Subject: Gearing up for 2022.1 release...
Posted by mirek on Thu, 03 Mar 2022 08:29:49 GMT
View Forum Message <> Reply to Message

IME support took longer than expected, but with that done, I think it is time for another release.

Current list of major changes:

- sizeof(wchar) is changed to 4 (32 bits) to support non BMP unicode characters

  This might bring some incompatibilities in the code that expects wchar to be 16 bit, which escpecially involves dealing with Win32 (and to lesser extend MacOS) APIs, so if your application
  is doing that, please check all instances of WCHAR (UniChar on MacOS) or even wchar especially type casts.

  To support host APIs, char16 is introduced (but there is no 16-bit String varian).

  Use ToSystemCharsetW, FromSystemCharsetW to convert texts to Win32 API.

- Support of drawing non-BMP characters in GUI
- Vastly improved character font replacement code (when drawing characters missing with requested font, replacement font is used)
- Last instances of Win32 ANSI calls (those ending with A) are removed
- UTF handling routines are refactored and their's naming is unified
- RTF is now being able to handle non-BMP characters (RTF is used as clipboard format for RichText)
- Improved input method (aka preedit in Linux, aka marked text in MacOS) support

Other minor changes:

- fixed TryRealloc issue
- improved MemoryCheck
- Removed MemoryAlloc48/MemoryFree48
- In theide Background parsing should less often cause delays in the main thread

Win32 release:

- updated clang to actual version (llvm14)
- zlib updated to 1.2.12
- openssl updated to 1.1.1n
- jpeg updated to 9e

If you have any issue to fix before the release, please post it here.

Subject: Re: Gearing up for 2022.1 release...
Posted by pvictor on Thu, 03 Mar 2022 10:07:35 GMT
View Forum Message <> Reply to Message

Hello,

Is it possible to add ExecuteSelectDir() method to FileSelNative class?

Best regards,
Victor

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Thu, 03 Mar 2022 10:13:17 GMT
View Forum Message <> Reply to Message

pvictor wrote on Thu, 03 March 2022 11:07Hello,

Is it possible to add ExecuteSelectDir() method to FileSelNative class?

Best regards,
Victor

Probably, do we have implementation for all 3 platforms?

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by pvictor on Thu, 03 Mar 2022 10:21:46 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 03 March 2022 15:13pvictor wrote on Thu, 03 March 2022 11:07Hello,

Is it possible to add ExecuteSelectDir() method to FileSelNative class?

Best regards,
Victor

Probably, do we have implementation for all 3 platforms?

Mirek

But it is implemented somehow in FileSel for all platforms.

Victor

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Thu, 03 Mar 2022 15:11:54 GMT
View Forum Message <> Reply to Message

pvictor wrote on Thu, 03 March 2022 11:21mirek wrote on Thu, 03 March 2022 15:13pvictor wrote on Thu, 03 March 2022 11:07Hello,

Is it possible to add ExecuteSelectDir() method to FileSelNative class?

Best regards,
Victor

Probably, do we have implementation for all 3 platforms?

Mirek

But it is implemented somehow in FileSel for all platforms.

Victor

How is that supposed to help? :)

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by pvictor on Fri, 04 Mar 2022 04:34:06 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 03 March 2022 20:11

How is that supposed to help? :)

Mirek

It doesn't help, you are right.

Directory selection must be in all native APIs:
Linux: gtk_file_chooser_dialog_new(..., GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER, ...);
Windows: SHBrowseForFolderA(...);
Mac: NSOpenPanel can choose a directory.

Victor

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Fri, 04 Mar 2022 20:09:39 GMT
View Forum Message <> Reply to Message

Hello Mirek,

We need this (GitHub PR) UppHub improvements. It checks for git and adds git dependency.

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 06 Mar 2022 15:13:16 GMT
View Forum Message <> Reply to Message

pvictor wrote on Fri, 04 March 2022 05:34mirek wrote on Thu, 03 March 2022 20:11

How is that supposed to help? :)

Mirek

It doesn't help, you are right.

Directory selection must be in all native APIs:
Linux: gtk_file_chooser_dialog_new(..., GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER, ...);
Windows: SHBrowseForFolderA(...);
Mac: NSOpenPanel can choose a directory.

Victor

OK, not too much happy about this late new feature, but done... Pls check.

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 06 Mar 2022 19:51:27 GMT
View Forum Message <> Reply to Message

Klugier wrote on Fri, 04 March 2022 21:09Hello Mirek,

We need this (GitHub PR) UppHub improvements. It checks for git and adds git dependency.

Klugier

OK; I have fixed English in that Excalamtion...

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Sun, 06 Mar 2022 20:02:59 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Sure. Thanks for applying!

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by pvictor on Mon, 07 Mar 2022 13:08:48 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 06 March 2022 20:13
OK, not too much happy about this late new feature, but done... Pls check.

I've checked it on Linux.
It works OK.
Thank you.

Victor

Subject: Re: Gearing up for 2022.1 release...
Posted by pvictor on Wed, 09 Mar 2022 09:18:31 GMT
View Forum Message <> Reply to Message

pvictor wrote on Mon, 07 March 2022 18:08mirek wrote on Sun, 06 March 2022 20:13
OK, not too much happy about this late new feature, but done... Pls check.

I've checked it on Linux.
It works OK.
Thank you.

Victor

FileSelNative works fine both in Linux and Windows.
However the public interfaces aren't the same:

Windows	Linux
void Serialize(Stream& s);	
void New();	
bool IsNew() const;	
bool Execute(bool open, const char *title = NULL);	bool Execute(bool open, const char *title = NULL);
bool ExecuteOpen(const char *title = NULL);	bool ExecuteOpen(const char *title = NULL);
bool ExecuteSaveAs(const char *title = NULL);	bool ExecuteSaveAs(const char *title = NULL);
bool ExecuteSelectDir(const char *title = NULL);	bool ExecuteSelectDir(const char *title = NULL);
String Get() const;	String Get() const;
void Set(const String& s);	void Set(const String& s);
operator String() const;	operator String() const;
void operator=(const String& s);	void operator=(const String& s);
String operator~() const;	String operator~() const;
void operator<<=(const String& s);	void operator<<=(const String& s);
int GetCount() const;	int GetCount() const;
const String& operator[](int i) const;	const String& operator[](int i) const;
bool GetReadOnly() const;	
String GetActiveDir() const;	
FileSelNative& Type(const char *name, const char *ext);	FileSelNative& Type(const char *name, const char *ext);
FileSelNative& AllFilesType();	FileSelNative& AllFilesType();
FileSelNative& ActiveDir(const String& dir);	FileSelNative& ActiveDir(const String& dir);
FileSelNative& ActiveType(int i);	FileSelNative& ActiveType(int i);
FileSelNative& DefaultExt(const char *ext);	
FileSelNative& Multi(bool b = true);	FileSelNative& Multi(bool b = true);
FileSelNative& ReadOnlyOption(bool b = true);	
FileSelNative& Asking(bool b = true);	FileSelNative& Asking(bool b = true);
FileSelNative& NoAsking();	FileSelNative& NoAsking();
	FileSelNative& ShowHidden(bool b = true);

Best regards,

Victor

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Fri, 11 Mar 2022 11:35:03 GMT
View Forum Message <> Reply to Message

Hi,

I just tried to do a fresh install of the current nightly on a clean Raspberry Pi 4 system. I cannot seem to get the dependencies right. First it was package 'fontconfig' but I seem to end up in a complicated dependency mess that cannot be solved.

The OS is the latest Raspbian OS (11 bullseye) updated to date.

Does anybody have a similar setup to test this and possibly solve the issue.

Thanks and best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Fri, 11 Mar 2022 11:42:34 GMT
View Forum Message <> Reply to Message

Hi,

Under Windows, compiling with 'MSBT22x64 Release' I get:
C:\upp-git\upp.src\uppsrc\Draw\FontCR.cpp(440): warning C4244: 'argument': conversion from 'T' to 'int', possible loss of data
with
[
T=double
]

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Thu, 17 Mar 2022 12:23:47 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 11 March 2022 12:42Hi,

Under Windows, compiling with 'MSBT22x64 Release' I get:
C:\upp-git\upp.src\uppsrc\Draw\FontCR.cpp(440): warning C4244: 'argument': conversion from 'T' to 'int', possible loss of data
with
[
T=double
]

Best regards,

Tom

Hopefully fixed.

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Thu, 17 Mar 2022 13:32:14 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks!

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sat, 19 Mar 2022 09:39:42 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 11 March 2022 12:35Hi,

I just tried to do a fresh install of the current nightly on a clean Raspberry Pi 4 system. I cannot seem to get the dependencies right. First it was package 'fontconfig' but I seem to end up in a complicated dependency mess that cannot be solved.

The OS is the latest Raspbian OS (11 bullseye) updated to date.

Can you give me download link? (Just to be 100% sure I am testing the same thing).

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sat, 19 Mar 2022 10:38:15 GMT
View Forum Message <> Reply to Message

Hi,

I just upgraded a buster to bullseye by switching the sources to point to bullseye instead of buster, and then did a full distupgrade. After that I downloaded latest upp nightly and tried to proceed as normal with ./install.

Best regards,

Tom

EDIT: UPDATE: I booted my RPi 4 system to look for proper version info for you, but now I discovered additional issues with my OS. I may in fact need to do a fresh OS re-install from this link and see if it helps:

https://downloads.raspberrypi.org/raspios_arm64/images/raspi
os_arm64-2022-01-28/2022-01-28-raspios-bullseye-arm64.zip

Anyway, this is the one I'm supposed to be running.

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sat, 19 Mar 2022 16:52:19 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Now I'm on the freshly installed Raspberry OS 64-bit, the same as I linked for you. Here the ./install works for setting up the dependencies and starts compilation as expected.

However, the compilation fails with error:

```
./uppsrc/Core/Ops.h: In function 'Upp::byte Upp::addc64(Upp::uint64&, const uint64&, Upp::byte)':
./uppsrc/Core/Ops.h:223:9: error: '_addcarry_u64' was not declared in this scope
  223 |  return _addcarry_u64(carry, result, value, &result);
```

I do not know if there are more issues as the compilation stops to this error.

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Sat, 19 Mar 2022 17:42:24 GMT
View Forum Message <> Reply to Message

Hello Tom,

I have the same problem on M1 Mac when compiling application for native executable. _addcarry_u64 is not compatible with ARM architecture, so I think there must be a problem with #ifdefs.

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sun, 20 Mar 2022 07:47:20 GMT
View Forum Message <> Reply to Message

Hi Klugier and Mirek,

I agree. Tried to change the #if on line 219 of Ops.h:
#if defined(__SIZEOF_INT128__) && (__GNUC__ > 5 || __clang_major__ >= 5) && !defined(CPU_ARM)

Adding "&& !defined(CPU_ARM)", in the end of the line, enabled successful compilation on current Raspberry OS (64-bit). However, I do not know if this change would cause issues on some other platform.

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 20 Mar 2022 08:35:57 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 20 March 2022 08:47Hi Klugier and Mirek,

I agree. Tried to change the #if on line 219 of Ops.h:
#if defined(__SIZEOF_INT128__) && (__GNUC__ > 5 || __clang_major__ >= 5) && !defined(CPU_ARM)

Adding "&& !defined(CPU_ARM)", in the end of the line, enabled successful compilation on current Raspberry OS (64-bit). However, I do not know if this change would cause issues on some other platform.

Best regards,

Tom

Well, but that would bring in ugly mul64. I have tried better approach in the trunk... (but it is worth checking whether addc64 is compiled into something reasonable. It probably should get optimised out...)

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sun, 20 Mar 2022 09:00:00 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Good point. Your code compiles and seems to work fine on Raspberry. I'm just wondering if CPU_X86 is defined for 64-bit AMD architecture, or should you include CPU_AMD64 flag too?

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 20 Mar 2022 09:16:41 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 20 March 2022 10:00Hi Mirek,

Good point. Your code compiles and seems to work fine on Raspberry. I'm just wondering if CPU_X86 is defined for 64-bit AMD architecture, or should you include CPU_AMD64 flag too?

Best regards,

Tom

It is. (Check config.h)

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sun, 20 Mar 2022 09:24:42 GMT
View Forum Message <> Reply to Message

OK, thanks! So, it's all good now.

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by coolman on Sun, 20 Mar 2022 09:33:36 GMT
View Forum Message <> Reply to Message

Hi Mirek,

With the commit d3ef160f104a181eba9aed10173762382838f39f (TabBar: ConfirmCloseSome) I'm not able to compile TheIde. I got error

```
/Develop/upp/./uppsrc/Core/Function.h:17:50: error: no matching function for call to object of type '(lambda at /Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)'
            virtual Res Execute(ArgTypes... args) { return fn(args...); }
                                                    ^~
/Develop/upp/./uppsrc/Core/Function.h:19:3: note: in instantiation of member function 'Upp::Function<bool (Upp::ValueArray)>::Wrapper<(lambda at /Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>::Execute' requested here
            Wrapper(F&& fn) : fn(pick(fn)) {}
            ^
/Develop/upp/./uppsrc/Core/Function.h:56:57: note: in instantiation of member function 'Upp::Function<bool (Upp::ValueArray)>::Wrapper<(lambda at Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>::Wrapper' requested here
      template <class F> Function(F fn)         { ptr = new Wrapper<F>(pick(fn)); }
                                                      ^
/Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: in instantiation of function template specialization 'Upp::Function<bool (Upp::ValueArray)>::Function<(lambda at /Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>' requested here
      ConfirmCloseSome = []() { return true; };
                  ^
Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: candidate function not viable: requires 0 arguments, but 1 was provided
Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: conversion candidate of type 'bool (*)()'
```

```
clang -v
Ubuntu clang version  12.0.1-++20211029101322+fed41342a82f-1~exp1~20211029221816.4
Target: x86_64-pc-linux-gnu
Thread model: posix
```

```
uname -a
Linux 5.13.0-35-generic #40~20.04.1-Ubuntu SMP Mon Mar 7 09:18:32 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

BR, Radek

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Sun, 20 Mar 2022 12:31:43 GMT
View Forum Message <> Reply to Message

Hi,

I can confirm this issue on Raspberry when building theide. (Previously I did not notice it when working with nightly and then UWord for testing.)

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 20 Mar 2022 13:26:44 GMT
View Forum Message <> Reply to Message

coolman wrote on Sun, 20 March 2022 10:33Hi Mirek,

With the commit d3ef160f104a181eba9aed10173762382838f39f (TabBar: ConfirmCloseSome) I'm not able to compile TheIde. I got error

```
/Develop/upp/./uppsrc/Core/Function.h:17:50: error: no matching function for call to object of type
'(lambda at /Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)'
        virtual Res Execute(ArgTypes... args) { return fn(args...); }
                                                ^~
/Develop/upp/./uppsrc/Core/Function.h:19:3: note: in instantiation of member function
'Upp::Function<bool (Upp::ValueArray)>::Wrapper<(lambda at
/Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>::Execute' requested here
        Wrapper(F&& fn) : fn(pick(fn)) {}
        ^
/Develop/upp/./uppsrc/Core/Function.h:56:57: note: in instantiation of member function
'Upp::Function<bool (Upp::ValueArray)>::Wrapper<(lambda at
Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>::Wrapper' requested here
    template <class F> Function(F fn)         { ptr = new Wrapper<F>(pick(fn)); }
                                              ^
/Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: in instantiation of function template
specialization 'Upp::Function<bool (Upp::ValueArray)>::Function<(lambda at
/Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21)>' requested here
    ConfirmCloseSome = []() { return true; };
                ^
Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: candidate function not viable: requires 0
arguments, but 1 was provided
Develop/upp/uppsrc/TabBar/TabBar.cpp:334:21: note: conversion candidate of type 'bool (*)()'
```

clang -v

Ubuntu clang version 12.0.1-++20211029101322+fed41342a82f-1~exp1~20211029221816.4
Target: x86_64-pc-linux-gnu
Thread model: posix

uname -a
Linux 5.13.0-35-generic #40~20.04.1-Ubuntu SMP Mon Mar 7 09:18:32 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux

BR, Radek

Hopefully fixed.

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 20 Mar 2022 13:42:06 GMT
View Forum Message <> Reply to Message

pvictor wrote on Wed, 09 March 2022 10:18pvictor wrote on Mon, 07 March 2022 18:08mirek wrote on Sun, 06 March 2022 20:13
OK, not too much happy about this late new feature, but done... Pls check.

I've checked it on Linux.
It works OK.
Thank you.

Victor

FileSelNative works fine both in Linux and Windows.
However the public interfaces aren't the same:

Windows	Linux
void Serialize(Stream& s);	
void New();	
bool IsNew() const;	
bool Execute(bool open, const char *title = NULL);	bool Execute(bool open, const char *title = NULL);
bool ExecuteOpen(const char *title = NULL);	bool ExecuteOpen(const char *title = NULL);
bool ExecuteSaveAs(const char *title = NULL);	bool ExecuteSaveAs(const char *title = NULL);
bool ExecuteSelectDir(const char *title = NULL);	bool ExecuteSelectDir(const char *title = NULL);
String Get() const;	String Get() const;
void Set(const String& s);	void Set(const String& s);
operator String() const;	operator String() const;
void operator=(const String& s);	void operator=(const String& s);

```
String operator~() const;                    String operator~() const;
void operator<<=(const String& s);                 void operator<<=(const String& s);
int GetCount() const;                        int GetCount() const;
const String& operator[](int i) const;            const String& operator[](int i) const;

bool GetReadOnly() const;
String GetActiveDir() const;

FileSelNative& Type(const char *name, const char *ext); FileSelNative& Type(const char *name,
const char *ext);
FileSelNative& AllFilesType();               FileSelNative& AllFilesType();
FileSelNative& ActiveDir(const String& dir);       FileSelNative& ActiveDir(const String& dir);
FileSelNative& ActiveType(int i);            FileSelNative& ActiveType(int i);

FileSelNative& DefaultExt(const char *ext);

FileSelNative& Multi(bool b = true);             FileSelNative& Multi(bool b = true);

FileSelNative& ReadOnlyOption(bool b = true);

FileSelNative& Asking(bool b = true);              FileSelNative& Asking(bool b = true);
FileSelNative& NoAsking();                   FileSelNative& NoAsking();

                                        FileSelNative& ShowHidden(bool b = true);
```

Best regards,
Victor

Should be now improved (but 100% is impossible).

Subject: Re: Gearing up for 2022.1 release...
Posted by coolman on Sun, 20 Mar 2022 19:04:35 GMT
View Forum Message <> Reply to Message

Hi,

I can confirm, that your TabBar fix is OK. I can sucessfully build TheIde.

BR, Radek

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Tue, 29 Mar 2022 11:30:35 GMT

View Forum Message <> Reply to Message

Hi Mirek,

Any status update for 2022.1 release schedule? :)

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Thu, 31 Mar 2022 02:10:05 GMT
View Forum Message <> Reply to Message

LLVM 14.0.0 was released several days ago.
IMHO, it makes sense to ship UPP with llvm-mingw compiler based on LLVM 14.

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Sun, 03 Apr 2022 13:43:55 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I also proposed bumping C++ standard from c++14 to c++17. However, this could be done in the next release. There are a lot of risks here in context of compilation on various platforms.

Even if we do not have any features that particularly targets c++17 we should compile with that standard and our users should have access to it by default. Also, maybe this is a bug, but for MSVC we do not force any standard. It is always latest. I think it should change and we should target exactly the same standard as for GCC and CLANG.

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 03 Apr 2022 17:33:31 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 31 March 2022 04:10LLVM 14.0.0 was released several days ago.
IMHO, it makes sense to ship UPP with llvm-mingw compiler based on LLVM 14.

Unfortunately:

https://github.com/mstorsjo/llvm-mingw/issues/271

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Sun, 03 Apr 2022 17:36:07 GMT
View Forum Message <> Reply to Message

Klugier wrote on Sun, 03 April 2022 15:43Hello Mirek,

I also proposed bumping C++ standard from c++14 to c++17. However, this could be done in the next release. There are a lot of risks here in context of compilation on various platforms.

Even if we do not have any features that particularly targets c++17 we should compile with that standard and our users should have access to it by default. Also, maybe this is a bug, but for MSVC we do not force any standard. It is always latest. I think it should change and we should target exactly the same standard as for GCC and CLANG.

Klugier

Yeeah, I was thinking about it a lot, problem is we have so far universal package for Posixes and we are not 100% sure c++17 compliant compiler is there. I think we would need to add detection code before adding -std=c++17 to options, which is sort of complicated and quirky.

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Sun, 03 Apr 2022 22:48:55 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 03 April 2022 13:33Novo wrote on Thu, 31 March 2022 04:10LLVM 14.0.0 was released several days ago.
IMHO, it makes sense to ship UPP with llvm-mingw compiler based on LLVM 14.

Unfortunately:

https://github.com/mstorsjo/llvm-mingw/issues/271

I just double-checked that (added -fno-limit-debug-info to "Debug options").
Everything compiles just fine ...
The only "problem" that I see is a lot of "clang-14: warning: argument unused during compilation: '-mthreads' [-Wunused-command-line-argument]"...

P.S. I checked that on Linux.

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Mon, 04 Apr 2022 15:38:27 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 04 April 2022 00:48mirek wrote on Sun, 03 April 2022 13:33Novo wrote on Thu, 31 March 2022 04:10LLVM 14.0.0 was released several days ago.

IMHO, it makes sense to ship UPP with llvm-mingw compiler based on LLVM 14.

Unfortunately:

https://github.com/mstorsjo/llvm-mingw/issues/271

I just double-checked that (added -fno-limit-debug-info to "Debug options").
Everything compiles just fine ...
The only "problem" that I see is a lot of "clang-14: warning: argument unused during compilation: '-mthreads' [-Wunused-command-line-argument]"...

P.S. I checked that on Linux.

You are right, it was my fault. Tomorrow nightly should be clang-14 (pls check)

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Tue, 05 Apr 2022 04:46:37 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 04 April 2022 11:38
You are right, it was my fault. Tomorrow nightly should be clang-14 (pls check)
Compilation is fine (at least on Linux).
I do see a weird problem with Md5Stream. I'll check it tomorrow.

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Tue, 05 Apr 2022 10:21:36 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 03 April 2022 19:36Klugier wrote on Sun, 03 April 2022 15:43Hello Mirek,

I also proposed bumping C++ standard from c++14 to c++17. However, this could be done in the next release. There are a lot of risks here in context of compilation on various platforms.

Even if we do not have any features that particularly targets c++17 we should compile with that standard and our users should have access to it by default. Also, maybe this is a bug, but for MSVC we do not force any standard. It is always latest. I think it should change and we should target exactly the same standard as for GCC and CLANG.

Klugier

Yeeah, I was thinking about it a lot, problem is we have so far universal package for Posixes and we are not 100% sure c++17 compliant compiler is there. I think we would need to add detection code before adding -std=c++17 to options, which is sort of complicated and quirky.

Hello Mirek,

What about chaning approch? If it doesn't compile you could always back to older stable release of Upp. The old systems shouldn't hold us back and keep with old standard. We could make 2022.1 last official release with c++14 and the next one will be c++17.

The C++14 was introduced in 2017 in our code base. 3 years after official standard announcement. Right now we are 5 years after c++17. There is no consistency here.

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Tue, 05 Apr 2022 13:00:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 04 April 2022 18:38You are right, it was my fault. Tomorrow nightly should be clang-14 (pls check)

Hi,

The latest nightly (upp-win-16223.7z) shows version 11.0.0. for clang:
C:\upp\bin\clang\bin>clang.exe --version
clang version 11.0.0 (https://github.com/llvm/llvm-project.git 176249bd6732a8044d457092ed932768724a6f06)
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: C:\upp\bin\clang\bin

C:\upp\bin\clang\bin>

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Tue, 05 Apr 2022 13:29:23 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 05 April 2022 09:00
The latest nightly (upp-win-16223.7z) shows version 11.0.0. for clang:

I personally checked against git ...

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Tue, 05 Apr 2022 19:00:20 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 05 April 2022 00:46
I do see a weird problem with Md5Stream. I'll check it tomorrow.
It turned out to be a weird problem with FindAllPaths + wine + unix-style path.
For examle:
FindAllPaths("/home/ssg/data/download", "*.pdf")
will return
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\mysqluc2007-wikipedia-workbook.pdf
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\recsplit.pdf
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\StatisticsWithJuliaDRAFT.pdf
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\UHHA Info Update 5-8-20.pdf
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\web-prolog.pdf
v =
Z:\home\ssg\dvlp\cpp\code\upp\out\MyApps\MINGWcpp17.Debug.Debug_Full.Gui.Shared.Win32\home\ssg\data\download\whole-book.pdf

This can be a problem with wine, Upp or both ...

Another thing. In case of Wine + Upp log-file is created in an app folder. SetConfigGroup is ignored. I guess this is done "by design".

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Wed, 06 Apr 2022 07:07:53 GMT
View Forum Message <> Reply to Message

It turned out that Windows has a command line package manager called "winget" now. (This, probably, was already discussed on this forum)
IMHO, it would be cool to have an Upp package for it.
Installing Upp via "winget install Upp" seems to be the easiest way to get it.
And "winget search upp" seems to be the easiest way to find it.

Subject: Re: Gearing up for 2022.1 release...

Posted by mirek on Wed, 06 Apr 2022 09:59:34 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 06 April 2022 09:07It turned out that Windows has a command line package manager called "winget" now. (This, probably, was already discussed on this forum)
IMHO, it would be cool to have an Upp package for it.
Installing Upp via "winget install Upp" seems to be the easiest way to get it.
And "winget search upp" seems to be the easiest way to find it.

Go for it :)

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Wed, 06 Apr 2022 10:05:48 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 05 April 2022 12:21mirek wrote on Sun, 03 April 2022 19:36Klugier wrote on Sun, 03 April 2022 15:43Hello Mirek,

I also proposed bumping C++ standard from c++14 to c++17. However, this could be done in the next release. There are a lot of risks here in context of compilation on various platforms.

Even if we do not have any features that particularly targets c++17 we should compile with that standard and our users should have access to it by default. Also, maybe this is a bug, but for MSVC we do not force any standard. It is always latest. I think it should change and we should target exactly the same standard as for GCC and CLANG.

Klugier

Yeeah, I was thinking about it a lot, problem is we have so far universal package for Posixes and we are not 100% sure c++17 compliant compiler is there. I think we would need to add detection code before adding -std=c++17 to options, which is sort of complicated and quirky.

Hello Mirek,

What about chaning approch? If it doesn't compile you could always back to older stable release of Upp. The old systems shouldn't hold us back and keep with old standard. We could make 2022.1 last official release with c++14 and the next one will be c++17.

The C++14 was introduced in 2017 in our code base. 3 years after official standard announcement. Right now we are 5 years after c++17. There is no consistency here.

Klugier

Uhm, now we are mixing 2 things I guess:

(1) C++ that is required by U++
(2) default C++ setting of build methods

So far I was speaking about (2). As for (1) I do not as of now see anything in C++17 that would make U++ codebase significantly better (cleaner, shorter, faster), so IMO it is not worth it to break U++ for old systems yet.

I could go with (2), but that would require detection system for posix install. IDK, maybe running "c++ -std=c++17" and detecting the error code would do the job?

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Wed, 06 Apr 2022 17:04:29 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 06 April 2022 05:59Novo wrote on Wed, 06 April 2022 09:07It turned out that Windows has a command line package manager called "winget" now. (This, probably, was already discussed on this forum)
IMHO, it would be cool to have an Upp package for it.
Installing Upp via "winget install Upp" seems to be the easiest way to get it.
And "winget search upp" seems to be the easiest way to find it.

Go for it :)
I'm not a Windows person.)
First of all I need to figure out how to legally get/build a QEMU Windows 11 VM which I need for testing.
Your suggestions are welcome!

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Wed, 06 Apr 2022 17:21:25 GMT
View Forum Message <> Reply to Message

Another issue.
Linking options from the "Output mode" dialog completely overwrite these options from a builder.
This is fine when you develop for only one platform. ("All static" for Windows and "Use shared libs" for POSIX)
But when you need to cross-compile this becomes a problem.
Suddenly your Windows apps start requiring dependencies ...
It took me quite a lot of time to realize that.

P.S. Interesting, what happens in case of umk ...

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Wed, 06 Apr 2022 17:34:59 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 06 April 2022 06:05
(1) C++ that is required by U++
(2) default C++ setting of build methods

IMHO, it makes sense to ship Upp with different C++-versioned builders.
CLANG14, CLANG17, e.t.c instead of just a CLANG ...
And Upp should be tested against all versions of C++ because old code often doesn't compile with new versions of C++.
I personally use C++17 because my own code doesn't compile with older versions of C++ 8)

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Thu, 07 Apr 2022 08:42:44 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 06 April 2022 19:34mirek wrote on Wed, 06 April 2022 06:05
(1) C++ that is required by U++
(2) default C++ setting of build methods

IMHO, it makes sense to ship Upp with different C++-versioned builders.
CLANG14, CLANG17, e.t.c instead of just a CLANG ...
And Upp should be tested against all versions of C++ because old code often doesn't compile with new versions of C++.
I personally use C++17 because my own code doesn't compile with older versions of C++ 8)

Yes, that is another possibility. Anyway, I postpone this until next release.

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Thu, 07 Apr 2022 10:16:08 GMT
View Forum Message <> Reply to Message

Hi,

Bundled clang in windows now seems up to version 14.0.0. :)

However, when linking my project with CLANG I get the following error:
Linking...
ld.lld: error: duplicate symbol: std::__throw_bad_alloc()
>>> defined at C:/upp-git/out/program/Core/CLANGx64.Blitz.Gui.Protect/heap.o
>>> defined at libc++.a(new.cpp.obj)
clang-14: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (1:31.25)

Please note that this is not specific to CLANG-14 but also happened with previous CLANG version bundled.

Any idea where this comes from and suggestions how to proceed?

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Klugier on Thu, 07 Apr 2022 10:55:27 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I still think that we should bump c++ version everywhere. There are some features that could be use within Upp code base such as structural binding, nested namespaces or declaring variable in if, switch statement. There are also a lot of features in the standard library like std::optional. Anyway, we should go forward. C++20 is a head of us with much more important feature such as modules and concepts. If we do not want to switch to c++17 right now, so when we will switch to c++20. In 2030?

If the POSIX bundle will do not compile on the oldest system you could always back to previous stable release which supports older standards and you could always install newer compiler version to overcome the problem. In reality to support c++17 we need compiler from 2016/2017.

I am not sure introducing CLANG17 will give us something instead of additional configuration file. You could switch standard by modifying CLANG inside TheIDE after installation and replacing it by -std=c++17. Very easy, however it would be good to have it by default.

Klugier

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Thu, 07 Apr 2022 11:21:37 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 07 April 2022 12:16Hi,

Bundled clang in windows now seems up to version 14.0.0. :)

However, when linking my project with CLANG I get the following error:
Linking...
ld.lld: error: duplicate symbol: std::__throw_bad_alloc()
>>> defined at C:/upp-git/out/program/Core/CLANGx64.Blitz.Gui.Protect/heap.o

>>> defined at libc++.a(new.cpp.obj)
clang-14: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (1:31.25)

Please note that this is not specific to CLANG-14 but also happened with previous CLANG version bundled.

Any idea where this comes from and suggestions how to proceed?

Best regards,

Tom

It rings some bells:

https://github.com/mstorsjo/llvm-mingw/issues/91

Core/heap.cpp:302

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Thu, 07 Apr 2022 11:40:04 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I thought, now that CLANG is at version 14, the issue would have been solved.
 mstorsjo commented on Apr 23, 2021

This should have been fixed by the previous release (based on LLVM 11.0.0).
Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Thu, 07 Apr 2022 13:07:00 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I commented out the following line in heap.cpp:
void __attribute__((__noreturn__)) std::__throw_bad_alloc (void) { throw bad_alloc(); }
As a result the linker stopped complaining about it. Is this line required at all anymore?

(There are some external static libs I need that still prevent my app from running, but that's another story.)

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Thu, 07 Apr 2022 20:57:53 GMT
View Forum Message <> Reply to Message

Klugier wrote on Thu, 07 April 2022 06:55
I still think that we should bump c++ version everywhere.
I have a different proposal. :) Upp doesn't really use any of C++14 language features. C++14 added improved constexpr support. Upp doesn't use it. It uses just a couple of C++14-specific stl files (maybe, just one). So, by dropping of this dependency and implementing this functionality in Upp itself it will be possible to lower C++ version requirement to C++11. :)

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Thu, 07 Apr 2022 21:21:26 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 06 April 2022 13:34
And Upp should be tested against all versions of C++ because old code often doesn't compile with new versions of C++.

TheIDE cannot be compiled with C++20.
Clang + -std=c++20

```
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/CtrlCore/CtrlPos.cpp:152:11: error: use of overloaded operator '!=' is ambiguous (with operand types 'Upp::Rect' (aka 'Rec
    t_<int>') and 'Upp::Rect16' (aka 'Rect_<short>'))
            if(view != f.view) {
               ~~~~ ^  ~~~~~~
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:337:9: note: candidate function
        bool   operator!=(const Rect_& b) const                { return !operator==(b); }
               ^
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:336:9: note: candidate function
        bool   operator==(const Rect_& b) const;
               ^
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:336:9: note: candidate function (with reversed parameter order)
```

IMHO, this should be fixed.

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Fri, 08 Apr 2022 07:43:00 GMT
View Forum Message <> Reply to Message

Hi Novo,

I like your idea of flexibility here. It is far easier to maintain projects across platforms if the latest uppsrc works nicely (i.e. is compatible) with all the standards from c++11 to c++20 -- or whatever the latest is. Anyway, there are always dependencies that cause all kinds of trouble with their requirements, so keeping uppsrc widely compatible is an excellent choice.

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Fri, 08 Apr 2022 09:09:40 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 07 April 2022 23:21Novo wrote on Wed, 06 April 2022 13:34
And Upp should be tested against all versions of C++ because old code often doesn't compile with new versions of C++.

TheIDE cannot be compiled with C++20.
Clang + -std=c++20
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/CtrlCore/CtrlPos.cpp:152:11: error: use of overloaded operator '!=' is ambiguous (with operand types 'Upp::Rect' (aka 'Rec
t_<int>') and 'Upp::Rect16' (aka 'Rect_<short>'))
if(view != f.view) {
~~~~ ^ ~~~~~~
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:337:9: note: candidate function
bool operator!=(const Rect_& b) const { return !operator==(b); }
^
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:336:9: note: candidate function
bool operator==(const Rect_& b) const;
^
/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Core/Gtypes.h:336:9: note: candidate function (with reversed parameter order)

IMHO, this should be fixed.

Unfortunately, that just seems to be the tip of the iceberg.

Frankly, are they paid for screwing our code every 3 years or what? :)

(Fixing things now..)
~~~~

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Fri, 08 Apr 2022 09:19:29 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 07 April 2022 22:57Klugier wrote on Thu, 07 April 2022 06:55
I still think that we should bump c++ version everywhere.
I have a different proposal. :) Upp doesn't really use any of C++14 language features.

Actually, we use C++14 lambda features that cannot be worked around. But I would like to stay there (at C++ 14).

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by mr_ped on Fri, 08 Apr 2022 10:42:42 GMT
View Forum Message <> Reply to Message

IMHO:
- upp packages "should" compile also in C++20 mode, so user apps can use the C++20 (would be really nice to have this fixed before 2022.1 release)
- changing minimal requirement from C++14 to C++17 without using it is useless, just adding artificial hurdle (some users may be forced to C++14 in their projects)
- refactoring the code to use C++17 just before 2022.1 release just for the sake of using C++17 is also weird use of dev-time

My understanding is that U++ is almost ready for release, and unless there is some practical value in using C++17 for remaining tasks, it is IMHO much better to release it as C++14 compatible (but ideally with C++17 and C++20 compatibility too), that enables most options for users.

And things like bumping minimal version should be rather done at beginning of the release (ideally when some new features also make sense and will be used actively), not in last minute before release.

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Fri, 08 Apr 2022 13:34:55 GMT
View Forum Message <> Reply to Message

I am really frustrated with C++20. It breaks backward compatibility huge way, I will probably have to spend weeks to fix my codebase :(What were they thinking?

Subject: Re: Gearing up for 2022.1 release...
Posted by Tom1 on Fri, 08 Apr 2022 14:15:36 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 08 April 2022 16:34I am really frustrated with C++20. It breaks backward compatibility huge way, I will probably have to spend weeks to fix my codebase :(What were they thinking?

How about not going to C++20 just yet? Instead, release the stable U++ 2022.1 and only thereafter re-start to look at the C++20 standard issues.

Best regards,

Tom

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Fri, 08 Apr 2022 14:50:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 08 April 2022 16:15mirek wrote on Fri, 08 April 2022 16:34I am really frustrated with C++20. It breaks backward compatibility huge way, I will probably have to spend weeks to fix my codebase :(What were they thinking?

How about not going to C++20 just yet? Instead, release the stable U++ 2022.1 and only thereafter re-start to look at the C++20 standard issues.

Best regards,

Tom

OK, I have probably exaggerted a bit.. :) Right now I have just one really annoying warning to fix...

That said, breaking overloading rules is not a good idea. Had to add quite some lines to support C++20...

Mirek

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Fri, 08 Apr 2022 15:04:29 GMT
View Forum Message <> Reply to Message

ide now compiles with C++ 20 with some warnings. If anybody had time to investigate these, it would be much appreciated...

C:\u\upp.src\uppsrc\RichText\ParaData.cpp (344): warning: ISO C++20 considers use of

overloaded operator '!=' (with operand types 'const
WithDeepCopy<Vector<Upp::RichPara::Tab>>' and 'const
WithDeepCopy<Vector<Upp::RichPara::Tab>>') to be ambiguou
s despite there being a unique best viable function [-Wambiguous-reversed-operator]

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Fri, 08 Apr 2022 19:37:14 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 08 April 2022 05:19
Actually, we use C++14 lambda features that cannot be worked around. But I would like to stay there (at C++ 14).

Mirek

Sorry, I missed that ...

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Fri, 08 Apr 2022 19:39:57 GMT
View Forum Message <> Reply to Message

MacOS 10.15
./umk uppsrc ide CLANG -bus
In file included from /Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/MKeys.h:3:
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:12:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_CTRL_KEY,
^
= 0
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:13:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_ALT_KEY,
^
= 0
...

Subject: Re: Gearing up for 2022.1 release...
Posted by mirek on Mon, 11 Apr 2022 10:07:46 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 08 April 2022 21:39MacOS 10.15
./umk uppsrc ide CLANG -bus

In file included from /Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/MKeys.h:3:
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:12:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_CTRL_KEY,
^
= 0
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:13:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_ALT_KEY,
^
= 0
...

Should be now fixed.

Subject: Re: Gearing up for 2022.1 release...
Posted by Novo on Mon, 11 Apr 2022 17:14:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 11 April 2022 06:07Novo wrote on Fri, 08 April 2022 21:39MacOS 10.15
./umk uppsrc ide CLANG -bus
In file included from /Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/MKeys.h:3:
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:12:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_CTRL_KEY,
^
= 0
/Users/ssg/worker0/m-upp/build/uppsrc/CtrlCore/CocoKeys.h:13:1: error: default initialization of an object of const type 'const Upp::dword' (aka 'const unsigned int')
K_ALT_KEY,
^
= 0
...

Should be now fixed.
Thank you!
It is fixed now. Compilation on MacOS 11.1 is fine as well.
