M-Vデータ管理(その6)： 分割型順編成ファイル領域つめ合わせプログラムFCOMPについて

宇津宮, 孝一
九州大学大型計算機センター研究開発部

山岸, 和子
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1468007

出版情報：九州大学大型計算機センター広報. 5 (4), pp.3-7, 1972-09-05. 九州大学大型計算機センター
バージョン：
権利関係：

M－V　データ管理（その6）

分割型順編成ファイル領域つめ合わせプログラムFCOMPについて

宇津宮孝一※・山岸和子※

前号では、順編成，分割型順編成ファイルの領域の大きさを知るプログラムTRACKについて述べましたが、本号では分割型順編成ファイルつめ合わせの方法について述べます。

22．分割型順編成ファイル

22.1　特徴

既に述べてきたように分割型順編成ファイルは、大記憶媒体を、メンバ名をキイとするrandom accessの手法と一たん access pointが見つかると、メンバ全体を一気に読書きする sequential accessの手法を取り入れて、効率的に利用できるという点で、大記憶上に保存して使用するプログラムライブラリの参照には、最もすぐれたファイルの編成法だと思われます。このように完成されたプログラムライブラリのアクセスには非常に便利です。しかしながら、分割型順編成ファイルは、その名のように、内容の単位であるメンバ（sequential な構造）が、ファイルの領域を分割して使用しているため、一般的にupdateする量が決まらないソース形式プログラムでは、新たな空き領域に、updateしながらメンバを作り直すことになり、そのため、共用ボリュームのように限られた領域しか許されないファイルでは、たちまち確保した領域をoverしてしまうので、非常に使い辛い面を有しています。

22.2　メンバ更新

分割型順編成ファイルのメンバの更新は次のようにして行なわれてます。

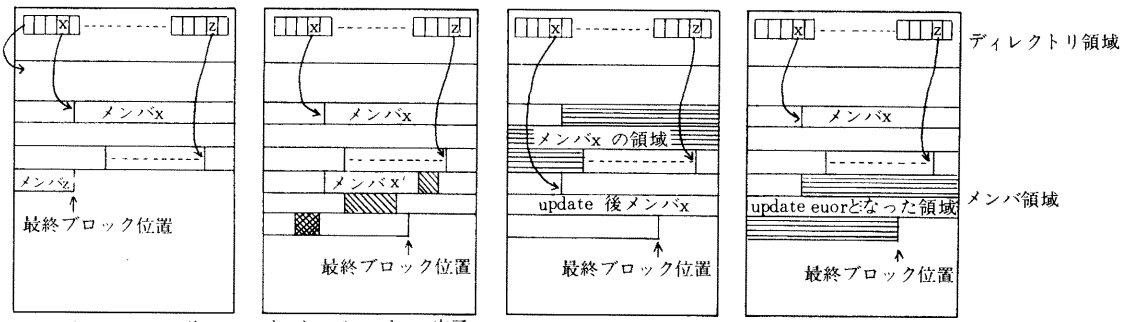


図22.1　分割型順編成ファイルの更新

a）メンバxをupdateする前のファイルのmapで、updateは最終メンバzの後に、メンバxをcopyしながら、uqdateするデータを追加，挿入，置換，削除して行きます。

b）メンバxのupdateが済んで、メンバx'ができ上った状態のファイルのmapで、▨は追加，挿入、■は置換を示します。メンバx'はメンバxより▨の部分が大きくなっています。

※九州大学大型計算機センター研究開発部

c) メンバxの開始位置が、updateされたメンバx´の位置に変更された直後のファイルのmapで、ファイルの使用領域は、元のメンバxの領域+▨が増加し、さらに悪いことには、元の領域▤は、これ以後使用できず、歯抜け状態となります。

d) メンバxのupdate中errorが発生した後のファイルのmapで、この場合メンバxは元のままですが、updateするために使用された領域▤は、実際は無意味なのに、これ以上使用できず、結局▤だけ大きな損をすることになります。

このようにして、updateするたびにファイルの使用領域はどんどん増加し、使用可能な領域(空き領域)はだんだん小さくなり、これ以上updateできないという状態になります。

$$R < \sum_{i} (Mi + mi) \quad \cdots\cdots \text{式 22 .1}$$

ここで　Mi…updateしようとするi番目のメンバの領域の大きさ

mi…i番目のメンバに追加，挿入（mi＞0），置換（mi＝0），削除(mi＜0）する領域の大きさ

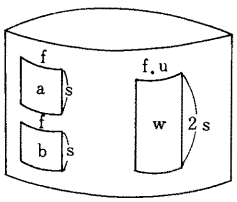
R…使用可能領域の大きさ

の時にupdateできない。つまりメンバが1つで置換の場合でも、最少限そのメンバ分の空き領域がなければ、updateできないということになります。

22.3　分割型順編成ファイル更新時の領域のつめ合わせ

ここでは分割型順編成ファイル更新時の領域のつめ合わせの方法とその問題点について述べます。

(1) 当センターでとってきた2ジョブステップ法



f ：ファイル名
s ：領域の大きさ
a ：元のファイル
w：workファイル
b ：新ファイル

図22.2　2ジョブステップ法

updateする際にジョブステップを2つに分ける。

第1ジョブステップでは：

1　元のファイル（a）名に、.uをつけ2倍の領域を持つファイルwを作成する。

2　aの内容をwにcopyする。

3　wの上でupdateする。updateが正しく終ると、ファイルaは制御プログラムより自動的にスクラッチされ第2ジョブステップへ進む。update errorが起ると、ファイルwは自動的にスクラッチされジョブは終了する。

第2ジョブステップでは：

1　元のファイルと同じファイル名のファイルbを作成する。

2　wの内容をbにcopyする。(copy時に領域はつめ合わされる)

3　copyが正常に終ると、ファイルwが自動的にスクラッチされる。

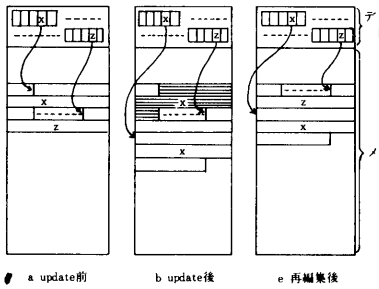
以上のように、この方法は、2ジョブステップにするだけで、簡単にupdate後の領域のつめ合わせができますが、次のような問題があります。

①このジョブの実行中にシステムダウンした場合、あるいは第2ジョブステップでwからbに

copyする際、bに入り切れなくなった場合に、いろんな形でファイルがボリューム上に残り復元したり消去したりする手間が大変です。

②bのファイルの大きさは、利用者が指定しなければいけないので、利用者にとっても、ボリューム管理者にとっても、かなり心配です。

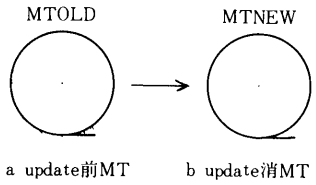
(2) 同一ファイル法



この方法は、update終了後、領域の歯抜け状態（b）が起った場合、そのままにしておかないで、同一ファイル上で再編集して領域のつめ合わせを行なうもので、work用のファイルをつくる必要もないので便利であるが、式22．1のようになった時には、updateができないという問題点があります。

図22.3　同一ファイル法

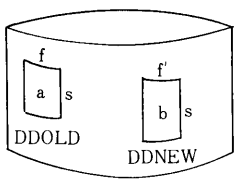
(3) 磁気テープモード法



この方法は、現在　LIBE(LIBraryEditor)　が、磁気テープをupdateする際に用いている方法を大記憶に応用するものです。

磁気テープのupdateは、aのMTの内容をupdateデータを見て追加，挿入，置換，削除等を行ないながら、bのMTに移していきます。この際updateする順序は、aのMT上に書かれたメンバの順序でなければなりません。

大記憶の場合は、ボリューム上に同一ファイル名のファイルは作成できないため多少処理内容は異なるが原理は同じです。



f：ファイル名
s：領域の大きさ
a：元のファイル
b：新ファイル

1　aと同じ大きさの領域を持つファイルbを作成する。bのファイル名は、aのファイル名に変換できるようなものにしておく。

2　updateデータは、alphabetical orderに前もって並べておき、aのメンバをalphadetical orderで、updateデータを見ながら、追加，挿入，置換，削除等を行ないながらbのファイルに移して行く。

3　updateが完了したら、aのファイルをスクラッチしてbのファイル名をaのファイル名にする。

update errorの場合は、bのファイルが自動的にスクラッチされる。

図22.4　磁気テープモード法

この方法は、ファイルのread／writeの回数も最少であり、領域のつめ合わせもupdate後に自動的にでき上っているので、大変効率がいい方法です。しかしながら、

①この方法は、現在、分割型順編成ファイルでupdateを行っているすべての処理プログラムを書き変える必要があり、メーカーに依存する他ない。

②システムダウンした時にｂのファイルが残る場合がある。

のような問題点があるが、非常によい方法だと思われます。

22.4　FCOMP（File COMPact）プログラムの実現

22．3で分割型順編成ファイルの領域のつめ合わせの方法について述べましたが、つめ合わせの方法として次のような結論を得ました。

①空き領域にupdateするデータが入る限り、updateできる。

② update errorが起った場合は、updateする以前の内容がファイルに残る。

③update途中でシステムダウンしても、容易に復元できる。

④updateする人がファイルの大きさを意識しないでいいし、ファイル名は変わらない。

⑤マルチジョブの可能なシステムでは、updateするジョブが同時に流れないようにする。

以上の点を考慮し、さらに次の前提条件をつけて、つめ合わせプログラムFCOMPを考えた。

⑥updateするファイルがあるボリューム上に、同じ大きさの領域のファイルが作成できる余裕がある。

⑦update時の領域のつめ合わせ、システムダウン時の復元ができれば、ファイルに対するアクセスの回数が増えても仕方がない。

22.4．1　FCOMPの概要

FCOMPは22．3で述べた(1)の方法を大幅に改善したものです。

(1) ファイルの関係

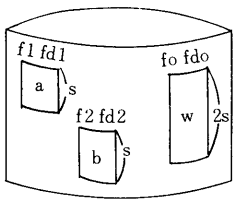


図22.5　ファイルの関係

a）updateの対象となるファイル

f1 ：ファイル名

fd1：ファイル定義名

b）update後のファイル

f2：ファイル名（最終的にはf1となる）

fd2：ファイル定義名

w）workファイル（領域の大きさはａ，ｂの高々２倍）

fo ：ファイル名

fdo：ファイル定義名

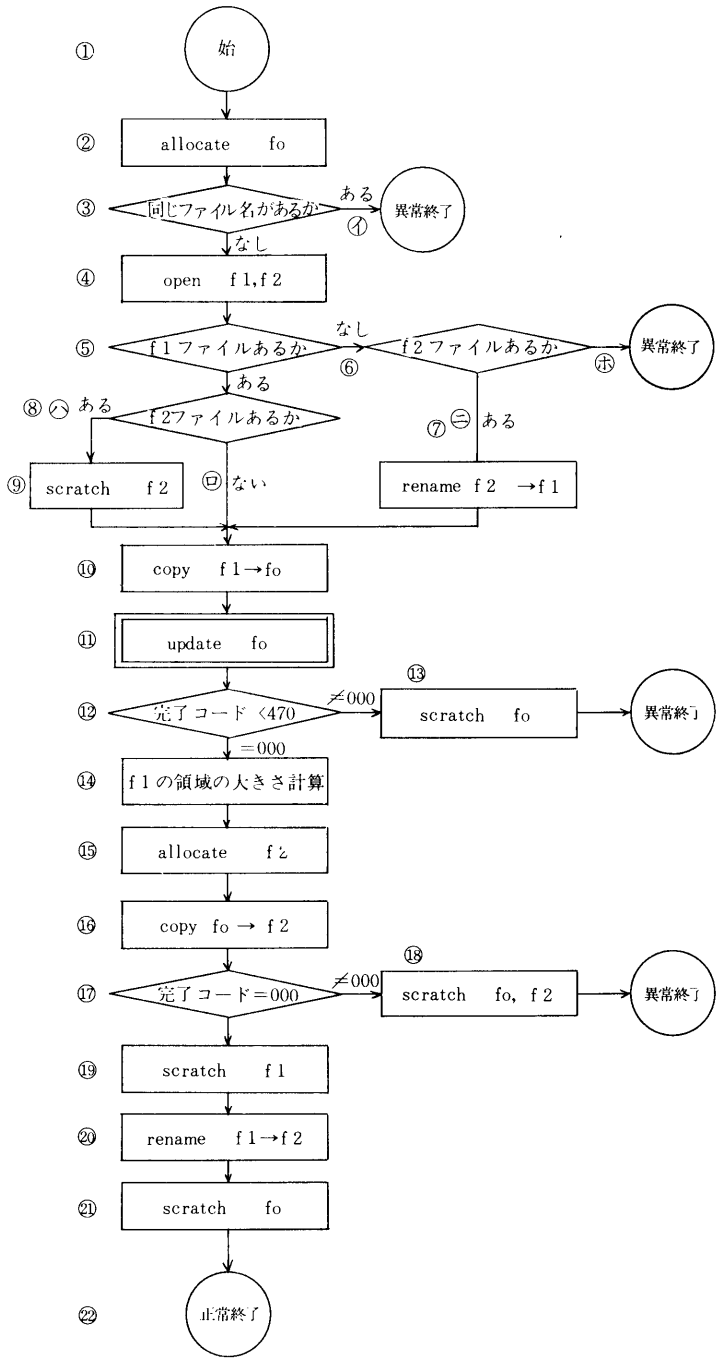
(2) ファイルの状態とその処置

FCOMPが処理を開始する際のファイルの状態は５通りです。

	fd0	fd1	fd2	状　　態	処　　置
㋑	あり			同一ファイルをupdateするジョブが同時に２個以上流れようとした。	このジョブをアボートする。
㋺	なし	あり	なし	updateが正常にできる	updpteを行なう。
㋩	なし	あり	あり	update時、ｆ０からｆ２にcopyする際中かcopyを終えｆ１をスクラッチする際中にシステムダウンしたものと思われる。	ｆ２をスクラッチしてupdateを行なう。
㋥	なし	なし	あり	update時、ｆ１をスクラッチ後にシステムダウンしたものと思われる。	ｆ２をｆ１に変更してupdateを行なう。
㋭	なし	なし	なし	両方ともファイルがないということはファイル名の指定エラーと思われる。	このジョブをアボートする。

表22.1　ファイルの状態とその処置

22.4.2　FCOMP処理の流れ



① FCOMPプログラムの開始

② workファイルf0の開設

③ f0がすでにあれば、updateジョブが同時に2つ以上流れたため、このジョブを異常終了させる

④ f1, f2 ファイルのopen

⑤ f1ファイルがあるかないかのチェック

⑥ f1ファイルがない場合f2ファイルのチェック、f2もなければファイル名の間違いとして異状終了する

⑦ f1がなくf2があるので、f2のファイル名をf1にする

⑧ f2ファイルがあるかのチェック

⑨ f2ファイルがあれば、f1もあるので、f2をスクラッチする

⑩ f1の内容をf0に移す

⑪ f0上でupdateする（分割型順編成ファイル / updateプログラム）

⑫ updateが異状（完了コード0でない）かどうかのチェック

⑬ 異状であればf0をスクラッチして異状終了する

⑭ 正常終了であればf1の領域の大きさを計算する

⑮ f2ファイルの開設

⑯ f0ファイルの内容をf2ファイルへ移す

⑰ 正常に終ったかどうかのチェック

⑱ 異状であればf0、f2をスクラッチして異状終了する

⑲ f1ファイルをスクラッチする

⑳ f2ファイル名をf1と同じファイル名にする

㉑ f0ファイルをスクラッチする

㉒ 正常終了

図22.6　FCOMP処理の流れ

以上FCOMPプログラムについて述べてきました。このプログラムの完成により分割型順編成ファイル領域のつめ合わせができ、システムダウンしてもファイルは完全に復元できることになると思われます。