NAME

aegis − project change supervisor

aegis (ee.j.iz) *n.,* a protection, a defense.

SPACE REQUIREMENTS

You will need up to 250MB to unpack and build the *aegis* package. (This is the worst case seen so far, most systems have binaries smaller than this, 200MB is more typical.) Your mileage may vary.

SITE CONFIGURATION

The **aegis** package is configured using the *configure* shell script included in this distribution.

The *configure* shell script attempts to guess correct values for various system-dependent variables used during compilation, and creates the *Makefile* and *common/config.h* files. It also creates a shell script *config.status* that you can run in the future to recreate the current configuration.

Upgrading

The `./configure` script will look for an existing install of Aegis and use the existing configuration settings. This works best if the version you are upgrading is 4.11 or later.

To disable looking for an existing installation (maybe because you want to change the prefix), use the `./configure --with-no-aegis-configured` option.

To change the `AEGIS_UID` and `AEGIS_GID` values (these control the ownership of Aegis' system files) you need to set environment variables of these names **before** running the *./configure* script. You almost never need to do this, so if you have no idea what this is about, don't try to change them.

Before You Start

Before you start configuring, it is worth reading the *OTHER USEFUL SOFTWARE* section, below.

The *configure* script checks for the internationalization library and functions. If your system does not have them, it is worth fetching and installing **GNU Gettext** before you run the *configure* script. Make sure that the *msgfmt* command from GNU Gettext appears earlier in your command search PATH than the existing system ones, if any (this is very important for SunOS and Solaris). You must do the GNU gettext install *before* running the *configure* script, or the error messages, even for English speakers, will be terse and uninformative. Remember to use the GNU gettext configure *−−with-gnu-gettext* option if your system has native gettext tools.

The *configure* script checks for compression libraries and functions. If your system does not have them, you must download and install the **GNU zlib** compression library (see `http://www.gzip.org/zlib/` for download) and the **bzip2** compression library (see `http://www.bzip.org/` for download) before you run the *configure* script. These libraries are essential, Aegis will not build without them. (**Note:** `zlib` is not the same thing as `zlib`**`c`** which does something completely different.)

The *configure* script checks for the regular expression library and functions. If your system does not have them, it is worth fetching and installing **GNU rx** compression library before you run the *configure* script. (Note: test 81 will fail if the POSIX regular expression functions are not available.)

The GNOME libxml2 library (`http://xmlsoft.org/`) is used to parse XML, you will need version 1.8.17 or later. You do not have to install the rest of GNOME as this library is able to be used by itself. This package is **not** optional, you need it to successfully build Aegis.

The libcurl library (`http://curl.haxx.se/`) is used to fetch remote files. This library is optional, but some functionality, particularly *aedist −replay, will not work without it. If you are using a package based install, you will need the libcurl-dev or libcurl-devel package as well.*

Running Configure

Normally, you just *cd* to the directory containing *aegis*' source code and type

```
% ./configure --sysconfdir=/etc
...lots of output...
%
```

If you're using *csh* on an old version of System V, you might need to type

```
% sh configure --sysconfdir=/etc
...lots of output...
%
```

instead to prevent *csh* from trying to execute *configure* itself.

Running *configure* takes a minute or two. While it is running, it prints some messages that tell what it is doing. If you don't want to see the messages, run *configure* with its standard output redirected to */dev/null*; for example,

```
% ./configure --sysconfdir=/etc --quiet
%
```

There is a known problem with GCC 2.8.3 and HP/UX. You will need to set `CFLAGS = -O` in the generated Makefile. (The configure script sets it to `CFLAGS = -O2`.) This is because the code optimization breaks the fingerprints. If test 32 fails (see below) this is probably the reason.

There is a known problem with IRIX builds. You need to use the following configuration

```
# systune ncargs 0x8000
```

to increase the length of command lines.

For mips IRIX and IRIX64 using the MipsPro compiler up to at least version 7.3 you must specify the flag to allow –I for loop initializations. You may give either of:

```
CXXFLAGS='LANG:ansi-for-init-scope=ON'
CXXFLAGS='LANG:std'
```

Also required is *–lCio* but configure will test for that. Even using that library there remains a link failure due to:

```
Unresolved text symbol
"std::_List_base<undo_item*,std::allocator<undo_item*> >::clear(void)"
```

on several of the binaries. A work around for this problem is not known at this time.

By default, *configure* will arrange for the *make install* command to install the **aegis** package's files in */usr/local/bin*, */usr/local/com/aegis*, */usr/local/lib/aegis*, */usr/local/man* and */usr/local/share/aegis*. There are a number of options which allow you to control the placement of these files.

`--prefix=`*PATH*
: This specifies the path prefix to be used in the installation. Defaults to */usr/local* unless otherwise specified. The rest of these building instructions assume you are using the default */usr/local* as the install prefix.

`--exec-prefix=`*PATH*
: You can specify separate installation prefixes for architecture-specific files and architecture-independent files. Defaults to *${prefix}* unless otherwise specified.

`--bindir=`*PATH*
: This directory contains executable programs. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/bin* unless otherwise specified.

`--datadir=`*PATH*
: This directory contains installed data, such as the documentation, reports and shell scripts distributed with Aegis. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/share/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--libdir=`*PATH*
: This directory contains installed data, such as the error message catalogues. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/lib/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--mandir=`*PATH*
: This directory contains the on-line manual entries. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/man* unless otherwise specified.

`--sharedstatedir=`*PATH*
: This directory contains share state information, such as the Aegis lock file, and information on the location of the various Aegis projects. On a network, this directory may be shared between all machines; it **must** be mounted **read-write**. Defaults to *${prefix}/com/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--sysconfdir=`*PATH*
: Location of system configuration files. You should almost always use the */etc* directory.

configure ignores any other arguments that you give it.

On systems that require unusual options for compilation or linking that the *aegis* package's *configure* script does not know about, you can give *configure* initial values for variables by setting them in the environment. In Bourne-compatible shells, you can do that on the command line like this:

```
$ CC='gcc -traditional' LIBS=-lposix \
  ./configure --sysconfdir=/etc
...lots of output...
$
```

Here are the *make* variables that you might want to override with environment variables when running *configure*.

Variable: CC
: C compiler program. The default is *cc*.

Variable: INSTALL
: Program to use to install files. The default is *install* if you have it, *cp* otherwise.

Variable: LIBS
: Libraries to link with, in the form `-l`*foo* `-l`*bar*. The *configure* script will append to this, rather than replace it.

If you need to do unusual things to compile the package, the author encourages you to figure out how *configure* could check whether to do them, and mail diffs or instructions to the author so that they can be included in the next release.

Common Problem

It is very common that other packages, such as *gettext*, *rx* and *zlib* are installed using */usr/local* as the prefix. However, the configure script can't work this out, even when it, too, is using */usr/local* as the prefix.

To cope with this, you need to say

```
$ CPPFLAGS=-I/usr/local/include LDFLAGS=-L/usr/local/lib \
  ./configure --sysconfdir=/etc
...lots of output...
$
```

when running configure. Substitute the appropriate prefix if you are using something other than the default */usr/local* prefix. Watch the output... it should now find your installed packages correctly.

GCC Version 3.*

On some operating systems, notabley MacOsX Jaguar and Panther, g++ versions 3.* will produce link-time errors complaining of missing typeinfo symbols. The only known fix for this problem is to use GCC version 2.95, 2.96 or 4.*. This means MacOsX Tiger does not have the problem.

AIX Command Line Lengths

For some reason, AIX has a very short command line length limit by default. You can extend this by using the command

```
$ systune ncargs 0x8000
$
```

You will need to do this to build Aegis. It has some very long link lines.

PRIVILEGES

There are a number of items in the generated *Makefile* and *common/config.h* file which affect the way *aegis* works. If they are altered too far, *aegis* will not be able to function correctly.

AEGIS_MIN_UID
: This specifies the minimum unprivileged uid on your system. UIDs less than this may not own projects, or play any other role in an aegis project. The default value is 100.

AEGIS_MIN_GID
: This specifies the minimum unprivileged GID on your system. GIDs less than this may not own projects, or play any other role in an aegis project. The default value is 10.

AEGIS_USER_UID
: This is the owner of files used by *aegis* to record pointers to your projects. It is *not* used to own projects (i.e. it must be less than AEGIS_MIN_UID). If possible, the *configure* script tries to work out what value was used previously, but you must specify the −−prefix option correctly for this to work. Because of operating system inconsistencies, this is specified numerically so that *aegis* will work across NFS. The default value is 3.

AEGIS_USER_GID
: This is the group of files used by *aegis* to record pointers to your projects. It is *not* used as the group for projects (i.e. it must be less than AEGIS_MIN_GID). If possible, the *configure* script tries to work out what value was used previously, but you must specify the −−prefix option correctly for this to work. Because of operating system inconsistencies, this is specified numerically so that *aegis* will work across NFS. The default value is 3.

DEFAULT_UMASK
: When *aegis* runs commands for you, or creates files or directories for you, it will use the defined project umask. This is a project attribute, and may be altered using the *aepa*(1) command. The DEFAULT_UMASK is the umask initially given to all new projects created by the *aenpr*(1) command. The default value of DEFAULT_UMASK is 026. See the comments in the *common/config.h* file for an explanation of the alternatives.

It is required that *aegis* run set-uid-root for all of its functionality to be available. It is **not** possible to create an "aegis" account and make *aegis* run set-uid-aegis. This is because *aegis* does things as various different user IDs, sometimes as many as 3 in the one command. This allows *aegis* to use UNIX security rather than inventing its own, and also allows *aegis* to work across NFS. To be able to do these things, *aegis* must be set-uid-root. Appendix D of the *Aegis User Guide* explains why *aegis* must run set-uid-root; please read it if you have concerns.

Remember Your Settings

It is important to remember your configuration settings. This way, it will be a simple matter when it comes time to upgrade Aegis.

BUILDING AEGIS

All you should need to do is use the

```
% make
...lots of output...
%
```

command and wait. When this finishes you should see a directory called *bin* containing several files: *aegis*, *aereport*, *aefind*, *aefp*, and *fmtgen*.

aegis The *aegis* program is a project change supervisor.

aefp The *aefp* program may be used to "fingerprint" files. It is used to test Aegis (see the testing section, below) but it isn't installed.

aereport The *aereport* program is used to query Aegis' database.

aefind The *aefind* program is used to find files.

fmtgen The *fmtgen* program is a utility used to build the *aegis* package; it is not intended for general use and should not be installed.

You can remove the program binaries and object files from the source directory by using the

```
% make clean
...lots of output...
%
```

command. To remove all of the above files, and also remove the *Makefile* and *common/config.h* and *config.status* files, use the

```
% make distclean
...lots of output...
%
```

command.

The file *aux/configure.in* is used to create *configure* by a GNU program called *autoconf*. You only need to know this if you want to regenerate *configure* using a newer version of *autoconf*.

Upgrading

When upgrading from one release to a newer one, it is important that all of the machines on your network are running the same release of Aegis. This minimizes the possibility of database incompatibilities. In general, Aegis is backwards compatible with earlier releases, but not forwards compatible in the face of new capabilities.

OTHER USEFUL SOFTWARE

Before describing how to test *aegis*, you may need to grab some other free software, because the tests require it in some cases, and because it is generally useful in others.

GNOME libxml2

The GNOME libxml2 library (`http://xmlsoft.org/`) is used to parse XML. Version 1.8.17 or later. You do not have to install the rest of GNOME as this library is able to be used by itself. This package is **not** optional, you need it to successfully build Aegis.

cook This is a dependency maintenance tool (DMT). An example of a well-known DMT is *make*(1), however this old faithful is mostly not sufficiently capable to meet the demands placed on it by the *aegis* program, but *cook* certainly is. The *cook* package is written by the same author as *aegis*. The *cook* package is necessary if test 11 is to be meaningful. It is also used in the documentation. The *cook* program may be found at the same archive site as the *aegis* program. The *cook* program is available under the terms of the GNU General Public License.

GNU diff

If the *diff*(1) utility supplied by your flavor of Unix does not have the **-c** option, you will need GNU diff for *aepatch*(1) to work (and the *aepatch*(1) tests to pass). Context differences are also helpful for reviewing changes. GNU diff is essential for Solaris, because the Solaris diff has bugs that Aegis' tests uncover.

GNU patch

For best results with the *aepatch*(1) and *aedist(1) when receiving change sets, you need the GNU patch utility.*

iso-codes

This package provides the ISO 639 and ISO 639-3 language code lists, the ISO 3166 territory code list, list as XML files.
Homepage: http://pkg-isocodes.alioth.debian.org/

RCS This is a source control package, and is available from any of the GNU archives. (It is best to compile and install RCS *after* GNU diff. This is because the RCS configuration hard-codes the pathnames of the GNU diff utilities it needs into the RCS executables.) This package isn't essential as Aegis comes with its own *aesvt*(1) history tool – although you are free to use any history tool you like.

GNU Gettext
Many systems do not yet supply the *gettext*(3) function. Aegis uses this function to internationalize its error messages. If your system does not have this function, you should fetch and install GNU Gettext *before* running the *configure* script. If you do not, Aegis will still work, but the error messages will be rather terse, even for English speakers. (You will be able to tell if your system has the internationalization library and functions, because the *configure* script will report finding `-lintl` and *(CWlibintl.h* and `msgfmt` in its running commentary.) Please note that the GNU Gettext implementation is likely to be superior to the one supplied with your system, if any. Remember to use the GNU gettext configure *−−with-gnu-gettext* option if your system has native gettext tools.

Please note: if you install GNU gettext package into */usr/local* (for example) you must ensure that the Aegis *./configure* script is told to also look in */usr/local/include* for include files (CFLAGS), and */usr/local/lib* for library files (LDFLAGS). Otherwise the *./configure* script will incorrectly conclude that GNU Gettext has not been installed.

GNU Gettext version 0.11.1 or later is recommended.

GNU Groff
This GNU software replaces the documentation tools which (sometimes) come with UNIX. They produce superior error messages, and support a wider range of functionality and fonts. The *Aegis* User Guide was prepared with GNU Groff. You need GNU Groff 1.14 or later.

bison This GNU software is a replacement for *yacc*(1). Some systems have very sick yaccs, and *bison* may be necessary if your system include files disagree strongly with your system's yacc. The generated *Makefile* will use bison if you have it.

fhist This software, available under the terms of the GNU General Public License, is a set of file history and comparison utilities. It was originally written by David I. Bell, and is based on the minimal difference algorithm by Eugene W. Myers. This copy is enhanced and maintained by the same author as *Aegis*, and may be found at the same archive site, in the same directory.

rx This library provides POSIX regular expressions, for systems which don't have them. (Note: test 81 will fail if the POSIX regular expression functions are not available.)

zlib This library provides access to the GNU Zip (de)compression algorithm(s). It is essential to have this installed before you build Aegis. The home page may be found at `http://www.gzip.org/zlib/` if you need to download it. Note: this is not the same as **`zlibc`** which is Linux specific.

tkdiff This program shows the difference between two text files, nicely highlighted in color. This is used by the *tkaer* and *aecomp* scripts (and probably others as they are contributed). By John M. Klassa, http://www.ede.com/free/tkdiff

libmagic If *libmagic*(3) is present, the *aeget*(1) CGI handler will use it to determine the MIME type of files. This is installed by **file** version 4.0 and later (`ftp://ftp.astron.com/pub/file/`), and uses the same database as the *file*(1) command. If this library is not present when Aegis is built, a much less accurate method will be used.

The tests also depend on the presence of a number of common UNIX programs, including but not limited to: *cc*, *cmp*, *diff*, *ed*, *find*, *make*, etc. Depending on your version of UNIX, some or all of these programs may be in optional packages. (This is especially true of Linux.) You need to ensure that these programs are correctly installed before you run the tests.

TESTING AEGIS

The *Aegis* program comes with a test suite. To run this test suite, use the command

```
% make sure
...lots of output...
Passed All Tests
%
```

The tests take a minute or two each, with a few very fast, and a couple very slow, but it varies greatly depending on your CPU.

Known Problems

In order to get the long form of the error messages on Solaris, it is necessary to install GNU Gettext before running ./configure, and once ./configure has been run you need to edit the Makefile to statically link the executables.

The *test/00/t0011a.sh* file assumes the *cook*(1) command by the author is somewhere in the command search path. This test reproduces the example used in Chapter 3 of the User Guide. If the *cook*(1) command is not available, this test gives a pass result without testing anything.

If you are using HPUX and GCC, test 32 fails if you use −O2. You need to edit the Makefile to only optimize at −O, delete the objects and rebuild.

If you are using Solaris' diff, test 133 will report "no result". You need to install GNU diff, because the Solaris diff has bugs.

If you are using Sun's *tmpfs* file system as your */tmp* directory, the tests will fail. This is because the *tmpfs* file system does not support file locking. Set the *AEGIS_TMP* environment variable to somewhere else before running the tests. Something like

```
% setenv AEGIS_TMP /usr/tmp
%
```

is usually sufficient if you are using C shell, or

```
$ AEGIS_TMP=/usr/tmp
$ export AEGIS_TMP
$
```

if you are using Bourne shell. Remember, this must be done before running the tests.

If the tests fail due to errors complaining of "user too privileged" you will need to adjust the *AEGIS_MIN_UID* defined in the *common/config.h* file. Similarly for "group too privileged", although this is rarer. This error message will also occur if you run the tests as root: the tests must be run as a mortal each time.

If the POSIX regular expression functions are not available, test 81 will fail. The GNU rx library provides these. Installing it and re-configuring and re-building Aegis will solve the problem.

TESTING SET-UID-ROOT

If the *Aegis* program is not set-uid-root then it runs in "test" mode which gives you some confidence that *Aegis* is working before being tested again when it is set-uid-root. Two pass testing like this means that you need not trust your system to a set-uid-root program which is not known to work.

You will need to do a little of the install, to create the directory which will contain *Aegis*' lock file. (Note that these building instructions assume you are using the default */usr/local* as the install prefix. You will need to substitute as appropriate if you are using some other prefix.)

```
# make install-libdir
mkdir /usr/local/lib/aegis
chown 3 /usr/local/lib/aegis
chgrp 3 /usr/local/lib/aegis
chmod 0755 /usr/local/lib/aegis
mkdir /usr/local/com/aegis
chown 3 /usr/local/com/aegis
chgrp 3 /usr/local/com/aegis
```

```
chmod 0755 /usr/local/com/aegis
chown root bin/aegis
chmod 4755 bin/aegis
#
```

As you can see, the previous command also changed *Aegis* to be set-uid-root. Once this has been done, *Aegis* should be tested again, in the same manner as before.

```
% make sure
...lots of output...
Passed All Tests
%
```

You should test *Aegis* as a mortal in both passes, rather than as root, to be sure the set-uid-root functionality is working correctly.

It is required that *Aegis* run set-uid-root for all of its functionality to be available. It is **not** possible to create an "aegis" account and make *Aegis* run set-uid-aegis. This is because *Aegis* does things as various different user IDs, sometimes as many as 3 in the one command. This allows *Aegis* to use UNIX security rather than inventing its own, and also allows *Aegis* to work across NFS. To be able to do these things, *Aegis* must be set-uid-root. Appendix D of the *Aegis User Guide* explains why *Aegis* must run set-uid-root; please read it if you have concerns.

INSTALLING AEGIS

As explained in the *SITE CONFIGURATION* section, above, the *Aegis* package is installed under the */usr/local* tree by default. Use the `--prefix=`*PATH* option to *configure* if you want some other path.

All that is required to install the *Aegis* package is to use the

```
% make install
...lots of output...
%
```

command. Control of the directories used may be found in the first few lines of the *Makefile* file if you want to bypass the *configure* script.

The above procedure assumes that the *soelim*(1) command is somewhere in the command search *PATH*. The *soelim*(1) command is available as part of the *GNU Groff* package, mentioned below in the *PRINTED MANUALS* section. If you don't have it, but you do have the *cook* package, then a link from *roffpp* to *soelim* will also work.

The above procedure also assumes that the *$(prefix)/man/man1* and *$(prefix)/man/man5* directories already exist. If they do not, you will need to *mkdir* them manually.

USER CONFIGURATION

The *Aegis* command is assumed to be in a generally accessible place, otherwise users will need to add the relevant directory to their PATH. Users should add

```
source /usr/local/lib/aegis/cshrc
```

to the end of their *.cshrc* file for the recommended aliases. (Note that these building instructions assume you are using the default */usr/local* as the install prefix. You will need to substitute as appropriate if you are using some other prefix.)

There is also a *profile* for users of the Bourne shell (it assumes you have a version of the Bourne shell which has functions). Users should add

```
. /usr/local/share/aegis/profile
```

to the end of their *.profile* file for the recommended aliases. (This *profile* assumes that users are using a Bourne shell which understands functions.)

The */usr/local/com/aegis/state* file contains pointers to "system" projects. Users may add their own project pointers (to their own projects) by putting a search path into the *AEGIS_PATH* environment variable. The system part is always automatically appended by *Aegis*. The default, already set by the */usr/local/lib/aegis/cshrc* file, is *$HOME/lib/aegis*. Do not create this directory, *Aegis* is finicky and wants to do this itself.

Where projects reside is completely flexible, be they system projects or user projects. They are not kept under the */usr/local/com/aegis* directory, this directory only contains pointers. (Note that these building instructions assume you are using the default */usr/local* as the install prefix. You will need to substitute as appropriate if you are using some other prefix.)

Web Interface

If you have a Web server, you may like to install the Aegis web interface. You do this by copying the *aeget* script from */usr/local/bin/aeget* into your web server's *cgi-bin* directory. There is a *aeget.instal* helper script, if you don't know where your web server's *cgi-bin* directory is.

You may prefer to use a symbolic link, as this will be more stable across Aegis upgrades. However, this requires a corresponding *follow-symlinks* setting in your web server's configuration file. (Use the *aeget.instal* *−s* option.)

You may need to wrap *aeget* with a script which sets the *AEGIS_PATH* environment variable, if you want it to be able to see more projects than just the global projects. You may also need to set the *PATH* environment variable, if you don't have the Aegis install path in the default path.

(Note that these building instructions assume you are using the default */usr/local* as the install prefix. You will need to substitute as appropriate if you are using some other prefix.)

PRINTED MANUALS

This distribution contains the sources to all of the documentation for *Aegis*, however the simplest way to get the documentation is by anonymous FTP; PostScript files of the User Guide and Reference Manual are available from the FTP sites listed in the README file.

The Reference Manual contains the README and BUILDING files, as well as all of the section 1 and section 5 manual pages. The Reference Manual is about 200 pages long.

The User Guide contains information about how to use Aegis, including a fully worked example. The User Guide is about 100 pages long.

TIME SYNCHRONIZATION

The *Aegis* program uses time stamps to remember whether various events have happened and when. If you are using *Aegis* in a networked environment, typically a server and data-less workstations, you need to make absolutely sure that all of the machines agree about the time.

If possible, use the time daemon. Otherwise, use *rdate*(8) via *cron*(8) every hour or less.

GETTING HELP

If you need assistance with *Aegis*, please do not hesitate to contact the author at

Peter Miller <pmiller@opensource.org.au>

Any and all feedback is welcome.

When reporting problems, please include the version number given by the

```
% aegis -version
aegis version 4.25.D510
...
%
```

command. Please run this command to get the exact number, do not send the text of this example.

Runtime Checking

In the *common/main.h* file, there is a define of *DEBUG* in comments. If the comments are removed, extensive debugging is turned on. This causes some performance loss, but performs much run-time checking and adds the **−TRAce** command line option.

When the **−TRAce** command line option is followed by one or more file names, it turns on execution traces in those source files. It is usually best to place this on the end of the command line so that names of the files to be traced are not confused with other file names or strings on the command line.

Problem Reports

If you send email to the author, please include the following information:

1. The type of UNIX
   The author will need to know the brand and version of UNIX you are using, or if it is not UNIX but something else. The output of "uname −sr" is usually sufficient (but not all systems have it).
2. The Version Number
   In any information you send, please include the version number reported in the *common/patchlevel.h* file, or `` `aegis -vers` `` if you can get it to compile.
3. The Archive Site
   When and where you obtained this version of *Aegis*. If you tell me nothing else, tell me this (and, hopefully, why you did nothing else).
4. Unpacking
   Did you have problems unpacking *Aegis*? This probably isn't a problem with the .tar.Z distribution, but you could have obtained a shar format copy.
5. Building
   Did you have problems building *Aegis*? This could have been the instructions included, it could have been the configure script, it could have been the Makefile, or anything else.
6. Testing, Non-Set-Uid
   Did you have problems with the tests? You could have had problems running them, or some of them could have failed. If some tests fail but not others, please let me know *which* ones failed, and include the fact that *Aegis* was **not** set-uid-root at the time. The −k option to *make* can be useful if some tests fail but not others.
7. Testing, Set-Uid-Root
   Did you have problems with the tests when *Aegis* was set-uid-root? You could have had problems running them, or some of them could have failed. If some tests fail but not others, please let me know *which* ones failed, and include the fact that *Aegis* was set-uid-root at the time.
8. Installation
   Did you have problems installing *Aegis*? This could have been the instructions, or anything else.

At this point it would probably be a very good idea to print out the manual entries and read them carefully. You will also want to print a copy of the User Guide; if you don't have groff, there should be a PostScript copy at the archive site. It is a known flaw that the User Guide is incomplete, contributions are most welcome.

9. The Example Project
   After reading the User Guide, it is often useful to manually run through the example in chapter 3. You will need to do more than one change, hopefully several; the first change is not representative of the system. Did you manually do the example? Did you find flaws in the User Guide or manual entries?
10. Using Aegis
    Did you have problems using *Aegis*? This is a whole can of worms. If possible, include a shell script similar to the tests which accompany *Aegis*, which reproduces the bug. Exit code 1 on failure (bug), exit code 0 on success (for when bug is fixed).
11. The Source Code
    Did you read the code? Did you write some code? If you read the code and found problems, fixed them, or extended *Aegis*, these contributions are most welcome. I reserve the right to modify or reject such contributions.

The above list is inclusive, not exclusive. Any and all feedback is greatly appreciated, as is the effort and interest required to produce it.

LICENSE

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

WINDOWS-NT

It is possible to build Aegis for Windows-NT. I have only done this using the Cygnus freeware CygWin32 system, though it may be possible with other Unix porting layers also.

Caveat

This document only describes a **single user** port of Aegis to Windows NT.

Aegis depends on the underlying security provided by the operating system (rather than re-invent yet another security mechanism). However, in order to do this, Aegis uses the POSIX *seteuid*(2) system call, which has no direct equivalent on Windows NT. This makes porting difficult. **Single user** ports are possible (*e.g.* using Cygwin (http://www.cygwin.com/), but are not usually what folks want.

Compounding this is the fact that many sites want to develop their software for both Unix and Windows NT simultaneously. This means that the security of the repository needs to be guaranteed to be handled in the same way by both operating systems, otherwise one can act as a "back door" into the repository. Many sites do not have the same users and permissions (sourced from the same network register of users) on both Unix and Windows NT, making the mapping almost impossible even if the security models did actually correspond.

Most sites using Aegis and Windows NT together do so by running Aegis on the Unix systems, but building and testing on the NT systems. The work areas and repository are accessed via Samba or NFS.

The Source

You need to FTP the Cygwin system from RedHat. It can be found at

http://www.cygwin.com/

and then follow the links. The original version used was B20.1, but more recently 1.1.7 has been used.

It is *absolutely essential* to run the *mkpasswd* and *mkgroup* commands, otherwise Aegis will give fatal errors about unknown users and groups. See the Cygwin README for instructions.

Mounting Things

You need to mount a directory onto `/tmp`, or lots of things, and especially *bash*(1), don't work. If you are in a heavily networked environment, like me, you need to know that using a networked drive for `/tmp` just doesn't work. I have no idea why. Use

mount C:/temp /tmp

instead. (Or some other local drive.)

Just a tip for all of you who, like me, know Unix much better than you know Windows-NT: the left-hand mount argument needs to be specified with a drive letter (*e.g.* `C:`*) rather than with a double slash (e.g. not* `//C`*) unless its Windows-NT name starts with* \\.

You need to follow the install instructions about */bin/sh*, otherwise shell scripts that start with `#!/bin/sh` don't work, among other things. This includes the `./configure` script, and the scripts it writes (*e.g.* `config.status`).

You will want to mount your various network drives onto the same places they appear on your Unix hosts. This way you don't need to learn two names for all your files.

Mounts persist across Cygwin sessions. They are stored in a registry file somewhere. You will not need to do all this every time!

Too Much Administrator

If you have administrator privilege on your Windows NT box, you need to get rid of it. (Have a second admin account instead.) This is because Windows NT will make the files belong to the wrong user for files on *some* partitions, like */tmp*. (This took me days to work out!) This confuses both Aegis *and* RCS.

If you get weird "Permission denied" errors from amazingly unlikely causes, this is probably why.

Before You Start

There are several pieces of software you need before you can build Aegis on Cygwin.

I'm going to keep mentioning "your local GNU mirror". You can find
: GNU at `http://www.gnu.org`, however you are better off using a local mirror, and these are scattered around the globe. Follow the "mirrors" link on their front page to find your closest mirror. Also, it's often a good idea to configure these packages with the "−with-gnu-gettext" option to their ./configure commands.

Do not use WinZip to unpack the tarball. It has a nasty habit of
: turning all of the newlines into CRLFs. This will confuse *lots* of utilities, especially GNU Groff. Use the "*tar xzf aegis-4.25.tar.gz*" command from within Cygwin.

Make sure the Cygwin you are using has GNU Groff 1.15 or later
: (use a "groff −v" command). Grab and install the latest from your local GNU mirror, if it isn't.

util-linux
: You need to get GNU rx, but to make it work you have to find a *tsort* command, so that GNU rx's *./configure* script works. Try the latest copy of `system/misc/util-linux-?.?.tar.gz` from the `metalab.unc.edu` Linux archive (or a mirror). Simply build and install *misc-utils/tsort.c* by hand.

GNU rx
: Once you have *tsort* installed, you will be able to get GNU rx configured. Get a copy from your local GNU mirror.

zlib
: You need to grab a copy of *zlib*; the same source as works for Unix will work for Cygwin. It will install as a static library.

GNU diffutils
: You need GNU diffults, because when you come to configure GNU RCS (next) it would otherwise complain about a stupid *diff* and a missing *diff3* command. The *install-sh* script is broken, so you'll need to do the final step in the install by hand.

GNU RCS
: All of Aegis' tests assume RCS is present. Also, you are going to need *something* for a history tool. The *install-sh* script is broken, so you'll need to do the final step in the install by hand.

Configure

The configure and build step should be the same as for Unix, as described above. All the problems I encountered were to do with getting the mounts just right. (But expect it to be dog slow compared to Linux or FreeBSD on the same box.)

Sharutils
: You need the *uudecode* command for several of the tests, and this may be found in the GNU Sharutils package. You can get a copy from your local GNU mirror.

The configure step is almost the same as for Unix. I know you are itching to get typing, but read through to the install section before you configure anything.

```
bash$ ./configure
...lots of output...
bash$
```

Build

The build step is exactly the same as for Unix, and you shouldn't notice any difference...

```
bash$ make
bash$
```

Test

The tests are run in the same way as the Unix tests, but you don't need to run the set-uid-root variants, because no such thing exists under Windows NT.

```
bash$ make sure
...lots of output...
Passed All Tests
bash$
```

Unfortunately, it isn't that simple. There are a number of things you will see go wrong...

- Several tests fail because *ed* isn't there.
- Several tests fail because *ci* (RCS 5.7) dumps core much too often for my liking.
- A couple of tests fail because they don't expect the ".exe" extension on executable files.
- A couple of tests (notably, the *aedist* tests) fail because of the CRLF *vs* NL dichotomy. This means that the expected results don't match, not that it isn't working.

Despite all the bad news, the vast majority of tests pass, and the others have good excuses.

Install

Installing the software works as usual, though you need to make some choices right at the start (I told you to read this all the way through first). If you want to use the "*/usr/local*" prefix (or any other install prefix) you mount it right at the start. For anything other than the "*/usr/local*" default prefix, you also needed to give a "**-prefix=***blahblah" argument to the configure script, right at the start.*

bash$ *make install*
...lots of output...
bash$

// vim: set ts=8 sw=4 et :