OUKA
The University of Osaka
Institutional Knowledge Archive

Title	ACOSのエラーメッセージ入門
Author(s)	塩野, 充
Citation	大阪大学大型計算機センターニュース. 1979, 35, p. 81-88
Version Type	VoR
URL	https://hdl.handle.net/11094/65432
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

ＡＣＯＳのエラーメッセージ入門

大阪大学工学部　塩　野　　充

〝省エネ〟という言葉で明け暮れた暑い夏も過ぎ，背広の腕をチョン切っただけの〝省エネルック〟　の在庫の山はどうなったのかは知らないが，いよいよ秋たけなわ，行楽のシーズンではある。けだし，４回生諸君にとってはこれからが卒業研究に本格的に取り組まなくてはならない季節であろう。ところで，近年の理工系（のみならず最近は経済等の文科系の一部についても言えることだが）の卒業研究で，計算機に全く無縁でＦＯＲＴＲＡＮなど見たことも聞いたこともないままに終る，というテーマはかなり稀なのではなかろうか。殆んどの場合，何らかの形で計算機とかかわりを持つことになるのではないかと思う。そこで本稿では４回生をはじめとする計算機利用の初心者（かく言う筆者も初心者とさして変りはなく，エラそうな口をきける者ではないが）が，当計算機センターのＡＣＯＳシステムを用いてジョブを行う場合に参考となるアドバイスを述べることにする。なお，ここでは最も基本的な話なのでジョブはバッチジョブ，使用言語はＦＯＲＴＲＡＮとする。

ところで，以下の話は初心者向けの話なので，ベテランの方や，あるいはその逆にＦＯＲＴＲＡＮを全然知らない人が読まれてもアクビが出るだけであるから，他のもっと面白くてタメになる本でも読まれることをおすすめする。

普通，ソフトウエアの作成，という程大げさなものでなくても，とにかく一件の計算をやりとげるには次のような一連のプロシージャを必要とする。まず問題が与えられ，それを解くためのアルゴリズムを考え，それを基にフローチャートを書く。ここで声を大にして言わなければならないのは，ここらあたりのステップを丁寧にしておくことが肝心であるということである。問題解決のアルゴリズムをきっちりと文書化（ｄｏｃｕｍｅｎｔａｔｉｏｎ）し，フローチャートもきれいに書いておく。ここで手を抜くとあとあとわけの分らないトラブルに悩まされ，いわゆる泥沼に落ち込んでついにはそのプログラムを廃棄処分にせざるをえないという，誠に情ない結末になる可能性がある。ところが困ったことに初心者程ここのステップを軽視し，問題を見ていきなりコーディングを始める人が多い。もっとも，例えば三角形の３つの辺の長さが与えられて面積を求めるというような簡単な演習問題であればそれでもよいかも知れないが，数千行にも及ぶような本格的なプログラムになるとそうはいかない。

ここのステップはソフトウェア作成上，最も重要なステップであると言える。最も重要な所で手抜き工事をしたのではロクなものはできない。さて話を元に戻して，次にフローチャートを元に

してコーディング用紙にプログラムを書き，それをカードにパンチしてソースプログラムデックを作り，ＪＣＬ（ジョブ制御言語，ＡＣＯＳでは全て＄又は¥で始まるが，ちなみに触れておくとおかしなことだが計算機では＄と¥は全く同じ記号として扱われる。パンチ機を見るとＩＢＭ製では＄となっているキーが国産のＪＵＫＩ製では¥となっており，計算機の記号に関する限り円相場というのは常に１ドル＝１円らしい？）カードを付け加え，データカードがあればそれも付け加えてカードリーダーから入力し，しばしの後，ＩＤカードでデマンドしてラインプリンタからゴトンと掃き出される出力結果を受け取って帰る，のであるが，実際には仲々そう簡単にはハッピーエンドで帰らせてはくれない。初歩のうちは種々のエラーが続々と現われてプログラマを悩ませる。そこで本稿では，出力用紙に印字される各種の（エラー）メッセージについてその意味と対策を説明することにしよう。このようなことはマニュアル（メーカーの作成した説明書）を見れば全て書いてあることだが，とにかくマニュアルというものは厳密さを第一にして書かれているので，初心者が“何回”読んでも“難解”なだけである。それゆえ，ここでは筆者が及ばずながらマニュアルの“コンパイラ”となって，マニュアルに書いてあることないことを，初心者向けにくだいた形で（くだきすぎるきらいもあるが）解説させてもらおう。しかし，全てのメッセージについて説明することはスペース的に，と言うよりは筆者の技量的に不可能であり，又その必要もないと思うので，出現しやすいメーセージに絞って説明することにする。

まず，メッセージの種類は４つに分けられる。印字される順番に並べると，システム（アボート）メッセージ，コンパイラメッセージ，ローダメッセージ，実行時エラーメッセージである。出力用紙の中でそれぞれの印字される箇所は次のとおりである。システムメッセージは表紙をめくるとまずＪＣＬがズラリと印字されているが，その後のＡＣＴＹ（アクティビティ）の中に印字される。一般的なジョブではＡＣＴＹ－０１がコンパイル過程，ＡＣＴＹ－０２が実行過程となる。いずれも正常終了したときは，ＮＯＲＭＡＬ　ＴＥＲＭＩＮＡＴＩＯＮとのみ印字されるが，アボート（Ａｂｏｒｔ，中断）されたときはアボートコードとアボートメッセージが印字される。コンパイラメッセージは初心者には最もなじみの深いもので，文法エラーや警告のあるときソースプログラムリストの中や終りに印字される。ローダメッセージは，ソースプログラムリストの終りの次の頁に印字され，種々のレポートメッセージと共に，ローダにおけるエラーがあればエラーメッセージが印字される。実行時エラーメッセージは実行が開始された頁以降に印字される。

以上が各メッセージの出力される箇所であるが，次に各メッセージの中で出現しやすいものを挙げて説明する。初心者の目につきやすい順としてまずコンパイラメッセージ，次に実行時エラーメッセージ，あとローダメッセージ，システムメッセージの順で述べよう。

まず，最も基本となるのはコンパイラメッセージである。翻訳時のエラー番号は１から１５０７

まであり（番号は飛んでいるのもあり，１５０７種類エラーがあるわけではない），それらはW（Warning，ウォーニングであり，ワーニングと読むのは間違いである。警告の意味である），F（Fatal Error，フェイタルエラー，致命的誤り），T（翻訳中断）の３種類のエラーレベルのいずれかに属する。印字されるときは＊＊＊＊＊の後にWかFかTが印字される。Tが出ることは少なく，たいていFかWであり，Fの場合は実行されないが，Wの場合は実行される。Fは初歩的な文法ミスやパンチミスが殆んどの原因であり，メッセージの英文を読めば分る場合が多い。Wはそのまま放っておいても差支えない場合が多いが，気になる人は修正すればよい。最もよく現われるWとして次の３つがある。まず，W７で，

＊＊＊＊＊W　　7 MEMORY EXPANDED.……

と印字される。これをたいていの人は「セブンメモリーエクスパンデッド（７つのメモリーが拡張された）」と読んでしまい，何のことだろうと思うらしい。これはWと７の間が５文字分も空いており，７とメッセージの間が１文字分しか空いていないのでそう読むのも無理のないことである。ここはWと７をもっと詰めるか，７をゼロサプレスしないで，０００７とするか，７の後に／や；の区切り記号を入れるべきであろう。何はともあれ，このW７はコンパイラの使う主記憶が標準の大きさ（３３K語）を越えたので自動的に拡張されたということで，放っておいて差支えない。気になる人は¥FORTRANカードの前に¥LIMITSカードで必要なメモリサイズ（例えば，３８K語ぐらい）を指定すればW７は出なくなる。W７は出ないよりもむしろ出る場合の方が多く，標準の３３K語というのは少し小さすぎるのではないかと思う。

次に多いのはW１４７０であり，

EQUALITY OR NON-EQUALITY COMPARISON MAY NOT BE MEANINGFUL IN LOGICAL IF EXPRESSIONS.

というメッセージが出る。これは実数どおしの．EQ．や．NE．による比較を論理ＩＦ文の中でやらない方が良いということである。これは異なる数式を経て求められる２つの実数が数学的には完全に一致しても計算機による演算では端数の切り捨て等の誤差の関係で完全には一致しない場合が多いからである。例えば，これは電卓でも確かめられるが，

A＝１０．０／３．０＊３．０

とした場合，Aは数学的には１０．０であるが，計算機では９．９９……９となる。こういうことは起り得ないという確信のある場合以外は実数では．EQ．や．NE．を使わずに，．LT．やGT．を使った方が良い。例えばA．EQ．Bの代りに，｜A－B｜．LT．ε（εは１０のマイナス何乗というような小さい数で，A＝Bと見なしうる誤差の最大値である）とすれば良い。整数どうしの場合は全く問題ない。

もう一つ多いのがW１４５７である。

□ {名前}MAY NOT BE REDEFINED IN CALL OR ABNORMAL FUNCTION.

というメッセージが出る。 □ の中はいろんな語句がその場々々に応じて入り，とりわけ多いのは，DO LOOP INDEXである。これはDOの制御変数がそのDOループの中にあるサブルーチンや関数の引用文の引数になってはいけないということである。例えば，

```
      DO  1  I=1, 5
      CALL  SUBR(X, I)
      ……
    1 CONTINUE
```

とした場合，サブルーチンSUBRの引数にDOの制御変数Iが入っており，もしSUBRの中でIに処理が及んで変化し，例えばI＝１０００となって戻ってくると，IのDOループがメチャクチャになってしまうからである。これを防ぐには，

```
      DO  1  I=1, 5
      II=I
      CALL  SUBR(X, II)
      ……
    1 CONTINUE
```

とIをIIに置き換えてやればW１４５７は出なくなる。 □ の中に入る語句は他にADJUSTABLE DIMENSIONその他であるが，同様な方法で防ぐことができる。以上が大体コンパイラメッセージで出やすいものの代表である。

次に，実行時エラーメッセージに移ろう。これはエラー番号1から１０７まであり，エラーレベルはAとCの２種類あり，Aは致命的エラーでこれが起こるとジョブは強制終了となる。Cは警告で実行は続けられる。実行時エラーメッセージは＜＊＞の３つの記号を繰り返し並べたものものしい模様に上下をはさまれて印刷される。このうち見るべき部分は左の方で，右の方は絶対番地等が印字されており，素人にはあまり参考にならない。印字例を示すと，

```
<*><*><*><*><*>……
ERROR#０３４;TRACE OF……
 CALLING          ID
 ROUTINE           #
 .FEOF.            47
 .FRD             461
 ABC               20
```

・・・・・・　　9

メッセージ

＜＊＞＜＊＞＜＊＞＜＊＞＜＊＞……

のようになる。ここで#０３４はエラー番号であり，．FEOF．と．FRDはそのエラーを検出した機能名であり出ないときもある。ABCはユーザのサブルーチン名とする。２０はABCの第２０行でエラーが発生したことを表わす。６つのドット．．．．．．は常にユーザのメインプログラムを表わし，９はその第９行でエラーが発生したことを表わす。メッセージの部分にはエラー#０３４の説明の英文が印字される。エラーレベル（AかCか）は印字されない。実行時エラーメッセージは常にこの＜＊＞が並んだ模様ではさまれた形式で表示されるが，例外として，オーバーフローやアンダーフロー，ゼロで割算をしたときなどに，FORTRANフォールトプロセッサという機能が作動して次のような簡単なメッセージを印字する。例えば実数型のオーバーフローでは，

ERROR #69 EXP OVERFLO AT LOCATION ○○○○ となる。

ここで○○○○はアドレスであり，エラーが発生したプログラム名，行番号は表示されず，素人にとってはあまり親切なメッセージではない。これを他のエラーメッセージと同様に＜＊＞の並んだ模様ではさまれ，プログラム名や行番号をも印字させる方法がある。コンパイラオプションにFLTCHK（フロートチェック）を指定すればよい。すなわち¥FORTRANカードの16カラムには普通，LSTIN（ソースプログラムリストを出力させるオプション）ぐらいを打つだけであるが，その後に続けてカンマを打ち，FLTCHKと打つだけでよい。これともう一つ，エラー検出のためのコンパイラオプションに配列の添字検査のためのSUBCHK（サブチェック）があるが，これは後のシステムメッセージと関連させて説明しよう。以上述べた実行時エラーメッセージで，出現しやすいものを挙げると，まず，

#０３４ END OF FILE READING FILE CODE fc……

がある。これは例えばREAD文でデータカードを読み込むときに，READする量が実際のデータカードより多く，データカードを全部読み込んでしまっているのに更に読もうとした場合や，FORMAT文が不適当な場合に発生する。これを防ぐには，例えば，

READ（５，１００，END＝９９）……

とすればよい。こうしておくとデータを全部読み終えた後になお読もうとしたときには文番号99へ飛ぶことになり，エラーにはならずに済む。

#０６７～#０７１は前述のFLTCHKのメッセージである。次の#０８７も多い。

#０８７ SPACE／CORE OBTAINED FOR LOG. FILE CODE #fc

これは後で述べる“配列破り”が発生したときに，プログラムがこわれるために出現することが多い。SUBCHKで原因究明されると共に解決する場合が多い。＃１０４はそのSUBCHKのメッセージである。

次にローダメッセージについて述べよう。ローダのエラーメッセージにもエラーレベルがある。Fは致命的エラーで実行は中断され，エラーメッセージが印字されると共にシステムメッセージの箇所にローダアボートコード（Ｌ１～Ｌ４のいずれか）が印字される。Nは非致命的エラーで実行は続行される。現われやすいものとしては次の２つがある。

＊＊＊ FATAL ERROR ＊ LOAD TABLE AND PROGRAM OVERLAP

これは¥ＬＩＭＩＴＳカードの第３パラメータ（－２Ｋ等）を省略すると発生する。（以前は省略できたのだが）。このエラーが起こるとローダアボートコードＬ３により実行は中断される。もう１つは，

＊＊＊ NON FATAL ERROR ＊ MISSING ROUTINE ○○○, ……

これは○○○というサブルーチン名がＣＡＬＬ文で使われているのに，その○○○というサブルーチンが存在しない場合に発生する。サブルーチンのカードデックを入れ忘れたり，多いのはセンターやメーカーのサブルーチンライブラリ（ＳＳＬ）を使おうとする際に，¥ LIBRARY の JCL カードを入れない場合である。ＳＳＬを使うときは，

１カラム	８カラム	１６カラム
¥	LIBRARY	CH，MH

なるカードを¥ＦＯＲＴＲＡＮカードの前に入れなければならない。

このエラーはレベルＮになっていて実行は続けられるが，プログラムの流れが実際にその○○○なるサブルーチンへ移った時点でローダアボートコードＬ１により実行は中断される。

最後にシステムメッセージについて説明する。ここには英数字や記号からなるシステムアボートコード（先程のローダアボートコードＬ１～Ｌ４と同様）と，英文のエラーメッセージが印字される。勿論，アボート以外のメッセージも出力する。まず多いアボートは，Ｉ８－RUN TIME EXHAUSTED

で，これは処理時間超過で，¥ＬＩＭＩＴＳカードを直すか，ジョブクラスを上げる。

０＞－OUTPUT LIMIT EXCEEDED

これは出力レコード数超過で，たいていの場合ＬＰ用紙の頁数オーバーで，やはり¥ＬＩＭＩＴＳカードを直すか，ジョブクラスを上げる。次に，

Ｆ０－MEMORY ADDRESS FAULT

および，

F7−UNDEFINED OP FAULT

の2つはたいていの場合，配列破りが原因である。配列破りとは，配列の添字が配列宣言したときの範囲から外へ出ることをいう。例えば，

```
      DIMENSION A(10)
      DO 1 I=1,15
      A(I)=0.0
    1 CONTINUE
```

などとすると，配列AはA(10)までしか宣言していないのにA(11)，A(12)，……とアクセスしてゆくことになる。このとき，主記憶上でA(10)の隣りのA(11)，A(12)，……に相当する部分にプログラムが入っていたとすると，プログラムをどんどんこわしてゆくことになり，メチャクチャな事態となる。その結果がF0やF7のアボートとなる。このエラーは簡単なプログラムではまず少ないが，少し複雑なプログラムになると非常に起りやすく，殆んどの人が経験しているか，又はこれから経験するだろうと思う。このエラーを防ぐには配列を全く使わなければよい，というのは暴論めいた冗談ではあるが，これを検出するための方法が前述したコンパイラオプションのSUBCHK(Subscript Check，添字検査)である。FLTCHKと同様に¥FORTRANカードのLSTINの後にカンマを打ち，SUBCHKと打つだけでよい。こうすれば実行時エラー#104として<*>の模様にはさまれた形式のメッセージとして，発生したプログラム名と行番号，配列名，破ったときの添字の値，宣言してあるサイズが印字される。

F0やF7が配列破りが原因ではなくて，オーバーフローやアンダーフロー，ゼロ割算が原因のときもある。このときは前述したFORTRANフォールトプロセッサがメッセージを出すので，それを見てFLTCHKを適用すればよい。

ところでSUBCHKを使うときに注意すべきことは，¥FORTRANカードで，OPTZ(Optimization，最適化；処理の高速化を図るためのオプション)をSUBCHKと同時に指定してはいけないことである。OPTZとSUBCHKを同時に指定するとSUBCHKの方が無視される。FLTCHKはOPTZと同時指定してもかまわない。FLTCHKとSUBCHKは同時指定するよりも別々に使ってデバッグした方が無難であろう。

FLTCHKやSUBCHK等のエラー検出用のコンパイラオプションを指定すると処理速度は相当遅くなるので，デバッグが完全に終了してプログラムに一匹の虫もいなくなった段階では，これらのオプションはすみやかに取りはずして代りにOPTZを指定するのがよい。OPTZを指定すると筆者の経験では，指定しないときより2〜3割方，処理時間が短縮される。そのうちにもっと速くなるように改良されるであろう。

システムメッセージにはアボートやその他のメッセージがないときはNORMAL TERMI

NATIONとのみ印字されるが，注意すべきはNORMAL　TERMINATIONというのはそのジョブがエラーなしにうまく実行が終了したという意味では決してないことである。

以上，コンパイラメッセージに始まり，実行時エラーメッセージ，ローダメッセージ，システムメッセージなる4つの種類のエラーメッセージについて，とりとめもない文章を書いたが，何分にも筆者とて初心者やド素人に毛の生えた程度の知識しか持ち合せていないために，思い違いや誤り（フェイタルエラー？）があるかも知れないがその点は御許し頂きたい。この拙稿が4回生諸君をはじめ，ACOSユーザの初心者諸兄のために少しでも御役に立てば筆者としてこれにすぐる喜びはない，という陳腐なフレーズをこのテクニカルエッセイ（？）のエピローグとしよう。

（プログラム相談員）