入出力管理ルーチン*

大　駒　誠　一**

1. まえがき

電子計算機のプログラミングのうちで，入出力に関する部分を効率よくしようと思うと，かなり複雑でむずかしく，プログラムを組むたびに，いちいちそのめんどうな手続きを考えるのは能率悪く賢明でない．しかし幸いなことに，この入出力操作の大部分がかなり一般的な形をしているので，あらかじめ入出力に関する標準的な操作手順のプログラムをアセンブラやコンパイラの中に組み込んでおき，プログラマが入出力操作を行なうときには，それに必要なパラメータや簡単なマクロ命令を与えるだけで，プログラム実行時にはモニタと緊密な連絡をとって計算機の演算と入出力装置の動作をスムーズに行なわせるようにしたのが入出力管理ルーチンで，どんな計算機システムでも名称こそ違え，必ず備えているものである．

一口に入出力管理ルーチンといっても，各メーカごと，あるいは計算機ごとにそれぞれ異なっているので，ここでは，特定のものについてはふれず，データ処理用の共通言語 COBOL の入出力部分を完全に行なえるものを，入出力管理ルーチンの一つの標準として話を進めることにする．したがって，COBOL が主として磁気テープのようなデータを逐次読み書きする入出力装置を取り扱うことになっているので，ここでも必然的に磁気テープに関する入出力操作が主体となる†．

また入出力管理ルーチンは，パラメータやマクロ命令によりオブジェクトプログラムを作り出す部分と，実行時に主プログラムといっしょに記憶装置中にあって入出力装置を操作する部分とあるが，ここでは主として後者について述べることにする．

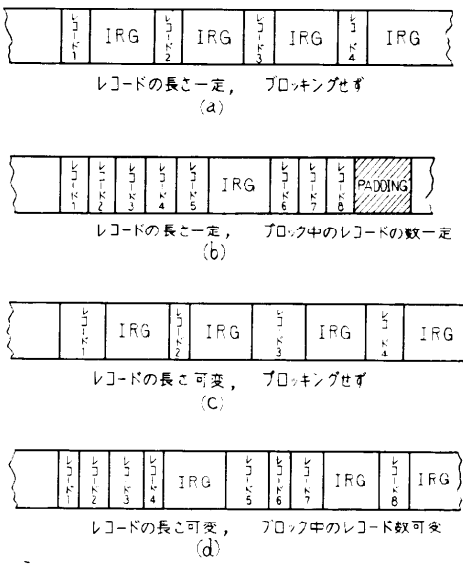
2. 入出力管理ルーチンの諸機能

入出力管理ルーチンが処理できなければならない最少限の機能をあげる．ここではファイルは磁気テープにあるものとし，テープ1巻にはいりきらず複数本のテープにはいっているファイルをマルチリールファイルと呼び，1巻のリールに複数個のファイルがはいっているときは，マルチファイルリールと呼ぶ．

2.1 ブロッキング

磁気テープ上のレコードは，磁気テープの容量を増すためと速度を上げるために，ブロッキングと称して，いくつか一まとめにブロックして記録するのが普通である．これにより，IRG (Inter Record Gap: ブロックの区切れ目で何も書いてない部分) の数が減り，したがって，テープの起動停止の回数も減って，容量，速度とも数倍ないし 10 数倍になる．しかし，ブロッキングしてあっても，主プログラムからの読み込みは見掛上1レコードずつはいってくるように入出力管理ルーチンが調整する．

ブロッキングのタイプには**第1図**のようにいくつかあるが（b）の場合には短いブロックには Padding をしなければならないし，(d) の場合にはブロックやレコードの長さの情報がそれぞれブロックごと，レコードごとにはいっていなければならない．



第1図　ブロッキング

2.2 バッファリング

いま入力エリア（バッファ）を磁気テープ1ブロッ

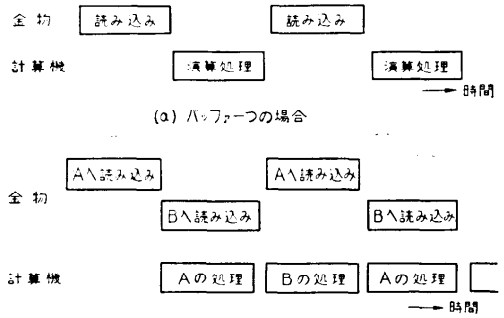
* Input Output Control Routine, by Seiichi Ohkoma (Faculty of Engineering Keio University)

** 慶応義塾大学工学部管理工学科

† 最近 COBOL には，ディスクのような番地のついたランダムアクセスのファイルも処理できる機能が加わった．

ク分だけしかもっていないとすると，**第2図**（a）のように入力している間は計算機が待って，計算している間は磁気テープが止まっているという状態になり，非常に効率が悪い．

ところが普通の計算機では，計算機本体が演算している間に入出力操作が可能なので，バッファリングといって一つのファイルに対してバッファを二つつくり，片方のバッファに磁気テープから読みながら，もう一方のバッファの内容を処理し，それが終ると交代して，今まで処理していたほうに読み込み，反対側の内容を処理する（**第2図**（b））．このレコードの先き読みということによりプログラムはやや複雑になるが全体の効率は向上する．



第2図　バッファリングによる計算機と磁気テープの動き

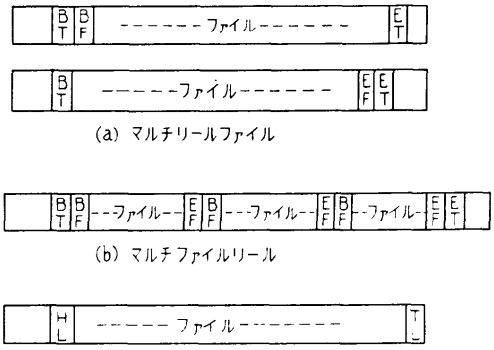
出力するときはこの逆で，1ブロック分のバッファを磁気テープに書き出している間にもう一つのほうのバッファに処理済みのレコードをためていく．

現状ではまだまだ入出力装置の速度は演算処理速度に比べて遅いので，計算機を有効に使うためには，このブロッキングやバッファリングを駆使して入出力速度を補ってやらなければならない．

2.3 ラベル

必要な磁気テープを誤って消してしまったり，間違ったファイルを使ったりしないためと，各種の検査のために磁気テープ1巻の始めと終りには，ラベルと呼ぶ特別のレコードをつける．

このラベルには**ファイル**の始めと終りを示すラベル（BEGINING-FILE-LABEL, ENDING-FILE-LABEL）と，**テープ**の始めと終りを示すラベル（BEGINING-TAPE-LABEL, ENDING-TAPE-LABEL）のそれぞれ2種類あるが，その配列順序は**第3図**のようでテープラベルは各テープリールの前後に必ずあり，ファイルのラベルのファイルは前後につく．しかしながら，実際には**第3図**（c）のようにファイルの始めのラベルとテープの始めのラベルは一つにして（HEADER-LABEL）共通に使うのが普通で，ファイルの終りのラベルとテープの終りのラベルも同様である（TRAILER-LABEL）．



BT: Begining-Tape-Label　**ET**: Ending-Tape-Label
BF: Begining-File-Label　**EF**: Ending-File-Label
HL: Header-Label　**TL**: Trailer-Label

第3図　ラベル

2.3.1 ラベルのもっている情報とその機能

始めのラベル

センチネル：テープまたはファイルの始めのラベルであることを示すもの．

リール番号：そのリールの固有の番号，ファイルの内容が変わってもこれは不変である．

作　製　日：ファイルをそのテープに書いた日付．

解　除　日：そのテープの内容をこわしてもよい日．

ファイル名：ファイルの名前

終りのラベル

センチネル：テープまたはファイルの終りのラベルであることを示すもの．

ブロック数：そのテープまたはファイルにはいっているブロックの数

レコード数：そのテープまたはファイルにはいっているレコード数．

Hash Total：各レコードの特定の欄の総和

ラベルにはこのほか，磁気テープの読み込み，書き出しの際のエラーの回数を記録しておいて，テープの廃棄の時期を決めたり，寿命の統計をとったりするほか，プログラマが必要と思うテープやファイルに関する情報を入れておく．

このラベルについてはまだプログラム言語のように国際的な標準はないので，各メーカで入出力管理ルーチンの取り扱える標準ラベルを決めており，その標準のもの以外はプログラマが処理する．

2.4 誤りの訂正

磁気テープは他の磁気ディスクや磁気ドラムなどの外部記憶装置に比べて，起動停止がはげしく，しかも外気に直接ふれるなど，使用条件が非常に苛酷なのでテープに傷がついたり，ゴミがついたりして，読み込み書き出しの際に誤りを起こす割合がずっと多い．しかし磁気テープの場合は，カードやプリンターと違って読み直し，書き直しが可能なのでプログラムで次のような処置をして実質的にエラーを減らすことができる．

2.4.1 読み込みの場合

読み込みの際エラーが起こるとすぐその部分をバックスペースして読み直し，正しく読めるまで10回ぐらい繰り返す．それでも，なお，エラーが起こるときはあらかじめ決めた処置をとらせる．たとえば，その旨オペレータに知らせて，以後の仕事を中断してコントロールをモニタへ帰したり，あるいはエラーした部分をプリンターで印刷したり，他のエラー専用の磁気テープに書いたりしてそのまま作業を続ける．

また，あるときは 10 回エラーしたら数ブロックまとめてバックスペースし，また戻して読んでみる．こうすると読めない部分が真空掃除されたことになりゴミがうまく除去され，読めるようになることがある．さらには1本のリールについてエラーが頻繁に起こる場合にもその旨オペレータに知らせるようになっていると良い．

2.4.2 書き出しの場合

書き出しの際にエラーが起こると，やはり一度バックスペースして書き直してみる．また，エラーが起こればその部分は消去して次の部分へ書く．そこもエラーが起これば，もう一度書き直してみる．こうして何回か悪い部分を消去して書き直す操作を繰り返してまだエラーの起こる場合は，磁気テープが相当傷んでいるか，磁気テープ装置そのものの故障であると考えて，その旨オペレータに知らせるなど，しかるべき処置をする．また，このように二度書いてみないで一度でもエラーすればその部分は消去してしまうこともある．書くときにエラーしたところは読み込みの際にもエラーする可能性が多いと考えるからである．

2.5 中断と再開

プログラム実行中に何らかの原因でその仕事を中断しなければならないことがある．たとえば緊急の仕事が割り込んできたときとか，入出力装置や記憶装置がエラーを起こしたときとか，入力データに誤りがあって先へ進むのが無意味なときとか，予定の時間や出力量を超過したときなどは，モニタあるいはオペレータの指示でプログラムを中断させる．この場合そこまでの作業が無駄にならないように記憶装置の全内容，磁気テープの位置などを適当な時点で特定の磁気テープなどの出力装置に記録しておいて，最初からやり直さないでも，続けられるようにしておく．

この再運転の情報を書き出す時点は，システムで決まっているものと，プログラマが与えるものとがあるが，テープ1巻が終るごと，一定の数のレコードを処理するごととか，何分おきとかが一般的である．

3. 入出力管理ルーチン

これまで述べてきたように，プログラム中で入出力操作を実行しようというときには，いつもブロッキング，バッファリング，ラベルの検査，あるいはエラーしたときの処置などを考慮しなければならないが，入出力管理ルーチンでは，これを一定の様式にしたがって，ファイルの形式，ブロックの大きさ，各種の検査のやり方などに関するパラメータを前もって与えておいて，実際にプログラム中で入出力させるところには簡単なマクロ命令を書くだけで，実行時に上述の必要な機能へリンケージをとるようなプログラムを作り出してくれて，プログラマの負担を軽減し，実行時の効率も向上させることができる．

3.1 パラメータ

プログラマが与える各ファイルについてのパラメータには次のようなものがあり必要に応じてそれぞれ決められた方法でコーディングシートに書く．

（a） ファイルの名前
（b） レコードの名前
（c） 入出力装置とその数
（d） マルチリールファイルかどうか
（e） マルチファイルリールかどうか
（f） マルチファイルリールのときはその位置
（g） バッファが標準の数以外のときその数
（h） 多重プログラムをするときその優先度
（i） 記録のモードたとえば BCD と Binary.

（j） 1ブロックにはいるレコード数または文字数，可変の場合は最大と最小
（k） 1レコードにはいる文字数，可変の場合は最大と最小
（l） ラベルの有無また標準ラベルかどうか
（m） ラベルの有る場合，検査すべきもの
（n） 入力か出力かあるいは両方か
（o） ファイルが終ったときの行き先
（p） 巻き戻しの方法
（q） エラーしたときの処置

まだこのほかに特定のインデックスレジスタを割当てたり，他のバッファと共通に使うことを指定したりすることがある．COBOL では上にあげた n, o, p の三つはあらかじめ決めないで，各動詞ごとに決めている．

レコード内のこまかい構成は入出力管理ルーチンにとって必ずしも必要ではない．

3.2 マクロ命令

入出力管理ルーチンではプログラム中で入出力の必要なときに，実際の金物の命令でなく，ファイルかレコードの名前を伴ったマクロ命令を書く．これがオブジェクトプログラムでいくつかパラメータをもったサブルーチンへのジャンプ命令におきかわる．

ここでは COBOL のおもな入出力用の動詞 OPEN, CLOSE, READ, WRITE の四つを取上げるが，この動詞の果たす機能を全部マクロ命令がもっていれば，COBOL の入出力用動詞のコンパイルは非常に簡単になる．

3.2.1 OPEN

ラベルの検査など各ファイルの入力や出力の準備をする．誤りがあればモニタへコントロールを戻す．

a）入力ファイル：始めのラベルを読んで，ファイルの名前，テープの番号などが正しいかどうか検査し，次の読み込み命令に備える．

b）出力ファイル：まずラベルを読んで解除日を調べ，解除日がきていればいったん巻き戻して，新しいファイル名，作製日，解除日などのはいったラベルをテープに書き，次の出力命令に備える．

3.2.2 CLOSE

ファイルに関する終りの作業をする．

a）入力ファイル：終りのラベルにあるブロック数，レコード数，Hash Total などを検査して指定に従って巻き戻しをする．

b）出力ファイル：バッファにまだ書いてないレコードが残っていれば，これを出力し，さらにブロック数，Hash Total，レコード数などのはいったラベルを書き，指定に従って巻き戻しをする．

3.2.3 READ

指定したファイルからレコードを一つ読んで，演算処理を可能にし，必要ならばレコード数，Hash Total の計算をする．実際には，このマクロ命令により入力装置が動いて記憶装置にはいってくるのではなく，前もってバッファに読み込んであったものが表面にあらわれてくるだけである．

マルチリールファイルのとき1巻の処理が終ると，終りのラベル中のレコード数や Hash Total とそれまで計算してきたものとを検査し，巻き戻して，次のテープのラベルを読んで続きのテープであることを確かめて最初のレコードを読み込む．

3.2.4 WRITE

レコードを一つ出力用エリアに移す．出力エリアがいっぱいになれば，いつでも磁気テープに書き出せる準備をして出力装置のあくのを待つ．むろんレコード数や Hash Total も計算する．

テープ1巻が終りになれば，レコード数や Hash Total を入れた終りのラベルを書き，巻き戻して，次の空いているテープの解除日を調べ，新しいラベルを書いてから次のレコードを書き出す．

3.2.5 その他

このほか入出力管理ルーチンにはプログラム再開用の情報を書き出したり，プリンタのキャリッジをコントロールしたり，カードのスタッカを選択したりするマクロ命令がある．これらはコンパイラのレベルでは表面にあらわれてこないが，コンパイラがオブジェクトプログラムに落す際のアセンブラなどの中間言語には持っていたいものである．この他2進 10 進変換，出力形式の編集なども含んでいることがあるが，要するに，プログラマがループを作ってブロッキングの処理をしたり，いちいちレコードの数や Hash Total の計算などわずらわしいことを考えないですむように，入出力管理ルーチンはプログラムを組むとかなり複雑であるが，ある程度標準的な形にしてさしつかえない入出力関係の操作全域にわたっているわけである．

2.3 主プログラムとの関連動作

入出力装置の動作と演算処理を並行して行なわせて待ち時間を減らして効率をあげようというのが入出力管理ルーチンの主目的であるが，実際には主プログラ

ムと入出力管理ルーチンが同時に動いているのではなく，入出力装置の動作中は主プログラムが働き，入出力装置の動作が完了すると，入出力管理ルーチンが動き始めて，必要な入出力装置を働かせて，主プログラムにコントロールを戻す．

入出力装置があいたことを知るにはそのことによって起こる“割り込み”を使うのが最も簡単で普通である．しかし計算機によっては割り込みの機能を持っていないものがあり，この場合には，待ち時間は多くなるが入出力のマクロ命令ごとに入出力管理ルーチンが働くようにしておいて，現在処理中の反対側のバッファのものを入出力するようにしておくか，あるいは，アセンブラやコンパイラが，オブジェクトプログラムを作りだすときに，プログラムの適当なところに（たとえば，50命令ごと，あるいは CLA: clear add の命令の直前など）入出力装置があいたかどうか調べる命令をそう入して，あいていたら入出力管理ルーチンにジャンプさせるようにする．しかし，命令の数は少ないが時間のかかるループにたまたまその命令がはいらなかったというような場合があるなどして，いずれにせよ割り込みがない場合は完全を期しがたい．

いまここではチャンネルがあいたという割り込みにより入出力管理ルーチンにコントロールがきたものとすると，まずアキュムレータや各種のレジスタを退避させ，入出力管理ルーチンが動作中に割り込みが起こると困るので割り込みを禁止しておく．次に前の入出力操作にエラーがなかったかどうか調べて，エラーがあればその回復処置を行なう．なければそのあいているチャンネルにつながるファイルのうち，最も優先度の高いファイルに関する入出力装置を動作させて，最後にアキュムレータや割り込みの禁止などを，もとに戻してコントロールを主プログラムに帰す．

この部分はパラメータやマクロ命令でプログラマが参照することはないので目だたないが主プログラムと入出力管理ルーチンが連繋動作をするためには最も重要な部分である．

4. むすび

最近は大型のコンパイラやいろいろなゼネレータの陰にかくれて，入出力管理ルーチンはやや人々の興味から遠のいている感があるが，最新の計算機でもその入出力操作が簡単になったわけではないので，コンパイラやゼネレータが必ずその下に入出力管理ルーチンを含んでおり，その重要度はいささかも低くなってはいない．また近頃はモニタが次第に強力になってきて，入出力操作に関する機能も含むようになり，入出力管理ルーチンとして独立したものよりも，モニタの一部になってしまう傾向があるが，要するに全体が効率よく動けばどちらでもかまわないわけである．しかし，モニタに入出力に関する機能も含ませた方が，モニタの占有面積が多少ふえてもすっきりするようである．

（昭和39年9月15日受付）