Subject: Variadic templates helper functions
Posted by mirek on Wed, 17 Jun 2020 16:07:14 GMT
View Forum Message <> Reply to Message

While modernizing U++ code and removing Expand macros, I have invented some little helper functions to simplify dealing with variadic templates:

```
template <class F, class V>
void foreach_arg(F fn, V&& v)
{
 fn(v);
}

template <class F, class V, typename... Args>
void foreach_arg(F fn, V&& v, Args&& ...args)
{
 fn(v);
 foreach_arg(fn, args...);
}

template <class I, typename... Args>
void iter_set(I t, Args&& ...args)
{
 foreach_arg([&](auto&& v) { *t++ = v; }, args...);
}

template <class I, typename... Args>
void iter_get(I s, Args& ...args)
{
 foreach_arg([&](auto&& v) { v = *s++; }, args...);
}

template <class C, typename... Args>
C gather(Args&& ...args)
{
 C x(sizeof...(args));
 iter_set(x.begin(), args...);
 return x;
}

template <class C, typename... Args>
int scatter_n(int n, const C& c, Args& ...args)
{
 if(n < sizeof...(args))
  return 0;
 iter_get(c.begin(), args...);
 return sizeof...(args);
```

```
}

template <class C, typename... Args>
int scatter(const C& c, Args& ...args)
{
 return scatter_n(c.size(), c, args...);
}
```

For example:

```
template <typename... Args>
bool SplitTo(const char *s, int delim, bool ignoreempty, Args& ...args)
{
 return scatter(Split(sizeof...(args), s, delim, ignoreempty), args...);
}



template <typename... Args>
void MergeWith(String& dest, const char *delim, const Args& ...args)
{
 foreach_arg([&](const String& arg) {
  if(arg.GetCount()) {
   if(dest.GetCount())
    dest << delim;
   dest << arg;
  }
 }, args...);
}



template <typename... Args>
String Format(int language, const char *s, const Args& ...args)
{
 return Format(language, s, gather<Vector<Value>>(args...));
}
```

Mirek
