Subject: Problem building from commandline on Linux
Posted by steffen on Sat, 24 Nov 2012 12:23:48 GMT
View Forum Message <> Reply to Message

Hi,

I'm trying to build my package from the command line, but I'm having trouble getting the right build flags.

In TheIde I have set the following build flags: GUI,MT,GCC,Optimal.
I have changed the "Output Mode" for my main package Debug=Full, to include the debug symbols for stack trace parsing.
This works well when compiling from inside TheIde.

When I try to compile it from the command line it seems to ignore my settings entirely:

theide UppDTS StandAlone GCC -brs +GUI,MT
/home/vcas/Projects/UppDTS/StandAlone/StandAlone
Using the -d flag for debug is not possible, because that changes the optimizations flags, enables all asserts and so on. All other attempts result in a program without debug symbols.

I have tried the -M flag and looked at the generated makefile and I cant get the same result as when I export a makefile from TheIde.
Macro_StandAlone = $(Macro) -DflagMAIN <-- Wrong
vs
Macro_StandAlone = $(Macro) -DflagDEBUG_FULL -DflagMAIN <-- Correct
Is TheIde "Output Mode" settings stored somewhere outside my package dir? Perphaps ".upp/theide/cfg", if so, can I point the command line to it somehow?

My goal is to get an optimal release build, with debug symbols.
Then I strip the debug symbols from the executable file and keep them in a separate file.
Now if my program segfaults in the field it will simply generate a small list of return addresses by calling backtrace, and I can use GDB with the program and symbol file to see where it crashed.
The program is running on a small embedded system, so a full debug release is not an option, neither is full core dumps.

Everything works fine, except the command line build.
If anyone could tell me what I'm doing wrong, I would really appreciate it.

Regards,
Steffen

Subject: Re: Problem building from commandline on Linux
Posted by dolik.rce on Sat, 24 Nov 2012 13:33:33 GMT
View Forum Message <> Reply to Message

steffen wrote on Sat, 24 November 2012 13:23My goal is to get an optimal release build, with

debug symbols.
Then I strip the debug symbols from the executable file and keep them in a separate file.
Now if my program segfaults in the field it will simply generate a small list of return addresses by calling backtrace, and I can use GDB with the program and symbol file to see where it crashed.
The program is running on a small embedded system, so a full debug release is not an option, neither is full core dumps.
Hi Steffen,

Fisrt of all: I see you are using theide to compile, but I'd rather recommend you the umk utility, it is better supported. It is not really possible to make fine-tuning similar to "output mode" dialog on a command line. However, there are at least two options that come on my mind right now:

1) Create a new build method (or copy&modify your GCC.bm) and add '-g' switch to RELEASE_OPTIONS. That will add debugging symbols in the executable. You can also fine tune some other things here, e.g. add DEBUG_MINIMAL to RELEASE_FLAGS (applies to all packages). Building then should be the same, just supply the edited build method on the commandline.

2) Use the makefile generated for release and modify it afterwards. It can by simply done in script using e.g. sed:sed '/Macro_StandAlone =/s/$/ -g -DflagDEBUG_FULL/' /tmp/Makefile

Best regards,
Honza

Subject: Re: Problem building from commandline on Linux
Posted by steffen on Tue, 27 Nov 2012 11:15:57 GMT
View Forum Message <> Reply to Message

Thank you very much, Honza,

I have build the umk tool and changed my build script to use it. I read somewhere in the forum, that TheIde was the umk tool on Linux, and they both seems to work equally from the command line. It might be old knowledge, so I will use the umk tool from now on.

I copied the GCC.bm file to a SYMBOL.bm file and added the DEBUG_MINIMAL flag. That seems to do the trick for my purpose.

I also changed my strip script, so I simply keep the build with the symbols around and release the stripped version for my embedded clients. Now when I get the stack addresses returned after a crash, GDB can easily convert the addresses into symbols and I can get some knowledge about where my program crashed.
Of course parameter values would be nice also, so I might add that later.

Now I have set up a vmware Slitaz based Linux, with my build script that automatically updates the source from svn, put the svn revision numbers in a non versioned file, which is then included

from a small cpp file, with only these two lines:

#pragma BLITZ_PROHIBIT.
#include "../svn.version"

This way svn does not set the modified flag, as it would if I updated the source file directly.

It then builds my program, strips the debug symbols and generates a 7z archive with some configuration files and an update script.
This will really save me some time and avoid mistakes when making a release in the future.

If anyone is interested and can make use of some of it, I have attached my scripts here.

Best regards,
Steffen

File Attachments

1) scripts.tar.gz, downloaded 295 times

Subject: Re: Problem building from commandline on Linux
Posted by dolik.rce on Tue, 27 Nov 2012 12:43:54 GMT
View Forum Message <> Reply to Message

Glad to hear you solved it

steffen wrote on Tue, 27 November 2012 12:15I read somewhere in the forum, that TheIde was the umk tool on Linux, and they both seems to work equally from the command line. It might be old knowledge, so I will use the umk tool from now on.Yes, it used to be this way for a long time (and it still mostly works), but then the code got refactored and umk is now separate package, although it reuses a lot of code from theide. The new umk is more flexible with the configuration files and the CLI syntax is slightly better.

Honza
