Subject: The power of Makefile
Posted by dolik.rce on Fri, 05 Feb 2010 14:45:27 GMT
View Forum Message <> Reply to Message

Hello everybody!

I just have to share this bit of kowledge I've just gained with all of you... Until yesterday, I thought that make is just another simple unix tool. But after reading Managing projects with GNU make, I changed my mind.

Actually make is more like a programming language. It provides a lot of features and possibilities I would ever think of. Like defining functions, built-in rules, pattern expansion and many more.

Just to show you (at least to those of you who don't know make really good): Following Makefile is just a few lines, but it can analyze U++ package dependencies. For now, it is restricted to one directory, so it works only for uppsrc or in directory with exported project. Also, it ignores flags (just includes everything in "uses") . Both of this could be probably fixed with few more lines. Another problem might be it's speed. For ide it takes about 12s on my laptop, but for average-sized packages, it's less than second (also it's definitely not optimal, some steps are duplicated). Ok, here is the code:

```
PKG=ide
DEP=$(PKG)

define get-upp
 $(if $(findstring /,$1),$1/$(notdir $1).upp,$1/$1.upp)
endef

define get-uses
 $(strip $(shell cat $(call get-upp,$1) | tr ",\n;" "  \n" | grep "uses" | sed 's/(.*)//'))
endef

define get-deps
 $(sort $1 $(foreach d,$(sort $(subst \,/,$(filter-out uses%,$(call get-uses,$1)))),$(call
get-deps,$d)))
endef

.PHONY: all

all:
 @echo "To build $(PKG) following packages are needed:"
 @echo "$(call get-deps,$(PKG))"
```

If you want to try it, just create file uppsrc/Makefile and copy this in. To execute call make or make PKG=AnyOtherPackage. Actually this is what I like best - the universality. Actually I believe that it would be possible to extend this simple example to not only analyze, but also compile any package. The only information that would have to be added would be assembly structure, flags and compiler/linker options (or path to .bm and .var file).

If you read all the way down here, thanks for your patience

Best regards,
Honza

Subject: Re: The power of Makefile
Posted by mirek on Fri, 05 Feb 2010 15:04:18 GMT
View Forum Message <> Reply to Message

Does it work with BSD-make too?

Mirek

Subject: Re: The power of Makefile
Posted by masu on Fri, 05 Feb 2010 15:32:34 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 05 February 2010 16:04Does it work with BSD-make too?
Unfortunately this does not work with BSD make.
It does not support function definitions and it also does not have build in functions like 'findstring'.

Matthias

Subject: Re: The power of Makefile
Posted by dolik.rce on Sat, 06 Feb 2010 19:39:15 GMT
View Forum Message <> Reply to Message

masu wrote on Fri, 05 February 2010 16:32luzr wrote on Fri, 05 February 2010 16:04Does it work with BSD-make too?
Unfortunately this does not work with BSD make.
It does not support function definitions and it also does not have build in functions like 'findstring'.

Matthias

It should work with gmake. From what I read, it is fairly standard on BSD. I think some ports even require gmake to build properly...

Bye,
Honza

Subject: Re: The power of Makefile
Posted by masu on Sat, 06 Feb 2010 21:46:20 GMT
View Forum Message <> Reply to Message

Hi Honza,

It works with gmake (GNU make) which by the way is used to build TheIDE from ports at the moment.
So you are right, concerning the make utility it wouldn't change anything.
But BSD make itself is a different utility (the default make on BSD systems).

Matthias

Subject: Re: The power of Makefile
Posted by masu on Sat, 06 Feb 2010 21:55:43 GMT
View Forum Message <> Reply to Message

I tried your Makefile with gmake on FreeBSD and it works fine .

Matthias

Subject: Re: The power of Makefile
Posted by dolik.rce on Sat, 06 Feb 2010 22:34:03 GMT
View Forum Message <> Reply to Message

masu wrote on Sat, 06 February 2010 22:55I tried your Makefile with gmake on FreeBSD and it works fine .

Matthias
Hi Matthias,

Thanks for testing! I was just playing with this idea, so I was not concerned about compatibility. After looking into BSD make docs, I believe I could do the same in BSD style too. It would just contain more shell commands. And it wouldn't be compatible with GNU make BSD make uses different syntax for conditional inclusions. So if we want to use more of the makes power, we should stick to GNU version as it is available pretty much anywhere, from Linux through BSD and mac to cygwin.

Now, does someone actually think (like me) that this is fascinating? Having universal Makefile for any project is tempting idea, even though I'm not sure if it would be really useful.

Honza

Subject: Re: The power of Makefile
Posted by mr_ped on Sun, 07 Feb 2010 00:42:24 GMT
View Forum Message <> Reply to Message

I like GNU make a lot too. Usually when I see somebody whining about make and asking for

scons or something else, it turns out he has no idea how to use make properly and it would suit him perfectly.

That said, make is not final solution, in some cases there are better tools (for building upp applications I strongly prefer TheIDE), but many people bashing make don't have idea...

Subject: Re: The power of Makefile
Posted by masu on Sun, 14 Feb 2010 20:08:14 GMT
View Forum Message <> Reply to Message

Hi,

Like I mentioned here:
http://www.ultimatepp.org/forum/index.php?t=msg&th=4968& amp; amp;start=0&
I took up Honza's tests with GNU make and I would like to extend the idea to have a Makefile for building arbitrary U++ packages.

The version I have at the moment is supporting packages that are located including all its prerequisite packages. It does not support building packages distributed over several assemblies, but for now the Makefile should be put into uppsrc assembly and can then be used to build TheIDE.

I have tested it with FreeBSD and Ubuntu. I have also tried it in OpenBSD, but there seem to be other problems not related to the make process. At least all parameters (i.e. source files, link libraries) were extracted correctly on that system, too.

Here is a checklist I would like to work on:

1. Add support for building from other directories (to only provide on Makefile in root directoy for source releases)
2. Add support for setting object files' output directory
3. Add support for package building over different assemblies

And of course any error fixing in the meantime if erros occur during testing.

Edit: I forgot to mention, that per default (if you just type make) TheIDE is build with GTK bindings and shared linking.
Tip: You van also build with make -jx where x is the number of parallel processes run by make. I used parameter -j2 on my Dual Core system.

Matthias

File Attachments

1) Makefile.tbz, downloaded 420 times

Subject: Re: The power of Makefile
Posted by dolik.rce on Sun, 14 Feb 2010 21:21:05 GMT
View Forum Message <> Reply to Message

Hi Matthias,

You've got my deep respect! I didn't expect anyone even trying my ideas, let alone doing some work. But you almost finished it and actually much faster than I expected that it would take. Very impressive!

I'm just testing the Makefile and trying to figure out how it works. I have problems to even find the parts that I wrote But it seems to be working perfectly. Theide compiled perfectly on my machine, so I guess there should be no problems with other packages. And by the way, thanks for the hint with -j parameter for parallel builds, I didn't know it is that easy.

Now about the Makefile. First of all, I love the way you made it verbose. Reporting every step is really user friendly. I would even suggest (optional) hiding of the commands, so only the commands would be visible. About your TODO list: Numbers 1 and 2 should be easy. Just adding a variable with path on proper places. I'll try to figure it out. Number 3 is trickier, but most important. Currently, only uppsrc packages can be built, since other standard assemblies span across two directories. Hopefully we will find a way how to search through both.

Oh, and I almost forgot, another great think about this, that was not possible with the exported Makefiles: if you put it in directory with all sources, you can build any number of (main) packages to producing several binaries, while effectively caching the .o files. Similar way as theide does it. I think this is the biggest feature we have got here. I guess we could even add one more cycle and build multiple binaries at once with something like make "PKG=SocketClient SocketServer". Something like this is quite difficult to do in theide...

I would be very happy if this Makefile could be once distributed with U++ as an alternative to current exported Makefiles. I'm aware that it will never supersede it, because of it's complexity and also because the exported files are BSDmake compatible. This one just looks better to me. Even if just because it compresses 37000 lines into 150 It is really elegant.

Best regards,
Honza

Subject: Re: The power of Makefile
Posted by masu on Mon, 15 Feb 2010 15:24:58 GMT
View Forum Message <> Reply to Message

Hi Honza,

first, thanks for your appreciation .

A problem I have is to find a way to define variables when calling make which can be extended then during the make flow.

I thought VAR+=x should work, but it does not work for env variables defined as parameter to make, like this:

make DEFS=-DflagNOGTK

In this case the variable DEFS is completely overwritten by -DflagNOGTK. But what I want is to add this flag to already defined flags inside the Makefile.

Do you have an idea?

Matthias

Subject: Re: The power of Makefile
Posted by dolik.rce on Mon, 15 Feb 2010 15:50:51 GMT
View Forum Message <> Reply to Message

masu wrote on Mon, 15 February 2010 16:24Hi Honza,

first, thanks for your appreciation .

A problem I have is to find a way to define variables when calling make which can be extended then during the make flow.
I thought VAR+=x should work, but it does not work for env variables defined as parameter to make, like this:

make DEFS=-DflagNOGTK

In this case the variable DEFS is completely overwritten by -DflagNOGTK. But what I want is to add this flag to already defined flags inside the Makefile.

Do you have an idea?

Matthias
Hi,

This is usually solved using another variable: FIXEDFLAGS=...
FLAGS=$(FIXEDFLAGS) $(DEFS).

Simple, stupid, but works

Honza

Subject: Re: The power of Makefile

Posted by masu on Tue, 16 Feb 2010 22:24:06 GMT
View Forum Message <> Reply to Message

Hi,

here is the latest version Honza and I made.
It should be possible to build any desired package in U++ root dir . If it is not a main package use APP= on the cmd line (see below).
It was not extensively tested, so expect some glitches .

The following variables (with defaults in []) are supported on the cmd line:

PKG: package name to build [ide]
APPNAME: name of output program file [$(PKG).out]
ASSEMBLY: list of directories searched for packages [uppsrc]
INCLUDES: extra include flags [too long]
LDFLAGS: extra link flags [-L/usr/X11R6/lib -L/usr/lib -L/usr/local/lib]
DEFS: list of additional U++ flags [GCC]
OBJDIR: output directory for object files and libraries [_out]
VERBOSE: verbose stdout [off]

Link libraries and mainconfig flags are extracted.
Object files and libraries are put into output directories separated by package flag configuration (similar to how TheIDE it does).

Examples:

TheIDE(SHARED): make
TheIDE(SHARED NOGTK): make DEFS=NOGTK
TheIDE(NOGTK): make DEFS=NOGTK SHARED=
Controls4U_Demo(SHARED): make PKG=Controls4U_Demo ASSEMBLY=bazaar

Matthias

File Attachments

1) Makefile.tbz, downloaded 373 times

Subject: Re: The power of Makefile
Posted by dolik.rce on Mon, 15 Mar 2010 07:57:17 GMT
View Forum Message <> Reply to Message

Hello all!

After a long time I finally got to back to the Makefile... and I finally finished everything I wanted to.

So the file I present you here is rewritten from scratch. It is faster and smarter. New features are:
Compiles multiple packages at once
Reads and uses most of the options from .upp in the same way as theide
Verbosity switches
Simulation mode
Help target
Commented code (a bit)
Several bugs fixed
Little ASCII Art bonus
The last thing I miss is blitz
BTW: There are even some options that can not be set in theide Or at least I couldn't find how...
E.g. optimize file/package for size.

To get started, just copy the file into upp directory. I advise you to start with typing make
#OR
make help Both show you a help text, that contains a description and default values for variables that can controll the build process.

If you want to try something more elaborated, you can test something like this: make "PKG=ide

BINEXT= JOBS=3 This will compile theide, usvn and all the packages in mentioned folders, using flags from first line in mainconfig and put the binaries into a bin subdirectory. All the compiling will use 3 parallel jobs to speed the things up. Expect some error messages, as some of the packages are win only, have errors in code or have non-working mainconfig.

I tested the makefile by building more than 200 applications with various flags and randomly running many of them, without any encountering any errors in the makefile. So I believe that it should work in most cases. If anyone tries this, please let me know about your experiences Also, if you have any questions, feel free to ask (on forum or IRC).

Best regards,
Honza

File Attachments

1) Makefile, downloaded 368 times

Subject: Re: The power of Makefile
Posted by chickenk on Mon, 15 Mar 2010 10:38:57 GMT
View Forum Message <> Reply to Message

Hi Honza,

I tried it, it is absolutely great. Thank you very much for this effort, much appreciated.

Just for record, I added a very small rule 'showflags' (patch attached) that displays the flags for the main package as they are guessed. Example:

$ make PKG=ide showflags
ide_FLAGS: GCC LINUX MAIN
$ make PKG=ide USEMAINCFG=y showflags
ide_FLAGS: GCC GUI LINUX MAIN
Can be useful to know these flags before actually trying to compile.

Another way to do that without the patch:

$ make PKG=ide USEMAINCFG=y -p | grep '^ide_FLAGS'
ide_FLAGS := GCC GUI LINUX MAIN

Just my 2 cents. Keep up this good work, will surely help u++ to expand.

File Attachments

1) showflags.diff, downloaded 477 times

Subject: Re: The power of Makefile
Posted by koldo on Mon, 15 Mar 2010 11:13:48 GMT
View Forum Message <> Reply to Message

Hello Honza

It looks great. What should they be the next steps to integrate this and Launchpad into U++ infrastructure ?

Subject: Re: The power of Makefile
Posted by masu on Mon, 15 Mar 2010 11:31:07 GMT
View Forum Message <> Reply to Message

Hi Honza,

thank you for continuing the effort, great job !

I propose to discard uppsrc/ide/Makefile for future releases and include Honza's Makefile in root dir.
Either we define defaults to build TheIDE or we also provide a wrapper Makefile to call Honza's with needed variables.

Matthias

Subject: Re: The power of Makefile
Posted by dolik.rce on Mon, 15 Mar 2010 12:02:49 GMT

chickenk wrote on Mon, 15 March 2010 11:38Hi Honza,

I tried it, it is absolutely great. Thank you very much for this effort, much appreciated.

Just for record, I added a very small rule 'showflags' (patch attached) that displays the flags for the main package as they are guessed. Example:

$ make PKG=ide showflags
ide_FLAGS: GCC LINUX MAIN
$ make PKG=ide USEMAINCFG=y showflags
ide_FLAGS: GCC GUI LINUX MAIN
Can be useful to know these flags before actually trying to compile.

Another way to do that without the patch:

$ make PKG=ide USEMAINCFG=y -p | grep '^ide_FLAGS'
ide_FLAGS := GCC GUI LINUX MAIN

Just my 2 cents. Keep up this good work, will surely help u++ to expand.

Thank you, chickenk!

If you think this is helpful, I will add it. Usability is my main goal Also in future I plan (optional) interactive mode, where user could choose which set of flags from mainconfig to use and some other tricks...

Regards,
Honza

Subject: Re: The power of Makefile
Posted by dolik.rce on Mon, 15 Mar 2010 12:12:35 GMT

koldo wrote on Mon, 15 March 2010 12:13Hello Honza

It looks great. What should they be the next steps to integrate this and Launchpad into U++ infrastructure ?

Hi Koldo!
This has a little common with Launchpad. I'm not sure why the nightly builds doesn't work yet, probably Mirek is busy... But yes, I'm thinking about using this instead of generated makefiles, it is actually as easy as overwriting one file and changing few lines of script. I'll have a look at it soon.

About integration into U++: This is easy too. Just include the file in builds and mention it in

documentation. Additionally someone might integrate it into theide. Probably in similar style as current makefiles, just copying this makefile into export directory and write correct defaults in it.

Regards,
Honza

Subject: Re: The power of Makefile
Posted by dolik.rce on Wed, 17 Mar 2010 23:18:00 GMT
View Forum Message <> Reply to Message

Hello all!

New version is here! Changelog follows:
 Fixed bug when running via -f option (thanks Matthias)
 Modified output to make Chickenk more happy with SIMULATE mode output
 Added export target
 Expanded help
 Various little bugs fixed
The most important change is the export target. It parses the makefile with given options and saves it to a file given in EXPORT variable. The resulting file is faster (especially if you work with FAST=n), simpler, smaller, BSD-make compatibleand doesn't require any commandline parameters, since they are all hardcoded in it. You can sure magine now what can it be used for The only downside is that it is a bit messy inside, due to the nature of the exporting process and I don't think that it could be fixed (at least not easily).

Example: cd ~/uppsvn
make -f ~/Makefile PKG=ide "FLAGS=GUI SSE2 MT SPEED" FAST=n BIN=bin EXPORT=Makefile.ide
#later, as many times you want, possibly on different machine(s) (within same directory tree)
make -f Makefile.ide
Best regards,
Honza

File Attachments

1) Makefile, downloaded 369 times

Subject: Re: The power of Makefile
Posted by masu on Thu, 18 Mar 2010 09:57:36 GMT
View Forum Message <> Reply to Message

Honza,

thanks for your effort!
I will test it on BSDs as soon as possible.

Matthias

Subject: Re: The power of Makefile
Posted by Reini on Fri, 19 Mar 2010 16:53:05 GMT
View Forum Message <> Reply to Message

Hello Guys,

I would like to give some Feedback about the Makefile from OSX.
There is a parsing error in the script on my machine.

sh-3.2# make -f Makefile PKG=ide "FLAGS=GUI SSE2 MT SPEED" FAST=n BIN=bin
EXPORT=Makefile.ide
Parsing package ide ...
sed: 2: "1h;1!H;${ g;s/;[\t \x0A ...": RE error: parentheses not balanced
  Preparing output directory structure ...
usage: mkdir [-pv] [-m mode] directory ...
make: *** [prep-dirs] Error 64

My personal opinion is to move away from this complicated oldschool make stuff to premake build system since its also very small and much more easier to maintain !

Greeting Reinhold

Subject: Re: The power of Makefile
Posted by dolik.rce on Sat, 20 Mar 2010 21:48:26 GMT
View Forum Message <> Reply to Message

Reini wrote on Fri, 19 March 2010 17:53Hello Guys,

I would like to give some Feedback about the Makefile from OSX.
There is a parsing error in the script on my machine.

sh-3.2# make -f Makefile PKG=ide "FLAGS=GUI SSE2 MT SPEED" FAST=n BIN=bin
EXPORT=Makefile.ide
Parsing package ide ...
sed: 2: "1h;1!H;${ g;s/;[\t \x0A ...": RE error: parentheses not balanced
  Preparing output directory structure ...
usage: mkdir [-pv] [-m mode] directory ...
make: *** [prep-dirs] Error 64

My personal opinion is to move away from this complicated oldschool make stuff to premake build system since its also very small and much more easier to maintain !

Greeting Reinhold

Hi Reini,
Thank you very much for feedback. I'll have a look at it soon. Looks like it is just different syntax of mkdir on mac... I don't have any mac around, so I'll probably contact you soon via PM for frther testing, hope you don't mind that

It is not very complicated, and since it was never tested on mac (just linux and BSD), it doesn't even surprise me, that there is a problem... And do you really think premake/cmake/scons etc. are any simpler?

Regards,
Honza

Subject: Re: The power of Makefile
Posted by masu on Mon, 22 Mar 2010 10:05:22 GMT
View Forum Message <> Reply to Message

Hi Honza,

unfortunately, the same error occurs on BSDs with the latest version.
This is rather an issue with sed than with mkdir since mkdir just complains about missing arguments which is a result of sed parsing error.

Regards,
Matthias

Subject: Re: The power of Makefile
Posted by Reini on Mon, 22 Mar 2010 19:32:06 GMT
View Forum Message <> Reply to Message

Hello Honza,

Thanks for the comments

In fact I think that one of the mentioned build systems is in the end easier to maintain for all platforms. They can generate project files for all commonly widespreaded IDEs out there and simplify a lot. Also for the acceptance of UPP it would help improve a lot.

For example to integrate an external lib is just some commands and to update to a newer revision also. UPP has at the moment just hard linked sources to libpng which I assume have not updated

since several months. I see this since I got a lot of warnings when compiling the sources.

I try to do a version with premake and give you feedback how far I progress

cYa

Subject: Re: The power of Makefile
Posted by masu on Tue, 23 Mar 2010 09:45:14 GMT
View Forum Message <> Reply to Message

Hi Reini,

I favor the use of old-fashioned Makefile, because you can use gmake that has quite a small footprint compared to other make systems that have dependencies and may require, e.g. Python (I think SCons is programmed in Python).

Admittedly, the version we have now is not easy to read, so maintenance is a point, here. But I suppose Honza will be there to support it.

It was the goal for our Makefile to take *.upp files as the basis for generating the dependencies and get all options available inside these project files. So essentially all U++ projects can be build exactly in the way TheIDE would build them (except for Blitz) and if you change some options inside your project it will be automatically taken into account by the Makefile the next time you run it.
So you even do not have to write extra commands to include changes !

Matthias

Subject: Re: The power of Makefile
Posted by dolik.rce on Tue, 23 Mar 2010 11:10:58 GMT
View Forum Message <> Reply to Message

Hi,

The summary from Matthias is very accurate. I would like to just add that the initial idea was to get theide-like possibilities even on systems where theide is not installed, mainly to make export and distribution of apps easier.

Reini, no one is going to stop you if you try implement this with premake or any other build system. Actually opposite is true - the more possibilities we offer, the more users should be happy because they find something the are familiar with.

BTW: There was attempt to use cmake to build u++ (by Sender Ghost), but AFAIK it was not succesful (yet) because of some problems with icpp files handling. Similar problems might arise with premake.

Best regards,
Honza

Subject: Universal Makefile
Posted by dolik.rce on Fri, 27 Aug 2010 22:30:07 GMT
View Forum Message <> Reply to Message

Hi everyone,

Even though it might looked like the development of the beast I personally call universal Makefile have ceased, the opposite is truth. I gave it a major rewrite, keeping the same idea and mostly same syntax, but writing brand new internals, especially the parser.

So what is new: It supports a wider selection of systems, I took care especially to make sure it works on BSD (using gmake), and that should make it work on most other POSIX compliant systems too. Also it doesn't require bash now, plain sh will suffice (or ash, dash, etc.). Some changes also happened to the parallel building, it now uses the number of cpus as a number of threads by default. The output is IMHO prettier and more informative now. The internals are also written with maintainability in mind that should save me some hair pulling in future

Anyone interested can find the new version in uppbox/lpbuild in svn or on-line here:
http://code.google.com/p/upp-mirror/source/browse/trunk/uppb ox/lpbuild/Makefile

Best regards,
Honza

Subject: Re: Universal Makefile
Posted by chickenk on Sat, 28 Aug 2010 07:22:45 GMT
View Forum Message <> Reply to Message

Awesome.

A small bug: I'd like to use the "export" feature, using that syntax (my locale is fr_FR.UTF-8):
$ make -f uppbox/lpbuild/Makefile PKG=ide FLAGS="GUI MT" export
  Exporting uppbox/lpbuild/Makefile to Makefile.export ...
make[1]: [ide] Erreur 1 (ignorée)
$ cat Makefile.export
.PHONY:default
default: all
$
What? That's all? Seems it's missing some content. Next I try to use the C locale:

$ export LC_ALL=C
$ make -f uppbox/lpbuild/Makefile PKG=ide FLAGS="GUI MT" export

Exporting uppbox/lpbuild/Makefile to Makefile.export ...
make[1]: [ide] Error 1 (ignored)
$ cat Makefile.export
[whole content, so it works]

So I suggest the subprocesses are executed with the locale set to C, so that words parsing later works.

UPDATE: I also changed one line for more precise output, so that you understand why yours may differ: line 337, I changed $(call echo," Exporting makefile to $(EXPORT) ...") to $(call echo," Exporting $(filename) to $(EXPORT) ...")

regards,
Lionel

Subject: Re: Universal Makefile
Posted by dolik.rce on Tue, 31 Aug 2010 17:37:22 GMT
View Forum Message <> Reply to Message

Hi Lionel,

It took bit longer than I expected, I found few more bugs in the process of fixing this one And the problem actually was not related to the locale, at least I think...

Here I post fixed version, please let me know if it works better.

Best regards,
Honza

File Attachments
1) Makefile, downloaded 436 times

Subject: Re: Universal Makefile
Posted by masu on Mon, 29 Nov 2010 19:57:55 GMT
View Forum Message <> Reply to Message

Hi Honza,

really good work !
Now, I did find time to test your Makefile.

One thing I observed is that with every make call *.cpp files are always recompiled even if there exists an object file with a newer time stamp.
I tested this on different OSs (Ubuntu, CentOS, FreeBSD) and on machines having either a single or two CPUs and always got the same behavior.

The command line was:

make PKG=ide FLAGS="NOGTK GUI" USEMAINCFG=y BINPREFIX=~/bin/
I expect to only get a recompilation if source file is newer than compiled object file (at least that is what classic make tool checks for).

Regards,
Matthias

Subject: Re: Universal Makefile
Posted by dolik.rce on Mon, 29 Nov 2010 20:31:20 GMT
View Forum Message <> Reply to Message

masu wrote on Mon, 29 November 2010 20:57One thing I observed is that with every make call *.cpp files are always recompiled even if there exists an object file with a newer time stamp.
Oops, looks like I broke something It is of course supposed to rebuild only when the source has newer timestamp. I'll look into it...

Thanks for testing, Matthias!

Honza

Subject: Re: Universal Makefile
Posted by masu on Mon, 29 Nov 2010 22:33:28 GMT
View Forum Message <> Reply to Message

Honza, sorry, but another thing:

Flags in *.upp files labeled "BSD" are not taken into account on BSD systems. BSD variants have these flags in common.

Regards,
Matthias
