Subject: Executable as DLL
Posted by Novo on Sun, 27 Feb 2011 05:23:00 GMT
View Forum Message <> Reply to Message

Executable can be used as DLL. No changes to UPP is required. A proof of this is attached.

Explanation of how this can be done on Windows.

1) Create an application, which uses UPP. Let's name it exe_dll.

```
#include <Core/Core.h>

using namespace Upp;

extern String test_function(int i);

CONSOLE_APP_MAIN
{
 Cout() << test_function(1) << EOL;
}
```

2) Export required symbols via def file

```
EXPORTS
	?FormatInt@Upp@@YA?AVString@1@H@Z
	?MemoryAlloc@Upp@@YAPAXI@Z
	?MemoryFree@Upp@@YAXPAX@Z
	?LFree@String0@Upp@@AAEXXZ
```

3) Compile application. This will create an exe_dll.lib export library.

4) Create a DLL, which is not using UPP. Let's name this DLL test_dll.

```
#include <Core/Core.h>

using namespace Upp;

__declspec(dllexport) String test_function(int i)
{
 return FormatInt(++i);
}
```

Although Core.h is included, DLL itself is not linked against UPP.

5) Link test_dll against exe_dll.lib. As a result you will get a test_dll.lib library.

6) Link exe_dll application against test_dll.lib.

7) run exe_dll.

This is a long way. But I believe this work-flow can be automated.

I hope this approach will be useful for the UPP community.

## File Attachments

1) exe_dll.zip, downloaded 458 times

---

Subject: Re: Executable as DLL
Posted by dolik.rce on Sun, 27 Feb 2011 12:05:28 GMT
View Forum Message <> Reply to Message

This is very interesting idea with a lot of potential...

I tried to get some more info about how to achive similar thing in Linux and I came to this article on gcc wiki. It suggests a way how to (platform independently) separate public API from internal stuff, which I think could help a lot with making this automated.

Honza

---

Subject: Re: Executable as DLL
Posted by Novo on Sun, 27 Feb 2011 18:45:28 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 27 February 2011 07:05This is very interesting idea with a lot of potential...

I tried to get some more info about how to achive similar thing in Linux and I came to this article on gcc wiki. It suggests a way how to (platform independently) separate public API from internal stuff, which I think could help a lot with making this automated.

Honza

This should be even easier on Unix because we do not need export libraries. The only problem is how to export required symbols. GCC linker doesn't export anything by default. This situation can be reversed by using -Wl,-E option. In this case all symbols will be exported. But we do not need all of them because this can affect performance and loading time as mentioned in the article.
I checked this linker option with a simple console application and got 374088 versus 587080 bytes of executable size. -fvisibility=hidden option doesn't seem to affect size of executable. This is strange.

---

Subject: Re: Executable as DLL
Posted by Novo on Mon, 28 Feb 2011 00:16:31 GMT
View Forum Message <> Reply to Message

I figured out how to export limited set of names in Unix (ELF format).

We just need to use export maps. For example, one can create an export map file foo.map

```
{
global: name1; name2;
local: *;
};
```

and pass this file to linker via -Wl,--version-script=foo.map option.

Problem solved.

Subject: Re: Executable as DLL
Posted by mirek on Thu, 03 Mar 2011 15:55:59 GMT
View Forum Message <> Reply to Message

How is this different from linking with DLL flag?

Subject: Re: Executable as DLL
Posted by Novo on Fri, 04 Mar 2011 05:36:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 03 March 2011 10:55How is this different from linking with DLL flag?

If you are linking a package with DLL flag you will get a DLL.
If you want to call functions from UPP in your DLL you need to link your DLL against UPP packages.
If you want to use UPP code in both application and DLL you need to link both application and DLL against UPP packages. In this case you will get duplicated code, two different memory allocators, e.t.c.
If you want to share UPP code between app and DLL, you will need to compile UPP itself as DLL and link against this DLL, what you do not want to do.

What I'm proposing is to use app as DLL because UPP code is already in there. You just need to export it.

Am I wrong?

Subject: Re: Executable as DLL

Posted by mirek on Fri, 04 Mar 2011 07:59:27 GMT
View Forum Message <> Reply to Message

Ah, I see. That is nice indeed

Could be easily automated using coff package.

Mirek

Subject: Re: Executable as DLL
Posted by Novo on Fri, 04 Mar 2011 19:16:14 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 04 March 2011 02:59Ah, I see. That is nice indeed

Could be easily automated using coff package.

Mirek

IMHO, coff package deals with internals of COFF format. Automation of "Executable as DLL" process requires filtering out mangled names of missed symbols, creating/updating of def or linker script file and relinking of application and DLL. Unless exporting of symbols can be done by using coff package, there is no need to use it. And, of course, we are missing an elf package.

On Linux we will just need to relink app with updated linker script.