オブジェクト指向分散環境 OZ++のプロセスと例外処理の実装

1H－2

濱崎 陽一 (電子技術総合研究所)　吉田 泰光* (日本ユニシス)
大西 雅夫* (東洋情報システム)　塚本 享治 (電子技術総合研究所)
*: 情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」研究員

1 はじめに

現在開発を進めているオブジェクト指向分散環境OZ++では、例外による大域脱出がネットワークをまたがって可能である。ここでは、並列実行の基本単位であるプロセスと例外処理の実装について述べる。

2 OZ++のプロセス

OZ++では、一連のメソッド呼び出しの連鎖をプロセスと呼ぶ。OZ++言語には、プロセスを扱うための演算子 (fork,join,detach,kill) が用意されている。プロセスは fork 演算をメソッド呼び出しに適用することより生成され、それを生成したプロセスとは並行に実行される。プロセスの同期は join 演算子を適用することにより、そのプロセスの終了を待合せて返り値を得ることができる。detach 演算子を適用すると独立したプロセスとなり、join 演算子を適用する事はできなくなる。

図1のプログラム例では、プロセス１からオブジェクトａのメソッドmaの呼び出しを起点とするプロセス２と、オブジェクトｂのメソッドmbの呼び出しを起点とするプロセス３が生成される。プロセス２はjoinにより消滅し、プロセス３はdetachされて、プロセス１とは独立したプロセスとなる。

3 例外

OZ++言語ではブロック構造に忠実な例外記述を採用している [2]。例外が発生すると、その例外を処理する例外ハンドラを持つ一番内側の例外文により捕捉される。

例外がオブジェクトＤのメソッドｄで発生した例を図2に示す。呼び出し元のメソッドＡの例外文にしかそ

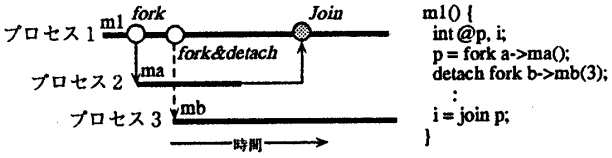
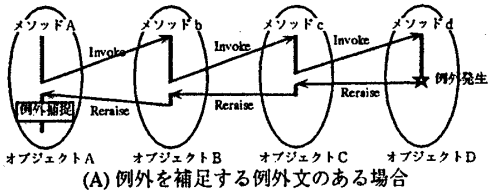
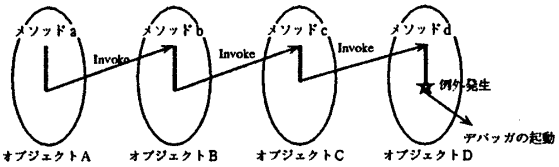


図 1: OZ++のプログラムとプロセスの例

れを捕捉できる例外ハンドラがない場合 (A) には、呼び出しの逆順に例外が reraise されてメソッドＡで例外が捕捉される。呼び出し側のどこにも発生した例外を捕捉できる例外文がない場合 (B) にはシステムで処理する必要がある。その対処には、呼び出し元まで例外を戻してランタイムエラーとする方法と例外発生時点で捕捉されない例外であることを検知してランタイムエラーとする方法が考えられる。前者では例外発生時の状況がランタイムエラーとなった時点では消滅しているためにデバッグが困難であり、特に分散プログラミングでのデバッグを容易にするためにOZ++では後者の方法を採用した。



(A) 例外を補足する例外文のある場合



(B) 例外を補足する例外文のない場合

図 2: 例外の補足

4 プロセスの実装

OZ++ではグローバルオブジェクトとローカルオブジェクトの２種類があり、それらのメソッドはスレッドにより実行される。グローバルオブジェクトとその部品であるローカルオブジェクトはセルとして常に同じ計算

An Implementation of processes and exception handling of OZ++: An Object-Oriented Distributed Systems Environment
Yoichi Hamazaki(Electrotechnical Laboratory),
Yasumitsu Yoshida*(Nihon Unisys, Ltd.),
Masao Onishi*(Toyo Information Systems, Co., Ltd.),
and Michiharu Tsukamoto(Electrotechnical Laboratory)
*: Research fellow of Open Fundamental Software Technology Project in Information-technology Promotion Agency, Japan

機上に存在するので、セル内のメソッドの実行は呼び出しを行ったスレッドの延長上で行われる。また、他のグローバルオブジェクトのメソッド呼び出しは、そのセルに新たに生成されたスレッドにより実行され、スレッド間のデータのやり取りと同期のためのチャネルを設けることによりオブジェクトを位置透過にみせている [1]。

プロセスはネットワーク上に分散したスレッドの集まりとなるため、システムで一意に識別可能な ID を各プロセスに付与している。

5 例外処理の実装

例外の処理は、処理系がスタック内にコンテキストや例外ハンドラのデータから成る例外フレームを埋め込み、実行系がデータのコピーやコンテキストの切替えを行うことで実現される。例外フレームは例外ハンドラが指定されている例外のリスト（捕捉例外リスト）を持つ。

発生した例外が最終的に捕捉されるか否かを判断するには、発生した例外が例外発生のメソッドのみならず呼び出し元からの入れ子になった例外文でも捕捉されないことを検出する必要がある。その実装方法としては呼び出しと逆の順序で例外フレームを例外発生時に調べる方法と、それ以前で捕捉可能な例外のリストを呼び出しのたびに渡して行く方法が考えられる。一つのスタック内では、効率の観点から前者の実装を行い、例外フレームの捕捉例外リストをリンクしておくこととした。また、異なるグローバルオブジェクトのメソッド実行の場合、前者の方式では通信を介して他のスレッドのスタック内の例外フレームに順次アクセスする必要があり、資源不足による例外（メモリ不足など）などの場合に複雑な通信処理を行うのは不適当であることから、後者の実装を行い、メソッド呼び出し時にそれより手前で捕捉可能な例外のリストを渡すこととした。

図3に二つのグローバルオブジェクトにまたがるプロセスの例を示す。図では五つのメソッドが順に呼び出されており、二つのスレッドが生成されている（図3(A)）。これらのスタックは図3(B) のようになり、チャネルで二つのスタックは関係づけられている。スタックはそれぞれのメソッドに対応したスタックフレームから成り、その中には例外フレームが埋め込まれている。

スレッドBには、メソッドA1,a1,a2 での捕捉例外リストの和集合がチャネルを介して渡される。任意の例外を捕捉できる例外ハンドラが含まれる場合には、任意の例外を示す ANY のみの捕捉例外リストが渡される。

図のメソッドb1 において例外が発生した場合、次の手順で処理される。

1) 実行系はメソッドb1 の捕捉例外リストに発生した例外が含まれるかどうかを調べる。あれば対応する例外ハンドラにコンテクストスイッチし、なければシステム組み込みのハンドラに制御を移す。

2) システム組み込みのハンドラでは、順次、メソッドB1 の捕捉例外リストおよびチャネルに渡された捕捉例外リストに発生した例外が含まれるかどうかを調べる。あればその例外を、なければランタイムエラーを reraise してメソッドB1 に制御が移る。

ランタイムエラーの場合には、デバッガを用いてスタックの内容を調べることができる。

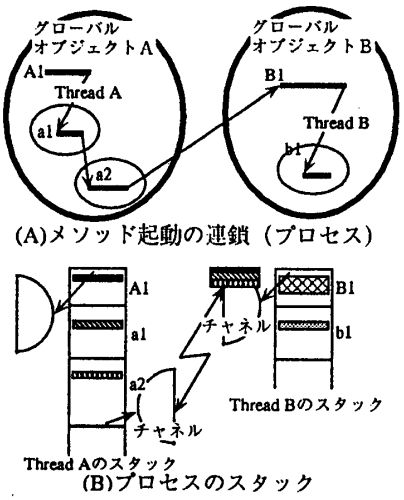


図 3: プロセスとそのスタック

6 まとめ

分散アプリケーションの開発を容易にする環境の提供を目指したオブジェクト指向分散環境OZ++の例外処理について述べた。ネットワークにまたがる一連のメソッド呼び出しであるプロセスを単位とした並行実行と例外によりネットワークをまたがる大域脱出を可能にしている。こうした機構により、排他制御や同期が必要な分散プログラムを例外を使って容易に記述できるようになっている。現在、初版のシステムに実装し評価を進めているところである。

本研究は、情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

参考文献

[1] 濱崎他:「オブジェクト指向分散環境OZ++の通信機構の実装」、情報処理学会第 49 回全国大会、Sep. 1994

[2] 音川他：「オブジェクト指向分散環境OZ++の言語における例外処理の記述」、情報処理学会第 50 回全国大会、Mar. 1995