Subject: GCC compilation options
Posted by Zbych on Tue, 29 Dec 2009 10:46:10 GMT
View Forum Message <> Reply to Message

Hi,

I've noticed that u++ doesn't force gcc to remove unused functions nor data, so binaries are usually bigger (comparing to MS compiler). So, my proposition is to add:
1. -ffunction-sections and -fdata-sections to compilation options for gcc
2. -Wl,--gc-sections to linker options

I know that "-ffunction-sections" is already used, but without --gc-sections it is useless.

Subject: Re: GCC compilation options
Posted by mirek on Tue, 29 Dec 2009 16:36:14 GMT
View Forum Message <> Reply to Message

Zbych wrote on Tue, 29 December 2009 05:46Hi,

I've noticed that u++ doesn't force gcc to remove unused functions nor data, so binaries are usually bigger (comparing to MS compiler). So, my proposition is to add:
1. -ffunction-sections and -fdata-sections to compilation options for gcc
2. -Wl,--gc-sections to linker options

I know that "-ffunction-sections" is already used, but without --gc-sections it is useless.

We have experimented with it way back (5 years ago). At least in Linux and BSD, it did not seem to result in any difference in binary size.

Maybe things evolved since then. Could you try please? Ideal test candidate is theide in release mode.

You can pass linker options via package organizer for the purpose of the experiment...

Mirek

Subject: Re: GCC compilation options

Posted by Zbych on Tue, 29 Dec 2009 20:50:39 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 29 December 2009 17:36At least in Linux and BSD, it did not seem to result in any difference in binary size.

You will not see huge size reduction of big applications, because such applications use a lot of functions from u++, so there is almost nothing to remove.
The biggest difference is in small applications.
I've made simple MT tcp/ip server. Without --gc-sections size of application was ~1,5MB. After I added --gc-sections size decreased to 0,4MB. For me it is a huge difference.

Compilation results for some applications from u++ package:
Application	size [B] without dead code removal	size [B] with dead code removal	Size reduction
ide	5809896	5148244	11,4%
Gui01	1760072	1090132	38%
HomeBudget	3641508	2037696	44%

gcc version 4.4.1, all applications optimized for size.

Subject: Re: GCC compilation options
Posted by mirek on Wed, 30 Dec 2009 17:53:13 GMT
View Forum Message <> Reply to Message

Zbych wrote on Tue, 29 December 2009 15:50luzr wrote on Tue, 29 December 2009 17:36At least in Linux and BSD, it did not seem to result in any difference in binary size.

You will not see huge size reduction of big applications, because such applications use a lot of functions from u++, so there is almost nothing to remove.
The biggest difference is in small applications.
I've made simple MT tcp/ip server. Without --gc-sections size of application was ~1,5MB. After I added --gc-sections size decreased to 0,4MB. For me it is a huge difference.

Compilation results for some applications from u++ package:
Application	size [B] without dead code removal	size [B] with dead code removal	Size reduction
ide	5809896	5148244	11,4%
Gui01	1760072	1090132	38%
HomeBudget	3641508	2037696	44%

gcc version 4.4.1, all applications optimized for size.

Well, things definitely improved! This is good news.

Now I only have to decide whether to hardcode it or rather put another "Link options" field into Build methods and set it as default...

Subject: Re: GCC compilation options
Posted by mirek on Wed, 30 Dec 2009 20:00:28 GMT
View Forum Message <> Reply to Message

Implemented via adding "Link options" to build methods and making necessarry flags default (and part of linux installation as well).

Thanks a lot, it rocks.

Mirek

Subject: Re: GCC compilation options
Posted by gxl117 on Fri, 01 Jan 2010 06:00:01 GMT
View Forum Message <> Reply to Message

Subject: Re: GCC compilation options
Posted by mirek on Fri, 01 Jan 2010 08:30:58 GMT
View Forum Message <> Reply to Message

avail.I use TDM-MINGW GCC4.4.1-2

Can you be more specific?

Are Debug/Release link options edit fields missing from build methods, are they ignored while linking (use verbose mode to see the commandline) or are they ignored by mingw (which would not be our bug...)?

Mirek

Subject: Re: GCC compilation options
Posted by gxl117 on Fri, 01 Jan 2010 08:51:13 GMT
View Forum Message <> Reply to Message

Theide is normal.
Link option -gc-sections is no effect,compiled file size has not changed.I compile Gui01,with -gc-sections,file size is 1807360(B).without -gc-sections filesize is same.I can't get Zbych's 1090132(B) compiled file size. Are that option only effect for Linux???
If add -fdata-sections compilation options,compiled file size is increase to 1985024 B.

Subject: Re: GCC compilation options
Posted by mirek on Fri, 01 Jan 2010 19:07:41 GMT
View Forum Message <> Reply to Message

gxl117 wrote on Fri, 01 January 2010 03:51Theide is normal.
Link option -gc-sections is no effect,compiled file size has not changed.I compile Gui01,with -gc-sections,file size is 1807360(B).without -gc-sections filesize is same.I can't get Zbych's 1090132(B) compiled file size. Are that option only effect for Linux???

Very likely. Actually, if you read this from the start, the were no effective in the past in Linux as well...

You might also try our linker for mingw, I believe Tom had gc-sections support there.

Mirek

Subject: Re: GCC compilation options
Posted by Zbych on Tue, 05 Jan 2010 10:26:15 GMT
View Forum Message <> Reply to Message

gxl117 wrote on Fri, 01 January 2010 09:51I can't get Zbych's 1090132(B) compiled file size. Are that option only effect for Linux???

I've made my test on linux. For some reason MINGW/TDM doesn't remove dead code.

Subject: Re: GCC compilation options
Posted by Novo on Thu, 07 Jan 2010 02:01:44 GMT
View Forum Message <> Reply to Message

I checked numbers on Linux x86_64 GCC 4.4.1

TheIDE: 6358272B 5495408B reduction: 14%
My own simple CGI app: 2198160B 798944B reduction: 64%

I definitely like these new options!
Thanks a lot for pointing out!

Some time ago I read about symbol visibility options, which also helped significantly reduce size of resulting executable (which was using boost libraries). Has somebody tried visibility options with UPP?
