store-extra: Useful Store admin and config applications

Petter Reinholdtsen <pere@td.org.uit.no>

2000-06-27

1 Introduction

The `store-extra` Store application is my collection of useful Store admin and config scripts which is not part of the Store base (`perl-internal`). Most of it used to be part of smaller applications, but I decided to collect it into one application instead.

This is the collection of what used to be `store-pere`, `env-config`, `mime-config`, `emacs-info`, `build-all` and some scripts from NTNUs `emacs` application.

I did not write all these scripts. I have lost track of all the authors, but some of them were written by Espen Skoglund, Bjrn Stabell, Arne H. Juul, and more. If your credit is missing here, please let me know.

2 Compiling new applications and versions

The build-all scripts provides the simplest way to compile new programs in store. It works best with source using GNU autoconf and Perl modules. It has limited support for X imake source and normal makefiles. If the application fails to compile using build-all, the original method described in the original Store documentation must be used instead.

When everything works as it should, this sequence would give a compiled version of a new application:

1. Make new application subdirectory in `/store/store/<master>/`.
2. Fetch source and update `registration` with source location. FTP sites are better then HTTP sites. Replace version number with `$version` to allow automatic detection of new versions.

   ```
   echo 'Source: ftp://ftp.gnu.org/pub/gnu/gcc-${version}.tar.gz' >> registration
   ```

3. Unpack source in application subdirectory. Rename source subdirectory to `src-<version>`.
4. Check source to find out if there is a `configure` script (=buildtype GNU) or a `Makefile.PL` file (=buildtype pm). If not, build-all is likely to fail.
5. Create a file `buildinfo` in the applicatio subdirectory

   with the information required to get build-all to compile this application. The absolute minimum file contains `appname`, `version` and `buildtype`.

```
appname=nasm
version=0.98
buildtype=GNU
```

6. Run `build-all [archs]` in application subdirectory and check the current compile status in the files `out.<arch>` to find out if everything works as planned.

   ```
   tail -f out.<arch>
   ```

7. If this was a perl module, run `pm-touch-compiled` to make sure platform dependent subdirectories are properly marked in Store.

8. Check content of ver-¡version¿ to make sure everything looks OK.

9. Run `chkapp` to make sure everything went OK., `register` to insert and update the info on the application and to log what was done.

10. Run `linkup` and test if the program is working.

3 Common environment settings

Two files are generated with the common environment settings, `/store/etc/src.sh` and `/store/etc/src.csh`. One file for the Bourne shell family (sh, ksh, zsh, bash, etc) and one for the C shell family (csh, tcsh). These files are generated by `/store/etc/internal/make-env.pl` based on the content in `/store/etc/ENV/ENV-*`. They are regenerate every time the nightly commands are being run.

The format is quite simple, with '#' as the comment marker, and the fields in the env lines being 'username;group;priority;type;variable=content.

```
#
# Set path, check if directories exists.
*:*:1:p:PATH=/store/gnu/bin:/store/opt/*/bin
*:*:2:p:PATH=/store/bin:/store/sbin
!0:*:5:p:PATH=/usr/games
#
# Include in path without checking for the directories
*:*:1:n:XFILESEARCHPATH=/store/lib/X11/app-defaults/%N
*:*:9:n:XFILESEARCHPATH=/usr/local/lib/X11/%L/app-defaults/%N
#
# Single variable
*:*:9:s:PAGER=more
```

Username and group must be '*' or user or group id number. '!' negates the test. In the example, this make sure user root (0) do not get /usr/games as part of the PATH.

The path environment type 'p' and 'n' will remove duplicate entries, and only keep the highest priority entry. The single entry environment 's' will only keep the highest priority entry. The priorities is sorted as perl strings, '12' having lower priority then '9'.

The 'p' env type content might use filename globbing, as shown with `/store/opt/*/bin`.

If you are installing an application which requires special environment settings, the best way to handle this is to use a small shell wrapper to set the environment and then call the program. Renaming the binaries from file to file.exe and making a new shell script file which sets the environment is the best way. Only when this is no option, you should use the etc/ENV/ENV-application files. I suggest naming the files en ENV/ after the application they belong to, i.e ENV-netscape for netscape env settings and ENV-less for less env settings.

4 MIME mailcap files

When installing MIME content handlers, one wants to make them automatically available in mail programs and web browsers. This is done by placing a file in /store/etc/mailcaps/. The files are named mailca-<app>-<priority>, and looks like normal MIME mailcap entries:

```
audio/x-mpeg; /store/bin/mpg123 %s;
```

Based on these files, a list of MIME content handlers are generated into /store/etc/mailcap by /store/etc/internal/make-mailcap.pl. The priorities are sorted as numbers.

5 Emacs info and default.el files

5.1 Emacs info directory

All emacs info files should be installed in /store/info/. These files can (and probably should) be compressed with gzip. Based on the content of this subdirectory, a emacs info directory file (/store/info/dir) is generated by /store/etc/internal/make-emacs-dir.pl. This program requires a section like this in the info files:

```
INFO-DIR-SECTION Programming
START-INFO-DIR-ENTRY
* Cpp: (cpp).                   The GNU C preprocessor.
END-INFO-DIR-ENTRY
```

All emacs info and texinfo files should have a section like this. If it is missing, send a patch to the author. When working with .texi files, a section like this will make sure the resulting info files will be included in the info/dir file:

```
@dircategory Programming
@direntry
* Cpp: (cpp).                   The GNU C preprocessor.
@end direntry
```

5.2 Emacs default.el

Generates /store/lib/emacs/site-lisp/default.el and /store/share/emacs/site-lisp/default.el from default.el-<app> in the same directory.

6 Netscape plugins

To make sure Netscape finds the installed plugins, they must be installed in a common directory, and environment `NPX_PLUGIN_PATH` must be set to point to this directory. The environment should be set in a wrapper shell to make sure it is always set when netscape is running.

I propose using `/store/opt/ns-plugins/` as this common directory. I suggest naming the Netscape plugin applications with prefix '`nsp-`', to make it easier to track the different netscape plugins available.

7 GIMP plugins

[I do not know enough about gimp to make a suggestion here.]

8 List of recently installed applications

To make it easier for the users to keep track of the applications available in /store, the script `/store/etc/internal/news.pl` generates UNIX news files in `/store/news`. These files can then be accessed by news(1) to get a list of recently installed applications.

9 Various tools

- store-pere

check-libtool-libs.pl findold.pl nightly-fix-suid.pl sourcewatch.pl chkapps.pl fix-docdirs.pl prognews.pl usage.pl collectusage.pl showusage.pl vusage.pl

compare-linktrees store-app-dependson

10 Build-all – parallel multiplatform autocompile

The buildall system consist of the following configuration files:

~store/etc/master.conf Site specific configuration (information about master store etc).

~store/etc/buildhosts The list Store architectures to compile for, and the hosts to compile the application for this architecture.

$appdir/buildinfo The application specific information, located in the application master directory.

The following programs are part of buildall:

/store/bin/build-all The program called when starting to compile. This must be run in the application master directory.

/store/etc/internal/buildsubs Script called by build-all to do the work required.

/store/bin/pm-touch-compiled Perl module compile support script.

store/etc/internal/etc/pm-a-by-a Mapping mellom store-arkitektur og Perl5 arkitekturnavn

10.1 buildinfo variables

References

Store documentation, http://www.pvv.org/˜arnej/store/storedoc.html

Automatisk kompilering av Store-programmer p alle plattformer

==

Petter Reinholdtsen ¡pere@td.org.uit.no¿, 1999-07-27

IT-avdelingen bruker et system kalt 'build-all' hentet fra NTNU for dette. Nr det virker gr kompilering mye raskere enn fr. Nr det ikke virker gjres kompilering som fr. Kopi av NTNUs buildinfo filer ligger i store-applikasjon build-all/ntnu-buildinfo/

For autokompilere for flere arkitekturer lager du et /bin/sh script 'buildinfo' i applikasjonskatalogen p master, parallelt med fila 'registration', for sette endel variabler. Deretter kjrer du 'build-all'. Hvis alt gikk bra, har du etter en stund kompilert opp applikasjonen for alle plattformer. Sjekk out.* for se hvordan det gr med bygging av applikasjonen.

Nr kompileringen er ferdig p alle plattformer (siste linje i out.* er "...done"), s kjres chkapp og register p vanlig mte.

Eksempelfil for GNU configure programmer:

appname=zsh version=3.1.5 buildtype=GNU

Eksempelfil for Perl5 moduler:

appname=Net-Netmask.pm version=1.6 buildtype=pm

Flgende variabler kan settes:

appname Navn p applikasjonen som skal genereres, dvs katalognavnet i Master Store. M settes, ingen default. version Versjon p applikasjonen som skal genereres. M settes, ingen default. buildtype Hvordan kompilering av applikasjonen skal gjres. M settes, ingen default. Flgende er stttet:

X xmkmf og make GNU GNU Autoconf oppsett pm Perl5 module, dvs applikasjon med 'Makefile.PL' make ???. Forutsetter at konfigurering allerede er gjort.

prefix Kan vre blank. Default er '/store'. confstring Kan vre blank. Parameter til configure for buildtype GNU. Default er '–prefix=$prefix'. confcmd Kan vre blank, da brukes confstring eller standard for buildtype GNU ('./configure $confstring'). maketargets Kan vre blank. Parameter til make. Default er intet parameter. installtargets Kan vre blank. Parameter til make for a installere applikasjonen. Default for X er 'install install.man', for alle andre 'install' linkdownfirst linkupafter Kan vre blank. Liste over applikasjoner som linkes henholdsvis ned og opp fra linktreet fr og etter installasjon. Space-separert liste. postpostinstcmds Kan vre blank. Kommando som kjres etter postinst p maskinen er installasjonen ble gjennomfrt. linkupthis Liste over applikasjoner som skal linkes opp fr kompilering. Formatet er 'app versjon store', og flere applikasjoner skilles med skrstrek(/). Eksempel: linkupthis='glib 1.2.2/gtk 1.2.2' rsh Kan vre blank. Hvilken kommando som brukes for kjre kommandoer p andre maskiner. Default er 'ssh -x'. nice Kan vre blank. Nive-niv for 'configure'- og 'make'-kommendoene. Default er 'nice -15' timeout Kan vre blank. Timeout-verdi for postinst. Default er 25.

Tilgjengelige shell-variabler i confcmd og postpostinstcmds:

SARCH Store-arkitektur PMA Perl5 arkitekturnavn (for pm buildtype)

Buildall-systemet bestr av flgende filer:

/store/bin/build-all Oppstart-scriptet som kjres fra master-dir for applikasjonen /store/etc/internal/buildsubs Skriptet som gjr jobben /store/etc/internal/etc/pm-a-by-a Mapping mellom store-arkitektur og Perl5 arkitekturnavn $HOME/etc/buildhosts Liste over hvilke maskiner de forskjellige arkitekturene kompileres p. $HOME/etc/master.conf Informasjon om master-store. $appdir/buildinfo Informasjon om applikasjonen som skal kompileres