(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3786419号
(P3786419)
(45)発行日　平成18年6月14日(2006.6.14)　(24)登録日　平成18年3月31日(2006.3.31)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 511C

請求項の数 6 (全 23 頁)

(21)出願番号　特願平5-519231
(86)(22)出願日　平成5年1月27日(1993.1.27)
(65)公表番号　特表平7-506446
(43)公表日　平成7年7月13日(1995.7.13)
(86)国際出願番号　PCT/US1993/000732
(87)国際公開番号　WO1993/022734
(87)国際公開日　平成5年11月11日(1993.11.11)
審査請求日　平成12年1月25日(2000.1.25)
(31)優先権主張番号　877,409
(32)優先日　平成4年5月1日(1992.5.1)
(33)優先権主張国　米国(US)

(73)特許権者　592045429
ウォング・ラボラトリーズ・インコーポレーテッド
WANG LABORATORIES INCORPORATED
アメリカ合衆国マサチューセッツ州０１８２１．ビレリカ，テクノロジー・パーク・ドライブ６００
(74)代理人　100089705
弁理士　社本　一夫
(74)代理人　100071124
弁理士　今井　庄亮
(74)代理人　100076691
弁理士　増井　忠弐

最終頁に続く

(54)【発明の名称】テーブル駆動ファイル・パーサーのためのコンピュータの方法及び装置

(57)【特許請求の範囲】

【請求項１】

（ｉ）異なるファイル・フォーマットの複数のファイル（１７）を保持する記憶メモリ（１９）と、（ｉｉ）メイン・メモリにおいて実行可能な複数の異なるアプリケーション（１５）とを有するデジタル・プロセッサ装置（１１）において、この装置は、

プロセッサ手段を備え、

前記デジタル・プロセッサ（１１）の作業メモリに記憶され、種々のファイル・フォーマットの成分部分を示すテーブル（２３）であって、異なるファイル・フォーマットのそれぞれに対して異なるエントリを有するが、それぞれのファイル・フォーマットの成分部分を示すために共通の表現の組を各エントリにおいて用いる、テーブル（２３）を更に備え、

前記プロセッサ手段は、前記記憶メモリ（１９）内の種々のフォーマットのファイルからパースされた形又はパースされていない形でアプリケーションによりデータを取り出すことを可能にするインターフェース（２１）を介して、前記テーブル（２３）に結合されるものであり、前記アプリケーションによって用いられるファイル・フォーマットとは異なるファイル・フォーマットを有する所望のファイルに対して、

（ａ）前記所望のファイルのファイル・フォーマットに対応する前記テーブル（２３）のエントリにアクセスし、

（ｂ）前記所望のファイルの前記成分部分を、前記所望のファイルの前記ファイル・フォーマットの前記成分部分を示すために用いられる前記共通の表現の組に従って前記テーブ

ルの前記エントリに示された成分部分へと決定し解釈する処理を行い、
（ｃ）前記アプリケーションによる前記所望のファイルのアクセスを提供するために、前記共通の表現の組に関して前記テーブルに示されるように前記成分部分を前記アプリケーションのファイル・フォーマットへと組み立てる処理を行い、前記プロセッサ手段は、異なるファイル・フォーマットに関して同じ前記テーブルをアクセスするが該テーブルの異なるエントリをアクセスし、各エントリは、それぞれのファイル・フォーマットの成分部分を示すために前記共通の表現の組を用いるものである、
ように構成される、
ことを特徴とする装置。
【請求項２】
請求項１に記載の装置において、前記テーブル（２３）は、異なるファイル・タイプに従って異なるファイル・フォーマットのそれぞれを相互参照するものであり、前記デジタル・プロセッサ（１１）の種々のファイルにはそれぞれのファイル・タイプが指定される、装置。
【請求項３】
請求項１に記載の装置において、前記異なるファイル・フォーマットはスプレッドシート・フォーマットを含む、装置。
【請求項４】
（ｉ）異なるファイル・フォーマットの複数のファイル（１７）を保持する記憶メモリ（１９）と、（ｉｉ）メイン・メモリにおいて実行可能な複数の異なるアプリケーション（１５）とを有するデジタル・プロセッサ（１１）における、異なるフォーマットのファイルにアクセスする方法であって、
プロセッサ手段を提供するステップを備え、更に、
前記デジタル・プロセッサ（１１）の作業メモリに記憶され、異なるファイル・フォーマットの成分部分を示し、且つそれぞれのファイル・フォーマットの成分部分を示すための各エントリに共通の表現の組を有するテーブルを提供するステップであって、該テーブルが、異なるファイル・フォーマットのそれぞれに対して異なるエントリを有するが、それぞれのファイル・フォーマットの成分部分を示すために各エントリにおいて前記共通の表現の組を用いられるものである、テーブルを提供するステップと、
前記記憶メモリ（１９）内の種々のフォーマットのファイルからパースされた形又はパースされていない形でアプリケーションによりデータを取り出すことを可能にするインターフェース（２１）を介して、前記プロセッサ手段を前記テーブル（２３）に結合させるステップと、
前記アプリケーションによって用いられるファイル・フォーマットとは異なるファイル・フォーマットを有する所望のファイルに対して、前記プロセッサ手段を動作させるステップであって、
（ａ）前記所望のファイルのファイル・フォーマットに対応する前記テーブル（２３）のエントリにアクセスするステップと、
（ｂ）前記所望のファイルの前記成分部分を、前記所望のファイルの前記ファイル・フォーマットの前記成分部分を示すために用いられる前記共通の表現の組に従って前記テーブルの前記エントリに示された成分部分へと決定し解釈するステップと、
（ｃ）前記アプリケーションによる前記所望のファイルのアクセスを提供するために、前記共通の表現の組に関して前記テーブルに示されるように前記成分部分を前記アプリケーションのファイル・フォーマットへと組み立てるステップであって、異なるファイル・フォーマットに関して、前記プロセッサ手段が、同じ前記テーブルをアクセスするが該テーブルの異なるエントリをアクセスし、それぞれのファイル・フォーマットの成分部分を示すために各エントリにおいて前記共通の表現の組を用いる、ステップと
を含む前記プロセッサ手段を動作させるステップと
を備えることを特徴とする方法。
【請求項５】

請求項４に記載の方法において、前記テーブル（２３）を提供する前記ステップは、異なるファイル・タイプに従って異なるファイル・フォーマットのそれぞれを相互参照する、前記テーブル内における相互参照を提供するステップを含み、前記デジタル・プロセッサ（１１）の異なるファイルにはそれぞれのファイル・タイプが指定される、方法。

【請求項６】

請求項４に記載の方法において、前記異なるファイル・フォーマットはスプレッドシート・フォーマットを含む、方法。

【発明の詳細な説明】

発明の背景

コンピュータ・システムにおいては、アプリケーションと呼ばれるユーザ・プログラムが、デジタル・プロセッサによって実行される。デジタル・プロセッサは、例えば、メインフレーム、ネットワークのホスト・プロセッサ、スタンドアローンのＰＣ等であり得る。アプリケーションは、例えば、スプレッドシート・プログラムやワープロ・プログラムなどである。異なるアプリケーションの実行によって、異なるファイル・フォーマットのファイルが作成される。すなわち、ワープロ・アプリケーションはテキスト記憶フォーマットのファイルを作成するのに対し、スプレッドシート・アプリケーションは別のフォーマットのファイルを作成し、更に、異なるワードプロセシング・アプリケーションの中では、作成されたファイルに対して異なるテキスト記憶フォーマットが用いられる。

１つのアプリケーション（要求側のアプリケーション）が未知のファイル・フォーマットのファイルにアクセスするためには、（ａ）要求側のアプリケーションが所望のファイルのファイル・フォーマットを知らなければならないか、又は、（ｂ）要求側のアプリケーションがその未知のファイル・フォーマットに従って所望のファイルにアクセスするインターフェースを用いなければならないか、のいずれかである。結果としては、適当なインターフェースが要求側アプリケーションにとって入手可能でなければならず、すなわち、要求側アプリケーションが種々の未知であるファイル・フォーマットのファイルにアクセスするには、種々のインターフェースが入手可能でなければならない。

米国特許第５０２１９９５号は、第１のタイプを有するソース・ファイルから論理レコードを読み出し、その論理レコードの内容を表す総称的な形式を生じるコンピュータ装置を開示している。この総称的な形式は、いかなるファイル・タイプにも依存しない。よって、この装置によれば、この総称的な形式から、第２のタイプを有する（すなわち、ソース・ファイルとはことなるタイプの）目的ファイルのための論理レコードを、更に作成できる。この特許は、更に、ファイル・タイプの４つの基本的なクラス、すなわち、ストリーム・ファイル・タイプ、レコード・ファイル・タイプ、レコード及びフィールド・ファイル・タイプ、文書（ドキュメント）ファイル・タイプそれぞれのファイル・タイプから総称的な形式を作成する方法も開示している。

種々のインターフェースを入手可能とするためのもう１つの解答は、１つのインターフェース内の特定のファイル・フォーマットそれぞれに対して、個別の、開く、閉じる、読み出す、書き込むためのアクセス・ルーチンを含んでいることであった。１つのインターフェースのために特定されたもの以外の異なるファイル・フォーマットに対しては、適切な、開く、閉じる、読み出す、書き込むルーチンが開発され、ユーザ／カスタマの要求に応じて、そのインターフェースに付加される。これは、付加的で／新たなファイル・フォーマットに応じて、そのインターフェースへの必要な改変を行うための開発時間を要求する。それにより、開発時間が長ければ長いほど、カスタマの不満足を生じるターンアラウンド・タイムが長くなる。

従って、異なるファイル・フォーマットの間でのインターフェースを改良する必要が存在する。

発明の概要

本発明は、従来技術の問題点を解決するインターフェースを提供する。更に詳しくは、本発明は、１つのインターフェースによってサポートされる付加的又は新たなファイル・フォーマットのための開発時間を短縮するインターフェースを提供する。

本発明の出願人は、そのような種々のインターフェースに対して、異なるファイル・フォーマットに対応する開く、閉じる、読み出す、書き込むアクセス・ルーチンにかなりの冗長度があることを発見した。これは、あるファイル・フォーマット間、例えば、異なるワードプロセシング・アプリケーションによって作成される異なるテキスト記憶フォーマットの間での翻訳をするインターフェースについて特に正しい。アプリケーションが異なる未知のフォーマットのファイルにアクセスして、新たなファイル・フォーマットにアクセスする能力を付加するための開発時間を短絡するような方法でこの冗長度を利用することを可能にするために、本出願人は、複数のファイル・フォーマットを定義する共通の表現形式を用いる方法及び手段を開発した。この共通の方法及び手段が、本発明のテーブル・トリドン・パーサーの基礎である。

ここで使用する「パーサー（parser）」の語は、１つのファイル・フォーマットの成分部分（component parts）を決定し解釈する方法及び手段を称する。

特に、本発明は、異なるファイル・フォーマットのそれぞれに対して該ファイル・フォーマットの成分部分を指示するエントリを有するテーブルと、そのテーブルに結合され少なくとも１つのアプリケーションと異なるファイル・フォーマットのファイルとの間でインターフェースするプロセッサ手段と、を備えたコンピュータ装置に関する。当該アプリケーションにとって未知であるそれぞれのファイル・フォーマットを有する所望のファイルに対して、プロセッサ手段は、テーブルにアクセスし、それぞれのファイル・フォーマットに対応するテーブルのエントリに指示された成分部分に従って所望のファイルを分解（parse）する。そうして、プロセッサ手段は、当該アプリケーションがその所望のファイルにアクセスすることを可能にする。本発明の１つの特徴によれば、テーブルは、異なるファイルのタイプに従って、異なるファイル・フォーマットのそれぞれをクロスリファレンスするが、この際、異なるファイルには対応するファイルのタイプが指定されている。すなわち、テーブルは、（予め定義されたファイル・タイプの組からの）ファイルのタイプに従って、異なるファイル・フォーマットをカテゴリに分ける。それぞれのファイルのタイプに対して、テーブルは、そのタイプのファイルのそれぞれの成分部分の指示と記述とを提供する。更に、テーブルは、各成分部分の記述が共通のフォーマットになっているように構成されている。

【図面の簡単な説明】

本発明の以上の及びそれ以外の目的、特徴、効果は、以下に続く、図面に示した好適実施例のより詳細な説明から明らかになろう。図面では、異なる視点を通じて、同じ部分には類似の参照番号を付してある。これらの図面は、必ずしも一定の割合で縮小されてはおらず、本発明の原理を図解する点に重点が置かれている。

図１は、本発明の実施例であるコンピュータ・システムの概略図である。

図２は、図１の実施例で用いられているパーサー・テーブルのブロック図である。

図３ａ～図３ｆは、図２のパーサー・テーブルの作業記述子レコード（Working Descriptor Records）のブロック図である。

図４ａ～図４ｃは、図１の実施例の読み出し動作の流れ図である。

図５ａ～図５ｂは、図１の実施例の書き込み動作の流れ図である。

好適実施例の詳細な説明

図１には、本発明のデジタル・プロセッサの実施例が図解されている。デジタル・プロセッサ１１内では、ＣＰＵ（中央処理装置）１３がメイン・メモリ内の種々のアプリケーション１５を実行する。アプリケーション１５の実行によって、ファイル１７が作成される。ファイル１７を作成するアプリケーション１５の指示に下に、ＣＰＵ１３は、各ファイル１７を記憶メモリ１９に記憶する。各ファイル１７は、そのファイルを作成したそれに対応するアプリケーション１５に従ってフォーマットされている。その結果として、記憶メモリ１９内に記憶された別のファイル１７は、別のファイル・フォーマットを有する。アプリケーション例えばアプリケーション１５ａが、未知のファイル、すなわちアプリケーション１５ａによって作成されたファイル１７ａとは異なるファイル・フォーマットを有するファイル１７ｂ、ｃ、にアクセスする際には、ＣＰＵ１３は本発明のインターフェ

ース２１を用いる。インターフェース２１は、種々の異なるファイル・フォーマットへのアクセスを共通に許容する開く、閉じる、読み出す、書き込むアクセス・ルーチンの組を含む。更に正確には、インターフェース２１は、異なるファイル・フォーマットの異なる特性を特定するテーブル２３を参照する。よって、インターフェース２１は、テーブル２３から所望のファイル・フォーマットの詳細を読み出し、それらの詳細を用いて所望のファイルを開き、そこから要求側のアプリケーション１５のためのデータを検索する。

よって、テーブル２３は、テーブル内に表された異なるファイル・フォーマットそれぞれに対してデータをいかにして抽出するかを特定する。これは、図２に図解され次に説明するテーブル２３の好適実施例において達成される。簡単に概観すると、本発明の好適実施例では、テーブル２３において表された異なるファイル・フォーマットは、（ファイルのタイプの組が予め定義してあり）ファイルのタイプに従ってカテゴリに分けられる。それぞれのファイルのタイプに対し、テーブル２３は、成分部分の指示と各成分部分の記述を提供する。更に、テーブル２３は、それぞれの記述が共通のフォーマットでなされているように構成されている。すなわち、１つのファイル・フォーマットの記述から次の記述では、１つの共通のフォーマットすなわち「メタ言語」（metalanguage）が使用されている。

結果として、インターフェース２１は、テーブル２３内に記憶されているソース・ファイル・フォーマットのメタ言語の記述に従ってソース・フォーマット・ファイルを成分部分に分解するテーブル・ドリブン・ファイル・パーサーとして働く。成分部分は、次に、やはりテーブル２３内に記憶されているターゲット・ファイル・フォーマットのメタ言語の記述に従って、直接にターゲット・フォーマット・ファイルに組み立てられる（assembled）。

このアプローチによって、テーブル２３内の新たなファイル・フォーマットの記述を付加することは容易になり、新たにサポートされるファイル・フォーマットから完全な翻訳プログラム（translator）を開発する必要性がなくなる。その代わりに、この新たなファイル・フォーマットのメタ言語の記述だけが、開発されテーブル２３に付加される必要がある。

図２に図解されるように、テーブル２３は、それぞれのファイル・タイプに対して１つのエントリ・レコード２７を従えたヘッダ・レコード２５とエントリ・レコード２７に対応する記述レコード２９とによって構成されている。

ヘッダ・レコード２５は、好ましくは１０バイトの長さである。最初の８バイトは、テーブル２３の識別及び（又は）バージョン番号を与える。ヘッダ・レコード２５の残りの２バイトは、テーブル２３内に表されるファイル・タイプの数を与える。

各エントリ・レコード２７は、好ましくは、２０バイトの長さである。最初の４バイトは、テーブル２３の最初からエントリ・レコード２７の名前部分において特定されるファイル・タイプに対応する記述レコード２９へのオフセットを指示する。名前部分が、エントリ・レコード２７の残りの１６バイトを占める。よって、テーブル２３内のエントリ・レコード２７の列（series）は、テーブルの位置を指示する及びクロスリファレンスするメンバーとして機能し、このメンバーは、記述レコード２９のメモリ位置と各記述レコードが対応するそれぞれのファイル・タイプとを指示する。

各記述レコード２９は長さが可変であるが、以下で説明し図３ａで図解する共通のフォーマットに従って配列されている。好適実施例では、記述レコード２９の第１の部分３５は、３５９バイトの長さであり、その記述レコード２９に対応するファイル・タイプの種々の特徴を示す。この第１の部分３５には一連の記述子レコード３１が続き、テキスト、数値（numeric）、リピート、主題のファイル・フォーマットを識別し記述するそれ以外の特性シーケンス、を特定する。

特に、第１の部分３５の第１のバイトは、フォーマット・スイッチ３３である。フォーマット・スイッチ３３では、最下位のオーダー・ビットは１に設定され、テキストがジャスティフィケーション文字（justification letters）内に含まれる（enclosed）ことを指定する。対応するファイル・フォーマットのテキストがジャスティフィケーション文字内

に含まれない場合には、そのビットはゼロに設定される。フォーマット・スイッチ３３の次のオーダー・ビットは１に設定され、ＰＣのファイルが含まれることを示し、それ以外の場合にはゼロに設定される。その次のオーダー・ビットは１に設定され、データ・レコードがこのタイプのファイルの中に挿入され得ることを示す。そして、レコードがこのタイプの既存のファイルに追加されるそれ以外の場合には、ゼロに設定される。その次のオーダー・ビットは１に設定され、ファイル・フォーマットが固定長のレコードを有することを示し、それ以外の場合にはゼロに設定される。その次のオーダー・ビットは１に設定され、含まれるファイル・フォーマットが圧縮されたレコードを有することを示し、それ以外の場合にはゼロに設定される。その次のオーダー・ビットは、ジャスティフィケーション・キャラクタの中にラベルが含まれるかどうかを指定する。その次のオーダー・ビットは、主題のファイル内のセルがロー及びコラムの順序からのものであるかどうかを指定する。

記述レコード２９の次のバイト３７は、デフォールト・コラム幅を指示する。記述レコード２９の次のバイト３９は、含まれるファイル・フォーマットのキャラクタの組を指示する。記述レコード２９の次の２バイトは、最大のレコード長を指示する。記述レコード２９の次の４バイトは、含まれるファイル・タイプに対する最大のローであるローの限界４０を示す。記述レコード２９の次の２バイトは、そのファイル・タイプに対する最大のコラムであるコラムの限界３６を示す。

以上によれば、記述レコード２９の最初の１１バイトは、オーダーを示す３２バイトのアレー４１であり、そこで作業記述子レコード３１（すなわち、記述レコード２９の第１の部分３５に続く部分であり、以下で説明される）がサーチされる。サーチ・オーダー・アレー４１の最後のエントリには、１６進値（hex value）ＦＦが指定され、サーチ・オーダー・アレー４１の終了を意味する。

記述レコード２９の第１の部分３５の次の８バイトは、書き込みオーダーを示すアレー４３である。これは、作業記述子レコード３１が書き込まれるオーダーである。書き込みオーダー・アレー４３の最後のエントリには、１６進値ＦＦが指定され、アレー４３の終了を意味する。

記述レコード２９の次の４０バイトは、テーブル２３の最初の記述レコード２９ａの最初から交替ストリングまでのバイト数を示す。入力では、交替ストリングによって、ファイルの特定のキャラクタが所望のキャラクタと交替されることが可能になり、それ以外のキャラクタは、出力においてファイルの特定のキャラクタによって交替される。指示された数のバイト（図３ａのブロック４５）は、オフセットを交替ストリング（交替されるべきストリング）の始め、変化代替ストリングの始め、変化代替ストリングの終わりなどに、与える。

記述レコード２９の第１の部分３５の次の１２４バイトは、記述レコード２９の始めから作業記述子レコード３１へのオフセット位置を示し、記述レコード２９に対応するファイル・タイプのファイル・フォーマットを記述する。これらの作業記述子レコード３１に対するオフセットは、次の順序で記述レコード２９の４７（図３ａ）に現れる。

Header_Record
Format_Record
Range_Record
Column_Width_Record
BOD_Record (Beginning of Data_Record)
BOR_Record (Beginning of Row_Record)
Data_Delimiter
Empty_Cell_Record
Integer_Record
Decimal_Record
Text_Record
Label_Record

Data_Record
Time_Record
Formula_Record
Left_Justify
Center
Right_Justify
Repeat_char
EOR_Record（End of Row_Record）
EOF_Record（End of File_Record）
Extra_1
Extra_2
Extra_3
Extra_4
Extra_5
Password
Format_Mask_Record
Format_Index_Record
これらの作業記述子レコード３１それぞれのフォーマットは、以下で説明する。
記述レコード２９の第１の部分３５の次の８８バイト（図３ａの４９）においては、記述レコード２９から作業記述子レコード３１又はその一部への、対応するファイル・タイプのファイル・フォーマットの種々のデータ・タイプを定義するオフセットしたバイト数がある。データ・タイプのオフセット４９は、次の順序のデータ・タイプに配列されている。
DX_BYTE_INT
DX_UBYTE_INT
DX_BYTE_INT
DX_SHORT_INT
DX_USHORT_INT
DX_LONG_INT
DX_SHORT_FLOAT
DX_LONG_FLOAT
DX_INT_TEXT
DX_FLOAT_TEXT
DX_SCI_TEXT
DX_BINARY_TEXT
DX_OCTAL_TEXT
DX_HEX_TEXT
DX_TEXT
DX_LABEL
DX_FORMULA
次の３２バイト（図３ａの５０）は、データ・フォーマット・マスクを記述する記録へのオフセットを示す。オフセットは次の順序で現れる。すなわち、一般的な数値フォーマット（１０進（デシマル）の変数）
デシマルなし（no decimals）
デシマルなしのマネタリ
２デシマルの数値
２デシマルのマネタリ
デシマルなしのパーセント
２デシマルのパーセント
科学的概念

時間
日付
記述レコード２９の第１の３５の部分の次のバイト５１は、ファイル・タイプのコラム数に対する各レコードへのオフセットしたバイト数を示す。このオフセット情報は、記述レコード２９のFormat_Record作業記述子レコード３１から来る。
記述レコード２９の次のバイト５３は、コラム数のファイル・タイプを示す。すなわち、コラム数が、作業記述子レコード３１に記憶されている数値変数情報から示されるように記憶されているタイプである。
記述レコード２９の第１の部分の次のバイト５５は、そのコラム数に対するキャラクタ出力のデフォールト最小数を示す。次のバイト５７は、１０進小数点（decimal point）の右への認識された桁数を示す。ここで、コラム数のフォーマット・タイプは、１０進データとして示されている。
記述レコード２９の次の２バイトは、コラム調整（adjustment）５９である。このコラム調整の値は、正又は負の方向にデフォールト第１コラムを調整するのに用いられる。デフォールト第１コラムはゼロである。
記述レコード２９の次のバイト６１は、ファイル・タイプのローの数に対する各レコード内のオフセットしたバイト数を示す。この情報は、上述のFormat Record作業記述子レコード３１から得られる。記述レコード２９の第１の部分の次のバイト６３は、ローの数のファイル・タイプを示す。すなわち、ローの数が、作業記述子レコード３１の数値変数情報から示されるように記憶されているタイプである。記述レコード２９の次のバイト６５は、ローの数に対するキャラクタ出力のデフォールト最小数を示す。次のバイト６７は、１０進小数点の右への認識された桁数を示す。ここで、ローの数のファイル・タイプが１０進データである。
記述レコード２９の次の２バイト６９は、ロー調整を示し、これは、正又は負の方向にデフォールト第１ローを調整するのに用いられる。デフォールト第１ローは、好ましくは、ゼロに設定されている。
記述レコード２９の次のバイト７１は、記述レコード２９の初めからレコード長（主題のファイル・タイプの記録の長さ）を保持するアドレスへのオフセットしたバイト数を示す。このアドレス情報は、上述のFormat Recordに対する作業記述子レコード３１から得られる。次のバイト７３は、レコード長のフォーマット・タイプを示す。すなわち、レコード長が、作業記述子レコード３１の数値変数情報から示されるように記憶されているタイプである。
次のバイト７５は、レコード長に対するキャラクタ出力のデフォールト最小数を示す。次のバイト７７は、１０進小数点の右への認識された桁数を示す。ここで、レコード長のファイル・タイプは１０進データとして示されている。
記述レコード２９の次の２バイトは、レコード長調整７９を示す。この調整変数の値は、正又は負の方向にデフォールト・レコード長を調整するのに用いられる。
記述レコード２９の次のバイト８１は、記述レコード２９の初めからセルのディスプレイ幅を保持するアドレス（すなわちレコード）へのオフセットしたバイト数を示す。このアドレス情報は、上述のFormat Recordに対する作業記述子レコード３１から得られる。
次のバイト８３は、セル幅のフォーマット・タイプを示す。すなわち、セル幅が、作業記述子レコード３１の数値変数情報から示されるように記憶されているタイプである。
次のバイト８５は、セル幅に対するキャラクタ出力のデフォールト最小数を示す。次のバイト８７は、１０進小数点の右への認識された桁数を示す。ここで、セル幅のファイル・タイプは１０進データとして示されている。
記述レコード２９の第１の部分の最後の２バイト８９は、セル幅の調整を示す。この調整変数は、正又は負の方向にセル幅を調整するのに用いられる。好ましくは、デフォールト・セル幅は９又は大きなほうの出力されているストリングの長さである。
上述のように、記述レコード２９の残りの部分には、記述レコード２９の第１の部分の４７にリスト化されているHeader_Record、Format_Record、Range_Record等に対する一連の

作業記述子レコード３１がある。作業記述子レコード３１の内容に依存して、レコードは、図３ｂ～図３ｆに図解されている以下のレコード構造の１つに従う。
好適実施例では、図３ｂ～図３ｆの記述子レコード３１は、作業ファイル記述をテーブル２３のレコードに変換するテーブル・コンパイラによって発生する。作業ファイル記述の詳細は、図３ｂ～図３ｆのレコード構造の説明に含まれる。まとめると、作業ファイル記述の視覚的ファイルは、ファイル・フォーマットの記述において当初用いられる（たとえば、ソフトウェア記述者によって）。視覚的ファイルは、次に、テーブル・コンパイラによってコンパイルされ、テーブル２３、特に、図３ｂ～図３ｆの構造では、作業記述子レコード３１を生じる。
図３ｂを参照すると、主題のファイル・タイプのテキスト情報を含む作業記述子レコード３１に対しては、作業記述子レコード３１ａ（図３ｂ）は、レコード・タイプを示す第１のバイト９１を有する。好適実施例では、第１のバイト９１における０１の１６進値が、レコード・タイプは厳密整合（exact match）レコードのそれに等しいことを示している。すなわち、定義されたファイル・フォーマット内に常に一貫して存在し、データの限界を定める（delimit）ために厳密に整合されなければならない情報である。この情報は、レコード記述子キャラクタ、データ・デリミタ（delimiter）キャラクタ・レコード・デリミタ・キャラクタを含む。
第１のバイト９１における０２又は１２の１６進値が、レコード・タイプは「存在すれば整合」（match-if-present）レコードであることを示している。これは、定義されているファイル・フォーマット内に存在し得る情報を含む。すなわち、この情報は、可変長であるか、又は、この記述子レコード・タイプによって定義されるファイル・フォーマットの幾つかの記録から欠けている。この情報は、入力において走査され、しかし、整合が見つからない場合には無視される。出力では、この情報は、特定のものとしてレコードに書き込まれる。更に、１つの「存在すれば整合」のキャラクタ・グループが別のそのようなキャラクタ・グループに従う場合には、第１のバイト９１の１６進値は、１６進値０２と１２との間でトグルされる。
レコード・タイプの第１のバイト９１の１６進値は０４に設定され、「スキップ固定」（skip-fixed）レコードを示す。これは、定義されているファイル・フォーマット内に常に存在する一定の長さのレコード情報に対して用いられる。しかし、このレコード情報は、その長さは同じままで変化し得る。この情報は、入力において無視され、このデータ・タイプの長さは処理を促進するためにスキップされる。出力では、この情報は、特定のものとしてレコードに書き込まれる。このタイプの成分の例は、フォーマット・バージョン数、又は、ファイル作成日付である。
作業記述子レコード３１ａの次のバイトは、テキストの長さを示す。それに続くバイト９３は主題のテキストを提供する。特に、主題のテキストのプリント可能なキャラクタ（たとえば、アルファベットのキャラクタと、句読点）は、この技術分野で知られているように、アスキーキャラクタ・フォーマットで表現され、プリント不可能なキャラクタ（たとえば、キャリッジ・リターンや、ライン・フィード・キャラクタ）は、１６進値キャラクタ（たとえば、キャリッジ・リターンに対してはＯＸＯＤであり、ライン・フィード・キャラクタに対してはＯＸＯＡ）で表現される。当初は、対応する作業ファイル記述では、キャリッジ・リターンは、＼ＯＤによって示され、ライン・フィード・キャラクタは、＼ＯＡによって示される。また、主題のテキストの存在すれば整合のキャラクタは、カッコで囲まれている。カッコのキャラクタが主題のテキストで用いられる場合には、１６進値キャラクタとして特定されるか、又は、バックスラッシュが先行する。主題のテキストの任意のスキップ固定キャラクタは、四角いブラケットで囲まれている。四角いブラケットのキャラクタが主題のテキストで使用されている場合には、１６進値キャラクタとして表現されるか、又は、バックスラッシュが先行する。
主題のファイル・タイプの数値情報を運ぶ記述子レコード３１ｂに対しては、以下のレコード構造が、図３ｃで図解されるように使用される。記述子レコード３１ｂの第１のバイト９５は、レコード・タイプを示す。０９の１６進値が、レコード・タイプは、数値変数

の厳密整合レコードであることを示している。０Ａ、１Ａの１６進値は、レコード・タイプは数値変数の存在すれば整合型のレコードであり、連続する一連の「存在すれば整合」型キャラクタ・グループの間で区別する。０Ｃの１６進値は、「数値変数のスキップ固定」型のレコード・タイプを指示する。
数値変数情報は、コラム数、ロー数、セル幅、レコード長、ローの数、コラムの数、などを含み、しかし、これらに限定されない。当初は、対応する作業ファイル記述において、数値変数情報は、一般に次にフォーマットを有する。

\argument±constant(width.precision,type)

ここで、”±constant”と、”.precision”とは、オプショナルなデータであり、”argument”と、”width”と、”type”とは、以下で説明する。
次のバイト９７は、１６進値を保持し、主題の数値変数の引数（numeric-variable arguments）を示す。好適実施例では、バイト９７の１６進値と、主題のファイル・タイプで見いだされる対応する当初の作業ファイル記述数値変数の引数は、次のとおりである。

１６進値	数値変数の引数
０１	Ａ　初めのコラム数
０２	Ｂ　初めのロー数
０３	Ｃ　最後のコラム数
０４	Ｄ　最後のロー数
０５	Ｅ　全体のコラム・カウント
０６	Ｆ　全体のロー・カウント
０７	Ｇ　コラム数から
０８	Ｈ　コラム数へ
０９	Ｉ　整数データ位置
０Ａ	Ｊ　ジャスティフィケーション・データ位置
０Ｂ	Ｋ　ラベル・テキスト位置
０Ｃ	Ｌ　レコード長
０Ｄ	Ｍ　データ・レコード長
０Ｅ	Ｎ　１０進データ位置
０Ｆ	Ｏ　発生又はリピート・カウント
１０	Ｐ　公式（formula）数値位置
１１	Ｑ　公式位置
１２	Ｒ　公式長
１３	Ｓ　テキスト・データ位置
１４	Ｔ　テキスト長
１５	Ｕ　フィールド・タイプ
１６	Ｖ　１０進位置の数
１７	Ｗ　セル幅

１８	Ｘ　現在のコラム数
１９	Ｙ　現在のロー数
１Ａ	Ｚ　日付データ位置
１Ｂ	ａ　時間データ位置
１Ｃ	ｂ　現在のワークシート数
１Ｄ	ｃ　マスク・リピート・カウント
１Ｅ	ｄ　基準コラム
１Ｆ	ｅ　基準ロー
２０	ｆ　基準ワークシート
２１	ｇ　フォーマット・マスク
２２	ｈ　フォーマット・マスク・インデックス

引数Ａ～Ｆのは、主題のファイル・タイプがレンジ・データ・レコードを含む場合には、レンジ記述子レコード３１において用いられる。引数Ｇ、Ｈ、Ｗは、コラム幅記述子レコード３１において用いられ、特定された幅を用いるセルのレンジを特定する。引数Ｉは整数記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内にその整数の位置を特定する。引数Ｉはテキスト記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内にそのテキスト・ジャスティフィケーション・キャラクタ（右、左、中央）の位置を特定する。引数Ｓ、Ｔはテキスト記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内にそのテキスト・ストリングの位置と長さそれぞれを特定する。

引数Ｋはラベル又はヘッダ記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内でコラム又はロー・ラベルに関連するテキストの位置を特定する。引数Ｌ、Ｍは、ストリーム・ファイルに対して記述子レコード３１において用いられ、そのファイルでは、それぞれの主題のデータ・レコードは、レコード長の指標（indicator）を含む。引数Ｎは、１０進記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内で１０進数の位置を特定する。引数Ｏは、フォーマット記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコードの一部がデータ情報を反復する回数を特定する。

引数Ｐ、Ｑ、Ｒは公式（formula）記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内で公式の位置を特定する。引数Ｕは、データ・タイプ記述子レコード３１において用いられ、データ・タイプ指標の位置を特定する。引数Ｖは、数値データ・アイテムが有する１０進位置の数を特定する。

引数Ｘ、Ｙ、ｂは、スプレッドシートに関連する主題のファイル・タイプに対して用いられる。特に、引数Ｘ、Ｙは、データ記述子レコード３１において用いられ、そのデータがどのスプレッドシートのセルに所属するかを定義する。引数Ｚは、データ記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内のデータの位置を特定する。

引数ａは、時間記述子レコード３１において用いられ、主題のファイル・タイプ・データ・レコード内の時間指示の位置を特定する。

記述子レコード３１ｂ（図３ｃ）の次のバイト９８は、一般の数値変数フォーマットの「タイプ」キャラクタを示す。この「タイプ」キャラクタは、関連する引数が、キャラクタ、ストリング、又は数として解釈されるかどうかと、それが、バイト、ワード、倍ワード、短い又は長いＩＥＥＥフォーマットで記憶されているかどうかを決定する。この情報か

ら、関連する記述レコード２９の第１の部分の、コラム数フォーマット・タイプ５３、ロー数フォーマット・タイプ６３、レコード長フォーマット・タイプ７３、セル幅フォーマット・タイプ８３のバイトが、上述のように定義される。バイト９８に対する好適な１６進値と、当初の作業ファイル記述において用いられる対応する数値変数「タイプ」キャラクタ及び意味は次の通りである。

１６進値	数値変数のタイプ
０１	C　１バイト・フィールド
０２	U　未指定の（unsigned）１バイト・フィールド
０３	H　２バイトの整数
０４	W　未指定の２バイトの整数
０５	L　４バイトの整数
０６	V　未指定の４バイトの整数
０７	R　EXCEL３．０ＲＫの未指定の４バイトの整数
０８	J　LOTUSの２バイトの浮動小数点フォーマット
０９	F　IEEEの浮動小数点フォーマット
０Ａ	D　IEEEの長い浮動小数点フォーマット
０Ｂ	K　１０バイトの浮動小数点フォーマット
０Ｃ	Z　１バイトのキャラクタ・ロジカル
０Ｄ	I　ASCIIフォーマットの整数
０Ｅ	N　浮動小数点のASCIIフォーマット
０Ｆ	E　科学的概念のASCIIフォーマット
１０	B　２進ASCIIフォーマット
１１	O　８進ASCIIフォーマット
１２	Y　４桁の年、２桁の月、２桁の日での日付指示（YYYYMMDD）ASCIIフォーマット
１３	X　１６進ASCIIフォーマット
１４	S　キャラクタ・ストリング

図３ｃの記述子レコード３１ｂのバイト９９のバイト９９においては、数値変数情報の「幅」は、キャラクタの最小数として示される。数値変数情報の「精度」は、記述子レコード３１ｂのバイト９６で示される。好ましくは、バイト９６によって保持される値は、ストリング（Ｓタイプ）内のキャラクタの最大数又は１０進（Ｎタイプ）数の１０進数の右への桁数である。

記述子レコード３１ｂの最後の２バイト９２は、数値変数の定数（numeric variable constant）を示す。この定数は、デフォールト・オフセット以外のオフセットを特定する。例えば、レコードの最初の２バイトが、これらの２バイトを除くレコードの長さを含む場合には、定数はマイナス２を意味する。これは、本発明のインターフェース２１は、長さ

はこれらのバイトを含むことを想定しているからである。もう１つの例として、スプレッドシート内の最も低いセルが位置１，１にある場合には、定数は変数Ｘ及びＹに対するプラス１を意味する。これは、インターフェース２１がそのファイル・タイプ（スプレッドシート）に対する第１のセルが位置０，０にあることを想定しているからである。
図３ｄは、「スキップ・ストップ」レコード３１ｃに対する記述子レコード構造を図解している。これらのタイプのレコード内に記憶されているのは、いつ認識されていないデータを無視するかを定義することを助ける情報である。この情報は、求められている２つ又は３以上の（存在すれば整合型の）ストリングを定義する。最も近接した「存在すれば整合型」のストリングが開始の走査位置に対して見つかると、この走査位置は、初めの走査位置と見つかった「存在すれば整合型」のストリングとの間のすべてのデータを無視することになる。出力においては、定義された第１の「存在すれば整合型」のストリングは、特定されるように、主題のファイル・タイプ・レコードに書き込まれる。更に、「スキップ・ストップ」キャラクタは、「スキップ・メニー」成分（後述）を終了させる特別の成分である。
図３ｄを参照すると、記述子レコード３１ｃの第１のバイト４２は、１６進値０Ｘ１０を保持し、スキップ・ストップ・レコードを示す。レコード・バイト４４は、走査対象である「存在すれば整合型」のキャラクタ・グループの数を保持する。特定の「存在すれば整合型」のキャラクタ・グループは、記述子レコード３２ｃに従う。
対応する最初のワーキング・ファイルの記述において、２つ以上の“マッチ－イフ－プレゼント（match-if-present）”キャラクタのグループが、所望される“スキップ－ストップ（skip-stop）”キャラクタを指定するために角括弧の間に格納される。角括弧キャラクタが主題のキャラクタ・グループにおいて用いられたならば、それらは１６進キャラクタで表されるか又は逆スラッシュ・キャラクタがそれらに先行する。まとめると、スキップ－ストップ（skip-stop）レコード・タイプに遭遇したとき、インターフェース２１はファイル・ポインタをそれが発見する最初の“マッチ－イフ－プレゼント”キャラクタに配置がえする。
図３ｅは“スキップ－メニー（skip-many）”コンポーネントを指定するデスクリプタ・レコード３１ｄ示す。そのようなコンポーネントは、定められたファイル・フォーマットが、主題のファイル・タイプのデータのレコードのこの位置において、入力において無視されるべき、未知の数の更なるキャラクタを含み得ることを示すために用いられる。これは、デスクリプタ・レコード３１の“マッチ－イフ－プレゼント”キャラクタが主題のファイル・タイプのデータのレコードにマッチしないであろう場合に便利であり、その場合において、本発明のインターフェース２１は次の認識可能なキャラクタにスキップする。出力には、この情報から定められるフォーマット・ファイル内のこの位置（“スキップ－メニー”キャラクタによって示される）にキャラクタが書き込まれない。デスクリプタ・レコード３１ｄには常に“イグザクト－マッチ（exact-match）”または“スキップ－ストップ（skip-stop）”が後続し、よって、インターフェース２１のレコード・マッチング機構はキャラクタをスキップするのをいつ止めるかを知る、ということに留意されたい。即ち、“スキップ－メニー”キャラクタは、“スキップ－メニー”キャラクタに後続するキャラクタが発見されるまで残りのキャラクタがスキップされるべきであることを指定する、“マッチ－イフ－プレゼント”キャラクタの特別な場合である。最初のワーキング・ファイルの記述において、“スキップ－メニー”キャラクタは括弧に格納された３つの期間（period）によって表される。括弧のキャラクタが主題のファイル・タイプのデータにおいて用いられるならば、それは１６進キャラクタで指定されるか又は逆スラッシュ・キャラクタがそれに先行する。
図３ｅを参照すると、“スキップ－メニー”コンポーネントに対する１バイトの長さのデスクリプタ・レコード３２ｄは、スキップ－メニーのレコード・タイプを示すために１６進値０ＸＦＦを保持する。一般に、図３ｅのレコード構造に後続するバイトは、“イグザクト－マッチ”または“スキップ－ストップ”の情報を保持する。
図３ｆを参照すると、繰り返すレコードを定めるデスクリプタ・レコード３１ｅに対する

レコード構造が示されている。即ち、主題のファイル・タイプのデータ・レコード内で繰り返す１グループの“イグザクト－マッチ”、“スキップ－フィックスト（skip-fixed）”または“可変数値（variable-numeric）”のコンポーネントが、図３ｆに示されたタイプのデスクリプタ・レコード３１ｅによって定められる。このタイプのコンポーネントはフォーマット・デスクリプタ・レコード３１において用いられ、そこでは、フィールド情報は、特定のファイル・タイプに対して定められたフィールドの数に依存する未知の回数繰り返し得る。繰り返されたコンポーネントのグループは、発生カウントが先行するかまたは“スキップ－ストップ”コンポーネントが後続し、よって、インターフェース２１は繰り返されるデータを探すことをいつ止めるかを知る。

図３ｆに示すように、デスクリプタ・レコード３１ｅの構造は、最初のバイトにおいて、繰り返すレコード・タイプを示す。好適な実施例では、そのタイプは１６進値０Ｘ２０で示される。次のバイト４６は、繰り返されたコンポーネントを構成する後続するバイトの数を示す。対応する最初のワーキング・ファイルの記述（description）において、繰り返すキャラクタは特別のコンポーネントであり、それは“イグザクト－マッチ”、“スキップ－フィックス”または“可変数値”のコンポーネントを繰り返し、そしてカールした括弧内に入れられる。カールした括弧が繰り返すキャラクタにおいて用いられたならば、それらは１６進キャラクタで表されるか、または逆スラッシュがそれらに先行する。

前記に従ってテーブル２３が規定かつ確立された後、好適な実施例におけるテーブル２３を利用するインターフェース２１のオペレーションが、以下のソフトウエア・ルーチンによって実施される。より正確には、以下では、インターフェース２１の機能を提供する１グループのファイル・アクセス・サブルーチンを説明する。

概観では、本発明のインターフェース２１は、アプリケーション１５（図１）にデータの検索に対して２つのオプションを可能とし、且つ所望されるファイルへの書き込みのために２つの書き込み機構を提供する。データの検索の２つのオプションに関して、１つのオプションにおいてインターフェース２１はターゲット・ファイルをアクセスし、ファイルのレコードをテーブル２３に従ってパースし、１つの時に１つのセル又はカラム、検索されたデータを呼び出しアプリケーション１５に戻し渡す。第２のオプションにおいて、インターフェース２１はターゲット・ファイルをアクセスし、データの何れをもパースも抽出もせずに１つの時に１つのレコード、任意のおよびすべてのファイル・レコードを呼び出しアプリケーション１５に戻し渡す。こととき、呼び出しアプリケーション１５は、ターゲット・ファイルのレコードの構造を知ることを必要とし、そして、従って、検索されたレコードをパースしなければならない。この第２のオプションはアプリケーション１５のために設計された低レベルのインターフェースであり、それは、簡単なデータ転送よりも、ターゲット・ファイルについてのより多くの情報を知る必要がある。

好適な実施例では、前者の読み取りオプションは“ＤＸｇｅｔｃｅｌｌ”１０７（図４ｂ）と呼ばれるルーチンによって達成され、後者の読み取りオプションは“ＤＸｇｅｔｒｅｃ”１０９（図４ｃ）と呼ばれるルーチンによって達成され、両方とも以下に説明される。

インターフェース２１の書き込み機構に関して、最初の書き込み機構は“ＤＸｐｃｅｌｌ”と呼ばれるルーチンにおいて実施される。このルーチンはデータを一度に１つのセル受け入れ、そしてデータを適切なファイル・フォーマットでターゲット・ファイルに書き込む。第２の書き込み機構は、呼び出しアプリケーション１５がレコードを直接に出力ファイル・ストリームに挿入することを可能にする。これら出力レコードは、ターゲットの出力ファイル・タイプに親しいフォーマットに既にフォーマットされていなければならない。一般に、この機構は、出力されるファイルのタイプに対して、現在生成され且つテーブル２３によって理解されているレコード以外の、更なる情報レコードが要求されたときのみ用いられる。この第２の書き込み機構は“ＤＸｐｕｔｒｅｃ”と呼ばれるルーチンによって達成される。これら２つの書き込みルーチンは図５ａ及びｂを参照して後に詳細に説明される。

図４ａを参照すると、インターフェース２１のデータ検索（読み取り）（retrieval(read

))オペレーション１０３が示されている。データがターゲット・ファイルからアクセスされ得る前に、呼び出しアプリケーション１５又はユーザがそのファイルをオープンせねばならない。これは、呼び出しアプリケーション１５がグローバル・メモリ・リード・スタック（global memory read stack）１０１を初期設定するか又は自由にし、ＣＰＵ１３（図１）に「オープン」コマンドを発行することによって達成される。「オープン」コマンドにおいて指定されるのは、所望されたターゲット・ファイル１７のファイル名及びそのファイルのファイル・タイプである。「オープン」コマンドに応答し、インターフェース２１は（リード・オペレーション１０３を経由で）そのファイル・タイプが以前にオープンされているかどうかを判定する。そのファイル・タイプが以前にオープンされているならば、以前に割り当てられたローカル・メモリ・スペース及び以前に生成された内部構造が用いられる。

そのファイル・タイプが以前にオープンされていなければ、インターフェース２１はローカル・メモリ・スペース１０５を割り当てる。次に、図４ａの１０２で、リード・オペレーション１０３はテーブル２３をオープンし、所望されるファイル１７の宛て先のファイル・タイプにマップするテーブル・エントリを探す。所望されるファイル・タイプに対応するファイル・タイプを有するテーブル・エントリを位置付けすると、リード・オペレーション１０３（即ち、図４ａの１０４でのインターフェース２１）は、そのエントリを読み、そのエントリのもとで指定されたデスクリプタ・レコード３１を解釈する。対応するファイル・タイプのデスクリプタ・レコード（descriptor record）３１の解釈において、リード・オペレーション１０３は割り当てられたメモリ・スペース１０５において１セットのオブジェクト・レコード（object record）１００を定める。これらオブジェクト・レコード１００の内容は、所望されるファイル・タイプのファイル・フォーマットを指定する。

オブジェクト・レコード１００において指定されたファイル・フォーマットに従い、インターフェース２１は、与えられたファイル名によって所望されるファイル１７をオープンすることによってオペレーション１０３のステップ１０６で続ける。ひとたびファイル１７がオープンされていると、そのファイルに含まれるデータは、本発明の２つのオプションのうちの１つに従ってアクセスされ且つ検索され得る。どのようにオプションが用いられても、検索されたデータは、呼び出しアプリケーション１５によって初期設定されたグローバル・メモリのリード・スタック１０１に配置される。好適な実施例では、ルーチン“ＤＸｇｅｔｃｅｌｌ”１０７は、オープンされたスプレッドシート／データベース・ファイル１７から一度に１つのセル／フィールドを検索するために呼ばれる。特に、“ＤＸｇｅｔｃｅｌｌ”１０７は、オープンされたファイル１７をアクセスし、テーブル２３のエントリから形成されたオブジェクト・レコード１００を用いてそのファイルのレコードを変換し、呼び出しアプリケーション１５によって読み取るために一度に１つのセルのデータをリード・スタック１０１に渡す。

互換的に、ルーチン“ＤＸｇｅｔｒｅｃ”１０９は、オープン・ファイル１７から所望のデータを検索するために呼ばれる。そのルーチン１０９はファイル１７を１つの時にそのネイティブのフォーマット、１つのレコードまたはセルでアクセスする。この場合、レコードまたはネイティブのセルは、それがパース（parse）される前にレコード・スタック１０１に配置され、呼び出しアプリケーション１５は、データを検索するためにそのレコードをパースしなければならない。以前に述べたように、このオプションは、一般に、呼び出しアプリケーション１５が、“ＤＸｇｅｔｃｅｌｌ”１０７アクセス方法を通じて戻されたセルの内容情報よりも多くの情報をターゲット・ファイル１７から要求する場合のみ用いられる。

更に、データの検索からの出力に、リード・オペレーション１０３は、（ｉ）アクセスされているファイル１７及びそれに対応するオブジェクト・レコード１００を識別するためのパラメータと、（ii）検索されたデータが書かれているリード・スタック１０１へのポインタと、を提供する。

オープン・ファイル１７がもはやアクセスされないとき、オペレーション１０３は１０８

（図４ａ）でファイル１７をクローズし得る。特に、リード・オペレーション１０３は、リード・スタック１０１に保持された検索されたデータを、呼び出しアプリケーション１５のもとにディスクを書き込み、続いてリード・スタック１０１をクリアする。これは、即ち、ＣＰＵ１３がメモリを使い果たした場合またはファイル１７がソートすることを必要とする場合に起こる。更に、これは、呼び出しアプリケーション１５のファイル・フォーマットに従って達成され、従って、レコード及びヘッダが検索されたファイル・データに対してデイスク上でフォーマットされ、データがそのフォーマットに書き込まれ、数または他の特別のキャラクタが呼び出しアプリケーション・ディスク・バージョンの検索データに対して適切に変換される。前記のことは、呼び出しアプリケーションのファイル・タイプのデスクリプタ・レコード３１の数値可変情報において指定された可変のタイプ及び幅に従ってなされる。最後に、スタック・ポインタが、リード・スタック１０１がクリアされていることを反映するのに適切なように、移動される。

図４ｂに示すのは、より特定的なターム（term）におけるルーチン“ＤＸｇｅｔｃｅｌｌ”１０７である。ルーチン１０７はオープン・ファイル１７から一度に１つのファイル・セグメントを読み取る（ステップ２００）。各読み取られたファイル・セグメントに対して、“ＤＸｇｅｔｃｅｌｌ”１０７はそのファイル・セグメントを構成する多種の部分を識別する。これは、“ＤＸｇｅｔｃｅｌｌ”１０７がファイル・セグメントの部分を、オブジェクト・レコード１００に従うデスクリプタ・レコードの指定した幅、サイズおよびデータ部分のタイプにマッピングまたはマッチングすることによって、ステップ２０２で達成される。

多種の部分が識別されると、ルーチン１０７は、データを記憶するために２０４でメモリを割り当て、読まれたデータをリード・スタック１０１に２０８で配置する。次に、ルーチン１０７は、オープン・ファイル１７のセルがファイル１７にランダムのオーダーで記憶されたかどうかを判定する。そうでないならば、ルーチン１０７は、リード・スタック１０１に配置されたデータを呼び出しアプリケーションに戻す。ルーチン１０７が、オープン・ファイル１７から読まれたデータがランダムのオーダーである、即ち、ファイル１７のセルが順が狂って記憶／読み取りされた、と判定すると、ルーチン１０７は、ファイル１７全体を読み取り、そして、検索されたデータをリード・スタック１０１を経て呼び出しアプリケーションに実際に戻し渡す前に、読まれたデータをソートする。これは、図４ｂのステップ２１６へのループ２１６によって示され、ここでは、ソートは、読まれたデータを保持する一時的ファイルを用いて好適に行われる。

ステップ２１８で、リード・スタック１０１が満杯であるか又は満杯になるならば、ルーチン１０７はバックアップ・ファイルを用いる。リード・スタック１０１の最初の内容は、バックアップ・ファイルに移動され（ステップ２１０）、未来に検索されるリード・データが、使用可能なリード・スタック１０１に配置される。

図４ｃを参照すると、ルーチン“ＤＸｇｅｔｒｅｃ”１０９が示されている。ユーザまたは呼び出しアプリケーション１５が、オープンされたファイル１７から読まれるべき所望された数のユニット（即ち、バイト、ファイル・セグメント、またはレコード）を指定すると、リード・オペレーション１０３はルーチン１０７の代わりにルーチン１０９を呼ぶ。

最初に、ルーチン１０９のステップ２１２は、読まれるように所望される指定された数のユニットを受信し解釈する。特に、ステップ２１２で、ルーチン１０９は、バイトまたはファイル・レコード・セグメントと等価のユニットで、つくるようにオープン・ファイル１７の読み取りアクセスのワーキング・ナンバー（working number）を確立する。ステップ２１２はそのワーキング・ナンバーをステップ２１４に渡す。

ステップ２１４は、オープン・ファイル１７からレコード・セグメント／バイトのワーキング・ナンバーを読み取るためのループを提供する。ループ２１４を通る各パスに対して、１つのそのようなレコード・セグメント／バイトは、ファイル１７から、直接、又は、図４ｂにおいて上記で説明したルーチン“ＤＸｇｅｔｃｅｌｌ”１０７への呼び出しによって、の何れかで読まれる。特定的には、オブジェクト・レコード１００によって表され

たデスクリプタ・レコード情報が十分であり、ルーチン１０９がステップ２１２で「レコード」が何であるかを決定できるならば、ループ２１４は、レコードまたはレコード・セグメントのその定義に対する適切な数のバイトをオープン・ファイル１７から直接に読み取る。そうでないならば、ループ２１４はルーチン１０７を呼び、ループ２１４をワーキング・ナンバーの回数回る。このように、ワーキング・ナンバーまたは指定された数のバイトが一度に１つ（即ち、ワーキング・ナンバーの回数が数えられるまで）ファイル１７から読まれる。

図５ａを参照すると、インターフェース２１のライト（write）・アクセス・オペレーション１１０のフロー図が示されている。リード・オペレーション１０３においてのように、ファイルに書き込みがなされるように、オペレーション１０１は最初にそのファイルをオープンする。これは、ユーザまたは呼び出しアプリケーション１５がグローバル・メモリ・ライト・スタック１１４を初期設定しかつ自由にし、ＣＰＵ１３（図１）に「オープン」コマンドを発行することによって達成される。「オープン」コマンドにおいて指定されるのは、所望されたターゲット・ファイル１７のファイル名及びそのファイルのファイル・タイプである。「オープン」コマンドに応答し、インターフェース２１は（ライト・オペレーション１１０を経由で）そのファイル・タイプが以前にオープンされているかどうかを判定する。そのファイル・タイプが以前にオープンされているならば、以前に割り当てられたローカル・メモリ・スペース及びそこに存在する内部構造が用いられる。

そのファイル・タイプが以前に用いられていなければ、インターフェース２１はローカル・メモリ・スペース１０５を割り当てる。次に、ステップ２３０で、ライト・オペレーション１１０はテーブル２３をオープンし、所望されるファイル１７の宛て先のファイル・タイプにマップするテーブル・エントリを探す。所望されるファイル・タイプに対応するファイル・タイプを有するテーブル・エントリを位置付けすると、ライト・オペレーション１１０（即ち、図５ａの２３５でのインターフェース２１）は、そのエントリを読み、そのテーブル・エントリのもとで指定されたデスクリプタ・レコード３１を解釈する。対応するファイル・タイプのデスクリプタ・レコード３１の解釈において、ライト・オペレーション１１０は割り当てられたメモリ・スペース１０５において１セットのオブジェクト・レコード２３６を定める。これらオブジェクト・レコード２３６の内容は、所望されるファイル１７のファイル・タイプに対するファイル・フォーマットを指定する。

オブジェクト・レコード２３６において指定されたファイル・フォーマットに従うと、インターフェース２１は、オペレーション１１０のステップ２３２で、与えられたファイル名によって、所望されるファイル１７をオープンする。ひとたびファイル１７がオープンされていると、所望されるデータがそのファイルに書かれ得る。データは、ユーザまたは呼び出しアプリケーション１５から受信され、グローバル・メモリ・ライト・スタック１１４に記憶される。ライト・オペレーション１１０のステップ２３８で、データ部分がライト・スタック１１４の最上部から「ポップされる（popped）」または受信される。そのデータ部分が、次に、本発明によって提供される２つの書き込み機構のうちの１つに従ってオープンされたファイル１７に書き込まれる。ライト・スタック１１４の最上部からのデータ部分の受信およびオープン・ファイル１７へのそのデータ部分の書き込みは、図５ａのループ２４０によって示すようにライト・スタック１１４が空にされるまで一度に１つのデータ部分ずつ、続ける。

オープン・ファイル１７がもはやアクセスされないとき、ライト・オペレーション１１０は図５ａのステップ２３４でファイル１７をクローズする。特に、ライト・オペレーション１１０は、オープン・ファイル１７のもとでディスクに、そこにまだ書き込まれていないライト・スタック１１４に保持された受信したデータを、書き込み、そして続いてライト・スタック１１４をクリアする。これはオブジェクト・レコード２３６に示されたファイル・フォーマットに従って達成され、従って、レコード及びヘッダが所望されたデータに対してディスク上でフォーマットされ、データがそのフォーマットに書き込まれ、数または他の特別のキャラクタがディスク・バージョンの受信したデータに対して適切に変換される。これは、オープン・ファイル１７の対応するファイル・タイプのデスクリプタ・

レコード３１の数値可変情報における可変のタイプ及び幅の仕様に従って行われる。最後に、スタック・ポインタが、ライト・スタック１１４がクリアされていることを反映するのに適切なように、移動される。

好適な実施例において、ライト・オペレーション１１０は、ルーチン“ＤＸｐｃｅｌｌ”１１１を呼び出すことによってユーザまたは呼び出しアプリケーション１５によって受信したセルまたはレコードをオープン・ファイル１７に書き込む。一般に、ルーチン“ＤＸｐｃｅｌｌ”１１１は、一度に１つのセルを受け入れ、テーブル２３のファイル・タイプ・エントリおよび、それ故、ファイル１７のファイル・フォーマットを反映するオブジェクト・レコード２３６に従ってそれを変換し、そしてそこからファイル１７のファイル・タイプに対して適切なレコードを生成する。次に、ルーチン１１１は生成されたレコードをディスクのオープン・ファイル１７に書き込む。

図５ｂはルーチン“ＤＸｐｃｅｌｌ”１１１をより特定的に詳細に示す。所望されるファイル１７に書き込まれるべきデータは、ユーザまたは呼び出しアプリケーション１５から受信され、ライト・スタック１１４に記憶される。ライト・スタック１１４が満杯になると、ライト・スタックの現在の内容を一時的スクラッチ（scratch）・ファイルに移動する手順が２５０で呼ばれる。

次に、ステップ２５２（図５ｂ）で、“ＤＸｐｃｅｌｌ”１１１は、ライト・スタック１１４から読み取られた各データ部分に対して、一度に１つの部分、以下のことを言う。最初に、２５４で、“ＤＸｐｃｅｌｌ”１１１は、スタック１４１から取られた（即ち、ライト・スタック１１４の最上部からポップされた）データ部分のデータ・タイプを決定する。更に、そのデータ部分のタイプ・サイズ及び幅が２５６で設定される。必要であると、オープン・ファイル１７に書き込むために、数値量が変換される。

必要であると、次のステップ２４８は、テーブル２３からのファイル・フォーマットを指定するオブジェクト・レコード２３６に従ってオープン・ファイル１７に対するワーキング・ヘッダ及びレコードをフォーマットする。ヘッダはまたフォーマットされたレコードに書き込まれる。また、ライト・スタック１１４からのデータ部分は、フォーマットされたワーキング・レコード２４６に書き込まれる。最終的に、ワーキング・レコードはステップ２４４（図５ｂ）でオープン・ファイル１７に書き込まれる。続いて、ライト・スタック・ポインタが、スタック１１４から読まれるべきデータの次の部分に移動される。

前記のステップが、ライト・スタック１１４および主題のデータを記憶するのに用いられる一時的ファイルから読まれた各データ部分に対して繰り返される。データ部分が一時的ファイルから読まれた場合、オープン・ファイル１７に書き込まれるべきレコードをフォーマットするステップ２４８はまた、主題のデータが現在のライト・スタック１１４からのものか又は一時的ファイルからのものかを決定する。すべての一時的ファイルおよびスタック１１４が空、そして、従って、すべての所望されるデータがオープン・ファイル１７に書き込まれたとき、ルーチン１１１は、図５ａのループ２４０の終わりでライト・オペレーション１１０に戻る。

ルーチン１１１と互換的に、ライト・オペレーション１１０は上記で述べたオプションの書き込み機構を用いる。これは、オペレーション１１０が、呼び出しアプリケーション１５またはユーザから１つの時に所望のデータの１つのレコードを受け入れ、そして受信したレコードまたは指定された数のバイトを直接にオープン・ファイル１７（即ち、ディスク）に書き込む、ことを含む。

リード・ライト・アクセスに加えて、テーブル２３はファイル・タイプ認識のために用いられる。特に、呼び出しアプリケーション１５はコマンド“ＤＸｆｉｌｅｔｙｐｅ”を発行し得る。このルーチンは、指定されたターゲット・ファイルのフォーマットを決定し、そしてファイルのフォーマットの表示を呼び出しアプリケーション１５に戻す。より特定的には、コマンド“ＤＸｆｉｌｅｔｙｐｅ”は指定されたターゲット・ファイル１７をオープンするためのＣＰＵ１３への呼び出しであり、テーブル２３を用いてオープン・ファイルを見てそのファイル・タイプを理解する。ＣＰＵ１３がファイル１７をオープンするために、このコマンドは、ＣＰＵ１３が知るすべてのファイル・タイプに対してファイル

をオープンするようにＣＰＵ１３に要求する。ＣＰＵのすべてのそのファイル・タイプのリストから、“ＤＸｆｉｌｅｔｙｐｅ”ルーチンはテーブル２３から最もマッチするファイル・タイプを理解する。

【図１】

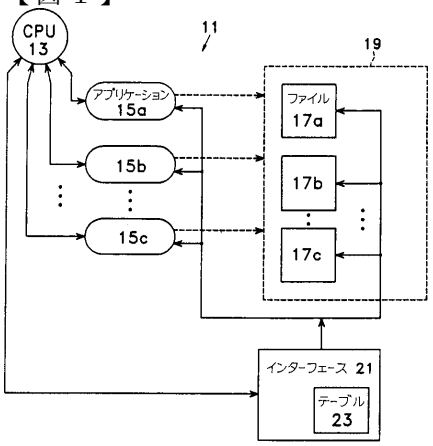

Fig. 1

【図２】

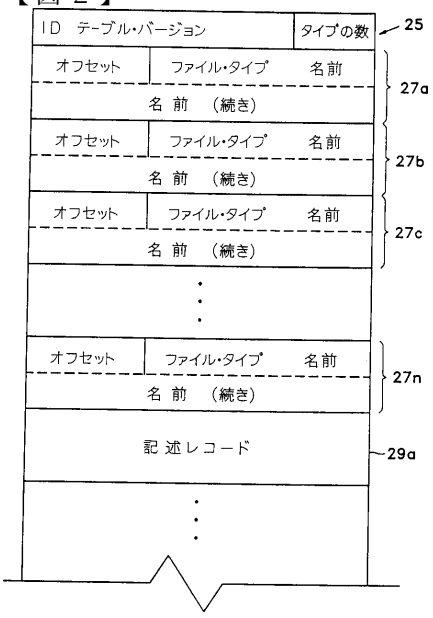

Fig. 2

【図３ａ】

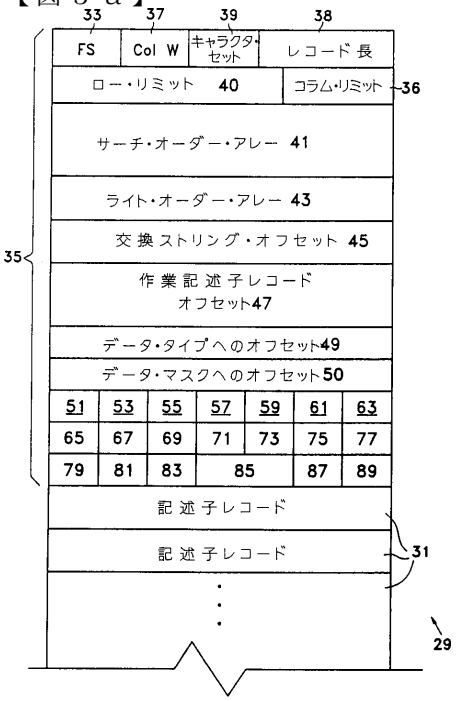

Fig. 3a

【図３ｂ】

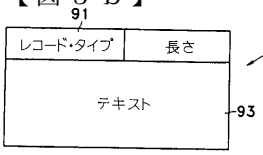

Fig. 3b

【図３ｃ】

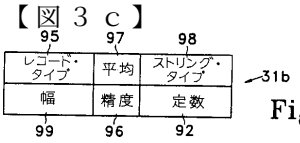

Fig. 3c

【図３ｄ】

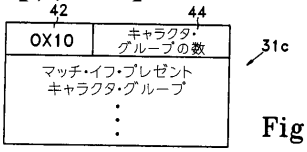

Fig. 3d

【図３ｅ】

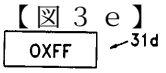

Fig. 3e

【図３ｆ】

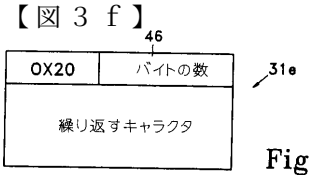

Fig. 3f

【図４ａ】

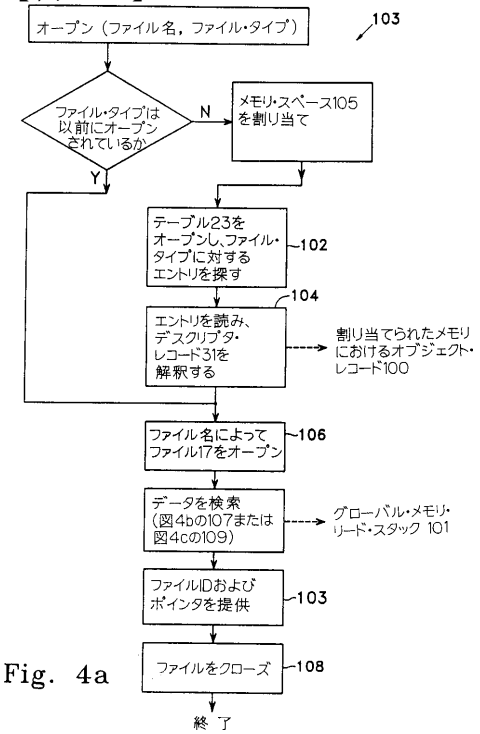

Fig. 4a

【図４ｂ】

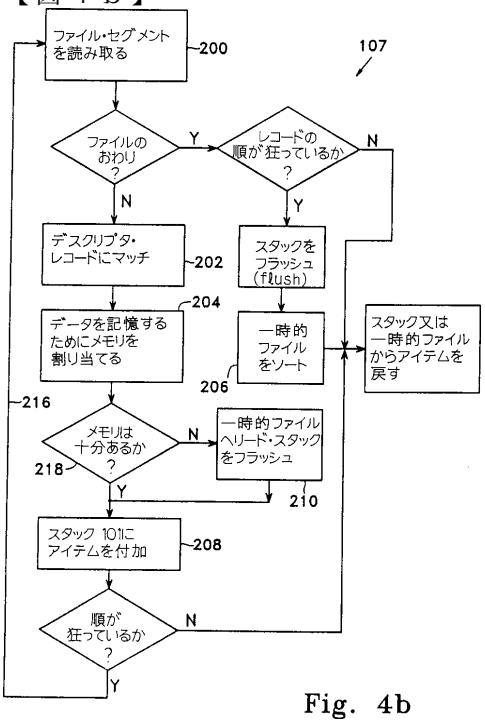

Fig. 4b

【図４ｃ】

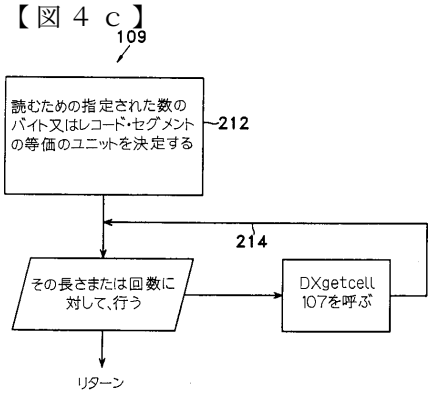

Fig. 4c

【図５ａ】

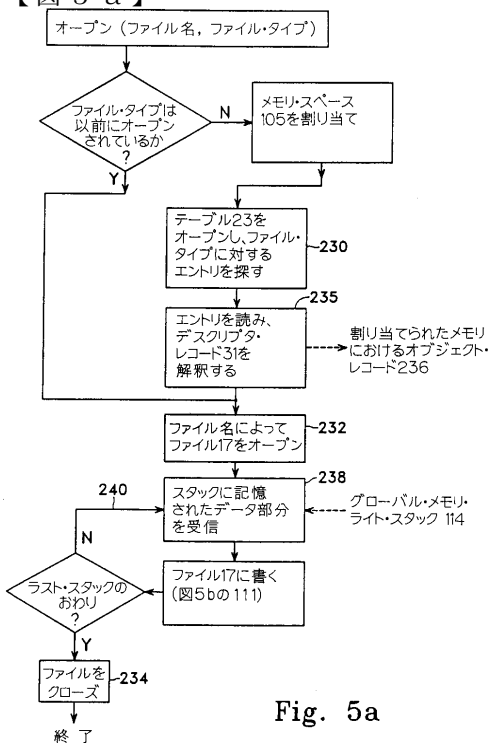

Fig. 5a

【図５ｂ】

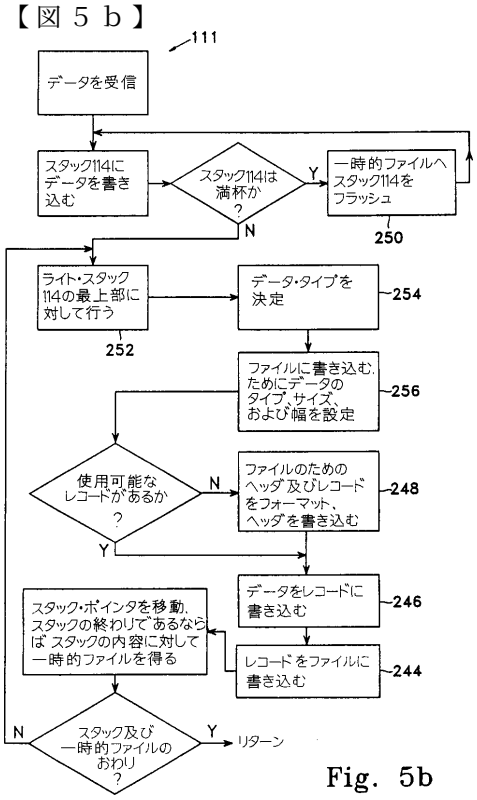

Fig. 5b

フロントページの続き

(74)代理人 100075236
弁理士 栗田 忠彦
(74)代理人 100075270
弁理士 小林 泰
(74)代理人 100084283
弁理士 秋元 芳雄
(72)発明者 メセ，エドワード・ディー
アメリカ合衆国マサチューセッツ州０２１１８－１４５０，ボストン，ハンソン・ストリート ４２，アパートメント ３

審査官 相崎 裕恒

(56)参考文献 特開平０１－１４０２３８（ＪＰ，Ａ）
特開平０３－１２３９３７（ＪＰ，Ａ）
特開昭６３－２８２８４６（ＪＰ，Ａ）
特開昭６０－０９１４５０（ＪＰ，Ａ）
特開昭５８－１６９２６５（ＪＰ，Ａ）
特開昭６１－０４２０３６（ＪＰ，Ａ）
特開平０４－００７７５９（ＪＰ，Ａ）
特開平０３－１７６７４０（ＪＰ，Ａ）
バインス情報センター，初めて使うdBASE III，日本，株式会社技術評論社，第1版，p. 90--102
阿部 友計，MS-Excel 操作ガイドブック，日本，ナツメ社，１９９０年 ４月３０日，p. 144--157

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/00,17/21 - 28