TECS ジェネレータ V1.1.0 の新機能

2012.6.12

本書の位置づけ

本書は TECS 仕様書の更新ができていない部分を補足するものである。

現時点の TECS 仕様書のバージョンは V1.0.2.32 (2011/5/3) である。V1.1.0 の仕様について、まとめて記したものがないため、本書により補足する。

TECS ジェネレータ V1.1の新機能

- エラーメッセージの日本語表示
  - 多言語対応（現時点では、日本語と英語のみ）
- セルの前方参照
  - セルのプロトタイプ宣言が不要
- 固定結合 (旧称 逆 require)
  - 初期化のための結合の記述が不要
- region によるモジュールの分割
- namespace による名前衝突の回避
- RPCプラグイン
  - エラー回復モデル、メモリリークしないコードの生成など製品レベル
- プラグインの拡充 (従来 throughプラグインのみ)
  - シグニチャプラグイン … RubyTECSBridgePlugin
  - セルタイププラグイン … ATK*Plugin
  - セルプラグイン … RepeatCellPlugin, RepeatJoinPlugin

エラーメッセージの日本語表示(多言語対応)

- 現時点では、日本語と英語のみ
- 言語の決定順序
  - Codepage (exerb 版のみ, chcp コマンドで変更可)
  - LANG 環境変数 (非exerb版のみ)
  - TECSGEN_LANG 環境変数
  - TECSGEN_FILE_LANG 環境変数 (ファイルのみ)
  - -k オプション (ファイルのみ)

  - 上から順に評価し、一番最後に指定されたものが有効となる
  - Cygwin コンソールでexerb版を動作させると、codepage に従う

エラーメッセージの日本語表示(多言語対応)

- CygwinやLinux 環境では、LANG=ja_JP.UTF-8 がデフォルト. ファイルの文字コード euc JPの場合、以下のように設定するとよい
  - export TECSGEN_FILE_LANG=ja_JP.eucJP
  - または –k euc を指定する
- cygwin で、exerb 版でファイルの文字コード sjis の場合
  - 言語のための設定は不要
  - ファイルの文字コードを eucJP とするには、上記と同じように指定する

セルの前方参照

- 従来：TECS CDL は、前方参照できない（一部例外を除く）
  - typedef, const, struct
  - signature, celltype, composite, cell

- V1.1.0：cell の前方参照を可能とした
  - cell のプロトタイプ宣言が不要となった
  - ただし、プラグインで生成されるセルは、評価順によって、プロトタイプ宣言が必要

固定結合(旧称: 逆 require)

- 初期化のための結合の記述を不要にできる
  - require の逆機能

```
[singleton,active]
celltype tInitializer {
   call sInitialize cInit[];        // 添数なしの呼び口配列
};
cell tInitializer Initializer{ };   // ← Cell1, Cell2 へ結合される

celltype tCelltype {
      entry sInitialize eEntry <= tInitializer . cInit;  // ←固定結合
};
cell tCelltype Cell1 { };       // Initializer から呼び出される
cell tCelltype Cell2 { };       // Initializer から呼び出される
```

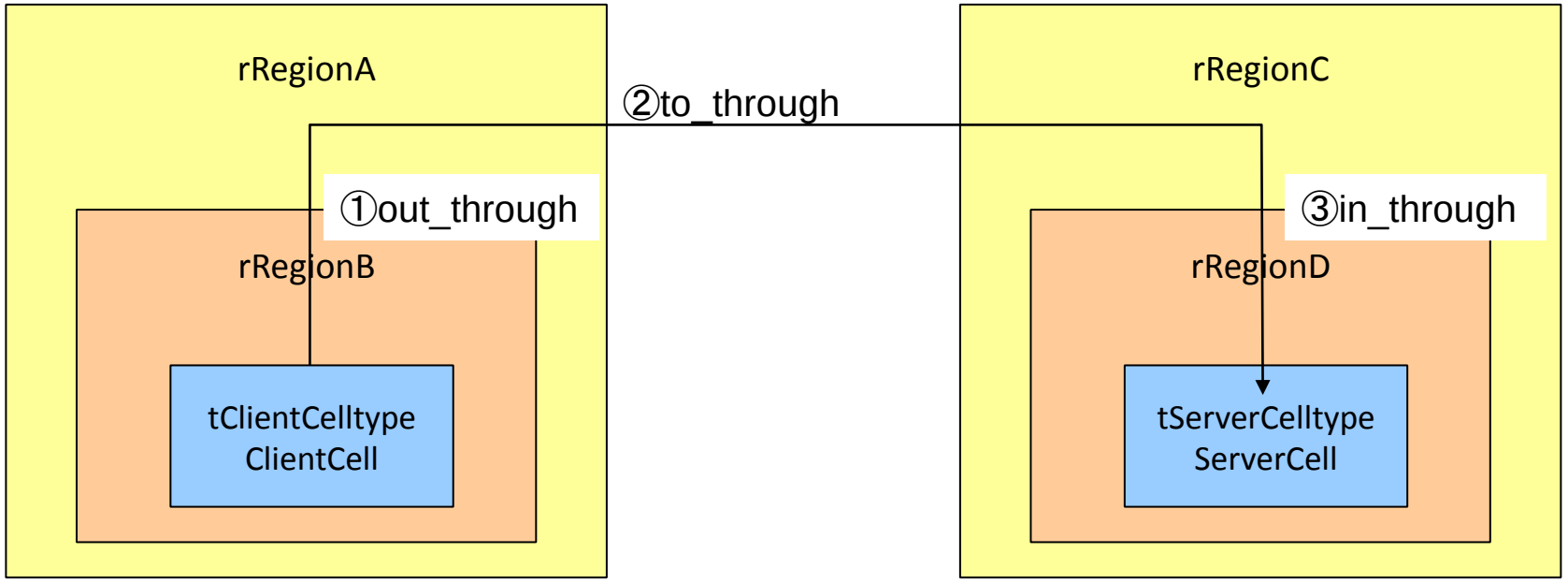
region によるモジュールの分割

- region の指定子 node, linkunit は、独立したリンク単位であることを表す
  - 別のregion のセルとは直接結合(リンク)できないことを表す
    - OpaqueRPC プラグインなどで分離する
  - 他の指定子 domain, class も追加になった
    - TOPPERSの仕様に合わせたもの
    - 現時点では記述しても、効果はない
  - Makefile は、リンク単位ごとに生成される
    - 一回の tecsgen 実行で、すべてのリージョンのコードが生成される
    - -G は、意味が変更された
      - 指定されている場合、指定されたリージョンのコードのみ生成

region によるモジュールの分割

- region は namespace の機能を含む
  - 名前衝突の回避
    - region 間の結合には、namespace名が必要になった
    - 子region から親region へは namespace 名は不要
- to_through の階層では、in_through, out_through が適用されなくなった
  - 次ページの例では、RegionA の out_through, RegionC の in_through が適用されなくなった
  - from_through が必要だが、未対応

region 間の through のプラグイン



プラグインが生成するセルの region は、以下のように整理された
プラグインは、start_region, end_region のいずれか、または両方にセルを生成してもよい。preferred_region は、いずれか一方にのみセルを生成する場合、より望ましいリージョンである（連接しているとき、変更される可能性ある）。

	through_type	start_region	end_region	preferred_region
①	out_through	rRegionB	rRegionA	rRegionB
②	to_through	rRegionA	rRegionC	rRegionA
③	in_through	rRegionC	rRegionD	rRegionD

region によるモジュールの分割

ビルド用の Makefile.templ は、以下の構造を想定して出力される
(rRegion1 と rRegion2 の下にセルが置かれる場合)

```
build_root/   <===  CDL を置くディレクトリ、ルートリージョンをビルドするディレクトリ
    |
    +--  opaqueRPC.cdl   (CDL ファイル)
    |
    +-- src  <===  C のソースコードを置くディレクトリ
    |
    +--  Makefile  … (1)  <=== tecsgen を実行する規則が含まれる
    |
    +-- rRegion1 <===  rRegion1 をビルドするディレクトリ
    |     |
    |     +-- Makefile … (2)
    |
    +-- rRegion1_rSubRegion2 <===  rRegion1_rSubRegion2 をビルドするディレクトリ
    |     |
    |     +-- Makefile … (3)
    |
    +--  gen   <==== tecsgen が出力ファイルを置くディレクトリ
          |
          +-- Makefile.templ  ⇒ (1) へ移動して使う、tecsgen を実行する規則が含まれる
          |
          +-- rRegion1
          |      |
          |      +-- Makefile.templ ⇒ (2) へ移動して使う
          |
          +-- rRegion1_rSubRegion2
                 |
                 +-- Makefile.templ ⇒ (3) へ移動して使う
```

(1), (2), (3) へ移動して使用する場合、パス調整は不要となるように Makefile.templ は生成される。（パス以外の調整は必要）。

namespace による名前衝突の回避

- signature, celltype を namespace の下に置くことができる
  - namespace の役割：上記の名前の衝突回避
  - typedef, const, struct を namespace の下に置くことはできない
    - C言語レベルでは define による名前衝突回避を行っているが、これらは define で名前衝突回避が難し
  - cell は namespace の下ではなく region の下に置く

- namespace と region の違い
  - namespace … 名前衝突の回避
  - region … cell の配置を制御、名前衝突の回避

OpaqueRPCPlugin

- OpaqueRPCPlugin: メモリ空間が異なる場合に用いるRPC (リモート呼び出し) のプラグイン
- 早期リリース版からの変更
  - region 仕様変更への対応
  - エラー回復モデルへの対応
    - V1.0.2.32 の仕様書　図6.9～6.11 および test/opaqueRPC 下のテストコード参照
  - エラー時のデアロケータ DEALLOC_RESET
    - エラーによる打ち切りでも send/receive がアロケートしたメモリのリークを防ぐ
  - オプション引数の変更、拡充
  - 複数の結合への対応
    - to_through で複数の結合をより扱いやすくした

RPCプラグインのオプションサマリー

- OpaqueRPCPlugin などオプションの拡充を図った

OpaqueRPCPlugin, SharedOpaqueRPCPlugin オプションサマリー

```
   "noClientSemaphore = false, " // OpaqueRPCPlugin のみ (shared の場合、セマフォは必須)
   "clientChannelCelltype=tSocketClient, "
   "serverChannelCelltype = tSocketServer, "
   "clientChannelCell = ... ,"   // デフォルトは Join_... のような名前(ジェネレータが自動生成)
   "serverChannelCell = ... ,"   // 同上
   "clientChannelInitializer = !portNo=8931+$count$; serverAddr=¥"127.0.0.1¥";!, "   // ポート番号とアドレス
   "serverChannelInitializer = !portNo=8931+$count$;!, "
   "clientErrorHandler = %ClientRPCErrorHanler.eHandler%, " // エラーハンドラーセル.受け口 (デフォルトは未指定)
   "serverErrorHandler = %ServerRPCErrorHanler.eHandler%, " // エラーハンドラーセル.受け口 (デフォルトは未指定)
   "clientSemaphoreCelltype = tSemaphore"
   "clientSemaphoreInitializer =  !count = 1; attribute = C_EXP( ¥"TA_NULL¥" );! ,"
   "TDRCelltype = tNBOTDR, "      // network byte order で送受信する TDR
   "PPAllocatorSize = N, "        // N は正の整数値 (デフォルトは非設定で、PPAllocator は生成されない)
   "substituteAllocator = %Alloc.eAlloc=>CAlloc.eAlloc%, "   // デフォルトは非設定、呼び口側のアロケータ名を置換する
   "noServerChannelOpenerCode = false, "   //  サーバーチャンネルの開閉コードの生成
   "taskCelltype = tTask, "                //  サーバー側タスクセルタイプ名
   "taskPriority = 11, "                   //  サーバー側タスクプライオリティ
   "stackSize = 4096,"                     //  サーバー側タスクスタックサイズ
```

RPCPlugin, sharedRPCPlugin オプションサマリー

```
*  "noClientSemaphore, "
*  "semaphoreCelltype = tSemaphore, "
   "taskPriority, "
   "channelCelltype, "
   "TDRCelltype, "
   "channelCellName"
   "PPAllocatorSize"
```

* は RPCPlugin のみ対応

through プラグインのオプションの名前置換

- through プラグインに限定の機能
  - OpaqueRPCPlugin, RPCPlugin, TracePlugin など
  - region 間で複数の結合がある場合、プラグインが生成するセル名やポート番号の衝突を避けるのに用いることができる

```
$source$        … 呼び元のセル名
$destination$   … 呼び先のセル名
$SOURCE$        … 呼び元のセル名（リージョン名を '_' で連結した global_name）
$DESTINATION$   … 呼び先のセル名（リージョン名を '_' で連結した global_name）
$next$          … 次のセル名
                  複数の through がつながっている場合、すぐ後ろに来るもの
$NEXT$          … 次のセル名（リージョン名を '_' で連結した global_name）
                  複数の through がつながっている場合、すぐ後ろに来るもの
$start_region$  … $source$ のセルの存在する region (global_name)
$end_region$    … $destination$ のセルの存在する region (global_name)
$preferred_region$  … 適切な region (global_name), start_region または end_region
$count$         … region 間の through の適用数
$$              … $ に置換
```

プラグインの拡充・セルの id 指定子

- シグニチャプラグイン
  - singnature に対して適用
  - 例) RubyTECSBridgePlugin
- セルタイププラグイン
  - セルタイプに適用
  - 例) ATK*Plugin
- セルプラグイン
  - セルに適用
  - 例) RepeatCellPlugin, RepeatJoinPlugin
- セルの id 指定子
  - セルの定義順とは異なる順序で id を付けたい
  - ATK プラグインで使用