NAME

perlbs2000 - building and installing Perl for BS2000.

SYNOPSIS

This document will help you Configure, build, test and install Perl on BS2000 in the POSIX subsystem.

DESCRIPTION

This is a ported perl for the POSIX subsystem in BS2000 VERSION OSD V3.1A or later. It may work on other versions, but we started porting and testing it with 3.1A and are currently using Version V4.0A.

You may need the following GNU programs in order to install perl:

gzip on BS2000

We used version 1.2.4, which could be installed out of the box with one failure during 'make check'.

bison on BS2000

The yacc coming with BS2000 POSIX didn't work for us. So we had to use bison. We had to make a few changes to perl in order to use the pure (reentrant) parser of bison. We used version 1.25, but we had to add a few changes due to EBCDIC. See below for more details concerning yacc.

Unpacking Perl Distribution on BS2000

To extract an ASCII tar archive on BS2000 POSIX you need an ASCII filesystem (we used the mountpoint /usr/local/ascii for this). Now you extract the archive in the ASCII filesystem without I/O-conversion:

cd /usr/local/ascii export IO_CONVERSION=NO gunzip < /usr/local/src/perl.tar.gz | pax -r

You may ignore the error message for the first element of the archive (this doesn't look like a tar archive / skipping to next file...), it's only the directory which will be created automatically anyway.

After extracting the archive you copy the whole directory tree to your EBCDIC filesystem. **This time you use I/O-conversion**:

cd /usr/local/src IO_CONVERSION=YES cp -r /usr/local/ascii/perl5.005_02 ./

Compiling Perl on BS2000

There is a "hints" file for BS2000 called hints.posix-bc (because posix-bc is the OS name given by `uname`) that specifies the correct values for most things. The major problem is (of course) the EBCDIC character set. We have german EBCDIC version.

Because of our problems with the native yacc we used GNU bison to generate a pure (=reentrant) parser for perly.y. So our yacc is really the following script:

-----8<-----/usr/local/bin/yacc-----8<----- #! /usr/bin/sh

Bison as a reentrant yacc:

save parameters: params="" while [[$# -gt 1]]; do params="$params $1" shift done

add flag %pure_parser:

tmpfile=/tmp/bison.$$.y echo %pure_parser > $tmpfile cat $1 >> $tmpfile

call bison:

echo "/usr/local/bin/bison --yacc $params $1\t\t\t(Pure Parser)" /usr/local/bin/bison --yacc $params $tmpfile

cleanup:

rm -f $tmpfile -----8<----------8<-----

We still use the normal yacc for a2p.y though!!! We made a softlink called byacc to distinguish between the two versions:

ln -s /usr/bin/yacc /usr/local/bin/byacc

We build perl using GNU make. We tried the native make once and it worked too.

Testing Perl on BS2000

We still got a few errors during `make test`. Some of them are the result of using bison. Bison prints *parser error* instead of *syntax error*, so we may ignore them. The following list shows our errors, your results may differ:

op/numconvert.....FAILED tests 1409-1440 op/regexp.....FAILED tests 483, 496 op/regexp_noamp.....FAILED tests 483, 496 pragma/overload.....FAILED tests 152-153, 170-171 pragma/warnings.....FAILED tests 14, 82, 129, 155, 192, 205, 207 lib/bigfloat.....FAILED tests 351-352, 355 lib/bigfltpm.....FAILED tests 354-355, 358 lib/complex.....FAILED tests 267, 487 lib/dumper.....FAILED tests 43, 45 Failed 11/231 test scripts, 95.24% okay. 57/10595 subtests failed, 99.46% okay.

Installing Perl on BS2000

We have no nroff on BS2000 POSIX (yet), so we ignored any errors while installing the documentation.

Using Perl in the Posix-Shell of BS2000

BS2000 POSIX doesn't support the shebang notation (`#!/usr/local/bin/perl`), so you have to use the following lines instead:

: # use perl eval 'exec /usr/local/bin/perl -S $0 ${1+"$@"}' if $running_under_some_shell;

Using Perl in "native" BS2000

We don't have much experience with this yet, but try the following:

Copy your Perl executable to a BS2000 LLM using bs2cp:

```
bs2cp /usr/local/bin/perl 'bs2:perl(perl,l)'
```

Now you can start it with the following (SDF) command:

```
/START-PROG FROM-FILE=*MODULE(PERL,PERL),PROG-MODE=*ANY,RUN-MODE=*ADV
```

First you get the BS2000 commandline prompt ('*'). Here you may enter your parameters, e.g. `-e 'print "Hello World!\\n";'` (note the double backslash!) or `-w` and the name of your Perl script. Filenames starting with `/` are searched in the Posix filesystem, others are searched in the BS2000 filesystem. You may even use wildcards if you put a `%` in front of your filename (e.g. `-w checkfiles.pl %*.c`). Read your C/C++ manual for additional possibilities of the commandline prompt (look for PARAMETER-PROMPTING).

Floating point anomalies on BS2000

There appears to be a bug in the floating point implementation on BS2000 POSIX 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.

Using PerlIO and different encodings on ASCII and EBCDIC partitions

Since version 5.8 Perl uses the new PerlIO on BS2000. This enables you using different encodings per IO channel. For example you may use

```
use Encode;
open($f, ">:encoding(ascii)", "test.ascii");
print $f "Hello World!\n";
open($f, ">:encoding(posix-bc)", "test.ebcdic");
print $f "Hello World!\n";
open($f, ">:encoding(latin1)", "test.latin1");
print $f "Hello World!\n";
open($f, ">:encoding(utf8)", "test.utf8");
print $f "Hello World!\n";
```

to get two files containing "Hello World!\n" in ASCII, EBCDIC, ISO Latin-1 (in this example identical to ASCII) respective UTF-EBCDIC (in this example identical to normal EBCDIC). See the documentation of Encode::PerlIO for details.

As the PerlIO layer uses raw IO internally, all this totally ignores the type of your filesystem (ASCII or EBCDIC) and the IO_CONVERSION environment variable. If you want to get the old behavior, that the BS2000 IO functions determine conversion depending on the filesystem PerlIO still is your friend. You use IO_CONVERSION as usual and tell Perl, that it should use the native IO layer:

```
export IO_CONVERSION=YES
export PERLIO=stdio
```

Now your IO would be ASCII on ASCII partitions and EBCDIC on EBCDIC partitions. See the documentation of PerlIO (without `Encode::`!) for further possibilities.

AUTHORS

Thomas Dorner

SEE ALSO

INSTALL, *perlport*.

Mailing list

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:

```
http://lists.perl.org/list/perl-mvs.html
```

There are web archives of the mailing list at:

```
http://www.xray.mpe.mpg.de/mailing-lists/perl-mvs/
http://archive.develooper.com/perl-mvs@perl.org/
```

HISTORY

This document was originally written by Thomas Dorner for the 5.005 release of Perl.

This document was podified for the 5.6 release of perl 11 July 2000.