TLVアプリログ機能マニュアル

TLV開発チーム

平成21年10月1日

目 次

0.1 概要

アプリログ機能を用いることで、次の 2 つのことができるようになります。

- 文字列可視化
- ユーザ定義状態可視化

これらを行った例が、図 1 です。

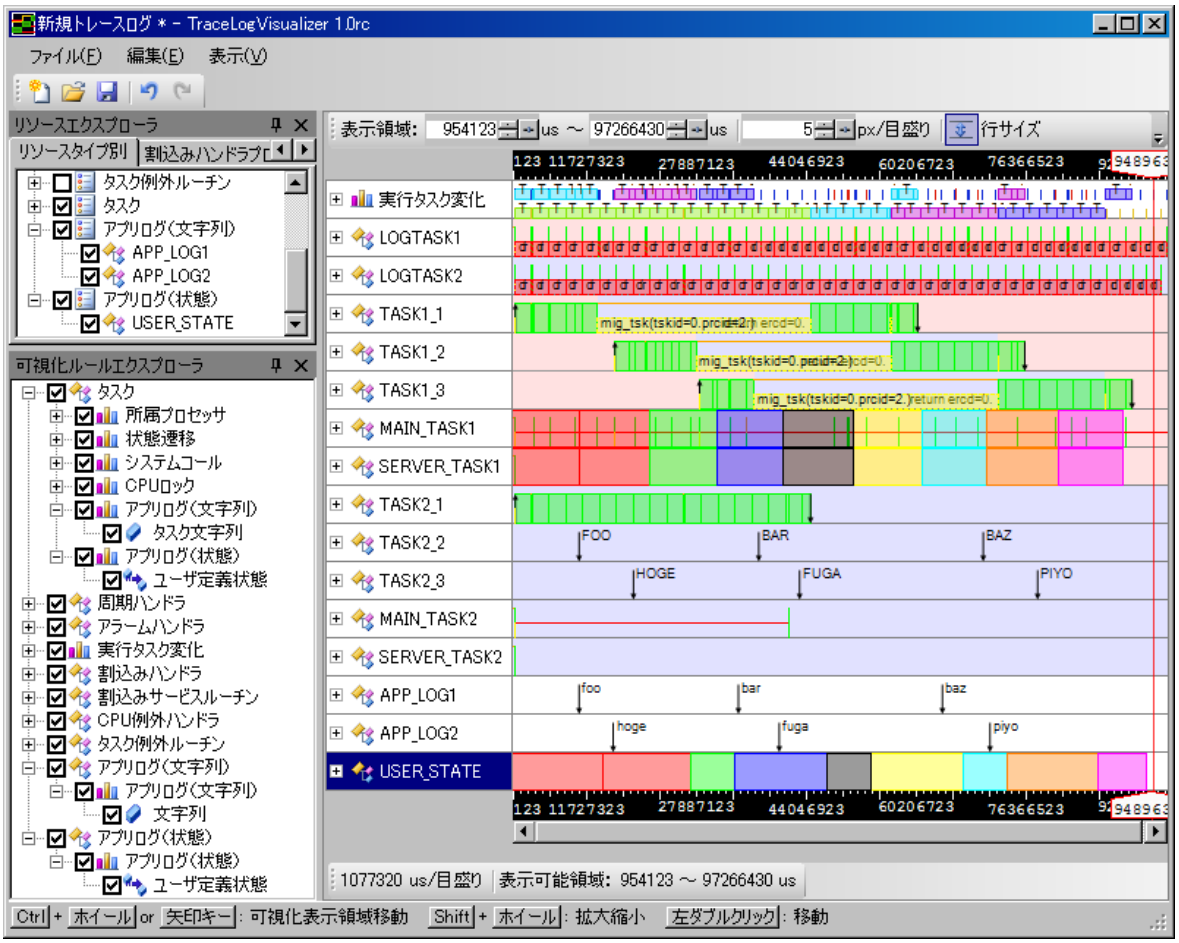

図 1: 全体スクリーンショット

また、本機能は TOPPERS/ASP および FMP に対応しています。

0.2　文字列可視化

文字列可視化機能を使うことで、TLV 上に任意の文字列を表示させることができます。これにより、printf デバッグが可能になりました。

文字列可視化には 2 種類あります。**文字列可視化**と、**タスク文字列可視化**です。**文字列可視化**では、独立した一つの行に文字列を並べていきます。こちらが一般的な利用方法です。図 2 のように可視化されます。**タスク文字列可視化**では、タスクの一属性として文字列を可視化します。タスクに関係することを出力・可視化したい場合は、こちらを利用した方がわかりやすいでしょう。図 3 のように可視化されます。

以下に、使い方を説明します。

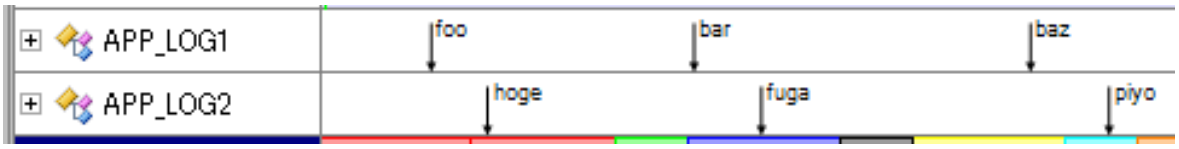

図 2: 文字列可視化

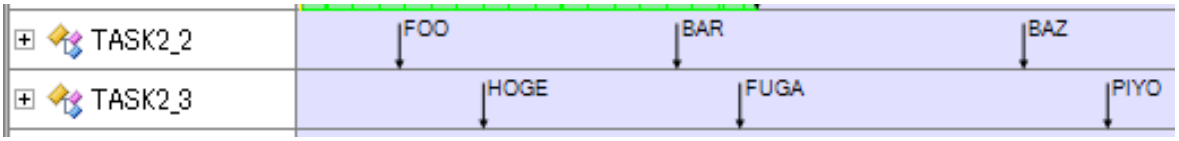

図 3: タスク文字列可視化

0.2.1　文字列可視化

リソースファイルへの追記　まず、リソースファイルにリスト 1 の内容を追記します。*APP_LOG1* という部分は可視化時にリソース名として表示されます。好きな名前に変更してください。*"id":"1"*という部分では、この行の ID を設定しています。ログ出力時に参照しますので、わかりやすい名前に変更してください。ただし、:（コロン）とスペースは含めないでください。

id を変えて複数の記述を登録することで、複数の行で文字列の可視化を利用できます。2 つ登録を行い、一つでセマフォの数、もう一方で変数の値を追跡するなどの利用が可能です。

Listing 1: リソースファイルに追記する内容

```
1 "APP_LOG1":{
2         "Type":"ApplogString",
3         "Attributes":
4         {
5                 "id":"1"
6         }
7 },
```

アプリケーションにログ出力を追加　TOPPERS カーネルのユーザアプリケーションにおいて syslog 関数を呼び出し、アプリログが用いるログを出力させます。

リスト 2 のように syslog 関数を呼ぶことで、アプリログ機能が利用するフォーマットでログを出力することができます。*rid* は、リソースファイルに記述した id を指定します。*str* には、可視化したい文字列を指定します。ただし、.（ピリオド）は含めないでください。

Listing 2: syslog 関数に指定するフォーマット

```
1 syslog("applog str : ID %s : %s.",rid,str);
```

アプリケーションの実行　アプリケーションを実行して、ログを取得してください。

TLV へのログ読み込み　TLV にログファイルとリソースファイルを読み込ませてください。

0.2.2　タスク文字列可視化

文字列可視化との違いは、次の 2 点です。

- リソースファイルへの追記が不要
- ログフォーマット

アプリケーションにログ出力を追加　タスク文字列可視化の場合、リスト3のようにsyslog関数を呼びます。*tid*では、タスクのidを指定します。

Listing 3: syslog関数に指定するフォーマット

```
1 syslog(”applog strtask : TASK %s : %s.”,tid,str);
```

0.2.3　出力例

tlv/sampleFiles/fmp_app.logおよびfmp_app.resを読み込ませたときの出力が図4です。リソースエクスプローラに**アプリログ（文字列）**というチェックボックスがあり、これのON/OFFでタスク文字列可視化の有無を切り替えることができます。可視化ルールエクスプローラの**タスク** → **アプリログ（文字列）**チェックボックスのON/OFFにより、タスク文字列可視化の有無を切り替えることができます。

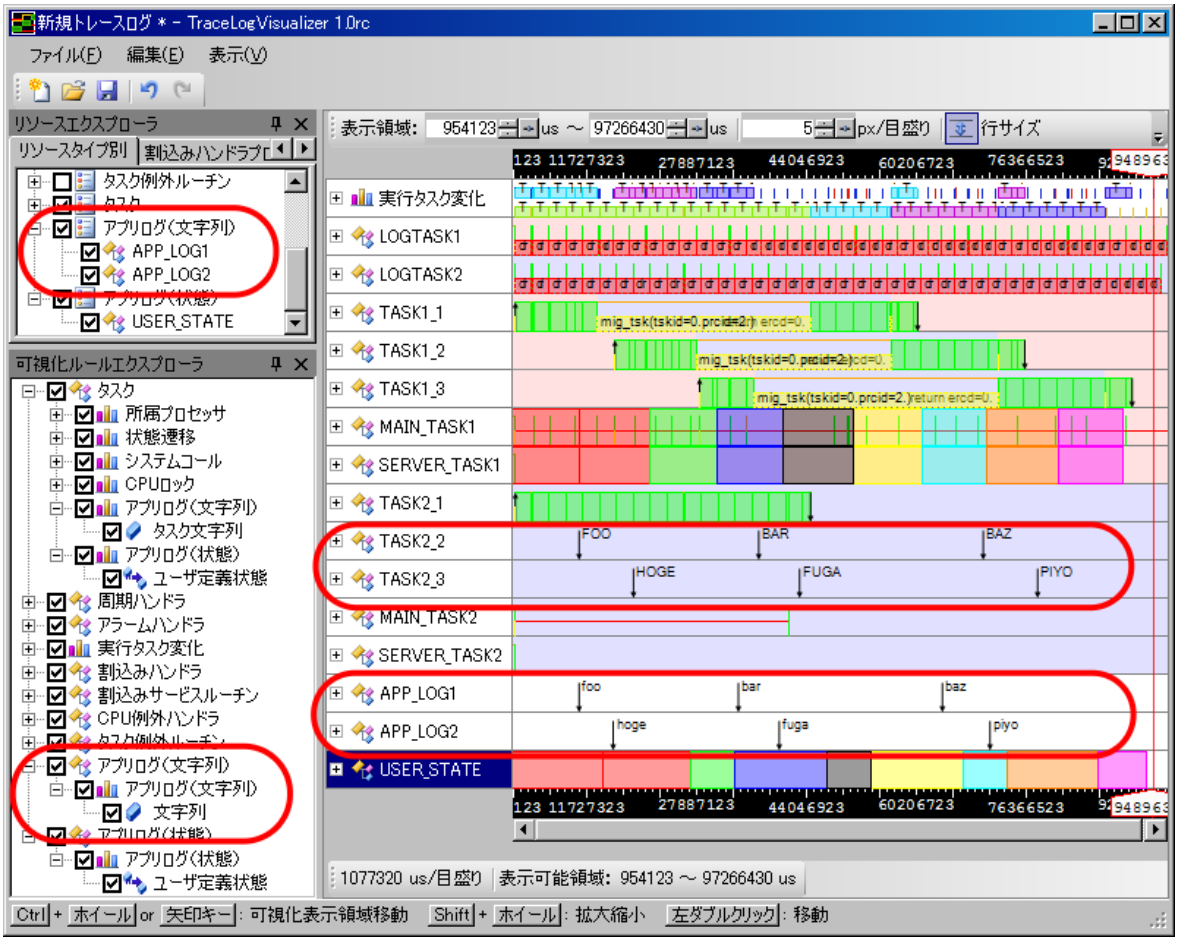

図 4: 文字列可視化スクリーンショット

0.3　ユーザ定義状態

ユーザ定義状態機能を使うことで、任意の状態を定義し、可視化することができます。ユーザ定義状態可視化にも、**ユーザ定義状態可視化**と**タスクユーザ定義状態可視化**の区分が存在します。

0.3.1　ユーザ定義状態可視化

リソースファイルへの追記　まず、リソースファイルにリスト 4 の内容を追記します。*USER_STATE* という部分は可視化時にリソース名として表示されます。好きな名前に変更してください。*"id":"1"*という部分では、この行の ID を設定しています。ログ出力時に参照しますので、わかりやすい名前に変更してください。ただし、:（コロン）とスペースは含めないでください。

id を変えて複数の記述を登録することで、複数の行でユーザ定義状態の可視化を利用できます。2 つ登録を行い、それぞれで別のユーザ定義状態の可視化を利用できます。

Listing 4: リソースファイルに追記する内容

```
"USER_STATE":{
        "Type":"ApplogState",
        "Attributes":
        {
                "id":"1"
        }
}
```

アプリケーションにログ出力を追加　syslog 関数を用いて、TLV が読み込むフォーマットでログを出力させます。リスト 5 のように syslog 関数を呼ぶことで、アプリログ機能が利用するフォーマットでログを出力することができます。*rid* は、リソースファイルに記述した id を文字列で指定します。*state* には、状態の色を数字で指定します。0 から 7 の 8 状態がすでに定義してありますので、それを利用するか、/visualizeRules/toppers_rules.viz を編集して好きな色を定義してください。

Listing 5: syslog 関数に指定するフォーマット

```
syslog("applog state : ID %s : %d.",rid,state);
```

アプリケーションの実行　アプリケーションを実行して、ログを取得してください。

TLV へのログ読み込み　TLV にログファイルとリソースファイルを読み込ませてください。

0.3.2　タスクユーザ定義状態可視化

ユーザ定義状態可視化との違いは、次の 2 点です。

- リソースファイルへの追記が不要
- ログフォーマット

アプリケーションにログ出力を追加　タスク文字列可視化の場合、リスト 6 のように syslog 関数を呼びます。*tid* では、タスクの id を数字で指定します。

Listing 6: syslog 関数に指定するフォーマット

```
1 syslog(”applog statetask : TASK %d : %d.”,tid,state);
```

改訂履歴

版番	日付	更新内容	更新者
1.0	09/9/9	新規作成	柳澤大祐