世界知的所有権機関
国際事務局

PCT

特許協力条約に基づいて公開された国際出願

(51) 国際特許分類 5 **G06F 12/00**	**A1**	(11) 国際公開番号	**WO 94/22086**
		(43) 国際公開日	1994年9月29日(29.09.94)

(21) 国際出願番号 PCT/JP94/00454
(22) 国際出願日 1994年3月22日(22. 03. 94)

(30) 優先権データ
特願平5/62181 1993年3月22日(22. 03. 93) JP

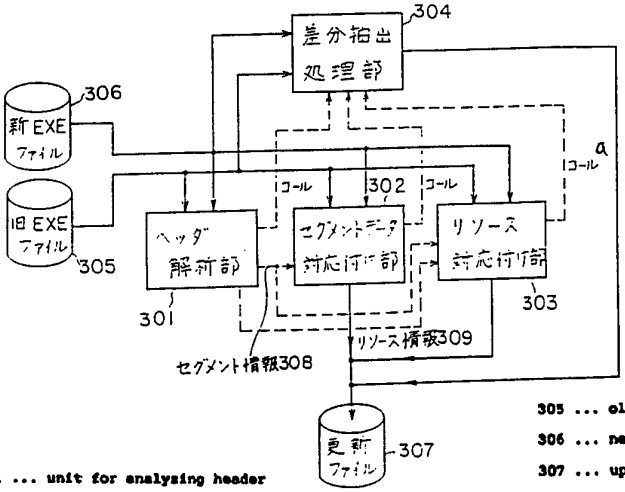
(71) 出願人(米国を除くすべての指定国について)
エヌ・ティ・ティ・データ通信株式会社
(NTT DATA COMMUNICATIONS SYSTEMS CORPORATION)
〔JP/JP〕
〒135 東京都江東区豊洲三丁目3番3号 Tokyo, (JP)
(72) 発明者;および
(75) 発明者/出願人(米国についてのみ)
谷村守正(TANIMURA, Morimasa)〔JP/JP〕
〒135 東京都江東区豊洲三丁目3番3号
エヌ・ティ・ティ・データ通信株式会社内 Tokyo, (JP)
(74) 代理人
弁理士 大菅義之(OSUGA, Yoshiyuki)
〒102 東京都千代田区麹町6丁目1番18号 麹町共栄ビル
Tokyo, (JP)

(81) 指定国
GB, US.

添付公開書類 国際調査報告書

(54) Title : APPARATUS AND METHOD FOR EXTRACTING AND UPDATING FINITE DIFFERENCE IN EXECUTABLE FILE

(54) 発明の名称 実行ファイル差分抽出/更新装置及び実行ファイル差分抽出方法



(57) Abstract

For the new and old EXE files, a header analyzing unit extracts the areas of their headers and their tables, and then, calls a unit for extracting the finite differences in those areas. A mapping unit maps the segmental data between the old and new EXE files, and then, calls the extracting unit for the mapped areas. Another mapping unit maps the resources between the new and old EXE files, and then, calls the extracting unit for the mapped resources. The extracting unit extracts the finite differences in units of one byte in the new and old files from the head to the end of each of the areas as specified above, and saves the finite differences in an update file.

(57) 要約

ヘッダ解析部は、新旧ＥＸＥファイルのそれぞれにつき、そのヘッダの領域及び各テーブル部の領域を抽出し、それらの領域につき差分抽出処理部をコールする。セグメントデータ対応付け部は、新旧ＥＸＥファイルの間のセグメントデータの対応付けを行い、対応付けされた領域について差分抽出処理部をコールする。リソース対応付け部は、新旧ＥＸＥファイル間のリソースの対応付けを行い、対応付けされた領域について差分抽出処理部をコールする。差分抽出処理部は、上述の指定された各領域の先頭から末尾まで、新旧ＥＸＥファイルに対するバイト単位の差分を抽出し、その差分を更新ファイルにセーブする。

情報としての用途のみ
PCTに基づいて公開される国際出願のパンフレット第1頁にPCT加盟国を同定するために使用されるコード

AM アルメニア
AT オーストリア
AU オーストラリア
BB バルバドス
BE ベルギー
BF ブルキナ・ファソ
BG ブルガリア
BJ ベナン
BR ブラジル
BY ベラルーシ
CA カナダ
CF 中央アフリカ共和国
CG コンゴー
CH スイス
CI コート・ジボアール
CM カメルーン
CN 中国
CS チェッコスロヴァキア
CZ チェッコ共和国
DE ドイツ
DK デンマーク
EE エストニア
ES スペイン
FI フィンランド
FR フランス
GA ガボン
GB イギリス
GE グルジア
GN ギニア
GR ギリシャ
HU ハンガリー
IE アイルランド
IT イタリー
JP 日本
KE ケニア
KG キルギスタン
KP 朝鮮民主主義人民共和国
KR 大韓民国
KZ カザフスタン
LI リヒテンシュタイン
LK スリランカ
LT リトアニア
LU ルクセンブルグ
LV ラトヴィア
MC モナコ
MD モルドバ
MG マダガスカル
ML マリ
MN モンゴル
MR モーリタニア
MW マラウイ
NE ニジェール
NL オランダ
NO ノルウェー
NZ ニュー・ジーランド
PL ポーランド
PT ポルトガル
RO ルーマニア
RU ロシア連邦
SD スーダン
SE スウェーデン
SI スロヴェニア
SK スロヴァキア共和国
SN セネガル
TD チャード
TG トーゴ
TJ タジキスタン
TT トリニダード・トバゴ
UA ウクライナ
US 米国
UZ ウズベキスタン共和国
VN ヴィェトナム

明　細　書

発明の名称

実行ファイル差分抽出／更新装置及び
実行ファイル差分抽出方法

技術分野

本発明は、バグの修正又は部分的な機能拡張を目的として実行ファイルが更新される場合に、新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新ファイルとから新実行ファイルが生成される実行ファイル差分抽出／更新装置及び方法に関する。

背景技術

ユーザに実行ファイルが配付されその実行ファイル（ＥＸＥファイル）により所定サービスが提供されている場合、その実行ファイルに存在するバグを修正したり、その実行ファイルによって実現される機能の部分的な拡張を目的として、図１Ａに示されるように、ユーザに配付されている実行ファイルにおいて、新ＥＸＥファイルから旧ＥＸＥファイルへの更新がなされる場合がある。

一方、最近では、コンピュータシステムの機能の発達により、実行ファイルのプログラムサイズが数

百キロバイトから数メガバイト或いは数十メガバイトに達しつつある。従って、上述のような実行ファイルの更新時に、新ＥＸＥファイルがそのまま配付されるのは、配付媒体に多くのデータ容量が要求されることになり、不経済である。

例えば、配付媒体が図２に示されるようなネットワークである場合、センタＴＣのホストコンピュータＨＯＳＴは、回線を介して、事務所ＴＣ内のホストコンピュータＨＯＳＴに対して新ＥＸＥファイルを配信することになるが、新ＥＸＥファイルが上述のような大きなサイズを有する場合、回線使用料が高くついてしまい、不経済である。

このような問題に対し、従来、新ＥＸＥファイルそのものが配付されるのではなく、図１Ｂに示されるように、旧ＥＸＥファイルと新ＥＸＥファイルとの間で比較処理がなされ、新ＥＸＥファイルの旧ＥＸＥファイルに対する差分が抽出されて更新ファイルとされ、その更新ファイルがユーザに配付されるという形態が採られるようになってきている。この場合、ユーザ側では、図１Ｂに示されるように、所定の更新プログラムによる更新処理によって、旧ＥＸＥファイルと更新ファイルとから新ＥＸＥファイルが生成される。新ＥＸＥファイルと旧ＥＸＥファイルとでは、プログラムコードが同じ部分が存在する場合が多いために、それらの差分である更新ファ

イルのサイズは、新ＥＸＥファイルのサイズに比較して小さい場合が多い。従って、更新ファイルがユーザに配付されることにより、配付媒体に要求されるデータ容量を削減することができる。

図２の例においては、センタＴＣのホストコンピュータＨＯＳＴは、回線を介して、事務所ＴＣ内のホストコンピュータＨＯＳＴに対して更新ファイルを配信する。事務所ＴＣ内のホストコンピュータＨＯＳＴでは、センタ間通信部が、上記更新ファイルを受信した後、更新プログラムを起動することにより、旧ＥＸＥファイルと更新ファイルとから新ＥＸＥファイルを生成する。この新ＥＸＥファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。そして、事務所ＴＣのホストコンピュータＨＯＳＴ内の配信処理管理部が、新ＥＸＥファイルを、事務所内通信部からＬＡＮ（ローカルエリアネットワーク）を介して、クライアントサーバＣＳＶに配信する。クライアントサーバＣＳＶでは、事務所内通信部が上記新ＥＸＥファイルを受信する。この受信された新ＥＸＥファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。

発明の開示

しかし、上述のような従来の差分抽出方式では、旧ＥＸＥファイルの先頭及び新ＥＸＥファイルの先

頭から各ファイルの末尾までのファイル全体にわたって、バイト単位でデータが一致するか否かが判定され、一致しないデータが差分のデータとして抽出されている。このために、例えば新ＥＸＥファイルの途中で１バイトでもデータの削除又は挿入が行われていると、そのバイト以降では全て旧ＥＸＥファイルのデータと新ＥＸＥファイルのデータとが一致しなくなり、その結果、更新ファイルのサイズが大きくなってしまうという問題点を有している。従って、図２の例のようにネットワークを介して更新ファイルが配信される場合などで、新ＥＸＥファイルがそのまま配信される場合に比較してそれほど回線の使用効率が向上しないという結果になってしまう。

本発明は、更新ファイルのサイズを削減することを目的とする。

本発明は、まず、実行ファイル差分抽出装置として、以下の構成を有する。

領域分割部は、旧実行ファイル及び新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する。この論理単位は、例えば、旧実行ファイル又は新実行ファイルのデータ属性の単位であるセグメント、リソース、ヘッダ、テーブル、セクションディレクトリ、セクションなどである。

次に、領域対応付け部は、領域分割部によって分割された旧実行ファイルの複数の領域と新実行ファ

イルの複数の領域との間で、互いに対応する領域同士を対応付けする。

また、差分抽出処理部は、領域対応付け部によって対応付けされた旧実行ファイルの領域と新実行ファイルの領域との間で、旧実行ファイルの領域のデータに対する新実行ファイルの領域のデータの差分情報を抽出する。この差分情報は、例えば、新実行ファイルにおいて旧実行ファイルに対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される。

そして、更新ファイル生成部は、差分抽出処理部において抽出された差分情報を更新ファイルに格納すると共に、領域対応付け部において対応付けが行われなかった旧実行ファイルの領域又は新実行ファイルの領域に対応する差分情報を抽出しそれを更新ファイルに格納する。

また、本発明は、実行ファイル更新装置として、上述の更新ファイル生成部により生成された更新ファイルから前記差分情報を順次取り出し、その差分情報と旧実行ファイルとから、新実行ファイルを生成するファイル更新部を有するように構成される。

本発明においては、差分抽出処理が実行される場合、例えば ✳ システムにおける実行ファイルについては、DOS EXE 形式の領域、 ✳ EXEヘッダ領域、

According PCT Article 21(6) this part should be omitted. (✳)
page 5
line 24, 25

各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で、差分抽出処理が実行される。また、例えば＊　　システムにおける実行ファイルについては、後述するように、＊　　EXEヘッダ領域、＊　　EXE ヘッダ領域、セクションディレクトリ領域、及びセクション領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。

このため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズが大幅に削減される。

図面の簡単な説明

図１Ａ、図１Ｂは、ファイル更新の説明図、

図２は、ファイル更新環境の説明図、

図３は、本発明の好適実施例の構成図、

図４は、本発明の好適実施例におけるファイル形式を示した図、

図５は、ヘッダ解析部の動作フローチャート（その１）、

図６は、ヘッダ解析部の動作フローチャート（その２）、

図７は、DOS EXE ヘッダと ＊ EXEヘッダのデー

According PCT Article 21(6) this part should be omitted. (*)
page 6
line 4-6, 25

タ構造を示した図、

図８は、✳ EXEヘッダ情報を示した図、

図９は、セグメントテーブル部のデータ構造を示した図、

図１０は、常駐／非常駐ネームテーブル部のデータ構造を示した図、

図１１は、インポートネームテーブル部のデータ構造を示した図、

図１２は、モジュール参照テーブル部とインポートネームテーブル部の関係を示した図、

図１３は、リソーステーブル部のデータ構造を示した図、

図１４は、セグメントデータ対応付け部の動作フローチャート、

図１５Ａ、図１５Ｂは、旧ＥＸＥファイル及び新ＥＸＥファイルのセグメントの配列を示した図、

図１６Ａ、図１６Ｂは、セグメントの対応関係の説明図、

図１７は、リロケーション情報の説明図、

図１８は、リソース対応付け部の動作フローチャート、

図１９Ａ、図１９Ｂは、旧ＥＸＥファイルと新ＥＸＥファイルのリソースの配列を示した図、

図２０は、リソースの対応関係の説明図、

図２１は、差分抽出処理の動作フローチャート、

According PCT Article 21(6) this part should be omitted. (✳)

page 7

line 2

図２２は、比較処理の動作フローチャート、

図２３は、比較処理の説明図、

図２４Ａ、図２４Ｂは、新ＥＸＥファイルに挿入が行われた場合の差分抽出処理の説明図、

図２５Ａ、図２５Ｂは、新ＥＸＥファイルで削除が行われた場合の差分抽出処理の説明図、

図２６Ａ、図２６Ｂは、新ＥＸＥファイルで置換が行われた場合の差分抽出処理の説明図、

図２７Ａ、図２７Ｂは、新ＥＸＥファイルで挿入、削除、及び置換が同時に行われた場合の差分抽出処理の説明図、

図２８Ａ、図２８Ｂは、新ＥＸＥファイルで特殊な挿入及び削除が同時に行われた場合の差分抽出処理の説明図、

図２９Ａ、図２９Ｂは、更新ファイルのデータ構造を示した図、

図３０は、本発明の他の好適実施例におけるファイル形式を示した図である。

発明を実施するための最良の形態

以下、図面を参照しながら本発明を実施するための最良の形態につき詳細に説明する。

図３は、本発明の好適実施例の構成図である。

まず、差分抽出処理部３０４は、後述するヘッダ解析部３０１、セグメントデータ対応付け部３０２、

又はリソース対応付け部３０３から指定された領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

ヘッダ解析部３０１は、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、そのDOS EXE ヘッダの領域及び ✳ EXEヘッダの領域を抽出し、それらの領域について差分抽出処理部３０４に対して差分抽出処理を指示する。また、ヘッダ解析部３０１は、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、上述の抽出されたDOS EXE ヘッダ及び ✳ EXEヘッダの内容を解析することにより後述する各テーブル部の領域を抽出し、抽出された各テーブル部領域についても差分抽出処理部３０４に対して差分抽出処理を指示する。更に、ヘッダ解析部３０１は、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、抽出されたセグメントテーブル部から後述するセグメント情報３０８を抽出して、それを後述するセグメントデータ対応付け部３０２及びリソース対応付け部３０３に設定し、また、抽出されたリソーステーブル部から後述するリソース情報３０９を抽出して、それを後述するリソース対応付け部３０３に設定する。

According PCT Article 21(6) this part should be omitted. (✳)
page 9
line 8, 13

セグメントデータ対応付け部３０２は、ヘッダ解析部３０１からのセグメント情報３０８に基づいて、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との間のセグメントデータの対応付けを行い、対応付けされた各セグメントデータのペアの領域について差分抽出処理部３０４に対して差分抽出処理を指示する。旧ＥＸＥファイル３０５に存在するセグメントデータに対応するものが新ＥＸＥファイル３０６に存在しない場合は、そのセグメントデータを削除すべき旨の差分情報を更新ファイル３０７にセーブする。逆に、新ＥＸＥファイル３０６に存在するセグメントデータに対応するものが旧ＥＸＥファイル３０５に存在しない場合は、そのセグメントデータを挿入すべき旨の差分情報を更新ファイル３０７にセーブする。

リソース対応付け部３０３は、ヘッダ解析部３０１からのセグメント情報３０８及びリソース情報３０９に基づいて、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との間のリソースの対応付けを行い、対応付けされた各リソースのペアの領域について差分抽出処理部３０４に対して差分抽出処理を指示する。旧ＥＸＥファイル３０５に存在するリソースに対応するものが新ＥＸＥファイル３０６に存在しない場合は、そのリソースを削除すべき旨の差分情報を更新ファイル３０７にセーブする。逆に、新

ＥＸＥファイル３０６に存在するリソースに対応するものが旧ＥＸＥファイル３０５に存在しない場合は、そのリソースを挿入すべき旨の差分情報を更新ファイル３０７にセーブする。

上述の構成を有する好適実施例の動作について、以下に順次説明する。

まず、図４は、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の共通のファイル形式を示した図である。

本好適実施例で対象とされるファイルは、コンピュータのためのマルチタスクオペレーティングシステムの１つである ✳ システムのもとで実行可能なＥＸＥファイルである。なお、本発明は、このようなファイル限定されるものではなく、後述するように、例えば ✳ オペレーティングシステムのもとで実行可能なＥＸＥファイルなども、本発明の適用範囲である。

図４に示される構成を有するＥＸＥファイルは、コンピュータのための従来からあるシングルタスクオペレーティングシステムの１つである ✳ システムのもとでも実行可能なように構築されている。そのため、このＥＸＥファイルは、図４に示されるように、DOS EXE 形式の領域と ✳ EXE形式の領域とから構成されている。

始めに、ファイルの前半部分には、DOS EXE 形式

According PCT Article 21(6) this part should be omitted. (✳)
page 11
line 12, 15, 20, 23

の領域が存在する。このDOS EXE 形式の領域は、✳✳ 用プログラムを実行するための各種情報と後述する ✳ EXEヘッダへのポインタ情報等が格納されるDOS EXE ヘッダ領域と、 ✳ システムのもとで動作するプログラムの本体が格納されるDOS ロードモジュール領域とから構成される。

ファイルの後半部分には、 ✳ EXE形式の領域が存在する。 ✳ EXE形式の領域は、 ✳ 用プログラムを実行するための各種情報と後述する各種テーブル部へのポインタ情報等が格納される ✳ EXEヘッダ領域、後述する各種テーブル部領域、 ✳ システムのもとで動作するプログラムの本体が格納されるセグメントデータ領域、及び ✳ プログラムで利用されるリソースと呼ばれるデータが格納されるリソース領域とから構成される。

本好適実施例においては、差分抽出処理が実行される場合、DOS EXE 形式の領域、 ✳ EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との差分抽出処理が実行され、その後、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との間のセグメントデータ及びリソースの対応付けが行われた上で、その対応付けされた各セグメントデータ及びリソースのペアの領域毎に差分抽出処理が実行されることが、大きな特徴である。このように、所定の

According PCT Article 21(6) this part should be omitted. (✳)
page 12
line 1-4, 7, 8, 10, 11, 13, 17

論理単位毎に差分抽出処理が実行されることにより、従来技術に比較して、生成される更新ファイル３０７のサイズを大幅に削減することが可能となる。

次に、ヘッダ解析部３０１、セグメントデータ対応付け部３０２、リソース対応付け部３０３、及び差分抽出処理部３０４の各部の動作について、順次説明する。

本好適実施例では、まず、ヘッダ解析部３０１が起動される。ヘッダ解析部３０１で実行される処理の動作フローチャートは、図５及び図６に示される。このヘッダ解析部３０１の動作は、特には図示しないコンピュータシステムにおいて、ＣＰＵ（中央処理装置）が、ディスク装置などに記憶された旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

まず、ステップＳ５０１で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット０に格納されている１ワードのコードが“ＭＺ”であるか否かが判定される（図７参照）。そのコードが“ＭＺ”でない場合、そのファイルはＥＸＥファイルではないため、ステップＳ５２３でエラー表示が行われ差分抽出処理を終了する。

上述のコードが“ＭＺ”である場合、ステップＳ

５０２で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット１８ｈ（“ｈ”は、それが付加された１ワードの値が１６進数であることを示す。）に格納されている値が４０ｈであるか否かが判定される（図７参照）。その値が４０ｈでない場合は、そのファイルは ✳ EXE形式のファイルではないため、ステップＳ５２３でエラー表示が行われ差分抽出処理を終了する。

上述の値が４０ｈである場合は、ステップＳ５０３で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット３Ｃｈに格納されている値で示される１ワードのオフセットアドレスが ✳ EXEの領域のアドレスであるか否かが判定される（図７参照）。そのアドレスが ✳ EXEの領域のアドレスでない場合は、そのファイルは ✳ EXE形式のファイルではないため、ステップＳ５２３でエラー表示が行われ差分抽出処理を終了する。

上述したアドレスが ✳ EXEの領域のアドレスである場合には、更にステップＳ５０４で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のそれぞれについて、 ✳ EXEヘッダ内の ✳ EXE相対オフセット３６ｈに格納されている１ワードの値が１であるか否かが判定される（図７参照）。ここで、

According PCT Article 21(6) this part should be omitted. (✳)
page 14
line 7, 14, 16, 17, 20, 23

✳ EXE相対オフセットとは、✳ EXE形式の領域の先頭アドレスからの相対オフセットをいう。その値が１でない場合には、そのファイルは✳ EXE形式のファイルではないため、ステップＳ５２３でエラー表示が行われ差分抽出処理を終了する。

上述した値が１である場合には、ステップＳ５０５で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき、DOS EXE 形式のデータ領域が、ファイルオフセット０から、DOS EXE ヘッダ内のファイルオフセット３Ｃｈの内容として示される✳ EXEの領域のアドレスの手前のアドレスまでの領域として抽出される（図４参照）。

次に、ヘッダ解析部３０１は、ステップＳ５０６で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき抽出されたDOS EXE 形式のデータ領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理により、ヘッダ解析部３０１から渡されたDOS EXE 形式のデータ領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

次に、ステップＳ５０７で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６における図７に示

According PCT Article 21(6) this part should be omitted. (✳)
page 15
line 1, 3, 11

される ✳ EXEヘッダの内容のうち、●印が付与されたデータが抽出され、それぞれ図８に示される各変数に格納される。この変数は、特には図示しないメモリに保持される。これらの変数については後述する。

続いて、ステップＳ５０８で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき、✳ EXEヘッダ領域が、前述したDOS EXE ヘッダ内のファイルオフセット３Ｃｈの内容として示される ✳ EXEの領域のアドレスから、変数segtbl off（図８参照）で示されるセグメントテーブル部の先頭アドレスの手前のアドレスまでの領域として抽出される（図４参照）。

次に、ヘッダ解析部３０１は、ステップＳ５０９で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出された ✳ EXEヘッダ領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理により、ヘッダ解析部３０１から渡された ✳ EXEヘッダ領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

続いて、ステップＳ５１０で、旧ＥＸＥファイル

According PCT Article 21(6) this part should be omitted. (✳)
page 16
line 1, 8, 10, 17, 20

３０５と新ＥＸＥファイル３０６のそれぞれについて、変数segtbl off及びsegtbl cnt（図８参照）に基づいて抽出されるセグメントテーブル部（図４参照）が解析される。セグメントテーブル部のデータ構造は、図９に示される。セグメントテーブル部は、セグメントデータの先頭アドレスを示す１ワードのファイルオフセットデータ、セグメントデータの実サイズを示す１ワードの実セグメントデータサイズデータ、及びセグメントデータのタイプ及び属性を示す１ワードのセグメントタイプ／属性データからなるデータセットを、後述するセグメントデータのそれぞれに対応して有している。ステップＳ５１０では、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、このデータ構造が、例えばデータ構造体の形式で、セグメント情報３０８（図３参照）として、後述するセグメントデータ対応付け部３０２及びリソース対応付け部３０３からアクセス可能な特には図示しないメモリに設定される。

次に、ステップＳ５１１で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、上述のセグメントテーブル部領域（図４参照）が抽出される。この領域は、変数segtbl offで示されるセグメントテーブル部の先頭のオフセットアドレスから、変数entrytbl offで示されるエキスポートエ

ントリテーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図８、図４参照）。

続いて、ヘッダ解析部３０１は、ステップＳ５１２で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出されたセグメントテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理により、ヘッダ解析部３０１から渡されたセグメントテーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

次に、ステップＳ５１３で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき、エキスポートエントリテーブル部領域（図４参照）が抽出される。この領域は、変数entrytbl offで示されるエキスポートエントリテーブル部の先頭のオフセットアドレスから、変数resnamtbl off で示される常駐ネームテーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図８、図４参照）。

エキスポートエントリテーブル部には、それが含まれる＊ EXE形式のプログラムが、他の＊プロ

According PCT Article 21(6) this part should be omitted. (＊)
page 18
line 25

グラムから参照され得るプログラム、例えばＤＬＬ（ダイナミックリンクライブラリ）プログラムである場合に、他の✳プログラムがこのテーブル部が含まれるプログラムの一部として提供される関数を参照できるように、その関数へのファイル内のセグメント番号とオフセットが格納される。

続いて、ヘッダ解析部３０１は、ステップＳ５１４で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出されたエキスポートエントリテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理により、ヘッダ解析部３０１から渡されたセグメントテーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

次に、ステップＳ５１５で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、常駐ネームテーブル部及び非常駐ネームテーブル部（図４参照）の領域が、１つの領域として抽出される。この領域は、変数resnamtbloffで示される常駐ネームテーブル部の先頭のオフセットアドレスから、変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスの手前のアドレスま

According PCT Article 21(6) this part should be omitted. (✳)
page 19
line 3

での領域として抽出される（図８、図４参照）。

常駐ネームテーブル部及び非常駐ネームテーブル部は、図１０に示される共通の構成を有し、常駐ネームテーブル部には、プログラム実行時にメモリに常駐するモジュール名とそのバイト数及びエントリされるテーブルのインデクスが格納され、非常駐ネームテーブル部には、プログラム実行時に必要に応じてそのプログラム自体が呼出すモジュール名とそのバイト数及びエントリされるテーブル（モジュール）のインデクスが格納される。

続いて、ヘッダ解析部３０１は、ステップＳ５１６で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出された常駐／非常駐ネームテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、ヘッダ解析部３０１から渡された常駐／非常駐ネームテーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

次に、ステップＳ５１７において、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、モジュール参照テーブル部及びインポートネームテーブル部（図４参照）の領域が、１つの領

域として抽出される。この領域は、変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスから、変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図８、図４参照）。

インポートネームテーブル部には、図１１に示されるように、それが含まれるプログラムが参照する他のＤＬＬプログラムのモジュール名とそのバイト長が格納され、モジュール参照テーブル部には、図１２に示されるように、インポートネームテーブル部の各要素へのオフセット値がワード配列として格納される。

続いて、ヘッダ解析部３０１は、ステップＳ５１８で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出されたモジュール参照／インポートネームテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、ヘッダ解析部３０１から渡されたモジュール参照／インポートネームテーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

次に、ステップＳ５１９で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、変数resrctbl off及びresrctbl cnt（図８参照）に基づいて抽出されるリソーステーブル部（図４参照）が解析される。リソーステーブル部のデータ構造は、図１３に示される。リソーステーブル部は、リソースのタイプの識別子を示す１ワードのリソースタイプＩＤデータと、リソースネームの識別子を示す１ワードのリソースネームＩＤデータとからなるデータセットを、後述するリソースのそれぞれに対応して有している。ステップＳ５１９では、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、このデータ構造が、リソース情報３０９（図３参照）として、後述するリソース対応付け部３０３からアクセス可能な特には図示しないメモリに設定される。

次に、ステップＳ５２０で、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて、上述のリソーステーブル部領域（図４参照）が抽出される。この領域は、変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスから、変数resrctbl cntで示されるリソースエントリ数分だけ進んだアドレスまでの領域として抽出される（図８、図４参照）。

続いて、ヘッダ解析部３０１は、ステップＳ５２

１で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき抽出されたリソーステーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理により、ヘッダ解析部３０１から渡されたリソーステーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

以上のようにして、ヘッダ解析部３０１によって、DOS EXE 形式の領域、＊ EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との差分抽出処理が実行される。

そして、ヘッダ解析部３０１は、ステップＳ５２２で、図３のセグメントデータ対応付け部３０２を起動して、処理を終了する。

セグメントデータ対応付け部３０２で実行される処理の動作フローチャートは図１４に示される。このセグメントデータ対応付け部３０２の動作は、ヘッダ解析部３０１の場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６に対して、メモリに記憶

According PCT Article 21(6) this part should be omitted. (＊)
page 23
line 12

された制御プログラムを実行する動作の一部として実現される。

ここで、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６とで対応するセグメントデータがある場合、変更されていないセグメントについてはそれらの内容が一致し、変更されたセグメントについてのみ差分が生ずる。そこで、セグメントデータ対応付け部３０２では、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との間で、セグメントデータの属性やその主要部のデータを比較することによりその対応付けを行い、その対応付けしたセグメントデータ領域毎に差分抽出処理部３０４に対して差分抽出処理を指示する。

まず、ステップＳ１４０１で、前述した図６のステップＳ５１０でヘッダ解析部３０１から設定されたセグメント情報３０８に基づいて、セグメントの対応付けが行われる。

この処理では、まず、旧ＥＸＥファイル３０５のセグメント情報３０８（図９参照）に基づいて、旧ＥＸＥファイル３０５のセグメントが一列に並べられる。なお、セグメントは、リンカによってファイルオフセットが増大する方向に並べられている。並べられた旧ＥＸＥファイル３０５のセグメントの例を図１５Ａに示す。

次に、新ＥＸＥファイル３０６のセグメント情報

３０８（図９参照）に基づいて、新ＥＸＥファイル３０６のセグメントが一列に並べられる。並べられた新ＥＸＥファイル３０６のセグメントの例を図１５Ｂに示す。

図１５Ａ、図１５Ｂの例では、新ＥＸＥファイル３０６においては、旧ＥＸＥファイル３０５に対して、セグメント番号３のセグメントが追加されている。従って、この状態が検出される必要がある。

今、本好適実施例では、旧ＥＸＥファイル３０５から新ＥＸＥファイル３０６への更新は、セグメントを削除、置換、又は追加することにより行われるとする。従って、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のセグメントの対応関係が図１６Ａに示されるようなものになることはあり得ず、その対応関係は例えば図１６Ｂに示される如くとなる。この例では、旧ＥＸＥファイル３０５のセグメント番号４のセグメントが削除され、新ＥＸＥファイル３０６のセグメント番号２のセグメントが追加され、その他のセグメントのペアについては、変更がないか、又は置換がなされている。

そこで、一列に並べられた旧ＥＸＥファイル３０５のセグメント及び新ＥＸＥファイル３０６のセグメントについて、以下の対応付け処理がなされる。

処理１：旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のセグメント数が比較される。

処理２：セグメント数が同じか、新ＥＸＥファイル３０６のセグメント数の方が多い場合は、旧ＥＸＥファイル３０５で定義されているセグメントをキーとして、新ＥＸＥファイル３０６中のセグメントがサーチされる。旧ＥＸＥファイル３０５のセグメント数の方が多い場合は、新ＥＸＥファイル３０６で定義されているセグメントをキーとして、旧ＥＸＥファイル３０５中のセグメントがサーチされる。

処理３：サーチ対象であるセグメントにおいて、現在のキーとされているセグメントに対し、セグメントタイプ／属性が同じで、かつ、セグメントデータの最初の１６バイトの内容が一致するものがペアセグメントとされる。ここで、図１７に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを含むセグメントデータが存在する。キーとされているセグメント又はサーチ対象であるセグメントの何れかがこのようなデータである場合には、このリロケーションテーブルの直後の最初の１６バイトの内容が一致するものがペアセグメントとされ、リロケーションテーブルについては、後述するように別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、セグメント情報３０８のセグメントタイプ／属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによっ

て判別できる。

サーチ対象であるセグメントにおいて、上記一致関係が成立するセグメントがない場合には、キーとなったセグメントについては、サーチは諦められ、そのセグメントは旧ＥＸＥファイル３０５の孤立セグメントとされる。そして、次のセグメントがキーとされて、サーチが行われる。

全てのサーチが終了した時点で、新ＥＸＥファイル３０６においてペアとならなかったセグメントは、新ＥＸＥファイル３０６の孤立セグメントとされる。

このようにして得られた、ペアセグメント、旧ＥＸＥファイル３０５の孤立セグメント、及び新ＥＸＥファイル３０６の孤立セグメントが、セグメント対応付けデータとしてメモリなどに保持される。

上述のようにして、図１４のステップＳ１４０１で、全てのサーチが終了した後、セグメント対応付けデータのそれぞれに対して、以下に示されるステップＳ１４０２～Ｓ１４１６の処理が繰り返し実行される。

即ち、まず、ステップＳ１４０２で、１つのセグメント対応付けデータがメモリなどから抽出される。

セグメント対応付けデータが存在するならステップＳ１４０３の判定はＹＥＳとなり、続いて、ステップＳ１４０４で、そのセグメント対応付けデータがペアセグメントか否かが判定される。

セグメント対応付けデータがペアセグメントであってステップＳ１４０４の判定がＹＥＳの場合には、ステップＳ１４０８、Ｓ１４１０、及びＳ１４１２で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の両方にリロケーション情報が設定されている場合は、ステップＳ１４０８及びＳ１４１０の判定がＮＯとなりステップＳ１４１２の判定がＹＥＳとなって、ステップＳ１４１３が実行される。ステップＳ１４１３では、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき、リロケーションテーブル部領域が抽出される。この領域は、セグメントデータの先頭アドレスから、その先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、セグメントデータ対応付け部３０２は、ステップＳ１４１４で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、セグメントデータ対応付け部３０２から渡されたリロケーションテーブル部領

域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

旧ＥＸＥファイル３０５のみにリロケーション情報が設定されている場合は、ステップＳ１４０８の判定がＹＥＳとなり、ステップＳ１４０９が実行される。ステップＳ１４０９では、旧ＥＸＥファイル３０５のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

新ＥＸＥファイル３０６のみにリロケーション情報が設定されている場合は、ステップＳ１４０８の判定がＮＯ、ステップＳ１４１０の判定がＹＥＳとなり、ステップＳ１４１１が実行される。ステップＳ１４１１においては、新ＥＸＥファイル３０６のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

以上のステップＳ１４１４、Ｓ１４０９、又はＳ１４１１の処理の後、又はステップＳ１４１２の判

定がＮＯとなった後、ステップＳ１４１５が実行される。ステップＳ１４１５では、リロケーション情報を除いたセグメントデータ本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはセグメントデータの先頭アドレスから、リロケーション情報が設定されている場合にはセグメントデータの先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報３０８から抽出される次のセグメント番号のセグメントデータの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

続いて、セグメントデータ対応付け部３０２は、ステップＳ１４１６で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき抽出されたセグメントデータ領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、セグメントデータ対応付け部３０２から渡されたセグメントデータ領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

一方、前述したステップＳ１４０４の判定において、セグメント対応付けデータがペアセグメントで

ない場合であってステップＳ１４０４の判定がＮＯの場合には、更にステップＳ１４０５で、セグメント対応付けデータが旧ＥＸＥファイル３０５の孤立セグメントであるか否かが判定される。

セグメント対応付けデータが旧ＥＸＥファイル３０５の孤立セグメントであってステップＳ１４０５の判定がＹＥＳなら、ステップＳ１４０６が実行される。ステップＳ１４０６では、旧ＥＸＥファイル３０５の孤立セグメントの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

セグメント対応付けデータが旧ＥＸＥファイル３０５の孤立セグメントでなくステップＳ１４０５の判定がＮＯなら、それは新ＥＸＥファイル３０６の孤立セグメントであり、この場合には、ステップＳ１４０７が実行される。ステップＳ１４０７では、新ＥＸＥファイル３０６の孤立セグメントの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

以上のステップＳ１４０２～Ｓ１４１６の処理が、全てのセグメント対応付けデータに対して繰り返し実行される。そして、セグメント対応付けデータがなくなると、ステップＳ１４０３の判定がＮＯとなる。このようにして、セグメントデータ対応付け部３０２によって、旧ＥＸＥファイル３０５と新ＥＸ

Ｅファイル３０６との間で、セグメントデータの対応付けが行われた上で、その対応付けされたセグメントデータ領域毎に差分抽出処理部３０４で差分抽出処理が実行される。

そして、セグメントデータ対応付け部３０２は、ステップＳ１４１７で、図３のリソース対応付け部３０３を起動して、処理を終了する。

リソース対応付け部３０３で実行される処理の動作フローチャートは、図１８に示される。このリソース対応付け部３０３の動作は、ヘッダ解析部３０１などの場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

ここで、リソースはセグメントデータの一属性として規定され、従って、前述したセグメントデータの場合と同様、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６とで対応するリソースがある場合、変更されていないリソースについてはそれらの内容が一致し、変更されたリソースについてのみ差分が生ずる。そこで、リソース対応付け部３０３でも、セグメントデータ対応付け部３０２の場合と同様に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６との間で、リソースの対応付けを行い、その対応付

けしたリソース領域毎に差分抽出処理部３０４に対して差分抽出処理を指示する。

この場合、特にリソースでは、前述したリソーステーブル部（図１３参照）において、リソースネームＩＤが設定されている。そこで、リソース対応付け部３０３は、リソースネームＩＤを比較することによりその対応付けを行い、その対応付けしたリソース領域毎に差分抽出処理部３０４に対して差分抽出処理を指示する。

まず、ステップＳ１８０１で、前述した図６のステップＳ５１０及びＳ５１９でヘッダ解析部３０１から設定されたセグメント情報３０８及びリソース情報３０９に基づいて、リソースの対応付けが行われる。

この処理では、まず、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のセグメント情報３０８（図９参照）並びにリソース情報３０９に基づいて、それらのリソースが抽出される。ここで、｛（セグメント情報３０８として設定されているセグメントの数）－（リソース情報３０９として設定されているリソースの数）＋１｝番目以降のセグメント情報３０８が、リソース情報３０９として設定されている各リソースネームＩＤに対応するリソースを示している。旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６において抽出されたリソースの例を図１

９Ａに示す。

次に、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６において抽出されたリソースが、セグメント情報３０８から抽出されるファイルオフセット順にソートされる。ソートされた旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のリソースの例を図１９Ｂに示す。

次に、ソートされた旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６のリソースにおいて、同じリソースネームＩＤを有するもの同士が対応付けられる。この結果、図１９Ａ、図１９Ｂの例における対応関係は、図２０に示される如くとなる。この例では、旧ＥＸＥファイル３０５のリソースネームＩＤ＝６のリソースが削除され、新ＥＸＥファイル３０６のリソースネームＩＤ＝９及び３８のリソースが追加され、その他のリソースのペアについては、変更がないか、又は置換がなされている。

そこで、ソートされた旧ＥＸＥファイル３０５のリソース及び新ＥＸＥファイル３０６のリソースについて、以下の対応付け処理がなされる。

処理１：旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のリソース数が比較される。

処理２：リソース数が同じか、新ＥＸＥファイル３０６のリソース数の方が多い場合は、旧ＥＸＥファイル３０５で定義されているリソースをキーとし

て、新ＥＸＥファイル３０６中のリソースがサーチされる。旧ＥＸＥファイル３０５のリソース数の方が多い場合は、新ＥＸＥファイル３０６で定義されているリソースをキーとして、旧ＥＸＥファイル３０５中のリソースがサーチされる。

処理３：サーチ対象であるリソースにおいて、現在のキーとされているリソースに対し、リソースネームＩＤが同じものがペアリソースとされる。ここで、前述したセグメントデータの場合と同様、図１７に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを有するリソースが存在する。キーとされているリソース又はサーチ対象であるリソースの何れかがこのようなデータである場合には、そのリロケーションテーブルについては、後述するように別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、リソースに対応するセグメント情報３０８のセグメントタイプ／属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによって判別できる。

サーチ対象であるリソースにおいて、上記一致関係が成立するリソースがない場合には、キーとなったリソースについては、サーチは諦められ、そのリソースは旧ＥＸＥファイル３０５の孤立リソースとされる。そして、次のリソースがキーとされて、サ

ーチが行われる。

全てのサーチが終了した時点で、新ＥＸＥファイル３０６においてペアとならなかったリソースは、新ＥＸＥファイル３０６の孤立リソースとされる。

このようにして得られた、ペアリソース、旧ＥＸＥファイル３０５の孤立リソース、及び新ＥＸＥファイル３０６の孤立リソースが、リソース対応付けデータとしてメモリなどに保持される。

上述のようにして、図１８のステップＳ１８０１で、全てのサーチが終了した後、リソース対応付けデータのそれぞれに対して、以下に示されるステップＳ１８０２～Ｓ１８１６の処理が繰り返し実行される。これらの処理波、セグメント対応付けデータに対する図１４のステップＳ１４０２～Ｓ１４１６の処理と同じである。

即ち、まず、ステップＳ１８０２で、１つのリソース対応付けデータがメモリなどから抽出される。

リソース対応付けデータが存在するならステップＳ１８０３の判定はＹＥＳとなり、続いて、ステップＳ１８０４で、そのリソース対応付けデータがペアリソースか否かが判定される。

リソース対応付けデータがペアリソースであってステップＳ１８０４の判定がＹＥＳの場合には、ステップＳ１８０８、Ｓ１８１０、及びＳ１８１２で、旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０

６の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の両方にリロケーション情報が設定されている場合は、ステップＳ１８０８及びＳ１８１０の判定がＮＯとなりステップＳ１８１２の判定がＹＥＳとなって、ステップＳ１８１３が実行される。ステップＳ１８１３では、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき、リロケーションテーブル部領域が抽出される。この領域は、リソースの先頭アドレスから、その先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、リソース対応付け部３０３は、ステップＳ１８１４で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、リソース対応付け部３０３から渡されたリロケーションテーブル部領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

旧ＥＸＥファイル３０５のみにリロケーション情

報が設定されている場合は、ステップＳ１８０８の判定がＹＥＳとなり、ステップＳ１８０９が実行される。ステップＳ１８０９では、旧ＥＸＥファイル３０５のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

新ＥＸＥファイル３０６のみにリロケーション情報が設定されている場合は、ステップＳ１８０８の判定がＮＯ、ステップＳ１８１０の判定がＹＥＳとなり、ステップＳ１８１１が実行される。ステップＳ１８１１においては、新ＥＸＥファイル３０６のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

以上のステップＳ１８１４、Ｓ１８０９、又はＳ１８１１の処理の後、又はステップＳ１８１２の判定がＮＯとなった後、ステップＳ１８１５が実行される。ステップＳ１８１５では、リロケーション情報を除いたリソース本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはリソースの先頭アドレスから、リロケーショ

ン情報が設定されている場合にはリソースの先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報３０８及びリソース情報３０８から抽出される次のリソース番号のリソースの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

続いて、リソース対応付け部３０３は、ステップＳ１８１６で、差分抽出処理部３０４をコールすると共に、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のそれぞれにつき抽出されたリソース領域を差分抽出処理部３０４に渡す。この結果、差分抽出処理部３０４は、後述する差分抽出処理によって、リソース対応付け部３０３から渡されたリソース領域の先頭から末尾まで、新ＥＸＥファイル３０６の旧ＥＸＥファイル３０５に対するバイト単位の差分を抽出し、その差分を更新ファイル３０７にセーブする。

一方、前述したステップＳ１８０４の判定において、リソース対応付けデータがペアリソースでない場合であってステップＳ１８０４の判定がＮＯの場合には、更にステップＳ１８０５で、リソース対応付けデータが旧ＥＸＥファイル３０５の孤立リソースであるか否かが判定される。

リソース対応付けデータが旧ＥＸＥファイル３０

5の孤立リソースであってステップＳ１８０５の判定がＹＥＳなら、ステップＳ１８０６が実行される。ステップＳ１８０６では、旧ＥＸＥファイル３０５の孤立リソースの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

リソース対応付けデータが旧ＥＸＥファイル３０５の孤立リソースでなくステップＳ１８０５の判定がＮＯなら、それは新ＥＸＥファイル３０６の孤立リソースであり、この場合には、ステップＳ１８０７が実行される。ステップＳ１８０７では、新ＥＸＥファイル３０６の孤立リソースの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル３０７にセーブされる。

以上のステップＳ１８０２～Ｓ１８１６の処理が、全てのリソース対応付けデータに対して繰り返し実行される。そして、リソース対応付けデータがなくなると、ステップＳ１８０３の判定がＮＯとなり、全ての処理を終了する。

最後に、上述したようにしてヘッダ解析部３０１、セグメントデータ対応付け部３０２、又はリソース対応付け部３０３からコールされる差分抽出処理部３０４の処理について説明する。

差分抽出処理部３０４で実行される差分抽出処理の動作フローチャートは図２１に、また、差分抽出

処理において実行される比較処理の動作フローチャートは図２２に示される。これらの動作は、ヘッダ解析部３０１などの場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル３０５及び新ＥＸＥファイル３０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

今、例えば図２４Ａに示されるように、旧ＥＸＥファイル３０５の領域として、バイトデータ列ａｂ・・・ｃｄｅ・・・・からなる領域が指定され、一方、新ＥＸＥファイル３０６の領域として、バイトデータｃとｄの間にバイトデータ列ＩＩ・・・Ｉが挿入された領域が指定されたとする。

図２１において、まず、ステップＳ２１０１で、指定された領域において比較すべきデータがなくなったか否かが判定される。

比較すべきデータがある場合、ステップＳ２１０２で、新ＥＸＥファイル３０６において指定された領域を比較側、旧ＥＸＥファイル３０５において指定された領域を比較対象側として、図２３に示されるように、比較側のバイトデータをキーとして比較処理が実行される。

比較処理の動作フローチャートは図２２に示される。

図２２のステップＳ２２０１で、特には図示しないメモリに確保されている変数srcendに、比較側の領域における終了位置又は後述する現在の比較側の比較開始位置より１０２４バイト前方の位置のうち、現在の比較側の比較開始位置から近い方の位置のアドレスが格納される。同様にして、特には図示しないメモリに確保されている変数distend に、比較対象側の領域における終了位置又は後述する現在の比較対象側の比較開始位置より１０２４バイト前方の位置のうち、現在の比較対象側の比較開始位置から近い方の位置のアドレスが格納される。

次に、ステップＳ２２０２で、特には図示しないメモリに確保されている変数src に、現在の比較側の比較開始位置のアドレスが格納される。図２４Ａの例では、変数src には最初は新ＥＸＥファイル３０６の指定領域における先頭のバイトデータａのアドレスが格納される。同様にして、特には図示しないメモリに確保される変数distとnextdistに、現在の比較対象側の比較開始位置のアドレスが格納される。図２４Ａの例では、変数distとnextdistには最初は旧ＥＸＥファイル３０５の指定領域における先頭のバイトデータａのアドレスが格納される。

その後、図２４Ａに示されるように、新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５とで内容が一致しているバイトデータａｂ・・・ｃの範囲にお

いては、ステップＳ２２０６で変数src 、dist、及びnextdistの値が順次インクリメントされながら、ステップＳ２２０３→Ｓ２２０４→Ｓ２２０５→Ｓ２２０６→Ｓ２２０３の処理が繰り返されることにより、図２４ＡのＴ１～Ｔ３に示されるように、変数src 及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

そして、図２４Ａに示されるように、ステップＳ２２０６で、比較側である新ＥＸＥファイル３０６において変数src の値が挿入されたバイトデータＩＩ・・・Ｉのうち最初のデータのアドレスにされ、比較対象側である旧ＥＸＥファイル３０５において変数distの値がバイトデータｄのアドレスにされた後、次にステップＳ２２０４が実行される時点で、その判定がＮＯとなる。

この後は、ステップＳ２２０８で変数distの値がインクリメントされながら、ステップＳ２２０４→Ｓ２２０７→Ｓ２２０８→Ｓ２２０９→Ｓ２２０４→Ｓ２２０７→Ｓ２２０８の処理が繰り返されることにより、変数src で示されるアドレスにおける新ＥＸＥファイル３０６で挿入された第１番目のバイトデータＩをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧ＥＸＥファイル３０５上のバイトデータｄｅ・・・の各値とが、それらが一致してステップＳ２２０

４の判定がＹＥＳとなるまで、順次比較される。この動作は、図２４ＡのＴ４、Ｔ５に示される。

この比較処理の繰り返しの結果、新ＥＸＥファイル３０６で挿入された第１番目のバイトデータＩに一致するバイトデータが旧ＥＸＥファイル３０５上で見つかれなければ、やがて変数distで示されるアドレスが変数distend で示される比較終了アドレスを越え、ステップＳ２２０９の判定がＹＥＳとなる。

この場合には、新ＥＸＥファイル３０６で挿入された第１番目のバイトデータＩをキーとするサーチは諦められ、ステップＳ２２１０で、変数src の値がインクリメントされることにより新ＥＸＥファイル３０６で挿入された第２番目のバイトデータＩがキーとされ、また、変数distに変数nextdistのアドレスが挿入されることにより、比較対象側である旧ＥＸＥファイル３０５上の比較開始位置が初めて不一致が発生したバイトデータｄのアドレスに戻される。

そして、変数src によって示されるアドレスが変数srcendで示される比較終了アドレスを越えておらずステップＳ２２０３の判定がＮＯであれば、再び、ステップＳ２２０８で変数distの値がインクリメントされながら、ステップＳ２２０４→Ｓ２２０７→Ｓ２２０８→Ｓ２２０９→Ｓ２２０４→Ｓ２２０７→Ｓ２２０８の処理が繰り返されることにより、変

数src で示されるアドレスにおける新ＥＸＥファイル３０６で挿入された第２番目のバイトデータＩをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧ＥＸＥファイル３０５上のバイトデータｄｅ・・・の各値とが、それらが一致してステップＳ２２０４の判定がＹＥＳとなるまで、順次比較される。この動作は、図２４ＡのＴ６、Ｔ７に示される。

以下、ステップＳ２２０９の判定がＹＥＳとなる毎に、新ＥＸＥファイル３０６で挿入された第３番目以降の各バイトデータＩについて同様の比較処理が繰り返される（図２４ＡのＴ８、Ｔ９）。

やがてステップＳ２２１０で、変数src に新ＥＸＥファイル３０６上のバイトデータｄのアドレスが代入され、変数distに変数nextdistで示される旧ＥＸＥファイル３０５上のバイトデータｄのアドレスが代入された後、ステップＳ２２０４が実行された時点で、変数src で示されるアドレスにおける新ＥＸＥファイル３０６のバイトデータｄと変数distで示されるアドレスにおける旧ＥＸＥファイル３０５のバイトデータｄとが一致し、その判定がＹＥＳとなる。この一致状態は、最初に新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃが一致し、次に、新ＥＸＥファイル３０６で挿入されたバイトデータＩＩ・・・Ｉが旧ＥＸＥ

ファイル３０５上のバイトデータｄｅ・・・と不一致となった後の一致状態であるため、ステップＳ２２０５の判定がＹＥＳとなる。

この結果、ステップＳ２２１１で、図２４Ａの新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点（バイトデータａのアドレス）と一致バイト数が算出され、また、新ＥＸＥファイル３０６で挿入されたバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

上述のように、一致部分と不一致部分のペアが見つかってステップＳ２２０５の判定がＹＥＳとなる時点で、また、図２８Ａ、図２８Ｂの説明で後述するように、不一致部分と一致部分のペアが見つかってステップＳ２２０７の判定がＹＥＳとなる時点で、一致部分の一致開始点と一致バイト数、及び不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了される。

以上の動作の結果、新ＥＸＥファイル３０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル３０５の指定領域の各バイトデータに対する図２１のステップＳ２１０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０３で、上述の一致バイト数が６４バイ

ト以上であるか否かが判定される。

一致バイト数が６４バイト以上であってステップＳ２１０３の判定がＹＥＳとなった場合には、ステップＳ２１０９で、上述の不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。例えば図２４Ａの新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃで示される一致部分の一致バイト数が６４バイト以上である場合には、新ＥＸＥファイル３０６において、旧ＥＸＥファイル３０５のバイトデータｃで示されるアドレスの直後にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が更新ファイル３０７へライトされる。

一致バイト数が６４バイトより少なくステップＳ２１０３の判定がＮＯとなった場合には、ステップＳ２１０４で、前述したステップＳ２１０２の場合とは逆に、旧ＥＸＥファイル３０５の指定領域を比較側、新ＥＸＥファイル３０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述した図２２で示される比較処理が実行される。

この比較処理の例として、前述した図２４Ａに対応する動作が図２４Ｂに示される。

始めに、ステップＳ２２０１とステップＳ２２０２で、変数srcendとdistend 、変数src 、及び変数distとnextdistに、図２４Ａの場合と同様の値が設

定される。

その後、前述したステップＳ２２０３→Ｓ２２０４→Ｓ２２０５→Ｓ２２０６→Ｓ２２０３の処理が繰り返されることにより、図２４ＢのＴ１～Ｔ３に示されるように、変数src 及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

そして、図２４Ｂに示されるように、ステップＳ２２０６で、比較側である旧ＥＸＥファイル３０５において変数src の値がｄのアドレスにされ、比較対象側である新ＥＸＥファイル３０６において変数distの値が挿入されたバイトデータＩＩ・・・Ｉのうち最初のデータのアドレスにされた後、次にステップＳ２２０４が実行される時点で、その判定がＮＯとなる。

この後は、ステップＳ２２０８で変数distの値がインクリメントされながら、ステップＳ２２０４→Ｓ２２０７→Ｓ２２０８→Ｓ２２０９→Ｓ２２０４→Ｓ２２０７→Ｓ２２０８の処理が繰り返されることにより、変数src で示されるアドレスにおける旧ＥＸＥファイル３０５のバイトデータｄをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける新ＥＸＥファイル３０６上のバイトデータＩＩ・・・Ｉの各値とが、それらが一致してステップＳ２２０４の判定がＹＥＳと

なるまで、順次比較される。この動作は、図２４ＢのＴ４～Ｔ６に示される。

この比較処理の繰り返しの結果、ステップＳ２２０８で、変数distに新ＥＸＥファイル３０６上のバイトデータｄのアドレスが代入された後、ステップＳ２２０４が実行された時点で、変数src で示されるアドレスにおける旧ＥＸＥファイル３０５のバイトデータｄと変数distで示されるアドレスにおける新ＥＸＥファイル３０６のバイトデータｄとが一致し、その判定がＹＥＳとなる。この一致状態は、最初に旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のバイトデータａｂ・・・ｃが一致し、次に、新ＥＸＥファイル３０６で挿入されたバイトデータＩＩ・・・Ｉが旧ＥＸＥファイル３０５上のバイトデータｄと不一致となった後の一致状態であるため、ステップＳ２２０５の判定がＹＥＳとなる。

この結果、ステップＳ２２１１で、図２４Ｂの旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点（バイトデータａのアドレス）と一致バイト数が算出され、また、新ＥＸＥファイル３０６で挿入されたバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、旧ＥＸＥファイル３０５の指

定領域の各バイトデータをキーとする新ＥＸＥファイル３０６の指定領域の各バイトデータに対する図２１のステップＳ２１０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０５で、全てのバイトが不一致であるか否かが判定される。

全てのバイトが不一致であってステップＳ２１０５の判定がＹＥＳとなった場合には、ステップＳ２１０９で、上述の不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。

一方、図２４Ａ、図２４Ｂの例のように、全てのバイトが不一致でなくステップＳ２１０５の判定がＮＯとなった場合には、ステップＳ２１０６で、まず、次式によって、新ＥＸＥファイル３０６のバイトデータをキーとする前述したステップＳ２１０２の比較処理における一致バイト数の割合RNEWOLD が算出される。

RNEWOLD=（新→旧の一致バイト数）
／｛（新→旧の一致バイト数
＋（新→旧の不一致バイト数）｝
・・・（１）

次に、次式によって、旧ＥＸＥファイル３０５のバイトデータをキーとする前述したステップＳ２１０

4の比較処理における一致バイト数の割合ROLDNEWが算出される。

ROLDNEW=（旧→新の一致バイト数）
／｛（旧→新の一致バイト数
＋（旧→新の不一致バイト数）｝
・・・（2）

そして、次式によって、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

RNEWOLD/ROLDNEW ＜1 ？ ・・・（3）

今、更新ファイル３０７には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さくステップＳ２１０６の判定がＹＥＳの場合には、ステップＳ２１０７で、旧ＥＸＥファイル３０５をキーとした場合のステップＳ２１０４の比較処理の結果が選択され、ステッ

プＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。一方、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW 以上であってステップＳ２１０６の判定がＮＯの場合には、ステップＳ２１０８で、新ＥＸＥファイル３０６をキーとした場合のステップＳ２１０２の比較処理の結果が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。

但し、図２４Ａ及び図２４Ｂの例では、RNEWOLD=ROLDNEW となるため、ステップＳ２１０８で、新ＥＸＥファイル３０６をキーとした場合のステップＳ２１０２の比較処理の結果が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル３０６において旧ＥＸＥファイル３０５のバイトデータｃで示されるアドレスの直後にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が、更新ファイル３０７へライトされる。

ステップＳ２１０９の処理の後は、ステップＳ２１１０で、次の比較開始位置が設定され、ステップＳ２１０１以降の処理に戻る。図２４Ａの例では、

次の比較開始位置は、新ＥＸＥファイル３０６では、最後に図２２のステップＳ２２０４の判定がＹＥＳとなったときの変数src に格納されているバイトデータｄのアドレス、旧ＥＸＥファイル３０５でも、同じくそのときの変数distに格納されているバイトデータｄのアドレスである。以後、これらの比較開始位置を起点として、前述した場合と全く同様の比較処理が繰り返される。

新ＥＸＥファイル３０６の指定領域と旧ＥＸＥファイル３０５の指定領域において、比較するバイトデータが存在しなくなったら、ステップＳ２１０１の判定がＹＥＳとなって、これらの指定領域における差分抽出処理を終了する。

次に、図２５Ａに示されるように、新ＥＸＥファイル３０６の指定領域において、旧ＥＸＥファイル３０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＤＤ・・・Ｄの領域が削除された場合を考える。

この場合、まず、図２５Ａに示されるように、図２１のステップＳ２１０２で、新ＥＸＥファイル３０６において指定された領域を比較側、旧ＥＸＥファイル３０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。

この比較処理は、前述した図２４Ｂの場合に対し、

比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図２５Ａの新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で削除されたバイトデータＤＤ・・・Ｄで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、新ＥＸＥファイル３０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル３０５の指定領域の各バイトデータに対する図２１のステップＳ２１０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

一致バイト数が６４バイト以上であってステップＳ２１０３の判定がＹＥＳとなった場合には、ステップＳ２１０９で、上述の不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。

一致バイト数が６４バイトより少なくステップＳ２１０３の判定がＮＯとなった場合には、ステップＳ２１０４で、前述したステップＳ２１０２の場合とは逆に、旧ＥＸＥファイル３０５の指定領域を比較側、新ＥＸＥファイル３０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前

述の図２２で示される比較処理が実行される。この比較処理の例として、前述の図２５Ａに対応する動作が図２５Ｂに示される。この比較処理は、前述の図２４Ａの場合に対して、比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図２５Ｂの旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で削除されたバイトデータＤＤ・・・Ｄで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、旧ＥＸＥファイル３０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル３０６の指定領域の各バイトデータに対する図２１のステップＳ２１０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０５で、全てのバイトが不一致であるか否かが判定される。

図２５Ａ、図２５Ｂの例では、全てのバイトが不一致でなくステップＳ２１０５の判定がＮＯとなるため、ステップＳ２１０６で、前述した（１）式、（２）式、及び（３）式に基づいて、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合 RNEWOLD の方が旧ＥＸＥファイル３０５をキーとし

た場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

図２５Ａと図２５Ｂの例では、RNEWOLD=ROLDNEWとなるため、ステップＳ２１０８で、新ＥＸＥファイル３０６をキーとした場合のステップＳ２１０２の比較処理の結果が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル３０６において旧ＥＸＥファイル３０５のバイトデータｃとｄの間のバイトデータＤＤ・・・Ｄが削除されたことを示す差分情報が、更新ファイル３０７へライトされる。

次に、図２６Ａに示されるように、新ＥＸＥファイル３０６の指定領域において、旧ＥＸＥファイル３０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＲＲ・・・Ｒの領域がそれと同じ長さを有する他のバイトデータ列ＲＲ・・・Ｒに置換された場合を考える。

この場合、まず、図２６Ａに示されるように、図２１のステップＳ２１０２で、新ＥＸＥファイル３０６において指定された領域を比較側、旧ＥＸＥファイル３０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、まず、図２６ＡのＴ１～Ｔ９で示される比較範囲では、前述の図２

４ＡのＴ１～Ｔ９で示される比較範囲のものと同様の処理であり、続いて、図２６ＡのＴ１０～Ｔ１３で示される比較範囲では、前述の図２５ＡのＴ４～Ｔ７で示される比較範囲のものと同様の処理である。

そして、この比較処理の結果、図２６Ａの新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で置換されたバイトデータＲＲ・・・Ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、新ＥＸＥファイル３０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル３０５の指定領域の各バイトデータに対する図２１のステップＳ２１０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

一致バイト数が６４バイト以上であってステップＳ２１０３の判定がＹＥＳとなった場合には、ステップＳ２１０９で、上述の不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。

一致バイト数が６４バイトより少なくステップＳ２１０３の判定がＮＯとなった場合には、ステップＳ２１０４で、前述したステップＳ２１０２の場合

とは逆に、旧ＥＸＥファイル３０５の指定領域を比較側、新ＥＸＥファイル３０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２２で示される比較処理が実行される。この比較処理の例として、前述の図２６Ａに対応する動作が図２６Ｂに示される。この比較処理は、まず、図２６ＢのＴ１～Ｔ９で示される比較範囲では、前述した図２５ＢのＴ１～Ｔ９で示される比較範囲のものと同様の処理であって、続いて、図２６ＢのＴ１０～Ｔ１３で示される比較範囲では、前述した図２４ＢのＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２６Ｂの旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で置換されたバイトデータＲＲ・・・Ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、旧ＥＸＥファイル３０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル３０６の指定領域の各バイトデータに対する図２１のステップＳ２１０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０５で、全てのバイトが不一致であるか

否かが判定される。

図２６Ａ、図２６Ｂの例では、全てのバイトが不一致でなくステップＳ２１０５の判定がＮＯとなるため、ステップＳ２１０６で、前述した（１）式、（２）式、及び（３）式に基づいて、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

図２６Ａと図２６Ｂの例では、RNEWOLD=ROLDNEWとなるため、ステップＳ２１０８で、新ＥＸＥファイル３０６をキーとした場合のステップＳ２１０２の比較処理の結果が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル３０６において旧ＥＸＥファイル３０５のバイトデータｃとｄの間のバイトデータＲＲ・・・Ｒが置換されたことを示す差分情報が、更新ファイル３０７へライトされる。

次に、図２７Ａに示されるように、新ＥＸＥファイル３０６の指定領域において、旧ＥＸＥファイル３０５の指定領域のバイトデータｃとｄの間に存在するバイトデータ列のうち、バイトデータ列Ｒ・・・Ｒ及びｒ・・・ｒの領域がそれと同じ長さを有する他のバイトデータ列Ｒ・・・Ｒ及びｒ・・・ｒに

置換され、バイトデータ列ＤＤ・・・Ｄが削除され、また、旧ＥＸＥファイル３０５の指定領域のバイトデータｃとｄの間に、そこに存在していなかった新たなバイトデータ列ＩＩ・・・Ｉが挿入された場合を考える。

この場合、まず、図２７Ａに示されるように、図２１のステップＳ２１０２で、新ＥＸＥファイル３０６において指定された領域を比較側、旧ＥＸＥファイル３０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図２６Ａの場合と同様であり、従って、図２７ＡのＴ１～Ｔ１０で示される比較範囲では、前述の図２４ＡのＴ１～Ｔ９で示される比較範囲のものと同様の処理であり、続いて、図２７ＡのＴ１１～Ｔ１３で示される比較範囲では、前述の図２５ＡのＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２７Ａの新ＥＸＥファイル３０６と旧ＥＸＥファイル３０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で挿入、削除、及び置換されたバイトデータＩ・・・ＩＲ・・・Ｒｒ・・・ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、新ＥＸＥファイル３０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル３０５の指定領域の各バイトデータに対する図２１のステップＳ２１０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

一致バイト数が６４バイト以上であってステップＳ２１０３の判定がＹＥＳとなった場合には、ステップＳ２１０９で、上述の不一致部分に関する差分情報が図３の更新ファイル３０７へライトされる。

一致バイト数が６４バイトより少なくステップＳ２１０３の判定がＮＯとなった場合には、ステップＳ２１０４で、前述したステップＳ２１０２の場合とは逆に、旧ＥＸＥファイル３０５の指定領域を比較側、新ＥＸＥファイル３０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２２で示される比較処理が実行される。この比較処理の例として、前述の図２７Ａに対応する動作が図２７Ｂに示される。この比較処理は、前述の図２６Ｂの場合と同様であり、まず、図２７ＢのＴ１～Ｔ９で示される比較範囲では、前述した図２５ＢのＴ１～Ｔ９で示される比較範囲のものと同様の処理であって、続いて、図２７ＢのＴ１０～Ｔ１２で示される比較範囲では、前述した図２４ＢのＴ４

～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２７Ｂの旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル３０６で挿入、削除、及び置換されたバイトデータＩ・・・ＩＲ・・・Ｒｒ・・・ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、旧ＥＸＥファイル３０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル３０６の指定領域の各バイトデータに対する図２１のステップＳ２１０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０５で、全てのバイトが不一致であるか否かが判定される。

図２７Ａ、図２７Ｂの例では、全てのバイトが不一致でなくステップＳ２１０５の判定がＮＯとなるため、ステップＳ２１０６で、前述した（１）式、（２）式、及び（３）式に基づいて、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

図２７Ａと図２７Ｂの例では、RNEWOLD=ROLDNEW

となるため、ステップＳ２１０８で、新ＥＸＥファイル３０６をキーとした場合のステップＳ２１０２の比較処理の結果が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報が更新ファイル３０７へライトされる。

ここで、図２７Ａにおいて、旧ＥＸＥファイル３０５の指定領域におけるバイトデータｃとｄの間のバイト数の方が、新ＥＸＥファイル３０６の指定領域におけるバイトデータｃとｄの間のバイト数より多ければ、まず、新ＥＸＥファイル３０６における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル３０７へライトされ、続いて、｛（旧ＥＸＥファイル３０５における当該バイト数）－（新ＥＸＥファイル３０６における当該バイト数）｝分のバイトデータが削除された旨の差分情報が更新ファイル３０７へライトされる。

逆に、図２７Ａにおいて、旧ＥＸＥファイル３０５の指定領域におけるバイトデータｃとｄの間のバイト数の方が、新ＥＸＥファイル３０６の指定領域におけるバイトデータｃとｄの間のバイト数より少なければ、まず、旧ＥＸＥファイル３０５における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル３０７へライトされ、続いて、｛（新ＥＸＥファイル３０６における当該バイト数）

－（旧ＥＸＥファイル３０５における当該バイト数）｝分のバイトデータが挿入された旨の差分情報が更新ファイル３０７へライトされる。

また、図２７Ａにおいて、旧ＥＸＥファイル３０５の指定領域におけるバイトデータｃとｄの間のバイト数が、新ＥＸＥファイル３０６の指定領域におけるバイトデータｃとｄの間のバイト数と等しければ、旧ＥＸＥファイル３０５又は新ＥＸＥファイル３０６における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル３０７へライトされる。

最後に、図２８Ａ、図２８Ｂに示されるような特殊な例を考える。この例では、新ＥＸＥファイル３０６の指定領域において、旧ＥＸＥファイル３０５の指定領域の先頭のバイトデータａの手前にバイトデータ列ＩＩ・・・Ｉが挿入され、旧ＥＸＥファイル３０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＤＤ・・・Ｄが削除された場合であって、挿入されたバイトデータ列ＩＩ・・・Ｉと削除されたバイトデータ列ＤＤ・・・Ｄが偶然に一致しているような場合が示されている。

この場合、まず、図２８Ａに示されるように、図２１のステップＳ２１０２で、新ＥＸＥファイル３０６において指定された領域を比較側、旧ＥＸＥファイル３０５において指定された領域を比較対象側

として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図２８ＡのＴ１～Ｔ３で示される比較範囲では、前述の図２５ＡのＴ４～Ｔ６で示される比較範囲のものと同様の処理であり、続いて、図２８ＡのＴ４～Ｔ６で示される比較範囲では、前述の図２４ＡのＴ１～Ｔ３で示される比較範囲のものと同様の処理である。なお、図２８ＡのＴ３→Ｔ４の変化時には、最初に新ＥＸＥファイル３０６上の挿入された第１番目のバイトデータＩと旧ＥＸＥファイル３０５上の各バイトデータａｂ・・・ｃとが不一致となった後に、新ＥＸＥファイル３０６上の挿入された第１番目のバイトデータＩと旧ＥＸＥファイル３０５上の新ＥＸＥファイル３０６において削除された第１番目のバイトデータＤとが一致した状態となるため、ステップＳ２２０５の判定はＮＯとなる。

そして、図２８ＡのＴ６で示される処理の後、ステップＳ２２０６で、変数src に新ＥＸＥファイル３０６上のバイトデータａのアドレスが代入され、変数distに旧ＥＸＥファイル３０５上のバイトデータｄのアドレスが代入されて、ステップＳ２２０４が実行された時点で、変数src で示されるアドレスにおける新ＥＸＥファイル３０６のバイトデータａと変数distで示されるアドレスにおける旧ＥＸＥファイル３０５のバイトデータｄとが不一致となり、

その判定がＮＯとなる。この不一致状態は、最初に新ＥＸＥファイル３０６上の挿入された各バイトデータＩＩ・・・Ｉと旧ＥＸＥファイル３０５上の各バイトデータａｂ・・・ｃとが不一致となり、更に、新ＥＸＥファイル３０６上の挿入された各バイトデータＩＩ・・・Ｉと旧ＥＸＥファイル３０５上の新ＥＸＥファイル３０６において削除された各バイトデータＤＤ・・・Ｄとが一致した後の不一致状態であるため、ステップＳ２２０７の判定はＹＥＳとなって、ステップＳ２２１１が実行される。ステップＳ２２１１では、図２８Ａの旧ＥＸＥファイル３０５上のバイトデータａｂ・・・ｃで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、新ＥＸＥファイル３０６上の挿入されたバイトデータＩＩ・・・Ｉと旧ＥＸＥファイル３０５上の新ＥＸＥファイル３０６で削除されたバイトデータＤＤ・・・Ｄとで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、新ＥＸＥファイル３０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル３０５の指定領域の各バイトデータに対する図２１のステップＳ２１０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

今、図２８Ａにおいて、一致部分であるバイトデータ列ＩＩ・・・Ｉ（＝ＤＤ・・・Ｄ）のバイト数が例えば数バイト程度と少なく、不一致部分であるバイトデータ列ａｂ・・・ｃのバイト数が多い場合には、一致部分が短く不一致部分が長い結果となる。

このような場合には、ステップＳ２１０３の判定がＮＯとなる結果、ステップＳ２１０４で、前述したステップＳ２１０２の場合とは逆に、旧ＥＸＥファイル３０５の指定領域を比較側、新ＥＸＥファイル３０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２２で示される比較処理が実行される。この比較処理の例として、前述の図２８Ａに対応する動作が図２８Ｂに示される。この比較処理は、図２８ＢのＴ１～Ｔ３で示される比較範囲では、前述の図２４ＢのＴ４～Ｔ６で示される比較範囲のものと同様の処理であり、続いて、図２８ＢのＴ４～Ｔ６で示される比較範囲では、前述の図２４ＢのＴ１～Ｔ３で示される比較範囲のものと同様の処理である。

そして、図２８ＢのＴ６で示される処理の後、ステップＳ２２０６で、変数src に旧ＥＸＥファイル３０５上のバイトデータＤのアドレスが代入され、変数distに新ＥＸＥファイル３０６上のバイトデータｄのアドレスが代入されて、ステップＳ２２０４が実行された時点で、変数src で示されるアドレス

における旧ＥＸＥファイル３０５のバイトデータＤと変数distで示されるアドレスにおける新ＥＸＥファイル３０６のバイトデータｄとが不一致となり、その判定がＮＯとなる。この不一致状態は、最初に旧ＥＸＥファイル３０５のバイトデータａと新ＥＸＥファイル３０６上の挿入された各バイトデータＩＩ・・・Ｉとが不一致となり、更に、旧ＥＸＥファイル３０５上の各バイトデータａｂ・・・ｃと新ＥＸＥファイル３０６上の各バイトデータａｂ・・・ｃとが一致した後の不一致状態であるため、ステップＳ２２０７の判定はＹＥＳとなって、ステップＳ２２１１が実行される。ステップＳ２２１１では、図２８Ｂの新ＥＸＥファイル３０６上のバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、旧ＥＸＥファイル３０５と新ＥＸＥファイル３０６上のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

以上の動作の結果、旧ＥＸＥファイル３０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル３０６の指定領域の各バイトデータに対する図２１のステップＳ２１０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ２１０５で、全てのバイトが不一致であるか

否かが判定される。

図２８Ａ、図２８Ｂの例では、全てのバイトが不一致でなくステップＳ２１０５の判定がＮＯとなるため、ステップＳ２１０６で、前述した（１）式、（２）式、及び（３）式に基づいて、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

今、前述したように、図２８Ｂにおいて、一致部分であるバイトデータ列ａｂ・・・ｃのバイト数が多く、不一致部分であるバイトデータ列ＩＩ・・・Ｉのバイト数が例えば数バイト程度と少ない場合には、一致部分が長く不一致部分が短い結果となる。

従って、図２８Ａと図２８Ｂの例では、RNEWOLD ＜ROLDNEW となる。前述したように、更新ファイル３０７には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、図２８Ａ及び図２８Ｂの場合のように、新ＥＸＥファイル３０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル３０５をキーとした場合の一致バイト数の割合ROLDNEW より小さくステップＳ２１０６の判定がＹＥＳの場合には、ステップＳ２１０７で、旧ＥＸＥファイル３０５をキーとした場合のステップＳ２１０４の比較処理の結果

が選択され、ステップＳ２１０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル３０６において旧ＥＸＥファイル３０５のバイトデータａで示されるアドレスの手前にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が、更新ファイル３０７へライトされる。

以上説明したようにして、差分抽出処理部３０４において差分抽出処理が実行される。

図２９Ａ、図２９Ｂに、更新ファイル３０７のデータ構造を示す。

更新ファイル３０７の各レコードは、図２９Ａに示されるように、そのレコードの属性を示すデータRdcordIDと、旧ＥＸＥファイル３０５に対するそのレコードの処理位置を示すデータDataOffset、更新される実データの長さを示すデータDataLength及び更新される実データ配列Data[n] とから構成される。

RdcordIDの属性は、図２９Ｂに示される。

まず、値が０のRdcordIDは、そのレコードが置換レコードであることを示し、この場合には、旧ＥＸＥファイル３０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが、実データ配列Data[n] で示されるデータに置換される。

また、値が１のRdcordIDは、そのレコードが挿入

レコードであることを示し、この場合には、旧ＥＸＥファイル３０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さの実データ配列Data[n] で示されるデータが挿入される。

更に、値が２のRecordIDは、そのレコードが削除レコードであることを示し、この場合には、旧ＥＸＥファイル３０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが削除される。なお、この場合には、実データ配列Data[n] は付加されていない。

このように生成された更新ファイル３０７に基づいて旧ＥＸＥファイル３０５が更新される場合の動作は、更新ファイル３０７から１つ１つレコードが取り出され、各レコードに対して上述の各レコード属性毎の処理が実行される動作として、簡単に実現される。

以上説明した好適実施例は、本発明を[*]システムのもとで実行可能なＥＸＥファイルの差分抽出処理に適用したものであって、DOS EXE 形式の領域、[*] EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。しかし、本発明はこれに限られるものではなく、他の論理単位、例えばページデータ単位などで対応付けが行われた上で差分抽出処理が実行

According PCT Article 21(6) this part should be omitted.(*)
page 71
line 17, 20

されるように構成することもできる。

図３０は、 ＊ オペレーティングシステムのもとで実行可能なＥＸＥファイルの形式の一例を示した図である。

このようなＥＸＥファイルについて差分が抽出される場合にも、上述した好適実施例と基本的に同様の本発明に基づく手法を適用することができる。

図３０に示される構成を有するＥＸＥファイルは、前述した図４のＥＸＥファイルの場合と同様に、コンピュータのための従来からあるシングルタスクオペレーティングシステムの１つである ＊ システムのもとでも実行可能なように構築されている。そのため、このＥＸＥファイルは、図３０に示されるように、 ＊ EXE形式の領域と ＊ EXE 形式の領域とから構成されている。

まず、ファイルの前半部分には、 ＊ EXE形式の領域が存在する。この ＊ EXE形式の領域は、 ＊ 用プログラムを実行するための各種情報及び後述する ＊ EXE ヘッダへのポインタ情報等が格納される ＊ EXEヘッダ領域と、 ＊ システムのもとで動作するプログラムの本体が格納される特には図示しない ＊ ロードモジュール領域とから構成される。

ファイルの後半部分には、 ＊ EXE 形式の領域が存在する。この ＊ EXE 形式の領域は、

According PCT Article 21(6) this part should be omitted. (＊)
page 72
line 2, 11, 14, 16, 17-19, 20, 22, 24, 25

1 ✳ 用プログラムを実行するための各種情報等が格納される ✳ EXE ヘッダ領域、後述するセクション領域を参照するためのセクションディレクトリ領域、このセクションディレクトリ領域によって参照され ✳ システムのもとで動作するプログラムの本体が格納されるセクション領域などから構成される。

上述のような形式を有するＥＸＥファイルに対して差分抽出処理が実行される場合にも、前述した好適実施例の場合と同様に、 ✳ EXEヘッダ領域、 ✳ EXE ヘッダ領域、セクションディレクトリ領域、及びセクション領域などのそれぞれの論理的な領域毎に、旧ＥＸＥファイルと新ＥＸＥファイルとで差分抽出処理を実行することができる。

この場合、例えば ✳ EXEヘッダ領域内の各種情報を判定することにより、前述した好適実施例における図５のステップＳ５０１～Ｓ５０４と同じようなチェックを行うことができる。

次に、 ✳ EXE ヘッダ領域の先頭アドレスは、 ✳ EXEヘッダ領域に設定されているオフセットアドレスにより取得でき、 ✳ EXE ヘッダ領域の範囲は、その領域内に設定されているヘッダサイズを参照することにより取得できる。

また、上述の ✳ EXE ヘッダ領域に続く領域としてセクションディレクトリ領域の先頭アドレ

According PCT Article 21(6) this part should be omitted. (✳)
page 73
line 1, 2, 5, 10, 11, 15, 19-21, 24

スを取得でき、 ✳ EXE ヘッダ領域内に設定されているセクション数を参照することにより、セクションディレクトリ領域の範囲を取得できる。

更に、セクションディレクトリ領域の情報を使って、複数のセクション領域のそれぞれの位置を取得できる。

以上のようにして取得された各領域毎に、前述した好適実施例における図３に示される差分抽出処理部３０４と同様のモジュールを起動することによって、旧ＥＸＥファイルと新ＥＸＥファイルとで差分抽出処理を実行することができる。

そのほか、上述のファイルに各種テーブルやリソースが追加されるような場合においても、前述した好適実施例の場合と同様の処理によって、各論理単位間の対応付けを行った上で、旧ＥＸＥファイルと新ＥＸＥファイルとで差分抽出処理を実行することができる。

産業上の利用可能性

以上説明したように、本発明によれば、差分抽出処理が実行される場合に、論理単位毎に、対応付けが行われた上で差分抽出処理が実行されるため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズを大幅に削減すること

According PCT Article 21(6) this part should be omitted. (✳)
page 74
line 1

が可能となる。

これにより、例えばネットワークを介して更新ファイルが配信される場合などで、新ＥＸＥファイルがそのまま配信される場合に比較して回線の使用効率を大幅に向上させることが可能となる。

請求の範囲

1．旧実行ファイル（３０５）を新実行ファイル（３０６）に更新するための更新ファイル（３０７）を、新実行ファイル（３０６）の旧実行ファイル（３０５）に対する差分として抽出する実行ファイル差分抽出装置であって、

前記旧実行ファイル（３０５）及び前記新実行ファイル（３０６）のそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段（３０１）と、

該領域分割手段（３０１）によって分割された前記旧実行ファイル（３０５）の複数の領域と前記新実行ファイル（３０６）の複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段（３０２、３０３）と、

該領域対応付け手段（３０２、３０３）によって対応付けされた前記旧実行ファイル（３０５）の領域と前記新実行ファイル（３０６）の領域との間で、前記旧実行ファイル（３０５）の領域のデータに対する前記新実行ファイル（３０６）の領域のデータの差分情報を抽出する差分抽出処理手段（３０４）と、

該差分抽出処理手段（３０４）において抽出された差分情報を前記更新ファイル（３０７）に格納すると共に、前記領域対応付け手段（３０２、３０３）

において対応付けが行われなかった前記旧実行ファイル（３０５）の領域又は前記新実行ファイル（３０６）の領域に対応する差分情報を抽出しそれを前記更新ファイル（３０７）に格納する更新ファイル生成手段（３０４、３０２、３０３）と、

を含むことを特徴とする実行ファイル差分抽出装置。

２．クレーム１に記載の装置であって、

前記論理単位は、前記旧実行ファイル（３０５）又は前記新実行ファイル（３０６）のデータ属性の単位であるセグメント、リソース、ヘッダ、又はテーブルである、

ことを特徴とする実行ファイル差分抽出装置。

３．クレーム１に記載の装置であって、

前記論理単位は、前記旧実行ファイル（３０５）又は前記新実行ファイル（３０６）のデータ属性の単位であるセクション、セクションディレクトリ、又はヘッダである、

ことを特徴とする実行ファイル差分抽出装置。

４．クレーム１に記載の装置であって、

前記差分情報は、前記新実行ファイル（３０６）において前記旧実行ファイル（３０５）に対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される、

ことを特徴とする実行ファイル差分抽出装置。

5．クレーム1に記載の旧実行ファイル（305）及び更新ファイル（307）とから新実行ファイル（306）を生成する実行ファイル更新装置であって、

前記更新ファイル（307）から前記差分情報を順次取り出し、該差分情報と前記旧実行ファイル（305）とから、前記新実行ファイル（306）を生成するファイル更新手段を含む、

ことを特徴とする実行ファイル更新装置。

6．新実行ファイル（306）の旧実行ファイル（305）に対する差分が抽出されて更新ファイル（307）とされ、旧実行ファイル（305）と更新ファイル（307）とから新実行ファイル（306）が生成される実行ファイル差分抽出／更新装置において、

前記旧実行ファイル（305）及び前記新実行ファイル（306）のそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段（301）と、

該領域分割手段（301）によって分割された前記旧実行ファイル（305）の複数の領域と前記新実行ファイル（306）の複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段（302、303）と、

該領域対応付け手段（302、303）によって

対応付けされた前記旧実行ファイル（３０５）の領域と前記新実行ファイル（３０６）の領域との間で、前記旧実行ファイル（３０５）の領域のデータに対する前記新実行ファイル（３０６）の領域のデータの差分情報を抽出する差分抽出処理手段（３０４）と、

該差分抽出処理手段（３０４）において抽出された差分情報を前記更新ファイル（３０７）に格納すると共に、前記領域対応付け手段（３０２、３０３）において対応付けが行われなかった前記旧実行ファイル（３０５）の領域又は前記新実行ファイル（３０６）の領域に対応する差分情報を抽出しそれを前記更新ファイル（３０７）に格納する更新ファイル生成手段（３０４、３０２、３０３）と、

前記更新ファイル（３０７）から前記差分情報を順次取り出し、該差分情報と前記旧実行ファイル（３０５）とから、前記新実行ファイル（３０６）を生成するファイル更新手段と、

を含むことを特徴とする実行ファイル差分抽出／更新装置。

７．旧実行ファイルを新実行ファイルに更新するための更新ファイルを、新実行ファイルの旧実行ファイルに対する差分として抽出する実行ファイル差分抽出方法であって、

前記旧実行ファイル及び前記新実行ファイルのそ

れぞれを、所定の論理単位毎に複数の領域に分割し、

該領域の分割によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けし、

該領域の対応付けによって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出し、

該抽出された差分情報を前記更新ファイルに格納すると共に、前記領域の対応付けにおいて対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する、

ことを特徴とする実行ファイル差分抽出方法。

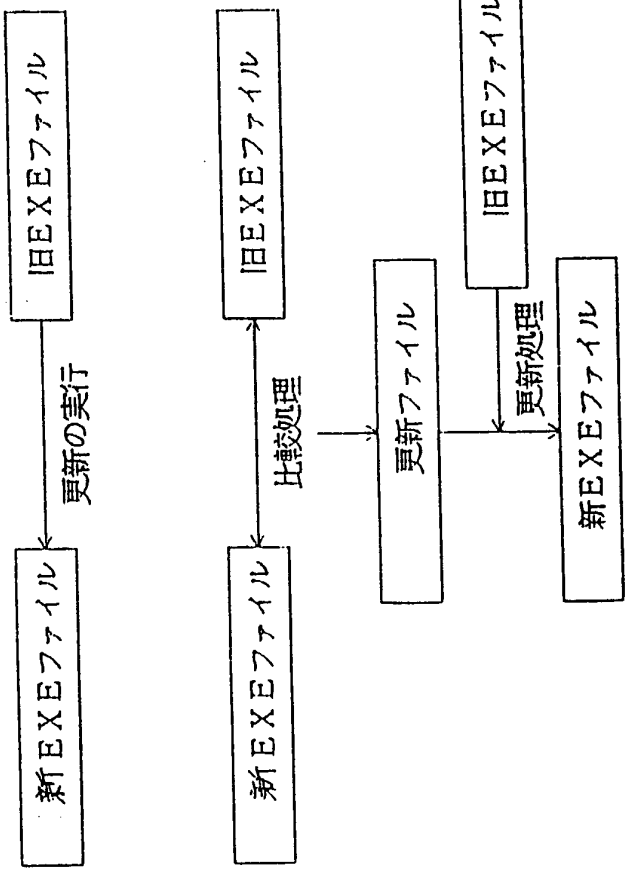

旧EXEファイル
更新の実行
新EXEファイル
旧EXEファイル
比較処理
新EXEファイル
更新ファイル
旧EXEファイル
更新処理
新EXEファイル

図1A

図1B

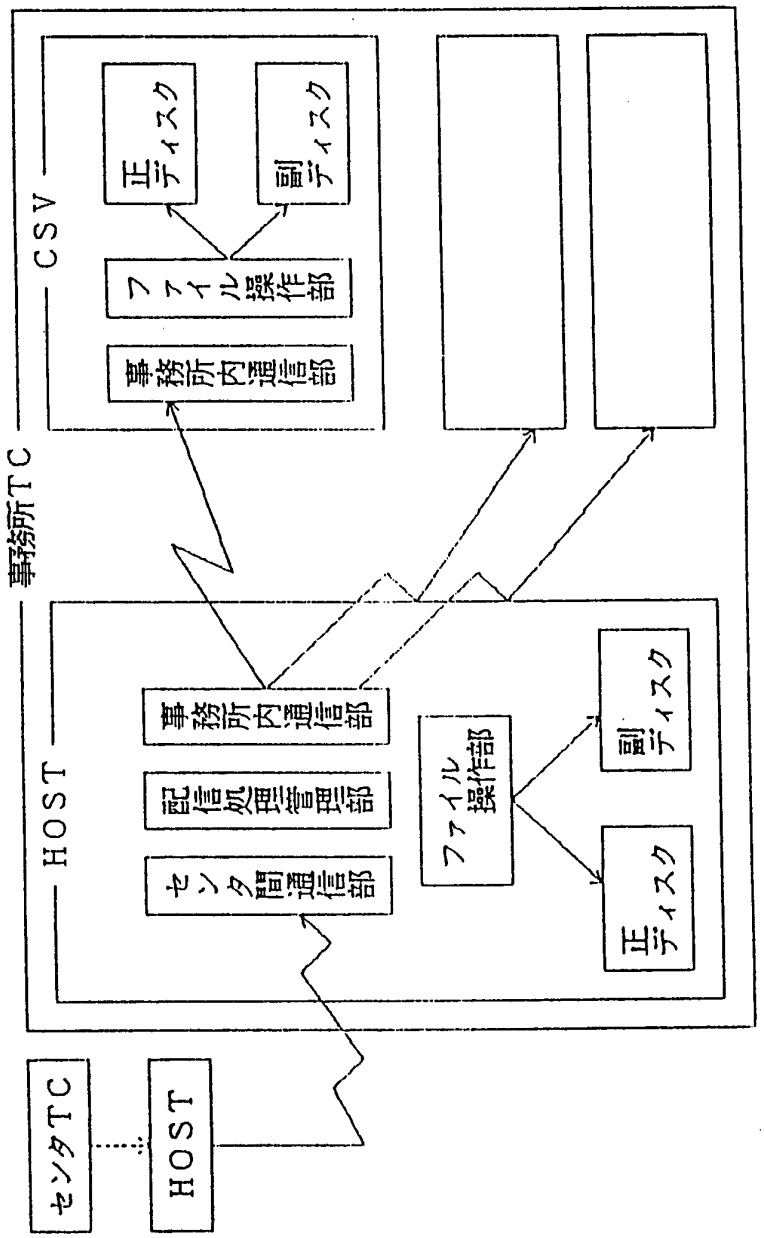

事務所TC
CSV
正ディスク
副ディスク
ファイル操作部
事務所内通信部
HOST
事務所内通信部
記憶処理管理部
センタ間通信部
ファイル操作部
副ディスク
正ディスク
センタTC
HOST

図2

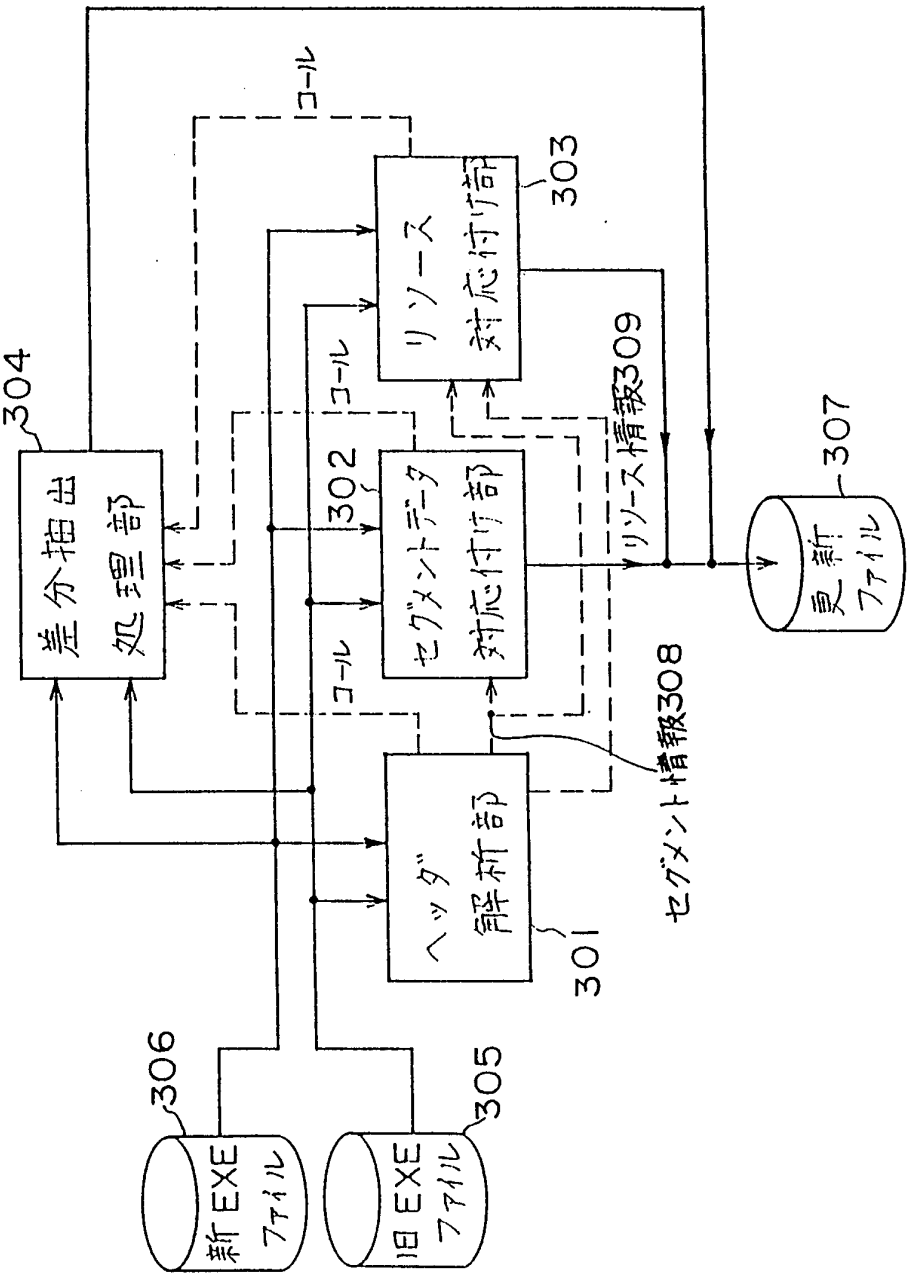

差分抽出処理部
304
ヘッダ解析部
301
セグメントデータ対応付け部
302
リソース対応付け部
303
コール
コール
コール
リソース情報309
セグメント情報308
更新ファイル
307
新EXEファイル
306
旧EXEファイル
305

図3

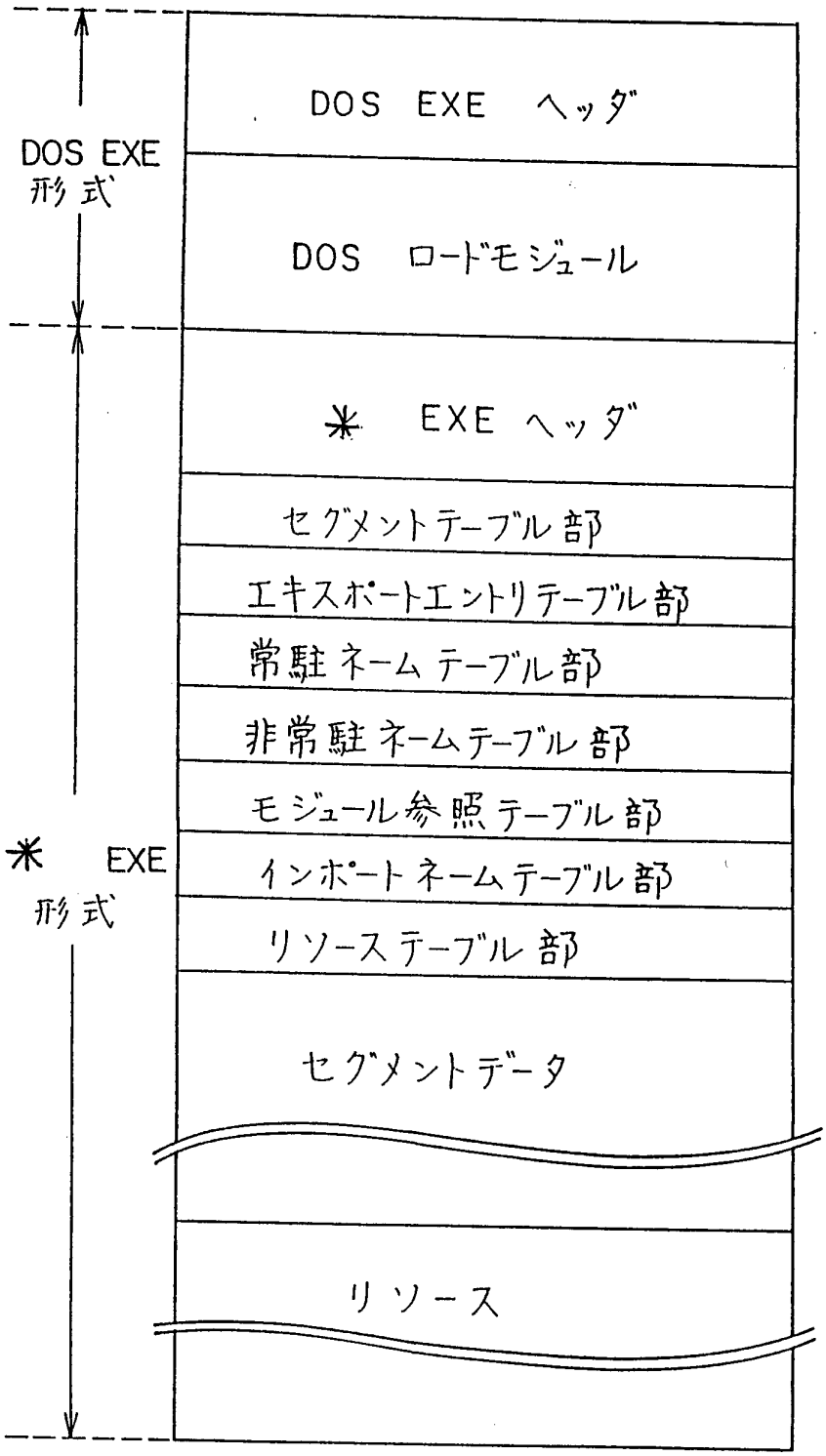


図4

According PCT Article 21(6) this part should be omitted. (✳)

Fig 4

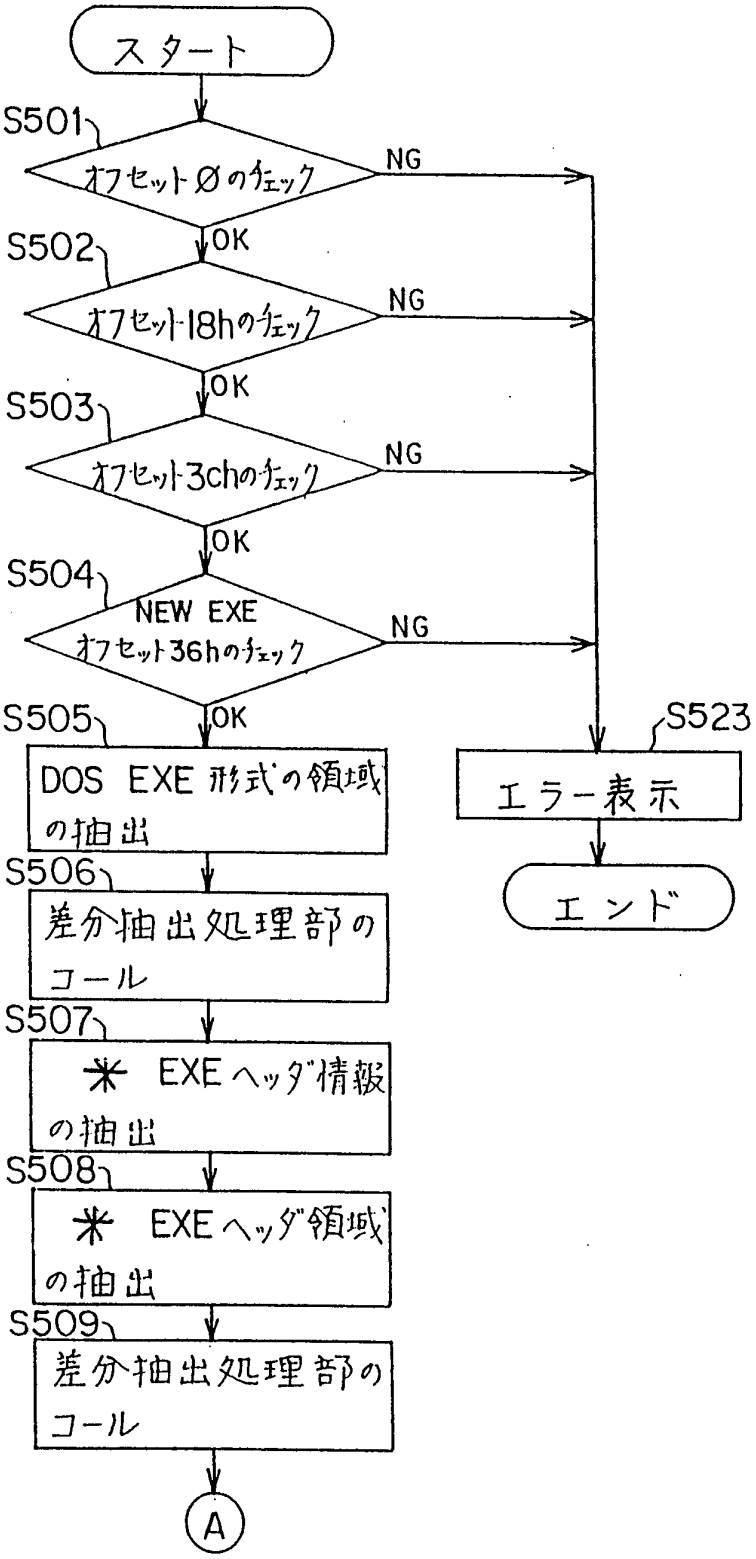


図 5

According PCT Article 21(6) this part should be omitted. (✳)
page Fig 5
line

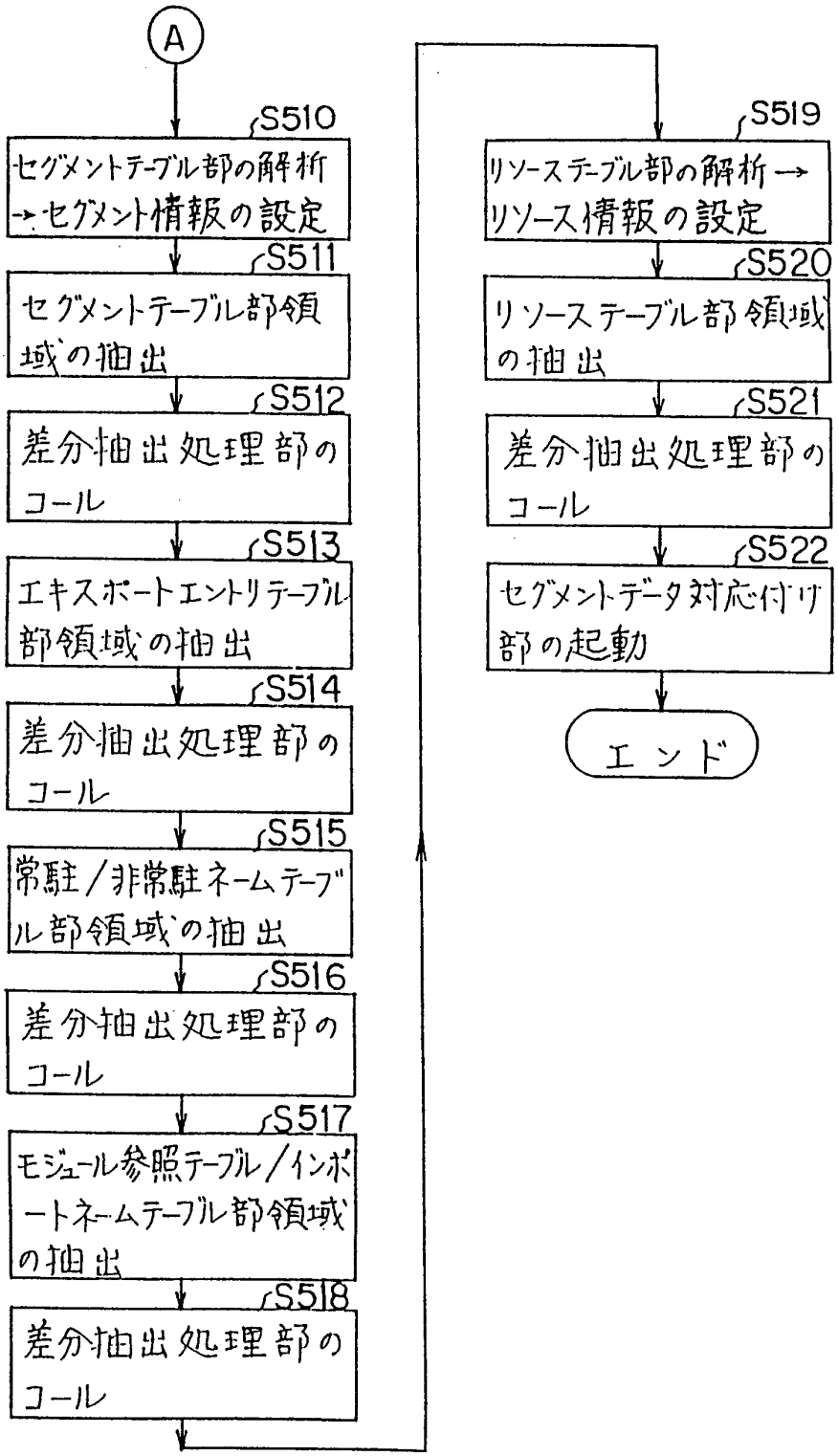

A
S510
セグメントテーブル部の解析→セグメント情報の設定
S511
セグメントテーブル部領域の抽出
S512
差分抽出処理部のコール
S513
エキスポートエントリテーブル部領域の抽出
S514
差分抽出処理部のコール
S515
常駐/非常駐ネームテーブル部領域の抽出
S516
差分抽出処理部のコール
S517
モジュール参照テーブル/インポートネームテーブル部領域の抽出
S518
差分抽出処理部のコール
S519
リソーステーブル部の解析→リソース情報の設定
S520
リソーステーブル部領域の抽出
S521
差分抽出処理部のコール
S522
セグメントデータ対応付け部の起動
エンド

図 6

サイズ	オフセット	内　　容
1ワード	0h	識別子　'MZ'
		：
1ワード	18h	※　EXE　識別子=0x40
		：
1ワード	3Ch	※　EXEへのオフセット

DOS EXE ヘッダの一部

	サイズ	オフセット	内　　容
	1ワード	0h	識別子　'NE'
	1バイト	2h	リンカのバージョン番号
	1バイト	3h	リンカのリビジョン番号
●	1ワード	4h	エキスポートエントリテーブルの02／2　EXE相対オフセット
●	1ワード	6h	エキスポートエントリテーブルバイトサイズ
	2ワード	8h	CRC−32
	1ワード	Ch	フラグ値 8xxxh → DLL 00xxh → Not Specified 01xxh → Not Window Compat（フルスクリーン専用） 02xxh → Window Compat（PMセッションで動作可） 03xxh → Window API（PMのAT）
	1ワード	Eh	自動データセグメントのセグメント番号
	1ワード	10h	ローカルヒープサイズ
	1ワード	12h	スタックサイズ
	2ワード	14h	CS：IPアドレス
	2ワード	18h	SS：SPアドレス
●	1ワード	1Ch	セグメントテーブルの項目数
	1ワード	1Eh	モジュール参照テーブルのエントリ数
●	1ワード	20h	非常駐ネームテーブルのバイトサイズ
●	1ワード	22h	セグメントテーブルの　※　EXE相対オフセット
●	1ワード	24h	リソーステーブルの　※　EXE相対オフセット
●	1ワード	26h	常駐ネームテーブルの　※　EXE相対オフセット
●	1ワード	28h	モジュール参照テーブルの　※　EXE相対オフセット
●	1ワード	2Ah	インポートネームテーブルの　※　EXE相対オフセット
●	2ワード	2Ch	非常駐ネームテーブルのファイルの先頭からのオフセット
	1ワード	30h	移動可能エントリの数
	1ワード	32h	セグメント境界シフトカウント（2^のx値）
●	1ワード	34h	リソーテーブルエントリ数
	1バイト	36h	OSタイプ（1=　※　）

※ EXE ヘッダの一部

※ ●のついている値は、旧ファイルと新ファイルのものをそれぞれ変数内に保持する。
※ 37h以降は、Windowsのみで使用されているエリアであり、OS／2では使用されない。
従って、単純比較により差分を抽出する。

図7

According PCT Article 21(6) this part should be omitted. (※)

Fig 7

変数	データの内容
entrytbl_off	エキスポートエントリテーブルの＊ EXE相対オフセット
entrytbl_len	エキスポートエントリテーブルバイトサイズ
segtbl_cnt	セグメントテーブルの項目数
segtbl_off	セグメントテーブルの＊ EXE相対オフセット
resnamtbl_off	常駐ネームテーブルの＊ EXE相対オフセット
modref_off	モジュール参照テーブルの＊ EXE相対オフセット
importnam_off	インポートネームテーブルの＊ EXE相対オフセット
nonresnam_len	非常駐ネームテーブルの長さ
nonresnam_off	非常駐ネームテーブルのファイル先頭からのオフセット
resrctbl_off	リソーステーブルの＊ EXE相対オフセット
resrctbl_cnt	リソーステーブルエントリ数

図8

According PCT Article 21(6) this part should be omitted. (＊)

Fig 8

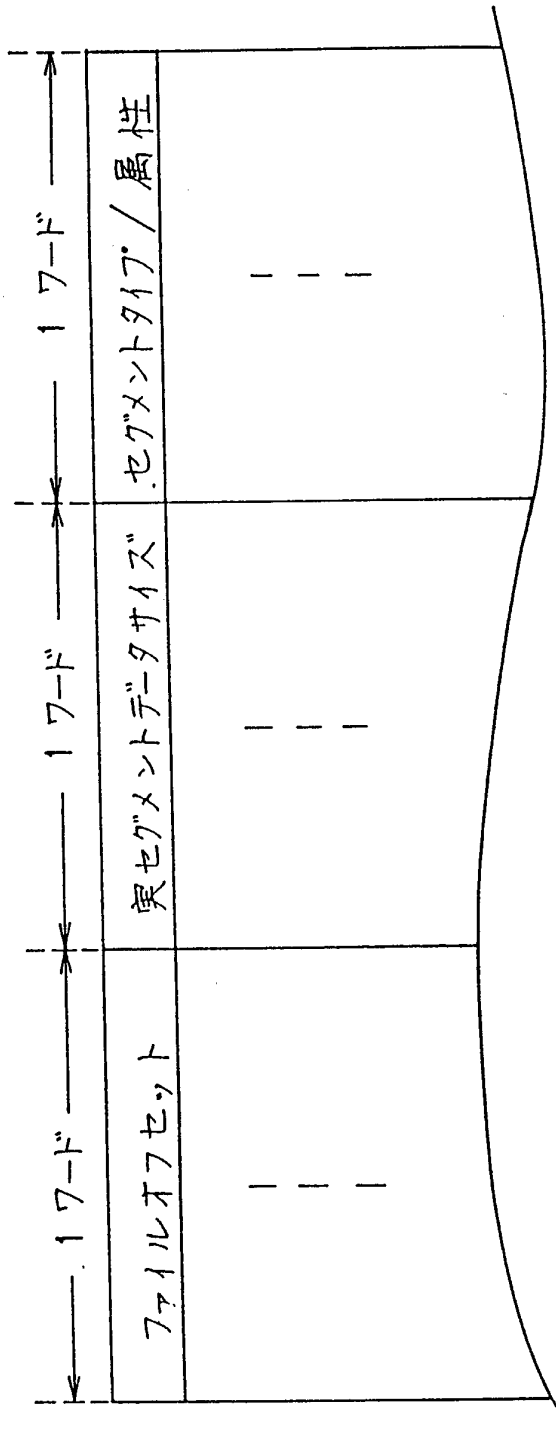


図9

サイズ	オフセット	内容
1バイト	0h	文字列のバイト長。0=テーブルエンド
nバイト	1h	名前列
1ワード	n+1h	エントリテーブルのインデックス

図10

サイズ	オフセット	内容
1バイト	0h	文字列のバイト長。0=テーブルエンド
nバイト	1h	名前列

図11

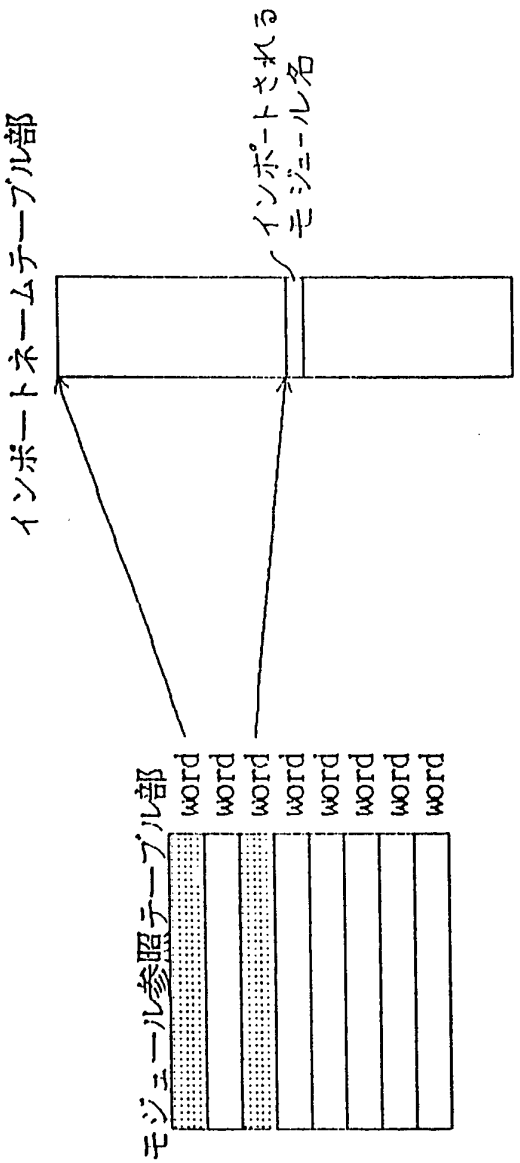

インポートネームテーブル部
インポートされるモジュール名
モジュール参照テーブル部
word
word
word
word
word
word
word
word

図12

サイズ	オフセット	内容
1ワード	0h	リソースのタイプID
1ワード	2h	リソースのネームID

図13

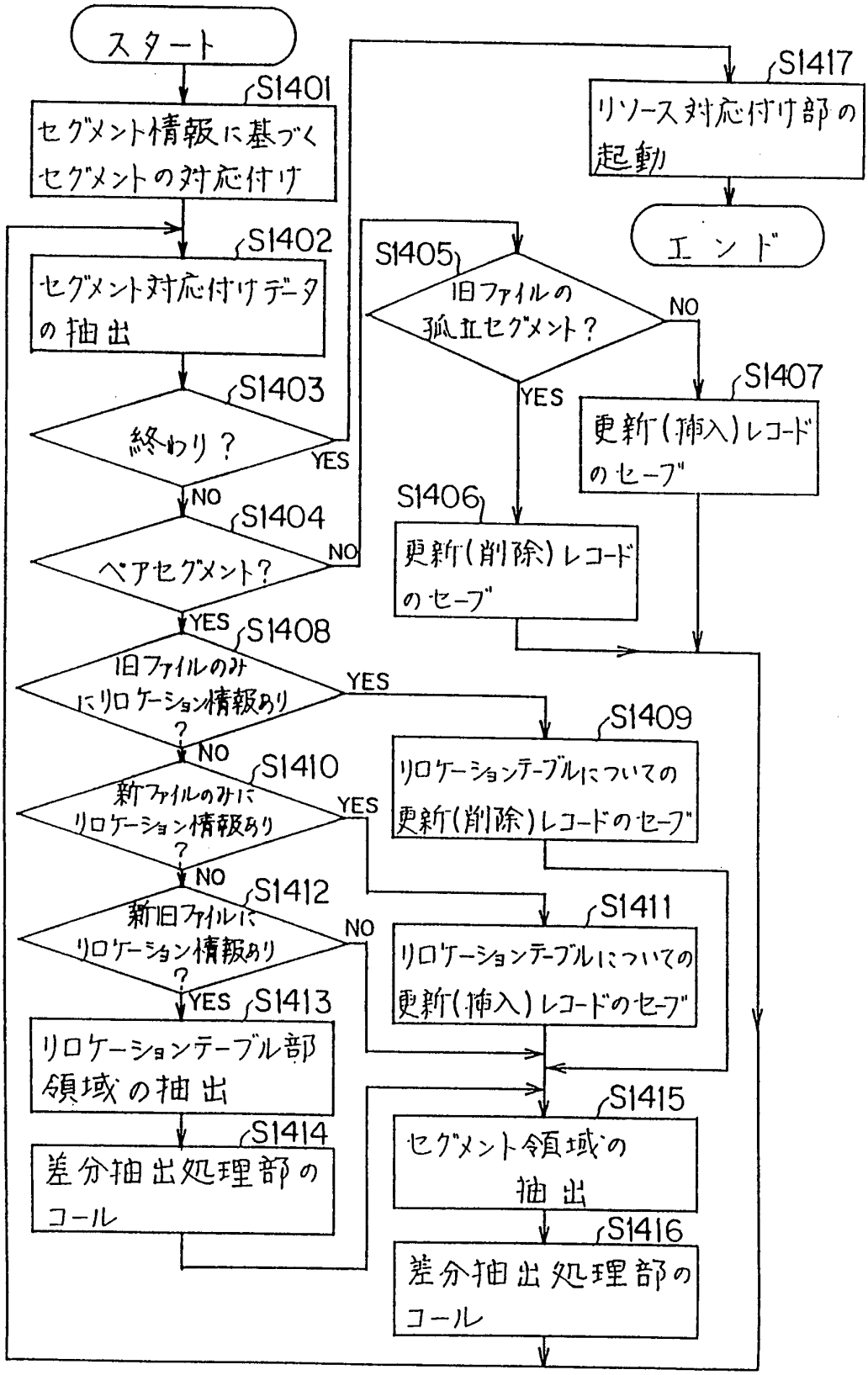

スタート
S1401
セグメント情報に基づくセグメントの対応付け
S1402
セグメント対応付けデータの抽出
S1403
終わり？
YES
NO
S1404
ペアセグメント？
NO
YES
S1408
旧ファイルのみにリロケーション情報あり？
YES
NO
S1409
リロケーションテーブルについての更新(削除)レコードのセーブ
S1410
新ファイルのみにリロケーション情報あり？
YES
NO
S1411
リロケーションテーブルについての更新(挿入)レコードのセーブ
S1412
新旧ファイルにリロケーション情報あり？
NO
YES
S1413
リロケーションテーブル部領域の抽出
S1414
差分抽出処理部のコール
S1415
セグメント領域の抽出
S1416
差分抽出処理部のコール
S1417
リソース対応付け部の起動
エンド
S1405
旧ファイルの孤立セグメント？
NO
YES
S1407
更新(挿入)レコードのセーブ
S1406
更新(削除)レコードのセーブ

図14

図15A

旧EXEファイル

Seg No.	offset	Size	Type
1	1123	8000	CODE, LOADONCALL...
2	2000	3000	DATA, ...
3	3000	4000	CODE, PRELOAD...
4	3500	3000	DATA, ...

図15B

新EXEファイル

Seg No.	offset	Size	Type
1	1123	8000	CODE, LOADONCALL...
2	2000	3000	DATA, ...
3	3000	4000	DATA, ...
4	4000	4000	CODE, PRELOAD...

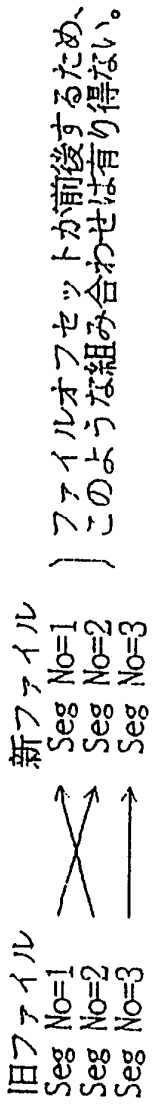

旧ファイル
Seg No=1
Seg No=2
Seg No=3
新ファイル
Seg No=1
Seg No=2
Seg No=3
ファイルオフセットが前後するため、
このような組み合わせは有り得ない。

図16A

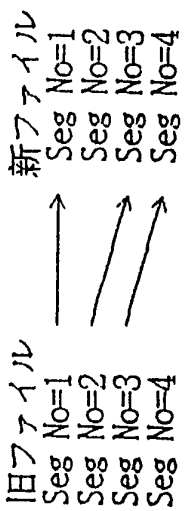

旧ファイル
Seg No=1
Seg No=2
Seg No=3
Seg No=4
新ファイル
Seg No=1
Seg No=2
Seg No=3
Seg No=4

図16B

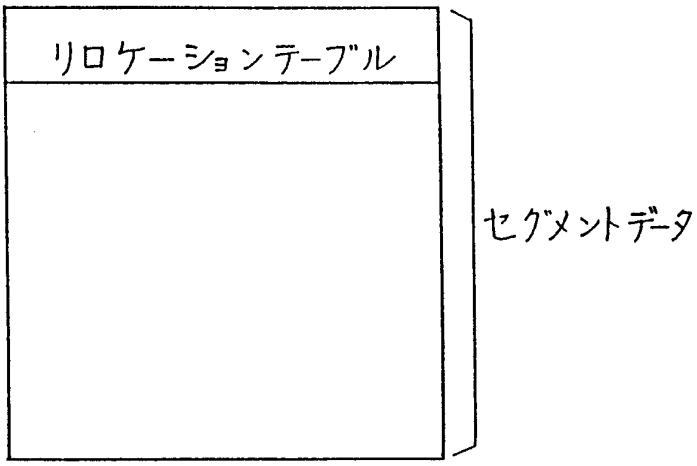


図17

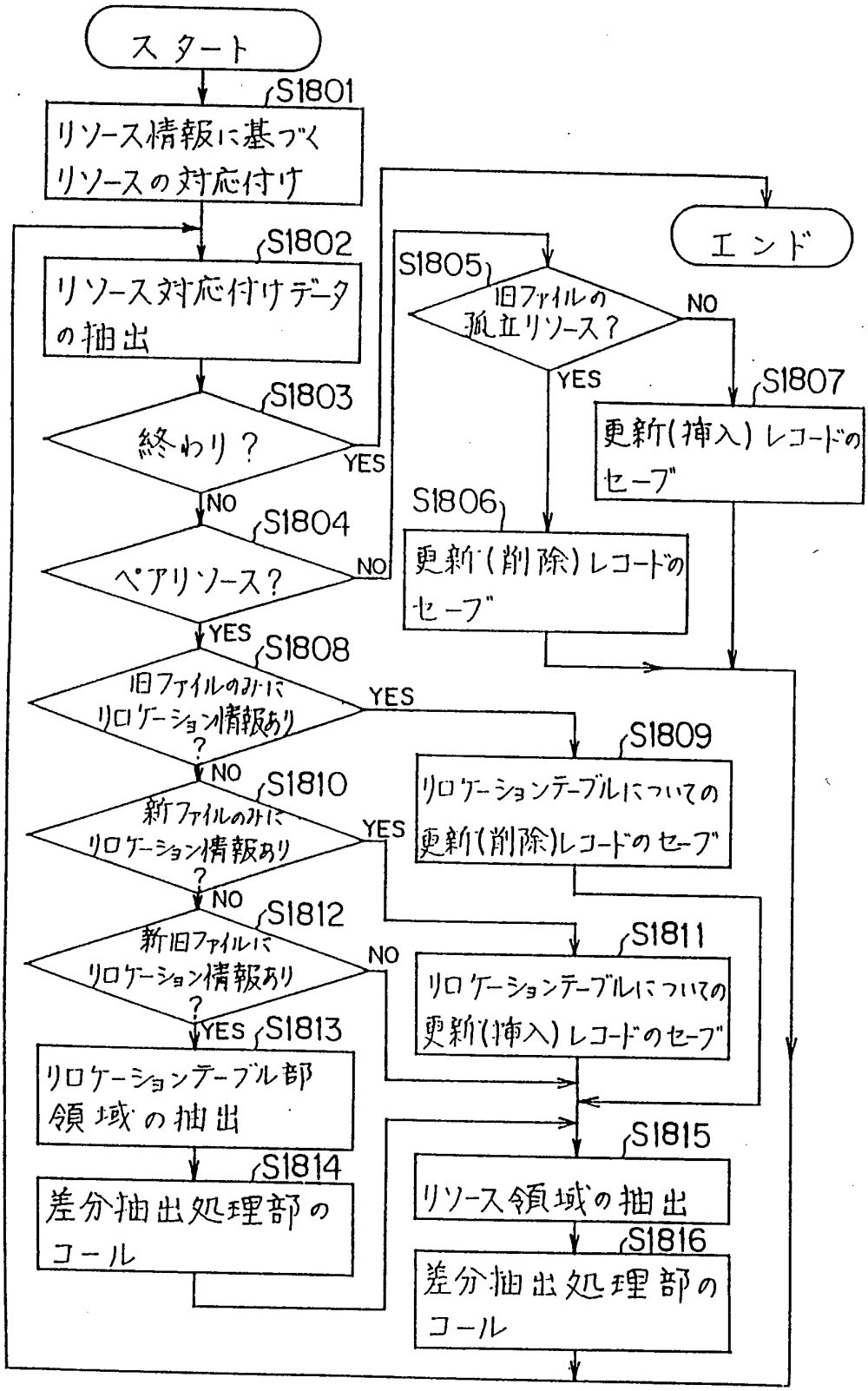

スタート
S1801
リソース情報に基づく
リソースの対応付け
S1802
リソース対応付けデータ
の抽出
S1803
終わり？
YES
NO
S1804
ペアリソース？
NO
YES
S1808
旧ファイルのみに
リロケーション情報あり
？
YES
NO
S1809
リロケーションテーブルについての
更新(削除)レコードのセーブ
S1810
新ファイルのみに
リロケーション情報あり
？
YES
NO
S1811
リロケーションテーブルについての
更新(挿入)レコードのセーブ
S1812
新旧ファイルに
リロケーション情報あり
？
NO
YES
S1813
リロケーションテーブル部
領域の抽出
S1814
差分抽出処理部の
コール
S1815
リソース領域の抽出
S1816
差分抽出処理部の
コール
S1805
旧ファイルの
孤立リソース？
NO
YES
S1807
更新(挿入)レコードの
セーブ
S1806
更新(削除)レコードの
セーブ
エンド

図 18

図19A

ソート前

旧ファイル

ID=4 offset=1000
ID=7 offset=3000
ID=6 offset=4000
ID=5 offset=8000
ID=10 offset=6000
ID=15 offset=5000
ID=30 offset=7000
ID=8 offset=2000

新ファイル

ID=4 offset=1000
ID=7 offset=3000
ID=5 offset=8000
ID=10 offset=6000
ID=15 offset=5000
ID=30 offset=7000
ID=8 offset=2000
ID=38 offset=6500
ID=9 offset=3500

図19B

ソート後

旧ファイル

ID=4 offset=1000
ID=8 offset=2000
ID=7 offset=3000
ID=6 offset=4000
ID=15 offset=5000
ID=10 offset=6000
ID=30 offset=7000
ID=5 offset=8000

新ファイル

ID=4 offset=1000
ID=8 offset=2000
ID=7 offset=3000
ID=9 offset=3500
ID=15 offset=5000
ID=10 offset=6000
ID=38 offset=6500
ID=30 offset=7000
ID=5 offset=8000

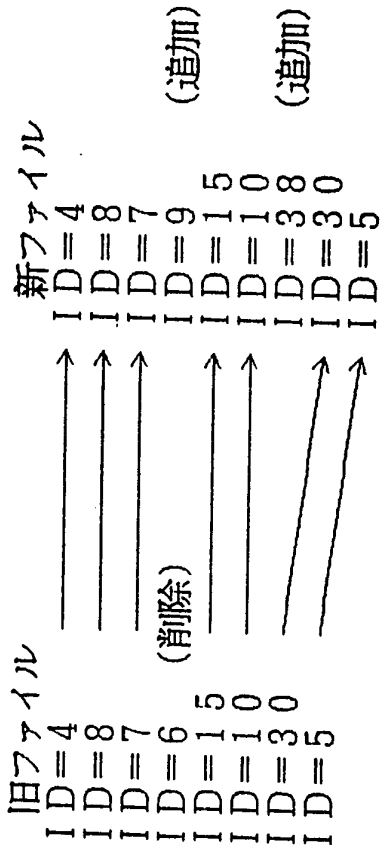

旧ファイル
ID=4
ID=8
ID=7
ID=6
ID=15
ID=10
ID=30
ID=5
(削除)
新ファイル
ID=4
ID=8
ID=7
(追加)
ID=9
ID=15
ID=10
(追加)
ID=38
ID=30
ID=5

図20

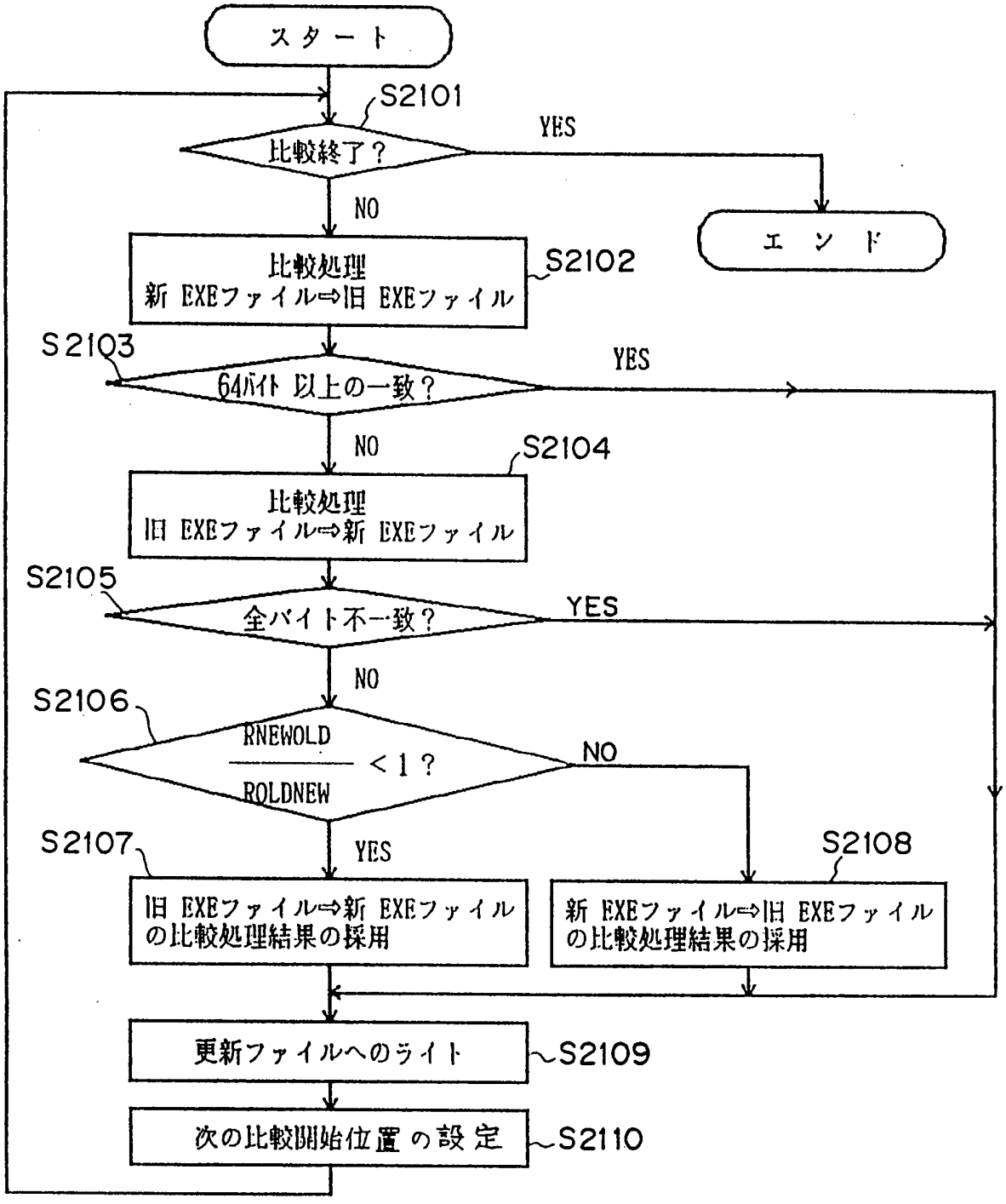

スタート
S2101
比較終了？
YES
NO
エンド
比較処理
新 EXEファイル⇨旧 EXEファイル
S2102
S2103
64バイト 以上の一致？
YES
NO
S2104
比較処理
旧 EXEファイル⇨新 EXEファイル
S2105
全バイト不一致？
YES
NO
S2106
RNEWOLD / ROLDNEW < 1？
NO
YES
S2107
旧 EXEファイル⇨新 EXEファイル
の比較処理結果の採用
S2108
新 EXEファイル⇨旧 EXEファイル
の比較処理結果の採用
更新ファイルへのライト
S2109
次の比較開始位置の設定
S2110

図 21

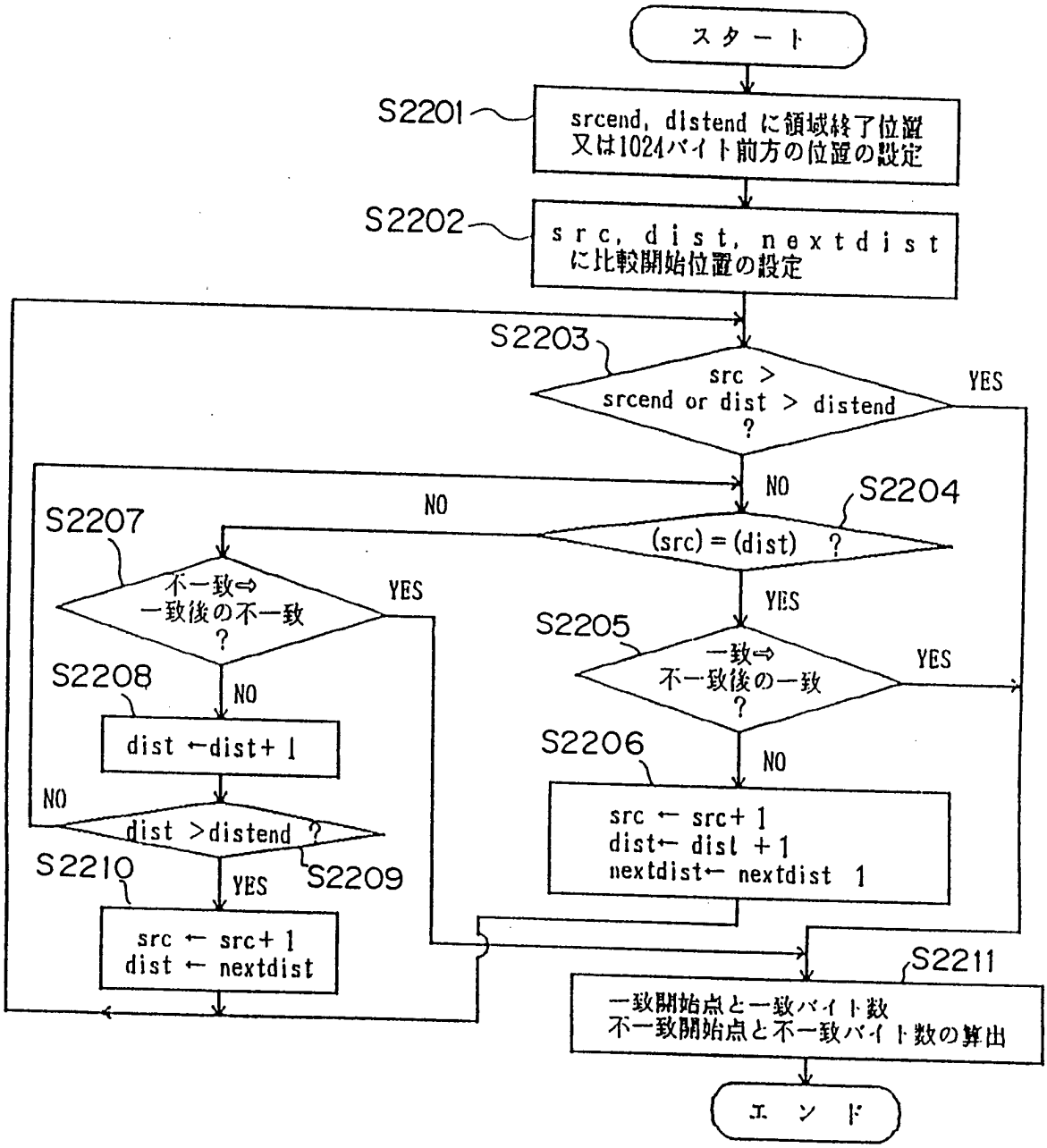

スタート
S2201
srcend, distend に領域終了位置
又は1024バイト前方の位置の設定
S2202
src, dist, nextdist
に比較開始位置の設定
S2203
src >
srcend or dist > distend
?
YES
NO
S2204
(src) = (dist) ?
NO
S2207
不一致⇨
一致後の不一致
?
YES
S2208
NO
dist ←dist+ 1
NO
dist >distend ?
S2209
S2210
YES
src ← src+ 1
dist ← nextdist
YES
S2205
一致⇨
不一致後の一致
?
YES
S2206
NO
src ← src+ 1
dist← dist +1
nextdist← nextdist 1
S2211
一致開始点と一致バイト数
不一致開始点と不一致バイト数の算出
エンド

図22

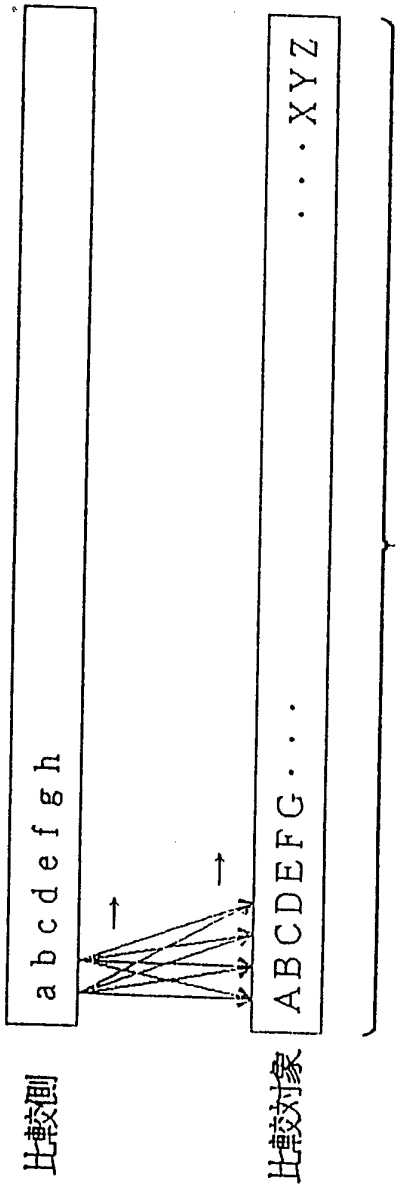
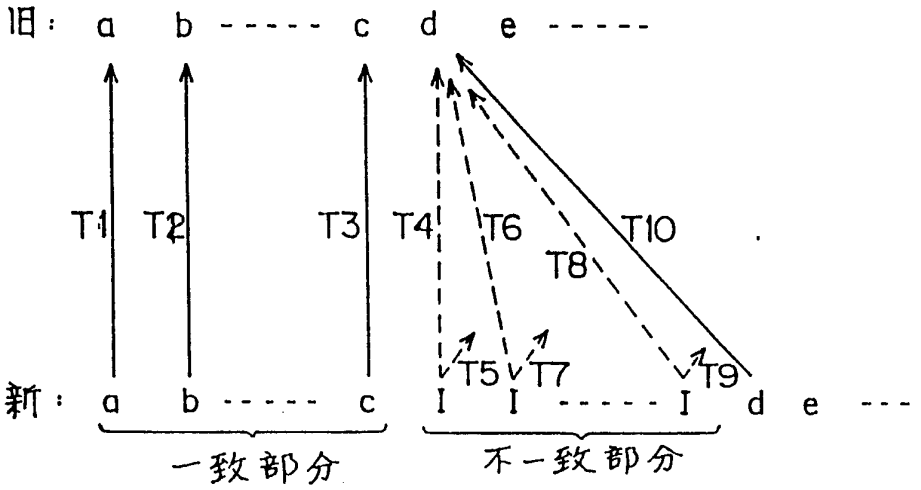


図 23

新EXEファイル → 旧EXEファイル

図24A



旧EXEファイル → 新EXEファイル

図24B

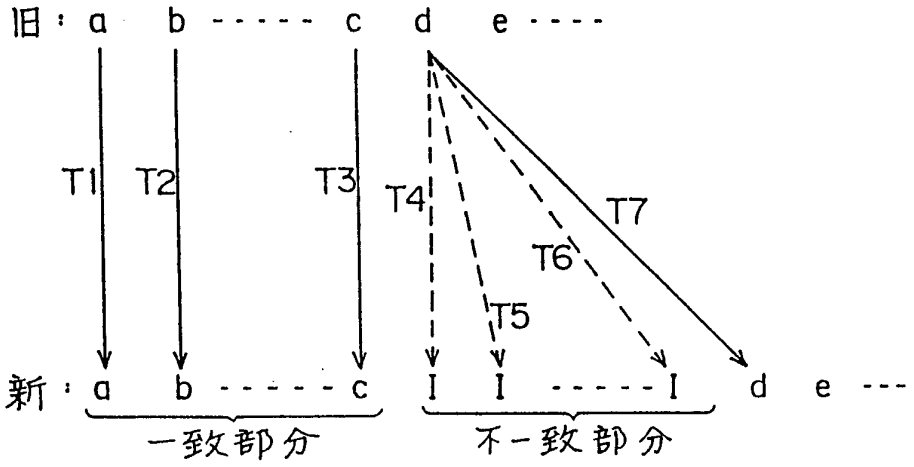


図25A

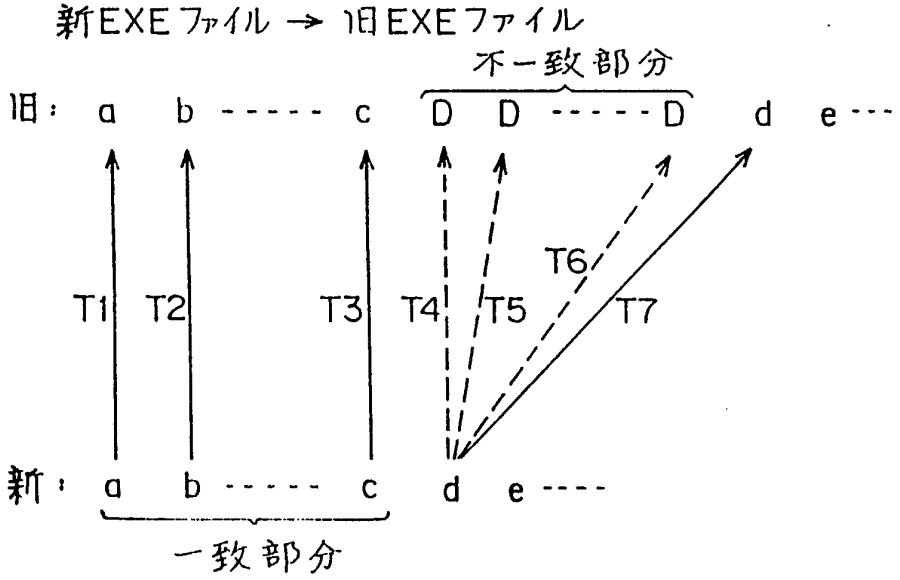
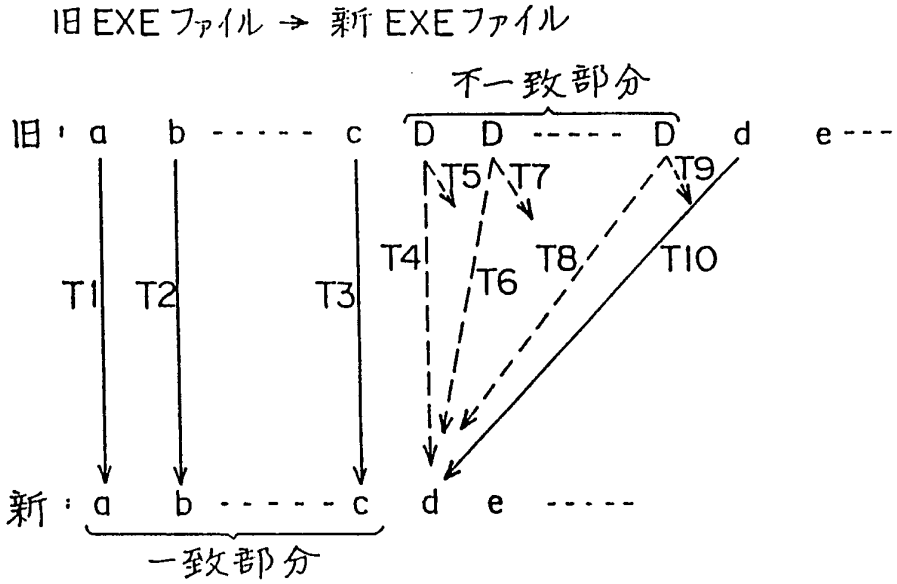
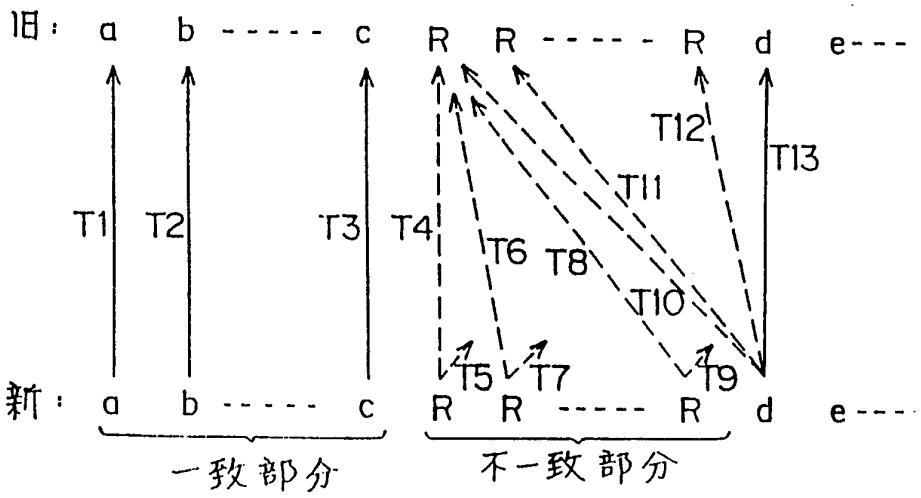

新EXEファイル → 旧EXEファイル
不一致部分
旧： a b ----- c D D ----- D d e ---
T1 T2 T3 T4 T5 T6 T7
新： a b ----- c d e ----
一致部分

図25B


旧EXEファイル → 新EXEファイル
不一致部分
旧： a b ----- c D D ----- D d e ---
T1 T2 T3 T4 T5 T6 T7 T8 T9 T10
新： a b ----- c d e -----
一致部分

新EXEファイル → 旧EXEファイル

図26A



旧EXEファイル → 新EXEファイル

図26B

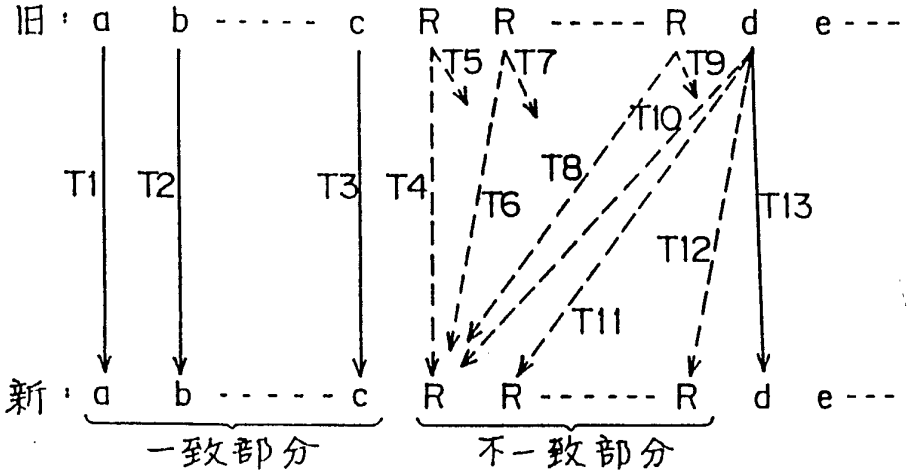


図27A 新EXEファイル → 旧EXEファイル

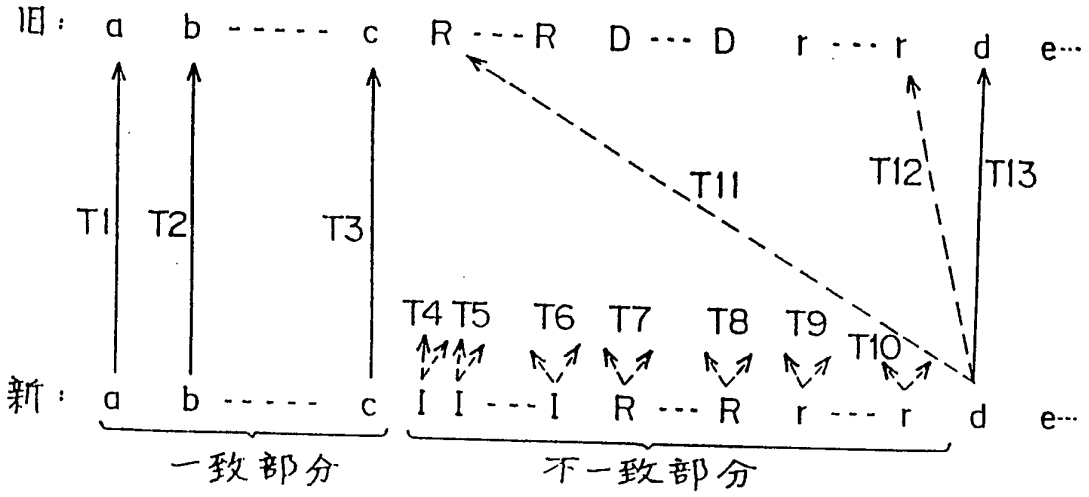


図27B 旧EXEファイル → 新EXEファイル

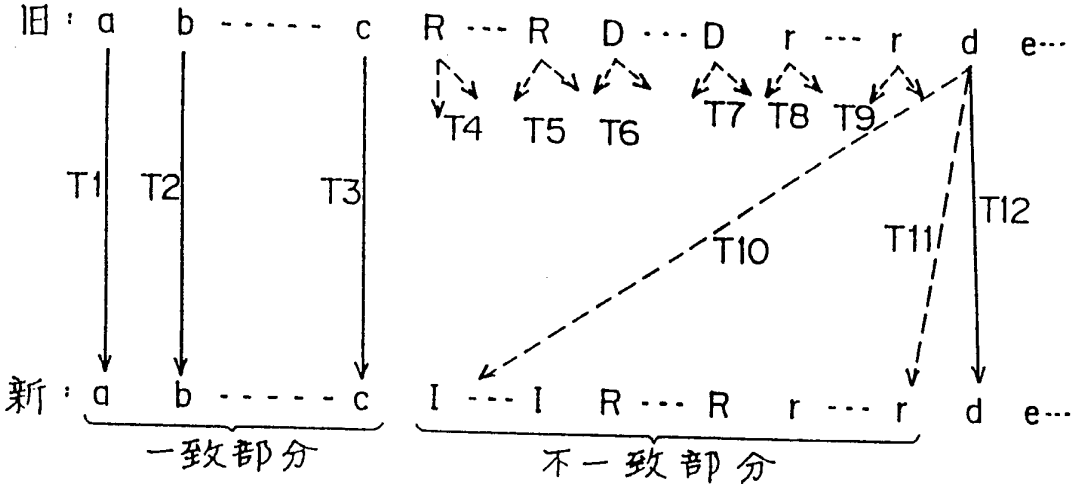


図 28A 新EXEファイル → 旧EXEファイル

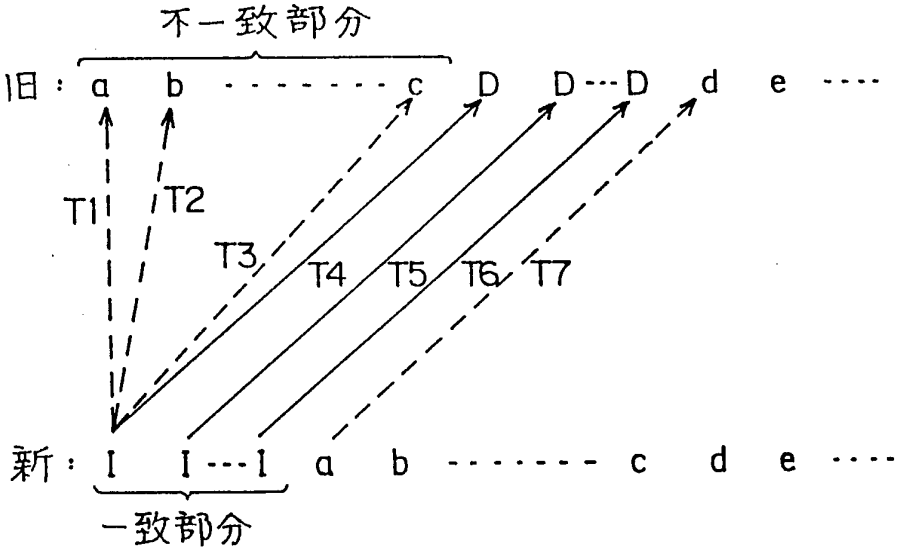


図 28B 旧EXEファイル → 新EXEファイル

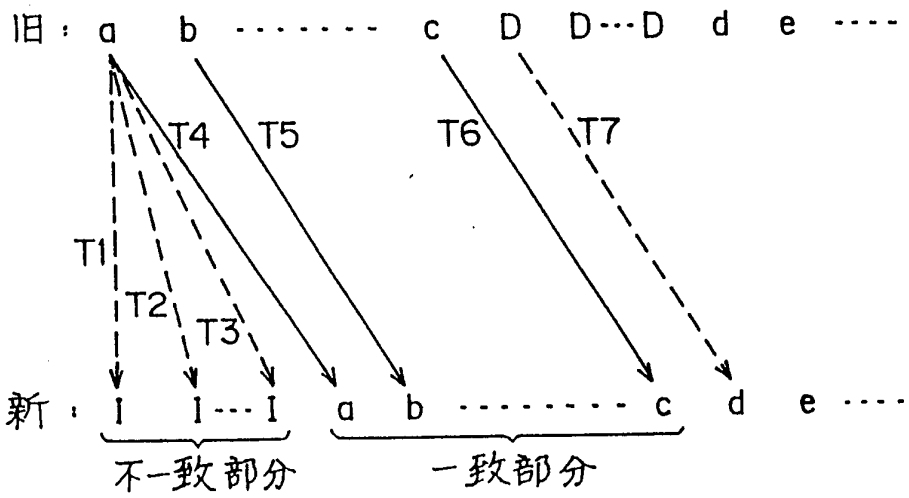


図29A ファイル構造

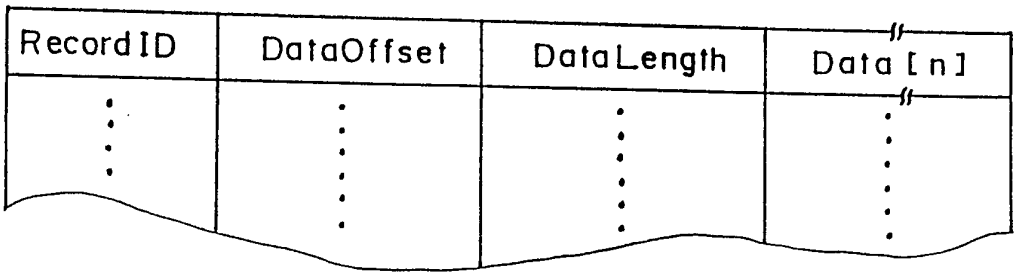


図29B RecordIDの意味

RecordID	意味	備考
0	置換レコード	DataLengthは、置換/挿入するデータ長を表す。
1	挿入レコード	
2	削除レコード	この場合、DataLength何バイト削除するのかを表し、Dataは存在しない

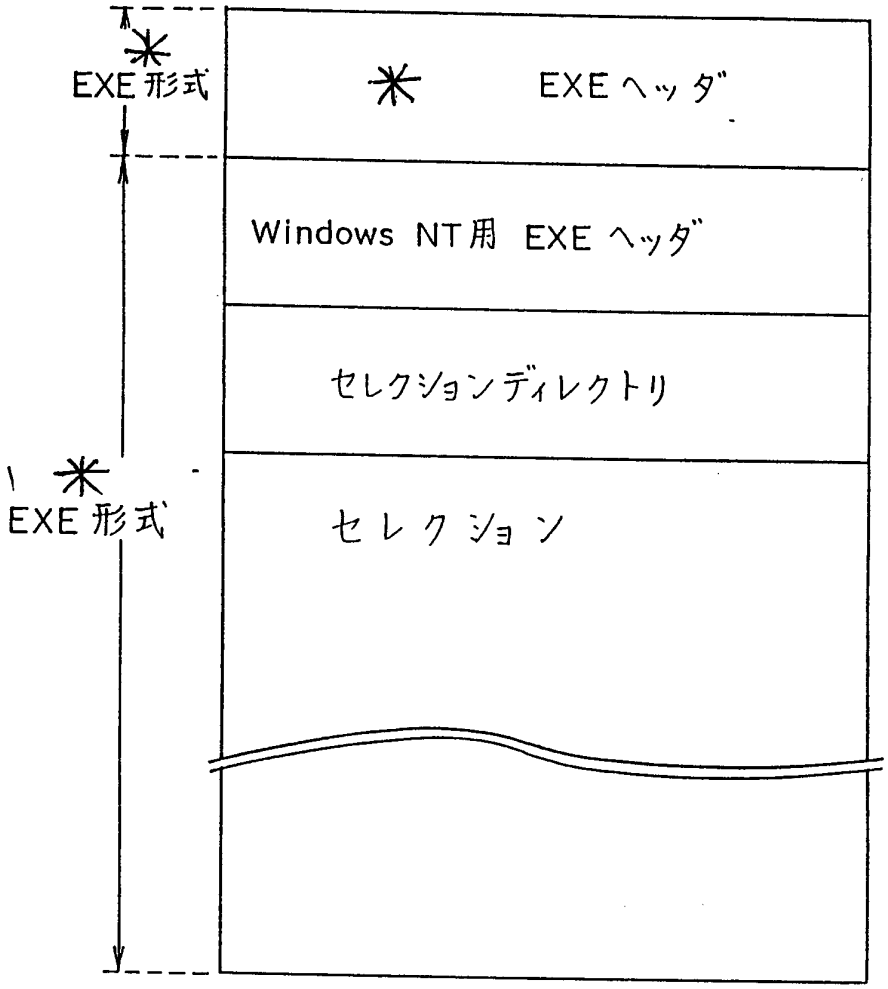


図30

According PCT Article 21(6) this part should be omitted. (※)

Fig 30

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP94/00454

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl^5 G06F12/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int. Cl^5 G06F12/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP, A, 63-76031 (Fujitsu Ltd.), April 6, 1988 (06. 04. 88), (Family: none)	1-4, 7
Y	JP, A, 4-165434 (NEC Corp.), June 11, 1992 (11. 06. 92), (Family: none)	1-4, 7
Y	JP, A, 4-362735 (Mitsubishi Heavy Industries, Ltd.), December 15, 1992 (15. 12. 92), (Family: none)	1-4, 7
Y	JP, A, 59-106063 (International Business Machines Corp.), June 19, 1984 (19. 06. 84), (Family: none)	5, 6

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
May 24, 1994 (24. 05. 94)	June 7, 1994 (07. 06. 94)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 1992)

国際調査報告

国際出願番号 PCT/JP 94/00454

A. 発明の属する分野の分類（国際特許分類（IPC））

Int. Cℓ[5] G06F12/00

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））

Int. Cℓ[5] G06F12/00

最小限資料以外の資料で調査を行った分野に含まれるもの

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP, A, 63-76031（富士通株式会社）, 6. 4月. 1988（06. 04. 88）（ファミリーなし）	1-4, 7
Y	JP, A, 4-165434（日本電気株式会社）, 11. 6月. 1992（11. 06. 92）（ファミリーなし）	1-4, 7
Y	JP, A, 4-362735（三菱重工業株式会社）, 15. 12月. 1992（15. 12. 92）（ファミリーなし）	1-4, 7

☑ C欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」先行文献ではあるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日
24. 05. 94

国際調査報告の発送日
07.06.94

名称及びあて先
日本国特許庁（ISA/JP）
郵便番号100
東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）
廣瀬文雄
5B 8526
電話番号 03-3581-1101 内線 3545

C（続き）. 関連すると認められる文献

引用文献のカテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP, A, 59-106063 (インターナショナル・ビジネス・マシーンズ・コーポレーション), 19. 6月. 1984 (19. 06. 84) (ファミリーなし)	5, 6