

---

Subject: Problem with UPP\_HEAP and multithread  
Posted by [kov\\_serg](#) on Thu, 08 Oct 2009 08:58:45 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

I have serious problem with Upp memory allocation in multithreads. This problem dissappear if I don't use UPP\_HEAP.

If I allocate, reallocate and free memory in different thread, I have strange bugs. If I force Defs.h "flagUSEMALLOC" problem doesn't occure. I think this is some kind of bug with heap monitor.

The simples example to reproduce this bug attached in file test.cpp (with option MT)

If I call th0(1) programm works fine.

But if argument th0(2) or more it will stop debugger in various places with out visible reason.

In real programm it cause unhandled exception when gui terminates or even fail on `vector<>.clear` inside GUI thread `o_0`.

hz1.png -- debugger stop in unknown place without visible reason. (test.cpp)

hz2.png -- screen shot from other program when this problem occure. (real project) I have exceptions from delete operator. Usually from `vector<>.clear()`. Exception may occure on different delete operators. And behaviour is very unstable.

Upp version: SVN.643

## File Attachments

---

- 1) [test.cpp](#), downloaded 372 times
- 2) [hz1.PNG](#), downloaded 441 times

C:\Apps\up\out\MSC8.Debug

```
0.00 born
0.19 create thread 1964
0.28 resume thread
0.30 thread1 start
0.39 clear & add item
0.50 thread1 done
0.59 n=1 x[n-1]=11
0.70 create thread 1960
0.80 resume thread
0.81 thread2 start
0.91 clear & add item
1.00 thread2 done
1.11 n=1 x[n-1]=12
1.20 create thread 1956
1.31 resume thread
1.31 thread3 start
1.41 clear & add item
1.52 thread3 done
1.61 n=1 x[n-1]=13
1.70 clear
1.70 kill
ready!
```

The screenshot shows the Visual Studio IDE with the following components:

- File Explorer:** Shows the file `test1.cpp`.
- Code Editor:** Contains the following C++ code:

```
22 Sleep(100);
23 t();printf("thread%d done\n",i);
24 return 0;
25 }
26
27 void th0() {
28     v = new std::vector<int>();
29     t();printf("born\n");
30     for(int i=1;i<4;++i) {
31         Sleep(100);
32         int h=(int)_beginthreadex(0,0,th1,(void*)i,CREATE_SUSPENDED);
33         t(); printf("create thread %d\n",h);
34         Sleep(100);
35         { Mutex::Lock lock(mutex);
```
- Output Window:** Displays the execution log from the left window.
- Debugger:** Shows the `Autos` tab with a list of memory addresses and their corresponding values, including `__ld12mul(px=12f9de->{ ld12-`.

3) [hz2.PNG](#), downloaded 468 times



heap.cpp

```

43 }
44
45 void Heap::RemoteFree(void *ptr)
46 {
47     LLOG("RemoteFree " << ptr);
48     Mutex::Lock __ (mutex);
49     FreeLink *f = (FreeLink *)ptr;
50     f->next = remote_free;
51     remote_free = f;
52 }
53
54 void Heap::FreeRemoteRaw()
55 {
56     while(remote_free) {

```

- Upp::Heap::RemoteFree(ptr=2db0080)
- Upp::Heap::Free(ptr=2db0080)
- Upp::MemoryFree\_(ptr=2db0080)
- Upp::MemoryFree(ptr=2db0090)
- Upp::MemoryFree32(ptr=2db0090)
- Upp::String0::LFree()
- Upp::String0::Free()
- Upp::String0::~String0()
- Upp::AString<Upp::String0>::~AString()
- Upp::Moveable<Upp::String,Upp::AString>
- Upp::String::~String()
- Upp::String::`scalar deleting destructor'
- std::\_Destroy<Upp::String>(\_Ptr=1041)
- std::allocator<Upp::String>::destroy(\_F)
- std::deque<Upp::String,std::allocator<Upp::String>
- std::deque<Upp::String,std::allocator<Upp::String>

```

lea ecx,[ebp-0x4]
call 0x401ad2
mov eax,[ebp+0x8]
mov [ebp-0x8],eax
mov ecx,[ebp-0x8]
mov edx,[ebp-0xc]
mov eax,[edx+0x740]
mov [ecx],eax
mov ecx,[ebp-0xc]
mov edx,[ebp-0x8]
mov [ecx+0x740],edx
lea ecx,[ebp-0x4]
EAX 02DB0080  ESI
EBX 7FFDD000  EDI
ECX 02DB0080  EBP
EDX 00173E88  ESP

```

Autos Locals Watches Explorer Memory 0xca0

f	2db0080->{ next=20 }
f->next	20->{ next=?? }
ptr	2db0080

Upp::Heap::RemoteFree(ptr=2db0080)