【A2】Delphiテクニカルセッション

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

Delphiでのマルチスレッドプログラミング

アナハイムテクノロジー株式会社・代表取締役
はやし　つとむ

アジェンダ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- マルチスレッドの基礎知識
- 排他制御について
- Windowsメッセージングの利用
- 某図書館システムでの実装

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

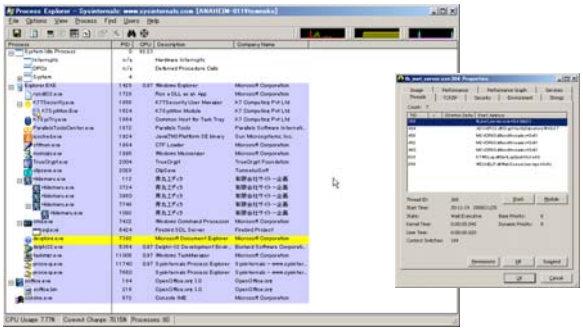
マルチスレッドの基礎知識

- プロセスモデルとスレッドモデル
  - モダンOS上では、各プロセスはメモリ空間を共有せず、OSが用意するプロセス間通信を利用しないと協調動作が出来ないが、それ故メモリ破壊を免れるため、堅牢である。
  - スレッドは、プロセス上に生成される軽量な疑似プロセスであり、親プロセスとメモリ空間を共有するため容易に協調動作を実現出来るが、それ故にメモリ破壊を引き起こしやすい。

マルチスレッドの基礎知識

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 便利なツール類
  - ProcessExplorer
  - Debugging Tools for Windows



マルチスレッドの基礎知識

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Windows上でのスレッド
  - Win32API CreateThread
    - *HANDLE CreateThread(*
      LPSECURITY_ATTRIBUTES lpThreadAttributes, // セキュリティ記述子
      DWORD dwStackSize, // 初期のスタックサイズ
      LPTHREAD_START_ROUTINE lpStartAddress, // スレッドの機能
      LPVOID lpParameter, // スレッドの引数
      DWORD dwCreationFlags, // 作成オプション
      LPDWORD lpThreadId // スレッド識別子
      );
  - 各スレッドにはデフォルトで1MBのスタック領域が割り当てられるので、最大で約2,000個のスレッドを生成出来る。(MSDN)
  - 論より証拠で実行してみると・・・
  - スタックエリアを20Kバイト以下にすれば、10万スレッドも可能なはずだが・・・

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

マルチスレッドの基礎知識

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Delphiでのスレッド利用
  - TThreadクラスを利用する
    通常は、TThreadクラスを使います。
    TThreadクラスを使うともっとも手軽にThreadを利用出来る
  - System.BeginThread関数を利用する
    よりプリミティブな使い方。

マルチスレッドの基礎知識

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- TThreadクラスの基本的な使い方
  - 主なメソッド
    - Executeメソッド
    - Terminateメソッド
    - Resumeメソッド
    - Synchronizeメソッド（StaticSynchronizeメソッド）
    - Queueメソッド（StaticQueueメソッド）
  - Synchronize と Queue メソッドでは、Delphi2009の新機能である匿名メソッドも利用出来る。

マルチスレッドの基礎知識

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- System.BeginThread関数
  - *function BeginThread(*
    SecurityAttributes: Pointer; *//セキュリティ識別子 nil を指定すればデフォルト値となる*
    StackSize: LongWord; *//スタックサイズ 0を指定すればデフォルト値(1MB)となる*
    ThreadFunc: TThreadFunc; *//スレッド関数へのポインタ*
    Parameter: Pointer; *//型無しポインタなのでなんでも渡せる*
    CreationFlags: LongWord; *//CREATE_SUSPENDED,*
    STACK_SIZE_PARAM_IS_A_RESERVATION を指定出来る
    var ThreadId: LongWord): Integer; *//スレッドIDが返される*
  - CreateThread APIをラップしたプリミティブな関数
  - APIの直接実行ではAPIルーチン以外は使えないが、Delphiの機能を利用出来るので通常はこちらを使う
  - 内部的にIsMultiThreadグローバル変数の設定や、Threadラッパーを通してThreadFuncを実行するなどの仕掛けがしてある。
  - ThreadFunc内では例外処理を内部で全てするべきだが、未処理の例外もスレッドラッパーを通じてDelphi側で処理出来る

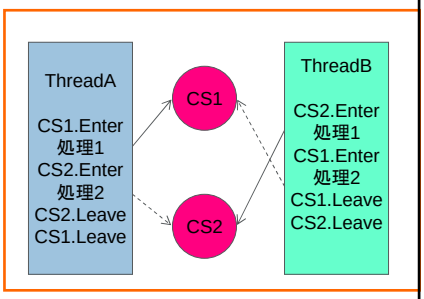
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

排他制御について

排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

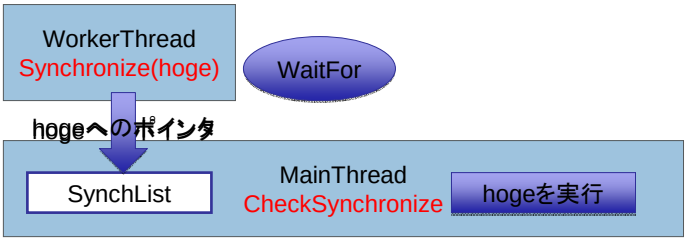
- 排他制御
  - TThreadの機能を利用する
    - Synchronize
    - Queue
  - クリティカルセクション　TCriticalSection
  - セマフォ　TSemaphore
  - ミューテックス　TMutex
  - イベント　TEvent
- 排他制御の問題点
  - デッドロック



排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Synchronize
  - TThreadのもっともメジャーな機能
  - ワーカースレッド側からメインスレッド側に制御を移して関数を実行する
  - 実行が終わるまでワーカースレッドは待たされる＝並行動作ではない
  - VCLのヴィジュアルコンポーネントなどにアクセスする際は必須
  - TThread側で呼び出す関数をキューに突っ込んで、それをTApplicationのメッセージループ内で処理している。



排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Synchronize()での匿名メソッドの使い方

匿名メソッドを使った場合

```
procedure TMythread.Execute;
var
  f:TThreadProcedure;
begin
  FMsg := 'Start thread ID = ' + IntToStr(ThreadID);
  f := procedure
  begin
    FMemo.Lines.Add(FMsg);
  end;
  Synchronize(f);
  while not terminated do
  begin
    sleep(0);
  end;
  FMsg := 'Terminated thread ID = ' + IntToStr(ThreadID);
  Synchronize(f);
end;
```

従来の書き方

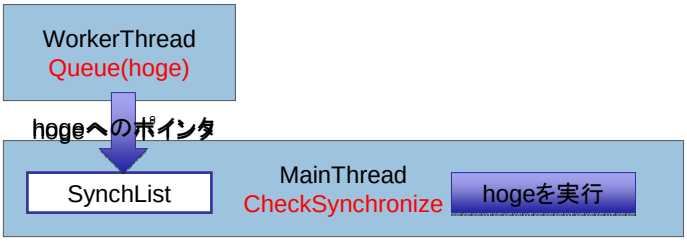
```
procedure TMythread.WriteMsg;
begin
  FMemo.Lines.add(FMsg);
end;

procedure TMythread.Execute;
begin
  FMsg := 'Start thread ID = ' + IntToStr(ThreadID);
  Synchronize(WriteMsg);
  while not terminated do
  begin
    sleep(0);
  end;
  FMsg := 'Terminated thread ID = ' +
IntToStr(ThreadID);
  Synchronize(WriteMsg);
end;
```


排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Queue
  - Delphi2005で追加された機能
  - ワーカースレッド側からメインスレッド側に制御を移して関数を実行する
  - メインスレッド側の実行は非同期＝並行して行われる
  - TThread側で呼び出す関数をキューに突っ込んで、それをTApplicationのメッセージループ内で処理している。



排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- クリティカルセクション
  - 複数のスレッドが同時に一つのリソース（メモリなど）にアクセスしないように排他制御を行う。
  - Delphiでは、Windows APIをラップしたTCriticalSectionクラスを用意している。
  - TCriticalSectionの主なメソッド
    - Acquire, Enter　・・・この二つは同じ
      クリティカルセクションに入れるまで待機状態になる
    - Release, Leave　　・・・この二つは同じ
      クリティカルセクションから出て、所有権を解放する
    - TryEnter
      クリティカルセクションに入れたら0以外、入れないと0を返す
      待機状態にならない

排他制御について

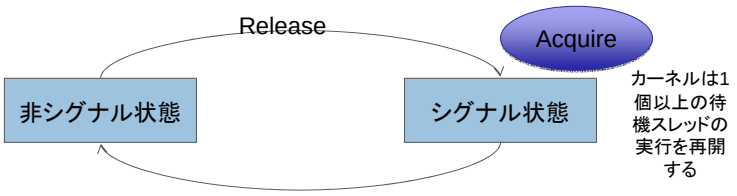
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- クリティカルセクションの使い方
  - スレッド間でリソースを共有する場合、その前後をクリティカルセクションで囲む。全てやらないと意味がない。
  - メインスレッドとワーカースレッドの同期であれば、ワーカースレッド側のメンバとして実装すると使いやすい。
  - 複数のワーカースレッド間での同期が必要な場合は、グローバル変数で定義して、initialize finalize で生成と解放を行う。

排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

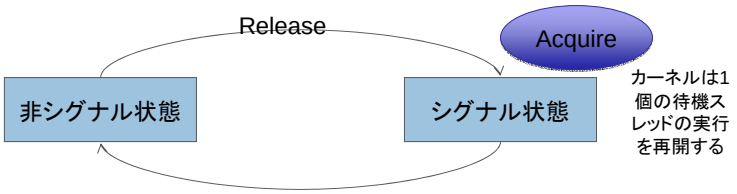
- TSemaphore
  - Delphi2009の新機能
  - セマフォの場合は、初期値として設定したカウンタを一つずつ減らしながら所有権を取得していく
  - 同時に一定数までのスレッドにリソースを使用させたり、動作させたりするために利用する



排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- TMutex
  - Delphi2005で追加された機能
  - Mutexの所有権を得たスレッドが動作することが出来る
  - TMutexでは、Acquire と Release を使えば良い
  - 同時に実行出来るのは一つのスレッドだけ



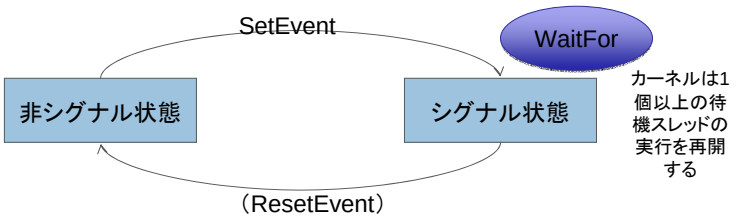
排他制御について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- TEvent
  - イベントは、一つだけのシグナルを持つセマフォ＝バイナリセマフォ
  - コンストラクタは二通り
    - *constructor TEvent.Create(*

```
EventAttributes: PSecurityAttributes;   //セキュリティ記述子  nilとすると規定値
ManualReset, InitialState: Boolean;     //Trueの場合、シグナルを受け取った後でResetEventする必要有り
const Name: string;                     //オブジェクトの名前、衝突した場合は失敗する
UseCOMWait: Boolean);
```

    - *constructor TEvent.Create(UseCOMWait: Boolean);*
      こちらは内部的に一つ目を呼び出す。その際の規定値は、Create(nil, True, False, '', UseCOMWait);



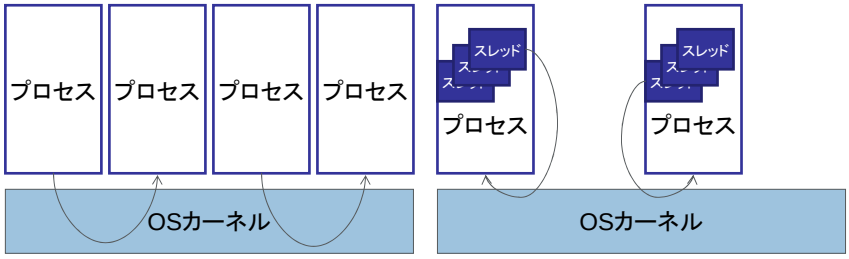
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

Windowsメッセージングの利用

Windowsメッセージングの利用

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

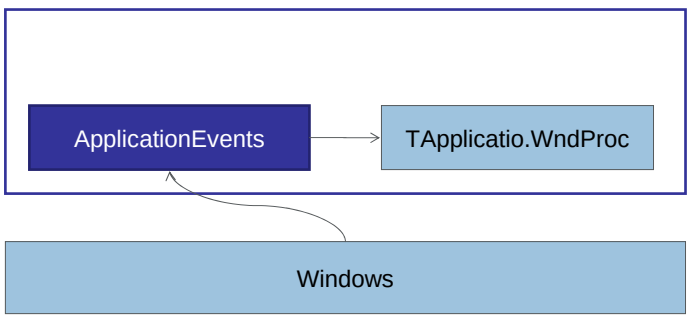
- スレッド間の通信手段としてWindowsメッセージングを利用する
- 複数のワーカースレッドからの通知をメインスレッドが非同期で受信出来る
- メインスレッド側にキューとクリティカルセクションをもうけるより、ワーカースレッドの実行を阻害しない



Windowsメッセージングの利用

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- TApplicationEvents
  - OnMessageイベントを利用すると、システムに送られる全てのメッセージをトラップ出来る
  - RegisterWindowMessage API を利用してメッセージを登録して利用



EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

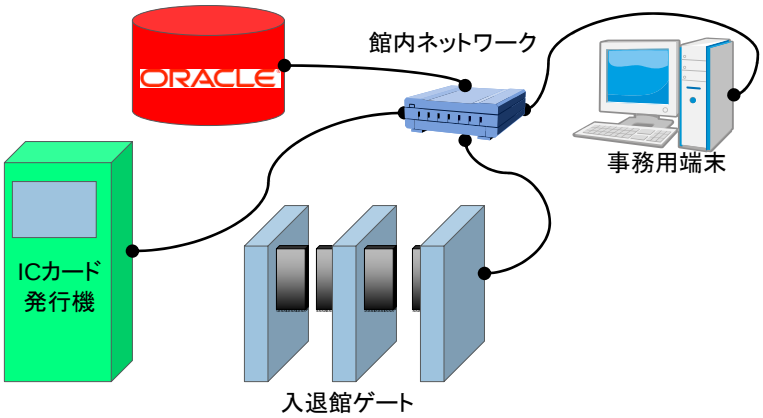
某図書館システムでの実装

某図書館システムでの実装

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

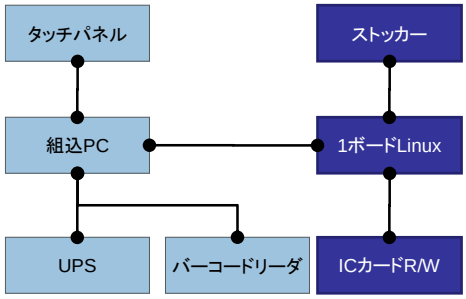
- ICカードを発行して、ゲートを通過し、内部で本を借りている間は外に出れないという仕組みを実現。そのICカード発行機を担当。



某図書館システムでの実装

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

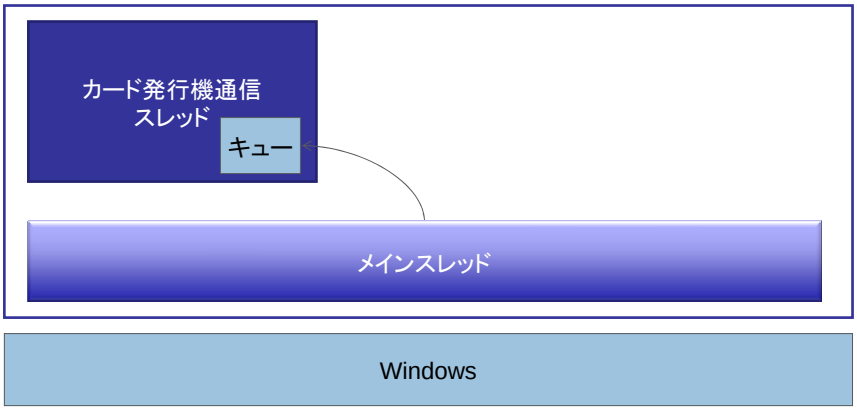
- ICカード券売機の中には下記の各機器がビルトインされていた
  - 組込用パソコン
  - 1ボードLinuxマシン
  - UPS
  - タッチパネル
  - バーコードリーダ
  - ICカードリーダー/ライター
  - ICカードストッカー
  - 全部RS-232Cです。



某図書館システムでの実装

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

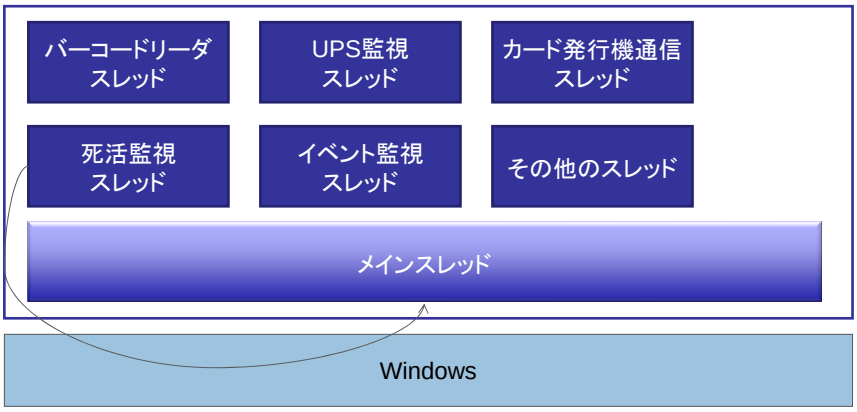
- スレッド化の概要
  - メインスレッドからワーカースレッドへの通知には、コマンドキューを利用した



某図書館システムでの実装

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- スレッド化の概要
  - ワーカースレッドからメインスレッドへの通知には、Windowsメッセージを利用した



アナハイムテクノロジーについて

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- アナハイムテクノロジー株式会社
  - 〒157-0072　世田谷区祖師谷1-22-26-S-208
  - TEL 03-5787-7791　FAX 03-5787-7792
  - http://www.anaheim-tech.com/
- Delphiとオープンソースの技術支援を行っています
- InterBase/Firebirdのご相談もぜひ当社へ！