
   05 status-key-2           pic x.
   05 status-key-2-binary redefines status-key-2 pic 99 comp-x.
…
procedure division.
open input in-file
if file-status not = "00"
   if status-key-1 = "9"
      if status-key-2-binary = 13
        display "ファイルが見つかりません。"
…
```

拡張ファイル状態コードを表示する場合、最大値 255 を格納できるだけの表示フィールドにファイル状態データ項目の 2 番目のバイトを移動する必要があります。

```
select in-file
```

```
   assign to "user.dat"
   file status is file-status.
…
working-storage section.
01 ans74-file-status.
   05 status-key-1           pic x.
   05 status-key-2           pic x.
   05 status-key-2-binary redefines status-key-2 pic 99 comp-x.
01 display-ext-status
   05 filler                 pic xx value "9/"
   05 display-key 2          pic 999
…
procedure division.
open input in-file
if file-status not = "00"
   display "エラー。ファイル状態 =" with no advancing
if status-key-1 = "9"
   move status-key-2-binary to display-key-2
   display display-ext-status
else
   display file-status
end-if
```

# 第5章 ファイルの共有

マルチユーザー環境でデータを共有する場合、複数のユーザーが同じデータを同時に変更できないようにする必要があります。 ファイルやレコードをロックすると、データの完全性を維持しながら、マルチユーザー環境のプログラムで共通ファイルを共有することができます。

## 5.1 共有モード

共有モードは、特定のファイルについてファイルの共有とレコードのロックを行う場合に指定します。

共有モードには 3 種類あります。

- SHARING WITH NO OTHER

  このモードを設定したファイルを他のユーザーと共有することはできません。

- SHARING WITH READ ONLY

  ファイルを入力ファイルとして開く場合、他のユーザーとの共有が可能です。

- SHARING WITH ALL OTHER

  他のすべてのユーザーとの共有が可能です。

各共有モードでのファイル オープン操作結果を示す詳細な表については、『*言語リファレンス*』の「*手続き部 - MERGE - OPEN*」の章にある「*別のファイル コネクタで現在開いている利用可能な共有ファイルを開く*」の表を参照してください。

これらの共有モードを指定するには、OPEN 文で、または、SELECT 文の一部として語句 SHARING を使用します。OPEN 文で語句 SHARING を使用すると、SELECT 文の語句 SHARING は無効になります。

SELECT 文または OPEN 文のどちらにも語句 SHARING を指定しない場合、共有モードは、次のどれかの条件が最初に満たされるときに決定されます。

- OPEN 文で語句 WITH LOCK を指定した場合、共有モードは SHARING WITH NO OTHER になります。
- SELECT 文で語句 LOCK MODE IS EXCLUSIVE を指定した場合、共有モードは SHARING WITH NO OTHER になります。
- 語句 LOCK MODE IS MANUAL または LOCK MODE IS AUTOMATIC を SELECT 文で指定した場合、共有モードは SHARING WITH ALL OTHER になります。
- オープン モードが OUTPUT、I-O、または EXTEND の場合、共有モードは SHARING WITH NO OTHER になります。

- オープン モードが INPUT の場合、共有モードは構成オプション OPENINPUTSHARED の設定によって決まります。このオプションをオンに設定した場合、共有モードは SHARING WITH ALL OTHER になります。それ以外の場合、共有モードは SHARING WITH READ ONLY になります。

## 5.2 レコード ロック

ファイルの共有が有効である場合、個々のレコードにロックを設定することができます。 レコードにロックを設定すると、他のユーザーは、ロックされたレコードにアクセスできず、ロック解除されているレコードにアクセスするようになります。

---

注記: 行順ファイルや入力ファイルとして開いたファイルでは、レコードをロックできません。

---

### 5.2.1 手動レコード ロックと自動レコード ロック

レコードをロックするには、手動または自動のどちらかで行います。

#### 5.2.1.1 手動レコード ロック

手動レコード ロックを使用する場合、READ 文を使用してレコード ロックを取得してください。

手動ロックを使用するには、次のように指定します。

1. ファイルに対して SELECT 文と LOCK MODE IS MANUAL 句を指定します。
2. READ 文と WITH LOCK 句を指定します。

例

```
select fd-name
   assign to "muser.dat"
   lock mode is manual
...
read fd-name with lock
```

#### 5.2.1.2 自動レコード ロック

自動レコード ロックを使用する場合、READ 文はレコード ロックを自動的に取得します。レコードをロックしない場合、READ 文で WITH NO LOCK 句を指定してください。

自動ロックを指定するには、ファイルに対して SELECT 文と LOCK MODE IS AUTOMATIC 句を使用してください。

```
select fd-name
   assign to "muser.dat"
   lock mode is automatic
...
read fd-name
```

## 5.2.2 単一および複数のレコード ロック

ロックするレコード数を、単一または複数のどちらかに指定できます。

### 5.2.2.1 単一レコード ロック

単一レコード ロックを使用する場合、1 回に 1 つのレコード ロックしか保持できません。ファイルで新しく入出力操作を行うたびに、前のロックが解放されます。

単一のレコード ロックを指定するには、LOCK MODE IS MANUAL 句、または LOCK MODE IS AUTOMATIC 句の後に WITH LOCK ON MULTIPLE RECORDS 句を使用しません。

### 5.2.2.2 複数レコード ロック

複数レコード ロックでは、同時に多くのレコード ロックを保持できます。

複数レコード ロックを指定するには、LOCK MODE IS MANUAL 句または LOCK MODE IS AUTOMATIC 句の後に WITH LOCK ON MULTIPLE RECORDS 句を続けます。例えば、次のようになります。

```
select fd-name
   assign to "muser.dat"
   lock mode is automatic with lock on multiple records
...
```

---

注記:

- WITH LOCK ON MULTIPLE RECORDS 句を使用しない場合、単一のレコード ロックが有効になります。
- ファイルに複数のレコード ロックを指定し、プログラムをコンパイルするときに WRITELOCK コンパイラを使用する場合、プログラムは、WRITE 文や REWRITE 文でファイルにアクセスするたびに、レコード ロックを取得します。

---

### 5.2.3 レコード ロックの処理

デフォルトでは、読み取り処理中にロックされたレコードを検出すると、レコード ロック状態コードを返します。この場合、レコード領域は更新され、レコード内容が表示されます。ただし、現在のレコード ポインタはロックされたレコードと関連付けられたままであるため、この操作の後に続く READ NEXT 操作で同じレコードへ再アクセスします。

次の構成オプションは、上記の処理方法に影響します。

- IGNORELOCK

  これを ON に設定すると、入力用に開いたファイルを読み取るときにレコード ロックを無視します。

- RETRYLOCK

  これを ON に設定すると、ロックされたレコードへのアクセス処理が成功するまで再試行します。 整数を設定すると、操作を中断するまでの最大再試行回数を指定できます。

- LOCKTYPE

  1 に設定すると、レコード ロックはレコード自体の上に配置されます。この設定では、読み取り処理中にロックされたレコードを検出した場合、そのデータ レコードは戻りません。この場合、読み取り処理後のレコード領域の内容は未定義の状態になります。

- SKIPLOCK

  これを ON に設定した場合、読み取り処理中にロックされたレコードを検出すると、READ NEXT 操作では、ロックされたレコードの次にあるレコードが返されます。

---

注記 ロックされたレコードをバイパスするために、START…KEY IS GREATER THAN.. 文を使用することもできます。

---

### 5.2.4 レコード ロックの解放

ロックを取得したプログラムが次のどれかの操作を行うと、すべてのレコード ロックが解放されます。

- ファイルを閉じる。
- COMMIT 文を実行する。
- ROLLBACK 文を実行する。

- ファイルに対して UNLOCK 文を実行する。
- ロックされたレコードを削除する。

さらに、ロックを取得したプログラムが START 以外のファイル操作でファイルの他のレコードにアクセスした場合、単一レコード ロックが解放されます。

## 5.3 ファイル状態コード

プログラムをマルチユーザー環境で実行する場合、次のファイル状態コードを常に確認することが必要です。

| | |
|---|---|
| 9/065 | この状態コードは、OPEN 文により返され、ファイルがロックされていることを示します。 |
| 9/068 | この状態コードは、READ 文、DELETE 文 または REWRITE 文により返され、レコードがロックされていることを示します。 |
| 9/213 | この状態コードは、最大許容ロック数に達したことを示します。最大許容ロック数は、ネットワークとオペレーティング システムによって異なります。 |

## 5.4 ファイル ロックとレコード ロックのアプリケーション例

ファイル ロックとレコード ロックのアプリケーション例は、NetExpress の基本インストール ディレクトリにある demo ディレクトリの locking ディレクトリにあります。

このアプリケーションは、メイン プログラムである LOCKING.CBL と、このメイン プログラムで呼び出される他のプログラムから構成されています。

このアプリケーションを実行するときには、最初にオプション 4 を使用してファイルを作成します。レコード ロックの影響を表示するには、複数のセッションでオプション 2 と 3 を使用してください。

# 第6章 ファイル ハンドラの構成

ファイル ハンドラの動作の一部は、構成することができます。これらの動作は、ファイル ハンドラ構成ファイルに値を入力して変更します。

## 6.1 構成ファイル

ファイル ハンドラ構成ファイルのデフォルトのファイル名は extfh.cfg です。ただし、次のように EXTFH 環境を設定し、別のファイル名を使用することもできます。

```
set extfh=c:¥mydir¥test.cfg
```

上記のように記述すると、ファイル名は c:¥mydir¥test.cfg に設定されます。

ファイル ハンドラ構成ファイルを使用して、個々のファイルまたはすべてのファイルについて、ファイル ハンドラ パラメータを変更することができます。すべてのファイルに適用する設定は、次に示すタグの下に一覧表示されます。

```
[XFH-DEFAULT]
```

また、個々のファイルに適用する設定は、次のような個々のファイル名の下に一覧表示されます。

```
[TEST.DAT]
```

すべてのファイルに対する設定より、個々のファイルに対する設定の方が優先されます。

## 6.2 構成オプション

次の表は、ファイル ハンドラ構成ファイルで設定できるパラメータのリストです。

| パラメータ | 説明 |
|---|---|
| COMMITFLUSH | FILESHARE を使用していないプログラムのコンテキストで、COMMIT 文や ROLLBACK 文により、レコード ロックを解放するだけでなく、すべてのファイル更新をディスクに書き出すかどうかを指定します。 |
| CONVERTSTATUS | 呼び出すプログラムの名前を指定し、ファイル ハンドラが処理を完了した後で、返された状態値をエミュレーション用にマップします。 |
| DATACOMPRESS | データ圧縮を有効化するかどうかと圧縮の種類を指定します。 |
| DATAFILE | OPEN 文に渡されたファイル名を別の名前にマップします。 |

| パラメータ | 説明 |
| --- | --- |
| KEYCOMPRESS | 使用中のキー圧縮の種類を指定します。 |
| LOCKTYPE | 使用中のレコード ロックの種類を指定します。 |
| MAINFRAMEPRINT | WRITE AFTER ADVANCING または WRITE を使用しているファイルについて、メインフレームのプリンター形式 (filetype(11)) を使用するかどうかを指定します。 |
| NAMEOPTIONS | OPEN 文に渡す名前に、角かっこ ([]) で囲まれたファイル ハンドラ指令を含めてよいかどうかを指定します。 |
| NFSFILELOCK | UNIX NFS ファイル システムのレコード ロックとファイル ロックをアプリケーションで検出できるようにします。 |
| NODESIZE | 索引ファイルに使用する索引ノードのサイズを指定します。 |
| OPENINPUTSHARED | 入力用に開いたファイルに LOCK MODE 句を指定していない場合、このファイルを他のユーザーと共有可能にするかどうかを指定します。 |
| OSVSREWRITE | 出力用に開いた順ファイルに対して WRITE 文を使用可能にするかどうかを指定します。WRITE 文を使用可能にした場合、これらの WRITE 文は、REWRITE 文と全く同じように動作します。 |
| READSEMA | ファイル修正以外の目的で入出力操作を実行する場合に、システムに共有ファイルのセマフォを獲得させるかどうかを指定します。 |
| RELDATBUF | ファイルにアクセスするときに使用するバッファのサイズを指定します。 |
| RETRYLOCK | ある操作がロックされたレコードに対してアクセスを試行したときに、この操作を再試行するかどうかを指定します。 |
| RETRYOPEN | ある操作がロックされたファイルに対してアクセスを試行したときに、この操作を再試行するかどうかを指定します。 |
| RETRYTIME | RETRYLOCK パラメータまたは RETRYOPEN パラメータに設定された整数が、試行回数と秒数のどちらを表すかを指定します。 |
| RUNITLOCKDETECT | 同じ実行ユニットで実行された別の OPEN 文によりレコードがロックされている場合に、これを検知できるようにするかどうかを指定します。 |

| パラメータ | 説明 |
|---|---|
| SEQDATBUF | ファイルにアクセスするときに使用するバッファのサイズを指定します。 |
| EXPANDPOSITIONING | WRITE AFTER POSITIONING 文 (OS/VS COBOL 互換) を使用するときに、レコードにキャリッジ制御情報を書き込むかどうかを指定します。 |
| EXPANDTAB | 行順ファイルまたは行送りファイルで READ 操作中に検出されたタブ コードを同じ数の空白文字に拡張するかどうかを指定します。 |
| FASTREAD | 索引ファイルを読み取るときに、ファイル ハンドラで、データの完全性を保証するための追加確認を実行するかどうかを指定します。 |
| FHREDIR | FILESHARE を使用してリモート サーバーでのファイル処理を可能にするかどうかを指定します。 |
| FILEMAXSIZE | 呼び出しのロック時に使用するオフセットを維持するために、必要なバイト数を指定します。 |
| FILEPOINTERSIZE | 形式 8 の索引ファイルについて、ファイル ポインタを格納するために使用するバイト数を指定します。 |
| IDXDATBUF | ファイルのデータ部分にアクセスするときに使用するバッファのサイズを指定します。 |
| IDXFORMAT | 索引ファイルを作成するときに使用する形式を指定します。 |
| IDXNAMETYPE | データ ファイルと、索引ファイル (存在する場合) の両方のファイル名の形式を指定します。 |
| IGNORELOCK | 入力用に開いたファイルを読み取るときに、ロックを無視するかどうかを指定します。 |
| INDEXCOUNT | 索引ファイルについてキャッシュする索引ノードの数を指定します。 |
| INSERTNULL | 行順ファイルや行送りファイルに WRITE 操作、または REWRITE 操作を実行する場合、印刷不可能な文字の前に NULL 文字 (x"00") を挿入するかどうかを指定します。また、READ 操作中にこれらの NULL 文字を削除するかどうかも指定します。 |
| INSERTTAB | 行順ファイルや行送りファイルに WRITE 操作、または REWRITE 操作を実行する場合、タブ コードを連続した空白文字に置き換えるかどうかを指定します。 |
| KEYCHECK | アプリケーションで定義されたキー定義と開いている索引ファイルのキー定義が一致することをファイル ハンドラで確認するかどうかを指定します。 |

| パラメータ | 説明 |
|---|---|
| SKIPLOCK | 順読み取り中にロックされたレコードが検出された場合に、現在のレコード ポインタまでジャンプするかどうかを指定します。 |
| SPACEFILL | 行順ファイルや行送りファイルに READ 操作を実行する場合、読み取ったデータ以外のレコード領域を空白文字で埋めるかどうかを指定します。 |
| STRIPSPACE | 行順ファイルや行送りファイルに WRITE 操作または REWRITE 操作を実行する場合、後続の空白文字を削除するかどうかを指定します。 |
| SUPPRESSADV | レコード順ファイルの場合、WRITE 文の ADVANCING 句を無視するかどうかを指定します。 |
| TRACE | Xfhtrace を有効化するどうかを指定します。 |
| TRACEFILE | TRACE オプションがアクティブなときに、作成するトレース ファイルの名前を指定します。 |
| WRITETHRU | ファイルの修正をすぐにディスクに書き出すかどうかを指定します。修正を COBOL システムまたはオペレーティング システムのどちらかによって内部でバッファに記録し、後でディスクに書き出すこともできます。 |

有効な設定の詳細とこれらのオプションのデフォルト値については、NetExpress オンライン ヘルプ ファイルにあるヘルプを参照してください(ヘルプ トピックのボタンをクリックし、[目次] タブがアクティブであることを確認してから、[参照]、[ファイル処理]、[ファイル ハンドラ]、[構成ファイル パラメータ]を選択してください)。 また、オンライン ヘルプ ファイルには、構成ファイルの例も掲載されています ([プログラミング]、[ファイル処理]、[例]、[ファイル ハンドラ]、[構成ファイル] を選択してください)。

## 6.3 構成可能な動作

ファイル ハンドラ構成ファイルで構成できるファイル ハンドラの動作は、次のとおりです。この項では、オンライン ヘルプよりも詳しく説明します。

### 6.3.1 大容量索引ファイル

デフォルトでは、ファイル ハンドラは、最大 2 ギガバイトのファイルを処理できます。 ただし、2 ギガバイトより大きい索引ファイルを作成する必要がある場合は、ファイル ハンドラ構成ファイルで IDXFORMAT パラメータを 8 に設定し、IDXFORMAT"8" ファイルを作成します。 通常、索引ファイルは .idx ファイルと .dat ファイルに分かれているのに対して、このファイルは単一のファイルとして構成され、最大 32 テラバイトのサイズにすること

ができます。

多くのオペレーティング システムでは、ファイル サイズに制限があります (例えば、Windows 95では 2 ギガバイト)。 これらのシステムで大容量ファイルを作成できるようにするには、ストライプ化オプションを使用します。ストライプ化は大容量論理ファイルを必要な数の物理的なファイルに分割します (最大数 256)。

## 6.3.2 ファイルのストライプ化

ファイル ストライプ化オプションを使用すると、最大 512 ギガバイトの論理ファイルを作成することができます。これらの論理ファイルは、最大 256 の物理ファイルで構成することができます。

これらの大容量の論理ファイルは複数の物理ファイルで構成されるため、ファイル ストライプ化をグローバルに行うと、オペレーティング システムでファイル ハンドルが足りなくなる可能性があります。そのため、ファイル ストライプ化をグローバルに行うことはできません。ストライプ化するファイルは、選択する必要があります。

ファイルをストライプ化するには、ファイル ハンドラ構成ファイルでファイルに STRIPING=ON パラメータを設定します。

### 6.3.2.1 オプション

ストライプ化オプションでは、大文字小文字を区別しません。

---

注記

- ストライプ化したファイルを使用する場合、ファイルを開くたびに、ファイル ハンドラ構成ファイルで必要なストライプ化オプションが指定されている必要があります。
- ファイルをストライプ化している場合に、ストライプの 1 つを削除すると (または、ディスク障害などの理由で使用不能になった場合)、ファイル全体を使用できなくなります。

---

`STRIPING=[ON/OFF]`

ファイルをストライプ化するかどうかを指定します。デフォルトの設定はオフです。

`STRIPENAMETYPE=[0/1]`

このオプションでは、ストライプに名前を付けるためのファイル命名規則を指定します。デフォルトの設定は 0 です。

StripeNameType を 0 に設定すると、ストライプ ファイルは基本ファイルと同じ名前になりますが、ファイル拡張子の前にある基本名にストライプ番号が追加されます。例えば、ファイル test.dat に 3 個のストライプがある場合、基本ファイルとそのストライプは次のように表記します。

test.dat - 基本ファイル
test01.dat - ストライプ番号 1
test02.dat - ストライプ番号 2
test03.dat - ストライプ番号 3

基本名が長く、そのままではストライプ ファイル番号を入れることができない場合、基本名の右端から文字が削除されます。例えば、ファイル testfile.dat に 3 個のストライプがある場合、ファイルとそのストライプは次のように表記されます。

testfile.dat - 基本ファイル
testfi01.dat - ストライプ番号 1
testfi02.dat - ストライプ番号 2
testfi03.dat - ストライプ番号 3

StripeNameType を 1 に設定すると、ストライプ ファイル名は基本ファイルと同じ名前になりますが、ファイル名のファイル拡張子の後にストライプ番号が追加されます。基本ファイル名はストライプ 0 に変更されることに注意してください。例えば、ファイル test.dat に 3 個のストライプがある場合、基本ファイルとそのストライプは次のように表記されます。

test.dat.00
test.dat.01
test.dat.02

MAXSTRIPESIZE=*n*

個々のストライプの最大サイズをバイト数で指定します。*n* のデフォルト値は 1 ギガバイト (1,073,741,824バイト) です。最小値は 65,536 バイトです。最大値は現在のところ 2 ギガバイト (2,147,483,648 バイト) です。

MAXSTRIPEFILES=*n*

基本ファイル以外のストライプ ファイル数を指定します。デフォルトでは、17 ファイルが作成されます。つまり、基本ファイルと 16 個のストライプです。*n* の最小値は 1 で、最大値は 255 です。

MAXSTRIPEDIGITS=*n*

ストライプに名前を付けるときに、基本ファイルの名前に追加する桁数を指定します。 例えば、2 と指定すると、test01.dat となり、5 と指定すると test00001.dat のようになります。5 と指定した場合にオペレーティング システムでファイル名が制限されていると、tes0001.dat となることがあります。*n* のデフォルト値は 2 です。最小値は 1 で、最大値は 5 です。

STRIPE-X=*path*[,*n*]

ストライプ ファイルのパスとサイズを指定します。*x* はストライプ番号、*path* はストライプがある場所、*n* はストライプのサイズを示します。*n* を省略すると、ストライプのサイズは前のストライプと同じになります。

STRIPE-X オプションを一度設定すると、もう一度 STRIPE-X オプションで変更するまで、すべてのストライプに

新しいパスとサイズが適用されます。Stripe- の後に続く数字はストライプ番号で、0 は基本ファイルを表します。基本ファイル以外のストライプには、0 という文字を指定しないでください。

#### 6.3.2.2 命名規則

ファイルをストライプ化すると、ファイルは基本ファイル (プログラムが指定するファイル名) と多くのストライプファイルで構成されるようになります。ストライプ ファイル名は基本ファイル名と同じですが、ストライプ番号が追加されます。例えば、ファイル test.dat に 3 個のストライプがある場合、基本ファイル名とそのストライプは次のように表記されます。

test.dat - 基本ファイル
test01.dat - ストライプ番号 1
test02.dat - ストライプ番号 2
test03.dat - ストライプ番号 3

基本名が長く、ストライプ ファイル番号を入れることができない場合、基本名の右端から文字が削除されます。例えば、ファイル testfile.dat に 3 個のストライプがある場合、ファイルとそのストライプは次のように表記されます。

testfile.dat - 基本ファイル
testfi01.dat - ストライプ番号 1
testfi02.dat - ストライプ番号 2
testfi03.dat - ストライプ番号 3

#### 6.3.2.3 例

ファイルのストライプ化を指定する方法について、次に例を示します。

例 1

ファイル test.dat に、同じサイズの 2 つのストライプを作成すると仮定します。

```
[test.dat]
Striping=on
Stripe-1=dir2
Stripe-2=dir3
```

この例では、test.dat を現在のディレクトリに、test01.dat を dir2 に、test02.dat を dir3 に格納します。

例 2

ファイル test.dat に 5 つのストライプを作成すると仮定します。

```
[test.dat]
```

```
Striping=on
Stripe-0=dir1
Stripe-4=dir2
```

この例では、test.dat、test01.dat、test02.dat、および test03.dat を dir1 に、 test04.dat と test05.dat を dir2 に格納します。

例 3

ファイル test.dat に 3 個のストライプを作成すると仮定します。

```
[test.dat]
Striping=on
Stripe-0=dir1,100000000
Stripe-1=dir2,200000000
```

この例では、test.dat のファイルサイズを 100,000,000 バイトに設定して dir1 に格納し、test01.dat のファイルサイズを 200,000,000 バイトに設定して dir2 に格納します。

### 6.3.3 データ圧縮

レコード順ファイルまたは索引ファイルでは、データを圧縮し、ディスク領域を節約することができます。

固定長形式の順ファイルにデータ圧縮を指定すると、可変長形式の順ファイルに変換されます。この操作は、REWRITE 文を使用しない限り、プログラムに影響しません。圧縮された順ファイルで圧縮の結果として長さが変更されたレコードに REWRITE 文を実行すると失敗します。

ファイル ハンドラ構成ファイルの DATACOMPRESS パラメータを使用すると、ファイルのデータ圧縮を指定することができます。また、プログラムのコンパイル時に、DATACOMPRESS コンパイラ指令を使用してデータ圧縮を指定することもできます。

ファイルのデータ圧縮は、そのファイルに対する SELECT 文の処理時に最後に処理された DATACOMPRESS 指令で決定されます。そのため、プログラムで SELECT 文のすぐ前に次のような形式の行を記述すると、個々のファイルのデータ圧縮を指定することができます。

```
$SET DATACOMPRESS
```

別のファイルを処理する前に $SET NODATACOMPRESS を解除することを忘れないでください。

### 6.3.4 キー圧縮

キーを圧縮すると、索引ファイルのキーを圧縮し、ディスク領域を節約できます。キー圧縮には、次の 4 種類があります。

- 後続ヌル文字の圧縮
- 後続空白文字の圧縮
- 先頭文字の圧縮
- 重複キーの圧縮

キー圧縮は、ファイル ハンドラ構成ファイルの KEYCOMPRESS パラメータで、圧縮の種類を示す次の整数を使用して指定します。

1　　重複キーの圧縮

2　　先頭文字の圧縮

4　　後続空白文字の圧縮

8　　後続ヌル文字の圧縮

これらの数字をいくつかまとめて追加すると、圧縮の種類を組み合わせて使用できます (ただし、相互排他的な後続ヌル文字と後続空白文字は一緒に指定することができません)。

また、プログラムのコンパイル時に、KEYCOMPRESS コンパイラ指令を使用してキー圧縮を指定することもできます。

ファイルのキー圧縮は、そのファイルに対する SELECT 文の処理時に最後に処理された KEYCOMPRESS 指令で決定されます。そのため、プログラムで SELECT 文のすぐ前に次のような形式の行を記述すると、個々のファイルのデータ圧縮を指定することができます。

```
$SET KEYCOMPRESS"8"
```

他のファイルを処理する前に、$SET NOKEYCOMPRESS を指定すると、キー圧縮を解除することができます。

#### 6.3.4.1 後続ヌル文字

キーに後続ヌル文字の圧縮を定義すると、キー値の後続ヌル文字はファイルに格納されません。

例えば、30 文字の長さの主キーまたは副キーの最初の 10 文字だけを使用し、残りがヌル文字であるレコードを書き込むとします。圧縮をしない場合、キーの 30 文字すべてが格納されるため、30 バイトが必要になります。後続ヌル文字の圧縮を行うと、11 バイトだけですみます (10 バイトを最初の 10 文字用に、1 バイトを後続ヌル文字用に使用する)。

#### 6.3.4.2 後続空白文字

キーに後続空白文字の圧縮を定義すると、キー値の後続空白文字はファイルに格納されません。

例えば、30 文字の主キーまたは副キーで最初の10文字だけを使用し、残りが空白文字であるレコードを書き込むとします。圧縮をしない場合、キーの 30 文字すべてが格納されるため、30 バイトが必要になります。後続空白文字の圧縮を行うと、11 バイトだけですみます (10 バイトを 10 文字用に、1 バイトを後続空白文字用に使用する) 。

#### 6.3.4.3 先頭文字

キーに先頭文字の圧縮を定義すると、前のキーの先頭文字と一致する先頭文字はすべてファイルに格納されません。

例えば、次のキー値を持つレコードを書き込むとします。

AXYZBBB BBCDEFG BBCXYZA BBCXYEF BEFGHIJ CABCDEF

先頭文字を圧縮すると、索引ファイルに実際に格納されるキーは、次のようになります。

AXYZBBB BBCDEFG XYZA EF EFGHIJ CABCDEF

カウントフィールドは、前のキーと同じ先頭文字の数を示します。

#### 6.3.4.4 重複したキー

副キーに重複キーの圧縮を定義すると、最初の重複キーだけがファイルに格納されます。

例えば、"ABC" という副キー値をもつレコードを書き込むとします。重複キーの圧縮を有効化し、同じキー値を持つ別のレコードを書き込んだ場合、重複した値は物理的にファイルに格納されません。

# 第7章 ファイル ハンドラ API

## 7.1 概要

次の構文を使用すると、プログラムでファイル ハンドラを明示的に呼び出すことができます。

```
call "EXTFH" using opcode fcd
```

各パラメータの内容は次のとおりです。

| | |
|---|---|
| "EXTFH" | ファイル ハンドラ インターフェイスのモジュール名です。 |
| *opcode* | ファイル ハンドラのオペレーション コードです。後述の「*オペレーション コード*」を参照してください。 |
| *fcd* | ファイル ハンドラがアクセスするファイルの詳細を保持する FCD (ファイル制御記述) と呼ばれるデータ領域です。後述の「*データ構造体*」の項を参照してください。 |

最初の呼び出し前に、次の手順を行う必要があります。

1. ファイル制御記述 (FCD)、レコード領域、ファイル名領域、キー定義ブロック (索引ファイルの場合) にデータ領域を割り当てます。
2. ファイル ハンドラが無効な値を受け取らないように、すべてのデータ領域をバイナリのゼロに初期化します。
3. FCD に次の領域へのポインタを設定します。
    - レコード領域
    - ファイル名領域
    - キー定義ブロック (索引ファイルの場合のみ)

その後で、各ファイル ハンドラ操作に対して次の手順を行います。

1. 選択したオペレーション コードに対応する FCD のフィールドに入力を行います。
2. ファイル ハンドラを呼び出します。
3. ファイル状態を確認して、ファイルの入出力操作が正常に行われたかどうかを判断します。
4. FCD フィールドやレコード領域のデータを処理します。

## 7.2 データ構造体

ファイル ハンドラでは、ファイルの入出力操作中に次の 4 つのデータ構造体を使用します。

- ファイル制御記述

  FCD は 100 バイトのデータ領域で、使用中のファイルに関する情報が記述されます。

- レコード領域

  レコード領域は、レコードの読み取りや書き出しを行うデータ領域です。

- ファイル名領域

  ファイル名領域は、使用中のファイルの名前を記述するデータ領域で、オペレーティング システムにより認識されます。

- キー定義ブロック

  キー定義ブロックには、ファイル ハンドラが索引ファイルの操作中に索引キー情報を記述します。

### 7.2.1 ファイル制御記述 (FCD)

ファイル制御記述 (FCD) は、使用中のファイルに関する情報を記述する 100 バイトのデータ領域です。

コピー ファイル xfhfcd.cpy (NetExpress 基本インストール ディレクトリにある ¥source ディレクトリに格納されています) には、FCD の COBOL 定義が含まれます。

ファイル ハンドラを使用するには、プログラムで FCD をセットアップし、該当するフィールドへの入力 (オペレーション コードによってフィールドが異なります) とファイル ハンドラの呼び出しを行います。FCD の該当フィールドに入力を行うと、ファイル ハンドラがプログラムに情報を返します。

FCD で未使用の領域や予約済みの領域は、バイナリ ゼロに設定する必要があります。

ファイル オープン呼び出しに使用した FCD は、その後このファイルに対してアクセスするときに必ず使用する必要があります。1 つのファイルに対して複数のオープン操作を実行できますが、各オープン操作ごとに別の FCD を使用する必要があります。

FCD には、レコード領域、ファイル名領域、キー定義ブロックへのポインタ (USAGE POINTER データ項目) を記述します。

FCD 構造体の詳細は、オンライン ヘルプ ファイルに記載されています。ヘルプ ファイルの索引で [*FCD*] を参照してください。

### 7.2.2 レコード領域

レコード領域は、レコードの読み取りと書き出しを行うデータ領域です。

レコード領域のサイズは、ファイルの最大レコードよりも 4 バイト大きい必要があります。

次の例は、FCD にレコード領域へのポインタを設定する方法を示します。

```
01 RECORD-AREA     PIC X(85).
...
   SET FCD-RECORD-ADDRESS TO ADDRESS OF RECORD-AREA
   ...
```

### 7.2.3 ファイル名領域

ファイル名領域は、使用中のファイルの名前を含むデータ領域で、オペレーティング システムにより認識されます。

ドライブやパスの情報だけではなく、ファイルの実際の名前も記述することができます。名前の最後は空白文字でなければなりません。

このファイル名領域は、ファイルに対する最初の操作より前に記述する必要があります。

次の例は、FCD でファイル名領域へのポインタを設定する方法を示します。

```
01 FILENAME-AREA   PIC X(65) VALUE "master.dat".
...
   MOVE 65 TO FCD-NAME-LENGTH
   SET FCD-FILENAME-ADDRESS TO ADDRESS OF FILENAME-AREA
   ...
```

### 7.2.4 キー定義ブロック

キー定義ブロックは、索引ファイルを開くときに、ファイル ハンドラに索引キー情報を渡すために使用します。キー定義ブロックは、次に示す 3 つのデータ領域で構成されます。

- グローバル情報領域
- キー定義領域
- 構成要素定義領域

#### 7.2.4.1 グローバル情報領域

グローバル情報領域には、キー定義領域のサイズとファイルに含むキーの数を記述します。

グローバル情報領域のレイアウトについては、オンライン ヘルプ ファイルで詳しく説明されています。ヘルプ ファイルの索引で [グローバル情報領域] を参照してください。

#### 7.2.4.2 キー定義領域

キー定義領域には、索引ファイルで使用するキーを記述します。

キー定義領域は、グローバル情報領域の次に配置され、ファイルにある各キーにつきキー定義を 1 つ記述します。

すべてのキーは、構成要素より前に定義する必要があります。

キー位置の順序に基づいてキーを識別するため、キーを定義する順序は重要です。 例えば、主キーと 2 つの副キーを持つ索引ファイルがある場合、キー定義領域には 3 つのキー定義を記述します。この場合、主キーをキー 0、最初の副キーをキー 1、2 番目の副キーをキー 2 とします。

キー定義領域のレイアウトについては、オンライン ヘルプで詳しく説明されています。 ヘルプ ファイルの索引で [キー定義領域] を参照してください。

#### 7.2.4.3 構成要素定義領域

構成要素定義領域は、キー定義領域の後にあります。

この領域では、各キーの構成要素を定義します。分割キー以外の各キーは、1 つの構成要素から成り立ち、各構成要素をそれぞれ定義する必要があります。

構成要素の定義では、キー構成要素の位置と長さを指定します。

数字キーの場合、構成要素定義領域を使用して、数値の型を指定することができます。その後で、IXNUMKEY コンパイラ指令を使用すると、数値の型にしたがってキーを順序付けることができます (デフォルトでは、キーは英数字順に並びます)。

構成要素定義領域のレイアウトについては、オンライン ヘルプ ファイルで詳しく説明されています。ヘルプ ファイルの索引で [*構成要素定義領域*] を参照してください。

## 7.3 FCD へのアクセス

プログラムで通常の COBOL 構文を使用してファイルの入出力操作を行う場合、各ファイルに対してファイル制御記述 (FCD) が自動的に作成されます。この自動的に作成された FCD にアクセスするには、FCDREG コンパイラ指令でプログラムをコンパイルします。これにより、FCD の読み取りや変更、FCD を使用したファイル ハンドラの明示的な呼び出しなどが可能になります。

FCD にアクセスするには、プログラムの連絡節に FCD 定義を設定する必要があります。FCD 定義の例は、NetExpress インストール ディレクトリの ¥source ディレクトリに格納されたファイル xfhfcd.cpy に保存されています。プログラムで次の SET 文を使用すると、読み取りや変更を行う FCD にこの定義をマップすることができます。

```
set address of fcd to address of fh--fcd of file
```

この場合、パラメータの内容は次のとおりです。

*fcd* プログラムの連絡節に記述する FCD 定義の名前。

fh--fcd (ハイフンが 2 つ必要なことに注意してください。) ファイルにアクセスするために、この COBOL システムで使用する FCD。

*file* ファイルの FD 名。

この SET 操作に続けて、連絡部で設定した FCD のデータ項目にアクセスすると、ファイルの FCD へアクセスすることができます。

同様に、次のように指定すると、キー定義ブロックにアクセスすることができます。

```
set address of kdb to address of fh--keydef of file
```

*kdb* プログラムの連絡節で記述するキー定義ブロックの名前。

fh--keydef (ハイフンが 2 つ必要なことに注意してください。) ファイルにアクセスするために、この COBOL システムで使用するキー定義。

*file* ファイルの FD 名。

FCD のアクセス例は、オンライン ヘルプ ファイルで説明されています。ヘルプ ファイルの索引で [*FCD*] を参照してください。

## 7.4 オペレーション コード

ファイル ハンドラが実行するファイル操作は、2 バイトのオペレーション コードで識別します。

オペレーション コードには、標準と特殊の 2 種類があります。

- 標準オペレーション コード

  標準オペレーション コードでは、MSB に x"FA" が格納されます。LSB は、具体的な操作を示します。

  標準オペレーション コードの詳細なリストは、オンライン ヘルプ ファイルに記載されています。ヘルプ ファイルの索引で [*ファイル ハンドラ*] を参照してください。

- 特殊オペレーション コード

  特殊オペレーション コードでは、MSB に x"00" が格納されます。LSB は、特殊操作を示します。

  特殊オペレーション コードの詳細なリストは、オンライン ヘルプ ファイルに記載されています。ヘルプ ファイルの索引で [*ファイル ハンドラ*] を参照してください。

## 7.5 相対バイト アドレス指定

レコードの相対バイト アドレスは、特定のレコードに対するすべてのファイル ハンドラの操作で FCD のオフセット 72 に配置されます (またはオフセット 93 にビット 4 を設定した場合はオフセット 13)。レコードの相対バイト アドレスを使用するには、READ 操作の後でこのフィールドの内容を保存するだけです。

相対バイト アドレスを使用すると、高速でレコードへアクセスできるようになります。ただし、次のような制限があります。

- 相対バイト アドレス操作は、現在のレコード ポインタに影響しません。そのため、相対バイト アドレス操作の後の READ (順) 操作では、通常のアクセス方法で最後にアクセスしたレコードの次のレコードを返します。相対バイト アドレス指定を使用してアクセスしたレコードの次のレコードを返すわけではありません。 ただし、相対バイト アドレス操作を実行するためのファイル ハンドラを呼び出す前に、FCD の構成フラグ (オフセット 93) にビット 5 を設定すると、現在のレコードポインタを更新し、相対バイト アドレス指定を使用してアクセスしたレコードを示すことができます。
- レコードの再書き込みや削除が行われると、相対バイト アドレス操作は索引を更新します。
- レコードの相対バイト アドレスの取得後にそのレコードを削除すると、その相対バイト アドレスを使用した読み書きは通常失敗します。レコードを別のレコードに置き換えることは可能ですが、このレコードを検出できないことがあります。

相対バイト アドレス操作では、レコード ロックもサポートされています。

レコードの相対バイト アドレスを取得すると、それを使用して次の操作を実行することができます。

- レコードの読み取り
- レコードの再書き込み
- レコードの削除

### 7.5.1 レコードの読み取り

相対バイト アドレスを使用してファイルから具体的なレコードを読み取る方法には 2 種類あります。

- FCD の相対バイト アドレス フィールドにアドレスを入力し、FCD の構成フラグ (オフセット93) にビット 6 を設定します。現在のレコード ポインタを更新する場合、構成フラグ (オフセット93) にビット 5 を設定してください。
- READ (ランダム) WITH NO LOCK、READ (ランダム) WITH LOCK、READ (ランダム) WITH KEPT LOCK、または READ (ランダム) 操作をファイルに実行します。

READ (直接) 操作も同様に機能しますが、FCD にビットを設定する必要はありません。READ (直接) 操作では、常

に FCD の相対バイト アドレス フィールドで指定されたアドレスのレコードを返し、現在のレコード ポインタを更新します。

上記のどちらの方法でも、現在の参照キーを別のキーに切り替えることができます。例えば、主キーにより READ (順) 操作を行っている場合に、次の手順を使用すると、現在のレコードの最初の副キーにより読み取りを開始することができます。

1. ファイルの次のレコードを読み取ります (このレコードのアドレスは、FCD の相対バイト アドレス フィールドで指定します)。
2. FCD の参照キー フィールド (オフセット52) に新しい参照キーを入力します。
3. READ (直接) 操作を実行して、このアドレスのレコードを返します。これで、参照キーは新しいキーに変更されます。
4. ファイルの次のレコードを読み取ります。このレコードは、新しい参照キーの索引にある次のレコードです。

### 7.5.2 レコードの再書き込み

FCD の相対バイト アドレス フィールドにアドレスを入力し、FCD の構成フラグ (オフセット93) にビット 6 を設定すると、レコードを特定のアドレスに再度書き込むことができます。

現在のレコード ポインタを更新する場合は、FCD の構成フラグ (オフセット93) にビット 5 を設定します。

次に、REWRITE 操作を実行します。

### 7.5.3 レコードの削除

FCD の相対バイト アドレス フィールドにアドレスを入力し、FCD の構成フラグ (オフセット93) にビット 6 を設定すると、特定のアドレスのレコードを削除することができます。

次に DELETE 操作を実行します。

## 7.6 カスタム ファイル ハンドラの作成

ユーザー独自のカスタマイズされたファイル ハンドラを作成することができます。

プログラムでファイル ハンドラを使用して COBOL 入出力構文を処理するには、CALLFH コンパイラ指令を使用します。例えば、USERFH というファイル ハンドラを使用する場合、次のようなコンパイラ指令でプログラムをコンパイルします。

```
CALLFH"USERFH"
```

これにより、すべての COBOL 入出力操作が USERFH の呼び出しにコンパイルされます。

---

注意: CBL_EXIT_PROC を使用する場合、ファイル ハンドラを呼び出しているアプリケーションの終了時に、ファイル ハンドラが適切にシャットダウンされるように、優先順位を 200 に設定する必要があります。

---

## 7.7 新しい索引ファイルの作成

特殊オペレーション コードを使用すると、次のように、既存の索引ファイルのデータ ファイル部から索引ファイルを再作成することができます。

1. *Create Index File* (特殊オペレーション コード x"07") を使用して、新しい索引ファイルを作成します。
2. *Get Next Record* (特殊オペレーション コード x"08") を使用して、データ ファイルから各レコードを読み取ります。
3. レコードの各キーの値については、*Add Key Value* (特殊オペレーション コード x"09") を使用して索引ファイルにキー値を追加します。

既存のデータ ファイルから新しい索引ファイルを作成する方法の例は、オンライン ヘルプに記載されています。ヘルプ ファイルの索引で [*ファイル ハンドラ*] を参照してください。

## 7.8 圧縮ルーチン

ファイル ハンドラがデータを圧縮するために使用する圧縮ルーチンは、スタンドアロン モジュールです。 そのため、次のように使用することができます。

- ユーザー アプリケーションで圧縮ルーチンを使用する。
- ファイル ハンドラでユーザーの圧縮ルーチンを使用する。

Micro Focus の圧縮ルーチンとユーザー独自の圧縮ルーチンは、どちらも最大 127 個まで使用できます。

### 7.8.1 Micro Focus 圧縮ルーチン

Micro Focus の圧縮ルーチンは CBLDC*nnn* というモジュールに格納されています。*nnn* は 001 から 127 までの数字です。

Micro Focus の圧縮ルーチンを使用するには、FCD のオフセット 78 のバイトを 001 から 127 までの値に設定します。

#### 7.8.1.1 CBLDC001

Micro Focus の圧縮ルーチン CBLDC001 では、ランレングス エンコーディング (RLC) 形式を使用します。RLC は、

同じ文字の文字列 (実行) を検出し、その文字列を文字の識別子、カウントまたは 1 つのオカレンスとして減らす圧縮方法です。

---

注意: 2 バイト文字 (2 バイトの空白文字を含む) は圧縮されないため、このルーチンは 2 バイト文字のオカレンスを格納するファイルでは無効です。

---

CBLDC001 は、特に空白文字、バイナリのゼロと文字のゼロ (1 文字に減らすことができる)、印刷可能文字 (カウント数とその後に続く反復文字の計 2 文字に減らすことができる) に対して実行することを目的としています。

圧縮ファイルのバイトは、次のような意味を持っています (16 進値が示されます)。

| | |
|---|---|
| 20-7F | ほとんどの印刷可能な文字です。通常の ASCII 文字です。 |
| 80-9F | それぞれ 1 文字から 32 文字までの空白文字。 |
| A0-BF | それぞれ 1 文字から 32 文字までのバイナリ ゼロ。 |
| C0-DF | それぞれ 1 文字から 32 文字までの文字のゼロ。 |
| E0-FF | 後に続く文字の 1 回から 32 回までのオカレンス数。 |
| 00-1F | 後に続く文字の 1 回から 32 回までのオカレンス数。圧縮コードとしてではなく、文字どおり解釈します。元のデータに 00-1F、80-9F、A0-BF、C0-DF、E0-FF のいずれかの範囲の文字が存在するときに使用されます。(このような文字の 1 文字は 2 バイトに拡張されます。それ以外の場合、圧縮により発生する弊害はありません。) |

### 7.8.1.2 CBLDC003

CBLDC001 と同様、このルーチンではランレングス エンコーディング (RLC) を使用しますが、1 バイトと 2 バイトの文字列 (実行) を検出します。このルーチンは DBCS 文字に適していますが、CBLDC001 の代わりに使用することもできます。

圧縮形式は、ヘッダー バイト 2 個の後に 1 つまたは複数の文字が続きます。ヘッダー バイトのビットの内容は次のとおりです。

| | |
|---|---|
| ビット 15 | 設定解除 - 1 文字 |
| ビット 14 | 設定 - 圧縮シーケンス<br>設定解除 - 非圧縮シーケンス |

| | |
|---|---|
| ビット 0-13 | 圧縮した文字、または圧縮しない文字のカウント |

文字列の長さはヘッダーのビットによって異なります。

| | |
|---|---|
| ビット 14 および 15 を設定した場合 | 2 つの反復文字 |
| ビット 14 のみを設定した場合 | 1 つの反復文字 |
| その他 | 1 文字から 63 文字までの非圧縮文字 |

### 7.8.1.3 Micro Focus の圧縮ルーチンの呼び出し

データ ファイルを圧縮する場合、ファイル ハンドラは DATACOMPRESS コンパイラ指令で指定する圧縮ルーチンを呼び出します。

データを圧縮するために圧縮ルーチンを使用する場合、プログラムから直接呼び出すことができます。

```
call "CBLDCnnn" using input-buffer,
                      input-buffer-size,
                      output-buffer,
                      output-buffer-size,
                      compression-type
```

この場合、パラメータの内容は次のとおりです。

| | |
|---|---|
| *nnn* | 001 から 127 までのデータ圧縮ルーチン。 |
| *input-buffer* | PIC X データ項目。圧縮または圧縮解除するデータ。最大サイズは 65,535 バイト。 |
| *input-buffer-size* | 2 バイトの (PIC XX COMP-5) データ項目。*input-buffer* のデータの長さを指定します。 |
| *output-buffer* | PIC X データ項目。結果データを格納するバッファ。 |
| *output-buffer-size* | 2 バイト (PIC XX COMP-5) のデータ項目。ルーチンを起動するときに、このデータ項目には使用可能な出力バッファのサイズが格納されます。ルーチンを終了するときには、*output-buffer* のデータ長を格納します。 |
| *compression-type* | 1 バイトの (PIC X) データ項目。入力データを圧縮するか、圧縮解除するかを指定します。<br>0 - 圧縮<br>1 - 圧縮解除 |

RETURN-CODE 特殊レジスタは、操作が成功したかどうかを示します。圧縮または圧縮解除は、出力バッファが小さく結果を受け付けられない場合にだけ失敗します。

### 7.8.2 ユーザー独自の圧縮ルーチン

ユーザー独自の圧縮ルーチンは、USRD*nnn* というモジュールに格納します。*nnn* は 128 から 255 までの数字です。

圧縮ルーチンを作成する場合、次のことに注意する必要があります。

- 圧縮ルーチンでファイル ハンドラを呼び出さないでください。無限ループが発生する可能性があります。
- 一旦ファイルに対してデータ圧縮を有効化すると、そのファイルには常に同じ種類の圧縮を指定する必要があります。同じ種類の圧縮を指定しない場合、ファイルを開くときに、ランタイム システム エラーを受け取ります。
- データ圧縮は索引ファイルまたはレコード順ファイル以外のファイルには影響しません。

# 第8章 Btrieve

Btrieve は Pervasive Software 社が提供するファイル処理システムです。Btrieve の詳細については、Pervasive Software 社からの Btrieve マニュアルを参照してください。

Btrieve ファイルを処理するために、Micro Focus ファイル ハンドラ API を使用する Xfh2btr 呼出し変換モジュールが NetExpress で提供されています。

---

注記:

- Btrieve ランタイムシステムは、NetExpress では提供されていません。Btrieve ランタイム システムのコピー、または Btrieve 開発者キットを既に持っている必要があります。
- Pervasive Software 社とのライセンス契約を持っていない場合は、サードパーティに Btrieve ランタイムを配布することはできません。詳細は Pervasive Software 社にお問い合わせください。

---

## 8.1 Xfh2btr 呼出し変換モジュール

Xfh2btr 呼出し変換モジュールを使用すると、この COBOL システムからBtrieve ファイルをアクセスするための Micro Focus ファイルハンドラ API を使用できます。

Btrieve は通常 ANSI 規格に準拠しませんが、デフォルトでは Xfh2btr 呼出し変換モジュールが Btrieve ランタイムに必要な呼出しを行い、ANSI 動作にエミュレートするようにさせます。非 ANSI モードで操作することもできます。

Xfh2btr は Btrieve 6.1*x* 以降と互換性があります。

Xfh2btr は COBOL システムの一部として提供されます。これは、リンク可能なモジュール xfh2btr.obj としても提供され、Btrieve ファイル上で I/O を行うどのアプリケーションともリンクできます。

呼出し変換モジュールが I/O を行うために Btrieve ランタイム システムを呼出す必要がある時は、モジュール _BTRV が呼出されます。このモジュールの機能は、呼出しをフォーマットし、Btrieve マイクロカーネル インターフェイス (Windows NT および Windows 95 用の wbtrv32.dll) に呼出しを渡すことです。モジュール _BTRV は、Btrieve API 呼出しで使用されるのと同じパラメータを使って COBOL プログラムから直接呼び出せます(詳細は *Btrieve プログラマーズ参照マニュアル*を参照してください)。

スタンドアロンの実行ファイルにアプリケーションをリンクする場合、_btrv.obj にリンクする必要があります。または Btrieve 開発者キットを持っている場合は、Pervasive Software 社から提供されるモジュールを _btrv.obj の代わりにリンクできます。これは cobrbtrv.obj と cobpbtrv.obj です。これらのモジュールの詳細については、Btrieve 開

発者キットで提供されるマニュアルを参照してください。

## 8.2 Xfh2btr の呼び出し

Xfh2btr 呼出し変換モジュールを呼出すには、次の３つの方法があります。

- CALLFHコンパイラ指令を使用する
- FILETYPEコンパイラ指令を使用する
- 外部ファイル名割当てを使用する

Btrieve では、サポートしないファイル操作とファイル形式がいくつかあるので注意が必要です。例えば、WRITE AFTER や行順ファイルなどはサポートされていません。このような時にコンバータにファイル I/O を直接行おうとすると、プログラムにエラーが返されます。

### 8.2.1 CALLFH コンパイラ指令

CALLFH コンパイラ指令は、プログラムによって使用されるファイルハンドラにすべての COBOL I/O 操作を実行することを指定します。デフォルトでは、ファイルハンドラを指定しない場合は Micro Focus ファイルハンドラが使用されます。

CALLFH コンパイラ指令を Xfh2btr 呼出し変換モジュールを指定するために使用する場合は、プログラムはすべての I/O を Xfh2btr 呼出し変換モジュールに渡して、COBOL I/O 操作を Btrieve I/O 操作に変換します。

次のようにして、ソースプログラムの先頭に最初の $SET 文として CALLFH を置きます。

```
$set callfh"xfh2btr"
```

### 8.2.2 FILETYPE コンパイラ指令

FILETYPE コンパイラ指令は、プログラムで作成されたファイルの形式を指定します。

次のようにして、プログラムの先頭に最初の $SET文 として FILETYPE コンパイラ指令を置きます。

```
$set filetype"n"
```

この場合*n*は 5 または 6 です。

FILETYPE "5" は、ファイル操作が ANSI 規格に準拠することが要求される場合に使用します。Btrieve は通常 ANSI 規格に準拠しないため、この操作モードでは Btrieve ランタイム システムに何度も呼出しを行って、ANSI 動作にエミュレートするようにさせることが必要です。

FILETYPE "6" は、速さが要求される場合に使用します。このモードでは、各 Micro Focus ファイル ハンドラ操作が最も近い Btrieve ランタイム呼出しにマップされます。ANSI 規格への準拠は行われません。

異なった形式のファイルをプログラム中に混在させることができます。次の例のように、FILETYPE コンパイラ指令を個々の SELECT 文の周りに置きます。

```
$set filetype"0"
select testfile-1 assign to "test-1.dat"
   organization indexed
   record key prime-key
   access sequential.
$set filetype"5"
select testfile-2 assign to "test-2.dat"
   organization relative
   access sequential.
```

上の例では、test-1.dat 上のすべての I/O は Micro Focusファイル ハンドラによって処理され、test-2.dat 上のすべての I/O はXfh2btr 呼出し変換モジュールによって処理されます。

---

注記: 索引ファイルを作成する時に使用する形式を指定するには、ファイル ハンドラ構成ファイルにあるファイル設定の下の IDXFORMAT パラメータを置くことができます。 ANSI 準拠または非 ANSI 準拠で Btrieve を示すには、 IDXFORMAT を 5 または 6 のいずれかに設定します。詳細については、*ファイル処理のマニュアル*にあるファイル ハンドラ構成を参照してください。

---

## 8.3 Btrieve 環境変数

Xfh2btr 呼出し変換モジュールの操作を切り替えるために使用する、2つの環境変数 BTRPAGE と BTRMAXREC があります。

### 8.3.1 BTRPAGE

BTRPAGE 環境変数には、Btrieve ファイルが作成される時に使用されるページサイズを指定します。次のコマンドを使用して設定できます。

```
set BTRPAGE=nnn
```

ここで *nnn* は、512 から 4096 の範囲にある 512 バイトの倍数とします。

BTRPAGE を設定しない場合、または不正な値を記述した場合、ページ サイズのデフォルトは 1024 バイトになります。

---

注記:

- ページ サイズは、ファイルに定義できるキー数や、レコードに置くキーの場所に影響を与えます。
- Btrieve Record Manager には、正しいページサイズをロードする必要があります。

BTRPAGE の値は、xfh2btr 構成ファイルを使用してファイルごとに設定できます。

---

### 8.3.2 BTRMAXREC

BTRMAXREC 環境変数には、最大レコードサイズを指定します。

BTRMAXREC は、次のコマンドを使用して設定できます。

`set BTRMAXREC=`*`nnnn`*

ここで *nnnn* は最大レコード長を指定するバイト値です。

BTRMAXREC には、アクセスされる最大レコードのレコード サイズを指定します。

BTRMAXREC を設定しない場合は、デフォルト値の 32K バイトが使用されます。

BTRMAXREC の値は、xfh2btr 構成ファイルを使用してファイルごとに設定できます。

## 8.4 Xfh2btr 構成ファイル

---

注記: Xfh2btr 構成ファイルを使用したい場合は、mfini.obj を使ってプログラムにリンクする必要があります。

---

Xfh2btr 構成ファイルを使用すると、すべてのファイルまたは個々のファイルに対して、ページサイズ、最大レコード長、Btrieve オープン モードを指定できます。

構成ファイルの名前は、XFH2BTR 環境変数で指定できます。この環境変数が設定されていない場合、デフォルトの構成ファイル名 xfh2btr.cfg が使用されます。構成ファイルはまず現行ディレクトリで検索され、次に COBDIR 環境変数で指定されたパスに沿って検索されます。

タグ [X2B-DEFAULTS] は、ページ サイズのデフォルト値、最大レコード長、Btrieve オープン モードを指定するのに使用されます。一方、ファイル名 (例えば [test.dat]) を含むタグは、特定のファイルに値を指定するのに使用されます。

- ページサイズ
- 最大レコード長
- Btrieve オープン モード

属性名はスペースまたはコロン (:) によって、属性値と分けることができます。

---

注記: 構成ファイルがある場合、Btrieve 環境変数 BTRPAGE および BTRMAXREC は無視されます。

---

また、Xfh2btr トレース オプションをオンにして、Xfh2btr 構成ファイルでトレースファイルの名前を指定することができます。

### 8.4.1 ページサイズ

Btrieve ファイルの作成時に使用するページ サイズは、Xfh2btr 構成ファイル中で次のように属性を設定することで指定できます。

BTRPAGE:*nnn*

ここで *nnn* は、512 から 4096 の範囲にある 512 バイトの倍数とします。

BTRPAGE を設定しない場合、または不正な値を指定した場合、ページ サイズのデフォルトは 1024 バイトになります。

### 8.4.2 最大レコードサイズ

Btrieve ファイル用の最大レコード サイズは、Xfh2btr 構成ファイル中で次のように属性を設定することで指定できます。

BTRMAXREC:*nnnn*

ここで *nnnn* はバイト値です。BTRMAXREC には、アクセスされる最大レコードのレコードサイズを設定します。BTRMAXREC を設定しない場合、デフォルト値 32K バイトが使用されます。

### 8.4.3 Btrieve ファイルオープンモード

Xfh2btr 構成ファイルは、ファイルを開く時に使う Btrieve オープンモードを指定するのに使用できます。

INPUT-EXCLUSIVE:*オープンモード*

INPUT-SHAREABLE:*オープンモード*

OUTPUT:*オープンモード*

EXTEND-EXCLUSIVE:*オープンモード*

EXTEND-SHAREABLE:*オープンモード*

I-O-EXCLUSIVE:*オープンモード*

I-O-SHAREABLE:*オープンモード*

*オープンモード*の値として指定できる値は、normal (通常)、accelerated (加速)、read-only (読み取り専用)、verify (確認)、exclusive (排他)です。これらのオープンモードの詳細については、Btrieve のマニュアルを参照してください。

### 8.4.4 Xfh2btr 構成ファイルの例

```
[X2B-DEFAULTS]
output:accelerated
i-o-shareable:accelerated
i-o-exclusive:accelerated
btrpage:2048
btrmaxrec:2000
[test1.dat]
output:exclusive
i-o-shareable:normal
btrpage:4096
```

上記のファイルは Xfh2btr 構成ファイルの例で、次のことを指定しています。

- OUTPUT で開かれたファイルに対するデフォルトのオープンモードは accelerated です。
- 共有 I/O で開かれたファイルに対するデフォルトのオープンモードは accelerated です。
- 排他 I/O で開かれたファイルに対するデフォルトのオープンモードは accelerated です。
- ファイル作成時に使用されるデフォルトのページ サイズは 2048 バイトです。
- デフォルトの最大レコード長は 2000 バイトです。
- ファイル test1.dat では、OPEN OUTPUT 操作はファイルを排他で開きます。
- ファイル test1.dat では、共有の OPEN I/O 操作は通常のオープンモードを使用します。
- ファイル test1.dat では、ファイル作成時に使用されるページサイズは 4096 バイトです。

### 8.4.5 トレースオプション

Xfh2btr トレース オプションを設定すると、Xfh2btr 呼出し変換モジュールによって実行される各操作および Btrieve Record Manager によって戻される各エラー コードを、画面に表示するかまたはトレース ファイルに記録できます。

Btrieve Record Manager によってエラーが戻された場合、Xfh2btr 呼出し変換モジュールは Btrieve エラーを、プログラムに戻す COBOL ファイル状態コードにマップします。Btrieve エラーがマップされた COBOL 状態コードも、トレースファイルに表示されます。

トレースで Btrieve エラーが示されないのに COBOL ファイル状態コードが戻された場合、このエラーは Xfh2btr 呼出し変換モジュール内で生成されます。例えば、入力用として開かれたファイルに書き込もうとすると、このようなエラーが生成されます。

トレース オプションに影響を与える属性として、TRACE-FILE と TRACE があります。

TRACE-FILE:*ファイル名*

これは、トレース情報を画面ではなくファイルに行くように指定します。ファイル名が記述されない場合は、デフォルトのファイル名 xfh2btr.lst が使用されます。トレース ファイルの属性は、タグ [X2B-DEFAULTS] の下でのみ指定できます。クライアント アプリケーションのいくつかが同じトレース ファイルにトレース情報を書き込んでいる場合、ファイルにこの情報が現れる順序は特定できません。

---

注記: トレース ファイル名の記述は、TRACE 機能をオンにしません。

---

TRACE

TRACE 属性はトレースをオンにします。これはタグ [X2B-DEFAULTS] の下に指定するか (この場合すべてのファイル上の操作がトレースされます)、またはファイル名のタグの下に指定します (この場合そのファイル上の操作だけがトレースされます)。

#### 8.4.5.1 トレースオプションの設定例

次の内容が Xfh2btr 構成ファイルにある場合:

```
[X2B-DEFAULTS]
trace-file:x2btrace.dat
trace:
```

各ファイル用でのトレースをオンにし、トレース情報をファイル x2btrace.dat に書き込みます。specify that tracing is on for every file and that the trace information is written to the file x2btrace.dat.

次の内容が Xfh2btr 構成ファイルにある場合:The following entries in the Xfh2btr configuration file:

```
[X2B-DEFAULTS]
trace-file:
[test1.dat]
trace:
```

ファイル test1.dat のトレースを指定し、トレース情報をファイル xfh2btr.lst に書き込みます (デフォルトのトレース ファイル名)。

### 8.4.5.2 トレース出力の例

以下に、トレース オプションが使用された時に生成されるトレース出力の例を示します。

```
                                          行
Input FA01 ifile1.tmp                     1
Output Btrieve Error=+0000 COBOL Error=0/0  2
Input FAF3 ifile1.tmp                     3
Output Btrieve Error=+0000 COBOL Error=0/0  4
Input FAF3 ifile1.tmp                     5
Output Btrieve Error=+0000 COBOL Error=0/0  6
Input FADC All Files                      7
Output Btrieve Error=+0000 COBOL Error=0/0  8
Input FA80 ifile1.tmp                     9
Output Btrieve Error=+0000 COBOL Error=0/0  10
Input FA02 ifile1.tmp                     11
Output Btrieve Error=+0000 COBOL Error=0/0  12
Input FAF5 ifile1.tmp                     13
Output Btrieve Error=+0000 COBOL Error=0/0  14
Input FAF5 ifile1.tmp                     15
Output Btrieve Error=+0000 COBOL Error=0/0  16
Input FAF5 ifile1.tmp                     17
Output Btrieve Error=+0009 COBOL Error=1/0  18
Input FA80 ifile1.tmp                     19
Output Btrieve Error=+0000 COBOL Error=0/0  20
```

```
Input FA01 test.dat                              21
Output Btrieve Error=+0029 COBOL Error=9/078     22
Input FA00 ifile1.tmp                            23
Output Btrieve Error=+0000 COBOL Error=0/0       24
Input FAF3 ifile1.tmp                            25
Output Btrieve Error=+0000 COBOL Error=4/8       26
Input FA80 ifile1.tmp                            27
Output Btrieve Error=+0000 COBOL Error=0/0       28
```

行 1 は入力行の例で、操作コードおよび操作が実行されているファイルの名前です。

行 2 は出力行の例で、Btrieve エラー状態および COBOL アプリケーションに戻されているエラーです。

行 7 は、すべてのオープン ファイルで行われる操作の例です (この場合 COMMIT)。

行 22 は、Btrieve ランタイム システムから戻されているエラー状態 29 を示します。これは COBOL アプリケーションに状態 9/078 として戻されます。Btrieve エラー状態 29 の詳細については、関連する Btrieve のマニュアルを参照してください。

行 26 は、Xfh2btr モジュール内で生成されるエラーを示します。INPUT 用に開かれたファイルへの書き込みが行われています。

## 8.5 Micro Focus ファイルハンドラと Xfh2btr の違い

Micro Focus ファイル ハンドラによって行われる COBOL ファイル操作の大部分は、Btrieve ファイル上で Xfh2btr によって行われるのと同じ方法で実行されます。ただし 2 つのシステムには違いがあるため、動作が異なる場合があります。動作が異なるものを次に示します。

- キー
- ロック レコードの検出
- OPEN OUTPUT 操作
- レコード長
- 現行レコード ポインタ (CRP)
- トランザクション処理
- 書込みロック(WRITELOCK)

注記: ファイル内でレコードの書込み、再書込み、削除をした後で順読み操作を行う時は、現行レコード ポインタ (CRP) の位置付けには特に注意してください。

### 8.5.1 キー

- Btrieve v5.x 以前を使用する時、ファイルごとのキー コンポーネントの総数 (通常のキーは 1 つのコンポーネントを持ち、分割キーは 1 つ以上を持ちます) は 24 を超えられません。このため、24 コンポーネントに 1 キー、1 コンポーネントに 24 キー、またはこれらの組合せで行います。
- Btrieve v6.x 以降を使用する時、キー コンポーネントの最大数はファイルのページ サイズによって変わります。ファイルのページ サイズは Btrieve 環境変数 BTRPAGE または Xfh2btr 構成ファイルで指定できます。4096 バイトのページ サイズを持つファイルでは、キー コンポーネントの最大数は 119 です。詳細については、Btrieve のマニュアルを参照してください。
- Xfh2btr モジュールは、ファイル中のキー コンポーネントの数を確認しません。使用する Btrieve のバージョンが必要なキー コンポーネントの数をサポートしていない場合は、Btrieve ランタイム システムからエラーが戻され、アプリケーション プログラムに報告されます。
- Btrieve ファイル中のキー長の合計は、255 バイトを超えることができません。
- Btrieve ファイル中のすべてのキーは、可変長ファイルの固定部分内に完全に含まれる必要があります。
- Btrieve を使用する時、一部のキーで START 操作を実行できません。全体のキーを使用する必要があります。

### 8.5.2 ロックレコードの検出

この COBOL システムが他のユーザによってロックされているレコードを読むと、レコード データが COBOL ファイル状態 9/068 (レコード ロック) と共にプログラムに戻されます。

Btrieve では、レコードがロックされている場合はデータが返されません。このため、ロックされたレコードの読み取りは Btrieve ランタイムへの複数の呼出しを必要とし、デフォルト レコードの読み取りは遅くなります。

読み取り操作の速度を向上させるには、次のような方法があります。

- 排他ロックでファイルを開く。

  これを行うと、レコード ロック状態を検出するための Btrieve ランタイムへの呼出しは不要です。これはファイルが排他でロックされるので、個々のレコードを確認する必要がないためです。

- NODETECTLOCK コンパイラ指令を使用する。

この指令を設定すると、レコードが他のユーザによってロックされているかを確認するための Btrieve ランタイムへの呼出しが行われません。

- トランザクション中のファイルにアクセスする。

  トランザクションの中から Btrieve ファイルをアクセスすると、ファイルに一時的に排他ロックがかかるため、レコード ロックを検出するための Btrieve ランタイムへの呼出しは不要です。

### 8.5.3 OPEN OUTPUT 操作

COBOL OPEN OUTPUT 操作は、ファイルを作成し、そのファイルを排他で開きます。これは、単一のオペレーティング システム呼出しで行われます。Btrieve では、この操作は Btrieve ランタイムへの 2 つの呼出しを必要とします。はじめの呼出しでファイルを作成し、2 つ目の呼出しでファイルを排他ロックで開きます。これらの 2 つの呼出しの間には多少の時間ロスがあるため、2 番目のユーザはファイルが作成され開かれるまでの間にファイルにアクセスできてしまいます。これが起こると、2 番目のユーザは空のファイルを見つけてファイルを読み取ろうとし、この結果 "終了" エラーになります。OPEN OUTPUT を行うユーザは、排他 OPEN 呼出しが失敗するため、"ファイルロック" エラーを受け取ります。

### 8.5.4 レコード長

Btrieve ファイルでは、可変長レコードは 2 つの部分、固定長部分と可変長部分から構成されます。固定長レコードは、固定長部分だけから構成されます。

可変長レコードでは、レコードの固定長部分は通常、COBOL プログラムで定義した最小レコード長によって決まります。しかし、重複を許すキーの数と使用されるページサイズによって、Btrieve はレコードの固定部分の最大レコード長に制限を課します。レコードの固定部分の最大長の決定方法は次の通りです。

p - 6 - ( 8 * *k* ) [- 4 可変長レコードの場合] [- 4 相対ファイルの場合]

ここで:

*p*ファイルのページサイズ

*k* 重複を許すキーの数

---

注記:

- Btrieve レコードの固定部分で許された最大サイズを超えるレコード長で固定長レコードの Btrieve ファイルを作成しようとすると、ファイルは可変長 Btrieve ファイルとして作成され、その固定部分には最大サイズが設定されます。

- 固定部分で許された最大サイズを超える最小のレコード長で可変長レコードの Btrieve ファイルを作成しよ

うとすると、固定部分には最大サイズが設定されたファイルが作成されます。

- Btrieve は、Btrieve レコードの固定部分内で定義されたすべてのキーを必要とします。
- 相対ファイルは、各レコードの前に４バイトの自動増加キーを付けた Btrieve キー ファイルとして作成されます。このキーの処理はユーザ プログラムには見えません。しかし実際の Btrieve ファイルは、上記の方法で計算したよりも 4 バイト大きい固定レコード長を持っています。

---

### 8.5.4.1 Btrieve レコード長の例

次に示す例は、1024 バイトのページ サイズを使用します。

- 1018 バイトのレコード長を持つ固定長レコード索引ファイルは、1018 バイトの固定レコード長を持つ固定長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1018 バイトにある必要があります。

- 1019 バイトのレコード長を持つ固定長レコード索引ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1014 バイトにある必要があります。

- 1010 バイトのレコード長で重複を許す 1 つのキーを持つ固定長レコード索引ファイルは、1010 バイトの固定レコード長を持つ固定長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1010 バイトにある必要があります。All of the keys must be in the first 1010 bytes of the record.

- 1011バイトのレコード長で重複を許す 1 つのキーを持つ固定長レコード索引ファイルは、1006 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1006 バイトにある必要があります。

- 1014 バイトのレコード長を持つ固定長レコード相対ファイルは、1018 バイトの固定レコード長を持つ固定長レコード Btrieve ファイルを作成します。

  追加の 4 バイトは、自動増加キーによって各レコードの前に自動的に追加されたものです。Micro Focus ファイル ハンドラ呼出しインタフェース用にオプションコード 06 の動作コードを使用している場合、戻されるレコード長は 1014 バイトです。

- 1015 バイトのレコード長を持つ固定長レコード相対ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。これは、最大 1014 バイトに制限されているためです。

  この長さには、自動増加キーによって各レコードの前に自動時に追加された 4 バイトが含まれています。

Micro Focus ファイル ハンドラ呼出しインタフェース用にオプションコード 06 の動作コードを使用している場合、戻されるレコード長は 1010 バイトです。

- 1014 バイトの最小レコード長を持つ可変長レコード索引ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1014 バイトにある必要があります。

- 1015 バイトの最小レコード長を持つ可変長レコード索引ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1014 バイトにある必要があります。

- 1006 バイトの最小レコード長で重複を許す 1 つのキーを持つ可変長レコード索引ファイルは、1006 バイトの固定レコード長の可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1006 バイトにある必要があります。

- 1007 バイトの最小レコード長で重複を許す 1 つのキーを持つ可変長レコード索引ファイルは、1006 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  すべてのキーは、レコードの最初の 1006 バイトにある必要があります。

- 1010 バイトの最小レコード長を持つ可変長レコード相対ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  追加の 4 バイトは、自動増加キーによって各レコードの前に自動的に追加されたものです。Micro Focus ファイル ハンドラ呼出しインタフェース用にオプション コード 06 の動作コードを使用している場合、戻されるレコード長は 1010 バイトです。

- 1011 バイトの最小レコードを持つ可変長レコード相対ファイルは、1014 バイトの固定レコード長を持つ可変長レコード Btrieve ファイルを作成します。

  追加の 4 バイトは、自動増加キーによって各レコードの前に自動的に追加されたものです。Micro Focus ファイル ハンドラ呼出しインタフェース用にオプションコード 06 の動作コードを使用している場合、戻されるレコード長は 1010 バイトです。

### 8.5.5 現行レコードポインタ (CRP)

この COBOL システムでは、現行レコード ポインタ (CRP) は、ファイル中で START、READ (順次またはランダム)、OPEN 操作が行われる時にだけ影響を受けます。

しかし Btrieve ランタイム システムでは、CRP は WRITE、REWRITE、DELETE 操作によって影響を受けます。これはこれらの操作に続いて Btrieve ファイル中の CRP を元の位置に戻し、順次 READ 文が影響を受けないよう

にする必要があることを意味します。

共有環境では、CRP の再位置付けは、再位置付けを行うレコードが他のユーザによって削除されている場合には失敗します。このような場合、順次 READ を実行しようとするとエラーが戻されます。

続く節では、 CRP の再位置付けは、次の順次 READ 文の時ではなく操作が完了した後で行われます。これはレコードが削除される可能性を減らし、 CRP の再位置付けを成功させる可能性を増やします。

---

注記: ANSI 規格に準拠しないモードの操作では、CRP を再位置付けしないために非常に速くなります。

---

#### 8.5.5.1 WRITE 操作後の順次 READ 操作

I/O を開いた共有ファイルが ANSI 準拠の操作モードが使用されているトランザクションにない場合は、ファイルは実際に 2 回開かれます。最初にファイルが開かれるのは、ファイルからレコードを読み取る時です。2 回目にファイルが開かれるのは、ファイルにレコードを書き込む時です。このため、読み取り位置は通常の WRITE 操作で影響を受けることがありません。しかし WRITELOCK 指令を使用した場合、挿入したレコードはロックされて読み戻されなければならず、これによりファイル内の CRP が変更されます。

WRITE 操作後の Btrieve ファイル位置指示子の再位置付けは、WRITELOCK 指令が使用されファイルが共有の場合にのみ失敗します。WRITE 操作後に再位置付けが失敗した場合は、順次 READ が行われる時にエラーが戻されます。

#### 8.5.5.2 DELETE 操作後の順次 READ 操作

ランダムまたは動的アクセスで I/O を開いたファイルでは、Btrieve ファイル位置指示子を削除するレコードの位置に移動する必要があります。DELETE 操作を行った後、ファイル位置指示子は元の位置に戻されますが、再位置付けするレコードが削除されている場合はこれが失敗することもあります。

この問題は順次アクセスのファイルには発生しません。ファイル位置指示子は既に削除するレコード上にあり、再び位置付ける必要がないためです。

---

注記: 参照キーが重複を許すキーにある場合、順次 READ と DELETE 文を続けて行うことはできません。これは、DELETE 操作の後で CRP を再び位置付けることができないためです。

---

#### 8.5.5.3 REWRITE 操作後の順次 READ 操作

ランダムまたは動的アクセスで I/O を開いたファイルでは、Btrieve ファイル位置指示子は変更するレコードの位置

に移動されます。REWRITE 操作を行った後、ファイル位置指示子は元の位置に戻されますが、再位置付けするレコードが削除されている場合はこれが失敗することもあります。

この問題は順次アクセスのファイルには発生しません。ファイル位置指示子は既に再書き込みするレコード上にあり、再び位置付ける必要がないためです。

### 8.5.6 トランザクション処理

この COBOL システムでは、トランザクション処理は Fileshare を使用している場合にだけ実行できます。しかし Btrieve ユーザは、トランザクション処理を実行するのに Fileshare が必要ありません。

Fileshare を使ってトランザクション内のファイルにあるレコードを更新する時、ファイルにある個々のレコードはロックされ、他のユーザがレコードを読むことを防ぎます。これはレコードがファイルから削除された時に ROLLBACK 操作が行われる可能性があるためです。しかしトランザクション内から Xfh2btr を介して Btrieve ファイルにアクセスする場合、他のユーザが同時にファイルをアクセスすることを防ぐために、ファイル全体に一時的な排他ロックがかけられます。Btrieve v6.x 以降では、共有トランザクションが使用され、トランザクション内でファイルを共有させることができます。

Btrieve トランザクションは、処理中のトランザクションがなく ROLLBACK を使って開かれているファイルに対して、REWRITE 、WRITE、または DELETE 操作が行われる時に開始されます。 この時点では、Btrieve はファイルに排他ロックをかけます。このファイル ロックは共有トランザクションが使用中の場合を除いて、トランザクションが終了するまで残ります (Btrieve v6.x 以降)。

---

注記:

- Btrieve v6.x 以降では、共有 Btrieve トランザクションが使用されます。その後 Btrieve はページ レベルでロックを実行します (詳細はBtrieve のマニュアルを参照してください)。
- トランザクションは COMMIT または ROLLBACK 操作が実行された時に終了します。トランザクションは、ROLLBACK で開かれたファイル上で再び I/O が行われた時に再開始します。

---

CLOSE 操作は、トランザクションがアクティブでない時にのみ Btrieve CLOSE 操作が行われるため、トランザクションを再開始しません。

ファイル ロック エラーは、この COBOL システムがファイル ロック エラーを戻す時のみ、ファイルが開かれた時に Xfh2btr によって戻されます。 トランザクション内でファイルをアクセスする時に Btrieve がファイル ロック エラーを戻すことができますが、これはトランザクション内でファイルをアクセスするときに、この COBOL システムが正当に戻すことができる状態の Xfh2btr によってファイル ロック エラーに変換されます。

### 8.5.7 WRITELOCK

この COBOL システムでは、WRITELOCK 指令を使って、ファイルにレコードを書き込んだり、書き込まれたレコードをロックすることができます。ファイルに再度書き込まれたレコードにもロックをかけることができます。

Btrieve では、レコードは最初にファイルへ書き込みまたは再書き込みされ、その後レコードをロックするために読み戻されます。これは Btrieve ランタイムへ数々の呼出しを必要とし呼出しの間に若干時間がかかります。このため最初のユーザによってロックされたレコードが読み戻される前に、2 番目のユーザがレコードにアクセスして書き込まれたレコードをロックすることが可能です。

## 8.6 ANSI COBOLに準拠しないファイル操作

Btrieve は通常、ANSI 規格に準拠しません。しかしデフォルトでは、Xfh2btr 呼出し変換モジュールは、ANSI 動作をエミュレートさせるために Btrieve ランタイム システムに必要な呼出しを行います。ただし、FILETYPE "6" へ FILETYPE コンパイラ指令を設定することにより、非 ANSI モードでの操作を選択できます。

このモードの操作では、各 Micro Focus ファイル ハンドラの操作は最も近い Btrieve ランタイム呼出しにマップされ、ANSI 規格の確認が行われなくなります。

- この COBOL システム内で設定したランタイム スイッチは、Xfh2btr 呼出し変換モジュールによって無視されます。
- 操作は、NODETECTLOCK コンパイラ指令を設定したように行われます。
- ファイル中のレコードを削除、再書き込み、または書き込みした後は、現行レコード ポインタ (CRP) の再位置付けは行われません。
- 各 COBOL I/O 呼出しは、最も近くの関連する Btrieve ランタイム呼出しに、次のようにしてマップされます。

| COBOL I/O 呼出し | Btrieve ランタイム呼出し |
|---|---|
| OPEN | Open |
| CLOSE | Close |
| WRITE | Insert |
| READ | Get |
| START | Get |
| DELETE | Get/Delete |
| REWRITE | Get/Delete |

| | |
|---|---|
| UNLOCK | Unlock |
| ROLLBACK | Abort transaction |
| COMMIT | End transaction |

## 8.7 Btrieve エラー メッセージ

Btrieve エラー メッセージの完全なリストは、NetExpress オンライン ヘルプ ファイルに提供されています。*Btrieve* ヘルプ ファイルの索引の下を参照してください。

Btrieve Record Manager にエラーが戻された場合は、Xfh2btr 呼出し変換モジュールは Btrieve エラーを、プログラムに戻す COBOL ファイル状態コードにマップします。これらのマッピングは NetExpress オンライン ヘルプ ファイルに表示されています。*Btrieve* ヘルプ ファイルの索引の下を参照してください。

# 第9章 ファイルのソー ト

COBOL プログラムで SORT 文を実行する場合、デフォルトでは、ランタイム COBOL ソート モジュールを使用します。さらに、この COBOL システムには、ファイルのソートとマージを行う方法が他に 2 種類あります。

- Mfsort ユーティリティ

  mfsort ユーティリティは、コマンド行から起動し、データ ファイルのソートとマージを可能にします。

- 呼び出し可能ソート モジュール

  呼び出し可能ソート モジュール (Extsm) は、COBOL プログラムから呼び出すことができるスタンドアロンのソート モジュールです。

## 9.1 Mfsort ユーティリティ

Mfsort を使用すると、データ ファイルのソートとマージが可能になります。Mfsort は、mfsort.exe として提供され、次に示すどちらかの方法で、NetExpress コマンド プロンプトから起動されます。

```
mfsort instructions
mfsort take filename
```

各パラメータの内容は次のとおりです。

*instructions* 後述する「*Mfsort 命令*」の項を参照してください。コマンド行で命令を指定するときは、オペレーティング システムが定めたコマンド行の最大長に注意してください。

*filename* Mfsort 命令を含むテキスト ファイル。後述する「*Mfsort 命令*」の項を参照してください。 この方法は、多数の命令を指定する必要がある場合に使用してください。

### 9.1.1 Mfsort 作業ファイル

ソート操作中やマージ操作中に、Mfsort は一時作業ファイルを使用します。この作業ファイルは現行ディレクトリのディスクにページングされるか、TMP 環境変数で指定されたディレクトリがある場合はそのディレクトリにページングされます。

Mfsort は、各入力ファイルからすべてのレコードを一時作業ファイルにコピーするときに、適宜、レコードの切り捨てや埋め込みを行います。次に、キー記述にしたがって、作業ファイルでソートやマージを行います。作業ファイルでのソートやマージが済むと、レコードは各出力ファイルにコピーされ、適宜、切り捨てられるか、埋め込まれます。

- レコードを切り捨てたくない場合には、作業ファイルのレコード長を、ソートする最長レコードに合わせて

十分確保するようにしてください。

- 入力ファイルが可変長であり、ソート作業ファイルが可変長でない場合、ソート作業ファイルと出力ファイルのレコードは可変長でなくなります。
- 入力ファイルが固定長であり、作業ファイルが可変長である場合、作業ファイルのレコードのレコード長は、入力ファイルの固定長または作業ファイルの最大レコード長のどちらか小さい方になります。

### 9.1.2 Mfsort 命令

mfsort コマンドの一般的な形式を次に示します。

```
mfsort [*] [CHAR-EBCDIC] [SIGN-EBCDIC] {SORT|MERGE}
  fields(instructions,...) [record definition]
   {USE input-file}...
   {GIVE output-file}...
```

| 命令 | 説明 |
|---|---|
| * | 行の残りの部分はコメントとして処理されます。この命令は、ファイルに各命令の目的を説明するコメントを追加できるため、テキスト ファイルで命令を指定する場合に便利です。 |
| CHAR-EBCDIC | EBCDIC データを指定します。CHAR-EBCDIC は SORT 命令、MERGE 命令、USE 命令、GIVE 命令のすべてより先に指定する必要があります。 |
| SIGN-EBCDIC | 数字の DISPLAY 項目を指定し、符合は EBCDIC 規則にしたがって解釈されます。CHAR-EBCDIC が指定されている場合には SIGN-EBCDIC は不要ですが、データを作成したプログラムが SIGN"EBCDIC" コンパイラ指令でコンパイルされている場合のように、ASCII 以外のデータには SIGN-EBCDIC を設定する必要があります。 SIGN-EBCDIC は、SORT 命令、MERGE 命令、USE 命令、GIVE命令のすべてより先に指定する必要があります。 |
| SORT | ソート操作を指定します。SORT 命令の次には、ソート操作に使用するフィールドを指定する FIELDS 命令を続ける必要があります。オプションで、ソート作業ファイルのレコード サイズと形式を指定する RECORD 命令を設定する場合、SORT 命令の次に続けます。SORT 命令と MERGE 命令は相互排他的に機能します。 |
| MERGE | マージ操作を指定します。この命令の次には、マージ操作に使用するフィールドを指定する FIELDS 命令を続けます。オプションで、ソート作業ファイルのレコード サイズと形式を指定する RECORD 命令を設定する場合、MERGE 命令の次に続けます。MERGE 命令と SORT 命令は相互排他的に機能します。 |

fields (*instructions*)　ファイルのソートやマージを行うフィールド。後述する「*FIELDS 命令*」の項を参照してください。

record *definition*　レコード サイズと形式。RECORD 命令は、作業ファイル、入力ファイル、出力ファイルの詳細を指定するために使用します。後述する「*RECORD 命令*」の項を参照してください。

USE *input-file*　USE 命令では、入力ファイルを指定します。USE 命令は必ず GIVE 命令の前に指定する必要があります。後述する「*Mfsort の入力ファイルと出力ファイル*」の項を参照してください。

GIVE *output-file*　GIVE 命令では、出力ファイルを指定します。後述する「*Mfsort の入力ファイルと出力ファイル*」の項を参照してください。

#### 9.1.2.1 FIELDS 命令

SORT 命令や MERGE 命令の次には、入力ファイルのソートやマージを行うフィールドを指定する FIELDS 命令を続ける必要があります。

FIELDS 命令の形式を次に示します。

```
fields({start,length,type,order},...)
```

*start*　レコード中のフィールドの開始位置。バイト 1 からカウントします。

*length*　フィールドの長さ (バイト数)。

*type*　フィールドのデータ型。後述する「*フィールドの型*」の項を参照してください。

*order*　出力の順番で、次のどちらかになります。
A - 昇順
D - 降順

パラメータ セット (*start*、*length*、*type*、*order*) を繰り返すことにより、最大 16 フィールドを指定することができます。パラメータとパラメータセットはコンマで区切る必要があります。

##### 9.1.2.1.1 フィールドの型

フィールドの型は、次のどれかになります。

| 型 | 定義 |
|---|---|
| CH | PIC X DISPLAY |
| NU | PIC 9 DISPLAY |

| | |
|---|---|
| PD | PIC S9 COMP-3 |
| FI | PIC s9.99 DISPLAY |
| BI | COMP |
| C5 | COMP-5 |
| C6 | COMP-6 |
| S5 | S9 COMP-5 |
| CX | COMP-X |
| LS | PIC S9 LEADING SEPARATE |
| TS | PIC S9 TRAILING SEPARATE |
| LI | PIC S9 LEADING INCLUDED |
| TI | PIC S9 TRAILING INCLUDED (コンパイルされた SIGN"EBCDIC") |

例えば、golf.datという相対ファイルを COBOL プログラムで次のように定義することができます。

```
file-control.
select members-file
   assign to "d:¥netexpress¥base¥workarea¥golf.dat"
   organization is relative
   access mode is random
   relative key is relative-key.
data division.
file section.
fd members-file
   record contains 28 characters.
01 members-record.
   03 members-number pic 9(6).
   03 members-lname pic x(10).
   03 members-fname pic x(10).
   03 members-handicap pic 9(2).
```

次の mfsort コマンドを使用すると、メンバーシップ番号を含むフィールドでファイル golf.dat をソートすることが

できます。

```
mfsort sort fields(1,6,nu,a)
   use golf.dat record f,28 org rl
   give members.dat
```

ソートしたファイルは、ファイル members.dat に書き込まれます。

### 9.1.2.2 Mfsort の入力ファイルと出力ファイル

入力ファイルを定義するには、次の命令を使用します。

```
USE input-file [record definition]
   [org organization] [key structure]
```

また、出力ファイルを定義するには、次の命令を使用します。

```
GIVE output-file [record definition]
   [org organization] [key structure]
```

これらの命令は、関連付けた USE 命令や GIVE 命令のすぐ後に配置する必要があります。上記の命令を一部でも省略すると、前回に指定した値が使用されます。そのため、入力ファイルと出力ファイルをすべて同じ型と形式にする場合、最初にファイルの値を指定すると後で指定し直す必要がありません。

#### 9.1.2.2.1 RECORD 命令

RECORD 命令はオプションです。RECORD 命令を使用して、ソート作業ファイルで、レコードの形式と長さを指定します。また、関連付けられた USE 命令または GIVE 命令にしたがう場合は、入力ファイルと出力ファイルでもレコードの形式と長さを指定します。

RECORD命令の形式は次のとおりです。

```
record format,rec-len,max-len
```

*format*　　レコード形式で、次のどちらかになります。

F - 長さが *rec-len* の固定長レコード

V - 最小長が *rec-len* で、最大長が *max-len* の可変長レコード

ソート作業ファイルに対して RECORD 命令を指定しない場合、デフォルトの固定長レコード形式になり、レコード サイズは USE 命令や GIVE 命令で指定された最大レコード長になります。

#### 9.1.2.2.2 ORG 命令

ORG 命令では、ファイル編成を指定します。ファイル編成は、次のどれかになります。

IX - 索引

RL - 相対

SQ - 順

LS - 行順

ORG のデフォルト値は SQ です。

#### 9.1.2.2.3 KEY 命令

KEY 命令では、ファイルのキー構造体を指定するため、索引ファイルには必ず指定する必要があります。他のファイル編成には関係ありません。

KEY 命令の形式を次に示します。

```
key ({start,length,ixkey},...)
```

*start* レコード中のキーの開始位置。バイト 1 からカウントします。

*length* キーのバイト数。

*ixkey* 次のどれかになります。

P - 主キー (常に最初に定義します)

A - 副キー

AD - 重複指定した副キー

C - 前回指定した主キーまたは副キーの構成要素

KEY 命令は、キー構造体全体を記述するために必要に応じて反復します。パラメータとパラメータ セット (*start*、*length*、*ixkey*) はコンマで区切る必要があります。

キーは、重要な順に定義します。主キー、主キーが分割されている場合はすべての構成要素、最初の副キー、そのすべての構成要素という順序で定義する必要があります。

次の例では、3 つのキーを定義します。

```
key (4,5,p,10,5,c,20,2,ad,40,2,a,46,10,c)
```

最初のキーである主キーは、文字位置 4 から始まる長さ 5 バイトの最初の構成要素と文字位置 10 から始まる長さ 5 バイト 2 番目の構成要素に分割されます。

2 番目のキー (副キー) は重複可能で、文字位置 20 で始まり、長さが 2 バイトになります。

3 番目のキーは副キーで、文字位置 40 から始まる長さが 2 バイトの最初の構成要素と、文字位置 46 から始まる長さ 10 バイトの 2 番目の構成要素に分割されます。

### 9.1.3 Mfsort コマンド例

国内の東西南北の 4 地域について、国内大会で国内組織のメンバーが達成したスコアを記録した索引ファイル (north.dat、south.dat、east.dat および west.dat) がそれぞれ 1 つずつ、計 4 つあると想定します。このうち north.dat を定義するには、次のような COBOL 構文を使用します。

```
file-control.
   select idxfile assign to "north.dat"
      organization is indexed
      record key is member-id.
data division.
file section.
fd idxfile
record contains 39 characters.
01 idxfile-record.
   03 member-id  pic 9(6).
   03 surname    pic x(15).
   03 first-name pic x(15).
   03 score     pic 9(3).
```

さらに、他のファイルも同様の方法で作成し、大会の結果をファイルに入力したと仮定します。

次の mfsort コマンドでは、これら 4 つのファイルを取り出し、メンバーの姓でソートしてから、相対ファイル members.dat に結果を出力します。

```
mfsort sort fields(7,15,ch,a)
   use north.dat org ix record f,39 key(1,6,p)
   use south.dat
   use east.dat
   use west.dat
   give members.dat org rl
```

次の mfsort コマンドでは、4 つのファイルを取り出し、メンバーのスコア順 (スコアの高い順に並べる) でソートしてから、その結果を相対ファイル scores.dat に出力します。

```
mfsort sort fields(37,3,nu,d)
   use north.dat org ix record f,39 key(1,6,p)
   use south.dat
```

```
   use east.dat
   use west.dat
   give scores.dat org rl
```

次の mfsort コマンドでは、4 つのファイルを取り出し、会員番号順 (主キー) でソートしてから、その結果を索引ファイル national.dat に出力します。

```
mfsort fields(1,6,nu,a)
   use north.dat org ix record f,39 key(1,6,p)
   use south.dat
   use east.dat
   use west.dat
   give national.dat
```

### 9.1.4 Mfsort エラー メッセージ

Mfsort エラー メッセージの詳細なリストは、オンライン ヘルプ ファイルに記載されています。 ヘルプ ファイルの索引で [*Mfsort*] を参照してください。

## 9.2 呼び出し可能ソートモジュール

呼び出し可能ソートモジュールは、データ ファイルのソートとマージを可能にするスタンドアロンのソートモジュールです。このモジュールは、COBOL プログラムで SORT 文または MERGE 文を使用すると、暗黙的に呼び出されます。

データ ファイルに SORT 操作を実行すると、重複キーをもつレコードは、SORT 文で DUPLICATES IN ORDER 句を指定したかどうかにかかわらず、元の順番に戻ります。

### 9.2.1 呼び出し可能ソートモジュールの呼び出し

プログラムに次の呼び出し文を記述すると、呼び出し可能ソートモジュールを明示的に呼び出し、ファイルのソートやマージを行うことができます。

```
call "EXTSM" using function-code, sort-fcd
```

この場合、パラメータの内容は次のとおりです。

*function-code* 実行する操作の種類を示す 2 バイトのコード。

*sort-fcd* プログラムのデータ部で指定したソートファイル制御記述 (FCD) の名前。FCD には、ファイルのレコード領域、ファイル名、照合順序、キー定義ブロック、およびファイル定義ブロックへのポインタを記述します。プログラムのデータ部で、SORT 操作に

必要な各ファイルに対して Sort FCD を指定する必要があります。

関数コードの詳細なリストと Sort FCD の詳細な記述は、オンライン ヘルプに記載されています。ヘルプ ファイルの索引で [*ファイルのソート*] を参照してください。

# 第10章 REBUILD

## 10.1 概要

REBUILD は、NetExpress コマンド プロンプトで呼び出すファイル管理ユーティリティです。主な機能は、次のとおりです。

- 索引ファイルの再編成
- 破損した索引ファイルのリビルド
- LEVEL II V2.5 COBOLファイルの変換
- 索引ファイルの検証

REBUILD がファイルに対して実行する操作は、REBUILD コマンド行で指定するオプションによって異なります。

---

警告

- システムが Fileshare を使用するように構成されている場合、REBUILD は Fileshare クライアントの構成情報を使用します。 Fileshare の詳細については、『*Fileshare ユーザー ガイド*』を参照してください。
- ファイルの再編成、リビルド、または変換を行う前に、常にファイルのバックアップ コピーを作成しておいてください。この作業は、コマンド行を誤って指定したためにファイルが破損してしまう場合に備えて行います。索引ファイルの場合、ファイルの索引部分とデータ部分の両方を保存しておくことが必要です。
- REBUILD は IDXFORMAT 8 ファイルの索引をリビルドするために一時ファイルを使用します。この一時ファイルは現行のディレクトリに配置されます。しかし、TEMP 環境変数を設定した場合は、一時ファイルの場所は TEMP 変数によって異なります。一時ファイルは最終的にはリビルドしているファイルと同じファイル サイズとなりますので、注意してください。このためこの操作を実行するには、十分なディスク容量を確保してください。
- IDXFORMAT 8 ファイルをリビルドまたは再編成する時は、 /q および /d オプションを使用しないでください。

---

## 10.2 コマンド行

REBUILD コマンド行の一般的な形式は、次のとおりです。

```
rebuild in-file [,out-file] [options]
```

*in-file* と *out-file* の両方に、ファイル名の拡張子を付ける必要があります。*out-file* のファイル名は *in-file* のファイル名と異なる必要があります。 *in-file* や *out-file* の代わりに環境変数を使用してファイル名マッピングを実行することができます。

REBUILD が実行する操作内容は、コマンド行で指定するオプションにより定義されます。

### 10.2.1 パラメータ ファイル

REBUILD コマンド行で at 文字 (@) の次にファイル名を記述すると、そのファイルに REBUILD コマンド行パラメータが含まれていることを示します。

ファイルを使用して REBUILD コマンド行パラメータを定義する場合、このファイルは ASCII テキスト ファイルである必要があります (このようなファイルの作成や編集には、「メモ帳」を使用することができます)。

REBUILD パラメータ ファイルは 1024 バイト以内である必要があります。

REBUILD パラメータ ファイルでは、空白文字は意味をもたず、行末は空白文字として処理されます。また、アスタリスク (*) と行末の間のテキストはコメントとして処理されます。

例

```
rebuild badms001.dat, ms001.dat @ms001.par
```

上記の例では、ms001.parは、REBUILD パラメータを含む ASCII テキスト ファイルです。

### 10.2.2 情報のリダイレクト

REBUILD ユーティリティで表示されるすべての情報は、標準的なオペレーティング システムのリダイレクトを使用して、テキスト ファイルにリダイレクトすることができます。

> 新規ファイルを作成します。

>> 既存のファイルを拡張します。

例

```
rebuild oldms001.dat,ms001.dat /k:1+20<N>/i >> rebuild.prt
```

上記の例では、REBUILD からの出力を既存のファイル rebuild.prt にリダイレクトします。

---

注意: 出力をリダイレクトする場合、/v オプションを使用しないでください。

---

### 10.2.3 システム パラメータ

次のシステム パラメータを、コマンド行で指定することができます。

//q　　バナーを表示しません。

//v　　REBUILD プログラム自体のバージョン番号を表示します。

## 10.3 REBUILD オプション

REBUILD により実行される処理は、コマンド行で指定したオプションにより定義されます。使用できるオプションは次のとおりです。

- /c - ファイル圧縮を指定します。
- /d - 破損したデータ レコードをスキップします。
- /e - 不正な重複キーを報告し、処理を続行します。
- /f - 索引ファイルを検証します。
- /i - 処理するファイルの情報を表示します。
- /k - 索引ファイルのキー構造を定義します。
- /n - ファイルに関する情報を表示します (他の処理は実行されません)。
- /o - 入力ファイルの編成を指定します。
- /r - ファイルのレコード長とレコード モードを定義します。
- /s - 入力ファイルの形式を指定します。
- /t - 出力ファイルの形式を指定します。
- /v - ファイルが処理されると増えるレコード カウントを表示します。
- /x - 索引ファイルを再編成したときにデータを書き込む順番を指定します。

オプションは、コマンド行のどの順番に置いてもかまいません。

オプションを指定する場合、コロン (:) を使用してパラメータとオプションを分離することができます。

---

注意: REBUILD がファイルと競合するデフォルト値を使用しないようにするため、REBUILD コマンドはできるだけ明示的に作成してください。

---

REBUILD オプションの詳細と指定方法は、オンライン ヘルプ ファイルに記載されています。ヘルプ ファイルの索引で [*REBUILD*] を参照してください。

## 10.4 索引ファイルの再編成

ファイルを再編成するコマンド行の形式は、次のとおりです。

```
rebuild in-file,out-file [/c] [/d {/k}] [/x] [/I] [/v]
```

索引ファイルは、レコードの追加、削除、変更などにより更新されるときに、索引構造体とデータ構造体に切り離されるため、処理効率が悪くなります。さらに、削除されたレコードに残された領域は常に再使用されるわけではないので、ファイルが必要以上に大きくなってしまいます。

多くの変更を行った場合、索引ファイルを再編成し、データと索引を順にリビルドし、空き領域を再使用すると便利です。この作業により、最適な性能とデータの完全性が保証されます。

ファイルをリビルドする意義は他にもあります。例えば、ファイルを順に処理する場合、ファイルを処理するためのアクセス時間は時間が経つにつれて増えます。これは、レコードが更新され、順番が変更されるためです。この状況でファイルを再編成すると、アクセス時間を短縮することができます。

例

```
rebuild infile.dat,outfile.dat
```

この例では、REBUILD は、索引ファイル (infile.idx) を使用してデータ ファイル (infile.dat) を読み込み、出力ファイル (outfile.dat) を作成します。infile.dat の削除されたレコードは出力ファイルに書き込まれません。

---

注意: 入力ファイルと出力ファイルに同じファイル名を使用することはできません。

---

## 10.5 破損した索引ファイルのリビルド

索引をリビルドするためのコマンド行の形式は、次のとおりです。

```
rebuild in-file [/c] [/k] [/I] [/v] [/e] [/f] [/q]
```

索引ファイルの破損理由はいくつかあります。次に例を示します。

- 索引ファイルを閉じる操作が失敗した。
- 索引ファイルが開いている間にパワー サージが発生した。
- 索引ファイルが開いている間にマシンがリブートされたか、電源がオフになった。

COBOL プログラムが破損した索引ファイルを開くときに、ランタイム システムがファイルの破損を検出すると、拡張ファイル状態コードを返します。

REBUILD では、破損したファイルを自動的にリビルドすることができます。

- ファイルのデータ部分が完全で、索引部分だけが破損している場合、REBUILD はデータ ファイルから完全なデータを持つ新しい索引 (.idx) ファイルを作成することができます。
- ファイルのデータ部分が破損している場合、REBUILD は、新しい索引ファイル (データ部分と索引部分の両方) をリビルドすることができます。ただし、REBUILD は検出したデータ レコードの破損部分を廃棄するため、データの損失が起こります。

---

注意: REBUILD を使用して既存のデータ ファイルから新しい索引ファイルを生成すると、元の索引に記述されているデータ ファイルの空き領域情報は失われます。これは、データ ファイルの空き領域を再使用できず、ファイルが必要以上に大きくなる可能性があることを意味します。この問題を解決するには、索引をリビルドした後でファイルを再編成する必要があります。

---

次の例では、REBUILD は、.idx ファイルを読み取ってキー情報を取得し、さらにデータ ファイル infile.dat を順に読み取って、新しい索引ファイルを作成します。

```
rebuild infile.dat
```

索引ファイルがない場合には、/k オプションを使用してキー構造体に関する情報を指定する必要があります。

### 10.5.1 ファイル索引の修正

NetExpress の [ツール] メニューから、[データ ツール]、[ファイル索引修正] の順に選択すると、破損した索引ファイルの索引をリビルドすることができます。

---

注記: ファイルの修正では IDXFORMAT 8 ファイルはサポートしていません。IDXFORMAT 8 形式のファイルの索

引をビルドするには、コマンド行から REBUILD を使用してください。

---

ファイル索引修正画面が次のように表示されます。


図 10-1 ファイル索引修正

破損した索引ファイルの名前を入力します (または [参照] ボタンを使用して選択します)。

ファイル ヘッダー情報を入手できる場合、[ファイル索引修正] ボタンを押すと索引ファイルを再作成できます。

ファイル ヘッダー情報を入手できない場合、メッセージが表示されます。Fix File がこの情報をファイル ヘッダーから取り出すことができないので、ファイル形式を選択し、ファイル キーを定義する必要があります。この操作を行うには、[高度なオプション] ボタンをクリックします。すると、次の画面が表示されます。

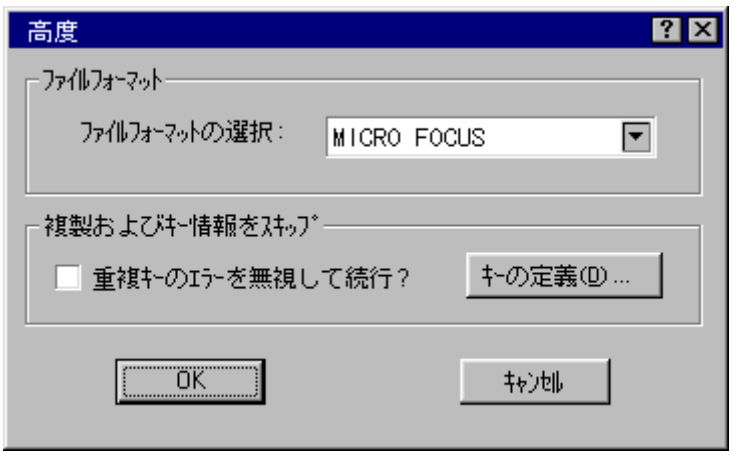


図 10-2 高度なオプション

このダイアログでは、ファイル形式を指定することができます (ファイル形式は、破損する前の元のファイルと同じでなければなりません)。また、重複キーをもつレコードを検出した場合、Fix File により重複を報告し、残りのレコードの処理を続行するように指定することもできます (重複キーをもつレコードは書き込まれません)。キー情報を指定するには、[キーの定義] ボタンをクリックします。


図 10-3 キー情報

(ヘッダーの読み取りが成功すると、ファイルのすべてのキー情報がここに表示されます。)

キーを作成するには、[キーを挿入] をクリックして最初のキーを追加します。ツリー構造が拡張され、フォーカスがオフセット フィールドに切り替えられます。各キーに対して、キー オプション、キー圧縮、およびキーの構成要素に関する情報を指定する必要があります。

キーの長さは、1,016 バイト、または最大のレコード長よりも長くすることはできません。キー オフセットは、最小レコード長、または 63,488 バイトのどちらか短い方よりも大きくすることはできません。

キーに複数の構成要素がある場合 (キーの構成要素はツリー構造の子項目として表示されます)、[前に構成要素を挿入] または [後に構成要素を挿入] ボタンを使用して、別の構成要素を挿入することができます。これらのボタンを表示するには、キーを選択しておく必要があるので注意してください。

ファイルに複数のキーが定義されている場合、[前にキーを挿入] ボタンまたは [後にキーを挿入] ボタンを使用して別のキーを挿入することができます。これらのボタンを表示するには、キーを選択しておく必要があるので注意してください。

## 10.6 ファイルの変換

ファイル変換用のコマンド行の形式は、次のとおりです。

```
rebuild in-file,out-file [/s] [/o] [/r] [/t] [/k] [/c] [/I] [/v]
```

REBUILD を指定すると、LEVEL II V2.5 COBOL ファイルをこの COBOL システムで使用する索引ファイルの形式に変換できます。

REBUILD でリビルドできるのは、このシステムで使用する形式の索引ファイルだけです。そのため、REBUILD を使用してファイルをこの形式に変換しておくと、破損の際にこれらのファイルをリビルドすることができます。

次の例では、REBUILD を使用して C-ISAM ファイルをこのシステムで使用する形式に変換し、ファイルのデータ部分を圧縮します。

```
rebuild infile.dat,outfile.dat /s:c-isam /c:d1 /t:mf
```

## 10.7 索引ファイルの検証

ファイルを検証するためのコマンド行の形式は、次のとおりです。

```
rebuild in-file /f[c] [d]
```

REBUILD を使用すると、索引ファイルの構造を検証することができます。

REBUILD を使用すると、多くの検証を実行できるうえ、これらの検証をすべて実行するか、または、サブセットだけを実行するかを選択することができます。確認する内容が多いほど、検証処理にかかる時間は長くなります。

次の例では、REBUILD はファイルの完全性をすべて検査します。

```
rebuild test.dat /f:c63d5
```

## 10.8 呼び出し可能 REBUILD

次のように指定すると、COBOL プログラムで REBUILD を呼び出すことができます。

```
call "CALLRB" using commands status
```

この場合、パラメータの内容は次のとおりです。

*commands* REBUILD コマンド行を格納する PIC X(600) 項目。REBUILD はオフセット 599 から 0 へ向かってコマンド行を調べるため、この項目の長さは 600 バイトになります。

*status* 返されたファイル状態を格納する PIC XX COMP X 項目。この項目は、REBUILD の呼び出し結果を示します。

REBUILD をプログラムで呼び出すと、/v オプションを指定しない限り、通常のメッセージは表示されません。/v オ

プションを設定すると、処理中のレコードの合計が表示されます。

エラーが発生した場合、またはリビルドが成功しなかった場合、RETURN-CODE にはゼロ以外の値が格納され、*status* には返されたファイル状態が格納されます。REBUILD を呼び出した後は、常に RETURN-CODE と *status* を確認してください。

RETURN-CODE に格納される値を次に示します。

| 値 | 説明 |
|---|---|
| 0 | REBUILD が正常に実行されたことを示します。 |
| 1 | ファイルが見つからない、ファイル形式が無効である、などの入力ファイル エラーが発生しています。状態パラメータを確認してください。 |
| 2 | 出力ファイルにエラーが発生したことを示します。状態パラメータを確認してください。 |
| 9 | オプションが無効である、オプションの組み合わせが無効である、などのエラーがパラメータ リストに含まれていることを示します。 |

エラーが発生した場合、同一のパラメータ セットを使用して同じファイルのコマンド行から REBUILD を実行します。画面出力では、エラー原因についてより詳細な情報が表示されます。

---

注意: /f オプション (索引ファイルの検証) を使用している場合、ゼロ以外の RETURN-CODE はファイルが破損していることを示します。その場合、より詳細な情報を取得するために、同一のパラメータ セットで同じファイルのコマンド行から再度 REBUILD を実行してください。

---

次の例は、COBOL プログラムによる REBUILD の呼び出しを示します。

```
01 parameters   pic x(600).
01 status      pic xx comp-x.
.
.
.
move "infile.dat,outfile.dat /s:lii /c:d1" to parameters
call "CALLRB" using parameters,status
end-call
```

## 10.9 カスタム REBUILD の作成

REBUILD ユーティリティは、この COBOL システムの rebuild.exe に格納されています。REBUILD ユーティリティには、実行できるすべてのジョブに対するモジュールが含まれているため、ファイルの容量が大きくなっています。ただし、これらのモジュールは、ユーザーが独自にカスタマイズした REBUILD を作成できるように個別に分かれています。次の表は、カスタム バージョン rebuild.exe や REBUILD を呼び出すプログラムを作成するためにリンクする必要のあるオブジェクトを示します。これらはオプションです。

### 10.9.1 必須オブジェクト

| | |
|---|---|
| REBUILD | REBUILD メイン モジュール (カスタマイズされた rebuild.exe 用) |
| または | |
| CALLRB | メインの呼び出し可能 REBUILD モジュール (ユーザー プログラム用)¡ |
| および | |
| RBLDMAIN | コマンド行処理 |
| RBLDSUB | ファイルを処理し、ファイル ハンドラを呼び出します。 |
| EXTERNL | REBUILD は外部データ項目を使用します。 |
| mFFH | Micro Focus ファイル ハンドラ |
| FHRSUB | カスタム REBUILD の作成の基本 |

### 10.9.2 オプションのオブジェクト

| | |
|---|---|
| MSCVSUB | MS 形式のファイルから Micro Focus 形式のファイルに変換する必要があります。 |
| CDFH | /d オプションを使用する場合に必要です。 |
| XFHNAME | ファイル名マッピングを行う場合に必要です。 |
| CBLDC001 | データ圧縮を使用する場合に必要です (DATACOMPRESS"1")。 |
| CBLDC003 | データ圧縮を使用する場合に必要です (DATACOMPRESS"3")。 |
| FHREDIR | Fileshare を使用してファイルをリビルドする場合に必要です。 |
| FHRDRPWD | FHREDIR に必要です。 |
| FHXSCOMP | FHREDIR に必要です。 |

IDXCHECK　　索引ファイルの検証に必要です。

## 10.10 エラー メッセージ

REBUILD は、索引ファイルのリビルド、変換、再編成を行うときに、エラー、情報、または警告メッセージを出力する場合があります。

REBUILD のエラー メッセージのリストは、オンライン ヘルプ ファイルに記載されています。ヘルプ ファイルの索引で [REBUILD] を参照してください。

---

注意: ファイル索引修正ツールでは、エラー メッセージでなくファイル状態コードが表示されます。

---

## 10.11 REBUILD の例

オンライン ヘルプ ファイルには、多くの REBUILD 例が記載されています。ヘルプ ファイルの索引で [*REBUILD*] を参照してください。

# 第11章 GUI データ ツール

この章では、PC でアプリケーションのデータ ファイルを作成し、維持するための NetExpress の機能について説明します。

## 11.1 はじめに

NetExpress では、次のような GUI データ ツールを提供し、データ ファイルの作成、維持、変換を支援します。

- データ ファイル エディタ

  さまざまな種類のデータ ファイルを作成、編集、印刷します。

- データ ファイル コンバータ

  データ ファイルをある形式から別の形式に変換します。

- レコード レイアウト エディタ

  レコードのレイアウトを作成し、維持します。 レコード レイアウトを使用すると、データ ファイル エディタとデータ ファイル コンバータの両方を個々のフィールド レベルで操作できます。

- ファイル索引の修正

  破損した索引ファイルの索引をリビルドします。

これらのデータ ツールの概要については、『*入門書*』の「データ ファイルの維持と作成」の章を参照することをお勧めします。

この章で説明する機能の使用方法に関する詳細は、NetExpress オンライン ヘルプのヘルプ情報を参照してください。(ヘルプ トピックのボタンをクリックし、[目次] タブがアクティブであることを確認してから、[開発環境]、[データ ツール] を選択してください。)

## 11.2 データ ファイルの種類

### 11.2.1 ファイルの種類とレコード長

NetExpress は、ファイルが開かれたときに、ファイルの種類とレコード長を識別する必要があります。索引ファイル、可変長順ファイル、可変長相対ファイルには、これらの情報を含むヘッダ レコードがあります。

行順編成、固定長順ファイル、固定長相対ファイルには、ファイル ヘッダがありません。 ファイルを開くたびに、入力したファイルの種類とレコード長を保存するには、データ ファイルと同じディレクトリにあるプロファイル ファイルにこの情報を保存することができます。プロファイル ファイルは、データ ファイルおよび .pro 拡張子と同

じファイル名があります。

NetExpress は、行順編成、固定長順ファイル、固定長相対ファイルを開く場合、データ ファイルと同じファイル名をもつ .pro ファイルを検索します。.pro ファイルがない場合、NetExpress がファイルの種類とレコード長を入力するよう求めてきます。 ソース コードを検査すると、ファイルの種類とレコード長を調べることができます。これらのファイル ヘッダの詳細を入力した後は、NetExpress がその詳細をプロファイル ファイルに保存するよう求めてきます。

#### 11.2.1.1 ファイルの種類の識別

行順ファイル、固定長順ファイル、および固定長相対ファイルを識別するには、SELECT 文の ORGANIZATION 句の入出力節 (Input-Output Section) を調べることが必要です。

- 行順ファイルは、ORGANIZATION IS LINE SEQUENTIAL で表されます。
- 固定長順ファイルは、ORGANIZATION IS RECORD SEQUENTIAL で表されます。
- 固定長相対ファイルは、ORGANIZATION IS RELATIVE で表されます。

---

注記:

- ORGANIZATION 句が ORGANIZATION IS SEQUENTIAL で、修飾子の LINE や RECORD がない場合、または、ORGANIZATION 句が全く存在せずに、SEQUENTIAL コンパイラ指令が LINE に設定されている場合、このファイルは行順ファイルです。その他の修飾子が設定されている場合や省略されている場合は、このファイルはレコード順ファイルです。
- ORGANIZATION IS は、付随的な用語なので、プログラムでは SEQUENTIAL、LINE SEQUENTIAL または RELATIVE というキーワードしか記述されない場合があります。詳細については、オンライン ヘルプの「*COBOL言語リファレンス*」の項を参照してください。

---

#### 11.2.1.2 レコード長の識別

レコード長を確認するためには、ファイル節の FD エントリとして記述されたレコード レイアウトを調べてください。レコード長は、FD グループ項目を構成する基本的なデータ項目の長さの合計です。

PIC X と PIC 9 の各項目では、1 つの文字位置が 1 バイトを占めます。計算用フィールド (COMP-1、COMP-2、COMP-3、COMP-4、COMP-5 および COMP-X) の長さを計算する方法の詳細については、オンラインヘルプの「*COBOL 言語リファレンス*」の項の「*概要 - 文字表現と基数の選択*」を参照してください。

## 11.3 データ ツールの設定

データ ツールの設定を変更すると、データ ツールの動作を修正することができます。これらの設定は、変更されないかぎり有効です。 設定を変更したり、表示するためには、［オプション］メニューから［データ ツール］を選択します。

これらの設定の多くは、データを誤って変更したり、削除したりした場合にデータを保護します。設定内容は次のとおりです。

- 編集前のファイル バックアップ

  デフォルトでは、ファイルを開くときに、バックアップが自動的に行われるように設定されています。ファイルを誤って編集してしまった場合に、バックアップを利用するとファイルのデータを失わないですみます。バックアップを使用すると、ファイルを編集前の状態に復元できます。

  バックアップ ファイルはデータ ファイルと同じディレクトリに格納されます。データ ファイルは、*filename*.dbk にコピーされます。データ ファイルに索引を作成する場合には、索引は *filename*.ibk にコピーされます。

- ファイルを開くときの読み取り専用設定

  データ ファイル エディタでは、ファイル オープンの方法を次のように設定することができます。

  - 読み取り専用

    他のプロセスもファイルを読み取ることができます。ただし、ファイルの内容は変更できません。

  - 読み書き可能

    これは編集モードと呼ばれます。ファイルのデータを編集することができ、他のソフトウェアがそのファイルにアクセスすることはできません。デフォルトでは、こちらが設定されています。

  ファイルに DOS の読み取り専用属性が設定されている場合、データ ファイル エディタの読み取り専用設定に関係なく、ファイルを更新することはできません。

- 更新時の警告

  デフォルトの設定では、データを編集するときにデータ ファイル エディタが警告メッセージを表示します。すべてのセッションについて、または現在のセッションのみについて、この警告をオフにすることができます。

- 削除時の警告

  デフォルトの設定では、レコードを削除するときにデータ ファイル エディタが警告メッセージを表示します。すべてのセッションについて、または現在のセッションのみについて、この警告をオフにすることがで

きます。

- 書式設定されていないビューでの上書き設定

  データを挿入または削除すると、すべてのデータ フィールドがずれる可能性があります。挿入編集の問題を避けるためには、常に上書きモードを使用するようにしてください。

  上書きモードに設定されているときに、Insert キーを押すと、挿入モードに切り替えるかどうかを確認するメッセージが表示されます。

  この設定は、書式設定されたビュー (「フィールド ビュー (書式設定されたビュー)」の項を参照)では、効果がありません。

- 埋め込みバイト

  レコードを挿入または作成するときに、埋め込みバイトを使用します。データ ファイル エディタが使用する埋め込みバイトは、使用中の文字セットによって異なります。 例えば、レコードを埋めるには、ヌル文字か空白文字を選択できます。EBCDIC と ANSI に対しては別の埋め込みバイトを使用することができます。

- 文字セット

  データ ファイルで使用されている文字セットをデータ ツールに認識させる必要があります。最も頻繁に使用する文字セットには、デフォルトの文字セットを設定する必要があります。特定のファイルを編集している間は、2 つの使用可能な文字セット (EBCDIC と ANSI) の間で切り替えることができます。

### 11.3.1 コードセット の構成

Using the Codecomp ユーティリティを使用すると、EBCDIC/ANSI 文字セット変換のためのカスタマイズされたマッピングの表を作成を作成できます。 NetExpress には多くの文字セットのサポートがビルトインされています。詳細については、NetExpress オンライン ヘルプ ファイルにあるヘルプを参照してください (ヘルプ トピックのボタンをクリックし、[目次] タブがアクティブであることを確認してから、[開発環境]、[データ ツール]、[コードセット の構成] を選択してください)。

## 11.4 データ ファイルの作成とオープン

データ ファイルを作成するために、データ ファイル エディタを使用することができます。

索引ファイルを作成した場合、キーを定義する必要があります(「キーの定義」の項を参照してください)。

ファイルがすでに作成されている場合には、ファイルを開いて編集することができます。

データ ファイルを作成するためには、まず、[ファイル] メニューの [新規作成] を選択します。次に、[新規作成] ダイアログ ボックスの [データ ファイル] を選択すると、「ファイルの作成」 ダイアログ ボックスが表示されます。


図 11-1 「ファイルの作成」 ダイアログ ボックス

次のように、レコード形式にしたがって「最大長」を指定します。

- 固定長形式

  固定長を指定します。

- 可変長形式

  最大レコード長を指定します。

次の表では、以下の内容をまとめてあります。

- 使用可能なファイル形式とレコード形式
- キーを定義する必要があるかどうか。
- 可変長レコード形式の場合、「最小長」に最小レコード長を入力する必要があるかどうか。

| ファイル形式 | レコード形式 | キー定義 | 最小長 |
|---|---|---|---|
| Micro Focus 順ファイル | 固定長、可変長 | 不要 | レコード形式が可変長の場合、設定する。 |
| Micro Focus 行順ファイル | 可変長 | 不要 | 設定しない。 |

| ファイル形式 | レコード形式 | キー定義 | 最小長 |
|---|---|---|---|
| Micro Focus 索引ファイル | 固定長、可変長 | 必要 | レコード形式が可変長の場合、設定する。最小長は、このファイルに定義された最大オフセットとキー長の合計より長い必要があります。 |
| Micro Focus 相対ファイル | 固定長、可変長 | 不要 | レコード形式が可変長の場合、設定する。 |
| IDXFORMAT(4) | 固定長、可変長 | 必要 | レコード形式が可変長の場合、設定する。 最小長は、このファイルに定義された最大オフセットとキー長の合計より長い必要があります。 |

### 11.4.1 キーの定義

索引ファイルの作成中には、「ファイルの作成」ダイアログ ボックスまたは「データ ファイルの変換」ダイアログ ボックスの［キーの定義］ボタンが有効化されます。[キーの定義] ボタンをクリックすると、「キーの情報」ダイアログ ボックスが表示されます。

データ ファイルのキーは、いくつかの構成要素に分けられるので、キーではなく、構成要素のオフセットと長さを指定します。ただし、キーとキーの構成要素の両方を扱うことができます。

データ ファイルのキーを定義する場合には、次の条件を満たす必要があります。

- キーを 1 つ以上定義する。
- 各キーに対して、構成要素を 1 つ以上定義する。
- 各構成要素について、長さを指定する。ただし、この長さは、1016 バイト未満または最大レコード長未満で、かつ、どちらか短い方が上限となる。
- 各構成要素について、オフセットを指定する。ただし、このオフセット値は、最小レコード長未満または 63,488 バイト未満で、かつ、どちらか短い方が上限となる。

さらに、次のような操作も可能です。

- 固有でないキーを使用する。つまり、キーの値が重複していてもかまわない。
- 現在のキーの前後に新しくキーを追加する。追加するキーは複数でもかまわない。「キーの情報」ダイアログ ボックスのキー リストにある最初のキーが常に主キーとなる。
- 現在の構成要素の前後に新しく構成要素を追加する。追加する構成要素は複数でもかまわない。
- キーや構成要素を削除する。

- キー圧縮を適用する。
- キーをスパース (疎) キーに指定する。キーにスパース (疎) 文字しかないレコードに、索引エントリが書き込まれることはない。

### 11.4.2 ファイルのオープン

順ファイルを開くと、データ ファイル エディタは、データ修正を行うためのメモリにファイルを読み込みます。作業した順ファイルを閉じるときに変更した内容を適用するかどうかをたずねるメッセージが表示されるまで、データ ファイルがディスクで更新されることはありません。

索引ファイルか相対ファイルを開く場合、この型のファイルは非常に大きくなることがあるため、メモリにロードされることはありません。これらのファイルのデータを修正し、レコードを外に移動させると、データ ファイル エディタはディスクのレコードを更新し、その結果を示すメッセージを表示します。

データ ファイル エディタには、データを誤って変更してしまわないようにファイルを読み取り専用として開くオプションがあります (「データ ツールの設定」の項を参照してください。) 。

[ファイル] メニューの [開く] を使用すると、ファイルを開くことができます。ファイルを開くと、データ ファイル エディタ ウィンドウが表示されます。

---

注記:バックアップ オプションを選択している場合に、バックアップ ファイルを格納できるだけの十分なディスク容量がないと、ファイルを開くことができません (「データ ツールの設定」を参照してください)。

---

## 11.5 データ ファイル エディタ ウィンドウ

データ ファイル エディタ ウィンドウは 2 つのペインに分割されます。

- レコード ビュー (書式設定されていないビュー)

  左側のペインには、各行が 1 レコードを持つ、書式設定されていないレコード データが表示されます。レコードに沿って左右にスクロールすると、同時に画面スクロール上のすべてのレコードを参照できます。

- フィールド ビュー (書式設定されたビュー)

  右側のペインには、フィールド レベルで書式設定されたデータが、 1 回に 1 レコードずつ表示されます。レコードに沿って左右にスクロールすると、同時に画面スクロール上のすべてのレコードを参照できます。区切り線で各フィールドを表示します。右側のペインを起動するには、レコード レイアウト ファイルを作成する必要があります。詳細については、「レコード レイアウト」の章を参照してください。

これらのビューについて、以下に詳細を説明します。

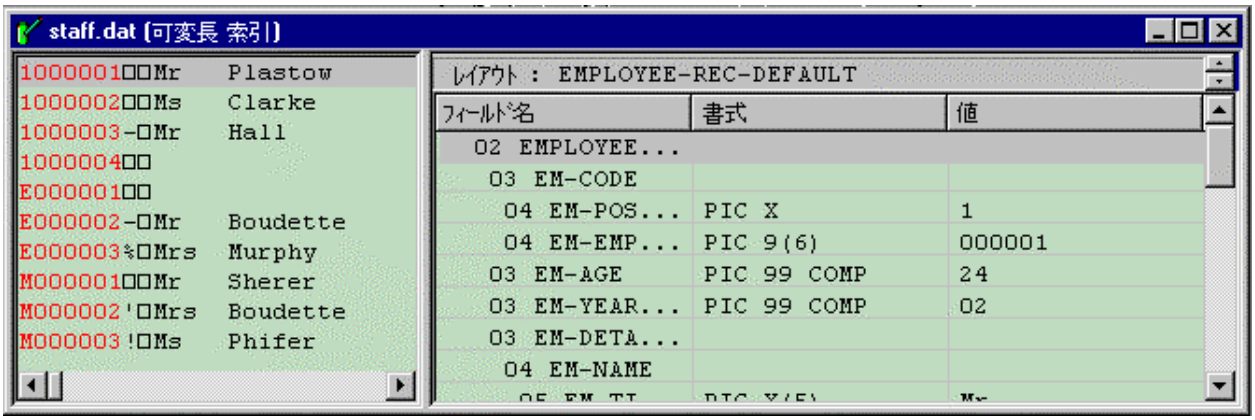


図 11-2: データ ファイル エディタ ウィンドウ

## 11.5.1 レコード ビュー (書式設定されていないビュー)

レコードを選択するには、フィールドをクリックするか、上下方向の矢印キーを使用します。現在選択されているレコードは、強調表示されます。

フォーカスがこのペイン上にある場合、NetExpress ウィンドウの最下部にあるステータス バーには、次の情報が左から右へ向かって表示されます (図 11-3 参照) 。

- 相対レコード番号 (相対ファイルの場合のみ)
- 現在、選択しているレコードの長さ

  可変長レコードについては、「最小長」および「最大長」で定義した最小長と最大長も表示されます。固定長の場合には、「(固定)」と表示されます。
- 行数で表すファイルのレコード数 (順ファイルの場合のみ)

  順ファイル以外のファイルは所定の場所で編集されるため、ファイルのレコード数は不明です (詳細については、「順ファイル以外のファイルの表示」を参照してください。)
- ファイルのカーソル位置 (レコード内の 0 から始まる行番号と列位置)
- 編集モード ("OVR" が表示される場合、上書きモード、空白の場合、挿入モードであることを示す。)。これらのモードは、[Insert] キーを押すと切り替えることができます。

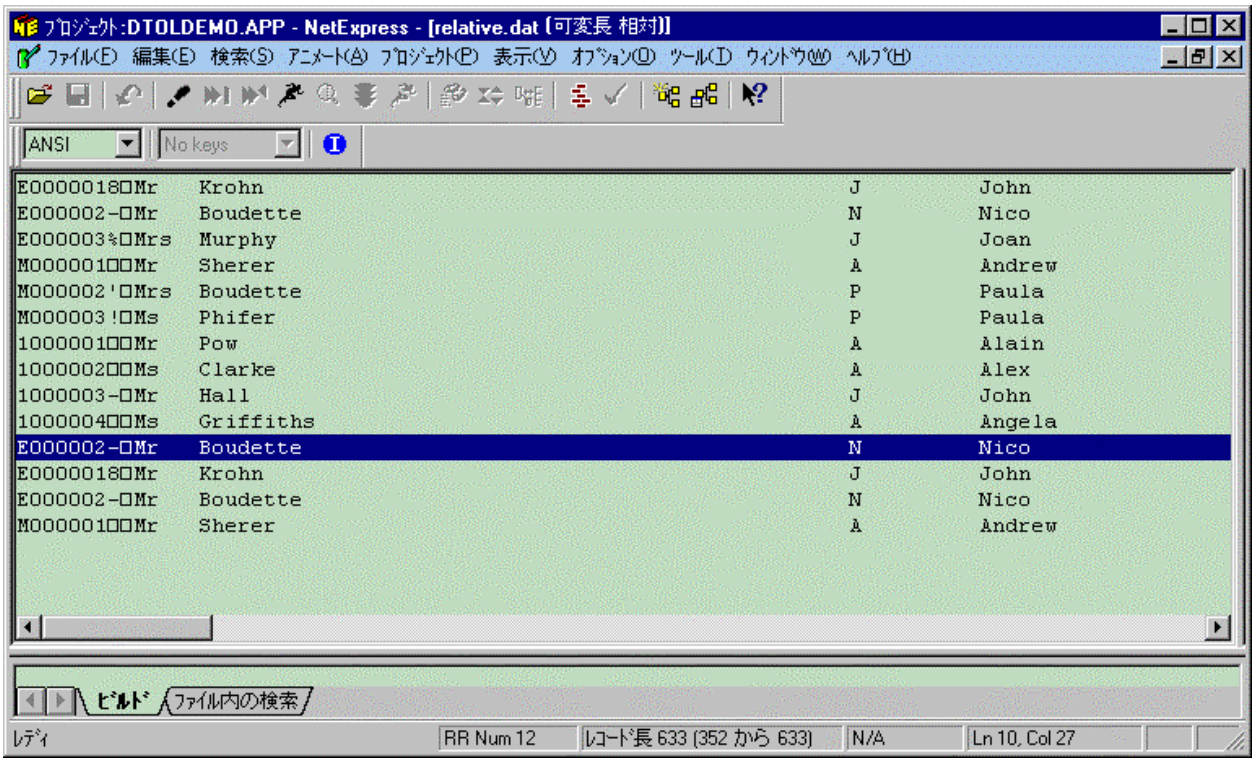


図 11-3 相対ファイルのレコード ビューとステータス バー

## 11.5.2 フィールド ビュー (書式設定されているビュー)

フィールド レベルでデータを表示する、またはフィールドの内容を編集するには、このビューを使用します。

- フィールド レベルでデータを表示する

  フィールドは、[フィールド名] または [書式設定] 列のいずれかから、フィールドをクリックして選択します。代わりに 2 つの列から上下方向の矢印キーを使用できます。現在選択されているフィールドは強調表示されます。最初のフィールドを選択するためには、[Home] キーを押します。最後のフィールドを選択するためには、[End] キーを押します。

- フィールドの内容を編集する

  フィールドの内容は、カーソルを列 [値] に合わせることで編集できます。最初のフィールドにカーソルを移動するには、[Home] キーを押し、最後のフィールドにカーソルを移動するには、[End] キーを押します。

レコード間を移動するには、フィールド ビュー ウィンドウの右ペインで右上部にある小さい上下方向の矢印ボタンを使用するか、レコード ビューペインの新しいレコードを選択します。

フィールド ビューペインが選択されている場合、NetExpress ウィンドウの最下部にあるステータス バーには、次の情報が左から右へ向かって表示されます (図 11-4 参照) 。

- 相対レコード番号 (相対ファイルのみ)

- 現在、選択しているレコードの長さ

  可変長レコードについては、「最小長」および「最大長」で定義した最小長と最大長も表示されます。固定長の場合には、「(固定)」と表示されます。

- レコードのフィールド数
- 現在選択されているフィールド番号
- 編集モード (空白)。数字フィールドは常に上書きモードで編集し、英数字フィールドは常に挿入モードで編集します。

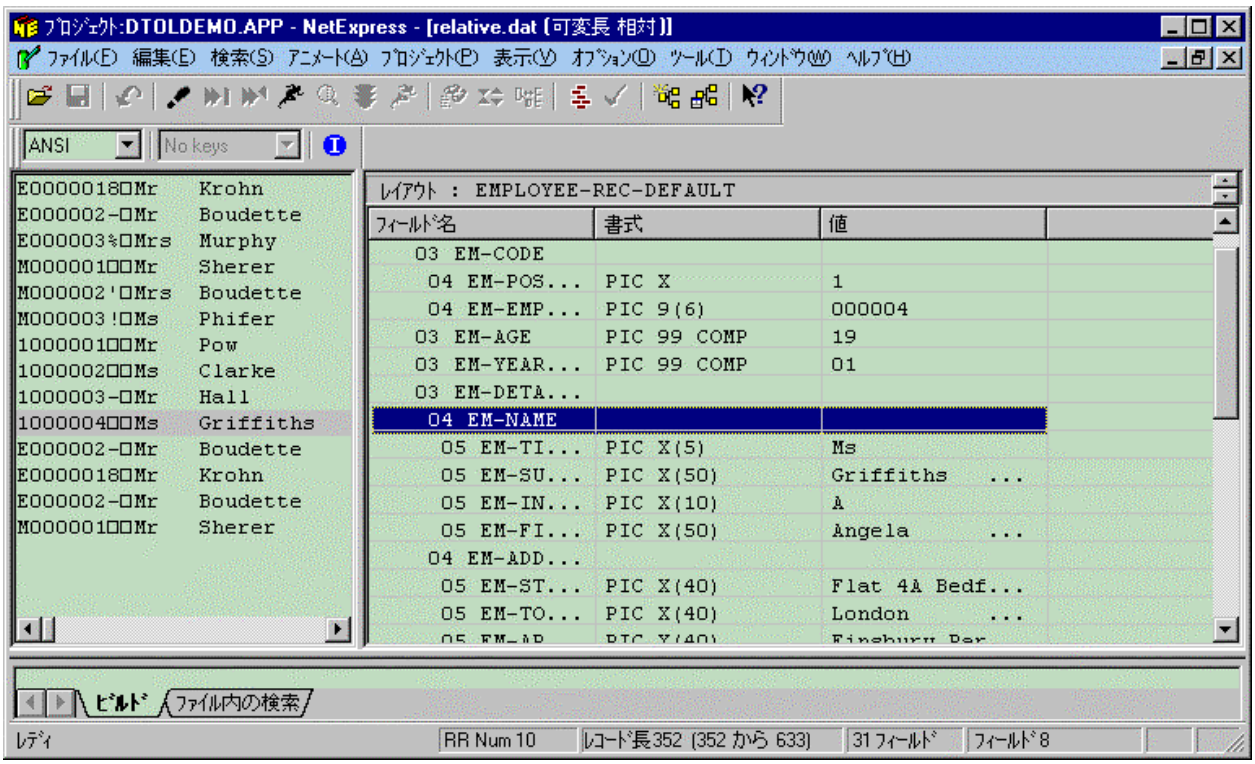


図 11-4 相対ファイルのフィールド ビューとステータス バー

## 11.5.3 2 つのビューの同期

前述した両方のビューを表示している場合、2 つのビューを同期させる方法は 2 種類あります。

フィールドまたはレコードにカーソルを合わせている場合、 [ビューの同期] を使用して、カーソルを他のペインの同じ位置に強制的に置くことができます。このオプションは、[ビュー] メニューにある [データ ツール] をクリックするか、データ ファイル エディタ ウィンドウを右クリックすると表示できます。

書式設定されていないビューのデータを更新した場合、[書式設定されたペインの最新表示] を選択すると、書式設定されたビューで変更結果を確認することができます。このオプションは、[ビュー] メニューの［データ ツール] をクリックすると表示できます。

## 11.6 レコードの作成

レコードの作成方法は、順ファイル、相対ファイル、および索引ファイルのそれぞれで異なります。これは、順ファイルにはキーがなく、また、相対ファイルと索引ファイルにはキーがあっても種類が異なるためです。ESDS ファイルには、特に配慮が必要です。

レコード レイアウト ファイルを読み込んでいる場合には、この中にある既存のレコード レイアウトの 1 つを使用した新規レコードを基礎とするか、または新規レコードに埋め込みバイトを書き込むかを選択することができます (詳細については、「レコード レイアウト」の項を参照してください)。 条件付きレコード レイアウトを作成してある場合、このうちの 1 つを選択して新規レコードに適用すると、条件フィールドが適切に設定されます。しかし、新規条件付きレコード レイアウトを追加するには、レコード レイアウト ファイルを再度作成する必要があります。

### 11.6.1 順ファイル レコードの作成

現在、選択されているレコードの前後に新しいレコードを挿入することができます。

新規レコードは、次に示す方法でも作成できます。

- 既存のレコードを一回コピーする。
- 既存のレコードを何回もコピーする。この方法は、複数反復と呼ばれ、順ファイルでテストデータを迅速に設定できます。

### 11.6.2 相対ファイル レコードの作成

相対ファイルで新規レコードを作成するときには、相対レコード番号を指定して、データ ファイル エディタが新しいレコードをファイルに配置できるようにしなければなりません。

新規レコードを作成するか、既存のレコードをコピーすることができます。

---

注記: 新規の相対レコードを作成するとき、ステータス バーに表示される行番号は 0 にリセットされます (「順ファイル以外のファイルの表示」の項を参照してください。)。

---

レコード レイアウトがロードされていない場合、適切な埋め込みバイトが次の値に達するまで書き込まれます。

- 可変長ファイルの場合、最小レコード長
- 固定長ファイルの場合、最大レコード長

### 11.6.3 索引ファイル レコードの作成

新規レコードを作成するか、既存のレコードをコピーすることができます。

索引ファイル レコードを作成するたび、またはコピーするたびに、新規レコードに一意のキーを指定します。主キーは常に一意のキーである必要があります。場合によっては、ファイルに 1 つまたは複数の一意の副キーがあることもあります。

レコード レイアウトを読み込んだレコードを挿入する場合、挿入処理中に入力されたすべてのキー情報は、選択したレコード レイアウトに関連付けられた条件付きデータよりも優先されます (「レコード レイアウト」の項を参照してください。)。

---

注記: 新しい索引ファイル レコードを作成する場合、ステータス バーに表示された行番号は 0 にリセットされます (「順ファイル以外のファイルの表示」の項を参照)。

---

### 11.6.4 ESDS ファイル レコードの作成

ESDS ファイルに新規レコードを作成するには、空のレコードをファイルの末尾に追加する方法しかありません。ファイルに副索引がある場合には、一意のキー値も指定する必要があります。

新規レコードは、常に、物理的にファイルの末尾に配置されますが、ファイルに副索引がある場合、ファイル中で参照キーにより指定された位置にある編集ウィンドウに空のレコードが表示されます。(「順ファイル以外のファイルの表示」の項を参照)。

レコードが追加されたら、編集セッションでフィールド値を入力します。

## 11.7 レコードの削除

ESDS ファイルを除き、すべての種類のデータ ファイルから 1 つまたは複数のレコードを削除することができます。

不適切なレコードがある場合でも、複数削除の手順を実行することができます。 削除処理は、ファイルの末尾まで継続されます。ファイルの末尾に到達すると、削除されたレコード数を表すメッセージが表示されます。

---

注記: 「警告を削除」チェック ボックスが選択されていることを確認してください。詳細については、「データ ツールの設定」の項を参照してください。

---

## 11.8 データの表示

データ ファイルは、次に示す 2 つの方法で表示することができます。

- レコード レベルで表示する方法 (書式設定されていないビュー)。複数のレコードが同時に表示されます。また、1 行に 1 レコードずつ表示します。このビューの詳細については、「レコード ビュー (書式設定されていないビュー)」の項を参照してください。
- フィールド レベルで表示する方法 (書式設定されたビュー)。この場合、アプリケーションが使用するフィールド レベルで書式設定されたレコードが表示されます。このビューの詳細については、「フィールド ビュー(書式設定されたビュー)」の項を参照してください。このビューを使用するためには、レコード レイアウトを記述するレコード レイアウト ファイルを作成する必要があります。アプリケーション ソース コードのレコード記述を使用して、このファイルを作成することができます。詳細については、「レコード レイアウト」の項を参照してください。

データ ファイル エディタでは、ANSI または EBCDIC のデータを 16 進で表示することができます。各バイトの 16 進値は、編集メイン ウィンドウの下にあるウィンドウに表示されます。各文字の 16 進表現は、すぐ下の 2 行にレコード レベルとフィールド レベルで表示されます。

図 11-6 は、16 進値で表現されたファイルを示します。

### 11.8.1 ファイル情報の表示

「ファイル情報」ダイアログ ボックスを使用すると、ファイルの詳細と、現在のデータ ファイル エディタの設定を確認することができます。「ファイル情報」ダイアログ ボックスを開くためには、[ビュー] メニューから [データ ツール] を選択し、さらに [ファイル情報] を選択します。

### 11.8.2 文字セットの変更

データ ファイル エディタでは、すべてのデータを ANSI または EBCDIC のどちらかで表示します。データ ファイル ツールバーを使用すると、この 2 つの文字セットの間で切り替えができます (図 11-5 参照)。このツールバーは、現在使用されている文字セットとキーを示します。

データ ファイル ツールバーを表示するには、[ビュー] メニューから [ドッキングできるウィンドウ] を選択し、さらに「データ ファイル ツールバー」チェック ボックスを選択します。

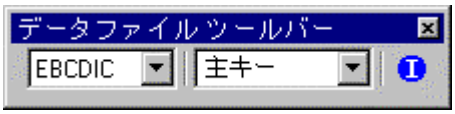


図 11-5 データ ファイル ツールバー

ファイル表示形式が正しいかどうかを確認するには、表示形式情報で空白文字を探します。EBCDIC 形式のファイルを ANSI 形式で表示すると、空白文字 (EBCDIC コード h"40") は "@" として表示されます。 一方、ANSI 形

式のファイルを EBCDIC 形式で表示すると、空白文字 (ANSI コード h"20") は英数字以外の文字として表示されます。この場合、実際に表示される文字はオペレーティング システムの文字セットによって異なります。

### 11.8.3 索引ファイルを表示するための参照キーの変更

デフォルトでは、データ ファイル エディタは、主キーの順番で索引ファイルのレコードを表示します。ファイルに副キーがある場合には、別の参照キーを選択することにより表示順序を変更することができます。副キーを選択し、索引ファイルを表示するためには、データ ファイル ツールバーを使用します (図 11-5 参照)。データ ファイル ツールバーには、現在選択されているキーと文字セットが表示されます。

### 11.8.4 順ファイル以外のファイルの表示

ファイル中で移動するには、次の方法を使用します。

- レコードの追加 (索引ファイルと相対ファイル)
- キーの変更 (索引ファイルのみ)
- キーの検索 (索引ファイルのみ)
- 現在の参照キーの変更 (索引ファイルのみ)

上記のどれかの方法を使用してファイル内で移動すると、NetExpress ウィンドウの最下部にあるステータス バーに表示される行番号 (レコード番号) は 0 から始まります。列番号もまた、行番号に対応して、 0 から始まります。

その後で、ファイルにある前のレコードへ移動すると、各レコードが移動するたびに行番号が 1 ずつ減り、負の番号として表示されます。

ファイルにある後のレコードに移動すると、各レコードが移動するたびに行番号が 1 ずつ増えます。

## 11.9 データの編集

Net Express では、次の操作が可能です。

- 編集するレコードの検索
- レコード レベル、またはフィールド レベルでのデータ編集
- 16 進データの編集
- フィールドとレコードの初期化
- 可変長ファイルのレコード長の変更

注記:

1. ファイルを誤って編集してしまった場合、編集ウィンドウを閉じ、エクスプローラーなどのファイル操作ツールを使用してバックアップ ファイルを編集ファイルにコピーすると、バックアップからファイルを復元することができます。ファイルに索引を作成する場合、索引とデータを確実に同期させるために、.ibk ファイルもコピーする必要があります。
2. レコードを選択すると、データ ファイル エディタがデータを保存します。新しいレコードを選択するまで、このデータはいつでも復元することができます。

---

### 11.9.1 データの検索

編集する特定のレコードを移動するには、いくつかの方法があります。

- ファイル内で順番に移動する (COBOL プログラムを使用してファイル データを連続して読み込む場合)。
- ファイル内で、0 から始まる行番号と列番号で定義された特定の位置に直接ジャンプする。
- 特定のレコードまたは文字列を検索する (詳細については「ファイルの検索」を参照してください)。

### 11.9.2 レコードの編集

レコード ビューで、データの上書き、挿入、または削除を行うことにより、データを編集することができます。

レコード ビューでデータの挿入や削除を行うと、すべてのデータがずれ、挿入箇所や削除箇所以降のフィールドが破損します。これを避けるためには、上書きモードに設定してください。

[更新を警告] が選択された場合、更新警告メッセージが表示されます。

これらの設定に関する詳細については、「データ ツールの設定」の項を参照してください。

### 11.9.3 フィールドの編集

フィールドの英数字データまたは数字データを直接編集することができます。

編集に使用できるモードは次のとおりです。

- 数字フィールド - 上書きモードのみ使用可。

  有効化された数字フィールドは、数値データしか受け付けません。

  数字を削除すると、各桁 が 0 にセットされます。

- 英数字フィールド - 挿入モードのみ使用可

フィールドに空白文字がある場合にのみ、新しい文字を挿入することができます。例えば、PIC X(4) フィールドにすでに4文字入力されている場合、これらを削除し、新しいデータに置き換えます。

英数字を削除すると、フィールドの右端に空白文字が挿入されます。

[更新を警告] が選択されている場合、更新警告メッセージが表示されます(「データ ツールの設定」を参照してください。)。

---

注記:

- フィールドのすべての桁にアスタリスク (*) が表示されることがあります。これは無効なデータを表します。このフィールドを選択すると、フィールドを初期化するかどうかをたずねるメッセージが表示されます。
- 索引ファイルの編集中には、キー値の変更に注意する必要があります。
    - 重複したキー値を使用できる場合以外は、1 つのレコードに同じ値をもつ複数のキーを設定しないでください。また、1 つのファイル内で主キーを複写しないでください。
    - キーを含むレコード領域を編集する場合、ファイルのレコードの論理的な位置を変更している可能性があるので注意してください。

---

### 11.9.4 16 進値の編集

16 進値が表示されている場合、この値を直接編集することができます。これにより、ファイルにテスト用の無効なデータを入れることができます。

各バイトの 16 進値は、編集メイン ウィンドウの下にあるウィンドウに表示されます。各文字の 16 進表現は、すぐ下の 2 行にレコード レベルとフィールド レベルで表示されます。文字とそれぞれの16進値の両方を編集することができます。

図 11-6 は、16 進値で表現されたファイルを示します。

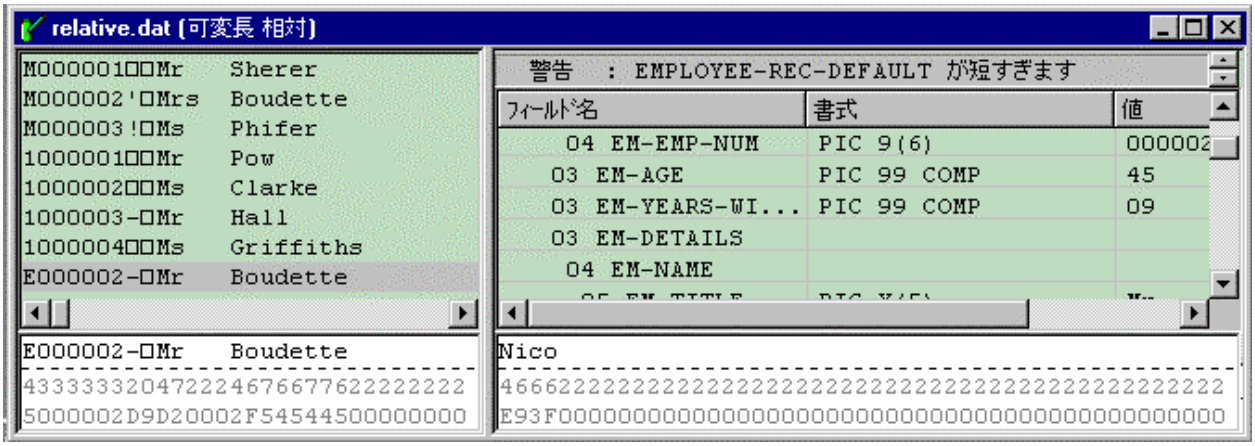


図 11-6 16 進値で表現されたファイル

### 11.9.5 データの初期化

フィールドを初期化すると、フィールド全体を空白文字または 0 にリセットすることができます。また、レコードのすべてのフィールドを初期化することもできます。英数字フィールドは空白文字に初期化され、数値フィールドは 0 に初期化されます。

条件付きレコード レイアウトを作成した場合 (「レコード レイアウトの種類」を参照してください。)、条件をテストするために使用されたフィールドはレコードの初期化により影響を受けることはありません。

また、ファイルに索引を作成しても、レコードのキーはレコードの初期化により影響を受けることはありません。

### 11.9.6 可変長ファイル レコードの長さの変更

可変長ファイル レコードの長さは、変更することができます。レコードの長さを短くする場合、レコードを新しい長さに合わせるために、レコードの右端が切り捨てられます。一方、レコードの長さを長くした場合、現在の埋め込みバイトが追加バイトに書き込まれます。

## 11.10 ファイルの検索

データ ファイル エディタでは、次のような検索方法を使用できます。

- 特定の文字列を検出するために、データ ファイルを検索することができます。
- ファイルの中で特定の文字列を検索し、検出した文字列を指定した別の文字列に置き換えることができます。置換方法は、次のとおりです。
    - ファイルの末尾に向かって置換する。
    - ファイルの先頭に向かって置換する。
    - ファイル中の検索文字列をすべて置換する。

検索文字列よりも置換文字列の方が短いと、空のバイトに埋め込み文字が書き込まれます。検索文字列よりも置換文字列の方が長いと切り捨てられます。このようにして、レコード中のフィールド位置の整合性が維持されます。

- 現在の参照キーを検索することができます (索引ファイルのみ)。

データ ファイルとレコード レイアウト ファイルを一緒に読み込んだ場合、または、これらを関連付けた場合、編集ウィンドウの右側のペインに検索結果が強調表示されます (「レコード レイアウト」の項を参照してください。)。

どの検索方法を使用する場合も、データ ファイル エディタはまず選択した検索方向で、検索条件と一致する最初のレコードを選択します。

一致する項目がない場合には、「現在の編集位置以降に*検索文字列* を検出できません」というメッセージが表示されます。

---

注記: 計算用 (COMPUTATIONAL) の数字キーを検索する場合には、16 進値を入力する必要があります。データ ファイル エディタは、レコード レイアウト ファイルが存在する場合でも、[キーの検索] ダイアログ ボックスに入力されたデータを書式設定しません。これは、キーが、それぞれ異なる書式をもつ可能性がある複数のフィールドにまたがることがあるためです。

---

## 11.11 データの印刷

データ ファイルは、書式設定されていないビューと書式設定されているビューのどちらでも印刷することができます。印刷する前にデータをプレビューすることもできます。レコード レイアウト ファイルを読み込んでいない場合、データの印刷やプレビューは、書式設定されていないビューでしか行えません (詳細については、「レコード レイアウト」の項を参照してください)。

### 11.11.1 書式設定されていないビューの印刷

書式設定されていないビューを印刷またはプレビューする場合には、次の内容を表示できます。

- 現在のレコードを印刷する。
- すべてのレコードを印刷する。
- 指定範囲のレコードを印刷する。この場合、ビューとステータス バーの両方で、レコード番号は 0 から始まる。
- 各レコードの全体を印刷する。

- 各レコードの一部を印刷する。バイト オフセットもまた 0 から始まる。
- バイト位置を示すルーラーを印刷する。ルーラー位置は 1 から始まります。
- レコード番号を印刷する。
- 16進値を印刷する。
- ヘッダーとフッターのどちらか一方、または両方に、ページ番号を付けて印刷する。また、ヘッダーとフッターのどちらか一方、または両方に、ページ番号を付けないで印刷することもできる。

レコードをプレビューする場合は、表示倍率を変更することができます。

---

注記:

- 印刷プレビューの場合、使用可能なメモリ容量によって印刷プレビューのサイズが制限されます。印刷プレビュー メモリを準備している間にメモリが足りなくなると、エラー メッセージが表示されます。ファイルの印刷では、そのような制限がありません。
- 印刷プレビューが表示されると、NetExpress ウィンドウの最下部にあるステータス バーにページ番号が表示されます。

---

## 11.11.2 書式設定されたビューの印刷

書式設定されたビューのデータをプレビューまたは印刷する場合に使用できる機能は、次の相違点を除き、書式設定されていないビューと同じです。

- 書式設定されたレコードに対してレコード レイアウト ファイルを読み込んでいる場合は、フィールド レベルでしかプレビューまたは印刷することができません。詳細については、「レコード レイアウト」の項を参照してください。
- 一致する使用可能なレイアウトがないレコード、または、無効な OCCURS DEPENDING ON RANGE を含むレコードが検出された場合、出力にエラーメッセージが含まれます。
- また、選択できない内容は、次のとおりです。
    - ルーラー
    - レコード番号

- 16進データ
- レコードの一部

## 11.12 レコード レイアウト

この章ではレコード レイアウトについてのバックグラウンド情報を説明します。レコード レイアウトの作成方法、およびレコード レイアウトをデータ ファイルと関連付ける方法を検索するには、NetExpress オンライン ヘルプ ファイルにあるヘルプを参照してください (ヘルプ トピックのボタンをクリックし、[目次] タブがアクティブであることを確認してから、[開発環境]、[データ ツール]、[レコード レイアウト エディタ] を選択してください)。

詳細については、『*入門書*』の「データ ファイルの維持と作成」を参照してください。この章には、レコード レイアウト ファイルを作成する方法の一部としてチュートリアルがあります。

レコード レイアウトは、データレコードの構造を表現するものです。データ ファイルに使用可能なレコード レイアウトがある場合、レコードのフィールド レベルでデータ ファイルを参照し、編集します。また、非表示フィールドに格納されたデータの数字を表示してください。COBOL プログラムには、レコード レイアウトが含まれます。COBOL レコード レイアウトの例を次に示します。

```
01 REC.
   02 EMP-CODE.
      03 EMP-POSITION        PIC X.
      03 EMP-NUM             PIC 9(6).
   02 EMP-AGE                PIC 99 COMP.
  02 EMP-YEARS-WITH-COMPANY  PIC 99 COMP.
```

この例では、EMP-AGE と EMP-YEARS-WITH-COMPANY が非表示の数字フィールドです。

データのレコード レイアウトを 1 つまたは複数含むレコード レイアウト (.str) ファイルを作成するには、レコード レイアウト エディタを使用します。レコード レイアウト ファイルには、データ ファイルのレコード、各フィールドの名前、長さ、ピクチャなどの情報が含まれます。データ ファイルに複数の種類のレコードが含まれる場合には、1 つのレコード レイアウト ファイルにこれらの複数のレコード レイアウトを取り込むことができます。

特定のデータ ファイルのレコード レイアウト ファイルを作成し、保存しておくと 、データ ファイルの操作を行うときにいつでも使用することができます。COBOL プログラムでレコード レイアウトを変更したり、新規条件付きレコード レイアウトを追加したいという場合には、レコード レイアウト ファイルを再作成するだけですみます。

レコード レイアウト エディタは、レコード レイアウト ファイルを作成するために、コンパイルした COBOL プログラムのデバッグ情報 (.idy) ファイルを使用します。関連付けた .idy ファイルがないかぎり、レコード レイアウト ファイルを作成することはできません。デバッグのためにプログラムをコンパイルすると、.idyを作成できます。COBOL レコード レイアウトに FILLER 項目が含まれている場合は、INCLUDE-FILLER コンパ イラ指令でコ

ンパイルし、レコード レイアウト ファイルに FILLER を含める必要があります。(デバッグためのコンパイルに関する詳細は、オンラインヘルプの「*開発環境*」のトピックを参照してください。)

レコード レイアウト ファイルは、データ ファイル コンバータでも使用されます (「データ ファイルの変換」の項を参照してください)。

---

注記: データ ファイルが非常に単純なレコード形式である場合には、データの書式設定されたビューに対して必ずしもレコード レイアウト ファイルを作成する必要はありません。

---

### 11.12.1 レコード レイアウトの種類

レコード レイアウトには、次の 2 つの種類があります。

- デフォルトのレコード レイアウト。このレイアウトは、1 つしか設定できません。条件は含まず、他のレコード レイアウトが使用できない場合や、他のレコード レイアウトの条件と合うものがない場合に使用されます。
- 条件付きレコード レイアウト。この種類のレイアウトは複数存在する場合があります。ファイルの中に異なるレコード定義がある場合、データ ファイル エディタは使用するレコード レイアウトを判断するために条件付きフィールドを使用し、これらの相違を識別します。一般的に、データ ファイルは異なるレコード レイアウトを使用して作成され、各レイアウトはこの条件が合うときだけに使用されます。

### 11.12.2 レコード レイアウトの使用方法

データ ファイルと一緒にレコード レイアウト ファイルを読み込んだ場合、また、これらを関連付けた場合、データ ファイル エディタ ウィンドウの右側のペインに書式設定したレコード データをフィールド レベルで表示することができます(「データ ファイル エディタ ウィンドウ」の項を参照してください。)。

一致する使用可能なレイアウトがないレコード、または、無効な OCCURS DEPENDING ON RANGE を含むレコードが検出された場合、フィールド ビューペインとステータス バーに警告メッセージが表示されます。

データ ファイルとレコード レイアウト ファイルを同じディレクトリに保存し、同じ基本名を使用することが必要です。

## 11.13 データ ファイルの変換

データ ファイル コンバータを使用して、データ ファイルの編成や形式を変換することができます。

ある形式から別の形式にファイルを変換すると、データを再構築できます。例えば、順ファイルから索引ファイルに変換したり、アプリケーションの変更に伴い副索引を追加することができます。

また、データ ファイル コンバータは、EBCDIC と ANSI の間で簡単にデータ変換を行えるので、作成済みのデータを再作成したり、特別な変換プログラムを作成する必要がありません。レコードに計算用データのような文字以外のデータ項目が含まれる場合、レコード レイアウトを使用して、変換するデータ項目と変換しないデータ項目を定義することができます。 レコード レイアウト ファイルの作成方法に関する詳細は、「レコード レイアウト」の項を参照してください。

データ ファイル コンバータを使用すると、レコード レイアウト記述により S/370 形式と IEEE 形式の間で内部浮動小数点形式を変換することもできます。

また、新しいファイルの最小レコード長と最大レコード長を増減することも可能です。

- 最小レコード長を長くした場合に、新しいファイルの最小長よりも短いソースファイル レコードがあると、新しいファイルのレコードは、新しいファイルの再小長に達するまで適切な埋め込みバイトが書き込まれます。
- 最大レコード長を長くする場合、新しいファイルのレコードに適切な埋め込みバイトが書き込まれます。
- 最大レコード長を短くする場合、新しいファイルのレコードでは新しい最大長に合わせて右端が切り捨てられます。

---

注記: 順ファイルを固有のキーをもつ索引ファイルに変換しようとしたときに、一部のレコードに重複するキー値が含まれていると、データ ファイル コンバータはこれらのレコードを無視し、変換しません。変換処理の最後に、データ ファイル コンバータは処理しなかったレコードの数を示します。

---

ファイル変換では、次のような変換が可能です。

- メインフレームから PC にダウンロードされたファイルの変換
- メインフレームから PC にダウンロードされた可変長の VRECGEN 形式ファイルの変換 (下記を参照)
- メインフレームに送信するために行う PC データ ファイルから VRECGEN2 形式ファイルへの変換
- ANSI データ形式ファイルと EBCDIC データ形式ファイルの間の変換
- 浮動小数点形式
- メインフレームと PC の間で行う固定長レポート ファイルの形式変換

メインフレームのファイルは、PC に転送するときにバイナリ形式にする必要があります。変換前のファイルが可変長ファイルである場合、システム管理者に、メインフレームのプログラム VRECGEN を使用して、メインフレームで最初に再フォーマットするように依頼することができます。このファイルは、データ ファイル コンバータを使用

して処理することができます。 ファイルをメインフレームにアップロードする場合、まず、データ ファイル コンバータを使用して、これを VRECGEN2 形式に変換します。次に、管理者に対して、メインフレーム プログラム VRECGEN2 を使用してメインフレーム用にデータを再フォーマットするように指示することが必要です。

レポート ファイルを変換することも必要になります。このレポート ファイルには、メインフレームで作成してから PC にダウンロードしたレポート ファイルも、PC で実行されるメインフレーム アプリケーションで作成したレポート ファイルも含まれます。メインフレーム レポート ファイルには、キャッリッジ制御文字が含まれます。データ ファイル コンバータは、キャッリッジ制御文字を PC で印字可能な形式に変換します。出力ファイルがメインフレーム ファイルであるとき、レコード長にはキャッリッジ制御文字も含まれます。出力ファイルが PC 印刷ファイルであるとき、レコード長はデータ長だけになります。

実際には PC で作成された入力ファイルをメインフレーム出力書式ファイルとして指定した場合、エラー メッセージ 139「**入力ファイルは、"本物の" メイン フレーム レポート形式ではありません。**」が表示されます。このようなエラーを防ぐには、通常のメイン フレーム ダイアレクトを使用して EBCDIC レポート ファイルを作成するようにします。

メインフレーム レポート ファイルは、PC 印刷ファイル以外に変換することができません。同様に、PC 印刷ファイルは、メインフレーム レポート ファイル以外に変換することができません。メインフレーム レポート ファイルは、EBCDIC 形式のコード ファイルでなければならず、PC 印刷ファイルは ANSI 形式のコード ファイルでなければなりません。

入力ファイル名を入力するときに、データ ファイル コンバータは、入力ファイルの詳細を読み込むために、「ファイルの種類とレコード長」の項で概説した処理を行います。また、出力ファイルの名前を指定することが必要です。データ ファイル コンバータが選択するデフォルトの編成とレコード書式の値は、変更することができます。ファイルの詳細が使用できない場合には、この情報を入力する必要があります。

出力ファイルの編成として索引ファイルを選択した場合、新しいファイルのキーを定義する必要があります (「キーの定義」の項を参照してください)。

VRECGEN、VRECGEN2、およびコマンド行インターフェイスの使用方法についての詳細は、「ファイル変換ユーティリティ」の章を参照してください。

## 11.14 ファイル索引の修正

破損した索引ファイルの索引をリビルドするには、ファイル索引の修正を使用します。 詳細については、「REBUILD」の章の「ファイル索引の修正」を参照してください。

# 索引