Subject: Small optimization
Posted by Didier on Tue, 18 May 2021 13:36:35 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I have a small optimisation proposition for cases where fundamental data types are used

in Vcont.h

Replace:

```
template <class T>
inline void Construct(T *t, const T *lim) {
 while(t < lim)
  new(t++) T;
}
```

with:

```
template <class T>
inline void Construct(T *t, const T *lim) {
 if ( ! std::is_fundamental<T>::value ) {
  while(t < lim)
   new(t++) T;
 }
}
```

Other methods near this one can benefit from this small optimisation (Destroy for example)

Note: Tested on my app : everything seems to work fine

Subject: Re: Small optimization
Posted by mirek on Thu, 27 May 2021 07:07:34 GMT
View Forum Message <> Reply to Message

Didier wrote on Tue, 18 May 2021 15:36Hello Mirek,

I have a small optimisation proposition for cases where fundamental data types are used

in Vcont.h

Replace:

```
template <class T>
```

```
inline void Construct(T *t, const T *lim) {
 while(t < lim)
  new(t++) T;
}
```

with:

```
template <class T>
inline void Construct(T *t, const T *lim) {
 if ( ! std::is_fundamental<T>::value ) {
  while(t < lim)
   new(t++) T;
 }
}
```

Other methods near this one can benefit from this small optimisation (Destroy for example)

Note: Tested on my app : everything seems to work fine

I am not fundamentally against it, but I am pretty sure that compiler already optimizes that out as dead code - which means it is just added noise.

```
never_inline
void Foo() {
 RLOG("---");
}

CONSOLE_APP_MAIN
{
 int x[4];
 Foo();
 Construct(x, x + 4);
 Foo();
}
```

```
400018B0  sub rsp,byte +0x28
400018B4  call dword 0x1400017e0 // call to Foo
400018B9  nop // this to align
400018BA  add rsp,byte +0x28
400018BE  jmp dword 0x1400017e0 // call to Foo
```

Mirek

Subject: Re: Small optimization
Posted by Didier on Thu, 27 May 2021 18:01:10 GMT
View Forum Message <> Reply to Message

You are perfectly right then: clearner code is better