NAME

perlce - Perl for WinCE

Building Perl for WinCE

WARNING

Much of this document has become very out of date and needs updating, rewriting or deleting. The build process was overhauled during the 5.19 development track and the current instructions as of that time are given in *CURRENT BUILD INSTRUCTIONS*; the previous build instructions, which are largely superseded but may still contain some useful information, are left in *OLD BUILD INSTRUCTIONS* but really need removing after anything of use has been extracted from them.

DESCRIPTION

This file gives the instructions for building Perl5.8 and above for WinCE. Please read and understand the terms under which this software is distributed.

General explanations on cross-compiling WinCE

- *miniperl* is built. This is a single executable (without DLL), intended to run on Win32, and it will facilitate remaining build process; all binaries built after it are foreign and should not run locally.

  miniperl is built using *./win32/Makefile*; this is part of normal build process invoked as dependency from wince/Makefile.ce

- After *miniperl* is built, *configpm* is invoked to create right *Config.pm* in right place and its corresponding Cross.pm.

  Unlike Win32 build, miniperl will not have *Config.pm* of host within reach; it rather will use *Config.pm* from within cross-compilation directories.

  File *Cross.pm* is dead simple: for given cross-architecture places in @INC a path where perl modules are, and right *Config.pm* in that place.

  That said, `miniperl -Ilib -MConfig -we 1` should report an error, because it can not find *Config.pm*. If it does not give an error -- wrong *Config.pm* is substituted, and resulting binaries will be a mess.

  `miniperl -MCross -MConfig -we 1` should run okay, and it will provide right *Config.pm* for further compilations.

- During extensions build phase, a script *./win32/buildext.pl* is invoked, which in turn steps in *./ext* subdirectories and performs a build of each extension in turn.

  All invokes of *Makefile.PL* are provided with `-MCross` so to enable cross- compile.

CURRENT BUILD INSTRUCTIONS

(These instructions assume the host is 32-bit Windows. If you're on 64-bit Windows then change "C:\Program Files" to "C:\Program Files (x86)" throughout.)

1. Install EVC4 from

```
http://download.microsoft.com/download/c/3/f/c3f8b58b-9753-4c2e-8b96-2dfe3476a2f7/eVC4.exe
```

Use the key mentioned at

```
http://download.cnet.com/Microsoft-eMbedded-Visual-C/3000-2212_4-10108490.html?tag=bc
```

The installer is ancient and has a few bugs on the paths it uses. You will have to fix them later. Basically, some things go into "C:/Program Files/Windows CE Tools", others go into "C:/Windows CE Tools" regardless of the path you gave to the installer (the default will be "C:/Windows CE Tools"). Reboots will be required for the installer to proceed. Also .c and .h associations with Visual Studio might get overridden when installing EVC4. You have been warned.

2. Download celib from GitHub (using "Download ZIP") at

```
https://github.com/bulk88/celib
```

Extract it to a spaceless path but not into the perl build source. I call this directory "celib-palm-3.0" but in the GitHub snapshot it will be called "celib-master". Make a copy of the "wince-arm-pocket-wce300-release" folder and rename the copy to "wince-arm-pocket-wce400". This is a hack so we can build a CE 4.0 binary by linking in CE 3.0 ARM asm; the linker doesn't care. Windows Mobile/WinCE are backwards compatible with machine code like Desktop Windows.

3. Download console-1.3-src.tar.gz from

```
http://sourceforge.net/projects/perlce/files/PerlCE%20support%20files/conso
le/
```

Extract it to a spaceless path but not into the perl build source. Don't extract it into the same directory as celib. Make a copy of the "wince-arm-pocket-wce300" folder and rename the copy to "wince-arm-pocket-wce400". This is a hack so we can build a CE 4.0 binary by linking in CE 3.0 ARM asm; the linker doesn't care. Windows Mobile/WinCE are backwards compatible with machine code like Desktop Windows.

4. Open a command prompt, run your regular batch file to set the environment for desktop Visual C building, goto the perl source directory, cd into win32/, fill out Makefile, and do a "nmake all" to build a Desktop Perl.

5. Open win32/Makefile.ce in a text editor and do something similar to the following patch.

```
-CELIBDLLDIR  = h:\src\wince\celib-palm-3.0
-CECONSOLEDIR = h:\src\wince\w32console
+CELIBDLLDIR  = C:\sources\celib-palm-3.0
+CECONSOLEDIR = C:\sources\w32console
```

Also change

```
!if "$(MACHINE)" == ""
MACHINE=wince-arm-hpc-wce300
#MACHINE=wince-arm-hpc-wce211
#MACHINE=wince-sh3-hpc-wce211
#MACHINE=wince-mips-hpc-wce211
#MACHINE=wince-sh3-hpc-wce200
#MACHINE=wince-mips-hpc-wce200
#MACHINE=wince-arm-pocket-wce300
#MACHINE=wince-mips-pocket-wce300
#MACHINE=wince-sh3-pocket-wce300
#MACHINE=wince-x86em-pocket-wce300
#MACHINE=wince-mips-palm-wce211
#MACHINE=wince-sh3-palm-wce211
#MACHINE=wince-x86em-palm-wce211
#MACHINE=wince-x86-hpc-wce300
#MACHINE=wince-arm-pocket-wce400
!endif
```

to

```
!if "$(MACHINE)" == ""
#MACHINE=wince-arm-hpc-wce300
#MACHINE=wince-arm-hpc-wce211
#MACHINE=wince-sh3-hpc-wce211
#MACHINE=wince-mips-hpc-wce211
#MACHINE=wince-sh3-hpc-wce200
#MACHINE=wince-mips-hpc-wce200
#MACHINE=wince-arm-pocket-wce300
#MACHINE=wince-mips-pocket-wce300
#MACHINE=wince-sh3-pocket-wce300
#MACHINE=wince-x86em-pocket-wce300
#MACHINE=wince-mips-palm-wce211
#MACHINE=wince-sh3-palm-wce211
#MACHINE=wince-x86em-palm-wce211
#MACHINE=wince-x86-hpc-wce300
MACHINE=wince-arm-pocket-wce400
!endif
```

so wince-arm-pocket-wce400 is the MACHINE type.

6. Use a text editor to open "C:\Program Files\Microsoft eMbedded C++ 4.0\EVC\WCE400\BIN\WCEARMV4.BAT". Look for

```
if "%SDKROOT%"=="" set SDKROOT=...
```

On a new install it is "C:\Windows CE Tools". Goto "C:\Windows CE Tools" in a file manager and see if "C:\Windows CE Tools\wce400\STANDARDSDK\Include\Armv4" exists on your disk. If not the SDKROOT need to be changed to "C:\Program Files\Windows CE Tools".

Goto celib-palm-3.0\inc\cewin32.h, search for

```
typedef struct _ABC {
```

and uncomment the struct.

7. Open another command prompt, ensure PLATFORM is not set to anything already unless you know what you're doing (so that the correct default value is set by the next command), and run "C:\Program Files\Microsoft eMbedded C++ 4.0\EVC\WCE400\BIN\WCEARMV4.BAT"

8. In the WinCE command prompt you made with WCEARMV4.BAT, goto the perl source directory, cd into win32/ and run "nmake -f Makefile.ce".

9. The ARM perl interpreter (perl519.dll and perl.exe) will be in something like "C:\perl519\src\win32\wince-arm-pocket-wce400", with the XS DLLs in "C:\perl519\src\xlib\wince-arm-hpc-wce400\auto".

To prove success on the host machine, run "dumpbin /headers wince-arm-pocket-wce400\perl.exe" from the win32/ folder and look for "machine (ARM)" in the FILE HEADER VALUES and "subsystem (Windows CE GUI)" in the OPTIONAL HEADER VALUES.

OLD BUILD INSTRUCTIONS

This section describes the steps to be performed to build PerlCE. You may find additional information about building perl for WinCE at *http://perlce.sourceforge.net* and some pre-built binaries.

Tools & SDK

For compiling, you need following:

* Microsoft Embedded Visual Tools

* Microsoft Visual C++

* Rainer Keuchel's celib-sources

* Rainer Keuchel's console-sources

Needed source files can be downloaded at *http://perlce.sourceforge.net*

Make

Normally you only need to edit *./win32/ce-helpers/compile.bat* to reflect your system and run it.

File *./win32/ce-helpers/compile.bat* is actually a wrapper to call `nmake -f makefile.ce` with appropriate parameters and it accepts extra parameters and forwards them to `nmake` command as additional arguments. You should pass target this way.

To prepare distribution you need to do following:

* go to *./win32* subdirectory

* edit file *./win32/ce-helpers/compile.bat*

* run compile.bat

* run compile.bat dist

Makefile.ce has `CROSS_NAME` macro, and it is used further to refer to your cross-compilation scheme. You could assign a name to it, but this is not necessary, because by default it is assigned after your machine configuration name, such as "wince-sh3-hpc-wce211", and this is enough to distinguish different builds at the same time. This option could be handy for several different builds on same platform to perform, say, threaded build. In a following example we assume that all required environment variables are set properly for C cross-compiler (a special *.bat file could fit perfectly to this purpose) and your *compile.bat* has proper "MACHINE" parameter set, to, say, `wince-mips-pocket-wce300`.

```
compile.bat
compile.bat dist
compile.bat CROSS_NAME=mips-wce300-thr "USE_ITHREADS=define" ^
  "USE_IMP_SYS=define" "USE_MULTI=define"
compile.bat CROSS_NAME=mips-wce300-thr "USE_ITHREADS=define" ^
  "USE_IMP_SYS=define" "USE_MULTI=define" dist
```

If all goes okay and no errors during a build, you'll get two independent distributions: `wince-mips-pocket-wce300` and `mips-wce300-thr`.

Target `dist` prepares distribution file set. Target `zipdist` performs same as `dist` but additionally compresses distribution files into zip archive.

NOTE: during a build there could be created a number (or one) of *Config.pm* for cross-compilation ("foreign" *Config.pm*) and those are hidden inside *../xlib/$(CROSS_NAME)* with other auxiliary files, but, and this is important to note, there should be **no** *Config.pm* for host miniperl. If you'll get an error that perl could not find Config.pm somewhere in building process this means something went wrong. Most probably you forgot to specify a cross-compilation when invoking miniperl.exe to Makefile.PL When building an extension for cross-compilation your command line should look like

```
..\miniperl.exe -I..\lib -MCross=mips-wce300-thr Makefile.PL
```

or just

```
..\miniperl.exe -I..\lib -MCross Makefile.PL
```

to refer a cross-compilation that was created last time.

All questions related to building for WinCE devices could be asked in *perlce-user@lists.sourceforge.net* mailing list.

Using Perl on WinCE

DESCRIPTION

PerlCE is currently linked with a simple console window, so it also works on non-hpc devices.

The simple stdio implementation creates the files *stdin.txt*, *stdout.txt* and *stderr.txt*, so you might examine them if your console has only a limited number of cols.

When exitcode is non-zero, a message box appears, otherwise the console closes, so you might have to catch an exit with status 0 in your program to see any output.

stdout/stderr now go into the files */perl-stdout.txt* and */perl-stderr.txt.*

PerlIDE is handy to deal with perlce.

LIMITATIONS

No fork(), pipe(), popen() etc.

ENVIRONMENT

All environment vars must be stored in HKLM\Environment as strings. They are read at process startup.

PERL5LIB

Usual perl lib path (semi-list).

PATH

Semi-list for executables.

TMP

- Tempdir.

UNIXROOTPATH

- Root for accessing some special files, i.e. */dev/null*, */etc/services*.

ROWS/COLS

- Rows/cols for console.

HOME

- Home directory.

CONSOLEFONTSIZE

- Size for console font.

You can set these with cereg.exe, a (remote) registry editor or via the PerlIDE.

REGISTRY

To start perl by clicking on a perl source file, you have to make the according entries in HKCR (see *ce-helpers/wince-reg.bat*). cereg.exe (which must be executed on a desktop pc with ActiveSync) is reported not to work on some devices. You have to create the registry entries by hand using a registry editor.

XS

The following Win32-Methods are built-in:

```
newXS("Win32::GetCwd", w32_GetCwd, file);
newXS("Win32::SetCwd", w32_SetCwd, file);
newXS("Win32::GetTickCount", w32_GetTickCount, file);
newXS("Win32::GetOSVersion", w32_GetOSVersion, file);
newXS("Win32::IsWinNT", w32_IsWinNT, file);
newXS("Win32::IsWin95", w32_IsWin95, file);
newXS("Win32::IsWinCE", w32_IsWinCE, file);
newXS("Win32::CopyFile", w32_CopyFile, file);
newXS("Win32::Sleep", w32_Sleep, file);
newXS("Win32::MessageBox", w32_MessageBox, file);
newXS("Win32::GetPowerStatus", w32_GetPowerStatus, file);
newXS("Win32::GetOemInfo", w32_GetOemInfo, file);
newXS("Win32::ShellEx", w32_ShellEx, file);
```

BUGS

Opening files for read-write is currently not supported if they use stdio (normal perl file handles).

If you find bugs or if it does not work at all on your device, send mail to the address below. Please report the details of your device (processor, ceversion, devicetype (hpc/palm/pocket)) and the date of the downloaded files.

INSTALLATION

Currently installation instructions are at *http://perlce.sourceforge.net/*.

After installation & testing processes will stabilize, information will be more precise.

ACKNOWLEDGEMENTS

The port for Win32 was used as a reference.

History of WinCE port

5.6.0

Initial port of perl to WinCE. It was performed in separate directory named *wince*. This port was based on contents of *./win32* directory. *miniperl* was not built, user must have HOST perl and properly edit *makefile.ce* to reflect this.

5.8.0

wince port was kept in the same *./wince* directory, and *wince/Makefile.ce* was used to invoke native compiler to create HOST miniperl, which then facilitates cross-compiling process. Extension building support was added.

5.9.4

Two directories *./win32* and *./wince* were merged, so perlce build process comes in *./win32* directory.

AUTHORS

Rainer Keuchel <coyxc@rainer-keuchel.de>

provided initial port of Perl, which appears to be most essential work, as it was a breakthrough on having Perl ported at all. Many thanks and obligations to Rainer!

Vadim Konovalov

made further support of WinCE port.

Daniel Dragan

updated the build process during the 5.19 development track.