mkfh()

Make a filehandle. Same kind of idea as Symbol::gensym().

__find_relocations

Works out what absolute paths in the configuration have been located at run time relative to $^X, and generates a regexp that matches them

NAME

ExtUtils::Packlist - manage .packlist files

SYNOPSIS

```
use ExtUtils::Packlist;
my ($pl) = ExtUtils::Packlist->new('.packlist');
$pl->read('/an/old/.packlist');
my @missing_files = $pl->validate();
$pl->write('/a/new/.packlist');

$pl->{'/some/file/name'}++;
   or
$pl->{'/some/other/file/name'} = { type => 'file',
                                   from => '/some/file' };
```

DESCRIPTION

ExtUtils::Packlist provides a standard way to manage .packlist files. Functions are provided to read and write .packlist files. The original .packlist format is a simple list of absolute pathnames, one per line. In addition, this package supports an extended format, where as well as a filename each line may contain a list of attributes in the form of a space separated list of key=value pairs. This is used by the installperl script to differentiate between files and links, for example.

USAGE

The hash reference returned by the new() function can be used to examine and modify the contents of the .packlist. Items may be added/deleted from the .packlist by modifying the hash. If the value associated with a hash key is a scalar, the entry written to the .packlist by any subsequent write() will be a simple filename. If the value is a hash, the entry written will be the filename followed by the key=value pairs from the hash. Reading back the .packlist will recreate the original entries.

FUNCTIONS

new()

This takes an optional parameter, the name of a .packlist. If the file exists, it will be opened and the contents of the file will be read. The new() method returns a reference to a hash. This hash holds an entry for each line in the .packlist. In the case of old-style .packlists, the value associated with each key is undef. In the case of new-style .packlists, the value associated with each key is a hash containing the key=value pairs following the filename in the .packlist.

read()

This takes an optional parameter, the name of the .packlist to be read. If no file is specified, the .packlist specified to new() will be read. If the .packlist does not exist, Carp::croak will be called.

write()

This takes an optional parameter, the name of the .packlist to be written. If no file is specified, the .packlist specified to new() will be overwritten.

validate()

This checks that every file listed in the .packlist actually exists. If an argument which evaluates to true is given, any missing files will be removed from the internal hash. The return value is a list of the missing files, which will be empty if they all exist.

packlist_file()

This returns the name of the associated .packlist file

EXAMPLE

Here's `modrm`, a little utility to cleanly remove an installed module.

```
    #!/usr/local/bin/perl -w

    use strict;
    use IO::Dir;
    use ExtUtils::Packlist;
    use ExtUtils::Installed;

    sub emptydir($) {
 my ($dir) = @_;
 my $dh = IO::Dir->new($dir) || return(0);
 my @count = $dh->read();
 $dh->close();
 return(@count == 2 ? 1 : 0);
    }

    # Find all the installed packages
    print("Finding all installed modules...\n");
    my $installed = ExtUtils::Installed->new();

    foreach my $module (grep(!/^Perl$/, $installed->modules())) {
       my $version = $installed->version($module) || "???";
       print("Found module $module Version $version\n");
       print("Do you want to delete $module? [n] ");
       my $r = <STDIN>; chomp($r);
       if ($r && $r =~ /^y/i) {
  # Remove all the files
  foreach my $file (sort($installed->files($module))) {
     print("rm $file\n");
     unlink($file);
  }
  my $pf = $installed->packlist($module)->packlist_file();
  print("rm $pf\n");
  unlink($pf);
  foreach my $dir (sort($installed->directory_tree($module))) {
     if (emptydir($dir)) {
 print("rmdir $dir\n");
 rmdir($dir);
     }
  }
      }
   }
```

AUTHOR

Alan Burlison <Alan.Burlison@uk.sun.com>