11 章 例外割り込み処理機能

COBOL 2002 では、他の多くのオブジェクト指向言語がサポートしている例外割り込み処理機能が追加になりました。
伝統的な COBOL プログラムでは、プログラム全体のステップ数に対してエラーチェックの処理が占める割合はかなり多くなります。個々の処理ステップごとに、エラーの種別に応じてログファイルを残したり、コンソールへアラームを表示するプログラミングを記述すると、ステップ数が膨らんで生産性が低化します。さらに、プログラムの本来の処理がエラー処理に埋没してたいへん読みにくいプログラムになってしまいます。
例外割り込み処理機能は、現在の COBOL でもサポートされている、USE 文を持つ DECLARATIVES 手続きと同様に、各種のエラー発生時に宣言部分で記述されたエラー処理手続きが自動的に呼び出される機能です。

11.1 例外割り込み処理機能の概要

例外は、COBOL 文の実行時に発生するイベントです。COBOL 実行システムは、このイベントを検知すると、処理系によってまたはプログラマによって定義された所定の手続きに分岐します。これが例外割り込み機能です。
一般には、例外処理はオペレーティングシステムの機能として提供されています。UNIX ではシグナル処理がこれにあたり、汎用機では、十進演算などの COBOL 用の機械語命令が例外を起こすと、COBOL 実行システムがオペレーティングシステムを経由してこれを検知し、エラー報告を行っています。
COBOL 2002 が提供する例外割り込み機能では、このようなハードウェアやオペレーティングシステムが管理する例外を取り扱うことができますが、さらに、COBOL 独自の例外やプログラマが定義する例外も取り扱えるようになっています。
例外が発生すると、宣言部分に書かれた所定の手続きが実行されますが、その後の処理は、
例外の種類によって異なります。「致命的な」例外と呼ばれる一群の例外では、プログラムは実行を停止します。「非致命的な」例外の場合は、引き続き実行を継続します。プログラマが定義する例外はすべて「非致命的な」例外となります。

11.2 例外名

例外には、システムが定義するものとプログラマが定義するものとがあります。前者は、さらにCOBOL規格が定義するものとCOBOL処理系作成者が定義するものに分かれます。
COBOL 規格が定義する例外には、表の添え字範囲あふれ、十進データ不正、ゼロによる割り算などのおなじみのものがあります。COBOL 2002 以前は、このようなエラーが発生したときの動作は、ON SIZE ERROR 指定などが書かれていない限り規定されていませんでした。
プログラマが定義する例外とは、例えば、「預金残高不足」とか「在庫数量割れ」といったものになるでしょう。
COBOL 2002 では、これら各種の例外をプログラマが識別するために、例外名を用います。例外名は、以下の 3 階層で分類されています：

レベル 1 例外名：
　　EC-ALL、すべての例外をあらわす。
レベル 2 例外名：
　　EC-I-O など、例外を 19 に類別したそれぞれの代表名。
レベル 3 例外名：
　　レベル 2 例外名に、ハイフンと文字列をつけた名前で、個別の例外をあらわす。

以降に、COBOL 規格が定める例外名の一覧を示します。カテゴリ欄は、例外が致命的であるかどうかを示します。「作成者定義」は、致命的であるかどうかを COBOL 処理系作成者が定めるものを示します：

例外名	カテゴリ	説明
EC-ALL		すべての例外
EC-ARGUMENT		引数エラー
EC-ARGUMENT-FUNCTION	致命的	関数引数エラー
EC-ARGUMENT-IMP	作成者定義	処理系作成者定義の引数エラー
EC-BOUND		区域外
EC-BOUND-IMP	作成者定義	処理系作成者定義の区域外
EC-BOUND-ODO	致命的	OCCURS DEPENDING データ項目が区域外
EC-BOUND-PTR	致命的	ポインタデータ項目の内容が区域外
EC-BOUND-REF-MOD	致命的	区域外の部分参照
EC-BOUND-SUBSCRIPT	致命的	区域外の添え字付け
EC-BOUND-TABLE-LIMIT	致命的	動的な表の大きさ違反
EC-DATA		データ例外
EC-DATA-IMP	作成者定義	処理系作成者定義のデータ例外

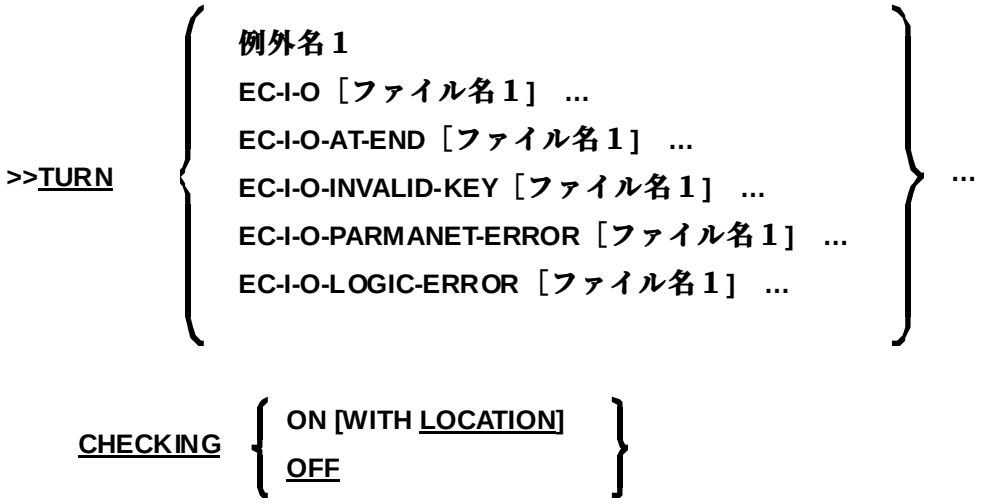
EC-DATA-INCOMPATIBLE	致命的	矛盾データ例外
EC-DATA-PTR-NULL	致命的	基底付き項目ポインタの NULL 参照
EC-FLOW		実行制御フロー違反
EC-FLOW-GLOBAL-EXIT	致命的	大域的な宣言手続き中での EXIT PROGRAM
EC-FLOW-GLOBAL-GOBACK	致命的	大域的な宣言手続き中での GOBACK
EC-FLOW-IMP	作成者定義	処理系作成者定義のフロー違反
EC-FLOW-RELEASE	致命的	RELEASE が SORT の範囲内にない
EC-FLOW-RETURN	致命的	RETURN が SORT や MERGE の範囲内にない
EC-FLOW-USE	致命的	USE 文が別の USE 文を実行させた
EC-I-O		入出力例外
EC-I-O-AT-END	非致命的	入出力状態「1x」
EC-I-O-FILE-SHARING	非致命的	入出力状態「6x」
EC-I-O-IMP	作成者定義	入出力状態「9x」
EC-I-O-INVALID-KEY	非致命的	入出力状態「2x」
EC-I-O-LINAGE	致命的	LINAGE データ項目の値が要求範囲外
EC-I-O-LOGIC-ERROR	致命的	入出力状態「4x」
EC-I-O-PERMANENT-ERROR	致命的	入出力状態「3x」
EC-I-O-RECORD-OPERATION	非致命的	入出力状態「5x」
EC-IMP		処理系作成者定義の例外
EC-LOCALE		地域固有仕様に関連するすべての例外
EC-LOCALE-IMP	作成者定義	処理系作成者定義の地域固有仕様関連例外
EC-LOCALE-INCOMPATIBLE	非致命的	参照した地域固有仕様に期待の文字指定がない
EC-LOCALE-MISSING	致命的	指定した地域固有仕様では使用できない
EC-LOCALE-SIZE	致命的	地域固有仕様の編集で数値は切り捨てられた
EC-OO		OO に関連するすべての既定儀例外
EC-OO-CONFORMANCE	致命的	オブジェクト修正子に対する不成功
EC-OO-EXCEPTION	致命的	例外オブジェクトが処理されなかった
EC-OO-IMP	作成者定義	処理系作成者定義の OO 例外
EC-OO-INTRINSIC	致命的	在来データ項目の型チェックが失敗した
EC-OO-METHOD	致命的	要求されるメソッドが使用可能でない
EC-OO-NULL	致命的	NULL オブジェクト参照によるメソッド起動
EC-OO-RESOURCE	致命的	オブジェクトのためのシステム資源不足
EC-OO-UNIVERSAL	致命的	実行時の型チェックが失敗した
EC-ORDER		順序付け例外
EC-ORDER-IMP	作成者定義	処理系作成者定義の順序付け例外
EC-ORDER-NOT-SUPPORTED	致命的	順序付けが未サポート
EC-OVERFLOW		桁あふれ条件
EC-OVERFLOW-IMP	作成者定義	処理系作成者定義の桁あふれ条件
EC-OVERFLOW-MEMORY	致命的	システム限界のため動的表が拡張不可能

EC-OVERFLOW-STRING	非致命的	STRING 文の桁あふれ条件
EC-OVERFLOW-UNSTRING	非致命的	UNSTRING 文の桁あふれ条件
EC-PROGRAM		プログラム間連絡例外
EC-PROGRAM-ARG-MISMATCH	致命的	引数の不一致
EC-PROGRAM-ARG-OMMITED	致命的	省略された引数への参照
EC-PROGRAM-CANCEL-ACTIVE	非致命的	取り消されるプログラムが活性状態
EC-PROGRAM-IMP	作成者定義	処理系作成者定義のプログラム間連絡例外
EC-PROGRAM-NOT-FOUND	致命的	呼ばれるプログラムが見つからず
EC-PROGRAM-PTR-NULL	致命的	NULL ポインタによる CALL 文
EC-PROGRAM-RECURSIVE-CALL	致命的	呼ばれるプログラムが活性状態
EC-PROGRAM-RESOURCES	致命的	呼ばれるプログラムで資源が使用不能
EC-RAISING		EXIT RAISING 例外
EC-RAISING-IMP	作成者定義	処理系作成者定義の EXIT RAISING 例外
EC-RAISING-NOT-SPECIFIED	致命的	ユーザ定義例外名が USING に書かれていない
EC-RANGE		範囲例外
EC-RANGE-IMP	作成者定義	処理系作成者定義の範囲例外
EC-RANGE-INDEX	致命的	指標が負の値か限界を超えている
EC-RANGE-INSPECT-SIZE	致命的	INSPECT 文での置き換え項目の大きさが異なる
EC-RANGE-PERFORM-VARYING	致命的	PERFORM VARYING の項目の値が負
EC-RANGE-SEARCH-INDEX	非致命的	SEARCH 文の指標の初期値が範囲外
EC-RANGE-SEARCH-NO-MATCH	非致命的	SEARCH 文で探索基準に合致する要素なし
EC-REPORT		報告書作成機能例外
EC-REPORT-ACTIVE	致命的	既に INITIATE された報告書に対する INITIATE
EC-REPORT-COLUMN-OVERLAP	非致命的	報告書項目の重なり
EC-REPORT-FILE-MODE	致命的	報告書ファイルのオープンモードが不正
EC-REPORT-IMP	作成者定義	処理系作成者定義の報告書作成機能例外
EC-REPORT-INACTIVE	致命的	未 INITIATE の報告書での GENERATE/TERMINATE
EC-REPORT-LINE-OVERLAP	非致命的	報告書行の重なり
EC-REPORT-LINE-WIDTH	非致命的	行幅を超えた

EC-REPORT-NOT-TERMINATED	非致命的	未 TERMINATE の報告書ファイルの CLOSE
EC-REPORT-PAGE-LIMIT	非致命的	縦方向のページの境界を越えた
EC-REPORT-SUM-SIZE	致命的	合計カウンタの桁あふれ
EC-SCREEN		画面操作例外
EC-SCREEN-FIELD-OVERLAP	非致命的	画面フィールドの重なり
EC-SCREEN-IMP	作成者定義	処理系作成者定義の画面操作例外
EC-SCREEN-ITEM-TRUNCATED	非致命的	行に対して画面フィールドが長すぎる
EC-SCREEN-LINE-NUMBER	非致命的	画面項目の行位置が端末の大きさを超えた
EC-SCREEN-STARTING-COLUMN	非致命的	画面項目の開始位置が行の大きさを超えた
EC-SIZE		桁あふれ例外
EC-SIZE-EXPONENTIATION	致命的	べき乗演算規則の違反
EC-SIZE-IMP	作成者定義	処理系作成者定義の桁あふれ例外
EC-SIZE-OVERFLOW	致命的	計算での算術桁あふれ
EC-SIZE-TRUNCATION	致命的	格納での有効桁切り捨て
EC-SIZE-UNDERFLOW	致命的	浮動小数点の下位桁あふれ
EC-SIZE-ZERO-DIVIDE	致命的	ゼロによる除算
EC-SORT-MERGE		SORT/MERGE の例外
EC-SORT-MERGE-ACTIVE	致命的	オープン済みファイルへの SORT/MERGE
EC-SORT-MERGE-FILE-OPEN	致命的	
EC-SORT-MERGE-IMP	作成者定義	処理系作成者定義の SORT/MERGE の例外
EC-SORT-MERGE-RELEASE	致命的	RELEASE レコードが長すぎまたは短すぎ
EC-SORT-MERGE-RETURN	致命的	AT END 後に RETURN が実行された
EC-SORT-MERGE-SEQUENCE	致命的	ファイルに対する順序エラー
EC-STORAGE		記憶領域割り当て例外
EC-STORAGE-IMP		処理系作成者定義の記憶領域割り当て例外
EC-STORAGE-NOT-ALLOC		割り当てられていない領域の FREE
EC-STORAGE-NOT-AVAIL		ALLOCATE 文で領域取得不可能
EC-VALIDATE		VALIDATE 例外
EC-VALIDATE-CONTENT		VALIDATE の内容エラー
EC-VALIDATE-FORMAT		VALIDATE の形式エラー
EC-VALIDATE-IMP		処理系作成者定義の VALIDATE 例外
EC-VALIDATE-RELATION		VALIDATE の比較エラー
EC-USER		利用者定義の例外
EC-USER-XXX		レベル 3 の利用者定義の例外

11.3 TURN コンパイル指令

プログラムで例外のチェックを行うかどうかは、コンパイル時に TURN 指令を使用して、例外名ごとに指定することができます。
この指令の省略値は何の例外もチェックしません。従って、例外処理機能を使用するには、以下の指令を指定することが必要です。



11.4 例外割り込み処理機能の手続き部

11.4.1 手続き部の枠組み

例外処理を使用するプログラムの手続き部は、次の形をとります：

```
PROCEDURE DIVISION RAISING 例外名 1. Á 例外名 1 を起こす副プログラムを宣言
DECLARATIVES.
例外処理 SECTION.
    USE AFTER EXCEPTION 例外名 2.      Á 例外名 2 に対する処理手続きの宣言
        …
    RESUME                             Á 例外処理手続きからの戻り
        …
END DECLARATIVES.
手続き処理 SECTION.
        …
    IF エラー
      RAISE EXCEPTION 例外名 2.                Á プログラマ定義例外の発生
        …
    EXIT PROGRAM
      RAISING EXCEPTION 例外名 1.      Á 呼び出し側に例外を発生させる
```

11.4.2 手続き部見出し

手続き部見出しの RAISING 指定では、プログラマ定義の例外名（EC-USER-XXX）を書き、プログラムが EXIT 文で発生させる可能性のある例外を宣言します。

```
PROCEDURE DIVISION
  [USING 指定]
  [RETURNING データ名 2]
[ RAISING { 例外名 1          } ... ]  .
          { クラス名 1        }
          { インタフェース名 1 }
```

例えば、「倉庫から商品を出荷する」というプログラムが、「在庫数量割れ」という例外を発生するように作成したとします。このプログラムを呼び出したプログラムは、CALL 文でその例外が発生し、呼び出し側に宣言された例外処理手続きが実行されることになります。

11.4.3 宣言部分

宣言部分の構成は、従来の入出力エラー処理手続きの宣言と変わりありません。変わったのは USE 文の新しい書き方が追加になっただけです。
11.2 で説明したように、例外名には入出力例外も含まれていますので、従来のファイル例外についての宣言手続きの書き方と、新しい例外処理の宣言手続きは、同じ目的のために同居することができます。

```
[ DECLARATIVES.
  { 節名  SECTION .  USE 文 .
    [完結文] ...  [ 段落名.  [ 完結文 ] ... ] ... } ...
  END  DECLARATIVES.  ]
```

11.4.4 EXIT PROGRAM 文

手続き部見出しの RAISING 指定で宣言した例外名は、EXIT PROGRAM 文の RAISING 指定に書くことによって例外を発生させます。
RAISING 指定の無い EXIT PROGRAM 文は、例外を発生させること無く呼び出し側プログラムに正常リターンします。呼び出し側プログラムに対して例外の発生を知らせたいときだけ RAISING 指定のある EXIT PROGRAM 文で戻ります。
従来のプログラミングでは、副プログラムを呼び出すと必ずそのあとで副プログラムが返すエラーコードをチェックするコードが書かれていました。この機能を使うと、呼び出し側で宣言手続きを書くだけで、各呼出し毎に毎回チェックする必要がなくなります。

```
EXIT  PROGRAM
[  RAISING { 例外名 1
           { クラス名 1
           { インタフェース名 1 }  ...  ]  .
```

11.4.5 RESUME 文

致命的でない例外については、宣言手続きの実行後、プログラム実行を再開することができます。通常は、宣言手続きが末尾まで実行されるか、EXIT 文に到達するところで、例外を生じた文の次の実行文から再開することになります。宣言手続き中に RESUME 文を書くと、即座に例外処理を停止して処理を再開することができます。

```
RESUME  AT  { NEXT STATEMENT }
            { 手続き名 1      }
```

11.4.6 RAISE 文

プログラマ定義の例外を明示的に発生させるには RAISE 文を使います。

```
RAISE  { EXCEPTION 例外名 1 }
       { 一意名 1           }
```

11.4.7 USE 文

例外処理で使用する宣言手続きに対しては、以下の形式の USE 文を使用して、特定の例外や特定のファイルに対する特定の例外についての例外処理を指定します。

```
USE AFTER EXCEPTION  { 例外名 1                    }
                     { 例外名 2 { ファイル名 2 }   }
```

例外名には、レベル 1、レベル 2、レベル 3 のどの例外名を書くこともできます。例外が発生したときに、その例外の種別が宣言部分に書かれた複数の USE 文に合致する場合、レベル 3 の例外名に関する USE 文が書かれていればそれが最優先となり、次いでレベル 2、レベル 3 の順で選択されます。

11.4.8 例外処理の組み込み関数

例外処理を使用するために便利ないくつかの組み込み関数が用意されています。

EXCEPTION-FILE 関数	**最近に発生した例外に関連するファイルのステータス値とファイル名を返す。**
EXCEPTION-LOCATION 関数	**最近に発生した例外の発生箇所を返す。プログラム名、段落名、行番号などの処理系作成者定義形式。**
EXCEPTION-STATEMENT 関数	**最近に例外を発生した COBOL 文の名前 (「MOVE」など)を返す。**
EXCEPTION-STATUS 関数	**最近に発生した例外の例外名を返す。**