実行時のファイル利用について

石田, いつ子
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1470293

出版情報：九州大学大型計算機センター広報. 9 (1), pp.34-50, 1976-03-05. 九州大学大型計算機センター
バージョン：
権利関係：

実行時のファイル利用について

石田　いつ子[*1]

利用者がファイルを利用する際の目的として，大別すると2通りある．1つはプログラムを登録して使う方法（この場合のファイルをライブラリファイルという）であり，あとの1つはプログラムの実行時に中間結果を保存しておくために使う方法（この場合のファイルを補助記憶用ファイルという）である．

これから述べるのはこの補助記憶用ファイルの利用についてであるが，ここではFORTRAN-D[*2]実行時（以下，実行時という）の説明をしていくことにする．

どの様な場合に補助記憶用ファイルを使うと有効であるか，主なものを挙げてみよう．

1. 大量の同一データをしばしば使う．
2. 実行結果を次の計算に使いたい．(現在，カードに出力している人がいるが，出力の際に値が丸められるので，後で述べるようにバイナリ形式でファイルに出力したほうがよい．)
3. コアの領域が足りない．
   ①中間データが大量に出る．
   ②大きな配列を処理する必要がある．
   ③その他

利用者が上記のような場合にファイルを利用すればきわめて有効であるが，しかし，システム側からみるとファイルのアクセス回数が多くなるということになり，システム効率が下がることになる．[*3]

利用者もこの点を留意してファイルを使っていただきたい．

補助記憶用ファイルは，利用の仕方により，利用者自身の専用ファイルを使う場合と，システムが提供する作業用ファイルを使う場合がある．2つのファイルの違いは，前者は利用者が希望するまで保存させておけるが，後者はジョブが終わると同時に消去されてしまうことである．

1，2のような場合は利用者自身の専用ファイル，3のような場合は作業用ファイルを使うとよいだろう．

利用者が実際にファイルを利用する場合としては，

1. ファイルの確保の方法
2. プログラム上で入出力はどのようにしたらよいか．
3. できるだけアクセス回数をへらすにはどうしたらよいか．

＊1　九州大学大型計算機センター研究開発部

＊2　ファイルの取り扱いについてはFORTRAN−Hの方が機能が拡張されているが，現時点ではFORTRAN−Dの利用者が多いので，今回はこれに限った．

＊3　ファイルアクセス回数の非常に多いジョブが実行されると，そのジョブは中央処理装置はあまり使わないのに主記憶装置を占有してしまい，その間他のジョブは実行されないことになってシステム効率は著しく低下する．
現在，ファイルのアクセス回数に対しても負担金を課すように検討がなされている．

4. ファイルに書ける情報量はどの程度か.

などがある. 以下これらについて説明していくことにする.

1. ファイルの確保

利用者自身の一般ファイルは, ファイル確保用のマクロ¥ALLOCATE, あるいはCPSのコマンドALLOCATEで確保できる. どちらで確保してもよいが, CPSの方が結果が早くわかるという点で便利であろう.

作業用ファイルは, ファイル定義用のマクロ¥F. WORKをジョブ構成デックに挿入しておけば, システムが実行時に必要に応じて確保してくれる.

1・1 #ALLOCATEコマンド

	コマンド名	オペランド
#	ALLOCATE	ファイル定義名, ファイル局所名, GEN, SPACE：ファイル確保量, FORG：PS

パラメータの説明

ファイル定義名……12文字以内の英数字で第1文字が英字である名前を指定する.

ファイル局所名[*1]……6文字以内の英数字で第1文字が英字である名前を指定する. ただし, 先頭2文字をPTとしてはならない.
このときファイル名は, ユーザ名・ファイル局所名となる.

ファイル確保量……K(キロ)バイト単位で指定する. 1ファイルの確保量は200～1000Kバイトまでとする. 必要量の計算方法は後述する.

GEN ……一般ファイルであることを示し, これを指定する.

PS ……ファイル編成を示し, 順編成ファイル, 直接編成ファイルともにPSと指定する.

1・2 ¥ALLOCATEマクロ

	マクロ名	オペランド
¥	ALLOCATE	ファイル局所名, ファイル確保量, PS

パラメータの説明

ファイル局所名……#ALLOCATEコマンドの場合と同じ

ファイル確保量……　〃

PS ……　〃

＊1 ファイル局所名の意味は, 広報Vol. 8, No. 4を参照.

1・3　¥F・WORK

	マクロ名	オ　ペ　ラ　ン　ド
¥	F・WORK	ファイル定義名〔，TRK＝ファイル確保量〕

パラメータの説明

ファイル定義名……表2・2を参照.

ファイル確保量……トラック単位で指定する．直接編成ファイルの場合に必ず指定しなければならない．必要量の計算方法は後述する.

2. ファイルの入出力方法

補助記憶用ファイルの入出力方法として，順編成入出力と直接編成入出力があるが，どちらを使うかは目的に応じて使いわければよい．順編成入出力で取り扱うファイルを順編成ファイルといい，直接編成入出力で取り扱うファイルを直接編成ファイルという．いずれも確保するときは順編成ファイルとして確保する.

順編成入出力には書式つき（文字形式）および書式なし（バイナリ形式），直接編成入出力にはバイナリ形式の入出力の形式がある.

順編成入出力では処理できないことも直接編成入出力では可能な場合もあるが，直接編成入出力はシステム効率の点で順編成入出力に劣るので，できるだけ順編成入出力を利用していただきたい.

2・1　順編成ファイルの入出力方法

2・1・1　順編成入出力文

順編成ファイルの入出力は，カードリーダやラインプリンタに入出力する場合と同じように，READ/WRITE 文を使う．一般の入出力と異なるのは，ファイル参照番号（機番）が5，6，7以外であることと，書式なしで入出力する方法があることである.

書式なし入出力というのは計算機の内部表現のままで処理する方法で，バイナリ形式と呼ばれている.

表2・1　順編成入出力文

入　出　力　文	機　　能	形　　式
WRITE(u,f)　list	ファイルへ書式つきで書き出す.	文字形式
WRITE(u)　　list	ファイルへ書式なしで書き出す.	バイナリ形式
READ(u,f)　list	ファイルから書式つきで読み込む.	文字形式
READ(u)　　list	ファイルから書式なしで読み込む.	バイナリ形式

u：ファイル参照番号（1～8は整定数または整変数，9～99は整定数）

f：FORMATの文番号

ファイル参照番号5，6，7以外を使うときは，そのファイルを定義する必要がある．プログラムと物理的なファイルを論理的に結合するのがファイル定義名であり，ファイル参照番号とフ

ァイル定義名の対応は表２・２のように決まっている．したがって，ファイルにこのファイル定義名を与えてやればよい訳で，これを行なうのがファイル定義用のマクロである．

表２・２　ファイル参照番号とファイル定義名の対応

ファイル参照番号	ファイル定義名	ファイル参照番号	ファイル定義名
1	F01	7	SYSPCH *1
2	F02	8	F08
3	F03	9	F09
4	F04	10	F10
5	READ	⋮	⋮
6	SYSPRT	99	F99

なお，順編成ファイルはWRITE文を実行する度に，頭から順に書いていくので，読むときも最初のレコードから順番にWRITE文で書き出した長さに合わせて読んでいかないとエラーになる．

２・１・２　補助入出力文

READ/WRITE 文を使って，一応ファイルの利用ができるが，さらに次に説明する補助入出力文を組み合わせて利用すればなおいっそう便利である．たとえば，同一ジョブステップで何度もファイルを読み書きしたり，同じレコードを何度も読むことができる．

表２・３　補助入出力文

補助入出力文	機　　能
REWIND i	ファイルのCLOSEおよびWRITEモードのときは，ファイル終了記録（EOF：end of file）を書き，指針をファイルの始点におく． 最初のREWIND文は無視される．
BACKSPACE i	指針を１記録分だけ前にもどす．READ文と組み合わせて使う． 最初のBACKSPACE文は無視される．

i：ファイル参照番号

他に補助入出力文としてENDFILE文があるが，FORTRAN−Dではマルチファイル*2処理ができないので，使っても何も行なわない．

２・１・３　ファイルのOPEN/CLOSE

最初のREAD/WRITE文で，READ OPEN/WRITE OPENされ，REWIND文でCLOSE

＊１　FACOM 230 M−V/VI FORTRAN文法編，FACOM 230 M−V FORTRAN解説編(II)では‘PUNCH’になっているが，センターでは運用上，‘SYSPCH’におきかえている．

＊２　FORTRAN−Hでは可能である．

される．また，実行が終了するとCLOSEされる．

READ　OPENされた状態をREADモード，WRITE　OPENされた状態をWRITEモードという．WRITEモードのときCLOSEするとEOFが書かれる．

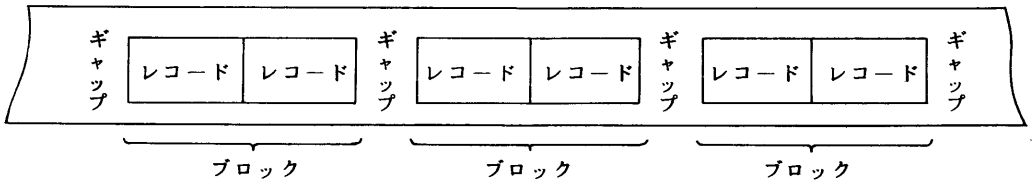
READモードでのWRITE文あるいはWRITEモードでのREAD文は使えない．すなわち同一ジョブステップで同時にファイルの読み書きを行なう場合，WRITE文の後すぐにREAD文がきたり，あるいはREAD文の後すぐにWRITE文がきたりしてはならない．必ず，REWINDした後READ/WRITEしないとエラーになる．

したがって，REWIND文がくるまでは，READ文とBACKSPACE文の組み合わせか，WRITE文のみしか使えない．

2・1・4　媒体上のデータ形式

（1）　レコードとブロック

データ処理時，対象とする情報の単位をレコードという．一方，主記憶と2次記憶（大記憶，磁気テープ上のファイル）間の転送の対象となる記録単位をブロックといい，通常1つ以上のレコードを含み，ギャップにはさまれている．



いくつかのレコードをまとめてブロックにすることをブロック化（blocking）といい，1ブロックのレコードの個数をブロック化率（blocking factor）という．

ブロック化のねらいは，記憶媒体上のギャップを減少させることによって，相対的にデータの転送速度と記録密度を高めることにある．また，ブロック化することによって，ファイルの入出力操作も減少するので，プログラムのファイル処理効率も高まる．

1ブロック転送することをアクセスという．アクセス回数を減らすには，ブロック化率を大きくして，ファイル内のブロック数を少なくすればよい訳である．

しかし，FORTRAN—Dが扱えるファイルのレコード形式は，固定長形式で非ブロック化レコード[*1]しか扱えない．固定長形式非ブロック化レコードというのは，ファイルを通してすべてのレコード長が一定であり，ブロックは1個のレコードからなっているものをいう．

そのため，FORTRAN—Dでは，ある程度まとめて処理するように考慮しなければならない．

また，ファイルを入出力する場合，入出力用のバッファ（作業用の記憶領域）が必要である．バッファはブロック長と同じか大きくとらなければならないので，ブロック長を大きくすれば，それだけプログラムも大きくなる．FORTRAN—Dの場合，バッファは最大1600語までとれる．

---

*1　FORTRAN—Hではブロック化レコードが扱える．

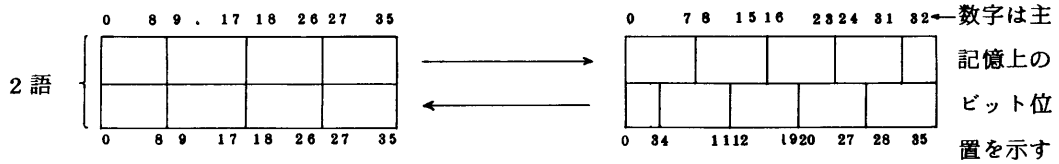
レコード長，ブロック長はマクロのパラメータで指定し，単位はバイトである．語からバイトへの変換は転送モードで異なる．転送モードというのは，ファイル媒体から主記憶，または主記憶からファイル媒体へ情報を転送する場合のモードで，８ビットモードと９ビットモードがある．

- ８ビットモード……（ｎ語×９／２）バイト　主記憶の８ビットを１バイトとして転送する．バイナリデータの転送に有効である．
- ９ビットモード……（ｎ語×４）バイト　主記憶の９ビットを１バイトとして転送する．バイナリデータは扱えない．

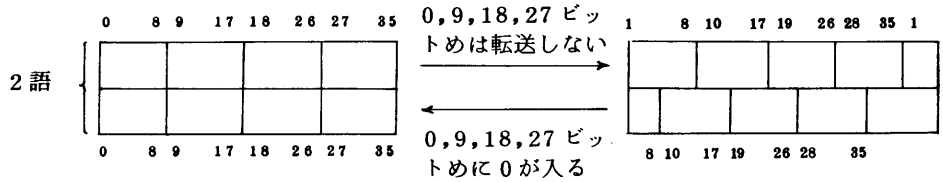
８ビットモードと９ビットモードの違いを示すと次のようになる．

主　記　憶　　ファイル媒体（主記憶イメージ）

８ビットモードの転送



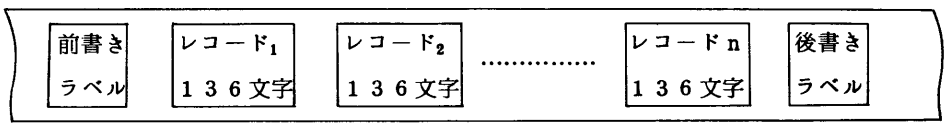
９ビットモードの転送



FORTRAN—Dの場合，文字形式，バイナリ形式ともに８ビットモードで転送される．

（２）文字形式

文字形式の１レコード（１ブロック）は１３６文字（=１５３バイト，３４語）で扱われ，装置上は次の様に配置されている．このレコード長を変更することはできず，指定しても無視される．



FORMATの１行分が１レコードとして書き出されるので，１行を１３６文字とするのが望ましいが，そうでない場合は次のような処置がとられる．

1. １３６文字以内のときは，残りは空白がつめられる．
2. １３６文字を越えたときは，１３６文字分だけ書き出して，越えた分は切捨てられる．

１つのWRITE文で書き出される長さを１記録というが，これは１つ以上のブロックから構成されている．

並びとFORMATで決められる全文字数がｎ，１行がℓのとき

Ｂ＝〔ｎ／ℓ〕，　Ｒ＝ｎ－ℓ×Ｂ　とおき

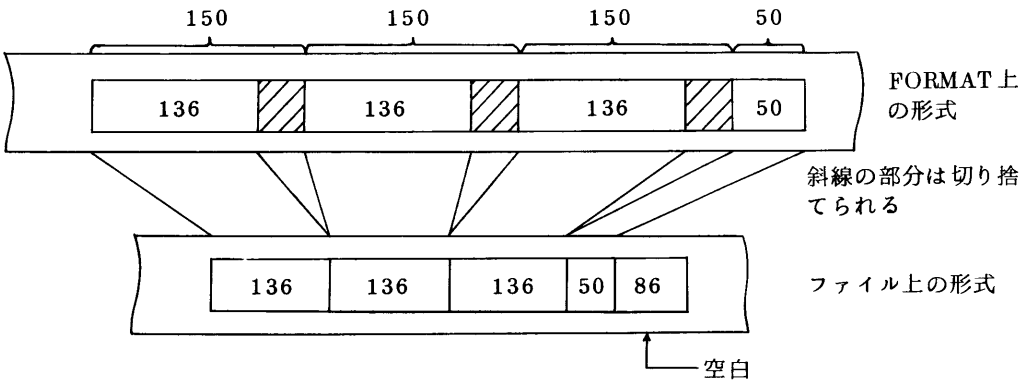
Ｒ＝０ならＢ個のブロックが，Ｒ≠０ならＢ＋１個のブロックがとられ，最後のブロックの有効データはＲ文字で，残り１３６－Ｒ文字は空白となる．

例
```
      DIMENSION A(100)
        :
      WRITE(1,100)A
100   FORMAT(30I5)
        :
```

全文字数が５００，１行の文字数が１５０であるから，

Ｂ＝〔５００／１５０〕＝３　Ｒ＝５００－１５０×３＝５０

したがって，４個のブロックガとられ，最後のブロックは５０文字で，残り８６文字は空白となる．

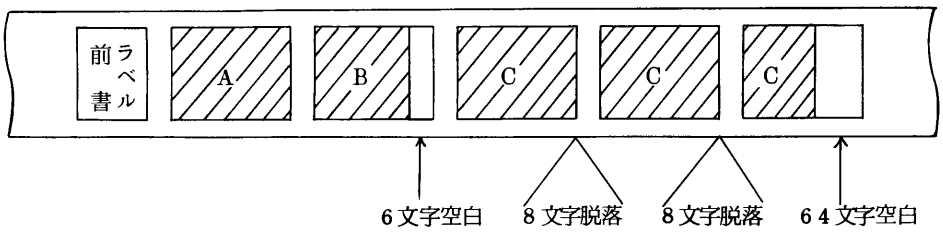


例
```
      DIMENSION A(10),B(10),C(30)
        :
      WRITE (1,100)A
      WRITE (1,200)B
      WRITE (1,300)C
100   FORMAT(9F14.7,F10.5)
200   FORMAT(10F13.6)
300   FORMAT(12F12.5)
        :
      STOP
      END
```

FORMAT上では1行144 文字分になっているが

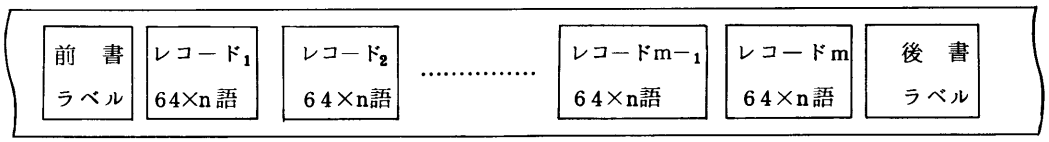
……ファイルに書き出すときは１行１３６文字となり

８文字は転送されない．



（３）　バイナリ形式

バイナリ形式の１レコード（１ブロック）は２８８×ｎバイト（＝６４×ｎ語）で扱われ，装置上は次の様に配置されている．



ｎの値は任意に指定できるが，標準的にはｎ＝４すなわち１１５２バイト（＝２５６語）が１レコードとなっている．ただし，１≦ｎ≦２５．

また，ｎを４より大きく指定するとき，たとえば，ｎ＝１０とするときは，マクロのパラメータRCDSIZE，BLKSIZEを２８８０バイトとするとともに，OPTION文でBUFFER＝１０と指定しなければならない．

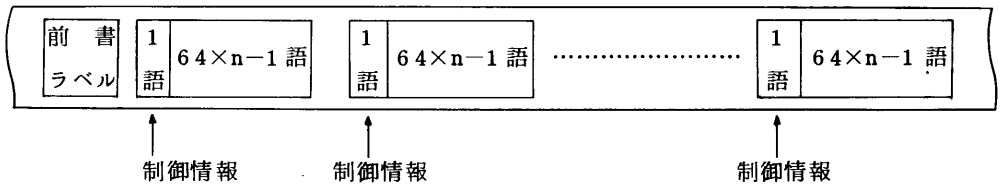
OPTION文は主プログラムの先頭に定義すればよいが以下に示すようにBUFFER=ｎは第２パラメータ以下に定義しなければならない．

例

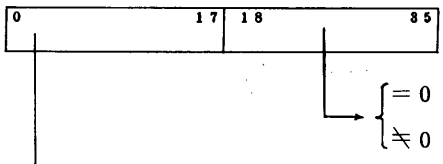
```
      OPTION   LIST, BUFFER=n
C     * MAIN *
      DIMENSION A(200)
        :
      WRITE(1)A
        :
```

これは転送のためのバッファを確保するもので６４×ｎ語分とられる．指定しなければｎ＝４となっている．バッファの語数はブロック長と同じかあるいは大きくとっていなければならない．

なお，バイナリ形式の場合，ブロックの最初の１語は制御情報として使われるので，実際に書けるのは６４×ｎ－１語となる．



制御情報の内容は次の様になっている.



1つの WRITE 文で書き出す長さを1記録というが，その語数がレコード長[*1]と異なるときは，次の様な処置がとられる.

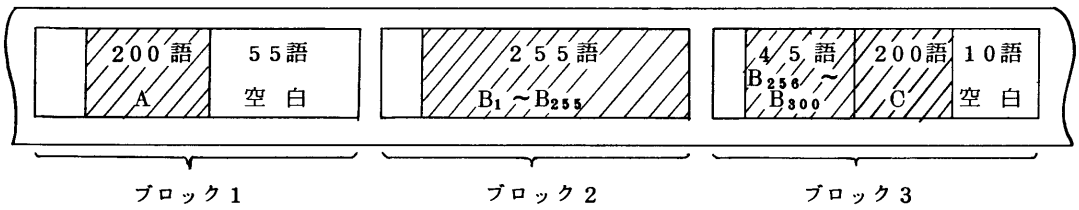
1. レコード長より小さいとき，残りは空白がとられる.
2. レコード長を越えたとき，全語数をnとし，

   B＝〔n／レコード長〕，　R＝n－レコード長×B　とおき，

   R＝0ならB個のブロックが，R≒0ならB＋1個のブロックがとられ，最後のブロックはR語で，残りレコード長－R語は空白がとられる.

例1　レコード長を標準の256語にしたとき

```
DIMENSION A(200),B(300),C(200)
  ⋮
WRITE(8)A
WRITE(8)B,C
  ⋮
```

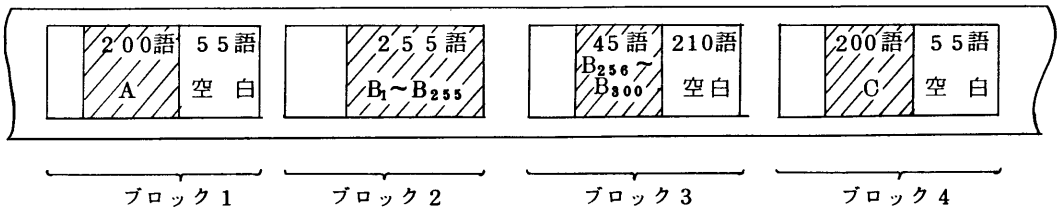


例2　例1の配列を次の様に書いたとき（レコード長＝256語）

```
DIMENSION A(200),B(300),C(200)
  ⋮
WRITE(8)A
WRITE(8)B
WRITE(8)C
  ⋮
```

---

*1　この場合のレコード長は有効レコード長をさす.



例1，2からわかるように，1記録の長さを有効レコード長の整数倍に近づけ，空白をなくすようにした方が記録密度が高く，アクセス回数も少なくなる．

文字形式とバイナリ形式を比較すると，プログラム上ではFORMATの有無の違いだけであるが媒体上のデータ形式からわかるように，バイナリ形式の方が1ブロックに多くの情報が書ける．

したがって，バイナリ形式で書き出すほうがファイルの節約になる．また，数値データを文字形式で読み書きすると，2進データから10進データへの変換またはその逆の変換で丸目誤差が生じるので，なるべく文字形式は使わない方がよい．

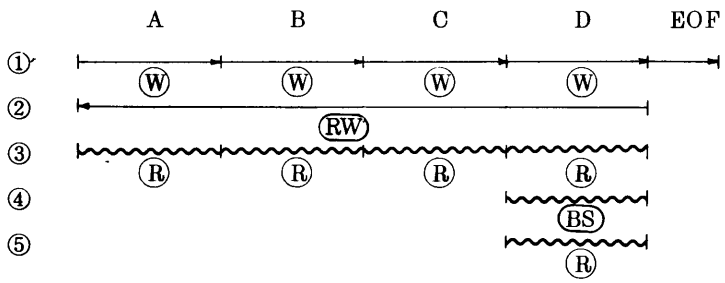
なお，これらを混合して使うことはできない．

2・1・5　順編成入出力文と補助入出力文の組み合わせ例

例1

```
      DIMENSION A(100), B(100), C(100), D(100)
      DIMENSION K(100), L(100), M(100), N(100)
        ⋮
①   WRITE(1)A
    WRITE(1)B
    WRITE(1)C
    WRITE(1)D
②   REWIND  1
③   READ(1) K
    READ(1) L
    READ(1) M
    READ(1) N
④   BACKSPACE 1
⑤   READ(1)A
        ⋮
```

①は配列A，B，C，Dの値をファイル参照番号1で示されるファイルに書き出す．②は指針をファイルの先頭にもどす．③は①で書き出した値を配列K，L，M，Nに読み込む．④は1記録分だけ前にもどす．すなわち配列Dの値の直前に位置する．⑤は配列Dの値を配列Aに読み込む．以上の事を図示すると次の様になる．



②でREWINDせずにREADしたり，⑤の後でさらにREADを出したりするとエラーになる．またエラーにはならないが，結果が保障されない場合などがある．

2・2　直接編成ファイルの入出力方法

直接編成ファイルと順編成ファイルの大きな違いは，次の様なことである．順編成ファイルの場合，ファイル内のあるレコードを読み込むには，最初のレコードより目的のレコードまで，読みとばしを行なう以外に方法がないし，また，途中のレコードに書き込むことはできない．これに対して直接編成ファイルは任意のレコードを制御変数を使って，ダイレクトに読み書きができる．

2・2・1　直接アクセス入出力文

直接アクセス入出力文	機　　能	備　　考
CALL DFILE(u, b, i)	直接編成ファイルを使用するための初期設定を行なう．	bはOPTION文のBUFFER指定より大きくしてはならない．
DWRITE(u, r) list	ファイルにrで示されるレコード位置を始めとして，listに従って書き出す．	バイナリ形式
DREAD(u, r) list	ファイルからrで示されるレコード位置を始めとして，listに従って読み込む．	バイナリ形式

u：ファイル参照番号（1～4，8の指定しかできない）

b：最大レコード長（単位は語）

i：レコード位置を制御する整数型変数．READ/WRITE後，次のブロック位置が設定される．

r：READ/WRITEするレコード位置を示す整数型変数，あるいは整数型定数．

rにiと異なる変数あるいは定数を指定した場合，レコード位置はiに設定されるので注意しなければならない．

例

```
      DIMENSION A(200), B(400)
      CALL DFILE(4, 200, I)
        ⋮
      J=1
①     DWRITE(4, J)A
②     DWRITE(4, J)B
        ⋮
```

①はJ＝1であるから，1番めのブロックから配列Aの値を書き，Iに2が設定される．

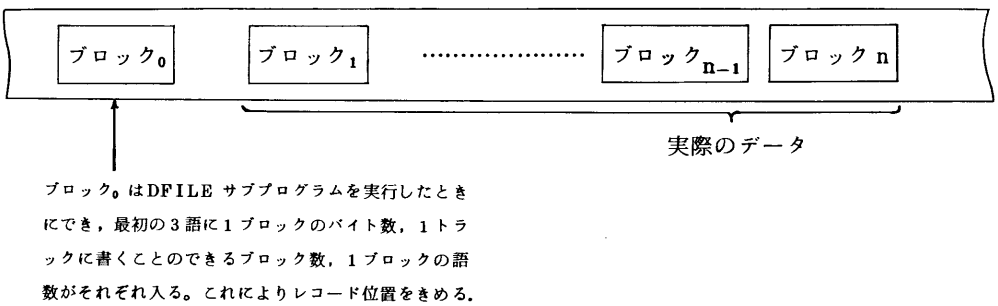
②もJ＝1であるから，1番めのブロックから配列Bの値を書き，Iに3が設定される．

2・2・2　媒体上のデータ形式

順編成ファイルと同じくレコード形式は，固定長形式で非ブロック化レコードしか扱えない．

1レコードの大きさはDFILE文で指定した最大レコード長がとられるので，マクロのパラメータで指定しても無視される．

装置上は次の様に配置され，最初のブロックはレコードを制御する情報が書き出されるので実際のデータは2番め以降のブロックに入る．



ブロック0はDFILE サブプログラムを実行したときにでき，最初の3語に1ブロックのバイト数，1トラックに書くことのできるブロック数，1ブロックの語数がそれぞれ入る。これによりレコード位置をきめる．

1記録の長さがレコード長と異なる時，次の様な処置がとられる．

1. レコード長より小さいとき，残りは空白がとられる．
2. レコード長を越えたとき，全語数をnとし，

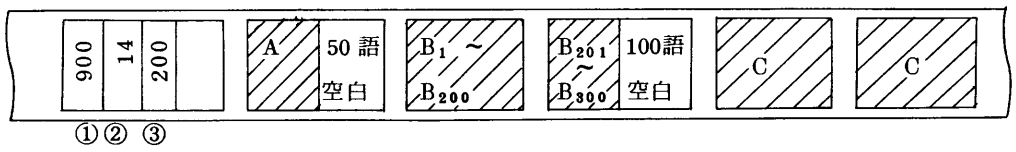
   B＝〔n／レコード長〕，R＝n－レコード長×B　とおき

   R＝0ならB個のブロックが，R≠0ならB＋1個のブロックがとられ，最後のブロックはR語で，残りレコード長－R語は空白がとられる．

   なお，1ブロックの大きさを256語より大きくする時は順編成ファイルと同様に，OPTION文のBUFFERパラメータを指定しなければならない．

例

```
      DIMENSION A(150), B(300), C(400)
      CALL DFILE(8, 200, I)
        :
      I=1
      DWRITE(8, I)A          ……Iに2が設定される.
      DWRITE(8, I)B          ……Iに4が設定される.
      DWRITE(8, I)C          ……Iに6が設定される.
        :
```



①　1ブロックのバイト数　　②　1トラックのブロック数　　③　1ブロックの語数

3　ファイルに書ける記憶容量

3・1　専用ファイルの確保量の求め方

専用ファイルが確保される記憶媒体は磁気ディスクパックでF477KとF472Kがある．これらの媒体の物理的な単位としてトラックがあり，1トラックの記憶容量はそれぞれ13030バイト，7294バイトとなっている．ファイル確保時に指定するバイト単位の量は，実際にはトラック単位の量に換算されてとられている．したがって，1トラックに書けるブロック数を計算して必要なトラック数を求め，それをバイト単位の量に換算すれば確実な必要量が求まる．

しかし，利用者にはファイルがF477KとF472Kのどちらにとられるのかわからないので，面倒だが，2通りの計算をして大きい方の値をとっていただきたい．

ブロックの大きさおよび1トラックに書けるブロック数の計算式は次の通りである．

（1）　ブロックの大きさ

ブロックの大きさはプログラムまたはマクロのBLKSIZEの指定で決まるが，媒体上に書き出されるときは，ブロック間のギャップやブロックに関する情報などが付加されるので，実際のブロックの大きさはBLKSIZE で指定した値より大きくなる．

また，ブロックはトラックにまたがって書かれることはないので，利用者からみた1トラックの有効な記憶容量は少なくなる．

装置名	1トラックの記憶容量	ブロックの大きさ	
		最終ブロック以外	最終ブロック
F477K	バイト 13030	135+DL	135+DL
F472K	バイト 7294	$101+\left(\frac{534\times DL}{512}\right)$ ＊1	DL

DL：プログラムおよびBLKSIZEで指定したバイト数

（2）　1トラックに書けるブロック数

・最終トラック以外のトラック

$$\text{ブロック数}=\left[\frac{\text{1トラックの記憶容量（バイト）}-\text{最終ブロックの大きさ（バイト）}}{\text{最終ブロック以外のブロックの大きさ（バイト）}}\right]_{\text{＊2}}+1$$

・最終トラック

$$\text{ブロック数}=\left[\frac{\text{1トラックの記憶容量（バイト）}-1}{\text{最終ブロック以外のブロックの大きさ（バイト）}}\right]_{\text{＊2}}$$

また，上記の方法はかなり面倒なので，次の様な計算式を使ってもよい．

ただし，この場合はかなり大まかな値なので100K（キロ）バイト単位で切り上げていただきたい．

ギャップ等の大きさを135バイトとして

1ブロックの大きさ＝DL＋135（バイト）　　DL：BLKSIZEで指定したバイト数

必要量＝ブロック数×1ブロックの大きさ

以上あげた3通りの方法による例を次に示す．

例　ブロック長を500語（2250バイト）で60ブロック書き出す場合の必要量はいくらか．

1. 単にブロック数から求めた場合

1ブロックの大きさ＝2250+135=2385

必要量＝60×2385=143100

したがって確保量を200Kバイトとする．

2. F477Kの計算から求めた場合

1ブロックの大きさ=2250+135=2385

最終トラック以外のブロック数＝$\left[\frac{13030-2385}{2385}\right]+1=5$

＊1　切り上げ演算を行なう．

＊2　小数点以下を切り捨てる．

$$\text{最終トラックのブロック数}=\left[\frac{13030-1}{2385}\right]=5$$

必要なトラック数は，12トラックでバイト単位に換算すると

必要量＝12×13030＝156360

したがって確保量を200Kバイトとする．

3. F472Kの計算式から求めた場合

$$\text{最終ブロック以外の大きさ}=101+\left[\frac{534\times2250}{512}\right]=2448$$

$$\text{最終ブロックの大きさ}=2250$$

$$\text{最終トラック以外のブロック数}=\left[\frac{7294-2250}{2448}\right]+1=3$$

$$\text{最終トラックのブロック数}=\left[\frac{7294-1}{2448}\right]=2$$

必要なトラック数は21トラックでバイト単位に換算すると

必要量＝21×7294＝153174

したがって確保量を200Kバイトとする．

3・2　作業用ファイルの確保量の求め方

作業用ファイルの記憶媒体は磁気ディスクパックでF477Kにとられる．順編成ファイルの場合は，システムが必要に応じて実行時に確保してくれるので，利用者が指定する必要はない．最大510トラックまで確保される．

直接編成ファイルの場合は，あらかじめ必要量を計算して指定しなければならない．単位はトラックで510トラックまで指定できる．

例　ブロック長400語（1800バイト）で50ブロック書き出したい場合の必要量はいくらか．

ブロックの大きさ＝1800＋135＝1935

$$\text{最終トラック以外のブロック数}=\left[\frac{13030-1935}{1935}\right]+1=6$$

$$\text{最終トラックのブロック数}=\left[\frac{13030-1}{1935}\right]=6$$

したがって，50ブロック書き出すには，9トラック必要である．

4. コントロールカードの入れ方

補助記憶用ファイルを利用するためのコントロールカードとしては，¥PSFILEと¥F・WORKがある．前者は利用者自身の専用ファイルを利用するとき，後者は作業用ファイルを利用するときに使う．

パラメータの説明については，利用の手引ファイル編を参照されたい．

4・1　¥PSFILEを使う場合

例1　ブロック長を標準の256語（1152バイト）とするとき

```
¥NO
¥USER
¥QJOB
¥FORTRAND
[ソースプログラム]
¥LIEDRUND
[デ　ー　タ]
¥PSFILE　ファイル定義名，ファイル名
¥JEND
```

例2　ブロック長を300語（1350バイト）とするとき

```
¥NO
¥USER
¥QJOB
¥FORTRAND
[ソースプログラム]
¥LIEDRUND
[デ　ー　タ]
¥PSFILE　　ファイル定義名，ファイル名，FCBPRM=YES，RCDSIZE=1350，／
                BLKSIZE=1350                          ↑72桁
¥JEND
```

4・2　¥F・WORKを使う場合

例1　順編成ファイルでブロック長を標準の256語（1152バイト）とするとき

```
¥NO
¥USER
¥QJOB
¥FORTRAND
[ソースプログラム]
¥LIEDRUND
```

```
データ
¥F・WORK     ファイル定義名
¥JEND
```

例2　順編成ファイルでブロック長を400語（1800バイト）とするとき

```
¥NO
¥USER
¥QJOB
¥FORTRAND
ソースプログラム
¥LIEDRUND
データ
¥F・WORK     ファイル定義名，FCBPRM=YES，RCDSIZE=1800，
             BLKSIZE=1800
¥JEND
```

例3　直接編成ファイルでブロック長400語50ブロック書き出したいとき

```
¥NO
¥USER
¥QJOB
¥FORTRAND
ソースプログラム
¥LIEDRUND
データ
¥F・WORK     ファイル定義名，TRK=9
¥JEND
```