Il concetto di ritorno al passato per le classi e i pacchetti*

Frank Mittelbach

Sommario

Nel 2015 è stato introdotto il concetto di "ritorno al passato" per il nucleo di LaTeX. Questa nuova funzionalità ci permise di apportare correzioni al software (cosa che più o meno non avvenne mai per quasi due decadi) mantenendo però la compatibilità col passato al massimo grado.

In questo articolo spieghiamo come ora abbiamo esteso questo concetto, che inizialmente non era stato implementato, al mondo dei pacchetti e delle classi. Siccome le classi e i pacchetti di estensione hanno requisiti diversi rispetto al nucleo di LaTeX, l'approccio è diverso (e più semplice). Ciò dovrebbe rendere più facile agli sviluppatori applicare la nuova tecnica ai loro pacchetti, e agli autori usarla quando è necessario.

Abstract

In 2015 a rollback concept for the LaTeX kernel was introduced. Providing this feature allowed us to make corrections to the software (which more or less didn't happen for nearly two decades) while continuing to maintain backward compatibility to the highest degree.

In this paper we explain how we have now extended this concept to the world of packages and classes, which was not covered initially. As classes and extension packages have different requirements compared to the kernel, the approach is different (and simplified). This should make it easy for package developers to apply it to their packages and authors to use when necessary.

1 Introduzione

Nel 2015 abbiamo introdotto il concetto di "ritorno al passato" per il nucleo di LaTeX,[1] che permette all'utente di richiedere il ritorno a una versione del kernel precedente rispetto a una specifica data mediante l'uso del pacchetto latexrelease (The LaTeX Team, 2018b). Per esempio

```
\RequirePackage[2016-01-01]{latexrelease}
```

serve per disfare tutte le modifiche del kernel (correzioni ed estensioni) apportate tra il primo di gennaio 2016 e la data corrente.[2] 'Disfare' significa reinstallare le definizioni correnti alla data richiesta e anche rimuovere nuovi comandi dalla memoria di TeX, cosicché `\newcommand` e dichiarazioni simili non falliscano per il fatto che un nome di comando è già stato definito.

Questo meccanismo aiuta nell'elaborare correttamente vecchi documenti che contengono rappezzi per cose assenti nei vecchi kernel, ma che nel frattempo sono state sistemate e che quindi farebbero fallire la compilazione del vecchio documento, oppure produrrebbero file di uscita diversi, se compilati con il nuovo kernel aggiornato.

Se necessario, il pacchetto latexrelease consente anche un "ritorno al futuro" senza bisogno di installare il nuovo formato. Per esempio, se l'installazione corrente è quella del 2016-04-01, ma si dispone di un documento che richiede un kernel datato 2018-01-01, allora questo si può ottenere richiedendo

```
\RequirePackage[2018-01-01]{latexrelease}
```

purché si disponga di una versione del pacchetto latexrelease che conosca le modifiche del kernel tra la data di quello di cui si dispone e la data richiesta. Procurarsi questa versione del pacchetto è semplice, poiché quella più recente può sempre essere scaricata da CTAN. Perciò si può elaborare il documento correttamente anche quando aggiornare l'installazione completa del sistema TeX non è consigliabile o è impossibile per una ragione qualsiasi.

Tuttavia, ritornare al passato con il kernel è soltanto metà dell'opera: l'universo LaTeX consiste di moltissimi pacchetti aggiuntivi, e questi non sono stati coinvolti nel ritorno al passato del kernel. Noi ora stiamo estendendo questo concetto mediante un metodo più semplice da usare con i pacchetti e le classi; un metodo che riteniamo di uso più diretto per gli sviluppatori e anche d'uso più facile per gli autori.

Differentemente dal metodo usato per il kernel, che rintraccia ogni modifica individualmente ed è capace di tornare al preciso stato che esso aveva a ogni specifica data, il nuovo metodo per i

*Questa è la traduzione in italiano, eseguita da Claudio Beccari, dell'articolo di Frank Mittelbach, *A rollback concept for packages and classes*, pubblicato su *TUGboat*, vol. 39, n. 2 (2018), p. 107-112. Eventuali errori sono da attribuire al traduttore.

1. D'ora in poi, sostituirò la locuzione 'nucleo di LaTeX' con la parola *kernel*. *N.d.T*

2. Ci sono delle eccezioni, in quanto alcune modifiche sono conservate: per esempio, l'abilità di accettare date nel formato ISO (p.es. 2016-01-01) in aggiunta alla precedente convenzione (p.es. 2016/01/01). Queste non sono riportate indietro perché rimuovere queste funzionalità potrebbe portare a inutili manchevolezze.

pacchetti e le classi dovrebbe applicarsi solo ai cambiamenti di maggiore importanza, per esempio, l'introduzione di nuove funzionalità o di modifiche (incompatibili) nella sintassi o nelle interfacce.[3]

Siccome avremo solo pochi punti di "ritorno" per ogni pacchetto o classe, le differenti versioni sono conservate in file diversi. Il file principale, dunque, necessita di una sola dichiarazione per ogni versione per consentire il ritorno al passato. L'inconveniente, perciò, consiste nel fatto che per ogni nuova versione con cambi importanti bisogna salvare l'intero file, invece di gestire solo le differenze fra versioni. Questo è il motivo per il quale questo approccio dovrebbe essere usato solo per modifiche importanti, cioè poche lungo la vita di un pacchetto.

Da un punto di vista tecnico è anche possibile usare il metodo introdotto con latexrelease, consistente, cioè, nel marcare le modifiche, con i comandi `\IncludeInRelease` e `\EndIncludeInRelease` — la documentazione del pacchetto (The LaTeX Team, 2018b) indica come applicarli nello scenario di un pacchetto — ma il loro uso nel codice del pacchetto è complicato e produce un codice difficile da leggere, specialmente quando vi sono molti cambiamenti di minore importanza. Questo prezzo da pagare è accettabile per un codice piuttosto stabile, come lo stesso kernel, poiché consente un controllo completo sul ritorno a qualsiasi data, ma non è davvero pratico nello sviluppo di un pacchetto o di una classe, cosicché, a nostra conoscenza, fino a oggi non è stato molto usato. Il paragrafo 5.4 dà alcuni consigli su come ottenere un controllo fine in modo più semplice.

2 Gli scenari tipici

Un esempio tipico, per il quale questa funzionalità di ritorno avrebbe prodotto grandi benefici (e lo farà per i pacchetti in futuro), è dato dal pacchetto caption di Axel Sommerfeldt. Questo pacchetto cominciò sotto il nome di caption con una certa interfaccia per l'utente. Dopo un po' di tempo, fu chiaro che in essa erano presenti delle manchevolezze nell'interfaccia per l'utente: per aggiustarle evitando di compromettere documenti già composti, Axel produsse il pacchetto caption2. Successivamente anche la sintassi di quello stesso pacchetto venne cambiata e nacque caption3, che alla fine venne rinominato caption. Ora i vecchi documenti che usavano il primo caption non possono più essere compilati senza modifiche, mentre i documenti del periodo intermedio richiedono ancora caption2 (che è dichiarato superato su CTAN, ma viene ancora distribuito con le distribuzioni più importanti). Di conseguenza, gli utenti abituati a copiare il preambolo da un loro documento al successivo, continuano ad usarlo senza notare che stanno usando un pacchetto difettoso con un'interfaccia limitata.

Un altro esempio è dato dal pacchetto fixltx2e, che per molti anni ha fornito delle correzioni al kernel. Nel 2015 queste correzioni sono state introdotte nel kernel, cosicché questo pacchetto è diventato un involucro quasi vuoto, che si limita a informare l'utente che non ha più bisogno di essere usato. Tuttavia se si compila un documento antecedente al 2015 che carica fixltx2e, le correzioni che quel pacchetto forniva (come, per esempio, le correzioni all'algoritmo di gestione degli oggetti flottanti) verrebbero persi se anche fixltx2e non ritornasse al passato — fortunatamente lo fa e l'ha fatto sempre, così che in effetti non è un involucro vuoto.

Un altro esempio, un po' differente, è dato da amsmath, che per circa un decennio non ha subito alcuna correzione, anche se diversi problemi erano emersi durante quel periodo di tempo. Se quelle manchevolezze venissero finalmente corrette, esse produrrebbero effetti su molti documenti prodotti dal 2000 in poi, in quanto i loro autori avrebbero provveduto a correggere a mano questa o quella manchevolezza. Naturalmente, come per il pacchetto caption, si sarebbero potuti introdurre i pacchetti amsmath2, amsmath3, ..., ma ciò caricherebbe inutilmente l'utente, costretto a specificare sempre l'ultima versione (invece di usare automaticamente la versione più recente, a meno che non sia espressamente necessaria una versione precedente).

3 L'interfaccia del documento

Per impostazione predefinit,a LaTeX usa automaticamente la versione corrente di qualunque classe o pacchetto — e prima di offrire il concetto di ritorno al passato, ha fatto sempre così, a meno che il pacchetto o la classe non disponesse un suo proprio meccanismo per fornire la versione giusta, o usando nomi alternativi, o selezionando a mano determinate opzioni.

3.1 Ritorno al passato globale

Con il nuovo concetto di ritorno al passato, tutto ciò che l'utente deve fare (se desidera compilare il proprio documento con una specifica versione del kernel o dei pacchetti) è aggiungere la chiamata al pacchetto latexrelease all'inizio del suo preambolo specificando la data nelle opzioni, come mostrato nel primo esempio:

```
\RequirePackage[2018-01-01]{latexrelease}
```

Questo produce il ritorno al passato del kernel allo stato in cui era quel giorno (come descritto in precedenza) e per ogni pacchetto o classe controlla se ci sono versioni alternative e sceglie quella più appropriata per quel pacchetto o classe in rapporto alla data specificata.

3. Le versioni contenenti tali importanti modifiche verrano definite nel seguito *versioni maggiori*. *N.d.T*

3.2 Ritorno al passato specifico

C'è un altro possibile aggiustamento fine: sia `\documentclass` sia `\usepackage` dispongono di un secondo argomento (poco conosciuto e quindi poco usato) che fino ad oggi poteva consentire di specificare una "data minima". Per esempio, specificando

```
\usepackage[colaction]
           {multicol}[2018-01-01]
```

si richiede che il pacchetto multicol non sia più vecchio dell'inizio del 2018. Se è disponibile solo una versione precedente, allora la compilazione del documento produce un avviso:

```
LaTeX\ Warning: You have requested, on input
   line 12, version ‘2028-01-01’ of package
   multicol, but only version
   ‘2017/04/11 v1.8q multicolumn formatting
   (FMi)’ is available.
```

L'idea dietro questo approccio è che i pacchetti raramente cambiano sintassi in un modo incompatibile, ma più spesso aggiungono nuove funzionalità: con una dichiarazione di questo genere vuol dire che si richiede una versione che offre certe funzionalità.

Il nuovo concetto del ritorno ora estende l'uso di questo argomento facoltativo permettendo di fornire una data precisa per il ritorno. Questo effetto si ottiene premettendo il segno di 'uguale' alla stringa della data. Per esempio:

```
\usepackage{multicol}[=2017-06-01]
```

richiede una versione di multicol corrispondente a quella che aveva nel giungo 2017.

Perciò, supponendo che in futuro avvenga una modifica importante di questo pacchetto che cambia il modo di bilanciare le colonne, la specifica precedente richiederebbe un ritorno del pacchetto alla versione di oggi, che ha la data 2017-04-11. Il vecchio modo di usare quell'argomento facoltativo è ancora disponibile perché è determinato dalla presenza o dall'assenza del segno `=`.

Lo stesso meccanismo è disponibile per le classi di documento attraverso la dichiarazione `\documentclass` e per `\RequirePackage`, nel caso in cui fosse necessario usarlo.

3.3 Specificare la versione invece della data

Specificare la data di ritorno è particolarmente adeguato se si vuole assicurare che il comportamento del programma di composizione (cioè il kernel e tutti i pacchetti) corrisponda a quella data specifica. Infatti, non appena si è finito di editare un documento, lo si può preservare per i posteri semplicemente aggiungendo questa riga di codice:

```
\RequirePackage[⟨data di oggi⟩]
               {latexrelease}
```

Questo significa che la sua compilazione sarà eseguita un po' più lentamente (poiché il kernel potrebbe dover ritornare al passato e ogni pacchetto viene controllato per versioni alternative), ma avrebbe il vantaggio che la compilazione, anche in un lontano futuro, probabilmente funzionerà ancora, senza che sia necessario aggiungere quella linea di codice in quel lontano momento.

Tuttavia, in un caso come quello di caption o, per esempio, di longtable, che potrebbe ricevere un aggiornamento importante dopo diversi anni, sarebbe bello poter specificare una versione per nome, invece che attraverso una data: per esempio, di quest'ultimo un utente potrebbe voler usare la versione 4, invece della versione 5, qualora queste due versioni abbiano sintassi differenti e incompatibili o producano risultati diversi.

Ciò è possibile anche ora se lo sviluppatore definisce per i pacchetti o per le classi le loro versioni per nome: l'utente può allora richiedere una versione per nome semplicemente usando questo secondo argomento con il nome preceduto da una segno di 'uguale'. Per esempio, se esisterà una nuova versione di longtable e la vecchia versione (quella che oggi è corrente) è etichettata "`v4`", allora per selezionare la vecchia versione sarà possibile usare semplicemente

```
\usepackage{longtable}[=v4]
```

Si noti che non è necessario sapere che la nuova versione ha la data 2018-04-01 (nemmeno richiedere una data precedente a questa) per usare ancora quella specifica vecchia versione.

Il nome della versione è una stringa arbitraria a discrezione dell'autore del pacchetto — ma, attenzione!, non deve essere scambiata con una data, cioè non deve contenere trattini o barre, poiché questi segni confonderebbero la routine di analisi della stringa che dovrebbe indicare una data.[4]

3.4 Dati errati

L'interfaccia per l'utente è piuttosto semplice e, per mantenere alta la velocità di elaborazione, l'analisi sintattica è molto leggera. Di base viene usata la stessa routine di parsing del kernel, che è piuttosto intollerante se trova dati inattesi.

In sostanza, ogni stringa che contiene un trattino e una barra fa partire l'algoritmo per analizzare una data che, quindi, si aspetta due trattini (per il caso di una data ISO) o due barre (altrimenti) e, oltre questi separatori, solamente cifre. Se trova qualsiasi altra cosa, è possibile che l'utente riceva il messaggio d'errore "`Missing \begin{document}`" o, magari, potrebbe avere luogo un riconoscimento strano e potrebbe essere eseguita una strana selezione. Per esempio, una data del tipo `2011/02` può

4. Ovviamente, un'analisi più sofisticata della stringa potrebbe sistemare questo aspetto, ma noi usiamo una routine semplice e veloce che cerca solo barre e trattini senza nessuna ulteriore analisi.

voler dire per noi "febbraio 2011", ma per la routine di analisi è un qualche giorno dell'anno 20 d.C. In altre parole, la stringa viene convertita nel solo numero `201102`, così che quando questo numero viene confrontato con il numero 20000101, sarà minore, e quindi precedente, anche se il secondo è la rappresentazione numerica del 1º gennaio 2000.

Pertanto, ultimo avviso: non si scrivano date ortograficamente errate e tutto andrà bene. Non è stato un problema in passato, perciò si spera che continui ad andar bene con questo modo semplice di analizzare le stringhe. Se non fosse così, saremmo costretti ad eseguire controlli più estesi durante l'analisi della stringa.

3.5 Consigli per i nuovi utenti

Se un documento usa le nuove funzionalità di ritorno al passato globale, dovrebbe essere compilabile con qualsiasi installazione più recente dell'inizio del 2015, quando è stato reso disponibile per la prima volta il pacchetto latexrelease. Se l'installazione è più datata, allora è necessario un aggiornamento, oppure aggiungere la versione corrente di latexrelease.

Tuttavia, se il documento usa il nuovo concetto del ritorno individuale per i pacchetti e le classi (cioè con la sintassi `=...` nell'argomento facoltativo) è necessario usare un'installazione del 2018 o successiva.[5] Le distribuzioni precedenti si bloccherebbero appena trovano il segno di uguale, perché si aspettano di trovare solo una stringa che corrisponde a una data valida.

4 L'interfaccia della classe o del pacchetto

Il meccanismo di ritorno al passato per i pacchetti e le classi viene reso possibile mettendo, all'inizio del file di codice, una sezione di dichiarazioni che informano il kernel in merito all'esistenza di versioni alternative.

Queste dichiarazioni devono precedere tutto il codice e devono essere in ordine di data perché il meccanismo di caricamento le deve valutare una ad una e, quando una versione adeguata è stata trovata, viene caricata e l'elaborazione del file principale del pacchetto o delle classe può terminare. Se non ci sono queste dichiarazioni, o se le vecchie versioni sono state cancellate per qualsiasi motivo, l'elaborazione continua leggendo tutto il file principale.

Le versioni precedenti sono contenute in file distinti, uno per ciascuna versione, e suggeriamo di usare uno schema del tipo ⟨*nome del pacchetto*⟩`-`⟨*data*⟩`.sty` perché questo è facile da capire e viene ordinato facilmente per data in una cartella. Tuttavia, qualsiasi altro schema va altrettanto bene, perché il nome fa parte della dichiarazione.

5. In alternativa, si potrebbe tentare il "ritorno al futuro" usando una versione corrente di latexrelease e specificando una data adeguata.

Il contenuto di questa versione di file è semplicemente il pacchetto o la classe in uso precedentemente. Questo significa che prima di fare una nuova versione, tutto ciò che bisogna fare consiste nel disporre di una versione verbatim del file corrente e nel darle un nuovo nome adeguato.[6]

In questo modo è anche immediato includere vecchie versioni dopo quanto si è fatto; in altre parole, riprendendo l'esempio di caption, Axel potrebbe assegnare alla sua primissima versione il nome caption-⟨*una data*⟩, a caption2 il nome caption-⟨*un'altra data*⟩, oltre a fornire le necessarie dichiarazioni alla versione corrente.

Le dichiarazioni necessarie nel file principale sono fornite dai comandi `\DeclareRelease` e `\DeclareCurrentRelease`, che devono essere usati nella sezione *selezione delle versioni* all'inizio del file. Per ciascuna precedente versione si può specificare un ⟨*nome*⟩, la ⟨*data*⟩ di quando quella versione era diventata disponibile e il ⟨*file esterno*⟩ che contiene il codice.

La sintassi per dichiarare le precedenti versioni è la seguente:

`\DeclareRelease`
`{`⟨*nome*⟩`}{`⟨*data*⟩`}{`⟨*file esterno*⟩`}`

Degli argomenti ⟨*nome*⟩ e ⟨*data*⟩, o l'uno o l'altro può essere vuoto, ma non tutti e due contemporaneamente. Se non si specifica la ⟨*data*⟩, è per riferirsi a versioni "beta", che gli utenti possono esplicitamente richiamare ma che non dovrebbero giocare alcun ruolo nel ritorno al passato.

Anche la versione corrente riceve una dichiarazione, ma con due soli argomenti: un ⟨*nome*⟩ (che di nuovo può essere vuoto) e una ⟨*data*⟩, perché il codice per questa versione costituisce il resto del file corrente:

`\DeclareCurrentRelease{`⟨*nome*⟩`}{`⟨*data*⟩`}`

Questa dichiarazione deve essere l'ultima nella sequenza delle dichiarazioni e termina l'elaborazione della *selezione delle versioni*.

L'ordine delle altre versioni deve iniziare con la più vecchia e procedere verso la più recente, perché il meccanismo di caricamento confronta ogni dichiarazione di versione con la data specificata per il ritorno al passato e finisce nel momento in cui ne trova una più recente della data specificata. Esso quindi seleziona la precedente versione, cioè una che abbia una data precedente o uguale a quella desiderata. Poiché le dichiarazioni eseguite con `\DeclareRelease` con l'argomento ⟨*data*⟩

6. Invece di fare una copia verbatim, si potrebbe aggiustare il commento aggiunto dal pacchetto docstrip all'inizio del file. Sebbene la cosa sia tecnicamente corretta, può ingenerare confusione se il file contiene la frase "was generate from ...", visto che ora si tratta di una versione congelata che rappresenta una particolare situazione nel tempo, piuttosto che una situazione che riguarda un file generato in un certo momento ma che può venire rigenerato quandunque si voglia se è necessario.

vuoto non giocano nessun ruolo nel ritorno al passato, esse possono essere messe dove si vuole nella sequenza delle dichiarazioni.

Un tipico esempio di *sezione delle versioni* è all'inizio del pacchetto multicol; ha l'aspetto seguente perché vi è stato un aggiornamento importante nell'aprile 2018. Si noti che a causa di alcune piccole modifiche eseguite dopo quella data, la data effettiva del pacchetto è già quella di giugno.

```
\NeedsTeXFormat{LaTeX2e}[2018-04-01]
\DeclareRelease{}{2017-04-01}
                 {multicol-2017-04-01.sty}
\DeclareCurrentRelease{}{2018-04-01}
\ProvidesPackage{multicol}[2018/06/26 v1.8t
              multicolumn formatting (FMi)]
```

Se il ritorno al passato si affida a un nome invece che a una data, il meccanismo lavora nello stesso modo, salvo che viene scelta una versione in base al nome che sia coincidente con quello richiesto. Se nessun nome coincide con quello richiesto viene emesso un avviso d'errore e l'elaborazione prosegue usando la versione corrente.

5 Considerazioni speciali per gli sviluppatori

Quando viene caricata una precedente versione di un pacchetto o una classe, entrambi i comandi di dichiarazione di versione sono cambiati in comandi *no-op*, inerti, così che, nel caso in cui i file che contengono il codice abbiano altre simili dichiarazioni, esse non producano alcun effetto. Questo permette di copiare semplicemente il codice di una versione da sostituire con una più recente, senza toccarne affatto il contenuto. Naturalmente, togliendo quelle dichiarazioni dai file delle versioni precedenti non si produce alcun danno e si rende l'elaborazione e il caricamento leggermente più veloce.

Come detto in precedenza, il miglior modo per specificare i nomi di versione è quello di appendere la data di versione al nome del pacchetto o della classe, ma l'argomento *file esterno* consente anche altri schemi.

Ci si può domandare perché si debba eseguire una dichiarazione per la versione corrente, visto che è poi seguita dalla dichiarazione `\Provides...`, che contiene anch'essa una data e una stringa descrittiva, e potrebbe segnalare essa stessa la fine della sezione delle versioni. Il motivo è questo: se si vuol dare alla versione corrente un nome, è bene che questo sia una cosa semplice, come `v4` (da mantenere così) anche se la versione corrente tecnicamente è già alla versione `v4.2c` e viene indicata così nella dichiarazione `\ProvidesPackage`. Per lo stesso motivo (visto che non necessariamente ogni versione con modifiche minori viene indicata come una versione diversa alla quale si possa applicare il ritorno al passato), la ⟨*data*⟩ nella dichiarazione `\DeclareCurrentRelease` riflette quella in cui è stata introdotta la versione con cambiamenti importanti. Pertanto, dopo un po' quella data potrebbe essere precedente a quella del pacchetto corrente.

5.1 Utenti dei primi tempi dall'introduzione del nuovo metodo

Per un periodo di uno o due anni dall'introduzione di questo nuovo metodo, c'è il pericolo che gli utenti con installazioni precedenti scelgano una forma specifica di un pacchetto, per esempio da CTAN, che contenga una dichiarazione con la quale il kernel (del 2017 o precedente) è del tutto incompatibile. Perciò, può essere raccomandabile che gli sviluppatori aggiungano anche le righe seguenti all'inizio dei pacchetti e delle classi quando usano questa funzionalità di ritorno al passato:

```
\providecommand\DeclareRelease[3]{}
\providecommand\declareCurrrentRelease[2]{}
```

In questo modo le dichiarazioni vengono ignorate nel caso che il vecchio kernel non sappia cosa farsene.

Una alternativa potrebbe essere quella di aggiungere una riga che stabilisce una data minima per la versione del kernel, cioè

```
\NeedsTeXFormat{LaTeX2e}[2018-04-01]
```

cosicché gli utenti ricevono un chiaro messaggio d'errore, cioè che essi devono aggiornare la loro installazione se vogliono usare il file corrente.

5.2 Una nuova versione maggiore a livello beta

Se si sta lavorando su una nuova versione maggiore del proprio pacchetto o della propria classe, si potrebbe metterla a disposizione anche per il pubblico, cosi che la si possa collaudare e si possa ricevere il necessario feedback. In questo caso la versione corrente è quella ufficiale da usare come predefinita, mentre la versione "beta" potrebbe essere selezionata solo espressamente. Per ottenere questo risultato, si può aggiungere

`\DeclareRelease{beta}{}{`⟨*file esterno*⟩`}`

prima di

`\DeclareCurrentRelease{}{`⟨*data*⟩`}`

cosicché i collaudatori possono esplicitamente accedere alla nuova versione richiedendola mediante

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[=beta]`

mentre gli altri, che caricano il pacchetto senza l'argomento supplementare, caricano la versione corrente.

5.3 Due nuove versioni già in uso

Uno scenario particolare per il quale questo metodo è adatto solo parzialmente è quello in cui ci sono due versioni maggiori entrambe continuamente in uso e mantenute attivamente (cioè ricevono correzioni e altri aggiornamenti di tanto in tanto). In questo caso è necessario considerare una versione come primaria e consentire all'altra (e i suoi aggiornamenti) di essere accessibile mediante i nomi: il ritorno al passato, ovviamente, funziona solo con la linea principale di sviluppo.

Per esempio, se entrambe le versioni `v4` e `v5` del pacchetto foo sono in uso, e se si considera la versione `v5` quella da portare avanti (anche se si sta ancora facendo manutenzione alla versione `v4`), allora si può seguire una strategia come la seguente:

```
% last v4 only release:
  \DeclareRelease{}{2017-06-23}
                   {foo-2017-06-23.sty}
% first v5 release:
  \DeclareRelease{}{2017-08-01}
                  {foo-2017-08-01.sty}
% patch to v4 after v5 got introduced:
  \DeclareRelease{v4.1}{}
                   {foo-v4-2017-09-20.sty}
% patch to v5:
  \DeclareRelease{}{2017-08-25}
                   {foo=2017-08-25.sty}
% another patch to v4:
  \DeclareRelease{v4.2}{}
                   {foo-v4-2017-10-01.sty}
% nickname for the latest v4 if you want
% users to have simple access via a name
  \DeclareRelease{v4}{}
                   {foo-v4-2017-10-01.sty}
% current v5 with further patches:
  \DeclareCurrentRelease{v5}{2018-01-01}
```

In questo modo, gli utenti possono usare `\usepackage{foo}[=v4]` per accedere all'ultima versione `v4`, oppure usano il nome più specifico come `[v4.1]` per accedere a una versione particolare. Questo significa che se si richiede il pacchetto foo nella versione `v4` (o una delle sue sotto versioni), esso non sarà cambiato, anche se esiste una specificazione di ritorno al passato attraverso latexrelease.

Questo normalmente dovrebbe andare bene, ma se veramente si chiede un ritorno al passato automatico per entrambe queste versioni maggiori, perché sono in effetti dello stesso rango, allora si sta essenzialmente dicendo che sono due lavori distinti con qualche cosa in comune. In questo caso, si dovrebbero usare due nomi separati, per esempio, chiamando la versione precedente foo-v4, quando si introduce la versione 5 di foo e da quel momento in poi le si gestisce in modo indipendente.[7]

7. Sebbene in rari casi questo potrebbe essere l'approccio migliore, si cerchi di evitarlo, perché nel lungo termine la gestione potrebbe diventare, come minimo, problematica.

5.4 Controllo fine (se necessario)

Come detto prima, l'interfaccia è deliberatamente progettata per essere semplice e facile da usare. Come prezzo da pagare, ogni ritorno al passato (per impostazione predefinita) punta a un file separato. L'idea dietro a tutto ciò è che non sembra una gran cosa stare dietro ad ogni singolo minimo cambiamento come punto di ritorno al passato, ma che questi siano solo quelli che possono alterare il comportamento di un pacchetto nell'elaborazione di un documento cosi che, quando si compilano documenti più vecchi, sia importante essere capaci di andare indietro a una data precedente.

Tuttavia, se ci si trova in una situazione in cui si hanno molti file per tornare indietro con differenze minori, e lo si consideri insoddisfacente, allora si può ricorrere a un altro comando che permette di combinare diversi file in un unico file. Dentro un file esterno, nominato in una dichiarazione `\DeclareRelease`, si può usare

`\IfTargetDateBefore{`⟨*data*⟩`}`
`{`⟨*codice prima di*⟩`}{`⟨*codice da*⟩`}`

Questo va usato dopo la sezione *selezione delle versioni* (se presente) e ha il seguente effetto: se l'utente ha richiesto, per esempio, `[=2017-06-01]`, il meccanismo prima seleziona il file che era corrente a quella data, cioè la versione che era stata resa disponibile in quella data o prima di quella data, poi la ⟨*data*⟩ viene confrontata con quella specificata, `2017-06-01` e, a seconda dell'esito del confronto, esegue il ⟨*codice prima di*⟩ quella data, oppure il ⟨*codice da*⟩ quella data in poi.

In questo modo un singolo file esterno può contenere informazioni per il ritorno al passato per diversi aggiustamenti eseguiti in date diverse; ma, naturalmente, il carico di lavoro ricade sullo sviluppatore che deve aggiungere i comandi appropriati, e questo è un po' più oneroso che non semplicemente copiare un file cambiandogli il nome.

L'alternativa consiste nell'usare `\IncludeInRelease` e `\EndIncludeInRelease`. La documentazione di latexrelease (The LaTeX Team, 2018b) dà consigli su come usare in pratica questi comandi.

5.5 L'uso di l3build per la gestione dei sorgenti

Se si usa l3build, (The LaTeX Team, 2018a), per gestire i propri file sorgente, è necessario essere certi che i file per le versioni precedenti siano copiati nella distribuzione; per consentire questo, la configurazione di default per l3build specifica

```
sourcefiles = {"*.dtx", ".ins",
               "*-????-??-??.sty"}
```

cioè che tutti i file `.dtx` e `.ins`, insieme a quelli con estensione `.sty`, secondo la convenzione consigliata in questo articolo per rinominare le versioni

precedenti, siano automaticamente inclusi nella costruzione del pacchetto di distribuzione dei file.

Se si preferisce una convenzione diversa, bisogna aggiustare le impostazioni nel file `build.lua` del proprio progetto. Altrimenti, bisogna andare avanti senza aggiustamenti.

6 Sommario dei comandi

6.1 L'interfaccia del documento per gli utenti

Per specificare un ritorno al passato di tipo globale si usi

`\RequirePackage[`⟨*data*⟩`]{latexrelease}`

all'inizio del proprio documento.

Per un ritorno al passato di singoli pacchetti o classi, si usi il secondo argomento facoltativo precedendo la data con il segno di uguale, cioè

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}[=`⟨*data*⟩`]`
`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[=`⟨*data*⟩`]`

6.2 L'interfaccia di classi e pacchetti per gli sviluppatori

Per dichiarare una versione precedente o speciale si usi

`\DeclareRelease{`⟨*nome*⟩`}{`⟨*data*⟩`}`
`{`⟨*file esterno*⟩`}`

Si lasci ⟨*nome*⟩ vuoto se si desidera che il ritorno al passato avvenga solo attraverso le date. Si lasci ⟨*data*⟩ vuoto se questo file speciale deve essere accessibile solo attraverso il ⟨*nome*⟩.

In ogni caso si termini la *selezione delle versioni* con la dichiarazione relativa alla versione corrente

`\DeclareCurrentRelease{`⟨*nome*⟩`}{`⟨*data*⟩`}`

In questa dichiarazione bisogna specificare la ⟨*data*⟩ ma il ⟨*nome*⟩ può essere lasciato vuoto (cosa che rappresenta il caso normale).

Nel file corrispondente all'ultima versione (ma dopo la sezione *selezione delle versioni*), si può specificare del codice condizionale da eseguire per un ritorno al passato con una data specifica, usando

`\IfTargetDateBefore{`⟨*data*⟩`}`
`{`⟨*codice prima di*⟩`}{`⟨*codice da*⟩`}`

Riferimenti bibliografici

The LaTeX Team (2018a). *The l3build package — Checking and building packages*. Leggibile con `texdoc l3build`. La versione del 2015 si trova in https://ctan.org/pkg/l3build.

— (2018b). *The latexrelease package*. Leggibile con `texdoc latexrelease`. Altrimenti disponibile in https://www/latex-project.org/help/documentation.

▷ Frank Mittelbach
https://www.latex-project.org