Initialization at Start of an R Session

Description

In R, the startup mechanism is as follows.

Unless `--no-environ` was given on the command line, R searches for site and user files to process for setting environment variables. The name of the site file is the one pointed to by the environment variable `R_ENVIRON`; if this is unset, '`R_HOME/etc/Renviron.site`' is used (if it exists, which it does not in a 'factory-fresh' installation). The name of the user file can be specified by the `R_ENVIRON_USER` environment variable; if this is unset, the files searched for are '`.Renviron`' in the current or in the user's home directory (in that order). See 'Details' for how the files are read.

Then R searches for the site-wide startup profile file of R code unless the command line option `--no-site-file` was given. The path of this file is taken from the value of the `R_PROFILE` environment variable (after tilde expansion). If this variable is unset, the default is '`R_HOME/etc/Rprofile.site`', which is used if it exists (it contains settings from the installer in a 'factory-fresh' installation). This code is sourced into the workspace (global environment). Users need to be careful not to unintentionally create objects in the workspace, and it is normally advisable to use `local` if code needs to be executed: see the examples. `.Library.site` may be assigned to and the assignment will effectively modify the value of the variable in the base namespace where `.libPaths()` finds it. One may also assign to `.First` and `.Last`, but assigning to other variables in the execution environment is not recommended and does not work in some older versions of R.

Then, unless `--no-init-file` was given, R searches for a user profile, a file of R code. The path of this file can be specified by the `R_PROFILE_USER` environment variable (and tilde expansion will be performed). If this is unset, a file called '`.Rprofile`' is searched for in the current directory or in the user's home directory (in that order). The user profile file is sourced into the workspace.

Note that when the site and user profile files are sourced only the **base** package is loaded, so objects in other packages need to be referred to by e.g. `utils::dump.frames` or after explicitly loading the package concerned.

R then loads a saved image of the user workspace from '`.RData`' in the current directory if there is one (unless `--no-restore-data` or `--no-restore` was specified on the command line).

Next, if a function `.First` is found on the search path, it is executed as `.First()`. Finally, function `.First.sys()` in the **base** package is run. This calls `require` to attach the default packages specified by `options("defaultPackages")`. If the **methods** package is included, this will have been attached earlier (by function `.OptRequireMethods()`) so that namespace initializations such as those from the user workspace will proceed correctly.

A function `.First` (and `.Last`) can be defined in appropriate '`.Rprofile`' or '`Rprofile.site`' files or have been saved in '`.RData`'. If you want a different set of packages than the default ones when you start, insert a call to `options` in the '`.Rprofile`' or '`Rprofile.site`' file. For example, `options(defaultPackages = character())` will attach no extra packages on startup (only the **base** package) (or set `R_DEFAULT_PACKAGES=NULL` as an environment variable before running R). Using `options(defaultPackages = "")` or `R_DEFAULT_PACKAGES=""` enforces the R *system* default.

On front-ends which support it, the commands history is read from the file specified by the environment variable `R_HISTFILE` (default '`.Rhistory`' in the current directory) unless `--no-restore-history` or `--no-restore` was specified.

The command-line option `--vanilla` implies `--no-site-file`, `--no-init-file`, `--no-environ` and (except for `R CMD`) `--no-restore` Under Windows, it also implies `--no-Rconsole`, which prevents loading the '`Rconsole`' file.

Details

Note that there are two sorts of files used in startup: *environment files* which contain lists of environment variables to be set, and *profile files* which contain R code.

Lines in a site or user environment file should be either comment lines starting with `#`, or lines of the form *`name`*`=`*`value`*. The latter sets the environmental variable *`name`* to *`value`*, overriding an existing value. If *`value`* contains an expression of the form `${foo-bar}`, the value is that of the environmental variable `foo` if that is set, otherwise `bar`. For `${foo:-bar}`, the value is that of `foo` if that is set to a non-empty value, otherwise `bar`. (If it is of the form `${foo}`, the default is `""`.) This construction can be nested, so `bar` can be of the same form (as in `${foo-${bar-blah}}`). Note that the braces are essential: for example `$HOME` will not be interpreted.

Leading and trailing white space in *`value`* are stripped. *`value`* is then processed in a similar way to a Unix shell: in particular (single or double) quotes not preceded by backslash are removed and backslashes are removed except inside such quotes.

For readability and future compatibility it is recommended to only use constructs that have the same behavior as in a Unix shell. Hence, expansions of variables should be in double quotes (e.g. `"${HOME}"`, in case they may contain a backslash) and literals including a backslash should be in single quotes. If a variable value may end in a backslash, such as `PATH` on Windows, it may be necessary to protect the following quote from it, e.g. `"${PATH}/"`. It is recommended to use forward slashes instead of backslashes. It is ok to mix text in single and double quotes, see examples below.

On systems with sub-architectures (mainly Windows), the files '`Renviron.site`' and '`Rprofile.site`' are looked for first in architecture-specific directories, e.g. '*`R_HOME`*`/etc/i386/Renviron.site`'. And e.g. '`.Renviron.i386`' will be used in preference to '`.Renviron`'.

There is a 100,000 byte limit on the length of a line (after expansions) in environment files (prior to R 4.1 it was 10,000).

Note

It is not intended that there be interaction with the user during startup code. Attempting to do so can crash the R process.

The startup options are for `Rgui`, `Rterm` and `R` but not for `Rcmd`: attempting to use e.g. `--vanilla` with the latter will give a warning or error.

Unix versions of R have a file '`R_HOME/etc/Renviron`' which is read very early in the start-up processing. It contains environment variables set by R in the configure process, and is not used on R for Windows.

`R CMD check` and `R CMD build` do not always read the standard startup files, but they do always read specific '`Renviron`' files. The location of these can be controlled by the environment variables `R_CHECK_ENVIRON` and `R_BUILD_ENVIRON`. If these are set their value is used as the path for the '`Renviron`' file; otherwise, files '`~/.R/check.Renviron`' or '`~/.R/build.Renviron`' or sub-architecture-specific versions are employed.

If you want '`~/.Renviron`' or '`~/.Rprofile`' to be ignored by child R processes (such as those run by `R CMD check` and `R CMD build`), set the appropriate environment variable `R_ENVIRON_USER` or `R_PROFILE_USER` to (if possible, which it is not on Windows) `""` or to the name of a non-existent file.

Prior to R 4.0.0, `${foo-bar}` in an environment file skipped an empty `foo`: this has been changed to match the POSIX rules for parameter substitution in shells.

See Also

For the definition of the 'home' directory on Windows see the '`rw-FAQ`' Q2.14. It can be found from a running R by `Sys.getenv("R_USER")`.

`.Last` for final actions at the close of an R session. `commandArgs` for accessing the command line arguments.

There are examples of using startup files to set defaults for graphics devices in the help for `windows.options`.

An Introduction to R for more command-line options: those affecting memory management are covered in the help file for Memory.

`readRenviron` to read '`.Renviron`' files.

For profiling code, see `Rprof`.

Examples

Run examples

```
## Not run:
## Example ~/.Renviron on Unix
R_LIBS=~/R/library
PAGER=/usr/local/bin/less

## Example .Renviron on Windows
R_LIBS=C:/R/library
MY_TCLTK="c:/Program Files/Tcl/bin"
```

```r
# Variable expansion in double quotes, string literals with backslashes in
# single quotes.
R_LIBS_USER="${APPDATA}"'\R-library'

## Example of setting R_DEFAULT_PACKAGES (from R CMD check)
R_DEFAULT_PACKAGES='utils,grDevices,graphics,stats'
# this loads the packages in the order given, so they appear on
# the search path in reverse order.

## Example of .Rprofile
options(width=65, digits=5)
options(show.signif.stars=FALSE)
setHook(packageEvent("grDevices", "onLoad"),
        function(...) grDevices::ps.options(horizontal=FALSE))
set.seed(1234)
.First <- function() cat("\n   Welcome to R!\n\n")
.Last <- function()  cat("\n   Goodbye!\n\n")

## Example of Rprofile.site
local({
  # add MASS to the default packages, set a CRAN mirror
  old <- getOption("defaultPackages"); r <- getOption("repos")
  r["CRAN"] <- "http://my.local.cran"
  options(defaultPackages = c(old, "MASS"), repos = r)
  ## (for Unix terminal users) set the width from COLUMNS if set
  cols <- Sys.getenv("COLUMNS")
  if(nzchar(cols)) options(width = as.integer(cols))
  # interactive sessions get a fortune cookie (needs fortunes package)
  if (interactive())
    fortunes::fortune()
})

## if .Renviron contains
FOOBAR="coo\bar"doh\ex"abc\"def'"

## then we get
# > cat(Sys.getenv("FOOBAR"), "\n")
# coo\bardoh\exabc"def'

## End(Not run)
```