カーネル勉強会資料

遠藤 友悟

2001 年 8 月 20 日

1　割込み処理モデル

割込み処理部分が，移植性の高い部分（割込みサービスルーチン）と移植性の低い部分（割込みハンドラ）に分けられている．

1.1　割込みハンドラ

- プロセッサの機能のみに依存して起動
- 割込みコントローラ（以下，IRC）を操作
- 実装により異なるため，実装定義

1.2　割込みサービスルーチン

- 割込みハンドラから起動
- 割込み要因に応じた実際の処理内容が書かれる
- 実装に依存しない

1.3　割込み処理の流れ

- 割込みハンドラの出入り口処理（カーネル）
  - レジスタ保存・復帰
  - スタック切替え
  - タスクディスパッチ処理
  - （プロセッサレベルでの）割込みハンドラからの復帰
- 割込みハンドラ（IRC の操作）
  - 割込み要因の取り出し
  - 取り出した割込み要因により分岐
  - IRC のエッジトリガのクリア

  - IRC の割込みサービス中フラグのクリア
- 割込みサービスルーチン

これらの処理のうち，実際にどれが必要となるかは場合によって異なる．

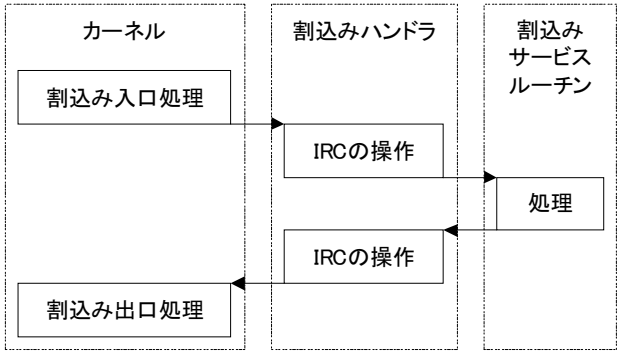

図 1: 割込み処理モデル

1.4　割込みの指定方法

1.4.1　割込みハンドラ番号（INHNO 型）

どの割込みを対象に割込みハンドラを登録するかを指定．一般的にはプロセッサの割込みベクトル番号に対応（割込みベクタテーブルがない場合は一つ）．

1.4.2　割込み番号（INTNO 型）

どの割込みを対象に割込みサービスルーチンを登録するかを指定．一般的には IRC への割込み要求入力ラインに対応．割込み要因と割込み要求が 1 対 1 で対応しない場合（一本の割込み要求入力ラインに複数のデバイスが接続されていて割込みベクトル番号が供給されない），一つの割込み番号に対して複数の割込みサービスルーチンを登録することができる．

割込みを個別に禁止・許可する場合は，割込み番号を指定する．

2　例外処理モデル

2.1　CPU 例外ハンドラ

プロセッサが CPU 例外を検出した場合に起動．ある一つの CPU 例外に対応する CPU 例外ハンドラはシステム全体で共通である．プロセッサに依存するため実装定義となるが，次の機能を持つ必要がある．

- CPU 例外が発生したコンテキストや状態の読出し
- CPU 例外が発生したタスクの ID 番号の読出し
- タスク例外処理の要求

2.2 タスク例外処理

タスクに発生した例外事象の処理を，そのタスクのコンテキストで行うための機能.

- タスク例外処理ルーチンの定義
- タスク例外処理の要求
- タスク例外処理の禁止・許可
- （タスク例外処理に関する情報を参照）

アプリケーションは，タスク毎に一つのタスク例外処理ルーチンを登録することができる．タスク例外処理要求時には，タスク例外要因が指定され，複数の要求は保留例外要因という形でタスク例外要因のビット毎の論理和で保存される.

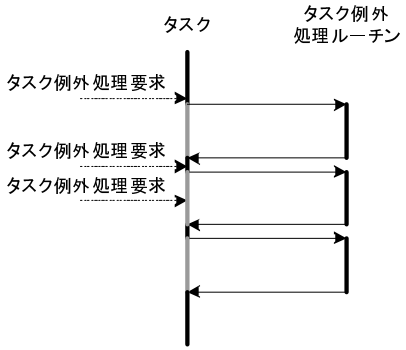

図 2: タスク例外処理

3 コンテキストとシステム状態

カーネルは次の処理単位の実効制御を行う.

- 割込みハンドラ
  - 割込みサービスルーチン

- タイムイベントハンドラ
- CPU 例外ハンドラ
- 拡張サービスコールルーチン
- タスク
  - タスク例外処理ルーチン

それぞれは独立したコンテキストで実行されるが，タスク例外処理ルーチンは，そのタスクのコンテキストで実行される.

カーネル自身は次の処理を行う．カーネルが実行されるコンテキストは，アプリケーションの振舞いには影響しないため，特に規定されない.

- サービスコール処理
- ディスパッチャ
- 割込みハンドラ・CPU 例外ハンドラの出入口処理など

3.1 タスクコンテキストと非タスクコンテキスト

タスクの処理の一部と見なせるコンテキストを総称してタスクコンテキスト，それ以外を非タスクコンテキストと呼ぶ．非タスクコンテキストから，暗黙で字タスクを指定するサービスコールや，自タスクを広義の待ち状態にする可能性のあるサービスコールを呼び出すことはできない（コンテキストエラー）.

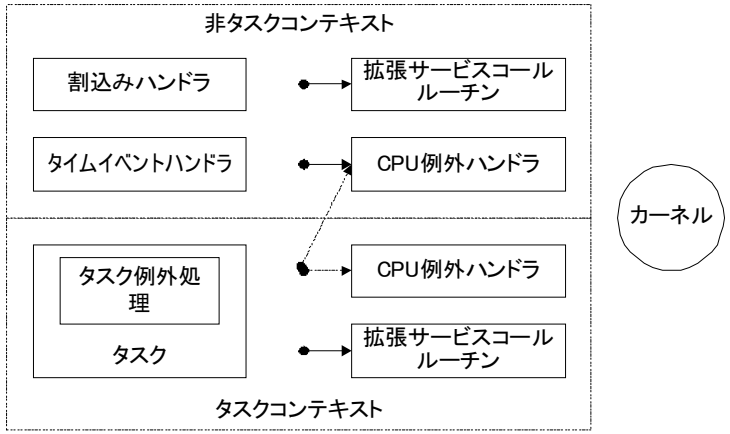

図 3: 各処理単位のコンテキスト

3.2 処理の優先順位

各処理単位とディスパッチャは次の優先順位で実行される．

- 割込みハンドラ，タイムイベントハンドラ，CPU 例外ハンドラ
- ディスパッチャ
- タスク

割込みハンドラ，タイムイベントハンドラ，CPU 例外ハンドラのそれぞれの間の優先順位の関係は主に実装定義である（詳細は省略）．

カーネルが行うそれぞれの処理は不可分に実行されなければならない（実装によってはこの原則を緩めることが許されている）．

3.3 CPU ロック状態

システムは CPU ロック状態か CPU ロック解除状態のいずれかの状態をとり，CPU ロック状態では，割込みハンドラ，タイムイベントハンドラ，ディスパッチは起動されず，基本的にサービスコールも呼び出せない．また，CPU ロック解除状態でも割込みが許可されているとは限らない．

	実行開始直後の状態	リターン（終了）時の状態
割込みハンドラ	実装依存	CPU ロック解除状態
割込みサービスルーチン	CPU ロック解除状態	CPU ロック解除状態
タイムイベントハンドラ	CPU ロック解除状態	CPU ロック解除状態
CPU 例外ハンドラ	変化しない	変化しない
拡張サービスコールルーチン	変化しない	変化しない
タスク	CPU ロック解除状態	CPU ロック解除状態
タスク例外処理ルーチン	変化しない	変化しない

表 1: CPU ロック状態の遷移

3.4 ディスパッチ禁止状態

システムはディスパッチ禁止状態かディスパッチ許可状態のいずれかの状態を取り，ディスパッチ禁止状態では，ディスパッチは起こらない．タスクコンテキストから，自タスクを広義の待ち状態にする可能性のあるサービスコールを呼び出した場合の振舞いは未定義である[1]．

3.5 ディスパッチ保留状態

ディスパッチャよりも優先順位の高い処理が行われている間，CPU ロック状態の間，ディスパッチ禁止状態の間は，ディスパッチが起こらず，ディスパッチ保留状態と呼ばれる．その間にディスパッチが起こる状況となった場合，それまで実行中であったタスクが実行状態であり，ディスパッチの対象となるタスクは実行可能状態である．

[1] ポーリングを行うサービスコールは呼び出せるが，タイムアウトありのサービスコールにポーリングをさせようとしても，その振舞いは未定義

	実行開始直後の状態	リターン（終了）時の状態
割込みハンドラ	変化しない	変化しない
割込みサービスルーチン	変化しない	変化しない
タイムイベントハンドラ	変化しない	変化しない
CPU 例外ハンドラ	変化しない	変化しない
拡張サービスコールルーチン	変化しない	変化しない
タスク	ディスパッチ許可状態	ディスパッチ許可状態
タスク例外処理ルーチン	変化しない	変化しない

表 2: ディスパッチ禁止状態の遷移

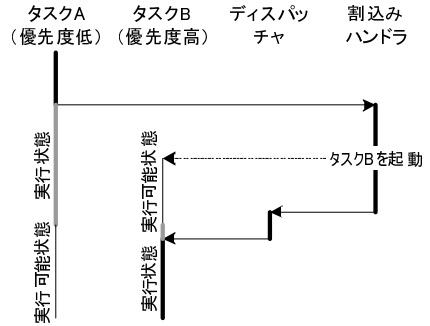

図 4: ディスパッチ保留状態