C++ではまる

boost勉強会 2011/2/26

サイボウズ・ラボ 光成滋生

自己紹介

C++好き，Haskellいまいち

gccとVCネタ

http://homepage1.nifty.com/herumi/prog/gcc-and-vc.html

C++は難しい

処理系依存とか

- "処理系のバグ"と騒いでも現実問題を解決しないと
  - まあ，大抵は自分のバグだが

boostも難しい

- コードを追いかけるのとかデバッグとか

自分や周囲ではまったことの中からいくつかピックアップしてみる

- 対象は主にgccやVC
- 易しいところから

Q. getRate(0) = ?

```
double getRate(double speed) {
    const double limit = 1.0; // 閾値
    const double power = 3.5; // エンジン性能
    if (speed < limit) return 0;
    return power / speed;
}
```

printf("%f¥n", getRate(0)); // Infと表示された

なぜ?

A. ソースがShift-JISだった

gccではこう見えてる

```
double getRate(double speed){
    const double limit = 1.0; // ????
    const double power = 3.5; // ????¥     ← バックスラッシュ
    if (speed < limit) return 0;
    ...
```

次の行と結合されてコメントアウトされた

```
double getRate(double speed) {
    const double limit = 1.0; // ????
    const double power = 3.5; // ????¥ if (speed < limit)
return 0;
    ...
```

解決法

ソースコードはEUC-jpかUTF8で保存する

- 逆にVCでは警告が出るけど

または

-finput-charset=cp932で文字コードを指定する

Q. 関数のオーバーロード

void*ならそのアドレス, const char *なら文字列を表示する関数を作った

```
void put(const void *addr) {
  printf("addr=%p¥n", addr);
}
void put(const char *str) {
  printf("str=%s¥n", str);
}
```

std::stringも受け付けたいので
char*をstd::string&に変更したらput("abc")は?

A. "abc"はvoid*にマッチする

暗黙の型変換は行われない

```
void put(const void *addr) {
  printf("addr=%p¥n", addr);
}
void put(const std::string& str) {
  printf("str=%s¥n", str.c_str());
}

put("abc"); // "addr=00403760"と表示される
```

Q. stream処理の曖昧さ

次の値はどうなる?

```
double a = boost::lexical_cast<double>("123,456");
```

1. 123
2. 456
3. 例外が飛ぶ

A. 状況による

"123,456"は不正なので例外が飛ぶことを期待

- しかし, streamはlocaleに依存している
  - 外部ライブラリの中で勝手に変えられることもある

```
std::locale::global(std::locale("C"));
boost::lexical_cast<double>("123,456"); // 例外が飛ぶ

std::locale::global(std::locale("Japanese"));
boost::lexical_cast<double>("123,456"); // 123456

std::locale::global(std::locale("French"));
boost::lexical_cast<double>("123,456"); // 123.456
```

CSVパーサなどでは気をつけて

Q. 今日から2万日後はいつ?

time_tとboost::posix_timeを合わせてみるか

```
time_t t;
time(&t); // 今の時間
t += 84600 * 20000; // 2万日後
std::cout << boost::posix_time::from_time_t(t) << std::endl;
```

- もちろんgcc, VC自体は2038年問題は対応済み
  - sizeof(time_t) = 8

A. from_time_tは使うな

1928-Sep-01 04:32:14とか表示された...

- sizeof(time_t) = 8な環境であっても, boost::posix_time::from_time_tは32bitしか考慮してないので2038年以降おかしくなる
- 1.46.0でも直ってない
- 2年ほど放置されてるので直す気がないのかも https://svn.boost.org/trac/boost/ticket/2818

Q. std::cin, cout遅い?

ファイルコピーをしてみる

```
#include <iostream>

int main() {
  for (char c; std::cin.get(c); ) std::cout << c;
}
```

- Linux(64bit, gcc 4.4)で128MBのファイルコピー
  - time ./a.out < data > /dev/null が53秒
- まずそもそもWindowsだとうまく動かない?

A. syncを切っとけ

Windowsではデフォルトがテキストモード

- 最初にバイナリモードにしておく

```
_setmode(_fileno(stdout), _O_BINARY);
_setmode(_fileno(stdin), _O_BINARY);
```

cin, coutは二つの同期を持つ

- printfなどのCの関数との同期
  - std::ios::sync_with_stdio(false); で切れる
- cinとcoutの同期
  - std::cin.tie(0); で切れる

rdbufとベンチマーク

とはいえcin.get(char&)は流石に(Cより)遅い

- rdbufを使うともっと効率よい

```
std::cin >> std::cout.rdbuf();
```

- Cループとも比較

```
int c; while ((c = getchar()) != EOF) putchar(c);
```

	標準	tieをoff	tieとsyncをoff
C++ loop	53	13	8
rdbuf	8.6	8.6	0.03
C loop	3	-	-

Q. boost::algorithmの注意点

自作trimに比べて遅いんだけどなんで?

- templateだから速いはずだよね

```
std::string a = "   abc   ";
const int N = 100000;
{
    boost::timer t;
    for (int i = 0; i < N; i++) {
        boost::algorithm::trim(a); // or my_trim(a);
    }
    printf("%f¥n", t.elapsed() / N * 10e6);
}
```

boost::algorithm::trim	mytrim
12.5usec	1.5usec

A. またもやlocale絡み

```
trim_copy(const SeqT& x, const std::locale& Loc=std::locale()){
 return ::boost::algorithm::trim_copy_if(x, is_space(Loc));
}
```

- std::localeが毎回作られて消える
- localeはシステムグローバル
  - だが，スレッドセーフでないといけない
    - その情報を取得するのにmutexを用いたglobal lockが必要
  - つまり初期コストが大きい
  - 巨大なテキストに対してはよいが，小さい文字列を大量に扱うときは無視できない

isspaceはロケール依存

my_isspaceを作って渡す方がより安全で速い

- その関数がlocaleに依存しないという意味で
- islowerなども同様

```
bool my_isspace(char c) {
  return c == ' ' || c == '¥t';
}
void my_trim(std::string& str) {
  boost::algorithm::trim_if(str, my_isspace);
}
```

- boost::algorithmの殆どの関数はlocale()を暗黙の引数に持つ

Q. 条件演算子

putBはBのインスタンスも一時変数もOK

```
struct A { int dummy_; };
struct B : A {
  int b_;
  B(int b = 0) : b_(b) {}
};
void putB(const A& a) { // aが実はBだった場合にのみを想定
  printf("%d¥n", static_cast<const B&>(a).b_);
}
B b(2);
putB(b); // bをconst A&で受ける 2を表示
putB(B(9)); // 一時変数もconst A&で受けられる 9を表示
```

Q. では条件演算子を使うと?

const A&なaが本当にBならそのまま, 違うなら一時的に作ってみる

```
void put(bool isB, const A& a) {
  putB(isB ? a : B(4));
}

put(true, b); // どうなるか
```

A. コピーされて壊れるかも

条件演算子の両方の型を揃えておく

```
void put2(bool isB, const A& a) {
  putB(isB ? a : static_cast<const A&>(B(4)));
}

put(true, b); // -1081772232 おかしなメモリを参照した
put2(true, b); // 2 正しくキャストできた
```