Subject: Using USEMALLOC flag leads to errors on Log.cpp
Posted by Oblivion on Sat, 04 Jan 2014 00:40:56 GMT
View Forum Message <> Reply to Message

Hello,
If I use USEMALLOC flag, I get the following errors with Core/Log.cpp. If I don't use the flag, then file compiles.
This happens with U++ latest nightly build (6715) under (arch Linux) LINUX 3.12-5, Gcc 4.8.2-7 and arch: i686 with no SSE2

```
.../uppsrc/Core/Log.cpp: In function 'Upp::String Upp::AsString(const Upp::MemoryProfile&)':
.../uppsrc/Core/Log.cpp:312:42: error: 'large_empty' is not a member of 'const struct
Upp::MemoryProfile'
  text << "Large free 64KB pages " << mem.large_empty
                                          ^
.../uppsrc/Core/Log.cpp:313:38: error: 'large_emty' is not a member of 'const struct
Upp::MemoryProfile'
       << ", total size " << 64 * mem.large_empty << " KB\n";
                                    ^
.../uppsrc/Core/Log.cpp:314:36: error: 'big_count' is not a member of 'const struct
Upp::MemoryProfile'
  text << "Big block count " << mem.big_count
                                  ^
.../uppsrc/Core/Log.cpp:315:37: error: 'big_size' is not a member of 'const struct
Upp::MemoryProfile' '
       << ", total size " << int(mem.big_size >> 10) << " KB\n";
                                    ^
```

The thing is, Upp::MemoryProfile actually does have those members.

Edit: After examining Log.cpp and Defs.h files more closely, I found the problem.
There are two MemoryProfile structures defined in Defs.h. One is to use with the UPP Heap model and the other one to use with USEMALLOC flag.
It seems that AsString() function, defined in Log.cpp line 287, does not differentiate between the two structures and always uses the MemoryProfile structure defined for the UPP heap model.
Adding an #ifdef/#endif preprocessor block where necessary seems to solves the problem:

```
String AsString(const MemoryProfile& mem)
{
 String text;
 int acount = 0;
 size_t asize = 0;
 int fcount = 0;
 size_t fsize = 0;
 for(int i = 0; i < 1024; i++)
```

```
  if(mem.allocated[i]) {
   int sz = 4 * i;
   text << Format("%4d B, %6d allocated (%5d KB), %6d fragmented (%5d KB)\n",
                  sz, mem.allocated[i], (mem.allocated[i] * sz) >> 10,
                  mem.fragmented[i], (mem.fragmented[i] * sz) >> 10);
   acount += mem.allocated[i];
   asize += mem.allocated[i] * sz;
   fcount += mem.fragmented[i];
   fsize += mem.fragmented[i] * sz;
  }
 text << Format(" TOTAL, %6d allocated (%5d KB), %6d fragmented (%5d KB)\n",
                acount, int(asize >> 10), fcount, int(fsize >> 10));
 text << "Free 4KB pages " << mem.freepages << " (" << mem.freepages * 4 << " KB)\n";
 text << "Large block count " << mem.large_count
      << ", total size " << (mem.large_total >> 10) << " KB\n";
 text << "Large fragments count " << mem.large_free_count
      << ", total size " << (mem.large_free_total >> 10) << " KB\n";
#ifdef UPP_HEAP                                                                  //
<---------------------------------------- Added.
 text << "Large free 64KB pages " << mem.large_empty
      << ", total size " << 64 * mem.large_empty << " KB\n";
 text << "Big block count " << mem.big_count
      << ", total size " << int(mem.big_size >> 10) << " KB\n";
#endif                                                                           //
<---------------------------------------- Added.
 return text;
}
```

I am not very familiar with the U++ core internals, so Is this valid? If so, could you please update the U++ source with necessary fix?

Thanks.

Regards.