

WHERE-IS

This is a new implementation of a facility similar to but not compatible with the Lyric library module WhereIs. Where-Is indexes all definers, but WhereIs only indexed Interlisp FNS definitions.

Requirements

Hash-File and Cash-File.

Installation

Load `WHERE-IS.DFASL` and the required `.DFASL` modules from the library.

Changed File Manager Functions

Where-Is allows the file manager to know of many more definitions than are actually in the files which have been noticed. To achieve this behavior, the following file manager functions are changed when Where-Is is loaded.

Both of these functions are called by the edit interface (the function `cl:ed`). Thus when Where-Is is loaded the contents of its databases are known to the editor.

`(il:whereis name type files filter)` [Function]

Performs as described in the *Interlisp Reference Manual*. Returns the subset of *files* that contain a *type* definition for *name*. *Files* defaults to `il:filelst` (all noticed files). When Where-Is is loaded and `il:whereis` is passed `t` as its *files* argument, `il:whereis` looks in the Where-Is databases.

`(il:typesof name possible-types impossible-types source filter)` [Function]

Performs as described in the *Interlisp Reference Manual*. Returns the subset of *possible-types* that *name* is defined as. *Possible-types* defaults to `il:filepkgtypes` (all define types). When Where-Is is loaded, `il:typesof` also includes the types for *name* in its databases.

Databases

Where-Is provides functions to use and build databases.

Using a Database

`(xcl::add-where-is-database pathname)` [Function]

Adds the database in the file named by *pathname* to the databases known to Where-Is. If a database on an older version of this file is already known, then Where-Is will start using the new version.

(xcl::del-where-is-database *pathname*) [Function]

Deletes the database named by *pathname* from the databases known to Where-Is.

xcl::*where-is-cash-files* [Variable]

Contains the list of databases known to Where-Is.

There is a proceed case for errors while accessing a database which will delete the offending database. This can be useful when a file server goes down.

Building a Database

(xcl::where-is-notice *database-file* &key *files new hash-file-size temp-file-name define-types quiet*) [Function]

Records the definers on files in the file named by *database-file*.

Files can be a *pathname* or a list of *pathnames*. The default for *files* is "**. ;*". Note that it is important to include the trailing semicolon so that only definers on the most recent version are recorded.

If *new* is true, a new database file is created, otherwise *database-file* is presumed to name an existing Where-Is database to be augmented. The default for *new* is nil.

Hash-file-size is only used when *new* is false and is passed as the *size* argument to *make-hash-file*. The default for *hash-file-size* is `xcl::*where-is-hash-file-size*`, which has a default top-level value of 10,000.

If *temp-file-name* is provided, all changes happen in the temporary file named, which will afterwards be renamed to *database-file*. This can both make things faster (if the temporary file is on a faster device) and doesn't generate a new version of a database until the new version is ready to be used. The use of a temporary file may slow things down when a large existing database is just being updated to reflect a small number of changes.

Define-types is the list of define types (file package types) which should be recorded. The default define types are all those on `IL:FILEPKGTYPES` which are not aliases for others and which are not in the list `xcl::*where-is-ignore-define-types*`.

Unless *quiet* is true, `xcl::where-is-notice` prints the name of each file as it is processed.

`xcl::where-is-notice` returns the *pathname* of the hash file written.

[This page intentionally left blank]