NAME

lua scripts - lua scripts that are run by pkg(8)

DESCRIPTION

pkg(8) run scripts at given phases of the process it is running.

The phases are the following:

pre-install:
run before installing the files on the system.

post-install:
run after installing the files on the system.

pre-deinstall:
run before removing the files on the system.

post-deinstall:
run after removing the files on the system.

A package can contain multiple scripts per phase, they will all be run inside their own lua VM.

In the particular case of an upgrade the scripts are run in the following order:

1. new pre-install
2. old pre-deinstall
3. replace binaries
4. new post-install

Lua scripts are always run after shell scripts (of the same phase).

API

All the regular lua API are available to the exception of the following changes:

io.open()
has been modified to only open files relatively to the rootdir if specified by the *-r* argument passed to pkg(8).

os.remove()
has been modified to only remove files relatively to the rootdir if specified by the *-r* argument passed to pkg(8).

os.rename()
: has been modified to only rename files relatively to the rootdir if specified by the *-r* argument passed to pkg(8).

os.execute()
: has been disabled.

The following variables are available defined to any lua scripts:

pkg_name
: name of the package.

pkg_prefix
: *PREFIX* defined within the package at build time.

pkg_rootdir
: represents the root directory where the package will be installed as specified by the **-r** arguments passed to pkg(8).

pkg_upgrade
: Boolean to inform the scripts that it is running or not in the context of an upgrade

The following functions have been added:

out **pkg.prefixed_path**(*in*)
: prepend *pkg_prefix* to *in* if needed and returns it as *out*.

pkg.print_msg(*msg*)
: send messages to the user that will be shown at the end of the pkg(8) process.

pkg.filecmp(*file1, file2*)
: Compare 2 files, return *0* if the files are identical, *1* if the files are different and *>1* if an error occured

pkg.copy(*source, destination*)
: Copy a file preserving its attributes. return *-1* if an error occured

st **pkg.stat**(*file*)
: return an object table *st* with the following fields: *type, size, uid, gid*

pkg.symlink(*source*, *destination*)

Create a symbolic link *destination* pointing at *source*

pkg.exec(*arguments*)

Will execute the command *arguments* expected in the following form: ‘{command, arg1, arg2, arg3, ...}’

res **pkg.readdir**(*path*)

Will return an *ipair* with the list of elements contained in the directory the specicial directory ‘.’ and ‘..’ are be filtered out.

SEE ALSO

pkg_create(3), pkg_printf(3), pkg_repos(3), pkg-keywords(5), pkg-repository(5), pkg-script(5), pkg-triggers(5), pkg.conf(5), pkg(8), pkg-add(8), pkg-alias(8), pkg-annotate(8), pkg-audit(8), pkg-autoremove(8), pkg-check(8), pkg-clean(8), pkg-config(8), pkg-create(8), pkg-delete(8), pkg-fetch(8), pkg-info(8), pkg-install(8), pkg-lock(8), pkg-query(8), pkg-register(8), pkg-repo(8), pkg-rquery(8), pkg-search(8), pkg-set(8), pkg-shell(8), pkg-shlib(8), pkg-ssh(8), pkg-stats(8), pkg-triggers(8), pkg-update(8), pkg-updating(8), pkg-upgrade(8), pkg-version(8), pkg-which(8)