Rocky Enterprise Linux 9.2 Manual Pages on command 'perlqnx.1'

$ man perlqnx.1

NAME

perlqnx - Perl version 5 on QNX

DESCRIPTION

As of perl5.7.2 all tests pass under:

QNX 4.24G

Watcom 10.6 with Beta/970211.wcc.update.tar.F

socket3r.lib Nov21 1996.

As of perl5.8.1 there is at least one test still failing.

Some tests may complain under known circumstances.

See below and hints/qnx.sh for more information.

Under QNX 6.2.0 there are still a few tests which fail. See below and hints/qnx.sh for more information.

Required Software for Compiling Perl on QNX4

As with many unix ports, this one depends on a few "standard" unix utilities which are not necessarily standard for QNX4.

/bin/sh

This is used heavily by Configure and then by perl itself. QNX4's version is fine, but Configure will choke on the 16-bit version, so if you are running QNX 4.22, link /bin/sh to /bin32/ksh

ar This is the standard unix library builder. We use wlib. With Watcom 10.6, when wlib is linked as "ar", it behaves like ar and all is fine. Under 9.5, a cover is required. One is included in ../qnx nm This is used (optionally) by configure to list the contents of libraries. I will generate a cover function on the fly in the UU directory.

cpp Configure and perl need a way to invoke a C preprocessor. I have created a simple cover for cc which does the right thing. Without this, Configure will create its own wrapper which works, but it doesn't handle some of the command line arguments that perl will throw at it.

make

You really need GNU make to compile this. GNU make ships by default with QNX 4.23, but you can get it from quics for earlier versions.

Outstanding Issues with Perl on QNX4

There is no support for dynamically linked libraries in QNX4.

If you wish to compile with the Socket extension, you need to have the TCP/IP toolkit, and you need to make sure that -lsocket locates the correct copy of socket3r.lib. Beware that the Watcom compiler ships with a stub version of socket3r.lib which has very little functionality. Also beware the order in which wlink searches directories for libraries. You may have /usr/lib/socket3r.lib pointing to the correct library, but wlink may pick up /usr/watcom/10.6/usr/lib/socket3r.lib instead. Make sure they both point to the correct library, that is, /usr/tcptk/current/usr/lib/socket3r.lib.

The following tests may report errors under QNX4:

dist/Cwd/Cwd.t will complain if `pwd` and cwd don't give the same results. cwd calls `fullpath -t`, so if you cd `fullpath -t` before running the test, it will pass.

lib/File/Find/taint.t will complain if '.' is in your PATH. The PATH test is triggered because cwd calls `fullpath -t`.

ext/IO/lib/IO/t/io_sock.t: Subtests 14 and 22 are skipped due to the fact that the functionality to read back the non-blocking status of a socket is not implemented in QNX's TCP/IP. This has been reported to QNX and it may work with later versions of TCP/IP.

t/io/tell.t: Subtest 27 is failing. We are still investigating.

QNX auxiliary files

The files in the "qnx" directory are:

qnx/ar

A script that emulates the standard unix archive (aka library) utility. Under Watcom 10.6, ar is linked to wlib and provides the expected interface. With Watcom 9.5, a cover function is required. This one is fairly crude but has proved adequate for compiling perl.

qnx/cpp

A script that provides C preprocessing functionality. Configure can generate a similar cover, but it doesn't handle all the command-line options that perl throws at it. This might be reasonably placed in /usr/local/bin.

Outstanding issues with perl under QNX6

The following tests are still failing for Perl 5.8.1 under QNX 6.2.0:

```
  op/sprintf.........................FAILED at test 91
  lib/Benchmark......................FAILED at test 26
```

This is due to a bug in the C library's printf routine. printf("'%e'", 0.) produces '0.000000e+0', but ANSI requires '0.000000e+00'. QNX has acknowledged the bug.

Cross-compilation

Perl supports cross-compiling to QNX NTO through the Native Development Kit (NDK) for the Blackberry 10. This means that you can cross-compile for both ARM and x86 versions of the platform.

Setting up a cross-compilation environment

You can download the NDK from <http://developer.blackberry.com/native/downloads/>.

See

<http://developer.blackberry.com/native/documentation/cascades/getting_started/setting_up.html>

for instructions to set up your device prior to attempting anything else.

Once you've installed the NDK and set up your device, all that's left to do is setting up the device and the cross-compilation environment. Blackberry provides a script, "bbndk-env.sh" (occasionally named something like "bbndk-env_10_1_0_4828.sh") which can be used to do this. However, there's a bit of a snag that we have to work through: The script modifies PATH so that 'gcc' or 'ar' point to their cross-compilation equivalents, which screws over the build process.

So instead you'll want to do something like this:

```
    $ orig_path=$PATH
    $ source $location_of_bbndk/bbndk-env*.sh
    $ export PATH="$orig_path:$PATH"
```

Besides putting the cross-compiler and the rest of the toolchain in your PATH, this will also provide the QNX_TARGET variable, which we will pass to Configure through -Dsysroot.

Preparing the target system

It's quite possible that the target system doesn't have a readily available /tmp, so it's generally safer to do something like this:

```
 $ ssh $TARGETUSER@$TARGETHOST 'rm -rf perl; mkdir perl; mkdir perl/tmp'
 $ export TARGETDIR=`ssh $TARGETUSER@$TARGETHOST pwd`/perl
 $ export TARGETENV="export TMPDIR=$TARGETDIR/tmp; "
```

Later on, we'll pass this to Configure through -Dtargetenv

Calling Configure

If you are targetting an ARM device -- which currently includes the vast majority of phones and tablets -- you'll want to pass -Dcc=arm-unknown-nto-qnx8.0.0eabi-gcc to Configure. Alternatively, if you are targetting an x86 device, or using the simulator provided with the NDK, you should specify -Dcc=ntox86-gcc instead.

A sample Configure invocation looks something like this:

```
    ./Configure -des -Dusecrosscompile \
        -Dsysroot=$QNX_TARGET          \
        -Dtargetdir=$TARGETDIR         \
        -Dtargetenv="$TARGETENV"       \
        -Dcc=ntox86-gcc                \
        -Dtarghost=... # Usual cross-compilation options
```

AUTHOR

Norton T. Allen (allen@huarp.harvard.edu)