5P-9

V70/V80 CTRON のカーネル例外処理とその実現

佐藤 直樹*, 内田 昭宏*, 中本 幸一*, 井沢 昭子*, 神田 正己**, 面川 晃徳**, 大鷹 正之***, 小泉 正*
* 日本電気(株)C&C 共通ソフトウェア開発本部 **日本電気技術情報システム開発(株)
***日本電気(株) 交換第1ネットワークシステム事業部

1 はじめに

筆者らはNECの32ビットオリジナルマイクロプロセッサV70/V80上にCTRON[1]カーネル[1]を実現した．本稿では，CTRONカーネルの例外処理の実現について述べる．CTRONカーネルの例外処理は，UNIX[2]のシグナル処理[2]と比較した場合，例外の保留処理及び，システムコール中で待ち状態になっているタスクに発生した例外の処理が大きく異なっており，実現方法も既存のOSとは異なるものとなる．

本稿では，はじめにCTRONカーネルの例外処理について述べ，UNIXのシグナル処理との比較を行なう．次に，CTRONカーネルの例外処理の実現について述べ，最後にまとめを行なう．

2 CTRONカーネルの例外処理

CTRONでは，タスクの正常な処理の流れに非同期に割り込む事象のことを例外と呼んでいる．CTRONカーネルでは，例外を内部例外と外部例外にわけている．

- **内部例外**
  走行中のプログラムが不正メモリアクセス，プログラムエラー等の原因により，正常な処理を行なえなくなった場合に発生する例外．
- **外部例外**
  タスクの処理に対し，外部から非同期通知が行なわれた時に発生する例外．必ずしもエラーを意味しない．他タスクからの割り込み，非同期入出力の結果通知などがある．

また，例外は原因によっていくつかのクラスに分類されており，それぞれのクラスに対して例外ハンドラが定義される．例外ハンドラは，例外が発生したタスクの環境で実行される．例外の発生は，例外マスク管理機能によってクラス単位に禁止・許可することが可能である．

CTRONカーネルの例外処理の特徴には次のものがある．

- 例外がマスクされている場合，外部例外は保留され，マスクが解除された時に例外が発生する．同一クラスの例外が複数発生しても全てが保留される．
- 待ち状態のタスクに例外が発生した場合，タスクは一時的に走行状態に遷移し例外ハンドラを実行し，実行が終了すると再び待ち状態に遷移する．ハンドラ中で待ち状態を解除したり，図1のように待ち状態に遷移することも可能である(この場合には2重に待ち状態に入ったことになる)．

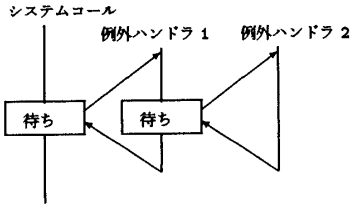

図1: 待ちのネスト

3 他のOSの例外処理との比較

次に，CTRONカーネルの例外処理が，UNIXのシグナル処理と異なる部分について述べ，実現する場合にどのような点が問題となるかを考察する．

UNIXでは，シグナルがマスクされている場合，そのシグナルは保留される．しかし，保留される同一シグナルは高々1つである．このような保留機構を実現するためには，シグナルの種類ごとに保留されているシグナルが存在することを示すビットを用意すれば良い[2]．しかし，CTRONカーネルではマスクされているクラスの例外が複数発生した場合，全てを保留する必要がある．そのため，例外の保留状態をビットで記憶する方法では実現できない．

また，UNIXではシステムコール中で待ち状態になっている時にシグナルが発生すると，システムコールが終了するか[2]，シグナルで中断したシステムコールを再実行する[3]．これに対し，V70/V80 CTRONカーネルでは例外処理が終了したのちに再び待ち状態に戻る．従ってCTRONカーネルでは，UNIXのシグナル処理で行なわれるシステムコールの終了や再実行とは異なる，待ち状態への復帰を行なう機構が必要となる．

4 例外処理の実現

前章で述べたように，CTRONカーネルではマスクされている時に発生した例外を全て保留する機構，待ち状態を中断し例外ハンドラを実行した後，再び待ちに戻るための機構が必要である．以下で，これらのV70/V80 CTRONカーネルでの実現方法について述べる．

4.1 例外処理のためのデータ構造

V70/V80 CTRONカーネルの例外処理を実現するために，図2に示す例外管理構造をタスク管理構造(Tcb)中に用意した．この例外管理構造は，処理中例外クラス，スタック上の情報，処理待ち例外キュー，保留例外キューを含んでいる．これらの役割については以下で述べる．また，例外の保留は，例外情報をキューにつなぐことにより実現している．

"V70/V80 CTRON Kernel Exception Handling and Implementation"
Naoki Sato*, Akihiro Uchida*, Yukikazu Nakamoto*, Akiko Izawa*, Masami Kanda**, Akinori Omokawa**, Masayuki Ootaka***, Tadashi Koizumi*
*C&C Common Software Development Laboratory, NEC Corporation,
**NEC Scientific Information System Development Ltd.,
***1st Switching Network Systems Division, NEC Corporation

[1]CTRONは"Communication and Central TRON"の略称，TRONは"The Realtime Operating system Nucleus"の頭文字です．

[2]UNIXは米国AT&T社が開発しライセンスしている．

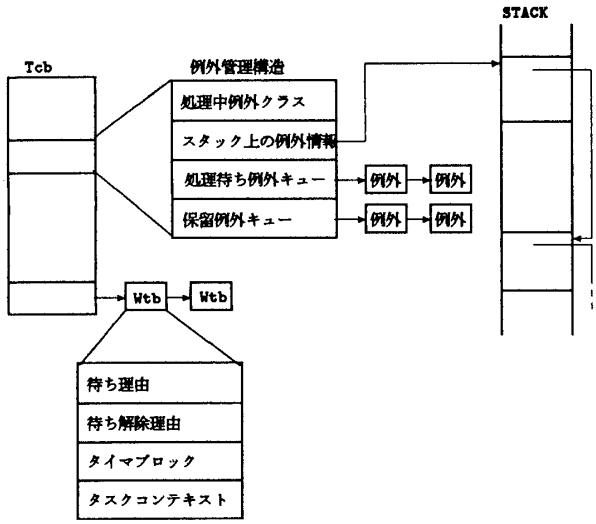

図 2: 例外処理のためのデータ構造

- **処理中例外クラス**
  現在処理中の例外のクラスを記憶する.
- **スタック上の例外情報**
  タスクに例外が発生した場合，発生時のレジスタ情報がスタック上に退避される．また，例外ハンドラ実行中に新たに例外が発生した場合には，処理中だった例外のクラスもスタック上に退避される．これらの情報は，例外の発生した順にスタック上にポインタでつながれている．最も最近発生した例外の情報は，Tcb からのポインタが指し示している.
- **処理待ち例外キュー**
  タスクに例外が発生し，その例外がマスクされていなければ例外情報をこの処理待ち例外キューにつなぐ．例外を処理できる状況になった場合，すぐに例外ハンドラを起動する.
- **保留例外キュー**
  保留された例外情報をつなぐ．マスクの解除など保留原因が解除されると処理待ちキューへつなぎかえる.

4.2 待ち状態で発生した例外の処理

CTRON カーネルでは，タスクの待ち状態で例外が発生した場合，待ち状態を中断して例外ハンドラが実行される．さらに，この例外ハンドラ中で再び別の待ち状態に入ることが可能である．待ち状態の途中で起動した例外ハンドラが再び待ち状態にはいることは，一つのタスクが複数の待ち要因を同時に持つことを意味する．これを実現するために，タスクの待ち状態に関する情報を Tcb の外に持たせ，複数の待ち状態をタスクが保持できる構造としている.

待ち状態を保持する構造を，待ち状態管理構造 (Wtb)(図 2) と呼ぶ.

4.2.1 待ち状態の中断

タスクは通常一つの Wtb を持ち，タスクのスイッチングでコンテキストを保存するための領域として Wtb を使用している．待ち状態から例外ハンドラが起動された場合，新たな Wtb 確保し，Tcb に接続する．例外ハンドラ中でタスクスイッチングを行なう場合，コンテキストは新たに接続した Wtb に保存する．従って，例外発生前の Wtb には待ち状態のタスクコンテキストが保存されたままになっており，この情報を使用して再び待ち状態に戻ることが可能である.

また，待ち状態にあるタスクの Tcb は待ち要因ごとの待ちキューにつないであるが，待ちを中断して例外処理を行なう場合には，待ちキューからはずす．これにより，待ちキューと待ちの中断の関係が完全に切り離され，待ちを中断して例外処理を特別扱いする必要がなくなっている.

4.2.2 待ち状態への復帰

例外処理から待ち状態への復帰は，例外処理中に使用していた Wtb を Tcb から切り離し，例外発生前の Wtb 中のタスクコンテキストを復帰することにより行なう．また，待ちキューから外されていた場合には待ちキューへ戻す.

4.2.3 中断された待ち状態の解除

4.2.2で例外処理終了後の待ち状態への復帰について述べた．しかし，例外ハンドラ実行中に，中断されている待ちの要因が解除され，待ち状態に復帰できない場合や復帰しても意味のない場合がある．そのため，例外処理を終了した場合，無条件に待ち状態に復帰するのではなく，待ち状態に復帰することが可能かどうかを確認する必要がある．これには以下の場合がある.

- **時限切れ**
  例外処理中に待ち時間がオーバーした場合.
- **強制起床**
  例外ハンドラ走行中に，強制起床を実行され最も最近の待ち状態から強制的に起床された場合.
- **待ち対象獲得**
  例外処理中に該当待ち条件が成立した場合.
- **待ち対象削除**
  例外処理中にタスク間通信に使用していた資源など，待ち対象が削除された場合.

例外処理の終了後に上記の状況の発生を確認し，もし発生していたならば，待ち状態への復帰させることはできないので，待ちキューからレディ・キューへつなぎかえる.

5 まとめ

CTRON カーネルの例外処理は，UNIX のシグナル処理と比較した場合，保留，システムコール中に発生した場合の処理が大きく異なる．V70/V80 CTRON カーネルでは，保留されている例外を記憶するためのキューを設け，同一クラスの例外の保留の問題を解決した．また，待ち状態のタスクのコンテキストをリストにして保持することにより，例外ハンドラの実行終了後再び待ち状態に遷移することを可能とした.

参考文献

[1] トロン協会 -CTRON 専門委員会編，“原典 CTRON 大系 2 カーネルインタフェース”，1988.
[2] Bach:“THE DESIGN OF THE UNIX OPERATING SYSTEM”, PRENTICE-HALL, 1986.
[3] Leffler,S.J et al.: “4.3BSD UNIX Operating System”, Addison-Wesley Publishing Company, 1989.