Subject: A couple of issues with a test app
Posted by cbpporter on Fri, 04 Jan 2008 14:51:17 GMT
View Forum Message <> Reply to Message

Hi!

I started to create a test app for a small Kanji flash-card program, and right now I'm at the phase where I need to load some XML, save them as some other format that is a lot smaller and saves space, and then implement the flash card interface.

But I have tree issues:

1. During loading the huge XML, the occupied memory grows to over 150MB, and after storing just a fraction of the data in a column list, the memory is not freed. Since I'm not doing manual memory managment, I don't know why it isn't freed. It is not freed even if I call Clear on the list (which it shouldn't, because the list should have no way to take up such amounts of memory), which leads me to the second issue.

2. When I call clear on the list, the scrollbar doesn't disappear, and it retains it's scroll interval.

3. Trying to execute the app in release mode, leads to a lock up under TheIDE and an ugly crash under normal executuion (WinXP, MINGW). The log file contains:
Tag/end-tag mismatch: <1066> </dic_number>
Maybe there is a mismatch in this freakin' huge XML (and I'll need to parse even a bigger one), but why does this crash my app, and if it has a good reason to do so, why doesn't it crash or act strangely in DEBUG mode.

File Attachments

1) KanjiFlash.rar, downloaded 481 times

Subject: Re: A couple of issues with a test app
Posted by mirek on Sat, 05 Jan 2008 08:19:12 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 04 January 2008 09:51Hi!

I started to create a test app for a small Kanji flash-card program, and right now I'm at the phase where I need to load some XML, save them as some other format that is a lot smaller and saves space, and then implement the flash card interface.

But I have tree issues:

1. During loading the huge XML, the occupied memory grows to over 150MB, and after storing just a fraction of the data in a column list, the memory is not freed. Since I'm not doing manual memory managment, I don't know why it isn't freed. It is not freed even if I call Clear on the list

(which it shouldn't, because the list should have no way to take up such amounts of memory), which leads me to the second issue.

Unfortunately, once memory is obtained from system, it is near to impossible to give it back...

The real problem is that the total number of memory blocks that are obtained from system is limited.

That can of course be "fixed" by allocating bigger blocks, but in that case there is much more unlikely that memory block is completely free.

So in the end U++ does never return the memory to the system. In fact, this in reality is a little problem - unused memory pages can be swapped away...

Mirek

Subject: Re: A couple of issues with a test app
Posted by cbpporter on Sat, 05 Jan 2008 09:06:49 GMT
View Forum Message <> Reply to Message

I'm not quite sure what you mean by "So in the end U++ does never return the memory to the system.". I thought that U++ solved memory allocation issues. My objects are on the stack, and when they are destroyed, they should be able to free all the hidden objects. Especially in the case of XML, where the data is allocated sequentially, and could be free in the same way. I never had problems with freeing block. I've written compilers which allocated huge amount of tokens and during their runtime heap sizes rapidly fluctuated between 10 and 400MB.

So if you could give some details why that memory can't be freed, I'm sure I'll be able to do something so that at least my app will have normal memory consumption. I don't want my app which with all data loaded will take up about 30MB to have 600 in the end, all because I had to parse 60MB of XML. Heck, gc would give a lot better results.

PS: I don't believe in swap

Subject: Re: A couple of issues with a test app
Posted by mirek on Sat, 05 Jan 2008 09:34:57 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sat, 05 January 2008 04:06I'm not quite sure what you mean by "So in the end U++ does never return the memory to the system.". I thought that U++ solved memory allocation issues. My objects are on the stack, and when they are destroyed, they should be able to free all the hidden objects. Especially in the case of XML, where the data is allocated sequentially, and could be free in the same way. I never had problems with freeing block. I've

written compilers which allocated huge amount of tokens and during their runtime heap sizes rapidly fluctuated between 10 and 400MB.

So if you could give some details why that memory can't be freed, I'm sure I'll be able to do something so that at least my app will have normal memory consumption. I don't want my app which with all data loaded will take up about 30MB to have 600 in the end, all because I had to parse 60MB of XML. Heck, gc would give a lot better results.

PS: I don't believe in swap

Well, heap size is one thing, comited memory another.

OK, details: Most of blocks average U++ code uses are concentrated in 4KB pages. It quite normal for such page to become free and it would seem that it could be returned to Win32.

ALAS, Win32 can only provide limited number of *separate* blocks. For 4KB blocks, that would mean at most 128MB allocated.

Therefore, we need to allocated these 4KB blocks in groups (the size of group in allocator gradually rises). Anyway, to release the group, it would require ALL pages in group to be free, which is statistically quite unlikely.

Sure, there is still possible that we have overlooked something in development, but it is not very likely. Actually, we have found this limit hard way a couple of years back, because originally U++ was capable of returning the memory back.

And, as far as I know, this policy (never return the memory) is quite standard (based on review of other allocators).

(Note that there is exception - very large blocks, >64KB, are actually returned to system on free - simply because they are allocated directly as single system block).

Mirek

Subject: Re: A couple of issues with a test app
Posted by mirek on Sat, 05 Jan 2008 09:49:02 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 04 January 2008 09:51
2. When I call clear on the list, the scrollbar doesn't disappear, and it retains it's scroll interval.

Stupid bug in U++.

Quick fix:

```
void ColumnList::Clear() {
 CancelMode();
 KillCursor();
 item.Clear();
 selcount = 0;
 Update();
 Refresh();
 SyncInfo();
 SetSb();
}
```

Mirek

Subject: Re: A couple of issues with a test app
Posted by mirek on Sat, 05 Jan 2008 10:01:53 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 04 January 2008 09:51
3. Trying to execute the app in release mode, leads to a lock up under TheIDE and an ugly crash under normal executuion (WinXP, MINGW). The log file contains:
Tag/end-tag mismatch: <1066> </dic_number>
Maybe there is a mismatch in this freakin' huge XML (and I'll need to parse even a bigger one), but why does this crash my app, and if it has a good reason to do so, why doesn't it crash or act strangely in DEBUG mode.

Not reproduced.. It works for me.

Mirek

Subject: Re: A couple of issues with a test app
Posted by mirek on Sat, 05 Jan 2008 10:03:36 GMT
View Forum Message <> Reply to Message

P.S.: The obvious and elegant solution for the memory problem is to use XMLParser directly. The resulting code would not be much more complicated... You could even add the progress bar that way...

Subject: Re: A couple of issues with a test app

Like this:

```
void KanjiFlash::Open()
{
 if(!fs.ExecuteOpen()) return;
 filename = fs;
 lstKanji.Clear();
 try {
  String s = LoadFile(filename);
  XmlParser p(s);
  while(!p.IsTag())
   p.Skip();
  p.PassTag("kanjidic2");
  while(!p.End())
   if(p.Tag("header"))
    while(!p.End()) {
     if(p.Tag("file_version"))
      lblVer = p.ReadTextE();
     else
     if(p.Tag("database_version"))
      lblDBVer = p.ReadTextE();
     else
     if(p.Tag("database_version"))
      lblDBDate = p.ReadTextE();
     else
      p.Skip();
    }
   else
   if(p.Tag("character")) {
    Kanji kanji;
    while(!p.End())
     if(p.Tag("literal"))
      kanji.Literal(p.ReadTextE());
     else
     if(p.Tag("misc"))
      while(!p.End()) {
       if(p.Tag("grade"))
        kanji.Grade(StrInt(p.ReadTextE()));
       else
       if(p.Tag("stroke_count"))
        kanji.StrokeCount(StrInt(p.ReadTextE()));
       else
        p.Skip();
      }
     else
```

```
      p.Skip();
    lstKanji.Add(RawToValue(kanji));
   }
   else
    p.Skip();
 }
 catch(XmlError) {
  Exclamation("Error reading the input file!");
 }
}
```

Memory consumption VP 6MB, and a bit faster too (obviously, you do not need to create XmlNode structure).

Mirek

Subject: Re: A couple of issues with a test app
Posted by cbpporter on Sat, 05 Jan 2008 16:24:59 GMT
View Forum Message <> Reply to Message

This approach seems to work. I have to look over the XmlParse source to see why this works, because it's quite interesting and is not SAX. I tried something similar inspired by the AdressBook example, but it did not work, so I tried the DOM like and this lead me to the problems.

Thank you for the help.

Quote:Stupid bug in U++.

Quick fix
I wonder how these very small but relatively frequent bugs are still present. Do they keep creeping in, or are they just simply unnoticed. I found some of them with ease just by doing simple stuff, and if they are unnoticed for long periods of time, it means that U++ user base is smaller than I thought.

PS: I don't know that you noticed, but about three-quarter through the list of 13000 characters, there are some which are mangled, being interpreted as 4 characters. I didn't give it too much attention, but it could be those 4 byte above 64K codepoint UTF-8 chars. Those things always seem to come back and haunt me. If that's the case, after I finish the KanjiFlash app, which I need ASAP, and the refresh issues while drawing in another app, I really have to sit down and finish that UnicodeEx package.

Subject: Re: A couple of issues with a test app

cbpporter wrote on Sat, 05 January 2008 11:24This approach seems to work. I have to look over the XmlParse source to see why this works, because it's quite interesting and is not SAX. I tried something similar inspired by the AdressBook example, but it did not work, so I tried the DOM like and this lead me to the problems.

Thank you for the help.

Quote:Stupid bug in U++.

Quick fix
I wonder how these very small but relatively frequent bugs are still present. Do they keep creeping in, or are they just simply unnoticed. I found some of them with ease just by doing simple stuff, and if they are unnoticed for long periods of time, it means that U++ user base is smaller than I thought.

You know, each time I encounter something like this, it makes me wonder too. This definitely was not crept in, it was there since the beginning.

The only explanation is that in cases like this, most people would use ArrayCtrl

Mirek