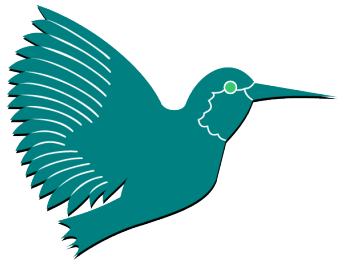
Haken, Steckdosen, ...

Frank Mittelbach



Darmstadt, April 2025

Was ist ein Haken?

Und wofür wird er gebraucht?

Charakteristiken von Haken

Hakenstatus

- Ein Haken kann leer sein
- An ihm können ein oder mehrere Gegenstände aufgehängt sein

Artikel am Haken

- Artikel können
  - hinzugefügt,
  - entfernt oder
  - neu sortiert werden

Offsite Lagerung

- Ein Haken kann genutzt werden, um Gegenstände zur späteren Verwendung aufzuheben

Charakteristiken von Haken

Hakenstatus

- Ein Haken kann leer sein
- An ihm können ein oder mehrere Gegenstände aufgehängt sein

Artikel am Haken

- Artikel können
  - hinzugefügt,
  - entfernt oder
  - neu sortiert werden

Offsite Lagerung

- Ein Haken kann genutzt werden, um Gegenstände zur späteren Verwendung aufzuheben

Charakteristiken von Haken

Hakenstatus

- Ein Haken kann leer sein
- An ihm können ein oder mehrere Gegenstände aufgehängt sein

Artikel am Haken

- Artikel können
  - hinzugefügt,
  - entfernt oder
  - neu sortiert werden

Offsite Lagerung

- Ein Haken kann genutzt werden, um Gegenstände zur späteren Verwendung aufzuheben

Geschichte der LaTeX-Haken (Hooks)

LaTeX 2.09

- Keine vorhanden — nur das Patchen von internen Befehlen war möglich

LaTeX 2ε

- Ein paar; hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Keine Verwaltung — wer zuerst kommt, mahlt zuerst

Mittlerweile

- Ein allgemeines Hook-Management
- Hooks an vielen Stellen (die Zahl wächst)
- Hook-Daten können von außen manipuliert werden

Geschichte der LaTeX-Haken (Hooks)

LaTeX 2.09

- Keine vorhanden — nur das Patchen von internen Befehlen war möglich

LaTeX 2ε

- Ein paar; hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Keine Verwaltung — wer zuerst kommt, mahlt zuerst

Mittlerweile

- Ein allgemeines Hook-Management
- Hooks an vielen Stellen (die Zahl wächst)
- Hook-Daten können von außen manipuliert werden

Geschichte der LaTeX-Haken (Hooks)

LaTeX 2.09

- Keine vorhanden — nur das Patchen von internen Befehlen war möglich

LaTeX 2ε

- Ein paar; hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Keine Verwaltung — wer zuerst kommt, mahlt zuerst

Mittlerweile

- Ein allgemeines Hook-Management
- Hooks an vielen Stellen (die Zahl wächst)
- Hook-Daten können von außen manipuliert werden

Das Problem mit den LaTeX 2ε-Hooks ...

Das Problem mit den LaTeX 2ε-Hooks ... Keine Änderungen möglich / keine externe Speicherung / nur wenige vorhanden

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Keine Änderungen möglich

- Die Reihenfolge der Codeausführung wurde durch die Reihenfolge festgelegt, in der der Code hinzugefügt wurde
- Im Falle von Problemen lautete der Rat daher: "Ändere die Reihenfolge des Paketladens"

Keine Vorratshaltung

- Man kann keinen Code für andere Pakete bereitstellen, es sei denn, diese sind bereits geladen

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Keine Änderungen möglich

- Die Reihenfolge der Codeausführung wurde durch die Reihenfolge festgelegt, in der der Code hinzugefügt wurde
- Im Falle von Problemen lautete der Rat daher: "Ändere die Reihenfolge des Paketladens"

  Aber das hat oft nicht funktioniert

Keine Vorratshaltung

- Man kann keinen Code für andere Pakete bereitstellen, es sei denn, diese sind bereits geladen

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Keine Änderungen möglich

- Die Reihenfolge der Codeausführung wurde durch die Reihenfolge festgelegt, in der der Code hinzugefügt wurde
- Im Falle von Problemen lautete der Rat daher: "Ändere die Reihenfolge des Paketladens"

  Aber das hat oft nicht funktioniert

Keine Vorratshaltung

- Man kann keinen Code für andere Pakete bereitstellen, es sei denn, diese sind bereits geladen
- Infolgedessen musste ein Paket wie `hyperref` komplexen bedingten Code verwenden
  - basierend auf bereits geladenen Paketen;
  - prüfen ob einige später geladen werden und
  - je nach Paketkombination unterschiedlichen Code ausführen

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Keine Änderungen möglich

- Die Reihenfolge der Codeausführung wurde durch die Reihenfolge festgelegt, in der der Code hinzugefügt wurde
- Im Falle von Problemen lautete der Rat daher: "Ändere die Reihenfolge des Paketladens"

  Aber das hat oft nicht funktioniert

Keine Vorratshaltung

- Man kann keinen Code für andere Pakete bereitstellen, es sei denn, diese sind bereits geladen
- Infolgedessen musste ein Paket wie `hyperref` komplexen bedingten Code verwenden
  - basierend auf bereits geladenen Paketen;
  - prüfen ob einige später geladen werden und
  - je nach Paketkombination unterschiedlichen Code ausführen

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an derselben Stelle einhaken, es sei denn, dass
  - sie voneinander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an der gleichen Stelle einhaken, es sei denn, dass
  - sie von einander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)
- Die Pakete sind oft mit anderen inkompatibel
- Aktualisierungen der Interna zerstören die Patches

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an der gleichen Stelle einhaken, es sei denn, dass
  - sie von einander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)
- Die Pakete sind oft mit anderen inkompatibel
- Aktualisierungen der Interna zerstören die Patches

Ergebnis: Die Benutzer waren oft unzufrieden

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an der gleichen Stelle einhaken, es sei denn, dass
  - sie von einander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)
- Die Pakete sind oft mit anderen inkompatibel
- Aktualisierungen der Interna zerstören die Patches

Ergebnis: Die Benutzer waren oft unzufrieden

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an der gleichen Stelle einhaken, es sei denn, dass
  - sie von einander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)
- Die Pakete sind oft mit anderen inkompatibel
- Aktualisierungen der Interna zerstören die Patches

Ergebnis: Die Benutzer waren oft unzufrieden

Keine Änderungen / Keine Vorratshaltung / Nur wenige

Nicht genügend Hooks

- Hauptsächlich `\AtBeginDocument` und `\AtEndDocument`
- Einige Pakete haben versucht, ein paar zusätzliche Hooks anzubieten

Für alles andere musste man Interna patchen (d.h. überschreiben)

Patching bedeutet

- Verschiedene Pakete können sich nicht an der gleichen Stelle einhaken, es sei denn, dass
  - sie von einander wissen und
  - die verschiedenen Szenarien berücksichtigen (d.h. welche Pakete vorhanden sind und in welcher Reihenfolge)
- Die Pakete sind oft mit anderen inkompatibel
- Aktualisierungen der Interna zerstören die Patches

Ergebnis: Die Benutzer waren oft unzufrieden

Das LaTeX Wartungs-/Verbesserungsdilemma

Option 1: Fehler nicht beheben / keine Verbesserungen vornehmen

- Benutzer sind unglücklich, wenn diese benötigt werden, aber nicht verfügbar sind
- Die zunehmende Inkompatibilität im Laufe der Zeit macht jeden unglücklich

Option 2: Fehlerbehebung / Verbesserungen vornehmen

- Macht Entwickler unglücklich, wenn ihre Patches durch Kernel-Fixes oder Verbesserungen beschädigt werden
- Macht Benutzer unglücklich — denn etwas Neues bricht immer irgendwo eine bestehende Anwendung (auch wenn das normalerweise schnell behoben wird)

Das LaTeX Wartungs-/Verbesserungsdilemma

Option 1: Fehler nicht beheben / keine Verbesserungen vornehmen

- Benutzer sind unglücklich, wenn diese benötigt werden, aber nicht verfügbar sind
- Die zunehmende Inkompatibilität im Laufe der Zeit macht jeden unglücklich

Option 2: Fehlerbehebung / Verbesserungen vornehmen

- Macht Entwickler unglücklich, wenn ihre Patches durch Kernel-Fixes oder Verbesserungen beschädigt werden
- Macht Benutzer unglücklich — denn etwas Neues bricht immer irgendwo eine bestehende Anwendung (auch wenn das normalerweise schnell behoben wird)

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Unsere Antwort bis 2016: Der LaTeX-Kernel bleibt unangetastet

Unsere Antwort heute

- Befreien wir uns vom Patchen in Paketen, indem wir geeignete Hooks bereitstellen
- Dies macht Entwickler immer noch unglücklich, da dies bedeutet, dass Pakete geändert werden müssen, um Hooks zu verwenden

Das ist aber hoffentlich für Entwickler nur ein einmaliger Aufwand!

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Unsere Antwort bis 2016: Der LaTeX-Kernel bleibt unangetastet

Unsere Antwort heute

- Befreien wir uns vom Patchen in Paketen, indem wir geeignete Hooks bereitstellen
- Dies macht Entwickler immer noch unglücklich, da dies bedeutet, dass Pakete geändert werden müssen, um Hooks zu verwenden

Das ist aber hoffentlich für Entwickler nur ein einmaliger Aufwand!

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Unsere Antwort bis 2016: Der LaTeX-Kernel bleibt unangetastet

Unsere Antwort heute

- Befreien wir uns vom Patchen in Paketen, indem wir geeignete Hooks bereitstellen
- Dies macht Entwickler immer noch unglücklich, da dies bedeutet, dass Pakete geändert werden müssen, um Hooks zu verwenden

Das ist aber hoffentlich für Entwickler nur ein einmaliger Aufwand!

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Unsere Antwort bis 2016: Der LaTeX-Kernel bleibt unangetastet

Unsere Antwort heute

- Befreien wir uns vom Patchen in Paketen, indem wir geeignete Hooks bereitstellen
- Dies macht Entwickler immer noch unglücklich, da dies bedeutet, dass Pakete geändert werden müssen, um Hooks zu verwenden

Das ist aber hoffentlich für Entwickler nur ein einmaliger Aufwand!

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Die Aufgabe

- Identifizierung aller Stellen, an denen ein Patchen als notwendig erachtet wurde
  - Zum Beispiel wird `\@footnotetext` derzeit von 7 Paketen an 4 verschiedenen Stellen gepatcht
- Geeignete Hooks bereitstellen, um die Notwendigkeit zu patchen zu vermeiden
- Paketautoren darauf hinweisen, dass der eine oder andere Patch schlicht nicht nötig war/ist
- Aktualisierung der Pakete
  - um diese Hooks dann zu verwenden
  - bzw. Patches auszubauen, wenn diese unnötig sind

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Die Aufgabe

- Identifizierung aller Stellen, an denen ein Patchen als notwendig erachtet wurde
  - Zum Beispiel wird `\@footnotetext` derzeit von 7 Paketen an 4 verschiedenen Stellen gepatcht
- Geeignete Hooks bereitstellen, um die Notwendigkeit zu patchen zu vermeiden
- Paketautoren darauf hinweisen, dass der eine oder andere Patch schlicht nicht nötig war/ist
- Aktualisierung der Pakete
  - um diese Hooks dann zu verwenden
  - bzw. Patches auszubauen, wenn diese unnötig sind

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Die Aufgabe

- Identifizierung aller Stellen, an denen ein Patchen als notwendig erachtet wurde
  - Zum Beispiel wird `\@footnotetext` derzeit von 7 Paketen an 4 verschiedenen Stellen gepatcht
- Geeignete Hooks bereitstellen, um die Notwendigkeit zu patchen zu vermeiden
- Paketautoren darauf hinweisen, dass der eine oder andere Patch schlicht nicht nötig war/ist
- Aktualisierung der Pakete
  - um diese Hooks dann zu verwenden
  - bzw. Patches auszubauen, wenn diese unnötig sind

Das LaTeX Wartungs-/Verbesserungsdilemma (Fortsetzung)

Die Aufgabe

- Identifizierung aller Stellen, an denen ein Patchen als notwendig erachtet wurde
  - Zum Beispiel wird `\@footnotetext` derzeit von 7 Paketen an 4 verschiedenen Stellen gepatcht
- Geeignete Hooks bereitstellen, um die Notwendigkeit zu patchen zu vermeiden
- Paketautoren darauf hinweisen, dass der eine oder andere Patch schlicht nicht nötig war/ist
- Aktualisierung der Pakete
  - um diese Hooks dann zu verwenden
  - bzw. Patches auszubauen, wenn diese unnötig sind

Hooks und ihre Code-Blöcke

Merkmale

- Ein Hook kann beliebig viele (beschriftete) Code-Blöcke enthalten
- Die Code-Blöcke können neu geordnet oder entfernt werden

Namen und Beschriftungen (labels)

- Hook-⟨*Namen*⟩ müssen im gesamten Dokument eindeutig sein
- Nur Code-Blöcke mit unterschiedlichen Beschriftungen können individuell manipuliert werden.

Verhalten

- Neue Hooks sind leer und verändern den Schriftsatz nicht
- Sie sind jedoch standardmäßig nicht vollständig transparent
- Für volle Transparenz, z.B. in `tabular`, wird eine spezielle Version von `\UseHook` benötigt (ohne Debug-Informationen)

Hooks und ihre Code-Blöcke

Merkmale

- Ein Hook kann beliebig viele (beschriftete) Code-Blöcke enthalten
- Die Code-Blöcke können neu geordnet oder entfernt werden

Namen und Beschriftungen (labels)

- Hook-⟨*Namen*⟩ müssen im gesamten Dokument eindeutig sein
- Nur Code-Blöcke mit unterschiedlichen Beschriftungen können individuell manipuliert werden.

Verhalten

- Neue Hooks sind leer und verändern den Schriftsatz nicht
- Sie sind jedoch standardmäßig nicht vollständig transparent
- Für volle Transparenz, z.B. in `tabular`, wird eine spezielle Version von `\UseHook` benötigt (ohne Debug-Informationen)

Hooks und ihre Code-Blöcke

Merkmale

- Ein Hook kann beliebig viele (beschriftete) Code-Blöcke enthalten
- Die Code-Blöcke können neu geordnet oder entfernt werden

Namen und Beschriftungen (labels)

- Hook-⟨*Namen*⟩ müssen im gesamten Dokument eindeutig sein
- Nur Code-Blöcke mit unterschiedlichen Beschriftungen können individuell manipuliert werden.

Verhalten

- Neue Hooks sind leer und verändern den Schriftsatz nicht
- Sie sind jedoch standardmäßig nicht vollständig transparent
- Für volle Transparenz, z.B. in `tabular`, wird eine spezielle Version von `\UseHook` benötigt (ohne Debug-Informationen)

Key takeaways Hook-Management

Einfach und schnell

- Pakete können leicht Hooks anbieten, die folgendes ermöglichen:
  - Koordination mit anderen Paketen
  - sichere und kontrollierte Erweiterungen
  - einfache Benutzeranpassungen
- Wenn ein Hook unbenutzt ist, gibt es fast keinen Overhead

Zukünftige Verwendung wird unterstützt

- Man kann Code zu einem Hook hinzufügen, auch wenn er noch nicht existiert
- Das Paket, dass den Hook definiert, kann später geladen werden oder auch nicht

Key takeaways Hook-Management

Einfach und schnell

- Pakete können leicht Hooks anbieten, die folgendes ermöglichen:
  - Koordination mit anderen Paketen
  - sichere und kontrollierte Erweiterungen
  - einfache Benutzeranpassungen
- Wenn ein Hook unbenutzt ist, gibt es fast keinen Overhead

Zukünftige Verwendung wird unterstützt

- Man kann Code zu einem Hook hinzufügen, auch wenn er noch nicht existiert
- Das Paket, dass den Hook definiert, kann später geladen werden oder auch nicht

Key takeaways Hook-Management (Fortsetzung)

Verbesserte Kompatibilität

- Verschiedene Pakete können Code ohne Konflikte zu einem Hook hinzufügen
- Wenn Code-Blöcke in einer bestimmen Reihenfolge verarbeitet werden müssen, können dafür Regeln aufgestellt werden
- Diese Regeln können proaktiv definiert werden (d.h. auch für Blöcke die nicht verwendet werden)
- Es ist kein destruktives Patchen erforderlich

Hook-Beispiele

Etwas in den Seitenhintergrund einfügen

```
\AddToHookNext{shipout/background}
  {\put(.5\paperwidth,-.5\paperheight)%
      {\makebox(0,0)%
         {\includegraphics{figures/hummingbird.png}}}}
```

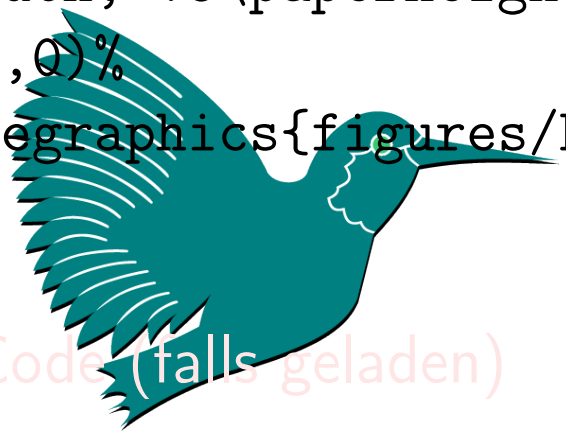
Patchen von Paket-Code (falls geladen)

```
\AddToHook{file/dinbrief.cls/after}[firstaid]
   {\FirstAidNeededT{dinbrief}{cls}%
                    {2000/03/02 LaTeX2e class}%
      {\AddToHook{env/document/begin}{\begingroup}}
```

Hook-Beispiele

Etwas in den Seitenhintergrund einfügen

```
\AddToHookNext{shipout/background}
  {\put(.5\paperwidth,-.5\paperheight)%
      {\makebox(0,0)%
         {\includegraphics{figures/hummingbird.png}}}}
```



Patchen von Paket-Code (falls geladen)

```
\AddToHook{file/dinbrief.cls/after}[firstaid]
   {\FirstAidNeededT{dinbrief}{cls}%
                    {2000/03/02 LaTeX2e class}%
      {\AddToHook{env/document/begin}{\begingroup}}
```

Hook-Beispiele

Etwas in den Seitenhintergrund einfügen

```
\AddToHookNext{shipout/background}
  {\put(.5\paperwidth,-.5\paperheight)%
      {\makebox(0,0)%
         {\includegraphics{figures/hummingbird.png}}}}
```

Patchen von Paket-Code (falls geladen)

```
\AddToHook{file/dinbrief.cls/after}[firstaid]
   {\FirstAidNeededT{dinbrief}{cls}%
                    {2000/03/02 LaTeX2e class}%
      {\AddToHook{env/document/begin}{\begingroup}}
```

Hook-Beispiele (Fortsetzung)

Mach mein Dokument bitte kürzer

```
\AddToHook{para/begin}{\looseness=-1 }

\newcommand\cancellooseness
  {\AddToHookNext{para/begin}{\looseness=0 }}
```

Notizen

- Versucht nicht, dasselbe mit `\looseness=1` zu tun
- Es wird zu vielen Absätzen mit nur einem einzigen Wort (oder schlimmer noch einem Teilwort) in der letzten Zeile führen!

Hook-Beispiele (Fortsetzung)

Mach mein Dokument bitte kürzer

```
\AddToHook{para/begin}{\looseness=-1 }

\newcommand\cancellooseness
  {\AddToHookNext{para/begin}{\looseness=0 }}
```

Notizen

- Versucht nicht, dasselbe mit `\looseness=1` zu tun
- Es wird zu vielen Absätzen mit nur einem einzigen Wort (oder schlimmer noch einem Teilwort) in der letzten Zeile führen!

Hook-Beispiele (Fortsetzung)

Geschachtelte Dateien (aus `structuredlog.sty`)

```
\AddToHook{file/before}
   { \__filehook_log_file_record:n { START } }
\AddToHookNext{file/after}
   { \AddToHook{file/after}
        { \__filehook_log_file_record:n { STOP } } }
```

Code-Blöcke in Hooks neu anordnen

```
\DeclareHookRule{begindocument}{showkeys}{before}{nameref}
```

Entfernen eines Codeabschnitts

```
\DeclareHookRule{enddocument/info}
    {kernel/testmode}{voids}{kernel/release}
```

Hook-Beispiele (Fortsetzung)

Geschachtelte Dateien (aus `structuredlog.sty`)

```
\AddToHook{file/before}
   { \__filehook_log_file_record:n { START } }
\AddToHookNext{file/after}
   { \AddToHook{file/after}
        { \__filehook_log_file_record:n { STOP } } }
```

Code-Blöcke in Hooks neu anordnen

```
\DeclareHookRule{begindocument}{showkeys}{before}{nameref}
```

Entfernen eines Codeabschnitts

```
\DeclareHookRule{enddocument/info}
      {kernel/testmode}{voids}{kernel/release}
```

Hook-Beispiele (Fortsetzung)

Geschachtelte Dateien (aus `structuredlog.sty`)

```
\AddToHook{file/before}
   { \__filehook_log_file_record:n { START } }
\AddToHookNext{file/after}
   { \AddToHook{file/after}
        { \__filehook_log_file_record:n { STOP } } }
```

Code-Blöcke in Hooks neu anordnen

```
\DeclareHookRule{begindocument}{showkeys}{before}{nameref}
```

Entfernen eines Codeabschnitts

```
\DeclareHookRule{enddocument/info}
      {kernel/testmode}{voids}{kernel/release}
```

Hook-Beispiele (Ein letztes Beispiel)

Frage (Einzug verhindern):

Wie kann ich verhindern, dass der erste Absatz jeder `multicols`-Umgebung einen Einzug hat?

Antwort:

- Am Beginn einer einer `multicols`-Umgebung tue etwas:
  `\AddToHook{env/multicols/begin}`
- Beim nächsten Absatz
  `{\AddToHook{para/begin}%`
- … unterdrücke den Einzug.
  `\noindent}`

Hook-Beispiele (Ein letztes Beispiel)

Frage (Einzug verhindern):

Wie kann ich verhindern, dass der erste Absatz jeder `multicols`-Umgebung einen Einzug hat?

Antwort:

- Am Beginn einer einer `multicols`- Umgebung tue etwas:

```
\AddToHook{env/multicols/begin}
```

- Beim nächsten Absatz ...

```
{\AddToHookNext{para/begin}%
```

- ... unterdrücke den Einzug:

```
{\OmitIndent}}
```

Hook-Beispiele (Ein letztes Beispiel)

Frage (Einzug verhindern):

Wie kann ich verhindern, dass der erste Absatz jeder `multicols`-Umgebung einen Einzug hat?

Antwort:

- Am Beginn einer einer `multicols`- Umgebung tue etwas:

```
\AddToHook{env/multicols/begin}
```

- Beim nächsten Absatz ...

```
{\AddToHookNext{para/begin}%
```

- ... unterdrücke den Einzug:

```
{\OmitIndent}}
```

Hook-Beispiele (Ein letztes Beispiel)

Frage (Einzug verhindern):

Wie kann ich verhindern, dass der erste Absatz jeder `multicols`*-Umgebung einen Einzug hat?*

Antwort:

- Am Beginn einer einer `multicols`- Umgebung tue etwas:

```
\AddToHook{env/multicols/begin}
```

- Beim nächsten Absatz ...

```
{\AddToHookNext{para/begin}%
```

- ... unterdrücke den Einzug:

```
{\OmitIndent}}
```

Was sind Steckdosen (sockets)?

Und was sind Stecker (plugs)?

Und was sind ihre Eigenschaften?

Code-Sockets und Code-Plugs

Charakteristika

- In einen Socket kann immer nur ein Plug eingesteckt werden
- In Analogie dazu kann der Socket-Code ersetzt, aber nicht erweitert werden

LaTeX Sockets und Plugs

- Ein Socket definiert einen (benannten) Ort im Code, an dem eine Auswahl von Alternativen eingesteckt "pluged in" werden kann
- Die Alternativen für einen Socket werden daher als seine "plugs" bezeichnet
- Jeder Socket und seine Auswahl an Plugs muss vor der Verwendung deklariert werden

Code-Sockets und Code-Plugs

Charakteristika

- In einen Socket kann immer nur ein Plug eingesteckt werden
- In Analogie dazu kann der Socket-Code ersetzt, aber nicht erweitert werden

LaTeX Sockets und Plugs

- Ein Socket definiert einen (benannten) Ort im Code, an dem eine Auswahl von Alternativen eingesteckt "pluged in" werden kann
- Die Alternativen für einen Socket werden daher als seine "plugs" bezeichnet
- Jeder Socket und seine Auswahl an Plugs muss vor der Verwendung deklariert werden

Code-Sockets und Code-Plugs (Fortsetzung)

Namen

- Wie Hooks müssen auch Sockets (d.h. ihre ⟨*Namen*⟩) im gesamten Dokument eindeutig sein
- Plug-⟨*Namen*⟩ müssen pro Socket eindeutig sein

Voreingestellte Plugs

- Jeder neue Socket hat den Plug `noop` eingesteckt. Das bedeutet, dass der Socket ignoriert wird (mit seinen Argumenten, falls vorhanden)
- Ausnahme: ein neuer Socket mit genau einem Argument hat den Plug `identity` eingesteckt, so dass sein Argument verarbeitet wird (nach Entfernen der äußeren Klammern)

Code-Sockets und Code-Plugs (Fortsetzung)

Namen

- Wie Hooks müssen auch Sockets (d.h. ihre ⟨*Namen*⟩) im gesamten Dokument eindeutig sein
- Plug-⟨*Namen*⟩ müssen pro Socket eindeutig sein

Voreingestellte Plugs

- Jeder neue Socket hat den Plug `noop` eingesteckt. Das bedeutet, dass der Socket ignoriert wird (mit seinen Argumenten, falls vorhanden)
- Ausnahme: ein neuer Socket mit genau einem Argument hat den Plug `identity` eingesteckt, so dass sein Argument verarbeitet wird (nach Entfernen der äußeren Klammern)

Haken oder Steckdose? — Wann verwendet man was?

Hooks verwenden

- an Stellen, an denen eine (allgemeine) Initialisierung stattfinden kann
- wenn Ergänzungen aus unterschiedlichen Paketen wahrscheinlich sinnvoll sind

Sockets verwenden

- wenn der verwendete Code streng kontrolliert werden muss
- in typischen "on/off" Situationen
- wenn verschiedene Verarbeitungsmodelle unterstützt werden sollen (das heißt, wenn ein Algorithmus durch einen anderen ersetzt wird)

Haken oder Steckdose? — Wann verwendet man was?

Hooks verwenden

- an Stellen, an denen eine (allgemeine) Initialisierung stattfinden kann
- wenn Ergänzungen aus unterschiedlichen Paketen wahrscheinlich sinnvoll sind

Sockets verwenden

- wenn der verwendete Code streng kontrolliert werden muss
- in typischen “on/off” Situationen
- wenn verschiedene Verarbeitungsmodelle unterstützt werden sollen (das heißt, wenn ein Algorithmus durch einen anderen ersetzt wird)

Zwei Socket-Beispiele

Seitenaufbau (Fußnoten/Float-Platzierung)

`\AssignSocketPlug{build/column/outputbox}{`⟨Plug⟩`}`

- Vorhandene Plugs:
  - `space-footnotes-floats`
  - `footnotes-space-floats`
  - `floats-space-footnotes`
  - `floats-footnotes`
  - `footnotes-floats` (neue Voreinstellung)
  - `footnotes-floats-legacy` (alte Voreinstellung)

Abstandsmessung auf Seiten

`\AssignSocketPlug{build/column/baselineattach}{`⟨Plug⟩`}`

- Vorhandene Plugs: `on` und `off`

Zwei Socket-Beispiele

Seitenaufbau (Fußnoten/Float-Platzierung)

`\AssignSocketPlug{build/column/outputbox}{`⟨Plug⟩`}`

- Vorhandene Plugs:
  - `space-footnotes-floats`
  - `footnotes-space-floats`
  - `floats-space-footnotes`
  - `floats-footnotes`
  - `footnotes-floats` (neue Voreinstellung)
  - `footnotes-floats-legacy` (alte Voreinstellung)

Abstandsmessung auf Seiten

`\AssignSocketPlug{build/column/baselineattach}{`⟨Plug⟩`}`

- Vorhandene Plugs: `on` und `off`

Zwei Socket-Beispiele

Seitenaufbau (Fußnoten/Float-Platzierung)

`\AssignSocketPlug{build/column/outputbox}{`⟨Plug⟩`}`

- Vorhandene Plugs:
  - `space-footnotes-floats`
  - `footnotes-space-floats`
  - `floats-space-footnotes`
  - `floats-footnotes`
  - `footnotes-floats` (neue Voreinstellung)
  - `footnotes-floats-legacy` (alte Voreinstellung)

Abstandsmessung auf Seiten

`\AssignSocketPlug{build/column/baselineattach}{`⟨Plug⟩`}`

- Vorhandene Plugs: `on` und `off`

Tagging Sockets (Steckdosen für barrierefreie Dokumente)

Eigenschaften

- Tagging Sockets sind normale Sockets mit ein paar Besonderheiten
- Ihre Namen starten immer mit `tagsupport/`
- Sie haben 0, 1 oder 2 Argumente
- Falls vorhanden
  - konfiguriert das erste Argument den Plug
  - und das zweite erhält Material, das "getaggt" werden soll

Verwendung

- `\UseTaggingSocket`
- `\SuspendTagging` und `\ResumeTagging`

Tagging Sockets (Steckdosen für barrierefreie Dokumente)

Eigenschaften

- Tagging Sockets sind normale Sockets mit ein paar Besonderheiten
- Ihre Namen starten immer mit `tagsupport/`
- Sie haben 0, 1 oder 2 Argumente
- Falls vorhanden
  - konfiguriert das erste Argument den Plug
  - und das zweite erhält Material, das "getaggt" werden soll

Verwendung

- `\UseTaggingSocket`
- `\SuspendTagging` und `\ResumeTagging`

Tagging Sockets (Fortsetzung)

Voreingestellte Plugs

- Jeder neue Tagging Socket ohne oder mit einem Argument hat den Plug `noop` eingesteckt
- Bei Tagging sockets mit zwei Argumenten ist der Plug transparent voreingestellt:
  - das erste Argument wird ignoriert und das zweite verarbeitet (nach Entfernen der äußeren Klammern)

Deklaration

- Wegen der Besonderheiten gibt es eigene Kommandos zum Deklarieren und Verwenden der Tagging Sockets:
  - `\NewTaggingSocket` und `\NewTaggingSocketPlug`
  - `\AssignTaggingSocketPlug`

Tagging Sockets (Fortsetzung)

Voreingestellte Plugs

- Jeder neue Tagging Socket ohne oder mit einem Argument hat den Plug `noop` eingesteckt
- Bei Tagging sockets mit zwei Argumenten ist der Plug transparent voreingestellt:
  - das erste Argument wird ignoriert und das zweite verarbeitet (nach Entfernen der äußeren Klammern)

Deklaration

- Wegen der Besonderheiten gibt es eigene Kommandos zum Deklarieren und Verwenden der Tagging Sockets:
  - `\NewTaggingSocket` und `\NewTaggingSocketPlug`
  - `\AssignTaggingSocketPlug`

Final advice — seit vorsichtig mit der Verkabelung

- . . . sonst wissen eure Benutzer nicht, wie man es benutzt
- Und übertreibt es nicht — sonst wird es zu langsam

Final advice — seit vorsichtig mit der Verkabelung

- . . . sonst wissen eure Benutzer nicht, wie man es benutzt
- Und übertreibt es nicht — sonst wird es zu langsam

Time Check

Die Details (wenn es die Zeit erlaubt) — sonst . . .

Dokumentation für Hooks

- `texdoc lthooks-doc` Hauptdokumentation
- `texdoc ltcmdhooks-doc` generische cmd/env-Haken
- Ergänzende Dokumentation in
  - `ltfilehook-doc`,
  - `ltmarks-doc`,
  - `ltpara-doc`,
  - `ltshipout-doc`

Dokumentation für Sockets und Plugs

- `texdoc ltsockets-doc` Hauptdokumentation

Die Details (wenn es die Zeit erlaubt) — sonst . . .

Dokumentation für Hooks

- `texdoc lthooks-doc` Hauptdokumentation
- `texdoc ltcmdhooks-doc` generische cmd/env-Haken
- Ergänzende Dokumentation in
  - `ltfilehook-doc`,
  - `ltmarks-doc`,
  - `ltpara-doc`,
  - `ltshipout-doc`

Dokumentation für Sockets und Plugs

- `texdoc ltsockets-doc` Hauptdokumentation

The new hook management

Declaring hooks

- `\NewHook{`⟨*name*⟩`}`
- `\NewReversedHook{`⟨*name*⟩`}`
- `\NewHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}`
- . . . plus a few more

Notes

- ⟨*name*⟩ has to be unique
  Best practice: ⟨*name*⟩ = ⟨*pkg*⟩ / ⟨*identifier*⟩
- Reversed hooks have the code chunks backwards
- ⟨*number*⟩ is the number of arguments

The new hook management

Declaring hooks

- `\NewHook{`⟨*name*⟩`}`
- `\NewReversedHook{`⟨*name*⟩`}`
- `\NewHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}`
- . . . plus a few more

Notes

- ⟨*name*⟩ has to be unique
  Best practice: ⟨*name*⟩ = ⟨*pkg*⟩ / ⟨*identifier*⟩
- Reversed hooks have the code chunks backwards
- ⟨*number*⟩ is the number of arguments

The new hook management

Using hooks

- `\UseHook{`⟨*name*⟩`}`
- `\UseOneTimeHook{`⟨*name*⟩`}`

Using hooks with arguments

- `\UseHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`
- `\UseOneTimeHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`

Notes

- Note that the ⟨*number*⟩ of arguments has to be explicitly given for hooks with arguments (for efficiency reasons)
- If a hook is empty it will be therefore bypassed with little overhead

The new hook management

Using hooks

- `\UseHook{`⟨*name*⟩`}`
- `\UseOneTimeHook{`⟨*name*⟩`}`

Using hooks with arguments

- `\UseHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`
- `\UseOneTimeHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`

Notes

- Note that the ⟨*number*⟩ of arguments has to be explicitly given for hooks with arguments (for efficiency reasons)
- If a hook is empty it will be therefore bypassed with little overhead

The new hook management

Using hooks

- `\UseHook{`⟨*name*⟩`}`
- `\UseOneTimeHook{`⟨*name*⟩`}`

Using hooks with arguments

- `\UseHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`
- `\UseOneTimeHookWithArguments{`⟨*name*⟩`}{`⟨*number*⟩`}{`⟨*...*⟩`}...`

Notes

- Note that the ⟨*number*⟩ of arguments has to be explicitly given for hooks with arguments (for efficiency reasons)
- If a hook is empty it will be therefore bypassed with little overhead

The new hook management

Adding code to hooks

- `\AddToHook{`⟨*name*⟩`}[`⟨*label*⟩`]{`⟨*code*⟩`}`
- `\AddToHookNext{`⟨*name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*label*⟩ identifies the code chunk
  default: package/class name; on document-level: `toplevel`
- You can use both commands with hooks taking arguments
  (if you are not referring to them)
- You can add to a hook that is not yet declared!
- If you add to a one-time hook after it was used, then
  ⟨*code*⟩ is used immediately

The new hook management

Adding code to hooks

- `\AddToHook{`⟨*name*⟩`}[`⟨*label*⟩`]{`⟨*code*⟩`}`
- `\AddToHookNext{`⟨*name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*label*⟩ identifies the code chunk
  default: package/class name; on document-level: `toplevel`
- You can use both commands with hooks taking arguments
  (if you are not referring to them)
- You can add to a hook that is not yet declared!
- If you add to a one-time hook after it was used, then
  ⟨*code*⟩ is used immediately

The new hook management

Adding code to hooks with arguments

- `\AddToHookWithArguments{`⟨*name*⟩`}[`⟨*label*⟩`]{`⟨*code*⟩`}`
- `\AddToHookNextWithArguments{`⟨*name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*code*⟩ can contain #1, #2, . . .
- Real #'s have to be doubled, i.e., entered as ##
- You can't add to a one-time hook this way after it was used

The new hook management

Adding code to hooks with arguments

- `\AddToHookWithArguments{`⟨*name*⟩`}[`⟨*label*⟩`]{`⟨*code*⟩`}`
- `\AddToHookNextWithArguments{`⟨*name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*code*⟩ can contain #1, #2, ...
- Real #'s have to be doubled, i.e., entered as ##
- You can't add to a one-time hook this way after it was used

The new hook management

Remove code from hooks

- `\RemoveFromHook{`⟨*name*⟩`}[`⟨*label*⟩`]`

Notes

- Without the optional argument the default ⟨*label*⟩ is used
- Special case: `[*]` remove all code (naughty)

The new hook management

Remove code from hooks

- `\RemoveFromHook{`⟨*name*⟩`}[`⟨*label*⟩`]`

Notes

- Without the optional argument the default ⟨*label*⟩ is used
- Special case: `[*]` remove all code (naughty)

The new hook management

Displaying the hook status

- `\ShowHook{`⟨*name*⟩`}` or `\LogHook{`⟨*name*⟩`}`

Output example

```
-> The hook 'enddocument':
> Code chunks:
>     pgfcore -> \ifpgf@external@grabshipout ...
>     beamerbasemisc -> \clearpage  ...
>     csquotes -> \ifnum \csq@qlevel >\z@ \csq@err@gleft \fi
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order:
>     pgfcore, beamerbasemisc, csquotes.
```

The new hook management

Displaying the hook status

- `\ShowHook{`⟨*name*⟩`}` or `\LogHook{`⟨*name*⟩`}`

Output example

```
-> The hook 'enddocument':
> Code chunks:
>     pgfcore -> \ifpgf@external@grabshipout ...
>     beamerbasemisc -> \clearpage  ...
>     csquotes -> \ifnum \csq@qlevel >\z@ \csq@err@gleft \fi
> Document-level (top-level) code (executed last):
>     ---
> Extra code for next invocation:
>     ---
> Rules:
>     ---
> Execution order:
>     pgfcore, beamerbasemisc, csquotes.
```

The new socket management

Declaring sockets and plugs

- `\NewSocket{`⟨*socket-name*⟩`}{`⟨*number-of inputs*⟩`}`
- `\NewSocketPlug{`⟨*socket-name*⟩`}`
  `{`⟨*socket-plug-name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*socket-name*⟩ has to be unique
  Best practice: ⟨*name*⟩ = ⟨*pkg*⟩ / ⟨*identifier*⟩
- ⟨*socket-plug-name*⟩ has to be unique per socket
  but can be reused in different sockets, e.g., `noop`

The new socket management

Declaring sockets and plugs

- `\NewSocket{`⟨*socket-name*⟩`}{`⟨*number-of inputs*⟩`}`
- `\NewSocketPlug{`⟨*socket-name*⟩`}`
  `{`⟨*socket-plug-name*⟩`}{`⟨*code*⟩`}`

Notes

- ⟨*socket-name*⟩ has to be unique
  Best practice: ⟨*name*⟩ = ⟨*pkg*⟩ / ⟨*identifier*⟩
- ⟨*socket-plug-name*⟩ has to be unique per socket
  but can be reused in different sockets, e.g., `noop`

The new socket management

Assigning plugs to sockets

- `\AssignSocketPlug{`⟨*socket-name*⟩`}{`⟨*socket-plug-name*⟩`}`
- Default assignments are
  - `identity` for sockets with one argument
  - `noop` for all others

Showing sockets

- `\ShowSocket{`⟨*socket-name*⟩`}` or `\LogSocket{`⟨*socket-name*⟩`}`

Using sockets

- `\UseSocket{`⟨*socket-name*⟩`}`
  - Number of arguments is implicit in LaTeX 2ε but explicit in L3 layer, e.g., `\socket_use:nn`

The new socket management

Assigning plugs to sockets

- `\AssignSocketPlug{`⟨*socket-name*⟩`}{`⟨*socket-plug-name*⟩`}`
- Default assignments are
  - `identity` for sockets with one argument
  - `noop` for all others

Showing sockets

- `\ShowSocket{`⟨*socket-name*⟩`}` or `\LogSocket{`⟨*socket-name*⟩`}`

Using sockets

- `\UseSocket{`⟨*socket-name*⟩`}`
  - Number of arguments is implicit in LaTeX 2_ε but explicit in L3 layer, e.g., `\socket_use:nn`

The new socket management

Assigning plugs to sockets

- `\AssignSocketPlug{`⟨*socket-name*⟩`}{`⟨*socket-plug-name*⟩`}`
- Default assignments are
  - `identity` for sockets with one argument
  - `noop` for all others

Showing sockets

- `\ShowSocket{`⟨*socket-name*⟩`}` or `\LogSocket{`⟨*socket-name*⟩`}`

Using sockets

- `\UseSocket{`⟨*socket-name*⟩`}`
  - Number of arguments is implicit in LaTeX 2_ε but explicit in L3 layer, e.g., `\socket_use:nn`