Subject: Re: How to handle a lack of memory
Posted by mirek on Thu, 07 Apr 2016 20:59:46 GMT
View Forum Message <> Reply to Message

It is not so easy...

Frankly, it all boils down to VectorMap::Add implementation as example:

```
 T&      Add(const K& k, const T& x)           { key.Add(k); return value.Add(x); }
```

Now if value.Add runs out of memory, VectorMap is left in inconsistent state. Not that this is just example, practical code is full of examples like this. E.g. what is wrong with this code:

```
struct Foo {
   Bar *ptr;

   Foo() { ptr = new Bar; }
   ~Foo() { delete ptr; }
}
```

It is certainly possible to solve that by adding exception block into Add, but correctly resolving this issue everywhere would result in significant increase in codebase (and probably decreased performace). And it is virtually untestable.

Plus, in 64 bit platform (but often with 32-bits too), you are rather going to freeze system first (because of swap trashing) that actually get to out-of-memory.

In practice, I do not remember getting out-of-memory in other situations that program bugs. Have you ever got out-of-mem in theide?

Mirek