Subject: GCC code size optimizations on ARM - GNU/Linux - uClibC
Posted by chickenk on Wed, 18 Jun 2008 16:39:04 GMT
View Forum Message <> Reply to Message

Hi everyone,

I am currently using theIde as a cross platform IDE for my embedded platform based on an arm926t core, running linux. So far so good, theIde is very useful and easy to configure to use a cross-toolchain.

I made a few test in compiling sample apps based on the Core library as well. Using my arm-linux-uclibc toolchain and a few customizations in the Core library to avoid MMX/SSE detection and other assembly code, I get a 456kB output file for the Core06 example.

In an attempt to reduce the output binary code size as much as possible, I added -ffunction-sections to the compiler options and --gc-sections to the linker options. It used to be a great solution for me on some previous projects. All I got is... 453kB. Not much better. That also means that the source files are well defined, since wrapping sections over object files or over functions is quite the same.

Then I added -fdata-sections to the compiler options, and this time, the size is reduced to 260kB. Great improvement, but makes me wonder:

does anybody here suspect this optimization to make my apps miserably fail later ? I will make tests of course (I'm sharing the device platform so I cannot right now, but tomorrow probably), but I may miss something that someone here already knows about the U++ library that needs this option not to be set.

Anyway, I'll try to give you informed of my results, if anyone interested...

Lionel

Subject: Re: GCC code size optimizations on ARM - GNU/Linux - uClibC
Posted by mirek on Wed, 18 Jun 2008 18:10:55 GMT
View Forum Message <> Reply to Message

chickenk wrote on Wed, 18 June 2008 12:39Hi everyone,

I am currently using theIde as a cross platform IDE for my embedded platform based on an arm926t core, running linux. So far so good, theIde is very useful and easy to configure to use a cross-toolchain.

I made a few test in compiling sample apps based on the Core library as well. Using my arm-linux-uclibc toolchain and a few customizations in the Core library to avoid MMX/SSE detection and other assembly code, I get a 456kB output file for the Core06 example.

In an attempt to reduce the output binary code size as much as possible, I added

-ffunction-sections to the compiler options and --gc-sections to the linker options. It used to be a great solution for me on some previous projects. All I got is... 453kB. Not much better. That also means that the source files are well defined, since wrapping sections over object files or over functions is quite the same.

Then I added -fdata-sections to the compiler options, and this time, the size is reduced to 260kB. Great improvement, but makes me wonder:

does anybody here suspect this optimization to make my apps miserably fail later ? I will make tests of course (I'm sharing the device platform so I cannot right now, but tomorrow probably), but I may miss something that someone here already knows about the U++ library that needs this option not to be set.

Anyway, I'll try to give you informed of my results, if anyone interested...

Lionel

Wow!

These are very interesting news - all of them

Please keep us informed about everything you are doing with ARM.

I am very surprised with data-sections results; maybe they could be applied to regular linux as well. I was not even aware about this option:)

Mirek

Subject: Re: GCC code size optimizations on ARM - GNU/Linux - uClibC
Posted by chickenk on Fri, 20 Jun 2008 07:29:23 GMT

View Forum Message <> Reply to Message

Hello,

a few updates:

1. I was totally wrong about the -fdata-sections option. In fact my linker option --gc-sections was not correctly set at first, and then I corrected it. I don't remember when I did the correction, but I made new compilations and in fact the big size reduction is due to -ffunction-sections, not -fdata-sections.

As an aside note, my linker option was wrong because I had to write "-Wl,--gc-sections" instead of just "--gc-sections". Wouldn't it be better to prepend automatically the -Wl and -Xlinker arguments to the linker options?

2. I compiled the Core06 example in various ways, with and without garbage-collecting sections, -Os/-O2, with and without -ffunction-sections, etc.

The only executable that runs correctly is the one with debug information. All Optimal/Size/Speed configurations have failed to run, most often with an unauthorized memory access exception and a segfault. When I have more time, I will analyse the strace logs more, but right now I've not found why this happens, and where exactly.

However, one thing to mention : my target has quite limited resources: 64MB RAM and around 60MB of free Flash space. The storage space is a Nand Flash and the filesystem type is jffs2. There is no RTC so the date starts at 01-01-1970 at each reset.

I will try to investigate more, just wanted to give some (not so good) news.

regards,
Lionel

Subject: Re: GCC code size optimizations on ARM - GNU/Linux - uClibC
Posted by phirox on Fri, 11 Jul 2008 10:03:40 GMT

View Forum Message <> Reply to Message

Not sure if this will work for ARM, but another very interesting optimization I found is "-frepo" for GCC/G++. Read more about it here: http://gcc.gnu.org/onlinedocs/gcc-4.0.2/gcc/Template-Instant iation.html

It optimizes template functions(which U++ uses a lot) and generally reduces code size by 25%