Subject: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 07:01:33 GMT
View Forum Message <> Reply to Message

Hi there,

trying to port the Core layer to The Blackfin / uCLinux world.
(The GUI layer later maybe, so far there is no real/extended X11 support there, except for nxlib/NanoX (former microwindows))

i have been using U++ for quite a time now, and almost forgot about my other IDEs. thanks guys..now thats my real first post.

Concerning settings, U++ works like a charm, the build method is set up for the BF537, using its toolchain, included the PATH, lib and include directives to my environment and everything works , almost...

I got 3 major errors, which in i need to be focused on a bit. down there the full test compile output from TheIDE

1. std::wstring support seems is not given in my toolchain. i will take care of that, is a uClibc thing, need to enable it

2. thread__ directive in Heahimp.cpp, Random.cpp and others, what is the dirctive about, what for are you using it, how can I circumvent it, there is no such directive in my compiler, now thats the huge problem

3. The CPU is uknown, i will take care of that, to exclude this case with a compiler flag, that comes in nature

(4.) the fork problem is a uClinux architecture specific thing, i will fix that (under most circumstances one can replace it with vfork)

BTW: is there ANY kind of source/development docu, the ideas behind what you put down there in the real bottom layers of U++, Core, Controls, i mean the "ideas behind the scene", like what is Heap class for..

wait to reading from you
kostah

File Attachments
1) new file, downloaded 304 times

Subject: Re: Porting U++ to Blackfin DSP
Posted by mirek on Fri, 03 Jul 2009 07:15:14 GMT

kohait00 wrote on Fri, 03 July 2009 03:01
1. std::wstring support seems is not given in my toolchain. i will take care of that, is a uClibc thing, need to enable it

Or not. We do not really need std::wstring, it is included onyl to provide minimal layer of unicode compatibility, maybe you can just use config.h to somehow to have some define and expel all std:: references.

Quote:
2. thread__ directive in Heahimp.cpp, Random.cpp and others, what is the dirctive about, what for are you using it, how can I circumvent it, there is no such directive in my compiler, now thats the huge problem

It designates TLS variables. Not needed for single-threaded builds.

Quote:
3. The CPU is uknown, i will take care of that, to exclude this case with a compiler flag, that comes in nature

Check (and fix) config.h.

Quote:
(4.) the fork problem is a uClinux architecture specific thing, i will fix that (under most circumstances one can replace it with vfork)

Well, obviously, there are parts you perhaps do not need to support to get working apps.

Quote:
BTW: is there ANY kind of source/development docu, the ideas behind what you put down there in the real bottom layers of U++, Core, Controls, i mean the "ideas behind the scene", like what is Heap class for..

Well, that is what srcimp topic group is intended for. But it is mostly empty yet...

Specifically, Heap class implements per-thread memory allocator.

Quote:
wait to reading from you
kostah

I will be glad to apply any patches for blackfin support...

Mirek

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 07:29:45 GMT
View Forum Message <> Reply to Message

hi mirek, now thats quick. if other linux posts were that quick

to std::wstring:
i'd like to change the least of code if possible. only what is really not avoidable, like the fork stuff. the support is there, but maybe you should consider, to genrelly have unicode stuff switchable on/off, but i am not much in that thing, as you do..anyway. if you have me a short idea how you mean expell std:: stuff (which is actually there, other thing compile fine, using std), i'd apriciate

to the thread thing:
actually, the uClinux progs on BF can just be normal multithreaded progs, only run w/o MMU, on bare memory. so i think this is the major problem in getting that thing to work.
maybe you coudl explain why and what for, each thread needs have its own Heap, where you use it.

to the srcimp:
i'll check what is in there.. basicly, i am pretty familiar with the upper layer code, containers, bit of Core stuff, the CtrlLib, great job, BTW, super readable (besides having no comments, but thats the reason maybe, self explainable code). but the porting layer and the work under the hood seem a bit invisble.

anyway, i'll be persuing that. would be great to have a real lightweight IDE like TheIDE having compile all the projects..eclipse and the others also support BF toolchain, but they are kind of bloated, and here we got a whole Helper class collection "for free" and can manage created projects.

cheers

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 09:26:08 GMT
View Forum Message <> Reply to Message

ok, i've done everything except for this __thread thing, which is actually difficult to make, i think. there is no TLS support for BF up to now, i think. any ideas how i could reimplement the things using __thread ?

BTW you are using thread__ sometimes, when GCC seems to be for __thread, they both work the same, but just in case..

For now it affects:

the blackfin flag modification, done
config.h

the wstring modification, done as you proposed
String.h
WString.cpp

vfork modification, done
LocalProcess.cpp
Util.cpp

TODO:
__thread modification
MT.cpp
HeapImp.h
sheap.cpp
Random.cpp
heapdbg.cpp

I will send the complete set of files, when done..against revision 1314, which i am using now

hints are welcome

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 09:33:49 GMT
View Forum Message <> Reply to Message

and one more: the link to the Blackfin uClinux forum, where this is discussed as well

https://blackfin.uclinux.org/gf/project/uclinux-dist/forum/?
_forum_action=ForumMessageBrowse&thread_id=34968&act ion=ForumBrowse&forum_id=39

stated there is, that TLS is recently available, so a "disabled TLS" supported version seems to be reasonable, they suggest. do you see any possibilities? i could get hands on that, if you have any idea about how ...

Subject: Re: Porting U++ to Blackfin DSP
Posted by mirek on Fri, 03 Jul 2009 10:56:34 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 03 July 2009 03:29
actually, the uClinux progs on BF can just be normal multithreaded progs, only run w/o MMU, on bare memory. so i think this is the major problem in getting that thing to work.
maybe you coudl explain why and what for, each thread needs have its own Heap, where you use it.

It is performance issue. API-wise, there is of course only one heap. Anyway, it is common that most allocations in thread are "local" (when new/delete happens in the same thread). That allows non-locking implementation of such allocation pairs. That is what Heap is helping to implement.

On interface level, you just use normal new/delete.

You can use USEMALLOC flag - that will use regular malloc/free for the heap. In fact, this might be better if you only low memory and low app - U++ heap is very effective (I believe , but is designed for apps that use more than ~200KB. Until than, it might be wasting memory a bit. (OTOH, for apps that use megabytes, it is likely to be saving it).

Mirek

Subject: Re: Porting U++ to Blackfin DSP
Posted by mirek on Fri, 03 Jul 2009 11:00:00 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 03 July 2009 05:26
BTW you are using thread__ sometimes, when GCC seems to be for __thread, they both work the same, but just in case..

Because __thread is GCC specific. thread__ is defined as __thread for gcc, but as "declspec(thread)" for MSC.

Quote:
I will send the complete set of files, when done..against revision 1314, which i am using now

Please, use the most recent version if possible. Saves some work

In fact, I am right now doing some very important work in Draw (which has direct impact on embedded market - it will decouple Draw from X11, in practice this will mean that you will be able to draw to Image buffers without X11). Maybe you should finish your work when this is done...

Mirek

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 11:30:07 GMT
View Forum Message <> Reply to Message

thanks mirek, that clarifies much of what i had in mind. for the GUI stuff, ill wait, till you're done.

Consider to integrate a mode to fully operate on framebuffer, so one wouldnt have to include all the bunch of X11 or others, and i think, U++ is far better off now, than, ie. fltk which I was using some months ago. having U++ compile really also on uC compatibles, running a kind of linux, would set U++ up in great position among all the APIs and libs, of which there are a *lot* now, since and espacially because TheIDE is doing a great job now. i have been having an eye on fltk, as mentioned, but it was too complicatet with its *fluid* designer, the gtk glade is overloaded, produces huge executables, SDL is not everywhere present (consider this also to be a down layer maybe, is widly used, Qt is not *really* free..and so the list goes on. U++ is really a difference here.

well, this one thing keeps to stand in way. To make TLS support siwtchable, would be really great. there are lots of arm's and BFs out there trying to setting up easy GUIs and having a small scalable framework underneeth. TLS is quiete rare there.

let me know if i can help in that some how.
kostah

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Fri, 03 Jul 2009 16:10:17 GMT
View Forum Message <> Reply to Message

i am half trough

specifying MT USEMALLOC as build options, i got some of the errors done. heapdbg.cpp was not behaving like expected, ignoring Heap, cause #include "HeapImp.h" is placed before the #if defined(UPP_HEAP) which excludes all the file. i corrected that.

in Random.cpp, a TLS pointer is used to initialize the random number generator per thread, to have different seeds basically, i think. well, we can live without, in blackfin, having a common seed, once. fix maybe later.

hard part last part:
The thread implementation, where a thread static flag is used, to indicate, if any new thread, besides the main thread, has anytime called Run() Function of *any* Thread instance..

i will try to figure out how to implement this defferently. and we are done , Upp on blackfin (so far..expecting runtime bugs)

have a nice weekend

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Sat, 04 Jul 2009 22:30:53 GMT
View Forum Message <> Reply to Message

Ok, Mirek,

done!!!, so far... testing needed, anyway, but i will do that soon.

down there is the source files archive, containing the changed sources against tonight's revision state, so normally just relplacing would do.

i tryed to stick to the rules i could see in code everywhere.

there are 10 files to be replaced in Core

it compiles well using GCC normal build mode AND it compiles well using the "BF537.bm" build method, which contains paths to mi local build root and tool chain.

the __thread problem was solved in that way:

Random.cpp: so far no need to initialize a random seed per thread, in BF we could live with that

in Mt.cpp: to indicate the per thread sMain as true, i placed the thread ids returned by pthread_self() in a static Index<pthread_t>, and later figured out, whaether the thread id is in there..BUG sourece!!! the Index cant shrink, no possibility to have a thread remove oneself's id from that Index, it will grow and grow with threads changing over time. but its a first shot

hope you can use it. i'll do
there is also a small test project, somehow i cant add it as second attachment, comes down there

UPDATE / BUGFIX: added the forgotten Mutex and made the static Index<pthread_t> remove thread ids in ~Thread

File Attachments

1) BF_port.tar.gz, downloaded 299 times

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Sat, 04 Jul 2009 22:52:56 GMT
View Forum Message <> Reply to Message

a test project:

need to set up a build environment, goto:
http://docs.blackfin.uclinux.org

File Attachments
1) objdirCons.tar.gz, downloaded 297 times

Subject: Re: Porting U++ to Blackfin DSP
Posted by mirek on Sun, 26 Jul 2009 15:19:51 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Sat, 04 July 2009 18:30Ok, Mirek,

done!!!, so far... testing needed, anyway, but i will do that soon.

down there is the source files archive, containing the changed sources against tonight's revision state, so normally just relplacing would do.

i tryed to stick to the rules i could see in code everywhere.

there are 10 files to be replaced in Core

it compiles well using GCC normal build mode AND it compiles well using the "BF537.bm" build method, which contains paths to mi local build root and tool chain.

the __thread problem was solved in that way:

Random.cpp: so far no need to initialize a random seed per thread, in BF we could live with that

in Mt.cpp: to indicate the per thread sMain as true, i placed the thread ids returned by pthread_self() in a static Index<pthread_t>, and later figured out, whaether the thread id is in there..BUG sourece!!! the Index cant shrink, no possibility to have a thread remove oneself's id from that Index, it will grow and grow with threads changing over time. but its a first shot

hope you can use it. i'll do
there is also a small test project, somehow i cant add it as second attachment, comes down there

UPDATE / BUGFIX: added the forgotten Mutex and made the static Index<pthread_t> remove thread ids in ~Thread

Minor comments:

Maybe the handling of IsMain in Mt.cpp is way to complex. Maybe we could rather store somehow main thread id into some global variable and compare with actual thread id?

Perhaps in Random.cpp, there should be a mutex in blackfin version to protect random generator?

Anyway, patch applied. Thank you for contributing!

Mirek

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Mon, 03 Aug 2009 09:14:07 GMT
View Forum Message <> Reply to Message

hi mirek,

thanks for adding me to contributors and the patch of corse.

if some more time left, ill check the 2 things mentioned. And with even more time left, i'll try to check the headless Draw facility on Blackfin boards. reports to come...

cheers

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Wed, 16 Dec 2009 13:20:21 GMT
View Forum Message <> Reply to Message

hi mirek,

considering the full porting of U++ to Blackfin, arise some little huge problems.

1) the BF is an interger DSP, which emulates floating point stuff. is it reasonable to think about integrating fixpoint arithmetik to compute sizes, colors etc. all the graphics stuff? what do you think, how severe is the huge of float/double operations inside the graphics layer?

2) the BF is running uClinux as flavor. no MMU!! so memory fragmentation is a problem, when doing a lot of malloc/free new/delete. how much dynamic allocation is done in the graphics layer? i know of the containers, which allocate on heap. how intense is the use there? could we modify the UPP ALLOCATER to somehow being able to manage a fixed chunk of memory? it means, when we start upp application, the Core allocates a given size of memoy (i.e. 64 MB) and is using this to service the heap alloc demands. (i would need to port to UPP MALLOC anyway for that porpuse, its not supportet currently)

these 2 major problems stand in performance way of porting full Upp to BF environment.

Subject: Re: Porting U++ to Blackfin DSP
Posted by mirek on Wed, 16 Dec 2009 13:42:12 GMT

kohait00 wrote on Wed, 16 December 2009 08:20hi mirek,

considering the full porting of U++ to Blackfin, arise some little huge problems.

1) the BF is an interger DSP, which emulates floating point stuff. is it reasonable to think about integrating fixpoint arithmetik to compute sizes, colors etc. all the graphics stuff? what do you think, how severe is the huge of float/double operations inside the graphics layer?

It will have an impact.

Quote:
2) the BF is running uClinux as flavor. no MMU!! so memory fragmentation is a problem, when doing a lot of malloc/free new/delete. how much dynamic allocation is done in the graphics layer? i know of the containers, which allocate on heap. how intense is the use there? could we modify the UPP ALLOCATER to somehow being able to manage a fixed chunk of memory? it means, when we start upp application, the Core allocates a given size of memoy (i.e. 64 MB) and is using this to service the heap alloc demands. (i would need to port to UPP MALLOC anyway for that porpuse, its not supportet currently)

these 2 major problems stand in performance way of porting full Upp to BF environment.

MMU does not really has a lot to do with process heap fragmentation. U++ generally is quite allocation intensive in all parts, maybe less than other libraries, but still is. But U++ allocator should cope with that.

Of course, it does not allocate memory from the system in small chunks. 4KB is the minimum size requested from system.

Mirek

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Wed, 16 Dec 2009 14:10:09 GMT

thanks, mirek.

i think i was misleading in my post..

1) was, how intensively does the grahics computation rely on float/double computation? would it be easy to rework the containing parts to fixpoint, like many other GUI libs have it?

2) the goal fot the UPP HEAP is, to allocate *once* a huge chunk, say 64 MB, in the upp

application, and after its done, it releases the complete chunk, thus the memory does get fragmentated only *inside* the upp application..

3) do you know of an easy way of emulating a TLS variable? in UPPHEAP we need it also, for the

```
thread__ Heap heap = {
 { DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI, DI }
};
```
in sheap.cp (what is it for btw. ?)

in Random and Mt.cpp we could circumvent the use of thread__, but here...i doubt. so why do you need TLS here..

tanks

Subject: Re: Porting U++ to Blackfin DSP
Posted by kohait00 on Wed, 16 Dec 2009 14:51:35 GMT
View Forum Message <> Reply to Message

***
i found out, that NPTL is supported with blackfin toolchaiin and uClibc, so there will be at least this opportunity, but better if not. since NPTL is GPL i think.
***

EDIT: WRONG information. NPTL *WONT* be supported in next time. i had a discussion on the blackfin forum, and the differences seem too huge, so porting wont be trivial. this means TLS is no option.

So i need to know what it is used for in the UPP_HEAP and how one could replace it..any ideas?
