Subject: [POLL] Should we upgrade umk command line behavior ?
Posted by amrein on Tue, 14 Apr 2020 11:15:53 GMT
View Forum Message <> Reply to Message

Here is the new command line behavior proposition.
Please comment if you would like additions of corrections to this list.

NAME
umk - Ultimate++ make utility to maintain programs

SYNOPSIS
umk <assembly> <package> [OPTION] [output]

DESCRIPTION
The umk utility will determine automatically which pieces of a program need to be recompiled, and issue the commands to recompile them.
It can also export all files or only used files to another directory for the selected package.

MAIN PARAMETERS
This parameters are the most used by umk users.

<assembly>
Comma separated paths to source directories (nest<,nest<,..>>) or path to .var file.
A nest is a directory containing one or more packages.

<package>
Package name. It's the same name as the directory containing the '<package>.upp' configuration file.

[output]
The name of the final executable, library or the name of the directory where umk will export project files.
If omitted, <package> name will be used for binary name or in export mode, it will use current path to export project files.

OPTIONS
umk accepts the following options.

!
Start package executable or build it first if it does not exist.

+FLAG<,FLAG<,..>>
Comma separated list of build flags. Default value is 'GUI,MT'.

--
Signal the end of options. This is useful to allow further arguments to the umk program itself to

start with a "-".
  This provides consistency with the argument parsing convention used by most other POSIX programs.

 -b <build_method>, --build-method=<build_method>
  Build method name or path to build method description file (file extension: .bm). If not provided, the default value is 'CLANG'.

 -C <path>, --directory=<path>
  Change to directory path before doing anything else.

 --cache=<path>
  Path to cache directory (default ./.cache/u++ if it exists or ~/.cache/u++ otherwise).

 --config=<path>
  Path to configuration files (default ./.config/u++/ if it exists or ~/.config/u++/ otherwise).

 -d[=FLAGS], --debug[=FLAGS]
  Print debugging information in addition to normal processing. The debugging information says which files are being considered
  for remaking, which file-times are being compared and with what results, which files actually need to be remade, which implicit
  rules are considered and which are applied e.g everything interesting about how umk decides what to do. If the FLAGS are omitted,
  then the behavior is the same as if -d or --debug was specified. FLAGS may be 'a' for all debugging output, 'b' for basic debugging
  (same as using -d), 'v' for more verbose basic debugging, 'i' for showing implicit rules, 'j' for details on invocation of
  commands, and 'm' for debugging while making the final result. Use 'n' to disable all previous debugging flags.

 -e <variable="value">, --environment-overrides <variable="value">
  Overwrite an internal build method value.

 -f file, --file=file, --makefile=FILE
  Use file as name for exported makefile.

 -h, --help
  Show this message.

 -i, --ignore-errors
  Ignore all errors in commands executed to remake files.

 -j <jobs>, --jobs=<jobs>
  Specifies the number of jobs (commands) to run simultaneously. If there is more than one -j option, the last one is effective. If the
  -j option is omitted, umk will run simultaneously as many jobs as available processor logical cores.

-n, --just-print, --dry-run
Print the commands that would be executed, but do not execute them.

-p, --print-data-base
Print the data base (rules and variable values) that results from reading package configuration files (extension: .upp) and the build
method file (extension: .bm); then execute as usual or as otherwise specified.

-q, --question
Return an exit status that is zero if the specified targets are already up to date or nonzero otherwise but do not run any commands,
or print anything.

-s, --silent, --quiet
Silent mode; do not print the commands as they are executed.

-o, --outmode=<mode<mode<..>>>
Mode:
a Clean cache and rebuild all.
b Use BLITZ technology to speed the build process.
c Clean cache.
v Print the pourcentage of avancement.
s Use shared libraries.
S Use shared libraries and build as shared libraries.
m Create a makefile (use the -f option to override the default name 'Makefile').
M Create a map file.
x Export project (to current directory or <output> directory if specified), export only files used by project.
X Export project (to current directory or <output> directory if specified), export all files.
k Do not delete target directory <output> before exporting. The target directory will not be deleted if one of the package
nest is inside the output directory but umk will still print a warning.

-v, --version
Print the version of the umk program and its copyright.

EXAMPLES

Build release version of TheIDE using CLANG build method:
umk uppsrc ide -o rs ./theide

Use custom build method using blitz technology and show progress poucentage:
umk uppsrc ide -b ./CPP.bm -o bsv ./theide

Rebuild Bombs example in full debug mode:
umk examples,uppsrc Bombs --debug=a

Build MyApp using its assembly configuration file if it exist in default u++ config directory:
umk myassembly.var MyApp

EXIT STATUS
umk exits with a status of zero if no targets that were built failed. A status of one will be returned if the -q flag was used and umk determines that a target needs to be rebuilt. A status of two will be returned if any errors were encountered.

Should we sacrifice backward compatibility in umk to make commandline nicer?(total votes: 8)

Yes, I 100% agree! 6/(75%)
Yes, but I would want a few additions here and there... 0/(0%)
No but I'm not against umk moving forward otherwise 2/(25%)
No! I depends on umk current behavior and I can't stick with an older binary 0/(0%)

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by mirek on Tue, 14 Apr 2020 11:38:58 GMT
View Forum Message <> Reply to Message

I have fixed the poll to make it more clear...

Still, I think that there is no reason to sacrifice compatibility. The differences in what is possible without sacrificing it and what is possible while doing so are IMO negligible.

EDIT: Actually, the only meaningfull difference there is that you insist that 3rd non-option parameter is "output", while backward compatibility requires "build method" (which you want to supply with -b only). Is that really worth it?

Mirek

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by dolik.rce on Tue, 14 Apr 2020 11:55:19 GMT
View Forum Message <> Reply to Message

Can't we just keep both old and new options parser for some time? It should be relatively simple to use correct one based on number of parameters, presence of '--' and/or '+FLAGS'...

Honza

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by amrein on Tue, 14 Apr 2020 13:12:12 GMT

mirek wrote on Tue, 14 April 2020 13:38I have fixed the poll to make it more clear...

Still, I think that there is no reason to sacrifice compatibility. The differences in what is possible without sacrificing it and what is possible while doing so are IMO negligible.

EDIT: Actually, the only meaningfull difference there is that you insist that 3rd non-option parameter is "output", while backward compatibility requires "build method" (which you want to supply with -b only). Is that really worth it?

Mirek

"Should we upgrade umk command line behavior" => "Should we sacrifice backward compatibility in umk to make command line nicer?" :lol:

It's clearly a more oriented question. Only early adopters who adapt easily to new things will say "yes, we should".
And personally, my understanding of this modified question is: "Should we break umk or not". :d

It's important to note that at present, noone has replied "I depends on umk current behavior and I can't stick with an older binary".

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by mirek on Tue, 14 Apr 2020 13:32:46 GMT

amrein wrote on Tue, 14 April 2020 15:12mirek wrote on Tue, 14 April 2020 13:38I have fixed the poll to make it more clear...

Still, I think that there is no reason to sacrifice compatibility. The differences in what is possible without sacrificing it and what is possible while doing so are IMO negligible.

EDIT: Actually, the only meaningfull difference there is that you insist that 3rd non-option parameter is "output", while backward compatibility requires "build method" (which you want to supply with -b only). Is that really worth it?

Mirek

"Should we upgrade umk command line behavior" => "Should we sacrifice backward compatibility in umk to make command line nicer?" :lol:

It's clearly a more oriented question. Only early adopters who adapt easily to new things will say "yes, we should".

Well, I think that the fact it actually IS breaking the compatibility should be clearly stated there.

Which it was not.

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by mirek on Tue, 14 Apr 2020 15:19:29 GMT
View Forum Message <> Reply to Message

amrein wrote on Tue, 14 April 2020 13:15
+FLAG<,FLAG<,..>>
Comma separated list of build flags. Default value is 'GUI,MT'.

Just a little comment, this should be ". MT flag is deprecated 4 years (it is MT always now) and IME you are as liely or more likely to build console apps. And empty string is more consistent default.

Subject: Re: [POLL] Should we upgrade umk command line behavior ?
Posted by amrein on Tue, 14 Apr 2020 16:09:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 14 April 2020 17:19amrein wrote on Tue, 14 April 2020 13:15
+FLAG<,FLAG<,..>>
Comma separated list of build flags. Default value is 'GUI,MT'.

Just a little comment, this should be ". MT flag is deprecated 4 years (it is MT always now) and IME you are as likely or more likely to build console apps. And empty string is more consistent default.

Thank for the info.
With the Makefile/domake I'm working on, you can build and start all packages from bazaar, examples, reference and tutorial from command line (as umk can do now).