NAME

README.cygwin - Perl for Cygwin

SYNOPSIS

This document will help you configure, make, test and install Perl on Cygwin. This document also describes features of Cygwin that will affect how Perl behaves at runtime.

NOTE: There are pre-built Perl packages available for Cygwin and a version of Perl is provided in the normal Cygwin install. If you do not need to customize the configuration, consider using one of those packages.

PREREQUISITES FOR COMPILING PERL ON CYGWIN

Cygwin = GNU+Cygnus+Windows (Don't leave UNIX without it)

The Cygwin tools are ports of the popular GNU development tools for Win32 platforms. They run thanks to the Cygwin library which provides the UNIX system calls and environment these programs expect. More information about this project can be found at:

```
http://www.cygwin.com/
```

A recent net or commercial release of Cygwin is required.

At the time this document was last updated, Cygwin 1.5.2 was current.

Cygwin Configuration

While building Perl some changes may be necessary to your Cygwin setup so that Perl builds cleanly. These changes are **not** required for normal Perl usage.

NOTE: The binaries that are built will run on all Win32 versions. They do not depend on your host system (Win9x/WinME, WinNT/Win2K) or your Cygwin configuration (*ntea*, *ntsec*, binary/text mounts). The only dependencies come from hard-coded pathnames like `/usr/local`. However, your host system and Cygwin configuration will affect Perl's runtime behavior (see *TEST*).

* `PATH`

  Set the `PATH` environment variable so that Configure finds the Cygwin versions of programs. Any Windows directories should be removed or moved to the end of your `PATH`.

* *nroff*

  If you do not have *nroff* (which is part of the *groff* package), Configure will **not** prompt you to install *man* pages.

* Permissions

  On WinNT with either the *ntea* or *ntsec* `CYGWIN` settings, directory and file permissions may not be set correctly. Since the build process creates directories and files, to be safe you may want to run a `chmod -R +w *` on the entire Perl source tree.

  Also, it is a well known WinNT "feature" that files created by a login that is a member of the *Administrators* group will be owned by the *Administrators* group. Depending on your umask, you may find that you can not write to files that you just created (because you are no longer the owner). When using the *ntsec* `CYGWIN` setting, this is not an issue because it "corrects" the ownership to what you would expect on a UNIX system.

CONFIGURE PERL ON CYGWIN

The default options gathered by Configure with the assistance of *hints/cygwin.sh* will build a Perl that supports dynamic loading (which requires a shared *libperl.dll*).

This will run Configure and keep a record:

```
./Configure 2>&1 | tee log.configure
```

If you are willing to accept all the defaults run Configure with **-de**. However, several useful customizations are available.

Stripping Perl Binaries on Cygwin

It is possible to strip the EXEs and DLLs created by the build process. The resulting binaries will be significantly smaller. If you want the binaries to be stripped, you can either add a **-s** option when Configure prompts you,

```
Any additional ld flags (NOT including libraries)? [none] -s
Any special flags to pass to gcc to use dynamic linking? [none] -s
Any special flags to pass to ld2 to create a dynamically loaded library?
[none] -s
```

or you can edit *hints/cygwin.sh* and uncomment the relevant variables near the end of the file.

Optional Libraries for Perl on Cygwin

Several Perl functions and modules depend on the existence of some optional libraries. Configure will find them if they are installed in one of the directories listed as being used for library searches. Pre-built packages for most of these are available from the Cygwin installer.

* `-lcrypt`

  The crypt package distributed with Cygwin is a Linux compatible 56-bit DES crypt port by Corinna Vinschen.

  Alternatively, the crypt libraries in GNU libc have been ported to Cygwin.

  The DES based Ultra Fast Crypt port was done by Alexey Truhan:

  ```
  ftp://ftp.uni-erlangen.de/pub/pc/gnuwin32/cygwin/porters/Okhapkin_Ser
  gey/cw32crypt-dist-0.tgz
  ```

  NOTE: There are various export restrictions on DES implementations, see the glibc README for more details.

  The MD5 port was done by Andy Piper:

  ```
  ftp://ftp.uni-erlangen.de/pub/pc/gnuwin32/cygwin/porters/Okhapkin_Ser
  gey/libcrypt.tgz
  ```

* `-lgdbm` (`use GDBM_File`)

  GDBM is available for Cygwin.

  NOTE: The GDBM library only works on NTFS partitions.

* `-ldb` (`use DB_File`)

  BerkeleyDB is available for Cygwin.

  NOTE: The BerkeleyDB library only completely works on NTFS partitions.

* `-lcygipc` (`use IPC::SysV`)

  A port of SysV IPC is available for Cygwin.

  NOTE: This has **not** been extensively tested. In particular, `d_semctl_semun` is undefined because it fails a Configure test and on Win9x the *shm*()* functions seem to hang. It also creates a compile time dependency because *perl.h* includes <*sys/ipc.h*> and <*sys/sem.h*> (which will be required in the future when compiling CPAN modules). CURRENTLY NOT SUPPORTED!

* `-lutil`

Included with the standard Cygwin netrelease is the inetutils package which includes libutil.a.

Configure-time Options for Perl on Cygwin

The *INSTALL* document describes several Configure-time options. Some of these will work with Cygwin, others are not yet possible. Also, some of these are experimental. You can either select an option when Configure prompts you or you can define (undefine) symbols on the command line.

* `-Uusedl`

  Undefining this symbol forces Perl to be compiled statically.

* `-Uusemymalloc`

  By default Perl uses the `malloc()` included with the Perl source. If you want to force Perl to build with the system `malloc()` undefine this symbol.

* `-Useperlio`

  Undefining this symbol disables the PerlIO abstraction. PerlIO is now the default; it is not recommended to disable PerlIO.

* `-Dusemultiplicity`

  Multiplicity is required when embedding Perl in a C program and using more than one interpreter instance. This works with the Cygwin port.

* `-Duse64bitint`

  By default Perl uses 32 bit integers. If you want to use larger 64 bit integers, define this symbol.

* `-Duselongdouble`

  gcc supports long doubles (12 bytes). However, several additional long double math functions are necessary to use them within Perl (*{atan2, cos, exp, floor, fmod, frexp, isnan, log, modf, pow, sin, sqrt}l, strtold*). These are **not** yet available with Cygwin.

* `-Dusethreads`

  POSIX threads are implemented in Cygwin, define this symbol if you want a threaded perl.

* `-Duselargefiles`

  Cygwin uses 64-bit integers for internal size and position calculations, this will be correctly detected and defined by Configure.

* `-Dmksymlinks`

  Use this to build perl outside of the source tree. This works with Cygwin. Details can be found in the *INSTALL* document. This is the recommended way to build perl from sources.

Suspicious Warnings on Cygwin

You may see some messages during Configure that seem suspicious.

* *dlsym()*

  ld2 is needed to build dynamic libraries, but it does not exist when `dlsym()` checking occurs (it is not created until `make` runs). You will see the following message:

```
Checking whether your C<dlsym()> needs a leading underscore ...
ld2: not found
I can't compile and run the test program.
I'm guessing that dlsym doesn't need a leading underscore.
```

  Since the guess is correct, this is not a problem.

* Win9x and `d_eofnblk`

Win9x does not correctly report EOF with a non-blocking read on a closed pipe. You will see the following messages:

```
  But it also returns -1 to signal EOF, so be careful!
  WARNING: you can't distinguish between EOF and no data!

  *** WHOA THERE!!! ***
      The recommended value for $d_eofnblk on this machine was
"define"!
      Keep the recommended value? [y]
```

At least for consistency with WinNT, you should keep the recommended value.

* Compiler/Preprocessor defines

The following error occurs because of the Cygwin `#define` of `_LONG_DOUBLE`:

```
  Guessing which symbols your C compiler and preprocessor define...
  try.c:<line#>: missing binary operator
```

This failure does not seem to cause any problems. With older gcc versions, "parse error" is reported instead of "missing binary operator".

MAKE ON CYGWIN

Simply run *make* and wait:

```
  make 2>&1 | tee log.make
```

Errors on Cygwin

Errors like these are normal:

```
  ...
  make: [extra.pods] Error 1 (ignored)
  ...
  make: [extras.make] Error 1 (ignored)
```

ld2 on Cygwin

During `make`, *ld2* will be created and installed in your $installbin directory (where you said to put public executables). It does not wait until the `make install` process to install the *ld2* script, this is because the remainder of the `make` refers to *ld2* without fully specifying its path and does this from multiple subdirectories. The assumption is that $installbin is in your current `PATH`. If this is not the case `make` will fail at some point. If this happens, just manually copy *ld2* from the source directory to somewhere in your `PATH`.

TEST ON CYGWIN

There are two steps to running the test suite:

```
  make test 2>&1 | tee log.make-test

  cd t;./perl harness 2>&1 | tee ../log.harness
```

The same tests are run both times, but more information is provided when running as `./perl harness`.

Test results vary depending on your host system and your Cygwin configuration. If a test can pass in some Cygwin setup, it is always attempted and explainable test failures are documented. It is possible for Perl to pass all the tests, but it is more likely that some tests will fail for one of the reasons listed below.

File Permissions on Cygwin

UNIX file permissions are based on sets of mode bits for {read,write,execute} for each {user,group,other}. By default Cygwin only tracks the Win32 read-only attribute represented as the UNIX file user write bit (files are always readable, files are executable if they have a *.{com,bat,exe}* extension or begin with `#!`, directories are always readable and executable). On WinNT with the *ntea* `CYGWIN` setting, the additional mode bits are stored as extended file attributes. On WinNT with the *ntsec* `CYGWIN` setting, permissions use the standard WinNT security descriptors and access control lists. Without one of these options, these tests will fail (listing not updated yet):

```
Failed Test           List of failed
-----------------------------------
io/fs.t               5, 7, 9-10
lib/anydbm.t          2
lib/db-btree.t        20
lib/db-hash.t         16
lib/db-recno.t        18
lib/gdbm.t            2
lib/ndbm.t            2
lib/odbm.t            2
lib/sdbm.t            2
op/stat.t             9, 20 (.tmp not an executable extension)
```

NDBM_File and ODBM_File do not work on FAT filesystems

Do not use NDBM_File or ODBM_File on FAT filesystem. They can be built on a FAT filesystem, but many tests will fail:

```
../ext/NDBM_File/ndbm.t       13  3328    71   59  83.10%  1-2 4 16-71
../ext/ODBM_File/odbm.t      255 65280    ??   ??       %  ??
../lib/AnyDBM_File.t           2   512    12    2  16.67%  1 4
../lib/Memoize/t/errors.t      0   139    11    5  45.45%  7-11
../lib/Memoize/t/tie_ndbm.t   13  3328     4    4 100.00%  1-4
run/fresh_perl.t                       97    1   1.03%  91
```

If you intend to run only on FAT (or if using AnyDBM_File on FAT), run Configure with the -Ui_ndbm and -Ui_dbm options to prevent NDBM_File and ODBM_File being built.

With NTFS (and CYGWIN=ntsec), there should be no problems even if perl was built on FAT.

fork() failures in io_* tests

A `fork()` failure may result in the following tests failing:

```
ext/IO/lib/IO/t/io_multihomed.t
ext/IO/lib/IO/t/io_sock.t
ext/IO/lib/IO/t/io_unix.t
```

See comment on fork in *Miscellaneous* below.

Specific features of the Cygwin port

Script Portability on Cygwin

Cygwin does an outstanding job of providing UNIX-like semantics on top of Win32 systems. However, in addition to the items noted above, there are some differences that you should know about. This is a very brief guide to portability, more information can be found in the Cygwin documentation.

* Pathnames

Cygwin pathnames can be separated by forward (/) or backward (\\) slashes. They may also

begin with drive letters (*C:*) or Universal Naming Codes (*//UNC*). DOS device names (*aux*, *con* , *prn*, *com**, *lpt?*, *nul*) are invalid as base filenames. However, they can be used in extensions (e.g., *hello.aux*). Names may contain all printable characters except these:

```
: * ? " < > |
```

File names are case insensitive, but case preserving. A pathname that contains a backslash or drive letter is a Win32 pathname (and not subject to the translations applied to POSIX style pathnames).

* Text/Binary

When a file is opened it is in either text or binary mode. In text mode a file is subject to CR/LF/Ctrl-Z translations. With Cygwin, the default mode for an `open()` is determined by the mode of the mount that underlies the file. Perl provides a `binmode()` function to set binary mode on files that otherwise would be treated as text. `sysopen()` with the `O_TEXT` flag sets text mode on files that otherwise would be treated as binary:

```
sysopen(FOO, "bar", O_WRONLY|O_CREAT|O_TEXT)
```

`lseek()`, `tell()` and `sysseek()` only work with files opened in binary mode.

The text/binary issue is covered at length in the Cygwin documentation.

* PerlIO

PerlIO overrides the default Cygwin Text/Binary behaviour. A file will always treated as binary, regardless which mode of the mount it lives on, just like it is in UNIX. So CR/LF translation needs to be requested in either the `open()` call like this:

```
open(FH, ">:crlf", "out.txt");
```

which will do conversion from LF to CR/LF on the output, or in the environment settings (add this to your .bashrc):

```
export PERLIO=crlf
```

which will pull in the crlf PerlIO layer which does LF -> CRLF conversion on every output generated by perl.

* *.exe*

The Cygwin `stat()`, `lstat()` and `readlink()` functions make the *.exe* extension transparent by looking for *foo.exe* when you ask for *foo* (unless a *foo* also exists). Cygwin does not require a *.exe* extension, but *gcc* adds it automatically when building a program. However, when accessing an executable as a normal file (e.g., *cp* in a makefile) the *.exe* is not transparent. The *install* included with Cygwin automatically appends a *.exe* when necessary.

* cygwin vs. windows process ids

Cygwin processes have their own pid, which is different from the underlying windows pid. Most posix compliant Proc functions expect the cygwin pid, but several Win32::Process functions expect the winpid. E.g. $$ is the cygwin pid of */usr/bin/perl*, which is not the winpid. Use `Cygwin::winpid_to_pid()` and `Cygwin::winpid_to_pid()` to translate between them.

* `chown()`

On WinNT `chown()` can change a file's user and group IDs. On Win9x `chown()` is a no-op, although this is appropriate since there is no security model.

* Miscellaneous

File locking using the `F_GETLK` command to `fcntl()` is a stub that returns `ENOSYS`.

Win9x can not `rename()` an open file (although WinNT can).

The Cygwin `chroot()` implementation has holes (it can not restrict file access by native Win32 programs).

Inplace editing `perl -i` of files doesn't work without doing a backup of the file being edited `perl -i.bak` because of windowish restrictions, therefore Perl adds the suffix `.bak` automatically if you use `perl -i` without specifying a backup extension.

Using `fork()` after loading multiple dlls may fail with an internal cygwin error like the following:

```
  C:\CYGWIN\BIN\PERL.EXE: *** couldn't allocate memory
0x10000(4128768) for
'C:\CYGWIN\LIB\PERL5\5.6.1\CYGWIN-MULTI\AUTO\SOCKET\SOCKET.DLL'
alignment, Win32 error 8

    200 [main] perl 377147 sync_with_child: child -395691(0xB8) died
before initialization with status code 0x1
   1370 [main] perl 377147 sync_with_child: *** child state child
loading dlls
```

Use the rebase utility to resolve the conflicting dll addresses. The rebase package is included in the Cygwin netrelease. Use setup.exe from *http://www.cygwin.com/setup.exe* to install it and run rebaseall.

Prebuilt methods:

`Cwd::cwd`

Returns current working directory.

`Cygwin::pid_to_winpid`

Translates a cygwin pid to the corresponding Windows pid (which may or may not be the same).

`Cygwin::winpid_to_pid`

Translates a Windows pid to the corresponding cygwin pid (if any).

INSTALL PERL ON CYGWIN

This will install Perl, including *man* pages.

```
  make install 2>&1 | tee log.make-install
```

NOTE: If STDERR is redirected `make install` will **not** prompt you to install *perl* into */usr/bin*.

You may need to be *Administrator* to run `make install`. If you are not, you must have write access to the directories in question.

Information on installing the Perl documentation in HTML format can be found in the *INSTALL* document.

MANIFEST ON CYGWIN

These are the files in the Perl release that contain references to Cygwin. These very brief notes attempt to explain the reason for all conditional code. Hopefully, keeping this up to date will allow the Cygwin port to be kept as clean as possible (listing not updated yet).

Documentation

```
INSTALL README.cygwin README.win32 MANIFEST
Changes Changes5.005 Changes5.004 Changes5.6
pod/perl.pod pod/perlport.pod pod/perlfaq3.pod
```

```
        pod/perldelta.pod pod/perl5004delta.pod pod/perl56delta.pod
        pod/perlhist.pod pod/perlmodlib.pod perl/buildtoc pod/perltoc.pod
```

Build, Configure, Make, Install

```
        cygwin/Makefile.SHs
        cygwin/ld2.in
        cygwin/perlld.in
        ext/IPC/SysV/hints/cygwin.pl
        ext/NDBM_File/hints/cygwin.pl
        ext/ODBM_File/hints/cygwin.pl
        hints/cygwin.sh
        Configure             - help finding hints from uname,
                                shared libperl required for dynamic loading
        Makefile.SH           - linklibperl
        Porting/patchls       - cygwin in port list
        installman            - man pages with :: translated to .
        installperl           - install dll/ld2/perlld, install to pods
        makedepend.SH         - uwinfix
```

Tests

```
        t/io/tell.t           - binmode
        t/lib/b.t             - ignore Cwd from os_extras
        t/lib/glob-basic.t    - Win32 directory list access differs from
      read mode
        t/op/magic.t          - $^X/symlink WORKAROUND, s/.exe//
        t/op/stat.t           - no /dev, skip Win32 ftCreationTime quirk
                                (cache manager sometimes preserves ctime of
       file
                                previously created and deleted), no -u
      (setuid)
        t/lib/cygwin.t - builtin cygwin function tests
```

Compiled Perl Source

```
        EXTERN.h              - __declspec(dllimport)
        XSUB.h                - __declspec(dllexport)
        cygwin/cygwin.c       - os_extras (getcwd, spawn,
      Cygwin::winpid_to_pid,
                                Cygwin::pid_to_winpid)
        perl.c                - os_extras
        perl.h                - binmode
        doio.c                - win9x can not rename a file when it is open
        pp_sys.c              - do not define h_errno, pp_system with spawn
        util.c                - use setenv
```

Compiled Module Source

```
        ext/POSIX/POSIX.xs    - tzname defined externally
        ext/SDBM_File/sdbm/pair.c
                              - EXTCONST needs to be redefined from
      EXTERN.h
        ext/SDBM_File/sdbm/sdbm.c
                              - binary open
```

Perl Modules/Scripts

```
  lib/Cwd.pm            - hook to internal Cwd::cwd
  lib/ExtUtils/MakeMaker.pm
                        - require MM_Cygwin.pm
  lib/ExtUtils/MM_Cygwin.pm
                        - canonpath, cflags, manifypods, perl_archive
  lib/File/Find.pm      - on remote drives stat() always sets
st_nlink to 1
  lib/File/Spec/Unix.pm - preserve //unc
  lib/File/Temp.pm      - no directory sticky bit
  lib/perl5db.pl        - use stdin not /dev/tty
  utils/perldoc.PL      - version comment
```

BUGS ON CYGWIN

Support for swapping real and effective user and group IDs is incomplete. On WinNT Cygwin provides `setuid()`, `seteuid()`, `setgid()` and `setegid()`. However, additional Cygwin calls for manipulating WinNT access tokens and security contexts are required.

AUTHORS

Charles Wilson <cwilson@ece.gatech.edu>, Eric Fifer <egf7@columbia.edu>, alexander smishlajev <als@turnhere.com>, Steven Morlock <newspost@morlock.net>, Sebastien Barre <Sebastien.Barre@utc.fr>, Teun Burgers <burgers@ecn.nl>, Gerrit P. Haase <gp@familiehaase.de>.

HISTORY

Last updated: 2005-02-11