NAME

ExtUtils::MM_Unix - methods used by ExtUtils::MakeMaker

SYNOPSIS

```
require ExtUtils::MM_Unix;
```

DESCRIPTION

The methods provided by this package are designed to be used in conjunction with ExtUtils::MakeMaker. When MakeMaker writes a Makefile, it creates one or more objects that inherit their methods from a package `MM`. MM itself doesn't provide any methods, but it ISA ExtUtils::MM_Unix class. The inheritance tree of MM lets operating specific packages take the responsibility for all the methods provided by MM_Unix. We are trying to reduce the number of the necessary overrides by defining rather primitive operations within ExtUtils::MM_Unix.

If you are going to write a platform specific MM package, please try to limit the necessary overrides to primitive methods, and if it is not possible to do so, let's work out how to achieve that gain.

If you are overriding any of these methods in your Makefile.PL (in the MY class), please report that to the makemaker mailing list. We are trying to minimize the necessary method overrides and switch to data driven Makefile.PLs wherever possible. In the long run less methods will be overridable via the MY class.

METHODS

The following description of methods is still under development. Please refer to the code for not suitably documented sections and complain loudly to the makemaker@perl.org mailing list. Better yet, provide a patch.

Not all of the methods below are overridable in a Makefile.PL. Overridable methods are marked as (o). All methods are overridable by a platform specific MM_*.pm file.

Cross-platform methods are being moved into MM_Any. If you can't find something that used to be in here, look in MM_Any.

Methods

os_flavor

Simply says that we're Unix.

c_o (o)

Defines the suffix rules to compile different flavors of C files to object files.

cflags (o)

Does very much the same as the cflags script in the perl distribution. It doesn't return the whole compiler command line, but initializes all of its parts. The const_cccmd method then actually returns the definition of the CCCMD macro which uses these parts.

const_cccmd (o)

Returns the full compiler call for C programs and stores the definition in CONST_CCCMD.

const_config (o)

Defines a couple of constants in the Makefile that are imported from %Config.

const_loadlibs (o)

Defines EXTRALIBS, LDLOADLIBS, BSLOADLIBS, LD_RUN_PATH. See *ExtUtils::Liblist* for details.

constants (o)

```
my $make_frag = $mm->constants;
```

Prints out macros for lots of constants.

depend (o)

Same as macro for the depend attribute.

init_DEST

```
$mm->init_DEST
```

Defines the DESTDIR and DEST* variables paralleling the INSTALL*.

init_dist

```
$mm->init_dist;
```

Defines a lot of macros for distribution support.

```
  macro         description                     default

  TAR           tar command to use              tar
  TARFLAGS      flags to pass to TAR            cvf

  ZIP           zip command to use              zip
  ZIPFLAGS      flags to pass to ZIP            -r

  COMPRESS      compression command to          gzip --best
                use for tarfiles
  SUFFIX        suffix to put on                .gz
                compressed files

  SHAR          shar command to use             shar

  PREOP         extra commands to run before
                making the archive
  POSTOP        extra commands to run after
                making the archive

  TO_UNIX       a command to convert linefeeds
                to Unix style in your archive

  CI            command to checkin your         ci -u
                sources to version control
  RCS_LABEL     command to label your sources   rcs
-Nv$(VERSION_SYM): -q
                just after CI is run

  DIST_CP       $how argument to manicopy()     best
                when the distdir is created

  DIST_DEFAULT  default target to use to        tardist
                create a distribution

  DISTVNAME     name of the resulting archive
$(DISTNAME)-$(VERSION)
                (minus suffixes)
```

dist (o)

```
my $dist_macros = $mm->dist(%overrides);
```

Generates a make fragment defining all the macros initialized in init_dist.

%overrides can be used to override any of the above.

dist_basics (o)

Defines the targets distclean, distcheck, skipcheck, manifest, veryclean.

dist_ci (o)

Defines a check in target for RCS.

dist_core (o)

```
my $dist_make_fragment = $MM->dist_core;
```

Puts the targets necessary for 'make dist' together into one make fragment.

dist_target

```
my $make_frag = $MM->dist_target;
```

Returns the 'dist' target to make an archive for distribution. This target simply checks to make sure the Makefile is up-to-date and depends on $(DIST_DEFAULT).

tardist_target

```
my $make_frag = $MM->tardist_target;
```

Returns the 'tardist' target which is simply so 'make tardist' works. The real work is done by the dynamically named tardistfile_target() method, tardist should have that as a dependency.

zipdist_target

```
my $make_frag = $MM->zipdist_target;
```

Returns the 'zipdist' target which is simply so 'make zipdist' works. The real work is done by the dynamically named zipdistfile_target() method, zipdist should have that as a dependency.

tarfile_target

```
my $make_frag = $MM->tarfile_target;
```

The name of this target is the name of the tarball generated by tardist. This target does the actual work of turning the distdir into a tarball.

zipfile_target

```
my $make_frag = $MM->zipfile_target;
```

The name of this target is the name of the zip file generated by zipdist. This target does the actual work of turning the distdir into a zip file.

uutardist_target

```
my $make_frag = $MM->uutardist_target;
```

Converts the tarfile into a uuencoded file

shdist_target

```
my $make_frag = $MM->shdist_target;
```

Converts the distdir into a shell archive.

dlsyms (o)

Used by some OS' to define DL_FUNCS and DL_VARS and write the *.exp files.

Normally just returns an empty string.

dynamic_bs (o)

Defines targets for bootstrap files.

dynamic_lib (o)

Defines how to produce the *.so (or equivalent) files.

exescan

Deprecated method. Use libscan instead.

extliblist

Called by init_others, and calls ext ExtUtils::Liblist. See *ExtUtils::Liblist* for details.

find_perl

Finds the executables PERL and FULLPERL

fixin

```
$mm->fixin(@files);
```

Inserts the sharpbang or equivalent magic number to a set of @files.

force (o)

Writes an empty FORCE: target.

guess_name

Guess the name of this package by examining the working directory's name. MakeMaker calls this only if the developer has not supplied a NAME attribute.

has_link_code

Returns true if C, XS, MYEXTLIB or similar objects exist within this object that need a compiler. Does not descend into subdirectories as needs_linking() does.

init_dirscan

Scans the directory structure and initializes DIR, XS, XS_FILES, C, C_FILES, O_FILES, H, H_FILES, PL_FILES, EXE_FILES.

Called by init_main.

init_MANPODS

Determines if man pages should be generated and initializes MAN1PODS and MAN3PODS as appropriate.

init_MAN1PODS

Initializes MAN1PODS from the list of EXE_FILES.

init_MAN3PODS

Initializes MAN3PODS from the list of PM files.

init_PM

Initializes PMLIBDIRS and PM from PMLIBDIRS.

init_DIRFILESEP

Using / for Unix. Called by init_main.

init_main

Initializes AR, AR_STATIC_ARGS, BASEEXT, CONFIG, DISTNAME, DLBASE, EXE_EXT,

FULLEXT, FULLPERL, FULLPERLRUN, FULLPERLRUNINST, INST_*, INSTALL*, INSTALLDIRS, LIB_EXT, LIBPERL_A, MAP_TARGET, NAME, OBJ_EXT, PARENT_NAME, PERL, PERL_ARCHLIB, PERL_INC, PERL_LIB, PERL_SRC, PERLRUN, PERLRUNINST, PREFIX, VERSION, VERSION_SYM, XS_VERSION.

init_others

Initializes EXTRALIBS, BSLOADLIBS, LDLOADLIBS, LIBS, LD_RUN_PATH, LD, OBJECT, BOOTDEP, PERLMAINCC, LDFROM, LINKTYPE, SHELL, NOOP, FIRST_MAKEFILE, MAKEFILE_OLD, NOECHO, RM_F, RM_RF, TEST_F, TOUCH, CP, MV, CHMOD, UMASK_NULL, ECHO, ECHO_N

init_linker

Unix has no need of special linker flags.

init_lib2arch

```
$mm->init_lib2arch
```

init_PERL

```
$mm->init_PERL;
```

Called by init_main. Sets up ABSPERL, PERL, FULLPERL and all the *PERLRUN* permutations.

```
PERL is allowed to be miniperl
FULLPERL must be a complete perl

ABSPERL is PERL converted to an absolute path

*PERLRUN contains everything necessary to run perl, find it's
     libraries, etc...

*PERLRUNINST is *PERLRUN + everything necessary to find the
     modules being built.
```

init_platform

platform_constants

Add MM_Unix_VERSION.

init_PERM

```
  $mm->init_PERM
```

Called by init_main. Initializes PERL_*

init_xs

```
$mm->init_xs
```

Sets up macros having to do with XS code. Currently just INST_STATIC, INST_DYNAMIC and INST_BOOT.

install (o)

Defines the install target.

installbin (o)

Defines targets to make and to install EXE_FILES.

linkext (o)

Defines the linkext target which in turn defines the LINKTYPE.

lsdir

Takes as arguments a directory name and a regular expression. Returns all entries in the directory that match the regular expression.

macro (o)

Simple subroutine to insert the macros defined by the macro attribute into the Makefile.

makeaperl (o)

Called by staticmake. Defines how to write the Makefile to produce a static new perl.

By default the Makefile produced includes all the static extensions in the perl library. (Purified versions of library files, e.g., DynaLoader_pure_p1_c0_032.a are automatically ignored to avoid link errors.)

makefile (o)

Defines how to rewrite the Makefile.

maybe_command

Returns true, if the argument is likely to be a command.

needs_linking (o)

Does this module need linking? Looks into subdirectory objects (see also has_link_code())

parse_abstract

parse a file and return what you think is the ABSTRACT

parse_version

```
my $version = MM->parse_version($file);
```

Parse a $file and return what $VERSION is set to by the first assignment. It will return the string "undef" if it can't figure out what $VERSION is. $VERSION should be for all to see, so `our $VERSION` or plain $VERSION are okay, but `my $VERSION` is not.

`<package Foo VERSION>` is also checked for. The first version declaration found is used, but this may change as it differs from how Perl does it.

parse_version() will try to `use version` before checking for `$VERSION` so the following will work.

```
$VERSION = qv(1.2.3);
```

pasthru (o)

Defines the string that is passed to recursive make calls in subdirectories.

perl_script

Takes one argument, a file name, and returns the file name, if the argument is likely to be a perl script. On MM_Unix this is true for any ordinary, readable file.

perldepend (o)

Defines the dependency from all *.h files that come with the perl distribution.

pm_to_blib

Defines target that copies all files in the hash PM to their destination and autosplits them. See *"DESCRIPTION" in ExtUtils::Install*

post_constants (o)

Returns an empty string per default. Dedicated to overrides from within Makefile.PL after all constants have been defined.

post_initialize (o)

Returns an empty string per default. Used in Makefile.PLs to add some chunk of text to the Makefile after the object is initialized.

postamble (o)

Returns an empty string. Can be used in Makefile.PLs to write some text to the Makefile at the end.

ppd

Defines target that creates a PPD (Perl Package Description) file for a binary distribution.

prefixify

```
  $MM->prefixify($var, $prefix, $new_prefix, $default);
```

Using either $MM->{uc $var} || $Config{lc $var}, it will attempt to replace it's $prefix with a $new_prefix.

Should the $prefix fail to match *AND* a PREFIX was given as an argument to WriteMakefile() it will set it to the $new_prefix + $default. This is for systems whose file layouts don't neatly fit into our ideas of prefixes.

This is for heuristics which attempt to create directory structures that mirror those of the installed perl.

For example:

```
    $MM->prefixify('installman1dir', '/usr', '/home/foo',
'man/man1');
```

this will attempt to remove '/usr' from the front of the $MM->{INSTALLMAN1DIR} path (initializing it to $Config{installman1dir} if necessary) and replace it with '/home/foo'. If this fails it will simply use '/home/foo/man/man1'.

processPL (o)

Defines targets to run *.PL files.

quote_paren

Backslashes parentheses `()` in command line arguments. Doesn't handle recursive Makefile `$(...)` constructs, but handles simple ones.

replace_manpage_separator

```
  my $man_name = $MM->replace_manpage_separator($file_path);
```

Takes the name of a package, which may be a nested package, in the form 'Foo/Bar.pm' and replaces the slash with `::` or something else safe for a man page file name. Returns the replacement.

cd

oneliner

quote_literal

escape_newlines

max_exec_len

Using POSIX::ARG_MAX. Otherwise falling back to 4096.

static (o)

Defines the static target.

static_lib (o)

Defines how to produce the *.a (or equivalent) files.

staticmake (o)

Calls makeaperl.

subdir_x (o)

Helper subroutine for subdirs

subdirs (o)

Defines targets to process subdirectories.

test (o)

Defines the test targets.

test_via_harness (override)

For some reason which I forget, Unix machines like to have PERL_DL_NONLAZY set for tests.

test_via_script (override)

Again, the PERL_DL_NONLAZY thing.

tool_xsubpp (o)

Determines typemaps, xsubpp version, prototype behaviour.

all_target

Build man pages, too

top_targets (o)

Defines the targets all, subdirs, config, and O_FILES

writedoc

Obsolete, deprecated method. Not used since Version 5.21.

xs_c (o)

Defines the suffix rules to compile XS files to C.

xs_cpp (o)

Defines the suffix rules to compile XS files to C++.

xs_o (o)

Defines suffix rules to go from XS to object files directly. This is only intended for broken make implementations.

SEE ALSO

ExtUtils::MakeMaker