Btrieve 2 SDK を Raspberry Pi 4 で使用するためのサンプルコード
【C++ と Python に対応】

時系列データを Raspberry Pi デバイスの内部センサーから取り込み、Actian Zen データベースに書き込むための、Btrieve 2 を使用した簡単なプログラムを紹介します。ハードウェアを追加することなく、Raspberry Pi だけでサンプルコードを実行できるように、Raspberry Pi 自体に含まれるセンサーのみ使用しています。

書き込まれたデータの内容を、コンソールに表示させることもできます。本プログラムで紹介した Btrieve 2 関数、操作方法等は、他にも多くの場面で再利用することができます。

本プログラムでは Raspberry Pi を使用していますが、Actian Zen と Btrieve 2 は、32 ビットおよび 64 ビット ARM Linux ベースのデバイスや、Intel 32 および 64 ビットベースの CPU にも対応しています。さらに、Linux 以外にも、Windows、macOS、Android および iOS の各 OS がサポート対象となります。これらの全ての環境で、同じデータファイル、データ型、Btrieve 2 のオペレーションを使用することができます。

【データファイルのレコードレイアウト】

本プログラムは、Raspberry Pi の CPU 温度、クロック周波数、使用率を定期的にセンサーで読み取り、下記のレコードレイアウトで Actian Zen のデータファイルに書き込みます：

フィールド名	サイズ	データ型
記録時間	8	AUTOTIMESTAMP
CPU 温度	8	DOUBLE
CPU クロック周波数	8	DOUBLE
CPU 使用率	8	DOUBLE

AUTOTIMESTAMP データ型を定義することによって、Actian Zen のエンジンが自動的に「記録時間」フィールドに対して、レコードが挿入された時間をナノ秒単位の精度で記録します。

【実行環境】

本プログラムは、32 ビット版 Raspberry Pi OS と Actian Zen v15 Edge Server がインストールされた Raspberry Pi 3/4 デバイス上で動作するように設計されています。
Actian Zen Edge Server のインストール手順と Btrieve 2 の C++と Python プログラムのビルド及び実行の手順については、下記の URL からダウンロードできる「Raspberry Pi 3/4 (Raspberry Pi OS) での Actian Zen v15 Edge 使用方法」の資料を参照して下さい。

▼ https://www.agtech.co.jp/actian/support/reference/

【ZIP ファイルの内容】

同じ機能を持つサンプルコードを C++ (TimeSeries_AutoTS.cpp) と Python (TimeSeries_AutoTS.py) の 2 つの言語で提供しています。

また、パフォーマンスを向上させるための Extended オペレーションを使用したサンプルコードのバージョンもあります (TimeSeries_AutoTS_bulk.cpp と TimeSeries_AutoTS_bulk.py) 。

【実行方法】

コンパイル後、C ++のサンプルプログラムは次のようにコマンドラインで実行できます。

```
./TimeSeries_AutoTS
```

また、Python のサンプルプログラムは次のようにコマンドラインで実行できます。

```
python3 TimeSeries_AutoTS.py
```

プログラムの動作は、次のコマンドラインパラメータで制御できます。

-b データベースからデータを読み取り、表示する開始時刻
（フォーマット mm/dd/yy HH:MM:SS.ff を使用して下さい）
-d <秒> ミリ秒単位の監視遅延
（デフォルト= 10）

-e データベースからデータを読み取り、表示する終了時刻
（フォーマット mm/dd/yy HH:MM:SS.ff を使用して下さい）
-h ヘルプを表示
-n データの記録回数
-r データベースからデータを読み取り、表示する
-s サイレント出力、データ記録のみ

【実行例】

1) センサーからデータを読み取り、Btrieve データファイルに記録する

＜C++の場合＞
./TimeSeries_AutoTS.py -d 50 -n 1000

をコマンドラインで実行します。

＜Python の場合＞
python3 TimeSeries_AutoTS.py -d 50 -n 1000

をコマンドラインで実行します。

センサーから 1000 回読み取りを行い、それぞれの読み取りの間に 50 ミリ秒 Wait します。-s がないため、データファイルだけではなく、コンソールにもデータを記録します。

2) Btrieve ファイルからすべてのデータを読み取り、コンソールに表示する

＜C++の場合＞
./TimeSeries_AutoTS.py -r

をコマンドラインで実行します。

＜Python の場合＞
python3 TimeSeries_AutoTS.py -r

をコマンドラインで実行します。

3) Btrieve ファイルから開始時刻（-b パラメータで設定）と終了時刻（-e パラメータで設定）の間のデータを読み取ってコンソールに表示する

＜C++の場合＞
./TimeSeries_AutoTS -r -b "12/27/21 19:05:24.65" -e "12/27/21 19:05:27.05"
./TimeSeries_AutoTS -r -b "12/27/21 19:05:24.65" -e "12/27/21 19:05:27.05"

のようなコマンドをコマンドラインで実行します。

＜Python の場合＞
python3 TimeSeries_AutoTS.py -r -b "12/27/21 19:05:24.65" -e "12/27/21 19:05:27.05"

のようなコマンドをコマンドラインで実行します。

【SQL アクセス】

下記 SQL の実行によって、サンプルコードが記録するデータに対して、SQL インタフェース経由でアクセスできます。

```
SET TRUENULLCREATE = OFF;
CREATE TABLE "TimeSeries_AutoTS" IN DICTIONARY USING
'TimeSeries_AutoTS.mkd' (
       "記録時間" AUTOTIMESTAMP DEFAULT '0',
       "CPU 温度" DOUBLE,
       "CPU クロック周波数" DOUBLE,
       "CPU 使用率" DOUBLE);
SET TRUENULLCREATE = ON;
```