Boost+ライブラリー周の旅

ver.1.44.0

高橋晶 (Akira Takahashi)

ブログ:「Faith and Brave - C++で遊ぼう」
http://d.hatena.ne.jp/faith_and_brave/

Boost.勉強会#2 2010/09/11(Sat)

Boost C++ Libraries

はじめに

前回は、Boost 1.40.0までを紹介しました。

今回は1.44.0までの差分を紹介します。

Boost C++ Libraries

本日紹介するライブラリ

1. Property Tree
2. Uuid
3. Range 2.0
4. Filesystem v3
5. Polygon
6. Meta State Machine(MSM)

Boost C++ Libraries

Property Tree 1/4

汎用的な、木構造をもつデータのプロパティ管理。
XML、JSON、INIファイルのパーサーを提供している。

全てのデータは、
`boost::property_tree::ptree`型
に対して操作を行う。

値の取得には、
失敗時に例外を投げる`ptee::get<T>()`と
`boost::optional`を返す`ptree::get_optional<T>()`
が用意されている。

Property Tree 2/4

XMLの読込、要素、属性の取得。
XMLパーサーにはRapidXmlを採用している。

```
<root>
  <elem attr="World">
    Hello
  </elem>
</root>
```

```
using namespace boost::property_tree;

ptree pt;
read_xml("test.xml", pt, xml_parser::trim_whitespace);

// 要素の取得
const string& elem = pt.get<string>("root.elem");

// 属性の取得 : <xmlattr>という特殊な要素名を介してアクセスする
const string& attr = pt.get<string>("root.elem.<xmlattr>.attr");
```

Property Tree 3/4

JSONの読込、データの取得。

```
{
  "Data": {
    "Value": 314,
    "Str": "Hello"
  }
}
```

```
using namespace boost::property_tree;

ptree pt;
read_json("test.json", pt);

const int     value = pt.get<int>("Data.Value");
const string& str   = pt.get<string>("Data.Str");
```

Property Tree 4/4

iniの読込、データの取得。

```
[Data]
Value = 314
Str = Hello
```

```
using namespace boost::property_tree;

ptree pt;
read_ini("test.ini", pt);

const int     value = pt.get<int>("Data.Value");
const string& str   = pt.get<string>("Data.Str");
```

Boost C++ Libraries

Uuid

ユニークIDの生成。
COMとか、分散環境での情報の識別とかで
使われることが多い。

```
using namespace boost::uuids;

// 擬似乱数生成器でのUUID生成。デフォルトはmt19937
uuid u1 = random_generator()();

// 文字列からUUID生成
uuid u2 = string_generator()("0123456789abcdef0123456789abcdef");

cout << u1 << endl;
cout << u2 << endl;
```

```
31951f08-5512-4942-99ce-ae2f19351b82
01234567-89ab-cdef-0123-456789abcdef
```

Range2.0 1/2

ユーティリティ程度だったBoost.Rangeに、
Rangeアルゴリズムと Rangeアダプタを拡張。

```
std::vector<int> v;

// Rangeアルゴリズム : イテレータの組ではなく範囲を渡す
boost::sort(v);
boost::for_each(v, f);

// Rangeアダプタ
using namespace boost::adaptors;
boost::for_each(v | filtered(p) | transformed(conv), f);
```

Rangeアルゴリズム : STLアルゴリズムのRange版
Rangeアダプタ : 遅延評価され、合成可能な範囲操作

Range 2.0 2/2

Boost.Foreachと組み合わせて使っても便利。

```
using namespace boost::adaptors;
std::map<std::string, int> m;

// mapのキーのみを操作
BOOST_FOREACH (const std::string& key, m | map_keys) {
  // something...
}

// mapの値のみを操作
BOOST_FOREACH (const int value, m | map_values) {
  // なにか・・・
}
```

Rangeライブラリとしてはovenも強力なのでそちらもチェックしてください！

Filesystem v3

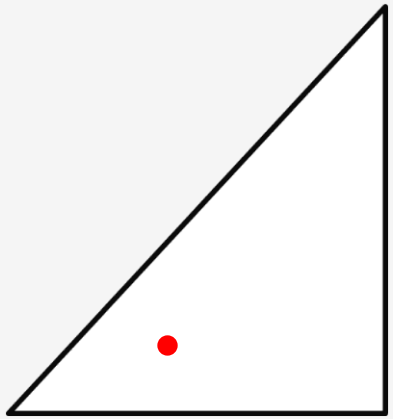
pathの日本語対応等。
stringとwstringの両方を使用するためにオーバーロードが必要なくなったり。

```
#define BOOST_FILESYSTEM_VERSION 3
#include <boost/filesystem.hpp>

void foo(const boost::filesystem::path& path) {}

int main()
{
    foo("english");
    foo(L"日本語"); // v2ではエラー
}
```

Polygon

平面多角形(2D)のアルゴリズムを提供するライブラリ。

以下は、三角形の内外判定。



```
#include <boost/polygon/polygon.hpp>
namespace polygon = boost::polygon;

int main()
{
    const std::vector<polygon::point_data<int>> ptrs = {
        {0, 0}, {10, 0}, {10, 10}
    };
    const polygon::polygon_data<int> poly(ptrs.begin(), ptrs.end());

    // 点が三角形の内側にあるか
    const polygon::point_data<int> p(3, 3);
    assert(polygon::contains(poly, p));
}
```

Boost C++ Libraries Meta State Machine(MSM) 1/2

新たな状態マシンライブラリ。状態遷移表を直接記述する。

```
namespace msm = boost::msm;
struct Active  : msm::front::state<> {};
struct Stopped : msm::front::state<> {};
struct StartStopEvent {};
struct ResetEvent {};

struct StopWatch_ : msm::front::state_machine_def<StopWatch_> {
    typedef Stopped initial_state;
    struct transition_table : boost::mpl::vector<
//           Start    Event           Next
        _row<Active,  StartStopEvent, Stopped>,
        _row<Active,  ResetEvent,     Stopped>,
        _row<Stopped, StartStopEvent, Active>
    > {};
};

typedef msm::back::state_machine<StopWatch_> StopWatch;
```

Boost C++ Libraries Meta State Machine(MSM) 2/2

新たな状態マシンライブラリ。状態遷移表を直接記述する。

```
int main()
{
    StopWatch watch;

    watch.start();
    watch.process_event(StartStopEvent()); // stop -> run
    watch.process_event(StartStopEvent()); // run  -> stop
    watch.process_event(StartStopEvent()); // stop -> run
    watch.process_event(ResetEvent());     // run  -> stop
}
```

Boost C++ Libraries

まとめ(?)

- 1.44.0になってライブラリがかなり充実しました。
- とくにRange 2.0はプログラミングスタイルを変えるほどのライブラリなのでオススメです。