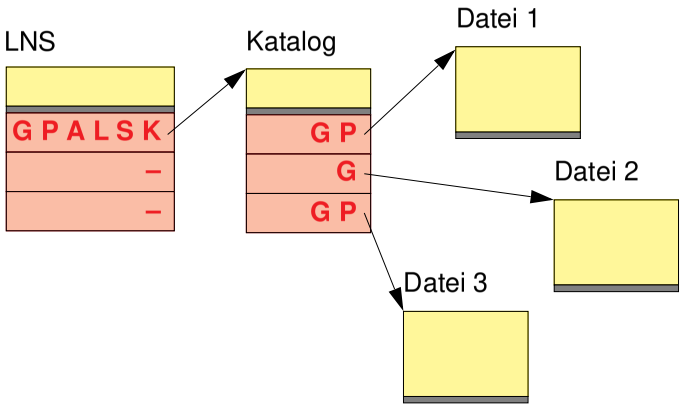
3 Zugriff auf Capabilities in Hydra (2)

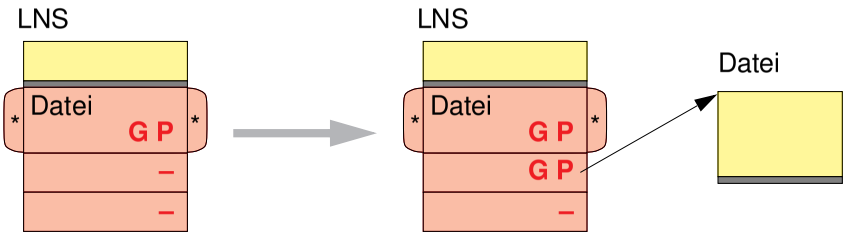
- Beispiel: Implementierung von Katalogen
  Katalog = Objekt, das Referenzen (Capabilities) auf Dateien enthält
  - *Load* erlaubt das Auflösen von Namen / Öffnen einer Datei (Aufrufer bekommt die Capability)
  - *Store* und *Append* erlauben das Hinzufügen von Dateien zum Katalog
  - *Delete* erlaubt das Austragen von Dateien aus dem Katalog



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Objekterzeugung in Hydra

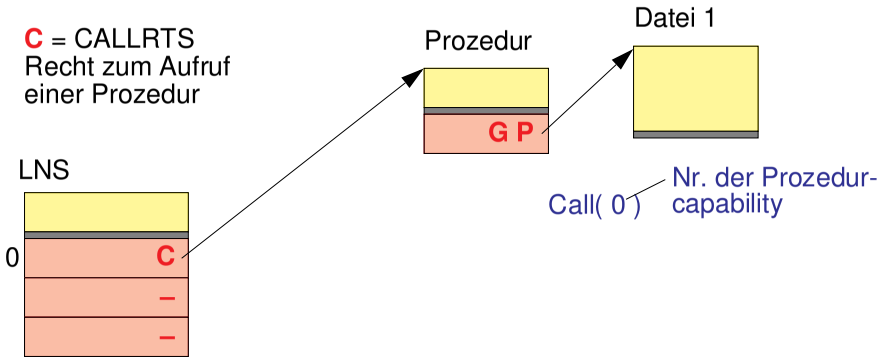
- Objekterzeugung über Erzeugungsschablonen *(Creation Templates)*
  - mit der *create*-Operation können neue Objekte eines Typs erzeugt werden
  - LNS benötigt hierzu eine Erzeugungsschablone
  - Erzeugungsschablone enthält den Typ des neu zu erzeugenden Objektes und eine Rechtemaske
  - der Aufrufer der *create*-Operation erhält eine Capability auf das neu erzeugte Objekt, die nur die in der Maske angeschalteten Rechte enthält



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra

- Prozedur ist ein Objekt, aus dem beim Aufruf ein neuer LNS im laufenden Prozess erzeugt wird
  - ◆ neuer LNS wird aktueller Kontext (alte LNS stehen auf einem Stack; sie werden wieder aktiviert, wenn Prozedur zu Ende)
- Aufruf einer Prozedur



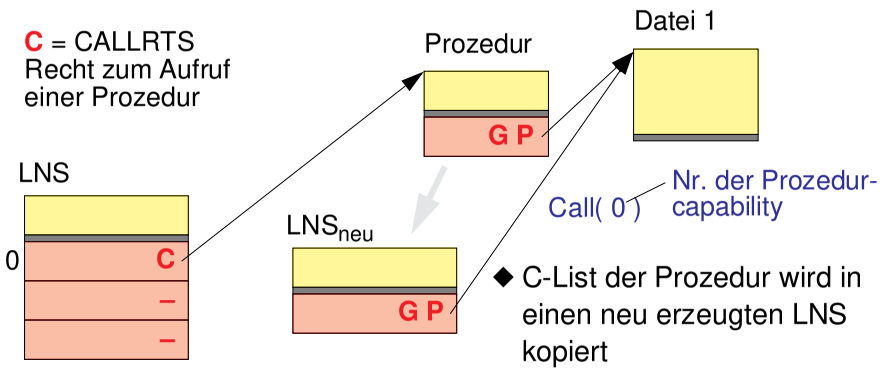
➤ Prozedur kann initiale Capabilities haben

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.4 Konzeptionelle Grundlagen von Capabilities

5 Prozeduraufruf in Hydra

- Prozedur ist ein Objekt, aus dem beim Aufruf ein neuer LNS im laufenden Prozess erzeugt wird
  - ◆ neuer LNS wird aktueller Kontext (alte LNS stehen auf einem Stack; sie werden wieder aktiviert, wenn Prozedur zu Ende)
- Aufruf einer Prozedur

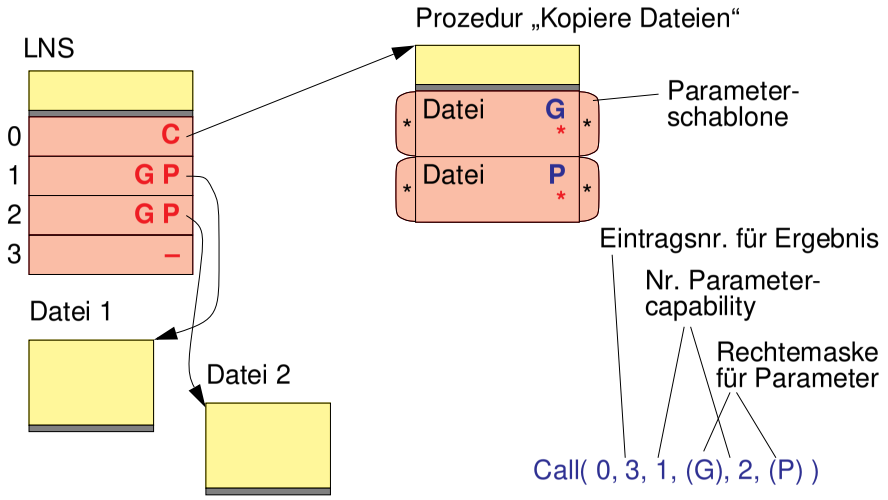


➤ Prozedur kann initiale Capabilities haben
... und weitere durch Parameter übergeben bekommen

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (2)

- Übergabe von Parametern
  - Beispiel: Prozedur zum Kopieren von Dateiinhalten

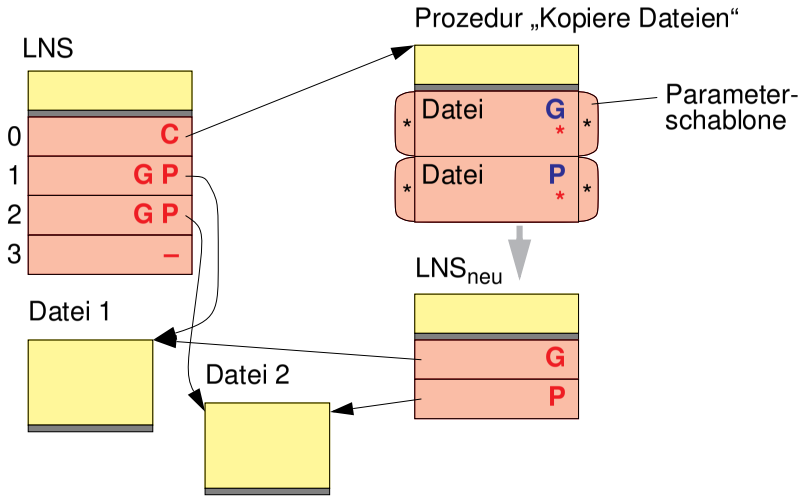


➤ Parameterschablone zum Überprüfen des Typs des Parameters und der mindestens erforderlichen Rechte

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (3)

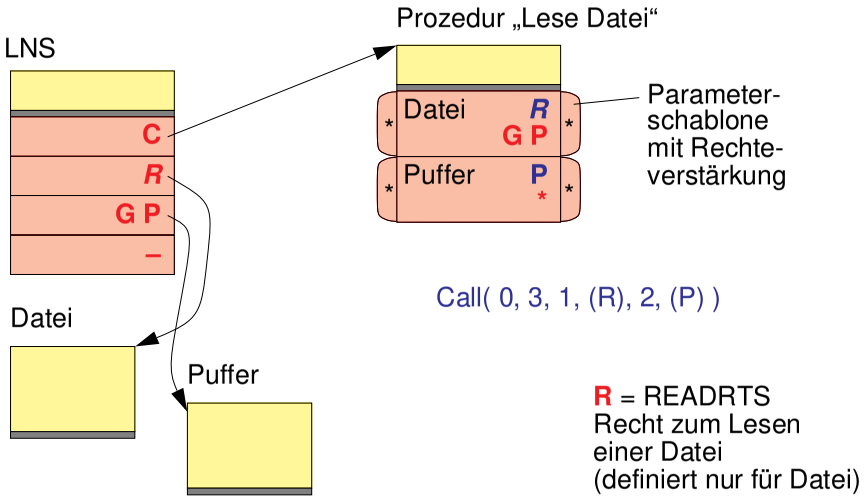
- Übergabe von Parametern
  - Beispiel: Prozedur zum Kopieren von Dateiinhalten



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeduraufruf in Hydra (3)

- Verstärken von Rechten
  - Beispiel: Prozedur zum Lesen von Dateiinhalten in einen Puffer



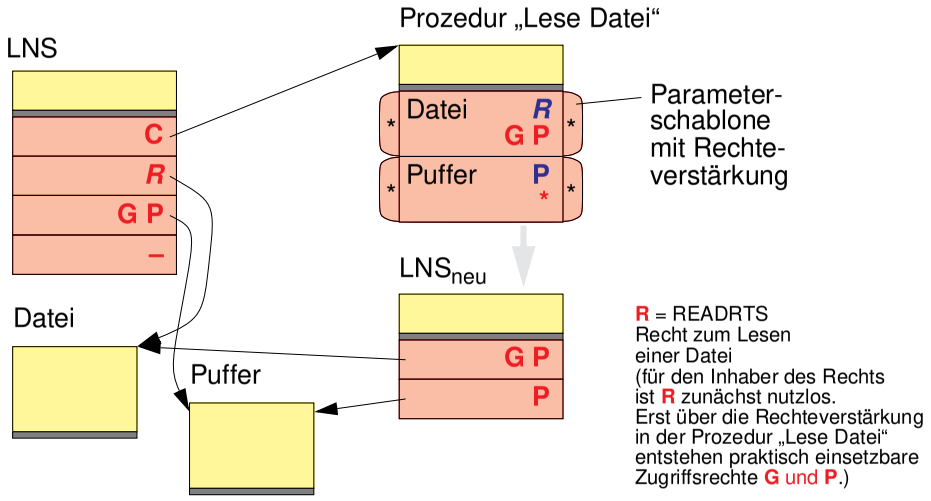
➤ Prozedur kann mehr Rechte besitzen, als der Aufrufer der Prozedur (analog zu s-bit bei UNIX-Programmen)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.4 Konzeptionelle Grundlagen von Capabilities

5 Prozeduraufruf in Hydra (3)

- Verstärken von Rechten
  - Beispiel: Prozedur zum Lesen von Dateiinhalten in einen Puffer



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Problem: Gegenseitiges Misstrauen

- Aufrufer misstraut einer Prozedur
  - Aufrufer möchte der Prozedur nur soviel Rechte einräumen wie nötig
- Aufgerufene Prozedur misstraut dem Aufrufer
  - Aufrufer soll nur soviel Rechte und Zugang bekommen wie erforderlich

★ Hydra Prozeduraufruf unterstützt diese Forderungen direkt

- Aufrufer übergibt Capabilities, die nötig sind
- Aufrufer kann Rechte bei der Übergabe maskieren und damit ausschalten
- Aufrufer erhält nur Zugang zu einem definierten Ergebnis
- Prozedur kann eigene Capabilities besitzen, die einem LNS zur Verfügung stehen und die dem Aufrufer verborgen bleiben können

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Problem: Gegenseitiges Misstrauen (2)

▲ Rechteverstärkung als Sicherheitslücke?

- Verstärkungsschablone wird nur an vertrauenswürdige Prozeduren ausgegeben und kann nicht einfach erzeugt werden

7 Problem: Modifikationen

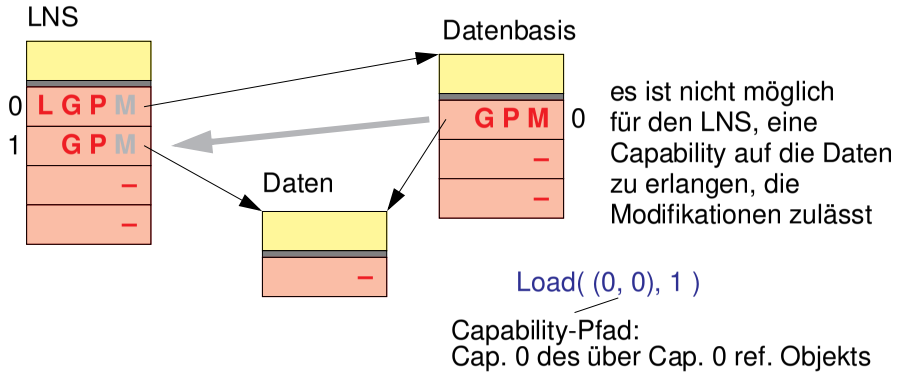
- Aufrufer möchte Modifikationen an und über Parameter ausschließen
  - eine Prozedur soll nichts verändern können
- Wegnehmen der entsprechenden Rechte reicht nicht
  - Prozedur kann lesend zu neuen Capabilities gelangen und über diese Änderungen vornehmen (Transitivität)
  - Rechteverstärkung könnte angewandt werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Problem: Modifikation (2)

★ Einführung des Modifikationsrechts ***M***DFYRTS

- ◆ für alle modifizierenden Operationen an Datenbereichen und C-Lists muss zusätzlich das Modifikationsrecht für das Objekt vorhanden sein
- ◆ Modifikationsrecht wird automatisch gelöscht, wenn eine Capability über einen Pfad geladen wird, auf dem eine der Capabilies kein Modifikationsrecht besitzt
- ◆ Modifikationsrecht kann nicht über Rechteverstärkung erlangt werden



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Problem: Modifikation (3)

- Parameterübergabe
  - ◆ Wegnahme des Modifikationsrecht bei Parametern stellt sicher, dass die aufgerufene Prozedur keinerlei Veränderungen beim Aufrufer durchführen kann

8 Problem: Ausbreitung von Capabilities

- Aufrufer will verhindern, dass eine übergebene Capability vom Aufgerufenen an einen Dritten weitergegeben wird *(Propagation Problem)*
  - ◆ Beispiel: Prozedur „Drucken“ soll niemandem eine Referenz auf die zu druckenden Daten weitergeben können

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Problem: Ausbreitung von Capabilities (2)

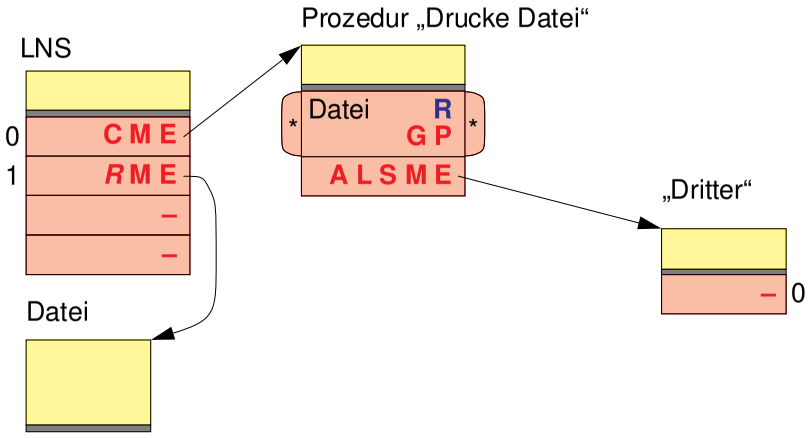
★ Einführung des Environment-Rechts *ENVRTS*

- ◆ für das Speichern oder Anfügen einer Capability an eine C-List muss die zu speichernde Capability selbst das Environment-Recht besitzen
- ◆ Environment-Recht wird automatisch gelöscht, wenn eine Capability über einen Pfad geladen wird, auf dem eine der Capabilities kein Environment-Recht besitzt
- ◆ Environment-Recht kann nicht über Rechteverstärkung erlangt werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Problem: Ausbreitung von Capabilities (3)

■ Versuchte Weitergabe einer Capability an einen Dritten

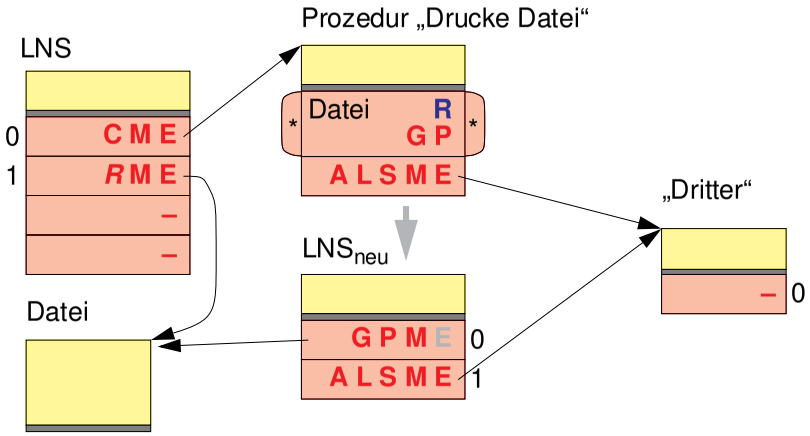


Call(0, 2, 1, (–E))

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Problem: Ausbreitung von Capabilities (3)

- Versuchte Weitergabe einer Capability an einen Dritten

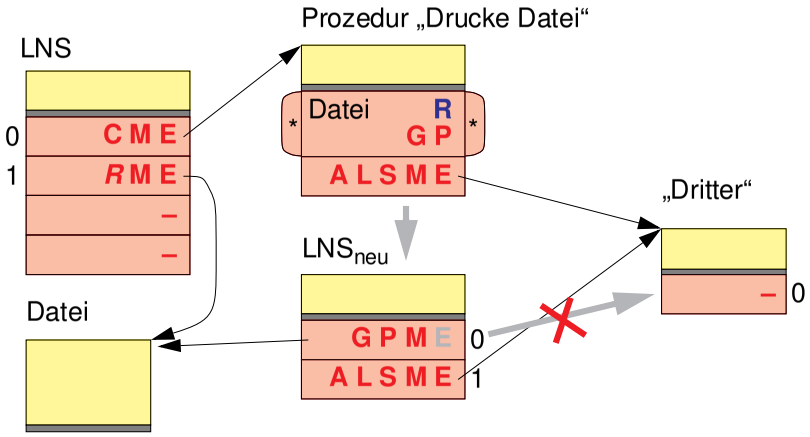


Call(0, 2, 1, (–E))

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Problem: Ausbreitung von Capabilities (3)

- Versuchte Weitergabe einer Capability an einen Dritten



Call(0, 2, 1, (–E)) Store(0, (1, 0))

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Problem: Aufbewahrung von Capabilities

- Aufrufer möchte sicher sein, dass Aufgerufener keine Capabilities nach der Bearbeitung des Aufrufs zurückbehalten kann *(Conservation Problem)*

★ Environment-Recht zusammen mit dem Aufrufmechanismus genügt

- ◆ Aufgerufener kann Capability ohne ENVRTS nicht weitergeben und folglich nicht abspeichern
- ◆ der LNS des Aufrufs wird mit Beendigung des Aufrufs vernichtet, so dass die übergebenen Capabilities nicht zurückbehalten werden können
- ◆ ENVRTS wirkt transitiv, so dass auch die über eine Parameter-Capability gewonnenen Capabilities nicht weitergegeben werden können

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

10 Problem: Informationsflussbegrenzung

- Aufrufer möchte die Verbreitung von Informationen aus übergebenen Parametern einschränken *(Confinement Problem)*
  - ◆ selektiv: bestimmte Informationen sollen nicht nach außen gelangen
  - ◆ global: gar keine Informationen sollen nach außen gelangen
  - ◆ ENVRTS ist nicht ausreichend, da Prozedur den Dateninhalt von Parameterobjekten kopieren könnte
    (ENVRTS wirkt nur auf die Weitergabe von Capabilities)
- Hydra realisiert nur globale Informationsflussbegrenzung

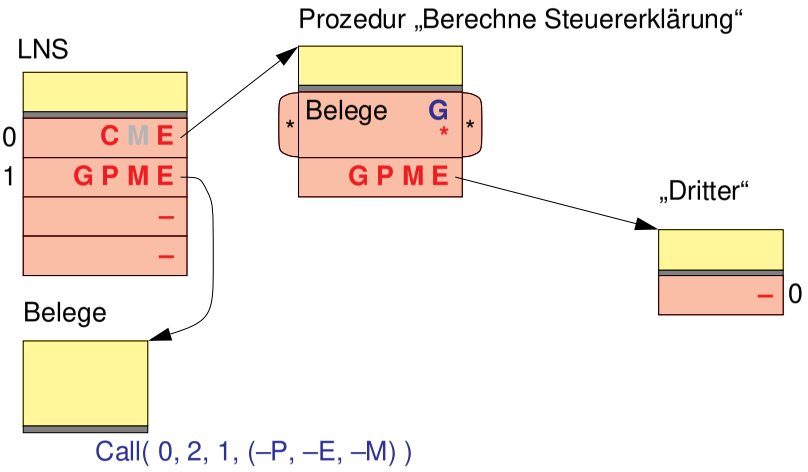
★ Modifikationsrecht auf der Prozedur-Capability

- ◆ wenn kein Modifikationsrecht vorhanden ist, werden bei allen in den LNS übernommenen Capabilities die Modifikationsrechte ausgeschaltet
  (gilt jedoch nicht für Parameter)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

10 Problem: Informationsflussbegrenzung (2)

- Beispiel: Prozedur zur Steuerberechnung
  - die übergebenen Beleg- und Buchhaltungsdaten sollen nicht weitergegeben werden können

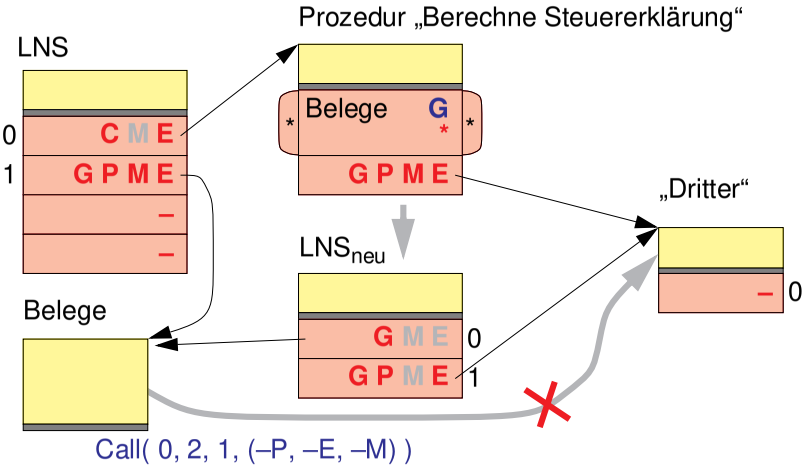


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.4 Konzeptionelle Grundlagen von Capabilities

10 Problem: Informationsflussbegrenzung (2)

- Beispiel: Prozedur zur Steuerberechnung
  - die übergebenen Beleg- und Buchhaltungsdaten sollen nicht weitergegeben werden können



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

11 Problem: Initialisierung

- Initialisierung von Objekten durch Prozeduren
  - Übergabe eines Objekts und verschiedener Capabilities, mit denen das Objekt initialisiert werden soll
  - Problem: Parameter-Capabilities müssen Environment-Recht besitzen (sonst ist das zu initialisierende Objekt nicht arbeitsfähig), gleichzeitig soll aber die Ausbreitung solcher Capabilities eingeschränkt werden
    - Lösung: Wegnahme des Modifikationsrechts auf der Prozedurcapability
  - Problem: Es muss verhindert werden, dass die Prozedur in das zu initialisierende Objekt eigene oder fremde Capabilities einsetzt, so dass es später Einfluss auf das zu initialisierende Objekt nehmen kann
- Beispiel: Prozedur zur Initialisierung eines Katalogs bekommt Capabilities auf die entsprechenden Dateien
  - es soll sichergestellt werden, dass Prozedur keine eigenen Dateicapabilities in den Katalog einfügt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

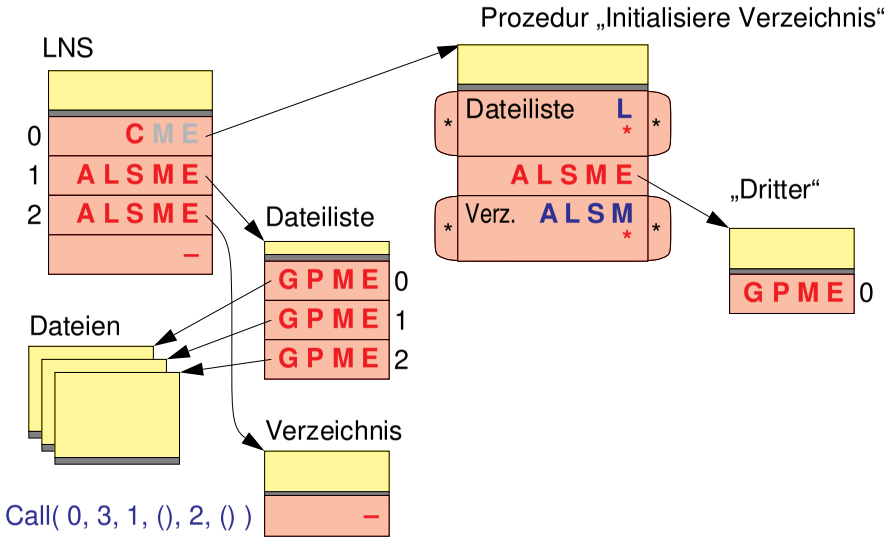
11 Problem Initialisierung (2)

- ★ Environment-Recht auf der Prozedur-Capability
  - wenn kein Environment-Recht vorhanden ist, werden bei allen in den LNS übernommenen Capabilities die Environment-Rechte ausgeschaltet (gilt jedoch nicht für Parameter)
  - durch das fehlende Environment-Recht können alle bereits vorhandenen Capabilities nicht in das zu initialisierende Objekt gespeichert werden

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

11 Problem: Initialisierung (3)

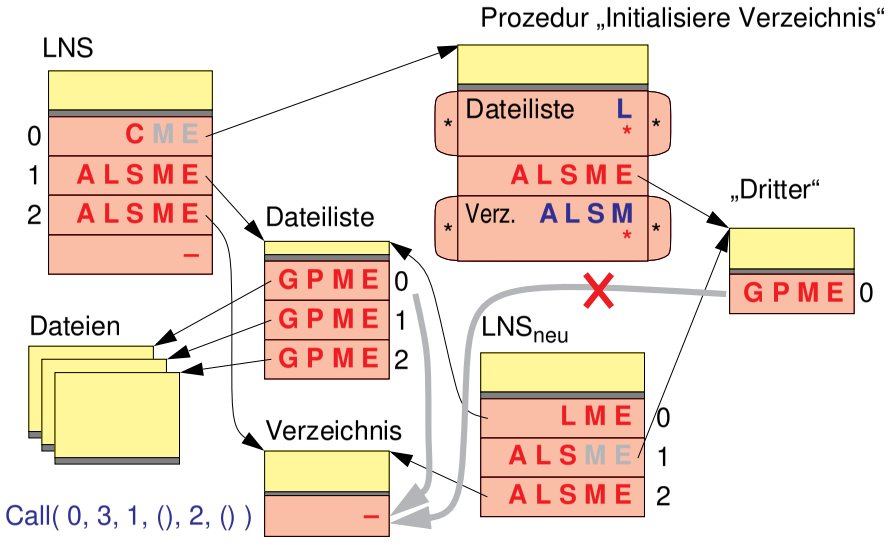
- Beispiel: Initialisierung eines Verzeichnis



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

11 Problem: Initialisierung (3)

- Beispiel: Initialisierung eines Verzeichnis



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.