NAME

perlos390 - building and installing Perl for OS/390 and z/OS

SYNOPSIS

This document will help you Configure, build, test and install Perl on OS/390 (aka z/OS) Unix System Services.

This document needs to be updated, but we don't know what it should say. Please submit comments to <https://github.com/Perl/perl5/issues>.

DESCRIPTION

This is a fully ported Perl for OS/390 Version 2 Release 3, 5, 6, 7, 8, and 9. It may work on other versions or releases, but those are the ones we have tested it on.

You may need to carry out some system configuration tasks before running the Configure script for Perl.

Tools

The z/OS Unix Tools and Toys list may prove helpful and contains links to ports of much of the software helpful for building Perl.
<http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html>

Unpacking Perl distribution on OS/390

If using ftp remember to transfer the distribution in binary format.

Gunzip/gzip for OS/390 is discussed at:

http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty1.html

to extract an ASCII tar archive on OS/390, try this:

```
pax -o to=IBM-1047,from=ISO8859-1 -r < latest.tar
```

or

```
zcat latest.tar.Z | pax -o to=IBM-1047,from=ISO8859-1 -r
```

If you get lots of errors of the form

```
tar: FSUM7171 ...: cannot set uid/gid: EDC5139I Operation not permitted
```

you did not read the above and tried to use tar instead of pax, you'll first have to remove the (now corrupt) perl directory

```
rm -rf perl-...
```

and then use pax.

Setup and utilities for Perl on OS/390

Be sure that your yacc installation is in place including any necessary parser template files. If you have not already done so then be sure to:

```
cp /samples/yyparse.c /etc
```

This may also be a good time to ensure that your /etc/protocol file and either your /etc/resolv.conf or /etc/hosts files are in place. The IBM document that described such USS system setup issues was SC28-1890-07 "OS/390 UNIX System Services Planning", in particular Chapter 6 on customizing the OE shell.

GNU make for OS/390, which is recommended for the build of perl (as well as building CPAN modules and extensions), is available from the "Tools".

Some people have reported encountering "Out of memory!" errors while trying to build Perl using GNU make binaries. If you encounter such trouble then try to download the source code kit and build GNU make from source to eliminate any such trouble. You might also find GNU make (as well as Perl and Apache) in the red-piece/book "Open Source Software for OS/390 UNIX", SG24-5944-00 from IBM.

If instead of the recommended GNU make you would like to use the system supplied make program then be sure to install the default rules file properly via the shell command:

```
cp /samples/startup.mk /etc
```

and be sure to also set the environment variable _C89_CCMODE=1 (exporting _C89_CCMODE=1 is also a good idea for users of GNU make).

You might also want to have GNU groff for OS/390 installed before running the "make install" step for Perl.

There is a syntax error in the /usr/include/sys/socket.h header file that IBM supplies with USS V2R7, V2R8, and possibly V2R9. The problem with the header file is that near the definition of the

SO_REUSEPORT constant there is a spurious extra '/' character outside of a comment like so:

```
#define SO_REUSEPORT    0x0200    /* allow local address & port
                                  reuse */                 /
```

You could edit that header yourself to remove that last '/', or you might note that Language Environment (LE) APAR PQ39997 describes the problem and PTF's UQ46272 and UQ46271 are the (R8 at least) fixes and apply them. If left unattended that syntax error will turn up as an inability for Perl to build its "Socket" extension.

For successful testing you may need to turn on the sticky bit for your world readable /tmp directory if you have not already done so (see man chmod).

Configure Perl on OS/390

Once you have unpacked the distribution, run "sh Configure" (see INSTALL for a full discussion of the Configure options). There is a "hints" file for os390 that specifies the correct values for most things. Some things to watch out for include:

Shell

A message of the form:

```
(I see you are using the Korn shell.  Some ksh's blow up on Configure,
mainly on older exotic systems.  If yours does, try the Bourne shell instead.)
```

is nothing to worry about at all.

Samples

Some of the parser default template files in /samples are needed in /etc. In particular be sure that you at least copy /samples/yyparse.c to /etc before running Perl's Configure. This step ensures successful extraction of EBCDIC versions of parser files such as perly.c and perly.h. This has to be done before running Configure the first time. If you failed to do so then the easiest way to re-Configure Perl is to delete your misconfigured build root and re-extract the source from the tar ball. Then you must ensure that /etc/yyparse.c is properly in place before attempting to re-run Configure.

Dynamic loading

Dynamic loading is required if you want to use XS modules from CPAN (like DBI (and DBD's), JSON::XS, and Text::CSV_XS) or update CORE modules from CPAN with newer versions (like

Encode) without rebuilding all of the perl binary.

This port will support dynamic loading, but it is not selected by default. If you would like to experiment with dynamic loading then be sure to specify -Dusedl in the arguments to the Configure script. See the comments in hints/os390.sh for more information on dynamic loading. If you build with dynamic loading then you will need to add the $archlibexp/CORE directory to your LIBPATH environment variable in order for perl to work. See the config.sh file for the value of $archlibexp. If in trying to use Perl you see an error message similar to:

```
 CEE3501S The module libperl.dll was not found.
   From entry point __dllstaticinit at compile unit offset +00000194
   at
```

then your LIBPATH does not have the location of libperl.x and either libperl.dll or libperl.so in it. Add that directory to your LIBPATH and proceed.

In hints/os390.sh, selecting -Dusedl will default to *also* select -Duseshrplib. Having a shared plib not only requires LIBPATH to be set to the correct location of libperl.so but also makes it close to impossible to run more than one different perl that was built this way at the same time.

All objects that are involved in -Dusedl builds should be compiled for this, probably by adding to all ccflags

```
 -qexportall -qxplink -qdll -Wc,XPLINK,dll,EXPORTALL -Wl,XPLINK,dll
```

Optimizing

Do not turn on the compiler optimization flag "-O". There is a bug in either the optimizer or perl that causes perl to not work correctly when the optimizer is on.

Config files

Some of the configuration files in /etc used by the networking APIs are either missing or have the wrong names. In particular, make sure that there's either an /etc/resolv.conf or an /etc/hosts, so that **gethostbyname()** works, and make sure that the file /etc/proto has been renamed to /etc/protocol (NOT /etc/protocols, as used by other Unix systems). You may have to look for things like HOSTNAME and DOMAINORIGIN in the "//'SYS1.TCPPARMS(TCPDATA)'" PDS member in order to properly set up your /etc networking files.

Build, Test, Install Perl on OS/390

Simply put:

```
sh Configure
make
make test
```

if everything looks ok (see the next section for test/IVP diagnosis) then:

```
make install
```

this last step may or may not require UID=0 privileges depending on how you answered the questions that Configure asked and whether or not you have write access to the directories you specified.

Build Anomalies with Perl on OS/390

"Out of memory!" messages during the build of Perl are most often fixed by re building the GNU make utility for OS/390 from a source code kit.

Building debugging-enabled binaries (with -g or -g3) will increase the chance of getting these errors. Prevent -g if possible.

Another memory limiting item to check is your MAXASSIZE parameter in your 'SYS1.PARMLIB(BPXPRMxx)' data set (note too that as of V2R8 address space limits can be set on a per user ID basis in the USS segment of a RACF profile). People have reported successful builds of Perl with MAXASSIZE parameters as small as 503316480 (and it may be possible to build Perl with a MAXASSIZE smaller than that).

Within USS your /etc/profile or $HOME/.profile may limit your ulimit settings. Check that the following command returns reasonable values:

```
ulimit -a
```

To conserve memory you should have your compiler modules loaded into the Link Pack Area (LPA/ELPA) rather than in a link list or step lib.

If the c89 compiler complains of syntax errors during the build of the Socket extension then be sure to fix the syntax error in the system header /usr/include/sys/socket.h.

Testing Anomalies with Perl on OS/390

The "make test" step runs a Perl Verification Procedure, usually before installation. You might encounter STDERR messages even during a successful run of "make test". Here is a guide to some of

the more commonly seen anomalies:

Signals

A message of the form:

```
io/openpid.....CEE5210S The signal SIGHUP was received.
CEE5210S The signal SIGHUP was received.
CEE5210S The signal SIGHUP was received.
ok
```

indicates that the t/io/openpid.t test of Perl has passed but done so with extraneous messages on stderr from CEE.

File::Temp

A message of the form:

```
lib/ftmp-security....File::Temp::_gettemp: Parent directory (/tmp/)
is not safe (sticky bit not set when world writable?) at
lib/ftmp-security.t line 100
File::Temp::_gettemp: Parent directory (/tmp/) is not safe (sticky
bit not set when world writable?) at lib/ftmp-security.t line 100
ok
```

indicates a problem with the permissions on your /tmp directory within the HFS. To correct that problem issue the command:

```
chmod a+t /tmp
```

from an account with write access to the directory entry for /tmp.

Out of Memory!

Recent perl test suite is quite memory hungry. In addition to the comments above on memory limitations it is also worth checking for _CEE_RUNOPTS in your environment. Perl now has (in miniperlmain.c) a C #pragma to set CEE run options, but the environment variable wins.

The C code asks for:

```
#pragma runopts(HEAP(2M,500K,ANYWHERE,KEEP,8K,4K) STACK(,,ANY,) ALL31(ON))
```

The important parts of that are the second argument (the increment) to HEAP, and allowing the stack to be "Above the (16M) line". If the heap increment is too small then when perl (for example loading unicode/Name.pl) tries to create a "big" (400K+) string it cannot fit in a single segment and you get "Out of Memory!" - even if there is still plenty of memory available.

A related issue is use with perl's malloc. Perl's malloc uses "sbrk()" to get memory, and "sbrk()" is limited to the first allocation so in this case something like:

```
HEAP(8M,500K,ANYWHERE,KEEP,8K,4K)
```

is needed to get through the test suite.

Installation Anomalies with Perl on OS/390

The installman script will try to run on OS/390. There will be fewer errors if you have a roff utility installed. You can obtain GNU groff from the Redbook SG24-5944-00 ftp site.

Usage Hints for Perl on OS/390

When using perl on OS/390 please keep in mind that the EBCDIC and ASCII character sets are different. See perlebcdic.pod for more on such character set issues. Perl builtin functions that may behave differently under EBCDIC are also mentioned in the perlport.pod document.

Open Edition (UNIX System Services) from V2R8 onward does support #!/path/to/perl script invocation. There is a PTF available from IBM for V2R7 that will allow shell/kernel support for #!. USS releases prior to V2R7 did not support the #! means of script invocation. If you are running V2R6 or earlier then see:

```
head ‘whence perldoc‘
```

for an example of how to use the "eval exec" trick to ask the shell to have Perl run your scripts on those older releases of Unix System Services.

If you are having trouble with square brackets then consider switching your rlogin or telnet client. Try to avoid older 3270 emulators and ISHELL for working with Perl on USS.

Floating Point Anomalies with Perl on OS/390

There appears to be a bug in the floating point implementation on S/390 systems such that calling **int()** on the product of a number and a small magnitude number is not the same as calling **int()** on the quotient of that number and a large magnitude number. For example, in the following Perl code:

```
my $x = 100000.0;
my $y = int($x * 1e-5) * 1e5; # ’0’
my $z = int($x / 1e+5) * 1e5;  # ’100000’
print "\$y is $y and \$z is $z\n"; # $y is 0 and $z is 100000
```

Although one would expect the quantities $y and $z to be the same and equal to 100000 they will differ and instead will be 0 and 100000 respectively.

The problem can be further examined in a roughly equivalent C program:

```
#include <stdio.h>
#include <math.h>
main()
{
double r1,r2;
double x = 100000.0;
double y = 0.0;
double z = 0.0;
x = 100000.0 * 1e-5;
r1 = modf (x,&y);
x = 100000.0 / 1e+5;
r2 = modf (x,&z);
printf("y is %e and z is %e\n",y*1e5,z*1e5);
/* y is 0.000000e+00 and z is 1.000000e+05 (with c89) */
}
```

Modules and Extensions for Perl on OS/390

Pure Perl (that is non XS) modules may be installed via the usual:

```
perl Makefile.PL
make
make test
make install
```

If you built perl with dynamic loading capability then that would also be the way to build XS based extensions. However, if you built perl with the default static linking you can still build XS based extensions for OS/390 but you will need to follow the instructions in ExtUtils::MakeMaker for building statically linked perl binaries. In the simplest configurations building a static perl + XS extension boils down to:

```
perl Makefile.PL
make
make perl
make test
make install
make -f Makefile.aperl inst_perl MAP_TARGET=perl
```

In most cases people have reported better results with GNU make rather than the system's /bin/make program, whether for plain modules or for XS based extensions.

If the make process encounters trouble with either compilation or linking then try setting the _C89_CCMODE to 1. Assuming sh is your login shell then run:

```
export _C89_CCMODE=1
```

If tcsh is your login shell then use the setenv command.

AUTHORS

David Fiander and Peter Prymmer with thanks to Dennis Longnecker and William Raffloer for valuable reports, LPAR and PTF feedback. Thanks to Mike MacIsaac and Egon Terwedow for SG24-5944-00. Thanks to Ignasi Roca for pointing out the floating point problems. Thanks to John Goodyear for dynamic loading help.

SEE ALSO

INSTALL, perlport, perlebcdic, ExtUtils::MakeMaker.

http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html

http://www.redbooks.ibm.com/redbooks/SG245944.html

http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty1.html#opensrc

http://www.xray.mpe.mpg.de/mailing-lists/perl-mvs/

http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ceea3030/

http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/CBCUG030/

Mailing list for Perl on OS/390

If you are interested in the z/OS (formerly known as OS/390) and POSIX-BC (BS2000) ports of Perl

then see the perl-mvs mailing list. To subscribe, send an empty message to perl-mvs-subscribe@perl.org.

See also:

https://lists.perl.org/list/perl-mvs.html

There are web archives of the mailing list at:

https://www.nntp.perl.org/group/perl.mvs/

HISTORY

This document was originally written by David Fiander for the 5.005 release of Perl.

This document was podified for the 5.005_03 release of Perl 11 March 1999.

Updated 12 November 2000 for the 5.7.1 release of Perl.

Updated 15 January 2001 for the 5.7.1 release of Perl.

Updated 24 January 2001 to mention dynamic loading.

Updated 12 March 2001 to mention //'SYS1.TCPPARMS(TCPDATA)'.

Updated 28 November 2001 for broken URLs.

Updated 03 October 2019 for perl-5.33.3+