Boostライブラリー周の旅

Ver 1.59.0 ～ 1.60.0

高橋 晶(Akira Takahashi)
faithandbrave@gmail.com

2015/12/05 Boost.勉強会 #19 東京

はじめに

この発表は、以下のような方を対象にして、

- Boostに興味はあるけど、触ったことがない
- バージョンアップについていけなくなった
- Boostの全容を知りたい

Boost 1.60.0時点での、なるべく全てのライブラリの
概要を知ってもらうためのものです。
この資料は、バージョン間の差分を紹介するものですが、
これまでの資料をマージしたバージョンも公開します。

バージョンアップの詳細

- boostjpサイトでは、Boostのリリースノートを翻訳したものを公開しています
- http://boostjp.github.io/document/version.html
  この階層以下に、各バージョンのリリースノートがあります
- 本家のリリースノートよりもくわしく書いています
  - チケットタイトルを直訳するのではなく、問題と修正の内容を確認して書いています
  - 新機能が入った際には、ドキュメントから概要とサンプルコードを持ってきたりもしています
  - 本家より優れたものを作ることが目的ではなく、日本語で情報を届けるのに適したやり方として情報補完をしています

Boostとは

- 標準ライブラリに満足できなかった人たちが作っている、C++の準標準ライブラリ。
- Boostから標準ライブラリに、多くの機能が採用されている
- 普段のプログラミング全般で使える基本的なものから、専門的なものまで、いろいろなライブラリがある。
- ライセンスはBoost Software License 1.0
  - 無償で商用利用可能
  - 著作権表記の必要なし
  - ソースコードの改変自由

本日紹介するライブラリ

- Convert
- VMD
- Test V3

質問は随時受け付けます

- この発表では、3ライブラリの紹介をします。
- 1ライブラリに付き、(ほぼ)ひとつのサンプルコードで解説する、というスタイルです。
- 発表時間に余裕があるので、随時質問してください。

Convert 1/3

boost::lexical_cast()を置き換えて使用できる、数値と文字列の型変換ライブラリ。
変換失敗時の挙動、基数や精度、フォーマットなどを設定できる。
戻り値の型はboost::optional<T>。

```
namespace cnv = boost::cnv;

// 文字列"123"をint型に変換する
// コンバータにはcstreamを使用する
cnv::cstream converter;
boost::optional<int> result =
                                boost::convert<int>("123", converter);
```

コンバータクラスはいくつか用意されており、現在正式にサポートされているのは、lexical_castと(c|w)streamの2つ。
コンバータを別定義できるので、文字コードの変換にも利用できる。

Convert 2/3

変換失敗時のエラーハンドリングは基本的に、
戻り値であるboost::optional<T>オブジェクトに対して行う。

```
boost::cnv::cstream converter;

// 変換後にoptionalの中身を取り出す。
// 変換失敗時はboost::bad_optional_access例外が送出される
int result2 = boost::convert<int>("123", converter).value();

// 変換失敗時に-1が返される
int result3 = boost::convert<int>("xxx", converter).value_or(-1);
```

Convert 3/3

基数や精度、フォーマットの設定は、標準ライブラリのマニピュレータと、
Boost.Convertが定義しているパラメータの、2種類が使用できる。
マニピュレータに指定は、コンバータの関数呼び出し演算子を使用する。

```
// 標準のマニピュレータ
cnv::cstream converter;
int result1 = boost::convert<int>(
    "  ff",
    converter(std::hex)(std::skipws) // 16進数、スペースを無視
).value(); // result == 255

// Boost.Convertのマニピュレータ
int result2 = boost::convert<int>(
    "ff",
    converter(arg::base = cnv::base::hex) // 16進数
).value(); // result == 255
```

VMD (Variadic Macro Data Library)

可変引数マクロを使用した、プリプロセッサメタプログラミングのライブラリ。Boost.Preprocessorを強化するためのもの。
各種データ型と、それをテスト・解析する機能が提供される。

```
#define SEQ 1 2 3
#define SEQ_SIZE BOOST_VMD_SIZE(SEQ)

void f(int a, int b, int c)
{ std::cout << a << " " << b << " " << c << std::endl; }

// 要素数を取得
std::cout << SEQ_SIZE << std::endl; // 3

// シーケンスをカンマ区切りパラメータに変換
f(BOOST_VMD_ENUM(SEQ));// 「1, 2, 3」に変換される
```

Test v3

Boost.Testがバージョン3にメジャーアップデートした。

汎用的なテストマクロBOOST_TESTが追加された(Power Assert)。

パラメタライズドテストに対応した。ドキュメントが読みやすくなった。

```
BOOST_AUTO_TEST_CASE(equal_test)
{
  int a = 1;
  int b = 2;
  BOOST_TEST(a == b); // これまではBOOST_CHECK_EQUAL(a, b)と書いてた
}
```

```
Running 1 test case...
main.cpp:8: error: in "equal_test": check a == b has failed [1 != 2]

*** 1 failure is detected in the test module "example"
```

本日の紹介はここまで

- 今回の「Boostライブラリー周の旅」では、
  Boost 1.59.0から1.60.0までの更新を紹介しました。

- 発表では差分のみを紹介していますが、
  これまで紹介したものをマージした資料も公開しています。

- 今回Boostに興味を持たれた方は、そちらのマージした資料もぜひご覧ください。