Subject: umk static version
Posted by mirek on Sun, 29 Mar 2020 14:12:45 GMT
View Forum Message <> Reply to Message

I have got an idea that umk in fact can be linked static, which means we can create universal binary for umk that would work on all linux distros (and looks like FreeBSD as well). That perhaps might improve out installation woes...

I am enclosing static build of umk ('umks'). I would welcome some info whether it runs on your distro.

P.S.: I am well aware this is sort of the same thing that long time ago dolik developed. However the problem was maintaining separate script that would have to reflect all changes in ide/Builders. This is just different compilation of the same thing.

File Attachments
1) umks.7z, downloaded 456 times

Subject: Re: umk static version
Posted by koldo on Sun, 29 Mar 2020 15:01:44 GMT
View Forum Message <> Reply to Message

It works in Ubuntu 18.04 LTS.
By the way, I thought this was not possible. This way, could be deployed binaries in Linux?

Subject: Re: umk static version
Posted by mirek on Sun, 29 Mar 2020 15:13:46 GMT
View Forum Message <> Reply to Message

koldo wrote on Sun, 29 March 2020 17:01It works in Ubuntu 18.04 LTS.
By the way, I thought this was not possible. This way, could be deployed binaries in Linux?

Well, the problem is that not all libraries have static variants. With umk, we are lucky, but you basically cannot build GUI application as required libraries are .so only.

(Also, this binary is AMD64 only, but I guess that is less of a problem.)

Mirek

Subject: Re: umk static version
Posted by dolik.rce on Sun, 29 Mar 2020 15:46:42 GMT
View Forum Message <> Reply to Message

It works on freshly upgraded ArchLinux as well as on ancient Debian wheezy.

This is actually much better than anything I ever proposed I always tried to build using theide using single Makefile, but maintaining that Makefile was a nightmare. This provides simple way to build U++ sources (anything, not just TheIDE) using the same code as used by TheIDE itself.

All you need to do is to add "umks" to sources, probably together with simple makefile that takes care about creating .bm files and than run umks. Or possibly cleaner solution, not requiring to have binary file in source tarball: Just download the latest "umks" in the Makefile (or let user download it manually in case of offline install, but that is a rare use-case). Just thinking aloud here, but I really like the idea. It's IMHO a win-win, since it will make the builds faster for user and the distribution of sources will be simpler for you :)when

Best regards,
Honza

Subject: Re: umk static version
Posted by mirek on Mon, 30 Mar 2020 08:24:26 GMT
View Forum Message <> Reply to Message

OK, let us go a bit further, above umks is AMD64, but it seems like it should be possible to have 32-bit binary that works on all x86 systems.

At least it works on my linux (I have to add new NOSO flag so that zlib and bz2 are included from internal sources).

Please test

(Also adding, for future reference, GCC32.bm for building 32-bit stuff on 64-bit system. You also need to "sudo apt-get install g++-multilib" to do that...)

File Attachments
1) umks32.7z, downloaded 438 times

Subject: Re: umk static version
Posted by mirek on Mon, 30 Mar 2020 08:25:20 GMT
View Forum Message <> Reply to Message

GCC32.bm

File Attachments
1) GCC32.bm, downloaded 451 times

Subject: Re: umk static version

Posted by dolik.rce on Mon, 30 Mar 2020 08:38:53 GMT
View Forum Message <> Reply to Message

Seems to work. I didn't actually compile anything, but the file executes successfully.

Subject: Re: umk static version
Posted by mr_ped on Mon, 30 Mar 2020 10:38:51 GMT
View Forum Message <> Reply to Message

JFYI: The 64b kernel does run 32b binaries only when compiled with support for 32b ABI, for example WSL2 in windows has strictly 64b-only kernel, not capable to execute 32b binaries (not really relevant to U++ use-case, just an example).

I'm not aware which distro has 64b-only kernel, I think none of the mainstream ones, but you may run into this issue with 32b binaries later in the future (but as the umk can be built easily as both 32 or 64 bit binary, it will be easy to deal with it, when it happens).

Subject: Re: umk static version
Posted by amrein on Mon, 30 Mar 2020 11:24:30 GMT
View Forum Message <> Reply to Message

A static umk binary in a source package is not a good idea in my opinion.
There is no need for this.

Proof:

Now that upp use pkg-config, you can run this command in linux and bsd and build theide out of the box:

make -C uppsrc -f Makefile.in -j8

The build scripts in the root directory (Makefile, domake, doinstall, ...) are mainly there to ease the build process + the install process + to handle variants like MacOS X for example or missing gcc compiler or too old compiler version, ...

Subject: Re: umk static version
Posted by mirek on Mon, 30 Mar 2020 12:55:17 GMT
View Forum Message <> Reply to Message

amrein wrote on Mon, 30 March 2020 13:24A static umk binary in a source package is not a good idea in my opinion.
There is no need for this.

It is potentially faster because of BLITZ.

(OTOH, some BLITZ form could be added to makefile too).

Still, consider this experimentation phase. Obviously, there is some dissatisfaction over Linux handling, so...

Mirek

Subject: Re: umk static version
Posted by dolik.rce on Mon, 30 Mar 2020 13:33:21 GMT
View Forum Message <> Reply to Message

I agree with amrein, that it would not be good idea to put it in the source package.

However, it is not entirely true that this is not needed. Currently, only theide can be build from theide using the pre-generated Makefile.in. With umks, the process is much more universal. You could build umk, or any of the examples, just by passing different arguments to make.

Also, if done properly, it would make it much easier for anyone to distribute their own U++ packages in source form. You'd just put uppsrc, your application sources and universal makefile (using umks) into tarball and you have source package built in an instant with little to no work.

I'm really tempted to write an example Makefile just to show you how easy it would be to build anything

Honza

Subject: Re: umk static version
Posted by dolik.rce on Mon, 30 Mar 2020 14:05:04 GMT
View Forum Message <> Reply to Message

Yep, I couldn't resist... Attached is a very simplistic Makefile allowing you to build anything simply by calling make <package_name>.

Just put it into the upp directory with umks (this wouldn't be needed, if it could be downloaded from U++ site) and call make to display help

EDIT: Updated the Makefile to actually work with downloaded umks (forgot to make it executable in the original version)

File Attachments

1) Makefile, downloaded 456 times

Subject: Re: umk static version
Posted by mirek on Mon, 30 Mar 2020 14:30:08 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Mon, 30 March 2020 16:05Yep, I couldn't resist... Attached is a very simplistic Makefile allowing you to build anything simply by calling make <package_name>.

Just put it into the upp directory with umks (this wouldn't be needed, if it could be downloaded from U++ site) and call make to display help

To make it more fun, umks is now uploaded to downloads...

Mirek

Subject: Re: umk static version
Posted by koldo on Mon, 30 Mar 2020 18:43:42 GMT
View Forum Message <> Reply to Message

How many posts, that is great!
If necessary, please review the install documentation accordingly .

Subject: Re: umk static version
Posted by dolik.rce on Mon, 30 Mar 2020 19:52:56 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 30 March 2020 16:30To make it more fun, umks is now uploaded to downloads...

Great I just fixed the Makefile in the original post so it actually works.

By the way: One thing I'd definitely change about umk is the "UI". Fixed position parameters, no real help, some params prefixed with '-', some with '+', no long options... Explaining this to a common user would be nightmare Isn't it time to tweak this a little as well? I'd be willing to do the work. I know it's not hard, but I guess you have more important things to do, right?

Honza

Subject: Re: umk static version

Posted by amrein on Mon, 30 Mar 2020 20:33:44 GMT
View Forum Message <> Reply to Message

The current build system can generate umk binary alone (make umk -j8).

Then theide could be build using umk to speed the process.

It know that this works on Linux because I tested a modified "domake" script doing just this in February 2017 (still on my computer). I didn't investigate further for other OS.

My main issue was: umk uses $HOME/umk/_out . This is incompatible with rpm based distro (and sandboxed compilation as on Debian I think).

edit: and now umk or umks create ~/.upp/umk/... ~/.upp/umks/... and this is still incompatible with binary package building in most Linux distributions.

Subject: Re: umk static version
Posted by mirek on Mon, 30 Mar 2020 21:28:06 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Mon, 30 March 2020 21:52mirek wrote on Mon, 30 March 2020 16:30To make it more fun, umks is now uploaded to downloads...

Great I just fixed the Makefile in the original post so it actually works.

By the way: One thing I'd definitely change about umk is the "UI". Fixed position parameters, no real help, some params prefixed with '-', some with '+', no long options... Explaining this to a common user would be nightmare Isn't it time to tweak this a little as well? I'd be willing to do the work. I know it's not hard, but I guess you have more important things to do, right?

Honza

'+' prefixes flags, which I think is not that bad.

I any case it would be good to show how do you want to change it before implementing...

Mirek

Subject: Re: umk static version
Posted by mirek on Tue, 31 Mar 2020 09:58:33 GMT
View Forum Message <> Reply to Message

amrein wrote on Mon, 30 March 2020 22:33The current build system can generate umk binary alone (make umk -j8).

Then theide could be build using umk to speed the process.

It know that this works on Linux because I tested a modified "domake" script doing just this in February 2017 (still on my computer). I didn't investigate further for other OS.

My main issue was: umk uses $HOME/umk/_out . This is incompatible with rpm based distro (and sandboxed compilation as on Debian I think).

edit: and now umk or umks create ~/.upp/umk/... ~/.upp/umks/... and this is still incompatible with binary package building in most Linux distributions.

I believe that the output dir can be specified as the last optional parameter...

Subject: Re: umk static version
Posted by amrein on Tue, 31 Mar 2020 10:52:50 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 31 March 2020 11:58amrein wrote on Mon, 30 March 2020 22:33The current build system can generate umk binary alone (make umk -j8).

Then theide could be build using umk to speed the process.

It know that this works on Linux because I tested a modified "domake" script doing just this in February 2017 (still on my computer). I didn't investigate further for other OS.

My main issue was: umk uses $HOME/umk/_out . This is incompatible with rpm based distro (and sandboxed compilation as on Debian I think).

edit: and now umk or umks create ~/.upp/umk/... ~/.upp/umks/... and this is still incompatible with binary package building in most Linux distributions.

I believe that the output dir can be specified as the last optional parameter...

It never worked for me:

```
tar zxf upp-x11-src-14204.tar.gz
cd upp-x11-src-14204
make -j8 umk
./umk uppsrc ide GCC myoutdir
```

this will still create $HOME/.upp/umk/_out/ and the last umk parameter won't be use.

The out parameter works only for the -x -X -M -Mx -mX options = exporting files necessary to build the project or exporting a Makefile.

Subject: Re: umk static version
Posted by dolik.rce on Tue, 31 Mar 2020 13:25:00 GMT
View Forum Message <> Reply to Message

The problem with using $HOME can be easily circumvented, just tell umk to use different one:

mkdir build
HOME="$PWD/build" ./umks "examples,reference,uppsrc" "ide" ./GCC.bm "-rbv" "+GUI,MT" "./theide"

But it's true, that this could be simpler.

Subject: Re: umk static version
Posted by dolik.rce on Tue, 31 Mar 2020 19:17:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 30 March 2020 23:28dolik.rce wrote on Mon, 30 March 2020 21:52One thing I'd definitely change about umk is the "UI". Fixed position parameters, no real help, some params prefixed with '-', some with '+', no long options... Explaining this to a common user would be nightmare Isn't it time to tweak this a little as well? I'd be willing to do the work. I know it's not hard, but I guess you have more important things to do, right?

I any case it would be good to show how do you want to change it before implementing...
I was thinking something like this:Usage:
  umk [options] <package> [<output>]

Arguments:
  <package> Package name
  <output> Output file or directory. If omitted, <package> name will be used.

Options:

-A, --assembly <paths>
  Comma separated paths to source directories or path to .var file.

-B, --build-method <build_method>
  Build method name or path to .bm file. Default value is 'GCC'.

-f, --flags <FLAGS> Comma separated list of flags. Default value is 'GUI,MT'.

-a, --all Rebuild all.

-b, --blitz Use BLITZ.

-v, --verbose Be verbose.

-l, --silent Silent mode.

-u, --target Use target directory.

-m, --map Create a map file.

-d, --debug=[no|minimal|full]
Select debug mode:
no = release mode, full optimizations (default)
minimal = no debug symbols, no optimizations
full = with debug symbols, no optimizations

-s, --shared Use shared libraries.

-S, --all-shared Use shared libraries and build as shared libraries.

-M, --makefile Create makefile (to file <output>).

-x, --export Export project (to directory <output>), export only files used.

-X, --export-all Export project (to directory <output>), export all files.

-k, --no-delete Do not delete target directory out when exporting.

-H, --threads <N> Build using <N> threads. Default is number of logical cores available.

-h, --help Show this text.

Examples:

Build release version of TheIDE:
umk -b -A uppsrc ide ~/theide

Use custom build method:
umk -b -A uppsrc -B ~/.upp/theide/CLANG.bm ~/theide

Rebuild Bombs example in debug mode:
umk -ab --debug=full -A examples Bombs

Build Bombs example using source paths
umk -A upp/examples,upp/uppsrc Bombs

What do you think?

Honza

Subject: Re: umk static version
Posted by Novo on Tue, 31 Mar 2020 20:20:06 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 31 March 2020 15:17What do you think?

"Use unique output directory per assembly (append assembly name to output directory)" is missing

Subject: Re: umk static version
Posted by amrein on Tue, 31 Mar 2020 20:34:33 GMT
View Forum Message <> Reply to Message

For testing:

Here is a Makefile that could potentially completely replace all the current build system.

* It uses the temporary HOME hack
* It shows help message if you don't give any argument to make
* It doesn't need domake script anymore

With this Makefile, developer are completely in control of what happen next.
This Makefile is easy to understand, so even developers should be ok with it .

File Attachments
1) Makefile, downloaded 415 times

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 06:53:52 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 31 March 2020 21:17mirek wrote on Mon, 30 March 2020 23:28dolik.rce wrote on Mon, 30 March 2020 21:52One thing I'd definitely change about umk is the "UI". Fixed position parameters, no real help, some params prefixed with '-', some with '+', no long options... Explaining this to a common user would be nightmare Isn't it time to tweak this a little as well? I'd be willing to do the work. I know it's not hard, but I guess you have more important things to do, right?

I any case it would be good to show how do you want to change it before implementing...
I was thinking something like this:Usage:
umk [options] <package> [<output>]

Arguments:
<package> Package name
<output> Output file or directory. If omitted, <package> name will be used.

Options:

-A, --assembly <paths>
Comma separated paths to source directories or path to .var file.

-B, --build-method <build_method>
Build method name or path to .bm file. Default value is 'GCC'.

-f, --flags <FLAGS> Comma separated list of flags. Default value is 'GUI,MT'.

-a, --all Rebuild all.

-b, --blitz Use BLITZ.

-v, --verbose Be verbose.

-l, --silent Silent mode.

-u, --target Use target directory.

-m, --map Create a map file.

-d, --debug=[no|minimal|full]
Select debug mode:
no = release mode, full optimizations (default)
minimal = no debug symbols, no optimizations
full = with debug symbols, no optimizations

-s, --shared Use shared libraries.

-S, --all-shared Use shared libraries and build as shared libraries.

-M, --makefile Create makefile (to file <output>).

-x, --export Export project (to directory <output>), export only files used.

-X, --export-all Export project (to directory <output>), export all files.

-k, --no-delete Do not delete target directory out when exporting.

-H, --threads <N> Build using <N> threads. Default is number of logical cores available.

-h, --help Show this text.

Examples:

Build release version of TheIDE:

umk -b -A uppsrc ide ~/theide

Use custom build method:
umk -b -A uppsrc -B ~/.upp/theide/CLANG.bm ~/theide

Rebuild Bombs example in debug mode:
umk -ab --debug=full -A examples Bombs

Build Bombs example using source paths
umk -A upp/examples,upp/uppsrc Bombs

What do you think?

Honza

Yes! It would be great. It's more POSIX friendly.
Options to add:

-c, --config <path>
Path to configuration files (default ~/.upp/umk).

-C, --cache <path>
Path to cache directory (default ~/.upp/umk/_out).

-e, --define <variable="value">
Overwrite an internal config value (advanced option).

Examples:

Build Bombs example using var paths and define a different cache location
umk -A examples.var -C $PWD/build Bombs

Build Bombs example using GCC build method but with a different compiler name
umk -A upp/examples,upp/uppsrc -e COMPILER="gcc-42" Bombs

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 07:25:11 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 31 March 2020 21:17mirek wrote on Mon, 30 March 2020 23:28dolik.rce wrote on Mon, 30 March 2020 21:52One thing I'd definitely change about umk is the "UI". Fixed position parameters, no real help, some params prefixed with '-', some with '+', no long options...

Explaining this to a common user would be nightmare Isn't it time to tweak this a little as well? I'd be willing to do the work. I know it's not hard, but I guess you have more important things to do, right?

I any case it would be good to show how do you want to change it before implementing...
I was thinking something like this:Usage:
   umk [options] <package> [<output>]

Arguments:
   <package>   Package name
   <output>   Output file or directory. If omitted, <package> name will be used.

Options:

-A, --assembly <paths>
            Comma separated paths to source directories or path to .var file.

-B, --build-method <build_method>
            Build method name or path to .bm file. Default value is 'GCC'.

-f, --flags <FLAGS> Comma separated list of flags. Default value is 'GUI,MT'.

-a, --all   Rebuild all.

-b, --blitz   Use BLITZ.

-v, --verbose   Be verbose.

-l, --silent   Silent mode.

-u, --target   Use target directory.

-m, --map   Create a map file.

-d, --debug=[no|minimal|full]
            Select debug mode:
               no   = release mode, full optimizations (default)
               minimal = no debug symbols, no optimizations
               full   = with debug symbols, no optimizations

-s, --shared   Use shared libraries.

-S, --all-shared   Use shared libraries and build as shared libraries.

-M, --makefile   Create makefile (to file <output>).

-x, --export   Export project (to directory <output>), export only files used.

-X, --export-all Export project (to directory <output>), export all files.

-k, --no-delete Do not delete target directory out when exporting.

-H, --threads <N> Build using <N> threads. Default is number of logical cores available.

-h, --help Show this text.

Examples:

Build release version of TheIDE:
umk -b -A uppsrc ide ~/theide

Use custom build method:
umk -b -A uppsrc -B ~/.upp/theide/CLANG.bm ~/theide

Rebuild Bombs example in debug mode:
umk -ab --debug=full -A examples Bombs

Build Bombs example using source paths
umk -A upp/examples,upp/uppsrc Bombs

What do you think?

Honza

I believe that -A as option does not make much sense. What will umk do if you do not specify -A? Probably the same for -B. I mean, you do not want to write

cp --from=sourcefile.txt --to=targetfile.txt

either, do you?

Also, it would really be good if the new interface was backward compatible, as not to break exiting scripts (not only mine, but I know that there are users that depend on umk). I think making '+' synonyme of '-f', perhaps even hidden, should not cause any problems.

Finally (and that is a problem with current sources too), the commandline parsing should be shared between theide and umk.

Mirek

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 07:36:45 GMT
View Forum Message <> Reply to Message

amrein wrote on Wed, 01 April 2020 08:53
-c, --config <path>
Path to configuration files (default ~/.upp/umk).

-C, --cache <path>
Path to cache directory (default ~/.upp/umk/_out).

I would like to change these defaults here. My current idea, maybe wrong, is that U++ will scan folders from the one where binary is up to root and tries to find ".config" (or .upp, not quite decided yet). Only if it does not find one, it would use ~/.upp/umk or ~/.config/umk.

Alternatively (or additionally), I can imagine that umk can easily run without configuration files...

Quote:
-e, --define <variable="value">
Overwrite an internal config value (advanced option).

You mean build method variable, right?

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 07:44:16 GMT
View Forum Message <> Reply to Message

The main argument for using switch like -A -B is to be flexible and POSIX friendly. You don't have to follow strict rules on parameters order anymore.

Those commands are the same:

umk -B GCC.bm -A uppsrc ide
umk -A uppsrc -B GCC ide
umk -A uppsrc ide

It will break old umk script yes... but do we really need to stick with older and weak behavior? Using umk in production scripts is so common nowadays?

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 08:05:23 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 01 April 2020 09:36amrein wrote on Wed, 01 April 2020 08:53

-c, --config <path>
Path to configuration files (default ~/.upp/umk).

-C, --cache <path>
Path to cache directory (default ~/.upp/umk/_out).

I would like to change these defaults here. My current idea, maybe wrong, is that U++ will scan folders from the one where binary is up to root and tries to find ".config" (or .upp, not quite decided yet). Only if it does not find one, it would use ~/.upp/umk or ~/.config/umk.

Alternatively (or additionally), I can imagine that umk can easily run without configuration files...

It's better to use default config folder on Windows, Linux or Mac and have the opportunity to define your own config folder for package sandboxing.
If not, you end up with config folders everywhere.

Quote:
Quote:
-e, --define <variable="value">
Overwrite an internal config value (advanced option).

You mean build method variable, right?

Yes. Those in GCC.bm for example. So it should be "Overwrite a build method variable value (advanced option)".
For OUTPUT and UPP (variables in .var files) there is -B and -C

Note: GCC.bm should be called CPP.bm. There is not g++ in this config file.

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 08:26:41 GMT
View Forum Message <> Reply to Message

Quote:
amrein wrote on Wed, 01 April 2020 10:05
I would like to change these defaults here. My current idea, maybe wrong, is that U++ will scan folders from the one where binary is up to root and tries to find ".config" (or .upp, not quite decided yet). Only if it does not find one, it would use ~/.upp/umk or ~/.config/umk.

Alternatively (or additionally), I can imagine that umk can easily run without configuration files...

It's better to use default config folder on Windows, Linux or Mac and have the opportunity to define your own config folder for package sandboxing.

Not sure about Linux/Mac, but after 15 years of trying various things, current approach used in Windows is far the best. That is: everything is stored in the same folder you get after unpacking the archive.

Using default config folder (like ~/.config) has various disadvantages compared to this simplistic approach. E.g. you cannot have two isolated U++ variants at the same time. If you delete U++ folder, you still have remnants in ~/.config. Etc...

So yes, my personal preference for improvement here is to sandbox it.

Now the suggested mechanism is not quite elegant. It is just the first approach that came to my mind to solve the problem in a way to keep things sanboxed while allowing normal ~/.config too (developed application would be normally produced to upp/out, so would pick upp/.config as well; when installed elsewhere, these would default to standard ~/.config).

Subject: Re: umk static version
Posted by dolik.rce on Wed, 01 Apr 2020 10:30:15 GMT
View Forum Message <> Reply to Message

Quote:I believe that -A as option does not make much sense. What will umk do if you do not specify -A? Probably the same for -B.

For -A, somewhat sane default could be to use all directories in parent directory of <package>. That would work out of the box in the unpacked U++ sources to build anything (from ide to examples). It would not work well with custom assemblies like MyApps, but those are probably not build with umk that often.

And for -B there definitely should be a default. If you open TheIDE, you don't have to select build method, one is always already selected (probably the first one found?). Umk could do the same, when no -B is given: look in the config directory, pick first build method it finds. Ideally with warning along the lines of "No build method specified, using 'XYZ' as default. Use -B to specify different method."

Quote:Also, it would really be good if the new interface was backward compatible, as not to break exiting scripts (not only mine, but I know that there are users that depend on umk). I think making '+' synonyme of '-f', perhaps even hidden, should not cause any problems.

Yes, I think it would not be too difficult to keep the old way working as well, at least for some time

---

Subject: Re: umk static version
Posted by Novo on Wed, 01 Apr 2020 13:38:30 GMT

dolik.rce wrote on Wed, 01 April 2020 06:30It would not work well with custom assemblies like MyApps, but those are probably not build with umk that often.
Well, I wouldn't agree with that. I'm using umk with MyApps all the time. This is my workflow. I'm using TheIDE only when I need debugger.

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 14:20:56 GMT

mirek wrote on Wed, 01 April 2020 10:26Quote:
amrein wrote on Wed, 01 April 2020 10:05
I would like to change these defaults here. My current idea, maybe wrong, is that U++ will scan folders from the one where binary is up to root and tries to find ".config" (or .upp, not quite decided yet). Only if it does not find one, it would use ~/.upp/umk or ~/.config/umk.

Alternatively (or additionally), I can imagine that umk can easily run without configuration files...

It's better to use default config folder on Windows, Linux or Mac and have the opportunity to define your own config folder for package sandboxing.

Not sure about Linux/Mac, but after 15 years of trying various things, current approach used in Windows is far the best. That is: everything is stored in the same folder you get after unpacking the archive.

Using default config folder (like ~/.config) has various disadvantages compared to this simplistic approach. E.g. you cannot have two isolated U++ variants at the same time. If you delete U++ folder, you still have remnants in ~/.config. Etc...

So yes, my personal preference for improvement here is to sandbox it.

Now the suggested mechanism is not quite elegant. It is just the first approach that came to my mind to solve the problem in a way to keep things sanboxed while allowing normal ~/.config too (developed application would be normally produced to upp/out, so would pick upp/.config as well; when installed elsewhere, these would default to standard ~/.config).

Let says we do just this. So we will have two user cases:

1. Source package

* We extract and compile theide and umk
* We create our own projects in the same directory (in $HOME/upp-1234/MyApps for example)
* theide uses the current $HOME/upp-1234/uppsrc and $HOME/upp-1234/config/upp/theide directories when it starts from $HOME/upp-1234/

* theide uses spellers from $HOME/upp-1234/config/theide/speller/
* We can checkout upp svn sources there (in $HOME/upp-1234/svn) or elsewhere using theide (if needed)
* theide and umk use $HOME/upp-1234/build.out as shared cache directory

2. Binary package (in /usr or /opt)

* theide and umk are in a read only directory ($bindir)
* We create our own projects somewhere else, in $HOME/upp-src/MyApps for example
* theide checkout upp svn sources in $HOME/upp-src/svn/ and uses it configs from $USER/.config/upp/theide directories
* theide uses spellers from $USER/.config/upp/theide/speller/
* theide and umk use something like $HOME/upp-src/build.out as cache directory (or /var/tmp/upp/build.out/ or /tmp/upp/build.out/)

Is that correct?

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 14:27:57 GMT
View Forum Message <> Reply to Message

Adding the notion of workspace could be interesting: a directory where your store your ide configuration + projects sources and binaries; even if yours projects can reference external sources or data.

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 14:54:29 GMT
View Forum Message <> Reply to Message

amrein wrote on Wed, 01 April 2020 16:20mirek wrote on Wed, 01 April 2020 10:26Quote:
amrein wrote on Wed, 01 April 2020 10:05
I would like to change these defaults here. My current idea, maybe wrong, is that U++ will scan folders from the one where binary is up to root and tries to find ".config" (or .upp, not quite decided yet). Only if it does not find one, it would use ~/.upp/umk or ~/.config/umk.

Alternatively (or additionally), I can imagine that umk can easily run without configuration files...

It's better to use default config folder on Windows, Linux or Mac and have the opportunity to define your own config folder for package sandboxing.

Not sure about Linux/Mac, but after 15 years of trying various things, current approach used in Windows is far the best. That is: everything is stored in the same folder you get after unpacking the archive.

Using default config folder (like ~/.config) has various disadvantages compared to this simplistic approach. E.g. you cannot have two isolated U++ variants at the same time. If you delete U++ folder, you still have remnants in ~/.config. Etc...

So yes, my personal preference for improvement here is to sandbox it.

Now the suggested mechanism is not quite elegant. It is just the first approach that came to my mind to solve the problem in a way to keep things sanboxed while allowing normal ~/.config too (developed application would be normally produced to upp/out, so would pick upp/.config as well; when installed elsewhere, these would default to standard ~/.config).

Let says we do just this. So we will have two user cases:

1. Source package

* We extract and compile theide and umk
* We create our own projects in the same directory (in $HOME/upp-1234/MyApps for example)
* theide uses the current $HOME/upp-1234/uppsrc and $HOME/upp-1234/config/upp/theide directories when it starts from $HOME/upp-1234/
* theide uses spellers from $HOME/upp-1234/config/theide/speller/
* We can checkout upp svn sources there (in $HOME/upp-1234/svn) or elsewhere using theide (if needed)
* theide and umk use $HOME/upp-1234/build.out as shared cache directory

2. Binary package (in /usr or /opt)

* theide and umk are in a read only directory ($bindir)
* We create our own projects somewhere else, in $HOME/upp-src/MyApps for example
* theide checkout upp svn sources in $HOME/upp-src/svn/ and uses it configs from $USER/.config/upp/theide directories
* theide uses spellers from $USER/.config/upp/theide/speller/
* theide and umk use something like $HOME/upp-src/build.out as cache directory (or /var/tmp/upp/build.out/ or /tmp/upp/build.out/)

Is that correct?

Basically yes, with very minor differences: I do not plan to add "-1234" to the package name and spellers will be in $HOME/upp. output will to ~/upp/.cache

Definitely binary package is completely different kind of beast, however I think the proposed solution adapts itself to it quite easily.

Not that things also work nicely if you move theide to e.g. ~/bin - it will start using 'regular' ~/.config and ~/.cache out of box...

BTW, I already have all this working, now trying to put together "demo" release for further discussion.

Mirek

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 14:55:47 GMT
View Forum Message <> Reply to Message

P.S.: You can as well easily achive ~/upp/theide to use ~/.config, ~/.cache : just delete ~/upp/.config and ~/upp.cache...

Subject: Re: umk static version
Posted by Novo on Wed, 01 Apr 2020 15:59:49 GMT
View Forum Message <> Reply to Message

I'm probably missing something, but I do not see a way to set configuration from mainconfig with umk.
Setting of individual flags is useful, but selecting a predefined configuration is much easier.

Another thing. IMHO, it would be useful to be able to launch a compiled app using umk. Output directory is different with TheIDE and umk even if build options are similar. That would simplify scripting and testing.

One more thing. IMHO, it would be useful to be able to pass similar options to TheIDE. Not just a project, but also configuration and build method.

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 18:39:34 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 01 April 2020 17:59I'm probably missing something, but I do not see a way to set configuration from mainconfig with umk.
Setting of individual flags is useful, but selecting a predefined configuration is much easier.

With current umk

- first configuration is selected if you do not specify any flags
- not sure how would you speficy other configuration - the most straighforward is using flags

Quote:
One more thing. IMHO, it would be useful to be able to pass similar options to TheIDE. Not just a project, but also configuration and build method.

theide actually understands umk commandline

Subject: Re: umk static version
Posted by Novo on Wed, 01 Apr 2020 19:08:46 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 01 April 2020 14:39theide actually understands umk commandline
This is what I see:
$ ide -h
Usage: theide -f [file..]
       theide [file..] // autodetection mode

Common options:
    -v or --version - displays information about version.
    -h or --help   - displays this site.

Advanced options:
    --scale $percent - scale interface by "percent" parameter.

Internal options (Should not be called by the user):
    --gdb_debug_break_process $pid - breaks debug process represented by "pid".

I can open a project via "ide /path/proj/proj.upp".
This is it.
Am I missing something?

This is not critical, but would be useful ...

I'm composing command line arguments for umk. It would be convenient to use the same set of arguments with theide when I need to debug my code.

At this time I'm able to generate log-file name for a project (this is where all log messages go). I'd like to keep this feature.

Subject: Re: umk static version
Posted by Novo on Wed, 01 Apr 2020 19:15:30 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 01 April 2020 14:39
- not sure how would you speficy other configuration - the most straighforward is using flags

mainconfig
 "Default" = "GUI MT",
 "StdAlloc" = "GUI MT USEMALLOC";

Something like "umk -C StdAlloc".
IMHO, this is much more convenient than "umk +GUI,MT,USEMALLOC".
And half of my projects define "StdAlloc" as "MT USEMALLOC" ...

Subject: Re: umk static version
Posted by amrein on Wed, 01 Apr 2020 19:47:12 GMT
View Forum Message <> Reply to Message

Current umk (rev 14206) doesn't search its config in .config/umk and it uses .config/umk/_out as cache dir instead of .cache/upp.out/

Subject: Re: umk static version
Posted by mirek on Wed, 01 Apr 2020 21:35:17 GMT
View Forum Message <> Reply to Message

amrein wrote on Wed, 01 April 2020 21:47Current umk (rev 14206) doesn't search its config in .config/umk and it uses .config/umk/_out as cache dir instead of .cache/upp.out/

Sorry, not yet committed, work in progress.

Subject: Re: umk static version
Posted by mirek on Thu, 02 Apr 2020 07:22:33 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 01 April 2020 21:15mirek wrote on Wed, 01 April 2020 14:39
- not sure how would you speficy other configuration - the most straighforward is using flags

mainconfig
 "Default" = "GUI MT",
 "StdAlloc" = "GUI MT USEMALLOC";

Something like "umk -C StdAlloc".
IMHO, this is much more convenient than "umk +GUI,MT,USEMALLOC".
And half of my projects define "StdAlloc" as "MT USEMALLOC" ...

BTW, MT is now (since ~2016) implicit.

So in reality, it is GUI,USEMALLOC. But OK, we can have that.

Subject: Re: umk static version
Posted by mirek on Thu, 02 Apr 2020 14:36:37 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 01 April 2020 17:59
Another thing. IMHO, it would be useful to be able to launch a compiled app using umk. Output directory is different with TheIDE and umk even if build options are similar. That would simplify scripting and testing.

Agreed, this one is must.

Subject: Re: umk static version
Posted by mirek on Thu, 02 Apr 2020 15:35:20 GMT
View Forum Message <> Reply to Message

Another NTH feature for umk: Semi-silent mode that would show text progress bar (similar to one shown e.g. when downloading packages).
