The \pkg{pkgloader} package*

Michiel Helvensteijn
mhelvens+latex@gmail.com
April 29, 2018

Development of this package is organized at github.com/mhelvens/latex-pkgloader.
I am happy to receive feedback there!

1 Introduction

\LaTeX{} can be extended by loading packages using a \texttt{\usepackage}, \texttt{\RequirePackage} or \texttt{\RequirePackageWithOptions} command. Similarly, documents load a document class using \texttt{\documentclass}, \texttt{\LoadClass} or \texttt{\LoadClassWithOptions}. Packages and classes can add definitions, change existing ones, or otherwise extend functionality of the language.

While the Turing-complete power of the \TeX{} language is quite useful at times, it does make it all too easy for independent package authors to step on each others toes. CTAN is full of conceptually independent packages that cannot be loaded together, or break if they are not loaded in a specific order.

Yet, until now, there has been no automated package management to speak of. Document authors are usually told to avoid certain package combinations, or to load packages in some specific order. Occasionally, package authors patch their code to be aware of specific other packages, circumventing known conflicts. But this makes maintenance more difficult, because these package authors are ‘mixing concerns’; they put code related to other packages into their own package. And it is all done in an ad-hoc fashion.

Enter \pkg{pkgloader}.

1.1 Package Description

Here is an example of main file for a \LaTeX{} document which uses \pkg{pkgloader}:

\begin{verbatim}
\RequirePackage{pkgloader}
\documentclass{article}
\usepackage{algorithm}
\usepackage{hyperref}
\usepackage{float}
\LoadPackagesNow
\begin{document} ...
\end{document}
\end{verbatim}

*This document corresponds to \pkg{pkgloader} v0.7.0, dated 2018/04/29.
The idea is to load \pkgloader before loading any other package or class. It can then intercept all loading requests, analyze them and load them properly, taking this burden off the shoulders of the average document author.

Between the second and fifth line, the loading of all packages and classes is postponed. The \LoadPackagesNow command then loads the packages in some valid order. This also happens automatically upon reaching the end of the preamble. During this process, ‘conflict resolving’ code may also be run, meant specifically to make other packages work together properly. If the above code were compiled without \pkgloader, the given package order would cause errors. The main advantage to this approach is that the complexity of dealing with package conflicts is moved to the \pkgloader package and handled in a systematic manner.

If you are a document author, this may be all you need to know to use \pkgloader. If you are interested in more advanced functionality, read on!

\section{The \pkgloader Area}

\RequirePackage {\pkgloader} ... \LoadPackagesNow

The \pkgloader area is the area between \RequirePackage{\pkgloader} and \LoadPackagesNow. Within it, the three traditional package-loading commands are ‘hijacked’, recording information rather than loading packages directly. Also, a \Load command is available, which offers more flexibility in regulating package loading behavior.

\pkgloader accepts sets of rules coming from outside packages, though support is still somewhat limited. Any \pkgloader-{something}.sty file can be loaded within the \pkgloader area by passing \textit{something} as a package option. The file \pkgloader-recommended.sty is loaded this way by default. You can overwrite this by passing recommended=false as an option. For example:

\begin{verbatim}
1 \RequirePackage[recommended=false, my-better-rules]{\pkgloader}
2 ... 
3 \LoadPackagesNow
\end{verbatim}

This area does not play by the recommended package loading rules, but uses the rules in \pkgloader-my-better-rules.sty instead.

\section{Package / Class Loading Requests}

\usepackage \RequirePackage \RequirePackageWithOptions \documentclass \LoadClass \LoadClassWithOptions

For requesting specific packages or classes inside the \pkgloader area, just use the commands always used for this purpose: \usepackage, \RequirePackage and \RequirePackageWithOptions, as well as \documentclass, \LoadClass and \LoadClassWithOptions. Their syntax and effective semantics are the same as they have always been. Their effects are just delayed, reordered and perhaps modified by the active package loading rules.

\section{Package Loading Rules}

\Load

Each invocation if the \Load command sets up a rule about a class, package or packages, which are not necessarily ever loaded. These rules can come from any number of different sources. A central registry will be maintained together with \pkgloader itself in the form of \pkgloader-recommended.sty, specifying well-known conflicts.
and resolutions. Individual package authors, however, can supply their own rules, as can
document authors. Though ideally, for the average document author, things should ‘just
work’.

The \Load command expects the following syntax:

\begin{verbatim}
⟨load⟩ ::= \Load ⟨package⟩ ⟨clause⟩₁ ... ⟨clause⟩ᵢ
| \Load class ⟨package⟩ ⟨clause⟩₁ ... ⟨clause⟩ᵢ
| \Load error ⟨clause⟩₁ ... ⟨clause⟩ᵢ
⟨package⟩ ::= ⟨options⟩ [ ⟨pkg⟩ ] [ ⟨version⟩ ]
⟨clause⟩ ::= ⟨order⟩ | ⟨condition⟩ | ⟨reason⟩
⟨order⟩ ::= before { ⟨pkg⟩₁,...,⟨pkg⟩ᵢ } 
| after { ⟨pkg⟩₁,...,⟨pkg⟩ᵢ } 
| early 
| late
⟨condition⟩ ::= always | if loaded | if { ⟨ϕ⟩ }
⟨ϕ⟩ ::= ⟨pkg⟩ | ⟨ϕ⟩ & ⟨ϕ⟩ | ⟨ϕ⟩ || ⟨ϕ⟩ | ! ⟨ϕ⟩ | ( ⟨ϕ⟩ )
⟨reason⟩ ::= because { ⟨token−list⟩ }
\end{verbatim}

⟨pkg⟩ represents a package or class name. ⟨token−list⟩ should expand to a human-readable
text without formatting.

if The ⟨condition⟩ clause determines under which package loading conditions any and all
parts of a rule are invoked. Here is an example of the use of the ⟨condition⟩ clause:

\begin{verbatim}
1 \Load { res-ie-lst } if { inputenc & listings }
2 \Load { fixltx2e } always
\end{verbatim}

res-ie-lst (a fictional package built specifically to resolve the conflict between
inputenc and listings) will be loaded if requested specifically, or if both inputenc
and listings are loaded. The fixltx2e package is always loaded, as it was created
to smooth over some mistakes in the \LaTeX core.

always The always keyword makes a rule unconditional. The if loaded directive makes
a rule conditional on its package already being loaded anyway. This can be used to order
two packages only when they are being loaded by other means, and is actually the default
behavior (in other words, if loaded really does nothing).

if loaded

before The before and after keywords should be pretty straightforward. They can be
used for things like:

\begin{verbatim}
1 \Load { xitxtra } after { graphicx } 
\end{verbatim}

which fixes the loading order between these two packages when they are both loaded.

early But the set of \LaTeX packages is constantly growing, and it appears that some big
packages should almost always be loaded early in the process, and others should almost
always be loaded late. Therefore the early and late stages are provided as a fallback
mechanism. If two packages are not related by an explicit application order, their loading
order may still be decided by their relative stages: early before ‘normal’ before late.

That way, conflicts may be avoided in a majority of cases. This is implemented with
pkgloader-early and pkgloader-late stubs in the loading order graph.

The following example uses the ⟨order⟩ clause in addition to the ⟨condition⟩ clause:
\Load {res-ie-lst} if \{inputenc \&\& listings\}
\after {inputenc , listings}
\Load {fixltx2e} always early

An important observation about the loading order is that it might form a cycle when contradictory ordering rules are specified:

\Load {pkg1} after \{pkg2\}
\Load {pkg2} after \{pkg1\}

In practice this could happen if the authors of pkg1 and pkg2 independently discover a conflict, and both try to solve it by patching their code and having their own package be loaded last. pkgloader can provide a clear error message when this happens, allowing the two package authors to seek contact and collaborate on a solution.

Now, about the \texttt{error} keyword. Initially all package combinations are valid. But if two packages are irredeemably incompatible, their combination can be made to trigger an error message by a command such as the following:

\Load error if \{algorithms \&\& pseudocode\}

These two packages provide almost identical functionality and conflict on many command names. It was generally agreed upon that they should never be loaded together. Document authors should simply choose one or the other.

Finally, the \texttt{⟨reason⟩} clause can be used to supply a human-readable explanation of a rule. We finish the above examples by providing reasons:

\Load {res-ie-lst} if \{inputenc \&\& listings\}
\after {inputenc , listings}
\because {it allows the use of 1 byte unicode characters in code listings}
\Load {fixltx2e} always early
\because {it fixes some imperfections in LaTeX2e}
\Load error when \{algorithms \&\& pseudocode\}
\because {they provide almost identical functionality and conflict on many command names}

In the future, reasons will be extracted automatically to generate documentation. For now, they are displayed with error messages related to the rule in question.

\subsection*{1.5 Status}

So far, pkgloader seems to work as expected, but has not yet been tested as extensively as it should be. Therefore, bug-reports on the pkgloader issue tracker on Github would be most welcome. Also, lots more recommended package loading rules are needed.

I nonetheless decided to publish the package now, because I've been promising to do so for a while now:

http://tex.stackexchange.com/questions/123174

I hope that, with feedback and community collaboration, use of this package will become widespread and package authors will be able to work in a more modular fashion.
Future versions of pkgloader will be able to intelligently merge package options and to track packages loaded by other packages in order to better inform the user — perhaps even fix problems by carrying information into the next run through auxiliary files. But most of this will depend on feedback.
2 Implementation

We now show and explain the entire implementation from `pkgloader.sty`.

2.1 Package Info

First, the mandatory package meta-information:

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\ProvidesExplPackage{pkgloader}{2018/04/29}{0.7.0}
\{managing the options and loading order of LaTeX packages\}
```

2.2 Required Packages

The following packages are required. Two standard expl3-related packages, one experimental package in l3regex and one user-contributed expl3 package in lt3graph:

```
\RequirePackage{xparse}
\RequirePackage{l3keys2e}
\RequirePackage{l3regex}
\RequirePackage{lt3graph}
```

2.3 Package Code

We need two global data-structures. One to keep track of all packages that are known, one to keep track of the packages that are actually going to be loaded, and their order:

```
\prop_new:N \g__pkgloader_known_pkg_prop
\graph_new:N \g__pkgloader_pkg_graph
```

We store pristine versions of the three package loading commands:

```
\cs_gset_eq:NN \__pkgloader_usepkg:wnw \usepackage
\cs_gset_eq:NN \__pkgloader_RPkg:wnw \RequirePackage
\cs_gset_eq:NN \__pkgloader_RPkgWithOptions:wnw \RequirePackageWithOptions
\cs_gset_eq:NN \__pkgloader_doccls:wnw \documentclass
\cs_gset_eq:NN \__pkgloader_LCls:wnw \LoadClass
\cs_gset_eq:NN \__pkgloader_LClsWithOptions:wnw \LoadClassWithOptions
```

And we define a command to clean up any and all commands that we change or introduce. It will be called when they are not needed anymore:

```
\tl_new:N \__pkgloader_cleanup_commands:
\tl_put_right:Nn \__pkgloader_cleanup_commands: {
\cs_gset_eq:NN \usepackage \__pkgloader_usepkg:wnw
\cs_gset_eq:NN \RequirePackage \__pkgloader_RPkg:wnw
\cs_gset_eq:NN \RequirePackageWithOptions \__pkgloader_RPkgWithOptions:wnw
\cs_gset_eq:NN \documentclass \__pkgloader_doccls:wnw
\cs_gset_eq:NN \LoadClass \__pkgloader_LCls:wnw
\cs_gset_eq:NN \LoadClassWithOptions \__pkgloader_LClsWithOptions:wnw
}
```
This function globally registers a package loading rule, which can be created with either
the \Load command or any of the hijacked \usepackage-like commands:

\cs_new_protected:Nn \__pkgloader_register_rule:nnnnnnnnn { %
\prop_if_in:NnF \g__pkgloader_known_pkg_prop {#1} { %
\prop_gput:Nnn \g__pkgloader_known_pkg_prop {#1} {}
\int_new:c {g__pkgloader_count_(#1)_int}
}\int_incr:c {g__pkgloader_count_(#1)_int}
\tl_set:Nf \l_tmpa_tl {\int_use:c {g__pkgloader_count_(#1)_int}}
\tl_set:cn {g__pkgloader_options_ (#1)_(\l_tmpa_tl)_tl} {#2}
\tl_set:cn {g__pkgloader_version_ (#1)_(\l_tmpa_tl)_tl} {#3}
\tl_set:cn {g__pkgloader_condition_ (#1)_(\l_tmpa_tl)_tl} {#4}
\tl_set:cn {g__pkgloader_compiled_condition_(#1)_(\l_tmpa_tl)_tl} {#5}
\tl_set:cn {g__pkgloader_predecessors_ (#1)_(\l_tmpa_tl)_tl} {#6}
\tl_set:cn {g__pkgloader_successors_ (#1)_(\l_tmpa_tl)_tl} {#7}
\tl_set:cn {g__pkgloader_command_ (#1)_(\l_tmpa_tl)_tl} {#8}
\bool_new:c {g__pkgloader_used_ (#1)_(\l_tmpa_tl)_bool}
}

These six macros are redefined to just register the loading information rather than load
the package or class immediately. The distinction between package and class is made by
prefixing the name with either .sty or .cls (which will be stripped off before the file is
actually loaded):

\RenewDocumentCommand \usepackage { o m o }
\{ \__pkgloader_usepackage_cmd:nnnnnnnn
\usepackage {#1} \{#2.sty\} \{#3\}
\}
\RenewDocumentCommand \RequirePackage { o m o }
\{ \__pkgloader_usepackage_cmd:nnnnnnnn
\RequirePackage {#1} \{#2.sty\} \{#3\}
}
\RenewDocumentCommand \RequirePackageWithOptions { o m o }{\__pkgloader_usepackage_cmd:nnnnnn}{\RequirePackageWithOptions} {#1} {#2.sty} {#3}{\__pkgloader_usepackage_cmd:nnnnnn}{\documentclass} {#1} {#2.cls} {#3}{\__pkgloader_usepackage_cmd:nnnnnn}{\LoadClass} {#1} {#2.cls} {#3}{\__pkgloader_usepackage_cmd:nnnnnn}{\LoadClassWithOptions} {#1} {#2.cls} {#3}{\__pkgloader_usepackage_cmd:nnnnnn}{pkgloader-cls-pkg.sty}{it~is~requested~by~the~author}{#1}{pkgloader-cls-pkg.sty}

Storing this information is delegated to the \_\_\_pkgloader_register_rule:nnnnnnnnn function:

\cs_new:Nn \__pkgloader_usepackage_cmd:nnnnnnn {
\__pkgloader_register_rule:nnnnnnnnn
{#3} {#2} {#4} % package name, options, version
{pkgloader-true.sty} % condition
{\c_true_bool} % compiled condition
{#5} {#6} % predecessors, successors
{it~is~requested~by~the~author} % reason
{#1} % command
}

This is a sophisticated user-level command for manipulating package loading order and conditions. It has a ‘non-standard’ but convenient syntax, which scans for clauses rather than taking standard parameters:

\NewDocumentCommand \Load {} {

Initialize the variables used for storing given data:

\tl_clear:N \l__pkgloader_load_extension_tl
\tl_clear:N \l__pkgloader_load_options_tl
\tl_clear:N \l__pkgloader_load_name_tl
\tl_clear:N \l__pkgloader_load_version_tl
\clist_clear:N \l__pkgloader_load_pred_clist
\clist_clear:N \l__pkgloader_load_succ_clist
\tl_clear:N \l__pkgloader_load_cond_tl
\tl_clear:N \l__pkgloader_load_because_tl
\tl_clear:N \l__pkgloader_load_cmd_tl
\bool_set_false:N \l__pkgloader_early_late_used_bool

Start scanning for input:
This function checks if the \texttt{class} keyword is given.

\begin{verbatim}
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_ext_:w {
  \peek_charcode_remove_ignore_spaces:NTF c {% % % class
    \__pkgloader_load_scan_ext_c:w
  }{% % % % % % % % % % % % % % % % % % % % % % % package details
    \tl_set:Nn \l__pkgloader_load_extension_tl {.sty}
    \tl_set:Nn \l__pkgloader_load_cmd_tl {\RequirePackage}
    \__pkgloader_load_scan_pkg_:w
}
\end{verbatim}

The \texttt{class} keyword indicates that this is a document class loading rule, rather than a package loading rule. We record this and then goes on to scan the details:

\begin{verbatim}
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_ext_c:w lass {
  \tl_set:Nn \l__pkgloader_load_extension_tl {.cls}
  \tl_set:Nn \l__pkgloader_load_cmd_tl {\LoadClass}
  \clist_put_right:Nn \l__pkgloader_load_succ_clist {pkgloader-cls-pkg.sty}
  \__pkgloader_load_scan_pkg_:w
}
\end{verbatim}

The following function starts scanning for the name of the package central to this rule, as well as the options and minimum version proposed for it. It also checks if the \texttt{error} keyword is given.

\begin{verbatim}
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_pkg_:w {
  \peek_charcode_remove_ignore_spaces:NTF e {% % % error
    \__pkgloader_load_scan_pkg_e:w
  }{% % package options
    \__pkgloader_load_scan_pkg_options:nw
  }{% % % % % % % % % % % % % % % % % % % % % % % package name
    \__pkgloader_load_scan_pkg_:nw
}
\end{verbatim}

The \texttt{error} keyword can take the place of a package name, options and version. It is shorthand for the \texttt{pkgloader-error} file, and then jumps ahead to scanning clauses:

\begin{verbatim}
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_pkg_e:w rror {
  \tl_set:Nn \l__pkgloader_load_name_tl {pkgloader-error.sty}
  \__pkgloader_load_scan_clause_:w
}
\end{verbatim}

This scans the options and goes ahead to scan the package name:
This scans the package name (and adds the proper extension), peeks ahead for a minimum version, and otherwise goes on to scanning for clauses:

\cs_new_protected_nopar:Npn \__pkgloader_load_scan_pkg_options:nw #1 \} {
\tl_set:Nn \l__pkgloader_load_options_tl {#1}
\__pkgloader_load_scan_pkg_:nw
}

This scans the version, and then goes ahead to scan for clauses:

\cs_new_protected_nopar:Npn \__pkgloader_load_scan_version:nw #1 \} {
\tl_set:Nn \l__pkgloader_load_version_tl {#1}
\__pkgloader_load_scan_clause_:w
}

This is the start- and return-point used to scan for (additional) \Load clauses:

\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause:w \{ %
\peek_charcode_remove_ignore_spaces:NTF a \{ %
\peek_charcode_remove:NTF l \{ % % % % % % % always
\__pkgloader_load_scan_clause_al:w
\}\{\peek_charcode_remove:NTF f \{ % % % % % % % after
\__pkgloader_load_scan_clause_af:w
\}\{\__pkgloader_load_end: a
\}
\}
\}\{\peek_charcode_remove_ignore_spaces:NTF b \{ %
\peek_charcode_remove:NTF e \{ %
\peek_charcode_remove:NTF c \{ % % % % % % % because
\__pkgloader_load_scan_clause_bec:w
\}\{\peek_charcode_remove:NTF f \{ % % % % % % % before
\__pkgloader_load_scan_clause_bef:w
\}\{\__pkgloader_load_end: be
\}
\}
\}\{\__pkgloader_load_end: b
\}
\}\{\peek_charcode_remove_ignore_spaces:NTF e \{ % % % early
\__pkgloader_load_scan_clause_e:w
\}\{\peek_charcode_remove_ignore_spaces:NTF i \{ % % % if
\__pkgloader_load_scan_clause_i:w
\}
This processes the “always” clause, which loads this package conditional on the “pkgloader-true” package being loaded (which always is):

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_al:w ways {
\tl_put_right:Nn \l__pkgloader_load_cond_tl {~||~pkgloader-true}
\__pkgloader_load_scan_clause_:w
}
```

This processes the “after” clause:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_af:w ter #1 {
\clist_map_inline:nn {#1} {
\clist_put_right:Nn \l__pkgloader_load_pred_clist {##1.sty}
}
\__pkgloader_load_scan_clause_:w
}
```

This processes the “because” clause:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_bec:w ause #1 {
\tl_set:Nn \l__pkgloader_load_because_tl {#1}
\__pkgloader_load_scan_clause_:w
}
```

This processes the “before” clause:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_bef:w ore #1 {
\clist_map_inline:nn {#1} {
\clist_put_right:Nn \l__pkgloader_load_succ_clist {##1.sty}
}
\__pkgloader_load_scan_clause_:w
}
```

This processes the “early” clause, which orders this package before the “pkgloader-early” stub:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_e:w arly {
\bool_set_true:N \l__pkgloader_early_late_used_bool
\clist_put_right:Nn \l__pkgloader_load_succ_clist {pkgloader-early.sty}
\__pkgloader_load_scan_clause_:w
}
```

This processes the “if” clause, which may still be a manual condition or the “loaded” keyword:
This processes the “if loaded” clause, which uses this package being loaded as the condition for the rule being used:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_if_l:woaded {
  \tl_put_right:Nn \l__pkgloader_load_cond_tl {~||~}
  \tl_put_right:NV \l__pkgloader_load_cond_tl \l__pkgloader_load_name_tl
  \__pkgloader_load_scan_clause_:w
}
```

This processes the “if” clause with a manual condition:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_if_:nw #1 {
  \tl_put_right:Nn \l__pkgloader_load_cond_tl {~||~#1}
  \__pkgloader_load_scan_clause_:w
}
```

This processes the “late” clause, which orders this package after the “pkgloader-late” stub:

```
\cs_new_protected_nopar:Npn \__pkgloader_load_scan_clause_l:w ate {
  \bool_set_true:N \l__pkgloader_early_late_used_bool
  \clist_put_right:Nn \l__pkgloader_load_pred_clist {pkgloader-late.sty}
  \__pkgloader_load_scan_clause_:w
}
```

This function processes the collected data and registers it:

```
\cs_new_protected_nopar:Nn \__pkgloader_load_end: { }
```

We remove the leading “||” from the condition:

```
\tl_remove_once:Nn \l__pkgloader_load_cond_tl {~||~}
```

If no condition is given at all, the default is: “if loaded”

```
\tl_if_empty:NT \l__pkgloader_load_cond_tl
  { \tl_set_eq:NN \l__pkgloader_load_cond_tl \l__pkgloader_load_name_tl }
```

We now take the condition and compile it to a `\bool_if:` kind of syntax. The original syntax is preserved to use in error messages and such:
If no reason was given for this rule, it was obviously ‘because of reasons’:

```
\tl_if_empty:NT \l__pkgloader_load_because_tl
{ \tl_set:Nn \l__pkgloader_load_because_tl {of~reasons} }
```

Having gathered and processed the data, the rule is registered:

```
\__pkgloader_register_rule:VVVVVVVVV
\l__pkgloader_load_name_tl
\l__pkgloader_load_options_tl
\l__pkgloader_load_version_tl
\l__pkgloader_load_cond_tl
\l__pkgloader_load_compd_cond_tl
\l__pkgloader_load_pred_clist
\l__pkgloader_load_succ_clist
\l__pkgloader_load_because_tl
\l__pkgloader_load_cmd_tl
```

And here’s the instruction to clean up the \Load command-name at the end:

```
\tl_put_right:Nn \__pkgloader_cleanup_commands:
{ \cs_undefine:N \Load }
```

This function decides, based on all loaded rules and package requests, which packages, options and versions end up being loaded, and in which order.
We first set up a graph to record accepted orderings:

\begin{verbatim}
\cs_new_protected:Nn \__pkgloader_select_packages: {

We then start a loop that runs at least once, then repeats while additional package configurations are still being added to the set. Eventually the loop reaches a fixed point and terminates.

\begin{verbatim}
\bool_do_while:Nn \l__pkgloader_selection_changed_bool {
\bool_set_false:N \l__pkgloader_selection_changed_bool

Then first, for all possible package configurations (a nested loop, but not doubly indented because it feels like one loop):

\begin{verbatim}
\prop_map_inline:Nn \g__pkgloader_known_pkg_prop {\ldots} {\ldots} {\ldots}
\int_step_inline:nncn {1} {1} {g__pkgloader_count_(##1)_int} {\ldots}
\end{verbatim}

If the current configuration should be loaded but still isn’t selected (nested conditional; but again, not indented):

\begin{verbatim}
\bool_if:cF {g__pkgloader_used_(##1)_(####1)_bool} {
\bool_if:vT {g__pkgloader_compiled_condition_(##1)_(####1)_tl} {

We mark the package configuration as being used:

\begin{verbatim}
\bool_set_true:c {g__pkgloader_used_(##1)_(####1)_bool}
\end{verbatim}

We record the configuration in the main package graph, which maps each package to a clist of selected configurations:

\begin{verbatim}
\graph_get_vertex:NnNTF \g__pkgloader_pkg_graph {##1}
\l__pkgloader_used_configs_tl {
\tl_if_empty:NTF \l__pkgloader_used_configs_tl {
\graph_gput_vertex:Nnf \g__pkgloader_pkg_graph {##1} {####1}
\} {
\graph_gput_vertex:Nnf \g__pkgloader_pkg_graph {##1}
\l__pkgloader_used_configs_tl, ####1}
\}
\end{verbatim}

In a separate graph, we record the associated (now activated) package loading orders. We don’t do this in the main graph, because it may involve packages that are themselves not yet selected. These edges are later filtered and added to the main graph:

\begin{verbatim}
\graph_put_vertex:Nn \l__pkgloader_order_graph {##1}
\clist_map_inline:cn {g__pkgloader_predecessors_(##1)_(####1)_tl} {
\graph_put_vertex:Nn \l__pkgloader_order_graph {####1}
\graph_get_edge:NnnNTF \l__pkgloader_order_graph {####1} {##1} \l__pkgloader_used_configs_tl {
\graph_put_edge:Nnnn \l__pkgloader_order_graph {####1} {##1} \l__pkgloader_used_configs_tl, ####1}
\graph_put_edge:Nnnn \l__pkgloader_order_graph {####1} {##1} {####1}
\}
\end{verbatim}

14
\clist_map_inline:cn {g__pkgloader_successors_(##1)_tl} { 
  \graph_put_vertex:Nn \l__pkgloader_order_graph {####1} 
  \graph_get_edge:NnnNTF \l__pkgloader_order_graph 
  {##1} {####1} \l__pkgloader_used_configs_tl 
  \graph_put_edge:Nnnn \l__pkgloader_order_graph 
  {##1} {####1} \l__pkgloader_used_configs_tl,####1 
  \graph_put_edge:Nnnn \l__pkgloader_order_graph 
  {##1} {####1} 
}

We then mark the change, so a next iteration will be entered:

\bool_set_true:N \l__pkgloader_selection_changed_bool

}

}
}
}
}
}

We put the applicable proposed orderings into the graph of selected packages:

\graph_gput_edges_from:NN \g__pkgloader_pkg_graph \l__pkgloader_order_graph

If there is a cycle in the derived package loading order: ERROR

\graph_if_cyclic:NT \g__pkgloader_pkg_graph 
  \msg_fatal:nn {pkgloader} {cyclic-order}

Finally, we apply some default orderings where needed:

- If a package should not specifically go early or late, it goes inbetween; and
- if a package should not specifically go before a class, it goes after.

\graph_map_vertices_inline:Nn \g__pkgloader_pkg_graph { 
  \seq_if_in:NnF \g__pkgloader_system_packages_seq {##1} { 
    \graph_acyclic_if_path_exist:NnnF \g__pkgloader_pkg_graph 
    {pkgloader-early.sty} {##1} 
    \graph_acyclic_if_path_exist:NnnF \g__pkgloader_pkg_graph 
    {pkgloader-late.sty} {##1} 
    \graph_put_edge:Nnn \g__pkgloader_pkg_graph {pkgloader-early.sty} {##1} 
    \graph_put_edge:Nnn \g__pkgloader_pkg_graph {##1} {pkgloader-late.sty} 
  } 
  \graph_acyclic_if_path_exist:NnnF \g__pkgloader_pkg_graph 
  {pkgloader-cls-pkg.sty} {##1} {pkgloader-cls-pkg.sty} 
  \graph_put_edge:Nnn \g__pkgloader_pkg_graph {pkgloader-cls-pkg.sty} {##1} 
  \graph_put_edge:Nnn \g__pkgloader_pkg_graph {pkgloader-cls-pkg.sty} {##1} 
}

\cs_generate_variant:Nn \int_step_inline:nnnn {nncn} 
\cs_generate_variant:Nn \bool_if:nT {vT} 
\cs_generate_variant:Nn \withargs:nnn {vvn} 
\cs_generate_variant:Nn \graph_gput_vertex:Nnn {Nnf} 
\graph_new:N \l__pkgloader_order_graph 
\tl_new:N \l__pkgloader_used_configs_tl
Now follows the user command to consolidate all package loading requests and do the ‘right thing’ (tm). Invoking this command ends the work of pkgloader.

We first select package configurations by their loading conditions:

Now to clean up after pkgloader, restoring and removing various command-names.

Then, for all used packages, in topological order...

...load that package. Though note that this code is still quite incomplete, because it loads the first viable configuration. It should:

1. use theWithOptions version of the command if necessary,
2. allow custom merging schemes for options, and
3. use the latest required version.

And it needs the following auxiliary function to strip filenames from their four character extension:
It’s a bit clunky. Is there a substring function in \texttt{expl3} we could use that I don’t know about?

And here’s the instruction to clean up the \texttt{\LoadPackagesNow} command-name at the end:

\begin{verbatim}
\tl_put_right:Nn \__pkgloader_cleanup_commands: { \cs:_undefine:N \LoadPackagesNow }
\end{verbatim}

Bootstrap \texttt{pkgloader} by inserting \texttt{pkgloader-true} in the graph directly, so all other packages can be inserted with rules, possibly using the \texttt{always} clause.

\begin{verbatim}
\graph_gput_vertex:Nn \g__pkgloader_pkg_graph {pkgloader-true.sty}
\end{verbatim}

We keep a list of all \texttt{pkgloader} dummy packages:

\begin{verbatim}
\seq_new:N \_\_pkgloader_system_packages_seq
\cs_generate_variant:Nn \_\_pkgloader_begin_env:n \begin
\tl_if_eq:nnT {#1} {document} {\LoadPackagesNow}
\_\_pkgloader_begin_env:n (#1)
}
\tl_put_right:Nn \__pkgloader_cleanup_commands: { \cs_gset_eq:NN \begin \_\_pkgloader_begin_env:n }
\seq_gset_from_clist:Nx \_\_pkgloader_system_packages_seq {\tl_to_str:n {pkgloader-true.sty, pkgloader-false.sty, pkgloader-early.sty, pkgloader-late.sty, pkgloader-cls-pkg.sty}}
\end{verbatim}

We then register the core logical rules of \texttt{pkgloader}, regarding fundamental package ‘stubs’ like \texttt{pkgloader-false}, \texttt{pkgloader-error}, \texttt{pkgloader-early}, and so on.

\begin{verbatim}
\withargs:nn {of the mandatory core rules of \pkgloader} { \Load error if \texttt{\pkgloader-false} because \texttt{#1} \Load \texttt{\pkgloader-true} always because \texttt{#1} \Load \texttt{\pkgloader-early} always because \texttt{#1} \Load \texttt{\pkgloader-late} always because \texttt{#1} }
\end{verbatim}
We process the options passed to \pkgloader as .sty files to be loaded before \pkgloader does its thing. This should be used to define new \pkgloader rules. Note, particularly, that any \usepackage-like command inside those .sty files is registered and processed by \pkgloader; not loaded directly.

First, we define the functions used to handle an option.

The recommended rules are loaded unless explicitly turned off.

Process the options to populate \l__pkgloader_rule_packages_seq.

Actually load the .sty files in \l__pkgloader_rule_packages_seq. Note that the actual file needs the \pkgloader- prefix.

Finally, we make a show of using the proper macros for \LaTeX's benefit. If we don’t, a \LaTeX error is issued.

Finally, here are the error messages this package can generate. First a simple error for cycles, which should be improved to show the cause of the cycle.

And the following is the error reported for certain package combinations that have been forbidden through an error rule.
\msg_new:nnnn {pkgloader} {illegal-combination}

\{ A \"combination\" of \"packages\" fitting \the \"following\" condition \}
\\\
\\\#1\\
\\\\
This is an error because \#2. \}
\{ A pkgloader rule was requested that prohibits the logical combination \above \for \the \"specified\" reason. \"It is probably a \"good\" reason. \} 

Change History

0.1.0
General: initial version \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.2.0
General: an important release, which implements most features described in the pkgloader TUGBoat article, and fixes a ton of bugs \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.3.0
General: fixed the package to work with more recent releases of expl3, which removed the individual \kernel .sty files; additionally, pkgloader now understands preferred loading orders of cleveref

0.4.1
General: integrates a number of new package loading rules contributed by the community \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.5.0
General: integrates a number of new package loading rules contributed by the community \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.5.1
General: fixed the package to work after _\_new:.Nu was changed to only allow 'N' and 'n' in function signatures (2016/08/20) \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.6.0
General: integrates a number of new package loading rules contributed by the community \ldots \ldots \ldots \ldots \ldots \ldots . 1

0.7.0
General: integrates a number of new package loading rules contributed by the community \ldots \ldots \ldots \ldots \ldots \ldots . 1

Index
The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>228</td>
</tr>
<tr>
<td>&amp;</td>
<td>228</td>
</tr>
<tr>
<td>(</td>
<td>228</td>
</tr>
<tr>
<td>)</td>
<td>228</td>
</tr>
<tr>
<td>.</td>
<td>230</td>
</tr>
<tr>
<td>|</td>
<td>405, 408</td>
</tr>
<tr>
<td>{</td>
<td>230</td>
</tr>
<tr>
<td>}</td>
<td>230</td>
</tr>
</tbody>
</table>

Numbers

<table>
<thead>
<tr>
<th>Number</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\0</td>
<td>230</td>
</tr>
<tr>
<td>\405</td>
<td>405</td>
</tr>
<tr>
<td>\begi</td>
<td>357, 358, 363</td>
</tr>
<tr>
<td>bool commands: \begin \begin</td>
<td>264</td>
</tr>
<tr>
<td>bool_do_while:Nn</td>
<td>268</td>
</tr>
<tr>
<td>bool_if:NTF</td>
<td>268</td>
</tr>
<tr>
<td>Command</td>
<td>Pages</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>\seq_gset_from_clist:Nn</td>
<td>366, 367</td>
</tr>
<tr>
<td>\seq_if_in:NnTF</td>
<td>305</td>
</tr>
<tr>
<td>\seq_map_inline:Nn</td>
<td>396</td>
</tr>
<tr>
<td>\seq_new:N</td>
<td>365, 388</td>
</tr>
<tr>
<td>\seq_put_right:Nn</td>
<td>386, 389</td>
</tr>
<tr>
<td>\seq_remove_all:Nn</td>
<td>387</td>
</tr>
<tr>
<td>\seq_remove_duplicates:N</td>
<td>395</td>
</tr>
<tr>
<td>\tl_clear:N</td>
<td>86, 87, 88, 89, 92, 93, 94</td>
</tr>
<tr>
<td>\tl_if_empty:NTF</td>
<td>224, 232, 273</td>
</tr>
<tr>
<td>\tl_if_eq:nnTF</td>
<td>245, 359, 385</td>
</tr>
<tr>
<td>\tl_new:N</td>
<td>17, 250, 251, 252, 253, 256, 257, 258, 323</td>
</tr>
<tr>
<td>\tl_put_right:Nn</td>
<td>18, 132, 177, 209, 210, 214, 260, 355, 362</td>
</tr>
<tr>
<td>\tl_remove_once:Nn</td>
<td>223</td>
</tr>
<tr>
<td>\tl_reverse:n</td>
<td>348, 350, 354</td>
</tr>
<tr>
<td>\tl_set:Nn</td>
<td>41, 42, 43, 44, 45, 46, 47, 48, 49, 102, 103, 108, 109, 123, 127, 131, 142, 187, 233</td>
</tr>
<tr>
<td>\tl_set_eq:NN</td>
<td>225, 226</td>
</tr>
<tr>
<td>\tl_tail:n</td>
<td>349</td>
</tr>
<tr>
<td>\tl_to_str:n</td>
<td>368</td>
</tr>
<tr>
<td>\l_tmpa_tl</td>
<td>41, 42, 43, 44, 45, 46, 47, 48, 49, 50</td>
</tr>
<tr>
<td>\usepackage</td>
<td>11, 19, 52, 54</td>
</tr>
<tr>
<td>\withargs:nn</td>
<td>329, 345, 374</td>
</tr>
<tr>
<td>\withargs:nnn</td>
<td>320</td>
</tr>
<tr>
<td>\withargs:nunn</td>
<td>330, 346</td>
</tr>
</tbody>
</table>