NAME

README.hpux - Perl version 5 on Hewlett-Packard Unix (HP-UX) systems

DESCRIPTION

This document describes various features of HP's Unix operating system (HP-UX) that will affect how Perl version 5 (hereafter just Perl) is compiled and/or runs.

Using perl as shipped with HP-UX

Application release September 2001, HP-UX 11.00 is the first to ship with Perl. By the time it was perl-5.6.1 in /opt/perl. The first occurrence is on CD 5012-7954 and can be installed using

```
swinstall -s /cdrom perl
```

assuming you have mounted that CD on /cdrom. In this version the following modules were installed:

```
ActivePerl::DocTools-0.04   HTML::Parser-3.19   XML::DOM-1.25
Archive::Tar-0.072          HTML::Tagset-3.03   XML::Parser-2.27
Compress::Zlib-1.08         MIME::Base64-2.11   XML::Simple-1.05
Convert::ASN1-0.10          Net-1.07            XML::XPath-1.09
Digest::MD5-2.11            PPM-2.1.5           XML::XSLT-0.32
File::CounterFile-0.12      SOAP::Lite-0.46     libwww-perl-5.51
Font::AFM-1.18              Storable-1.011      libxml-perl-0.07
HTML-Tree-3.11              URI-1.11            perl-ldap-0.23
```

That build was a portable hppa-1.1 multithread build that supports large files compiled with gcc-2.9-hppa-991112.

If you perform a new installation, then (a newer) Perl will be installed automatically. Preinstalled HP-UX systems now slao have more recent versions of Perl and the updated modules.

The official (threaded) builds from HP, as they are shipped on the Application DVD/CD's are available on http://www.software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?p roductNumber=PERL for both PA-RISC and IPF (Itanium Processor Family). They are built with the HP ANSI-C compiler.

Using perl from HP's porting centre

HP porting centre tries very hard to keep up with customer demand and release updates from the Open Source community. Having precompiled Perl binaries available is obvious.

The HP porting centres are limited in what systems they are allowed to port to and they usually choose the two most recent OS versions available. This means that at the moment of writing, there are only HP-UX 11.11 (pa-risc 2.0) and HP-UX 11.23 (Itanium 2) ports available on the porting centres.

HP has asked the porting centre to move Open Source binaries from /opt to /usr/local, so binaries produced since the start of July 2002 are located in /usr/local.

One of HP porting centres URL's is http://hpux.connect.org.uk/ The port currently available is built with GNU gcc.

Compiling Perl 5 on HP-UX

When compiling Perl, you must use an ANSI C compiler. The C compiler that ships with all HP-UX systems is a K&R compiler that should only be used to build new kernels.

Perl can be compiled with either HP's ANSI C compiler or with gcc. The former is recommended, as not only can it compile Perl with no difficulty, but also can take advantage of features listed later that require the use of HP compiler-specific command-line flags.

If you decide to use gcc, make sure your installation is recent and complete, and be sure to read the Perl INSTALL file for more gcc-specific details.

PA-RISC

HP's HP9000 Unix systems run on HP's own Precision Architecture (PA-RISC) chip. HP-UX used to run on the Motorola MC68000 family of chips, but any machine with this chip in it is quite obsolete and this document will not attempt to address issues for compiling Perl on the Motorola chipset.

The most recent version of PA-RISC at the time of this document's last update is 2.0. HP PA-RISC systems are usually refered to with model description "HP 9000". The last CPU in this series is the PA-8900. Support for PA-RISC architectured machines officially ends as shown in the following table:

PA-RISC End-of-Life Roadmap

HP9000 4-128 cores	Superdome	PA-8700	Spring 2011
		PA-8800/sx1000	Summer 2012
		PA-8900/sx1000	2014
		PA-8900/sx2000	2015
HP9000 2-32 cores	rp7410, rp8400 rp7420, rp8420 rp7440, rp8440	PA-8700	Spring 2011
		PA-8800/sx1000	2012
		PA-8900/sx1000	Autumn 2013
		PA-8900/sx2000	2015
HP9000 1-8 cores	rp44x0	PA-8700	Spring 2011
		PA-8800/rp44x0	2012
		PA-8900/rp44x0	2014
HP9000 1-4 cores	rp34x0	PA-8700	Spring 2011
		PA-8800/rp34x0	2012
		PA-8900/rp34x0	2014

A complete list of models at the time the OS was built is in the file /usr/sam/lib/mo/sched.models. The first column corresponds to the last part of the output of the "model" command. The second column is the PA-RISC version and the third column is the exact chip type used. (Start browsing at the bottom to prevent confusion ;-)

```
# model
9000/800/L1000-44
# grep L1000-44 /usr/sam/lib/mo/sched.models
L1000-44        2.0     PA8500
```

Portability Between PA-RISC Versions

An executable compiled on a PA-RISC 2.0 platform will not execute on a PA-RISC 1.1 platform, even if they are running the same version of HP-UX. If you are building Perl on a PA-RISC 2.0 platform and want that Perl to also run on a PA-RISC 1.1, the compiler flags +DAportable and +DS32 should be used.

It is no longer possible to compile PA-RISC 1.0 executables on either the PA-RISC 1.1 or 2.0 platforms. The command-line flags are accepted, but the resulting executable will not run when transferred to a PA-RISC 1.0 system.

PA-RISC 1.0

The original version of PA-RISC, HP no longer sells any system with this chip.

The following systems contained PA-RISC 1.0 chips:

```
600, 635, 645, 808, 815, 822, 825, 832, 834, 835, 840, 842, 845, 850,
852, 855, 860, 865, 870, 890
```

PA-RISC 1.1

An upgrade to the PA-RISC design, it shipped for many years in many different system.

The following systems contain with PA-RISC 1.1 chips:

```
705, 710, 712, 715, 720, 722, 725, 728, 730, 735, 742, 743, 744, 745,
747, 750, 755, 770, 777, 778, 779, 800, 801, 803, 806, 807, 809, 811,
813, 816, 817, 819, 821, 826, 827, 829, 831, 837, 839, 841, 847, 849,
851, 856, 857, 859, 867, 869, 877, 887, 891, 892, 897, A180, A180C,
B115, B120, B132L, B132L+, B160L, B180L, C100, C110, C115, C120,
C160L, D200, D210, D220, D230, D250, D260, D310, D320, D330, D350,
D360, D410, DX0, DX5, DXO, E25, E35, E45, E55, F10, F20, F30, G30,
G40, G50, G60, G70, H20, H30, H40, H50, H60, H70, I30, I40, I50, I60,
I70, J200, J210, J210XC, K100, K200, K210, K220, K230, K400, K410,
K420, S700i, S715, S744, S760, T500, T520
```

PA-RISC 2.0

The most recent upgrade to the PA-RISC design, it added support for 64-bit integer data.

As of the date of this document's last update, the following systems contain PA-RISC 2.0 chips:

```
700, 780, 781, 782, 783, 785, 802, 804, 810, 820, 861, 871, 879, 889,
893, 895, 896, 898, 899, A400, A500, B1000, B2000, C130, C140, C160,
C180, C180+, C180-XP, C200+, C400+, C3000, C360, C3600, CB260, D270,
D280, D370, D380, D390, D650, J220, J2240, J280, J282, J400, J410,
J5000, J5500XM, J5600, J7000, J7600, K250, K260, K260-EG, K270, K360,
K370, K380, K450, K460, K460-EG, K460-XP, K470, K570, K580, L1000,
L2000, L3000, N4000, R380, R390, SD16000, SD32000, SD64000, T540,
T600, V2000, V2200, V2250, V2500, V2600
```

Just before HP took over Compaq, some systems were renamed. the link that contained the explanation is dead, so here's a short summary:

```
HP 9000 A-Class servers, now renamed HP Server rp2400 series.
HP 9000 L-Class servers, now renamed HP Server rp5400 series.
HP 9000 N-Class servers, now renamed HP Server rp7400.


rp2400, rp2405, rp2430, rp2450, rp2470, rp3410, rp3440, rp4410,
rp4440, rp5400, rp5405, rp5430, rp5450, rp5470, rp7400, rp7405,
rp7410, rp7420, rp7440, rp8400, rp8420, rp8440, Superdome
```

The current naming convention is:

```
aadddd
||||`+- 00 - 99 relative capacity & newness (upgrades, etc.)
|||`--- unique number for each architecture to ensure different
|||     systems do not have the same numbering across
|||     architectures
||`---- 1 - 9 identifies family and/or relative positioning
||
|`----- c = ia32 (cisc)
|       p = pa-risc
|       x = ia-64 (Itanium & Itanium 2)
```

```
|       h = housing
`------ t = tower
        r = rack optimized
        s = super scalable
        b = blade
        sa = appliance
```

Itanium Processor Family (IPF) and HP-UX

HP-UX also runs on the new Itanium processor. This requires the use of a different version of HP-UX (currently 11.23 or 11i v2), and with the exception of a few differences detailed below and in later sections, Perl should compile with no problems.

Although PA-RISC binaries can run on Itanium systems, you should not attempt to use a PA-RISC version of Perl on an Itanium system. This is because shared libraries created on an Itanium system cannot be loaded while running a PA-RISC executable.

HP Itanium 2 systems are usually refered to with model description "HP Integrity".

Itanium, Itanium 2 & Madison 6

HP also ships servers with the 128-bit Itanium processor(s). The cx26x0 is told to have Madison 6. As of the date of this document's last update, the following systems contain Itanium or Itanium 2 chips (this is likely to be out of date):

```
BL60p, BL860c, cx2600, cx2620, rx1600, rx1620, rx2600, rx2600hptc,
rx2620, rx2660, rx3600, rx4610, rx4640, rx5670, rx6600, rx7420,
rx7620, rx7640, rx8420, rx8620, rx8640, rx9610, sx1000, sx2000
```

To see all about your machine, type

```
# model
ia64 hp server rx2600
# /usr/contrib/bin/machinfo
```

Building Dynamic Extensions on HP-UX

HP-UX supports dynamically loadable libraries (shared libraries). Shared libraries end with the suffix .sl. On Itanium systems, they end with the suffix .so.

Shared libraries created on a platform using a particular PA-RISC version are not usable on platforms using an earlier PA-RISC version by default. However, this backwards compatibility may be enabled using the same +DAportable compiler flag (with the same PA-RISC 1.0 caveat mentioned above).

Shared libraries created on an Itanium platform cannot be loaded on a PA-RISC platform. Shared libraries created on a PA-RISC platform can only be loaded on an Itanium platform if it is a PA-RISC executable that is attempting to load the PA-RISC library. A PA-RISC shared library cannot be loaded into an Itanium executable nor vice-versa.

To create a shared library, the following steps must be performed:

1. Compile source modules with +z or +Z flag to create a .o module
   which contains Position-Independent Code (PIC). The linker will
   tell you in the next step if +Z was needed.
   (For gcc, the appropriate flag is -fpic or -fPIC.)

2. Link the shared library using the -b flag. If the code calls
   any functions in other system libraries (e.g., libm), it must
   be included on this line.

(Note that these steps are usually handled automatically by the extension's Makefile).

If these dependent libraries are not listed at shared library creation time, you will get fatal "Unresolved symbol" errors at run time when the library is loaded.

You may create a shared library that refers to another library, which may be either an archive library or a shared library. If this second library is a shared library, this is called a "dependent library". The dependent library's name is recorded in the main shared library, but it is not linked into the shared library. Instead, it is loaded when the main shared library is loaded. This can cause problems if you build an extension on one system and move it to another system where the libraries may not be located in the same place as on the first system.

If the referred library is an archive library, then it is treated as a simple collection of .o modules (all of which must contain PIC). These modules are then linked into the shared library.

Note that it is okay to create a library which contains a dependent library that is already linked into perl.

Some extensions, like DB_File and Compress::Zlib use/require prebuilt libraries for the perl extensions/modules to work. If these libraries are built using the default configuration, it might happen that you run into an error like "invalid loader fixup" during load phase. HP is aware of this problem. Search the HP-UX cxx-dev forums for discussions about the subject. The short answer is that **everything** (all libraries, everything) must be compiled with +z or +Z to be PIC (position independent code). (For gcc, that would be `-fpic` or `-fPIC`). In HP-UX 11.00 or newer the linker error message should tell the name of the offending object file.

A more general approach is to intervene manually, as with an example for the DB_File module, which requires SleepyCat's libdb.sl:

```
# cd .../db-3.2.9/build_unix
# vi Makefile
... add +Z to all cflags to create shared objects
CFLAGS=         -c $(CPPFLAGS) +Z -Ae +O2 +Onolimit \
                -I/usr/local/include -I/usr/include/X11R6
CXXFLAGS=       -c $(CPPFLAGS) +Z -Ae +O2 +Onolimit \
                -I/usr/local/include -I/usr/include/X11R6

# make clean
# make
# mkdir tmp
# cd tmp
# ar x ../libdb.a
# ld -b -o libdb-3.2.sl *.o
# mv libdb-3.2.sl /usr/local/lib
# rm *.o
# cd /usr/local/lib
# rm -f libdb.sl
# ln -s libdb-3.2.sl libdb.sl

# cd .../DB_File-1.76
# make distclean
# perl Makefile.PL
# make
# make test
# make install
```

As of db-4.2.x it is no longer needed to do this by hand. Sleepycat has changed the configuration

process to add +z on HP-UX automatically.

```
# cd .../db-4.2.25/build_unix
# env CFLAGS=+DD64 LDFLAGS=+DD64 ../dist/configure
```

should work to generate 64bit shared libraries for HP-UX 11.00 and 11i.

It is no longer possible to link PA-RISC 1.0 shared libraries (even though the command-line flags are still present).

PA-RISC and Itanium object files are not interchangeable. Although you may be able to use ar to create an archive library of PA-RISC object files on an Itanium system, you cannot link against it using an Itanium link editor.

The HP ANSI C Compiler

When using this compiler to build Perl, you should make sure that the flag -Aa is added to the cpprun and cppstdin variables in the config.sh file (though see the section on 64-bit perl below). If you are using a recent version of the Perl distribution, these flags are set automatically.

Even though HP-UX 10.20 and 11.00 are not actively maintained by HP anymore, updates for the HP ANSI C compiler are still available from time to time, and it might be advisable to see if updates are applicable. At the moment of writing, the latests available patches for 11.00 that should be applied are PHSS_35098, PHSS_35175, PHSS_35100, PHSS_33036, and PHSS_33902). If you have a SUM account, you can use it to search for updates/patches. Enter "ANSI" as keyword.

The GNU C Compiler

When you are going to use the GNU C compiler (gcc), and you don't have gcc yet, you can either build it yourself from the sources (available from e.g. http://www.gnu.ai.mit.edu/software/gcc/releases.html) or fetch a prebuilt binary from the HP porting center. There are two places where gcc prebuilds can be fetched; the first and best (for HP-UX 11 only) is http://h21007.www2.hp.com/dspp/tech/tech_TechSoftwareDetailPage_IDX/1,1703,547,00.html the second is http://hpux.cs.utah.edu/hppd/hpux/Gnu/ where you can also find the GNU binutils package. (Browse through the list, because there are often multiple versions of the same package available).

Above mentioned distributions are depots. H.Merijn Brand has made prebuilt gcc binaries available on http://mirrors.develooper.com/hpux/ and/or http://www.cmve.net/~merijn/ for HP-UX 10.20, HP-UX 11.00, and HP-UX 11.11 (HP-UX 11i) in both 32- and 64-bit versions. These are bzipped tar archives that also include recent GNU binutils and GNU gdb. Read the instructions on that page to rebuild gcc using itself.

On PA-RISC you need a different compiler for 32-bit applications and for 64-bit applications. On PA-RISC, 32-bit objects and 64-bit objects do not mix. Period. There is no different behaviour for HP C-ANSI-C or GNU gcc. So if you require your perl binary to use 64-bit libraries, like Oracle-64bit, you MUST build a 64-bit perl.

Building a 64-bit capable gcc on PA-RISC from source is possible only when you have the HP C-ANSI C compiler or an already working 64-bit binary of gcc available. Best performance for perl is achieved with HP's native compiler.

Using Large Files with Perl on HP-UX

Beginning with HP-UX version 10.20, files larger than 2GB (2^31 bytes) may be created and manipulated. Three separate methods of doing this are available. Of these methods, the best method for Perl is to compile using the -Duselargefiles flag to Configure. This causes Perl to be compiled using structures and functions in which these are 64 bits wide, rather than 32 bits wide. (Note that this will only work with HP's ANSI C compiler. If you want to compile Perl using gcc, you will have to get a version of the compiler that supports 64-bit operations. See above for where to find it.)

There are some drawbacks to this approach. One is that any extension which calls any file-manipulating C function will need to be recompiled (just follow the usual "perl Makefile.PL; make; make test; make install" procedure).

The list of functions that will need to recompiled is: creat, fgetpos, fopen, freopen, fsetpos, fstat, fstatvfs, fstatvfsdev, ftruncate, ftw, lockf, lseek, lstat, mmap, nftw, open, prealloc, stat, statvfs, statvfsdev, tmpfile, truncate, getrlimit, setrlimit

Another drawback is only valid for Perl versions before 5.6.0. This drawback is that the seek and tell functions (both the builtin version and POSIX module version) will not perform correctly.

It is strongly recommended that you use this flag when you run Configure. If you do not do this, but later answer the question about large files when Configure asks you, you may get a configuration that cannot be compiled, or that does not function as expected.

Threaded Perl on HP-UX

It is possible to compile a version of threaded Perl on any version of HP-UX before 10.30, but it is strongly suggested that you be running on HP-UX 11.00 at least.

To compile Perl with threads, add -Dusethreads to the arguments of Configure. Verify that the -D_POSIX_C_SOURCE=199506L compiler flag is automatically added to the list of flags. Also make sure that -lpthread is listed before -lc in the list of libraries to link Perl with. The hints provided for HP-UX during Configure will try very hard to get this right for you.

HP-UX versions before 10.30 require a separate installation of a POSIX threads library package. Two examples are the HP DCE package, available on "HP-UX Hardware Extensions 3.0, Install and Core OS, Release 10.20, April 1999 (B3920-13941)" or the Freely available PTH package, available on H.Merijn's site (http://mirrors.develooper.com/hpux/).

If you are going to use the HP DCE package, the library used for threading is /usr/lib/libcma.sl, but there have been multiple updates of that library over time. Perl will build with the first version, but it will not pass the test suite. Older Oracle versions might be a compelling reason not to update that library, otherwise please find a newer version in one of the following patches: PHSS_19739, PHSS_20608, or PHSS_23672

reformatted output:

```
d3:/usr/lib 106 > what libcma-*.1
libcma-00000.1:
   HP DCE/9000 1.5               Module: libcma.sl (Export)
                                 Date: Apr 29 1996 22:11:24
libcma-19739.1:
   HP DCE/9000 1.5 PHSS_19739-40 Module: libcma.sl (Export)
                                 Date: Sep  4 1999 01:59:07
libcma-20608.1:
   HP DCE/9000 1.5 PHSS_20608    Module: libcma.1 (Export)
                                 Date: Dec  8 1999 18:41:23
libcma-23672.1:
   HP DCE/9000 1.5 PHSS_23672    Module: libcma.1 (Export)
                                 Date: Apr  9 2001 10:01:06
d3:/usr/lib 107 >
```

If you choose for the PTH package, use swinstall to install pth in the default location (/opt/pth), and then make symbolic links to the libraries from /usr/lib

```
# cd /usr/lib
# ln -s /opt/pth/lib/libpth* .
```

For building perl to support Oracle, it needs to be linked with libcl and libpthread. So even if your perl is an unthreaded build, these libraries might be required. See "Oracle on HP-UX" below.

64-bit Perl on HP-UX

Beginning with HP-UX 11.00, programs compiled under HP-UX can take advantage of the LP64 programming environment (LP64 means Longs and Pointers are 64 bits wide), in which scalar variables will be able to hold numbers larger than 2^32 with complete precision. Perl has proven to be consistent and reliable in 64bit mode since 5.8.1 on all HP-UX 11.xx.

As of the date of this document, Perl is fully 64-bit compliant on HP-UX 11.00 and up for both cc- and gcc builds. If you are about to build a 64-bit perl with GNU gcc, please read the gcc section carefully.

Should a user have the need for compiling Perl in the LP64 environment, use the -Duse64bitall flag to Configure. This will force Perl to be compiled in a pure LP64 environment (with the +DD64 flag for HP C-ANSI-C, with no additional options for GNU gcc 64-bit on PA-RISC, and with -mlp64 for GNU gcc on Itanium). If you want to compile Perl using gcc, you will have to get a version of the compiler that supports 64-bit operations.)

You can also use the -Duse64bitint flag to Configure. Although there are some minor differences between compiling Perl with this flag versus the -Duse64bitall flag, they should not be noticeable from a Perl user's perspective. When configuring -Duse64bitint using a 64bit gcc on a pa-risc architecture, -Duse64bitint is silently promoted to -Duse64bitall.

In both cases, it is strongly recommended that you use these flags when you run Configure. If you do not use do this, but later answer the questions about 64-bit numbers when Configure asks you, you may get a configuration that cannot be compiled, or that does not function as expected.

Oracle on HP-UX

Using perl to connect to Oracle databases through DBI and DBD::Oracle has caused a lot of people many headaches. Read README.hpux in the DBD::Oracle for much more information. The reason to mention it here is that Oracle requires a perl built with libcl and libpthread, the latter even when perl is build without threads. Building perl using all defaults, but still enabling to build DBD::Oracle later on can be achieved using

```
Configure -A prepend:libswanted='cl pthread ' ...
```

Do not forget the space before the trailing quote.

Also note that this does not (yet) work with all configurations, it is known to fail with 64-bit versions of GCC.

GDBM and Threads on HP-UX

If you attempt to compile Perl with (POSIX) threads on an 11.X system and also link in the GDBM library, then Perl will immediately core dump when it starts up. The only workaround at this point is to relink the GDBM library under 11.X, then relink it into Perl.

the error might show something like:

Pthread internal error: message: __libc_reinit() failed, file: ../pthreads/pthread.c, line: 1096 Return Pointer is 0xc082bf33 sh: 5345 Quit(coredump)

and Configure will give up.

NFS filesystems and utime(2) on HP-UX

If you are compiling Perl on a remotely-mounted NFS filesystem, the test io/fs.t may fail on test #18. This appears to be a bug in HP-UX and no fix is currently available.

perl -P and // and HP-UX

If HP-UX Perl is compiled with flags that will cause problems if the -P flag of Perl (preprocess Perl code with the C preprocessor before perl sees it) is used. The problem is that `//`, being a C++-style until-end-of-line comment, will disappear along with the remainder of the line. This means that common Perl constructs like

```
s/foo//;
```

will turn into illegal code

```
s/foo
```

The workaround is to use some other quoting separator than `"/"`, like for example `"!"`:

```
s!foo!!;
```

HP-UX Kernel Parameters (maxdsiz) for Compiling Perl

By default, HP-UX comes configured with a maximum data segment size of 64MB. This is too small to correctly compile Perl with the maximum optimization levels. You can increase the size of the maxdsiz kernel parameter through the use of SAM.

When using the GUI version of SAM, click on the Kernel Configuration icon, then the Configurable Parameters icon. Scroll down and select the maxdsiz line. From the Actions menu, select the Modify Configurable Parameter item. Insert the new formula into the Formula/Value box. Then follow the instructions to rebuild your kernel and reboot your system.

In general, a value of 256MB (or "256*1024*1024") is sufficient for Perl to compile at maximum optimization.

nss_delete core dump from op/pwent or op/grent

You may get a bus error core dump from the op/pwent or op/grent tests. If compiled with -g you will see a stack trace much like the following:

```
#0  0xc004216c in  () from /usr/lib/libc.2
#1  0xc00d7550 in __nss_src_state_destr () from /usr/lib/libc.2
#2  0xc00d7768 in __nss_src_state_destr () from /usr/lib/libc.2
#3  0xc00d78a8 in nss_delete () from /usr/lib/libc.2
#4  0xc01126d8 in endpwent () from /usr/lib/libc.2
#5  0xd1950 in Perl_pp_epwent () from ./perl
#6  0x94d3c in Perl_runops_standard () from ./perl
#7  0x23728 in S_run_body () from ./perl
#8  0x23428 in perl_run () from ./perl
#9  0x2005c in main () from ./perl
```

The key here is the `nss_delete` call. One workaround for this bug seems to be to create add to the file */etc/nsswitch.conf* (at least) the following lines

```
group: files
passwd: files
```

Whether you are using NIS does not matter. Amazingly enough, the same bug also affects Solaris.

Miscellaneous

HP-UX 11 Y2K patch "Y2K-1100 B.11.00.B0125 HP-UX Core OS Year 2000 Patch Bundle" has been reported to break the io/fs test #18 which tests whether utime() can change timestamps. The Y2K patch seems to break utime() so that over NFS the timestamps do not get changed (on local

filesystems utime() still works). This has probably been fixed on your system by now.

AUTHOR

H.Merijn Brand <h.m.brand@xs4all.nl> Jeff Okamoto <okamoto@corp.hp.com>

With much assistance regarding shared libraries from Marc Sabatella.

DATE

Version 0.8.0: 2007-09-09