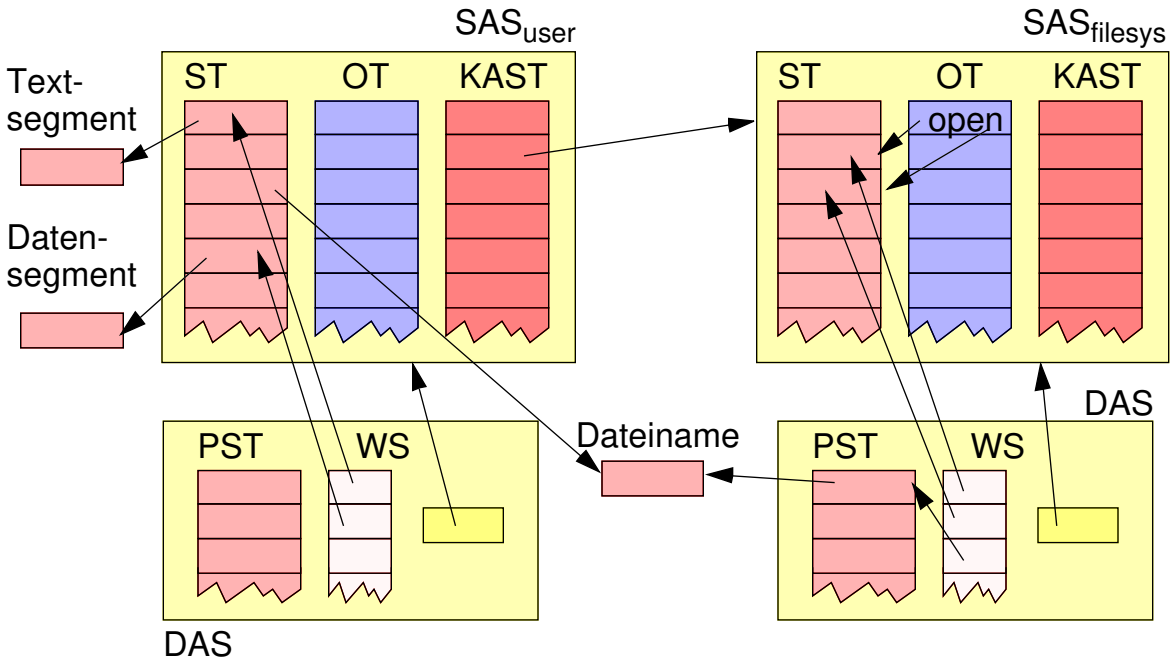
# 2 Beispielaufruf (2)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

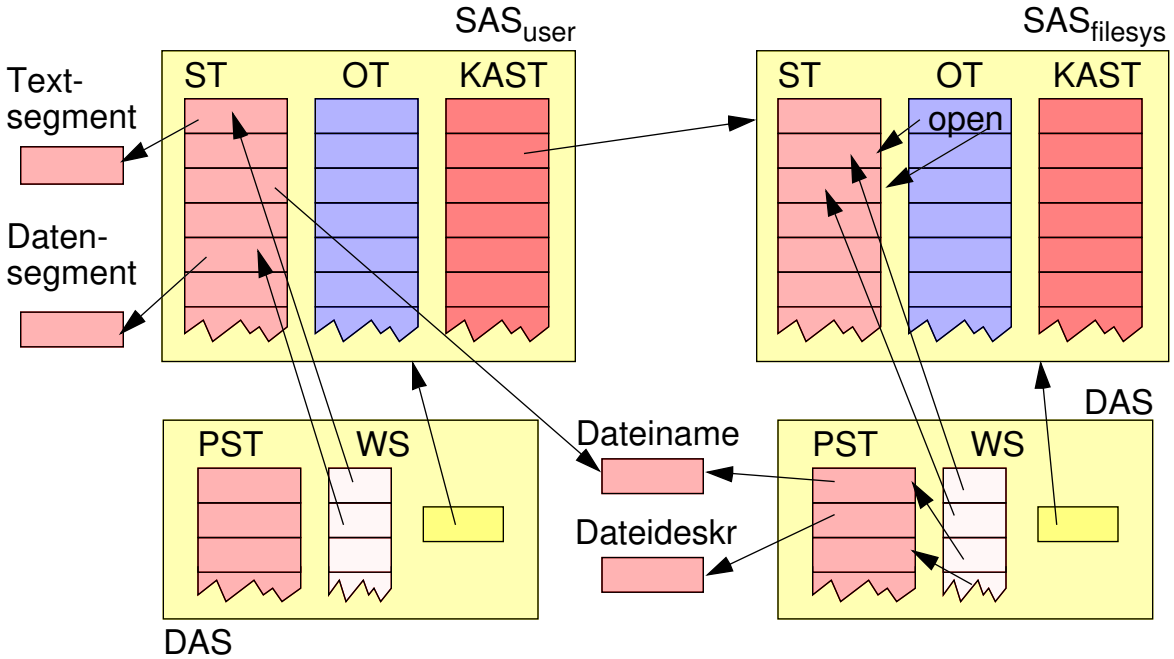



  - für den Aufruf von „open“ wird ein neues DAS erzeugt


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

# 2 Beispielaufruf (3)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

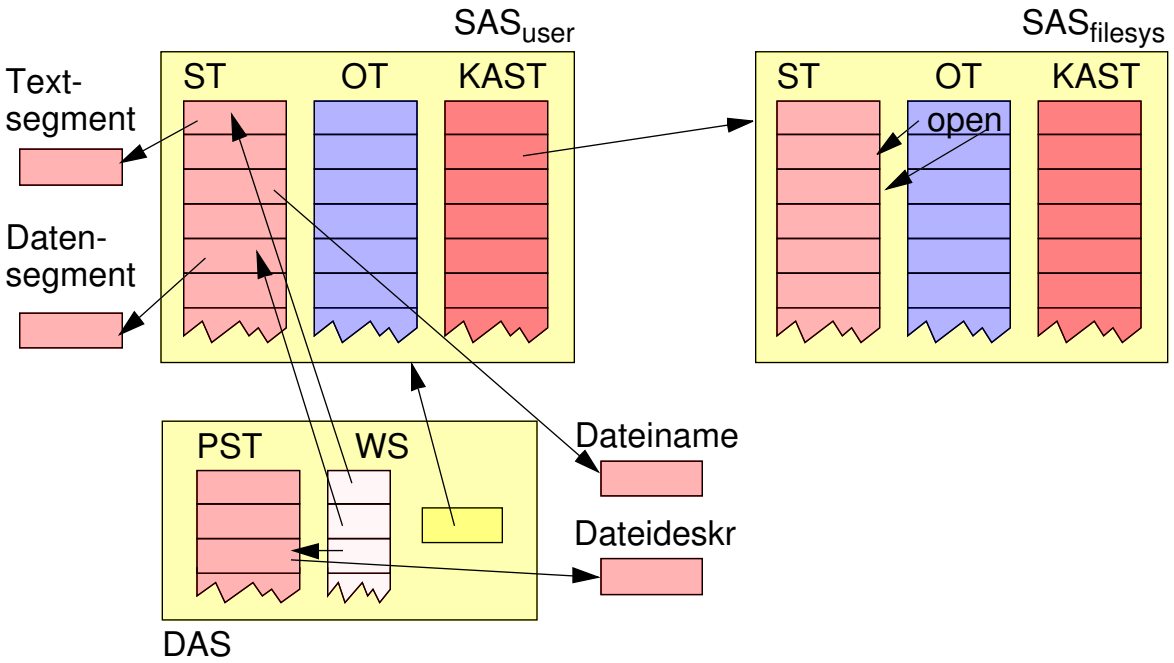



  - „open“ erzeugt neues Segment für den Dateideskriptor


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

# 2 Beispielaufruf (4)

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

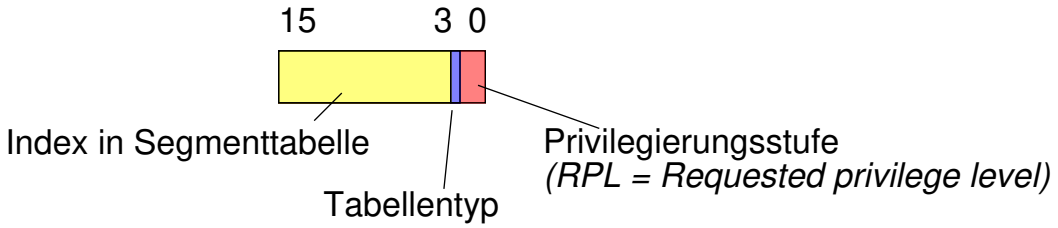



  - Ergebnissegment wird an den Aufrufer zurückgegeben


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

# 3 Beispiel: Pentium

- Privilegierungsstufen
  - Stufe 0: höchste Privilegien (privilegierte Befehle, etc.): BS Kern
  - Stufe 1: BS Treiber
  - Stufe 2: BS Erweiterungen
  - Stufe 3: Benutzerprogramme
- Segmentselektoren enthalten Privilegierungsstufe

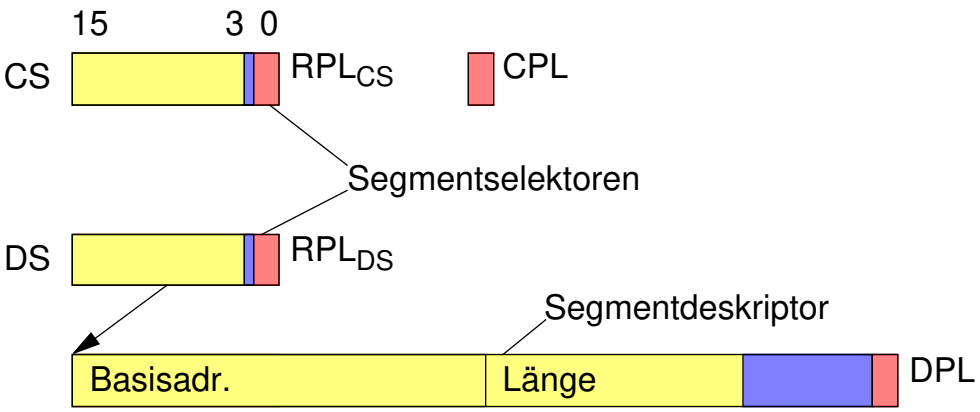



- Codesegmentselektor (CS) bestimmt aktuelle Privilegierungsstufe
  - CPL = *Current privilege level*


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

# 3 Beispiel: Pentium (2)

- Datenzugriff (z.B. auf Datensegment DS)

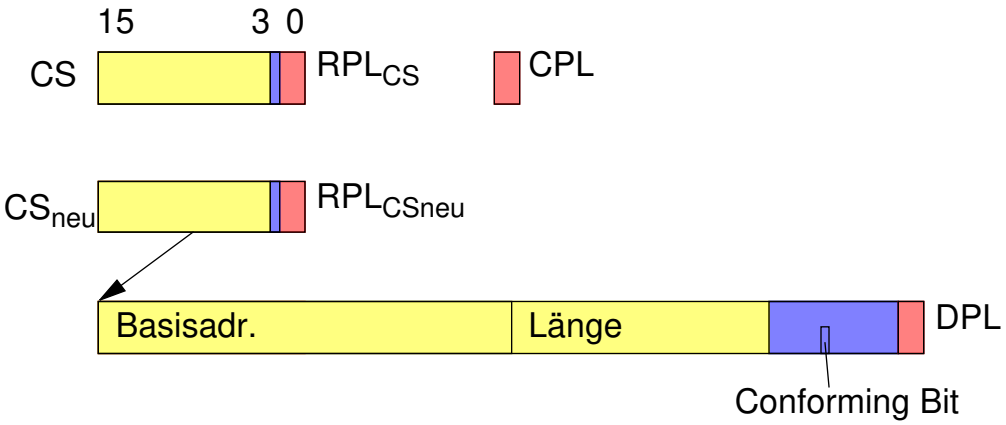



- DPL = *Descriptor privilege level*
- CPL ist normalerweise gleich RPL$_{CS}$
- Zugriff wird erlaubt, wenn: DPL ≥ max(CPL, RPL$_{DS}$)
- ansonsten wird Unterbrechung ausgelöst (Schutzverletzung)


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

# 3 Beispiel: Pentium (3)

- Sprünge in andere Codesegmente *(Far Call)*

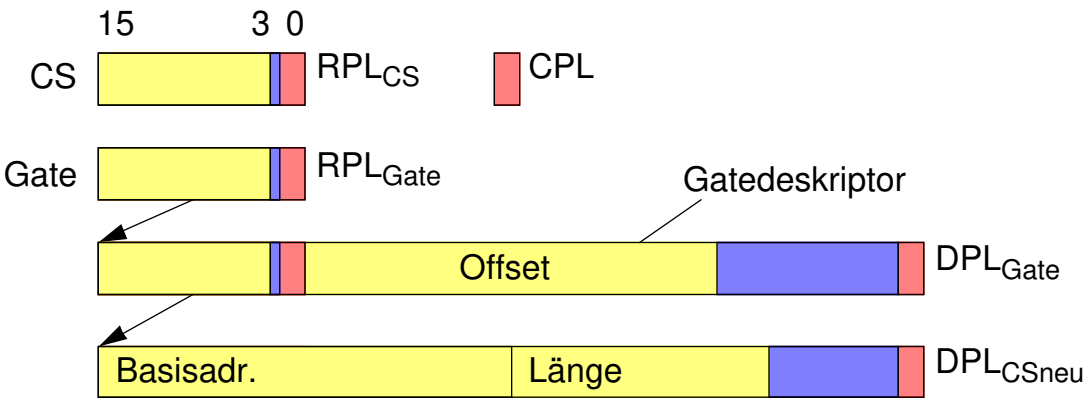



- Sprung wird erlaubt, falls: DPL = CPL oder Conforming Bit gesetzt und DPL ≤ CPL
- Im Falle von DPL ≤ CPL wird jedoch CPL nicht geändert (Codesegment hat höheres Privileg, CPL bleibt aber unverändert)


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

# 3 Beispiel: Pentium (4)

- Kontrolltransfer mit einem *Gate*

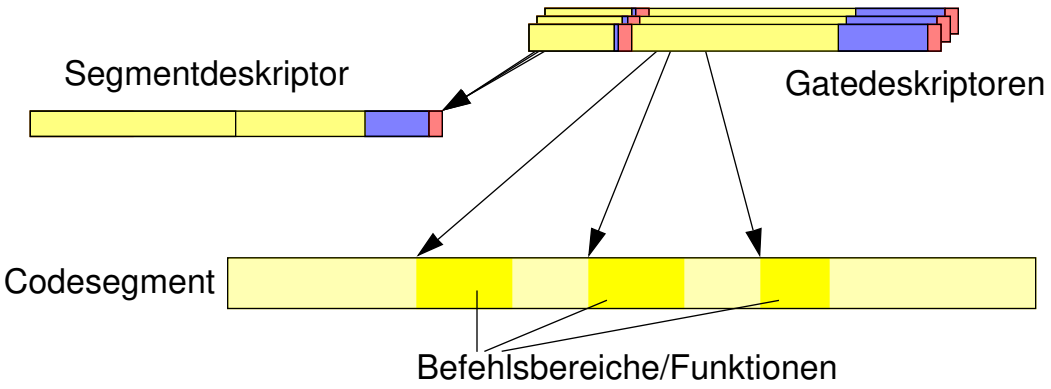



- Gatedeskriptoren stehen wie Segmentdeskriptoren in der Segmenttabelle
- Gatedeskriptor enthält Segmentselektor für das Codesegment und einen Offset zu diesem Segment, an dem der Einsprungpunkt liegt
- Kontrolltransfer (*CALL* Aufruf) wird erlaubt, falls:
  $DPL_{Gate} \geq \max(CPL, RPL_{Gate})$ und $CPL \geq DPL_{CSneu}$


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

# 3 Beispiel: Pentium (5)

- Gates erlauben den kontrollierten Sprung in ein privilegierten Befehlsbereich

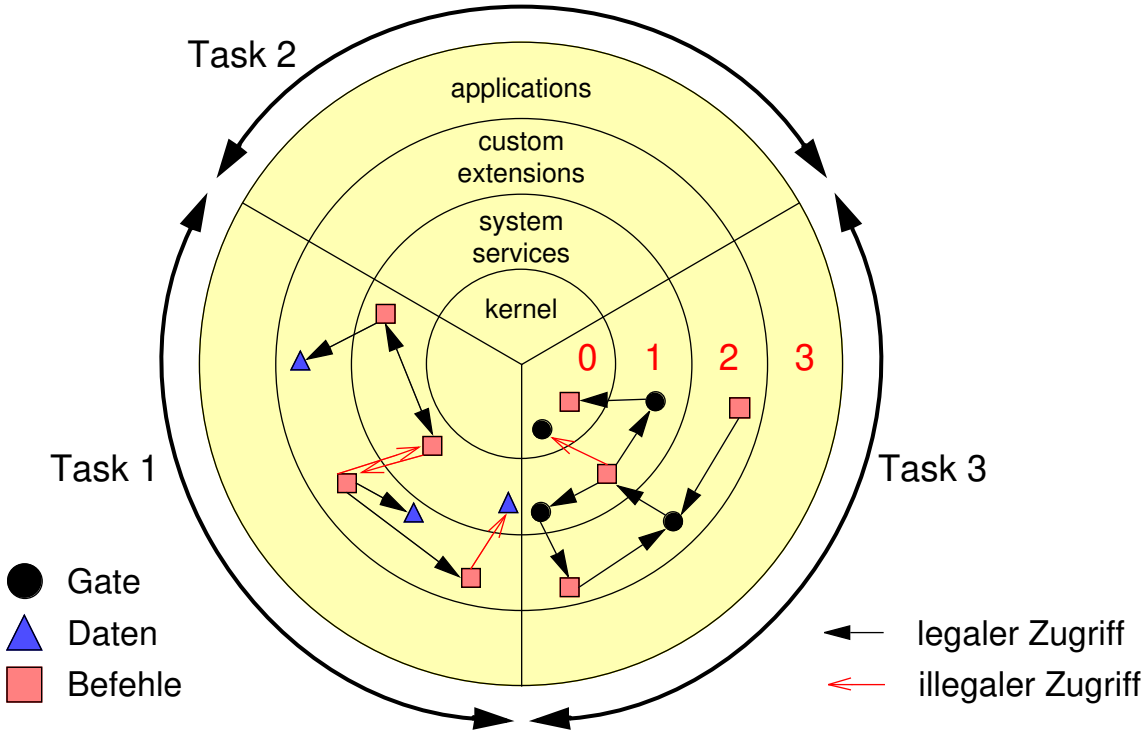



- es existiert auch der entsprechende Rücksprung
- für jede Privilegierungsstufe gibt es einen eigenen Stack; dieser wird mit umgeschaltet
- Parameter werden automatisch auf den neuen Stack kopiert (Anzahl wird im Gatedeskriptor vermerkt)


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

# 3 Beispiel: Pentium (6)

- Beispiel eines realisierten Schutzsystems

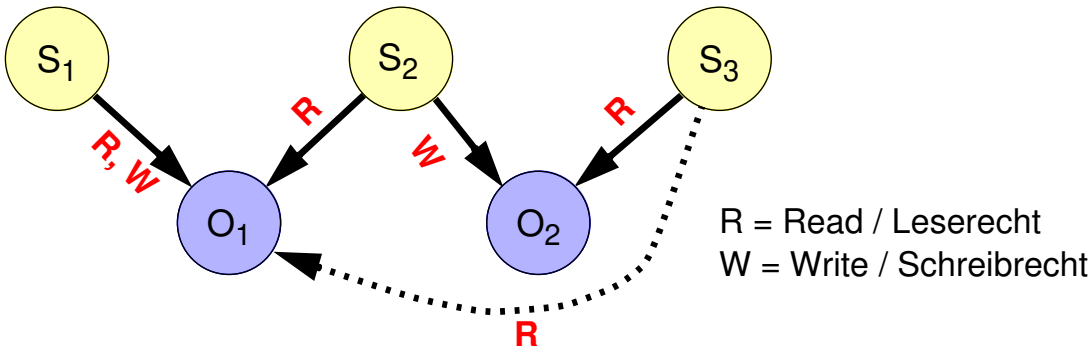




Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

# I.5 Capability-basierte Systeme

- Ein Benutzer (Subjekt) erhält eine Referenz auf ein Objekt
  - die Referenz enthält alle Rechte, die das Subjekt an dem Objekt besitzt
  - bei der Nutzung der Capability (Zugriff auf das Objekt) werden die Rechte überprüft




Subjekte und Objekte; Weitergabe einer Capability ($O_1$ von $S_2$ nach $S_3$)


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

# I.5 Capability-basierte Systeme (2)

★ Vorteile

- keine Speicherung von Rechten beim Objekt oder Subjekt nötig; Capability enthält Zugriffsrechte
- leichte Vergabe von individuellen Rechten
- einfache Weitergabe von Zugriffsrechten möglich

▲ Nachteile

- Weitergabe nicht kontrollierbar
- Rückruf von Zugriffsrechten nicht möglich
- Capability muß vor Fälschung und Verfälschung geschützt werden (z.B. durch kryptographische Mittel oder durch Speicherverwaltung)


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

# 1 Beispiel: Hydra

■ Hydra ist ein Capability-basiertes Betriebssystem

- entwickelt Mitte der Siebziger Jahre an der Carnegie-Mellon University
- lief auf einem speziellen Multiprozessor namens **C.mmp**
- Capability-Mechanismen sind integraler Bestandteil des Betriebssystems

■ Objekte in Hydra werden durch Capabilities angesprochen und geschützt

- Objekte haben einen Typ (z.B. Prozeduren, Prozesse, Semaphoren, Datei etc.)
- benutzerdefinierte Typen sind möglich
- generische Operationen für alle Typen implementiert durch das Betriebssystem
- Objekte besitzen eine Liste von Capabilities auf andere Objekte (genannt *C-List*)
- Objekte besitzen einen Datenbereich (implementiert durch geschütztes Segment)


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

# 1 Beispiel: Hydra (2)

- Prozesse (Subjekte)
  - Prozesse besitzen einen aktuellen Kontext, den LNS *(Local name space)*
  - LNS ist ein Objekt
  - LNS besitzt einen Aktivitätsträger (Thread)
  - LNS kann nur auf Objekte zugreifen, die in seiner C-List stehen (mehrstufige Zugriffe, z.B. auf die C-List eines Objekts, dessen Capabilities in der C-List des LNS steht, sind möglich)
- Capabilities
  - Capabilities haben einen Typ (z.B. Prozedur, LNS etc.)
  - Prozesse können nur über Systemaufrufe ihre Capabilities bzw. ihre C-List bearbeiten
  - Capabilities können nicht gefälscht oder verfälscht werden
  - Betriebssystem kann sicheres Schutzkonzept basierend auf Capabilities implementieren


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

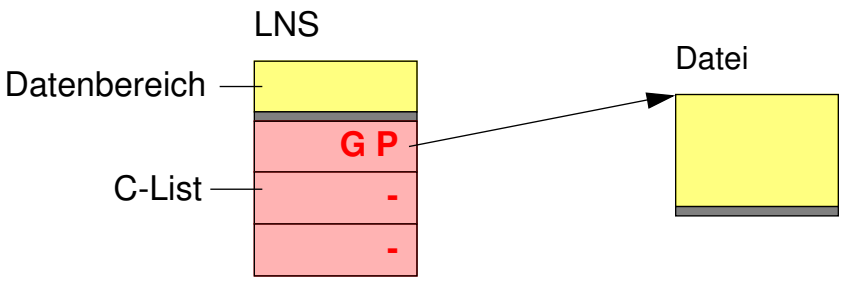
# 1 Beispiel: Hydra (3)

- Datenbereich
  - Operationen:
    - *Getdata:* kopiere Abschnitt aus dem Datenbereich eines Objekts in den Datenbereich des LNS
    - *Putdata:* kopiere Abschnitt aus dem Datenbereich des LNS in den Datenbereich eines Objekts
    - *Adddata:* füge Daten zu dem Datenbereich eines Objekts hinzu
  - Rechte:
    - ***G****ETRTS:* erlaubt den Aufruf von *Getdata*
    - ***P****UTRTS:* erlaubt den Aufruf von *Putdata*
    - ***AD****DRTS:* erlaubt den Aufruf von *Adddata*
  - Rechte müssen in der Capability zum Objekt gesetzt sein


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

# 1 Beispiel: Hydra (4)

- Implementierung von Dateien:
  - ◆ *Getdata* erlaubt das Lesen von Daten
  - ◆ *Putdata* erlaubt das Schreiben von Daten
  - ◆ *Adddata* erlaubt das Anhängen von Daten
- Entsprechende Rechte können pro Capability gesetzt werden





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

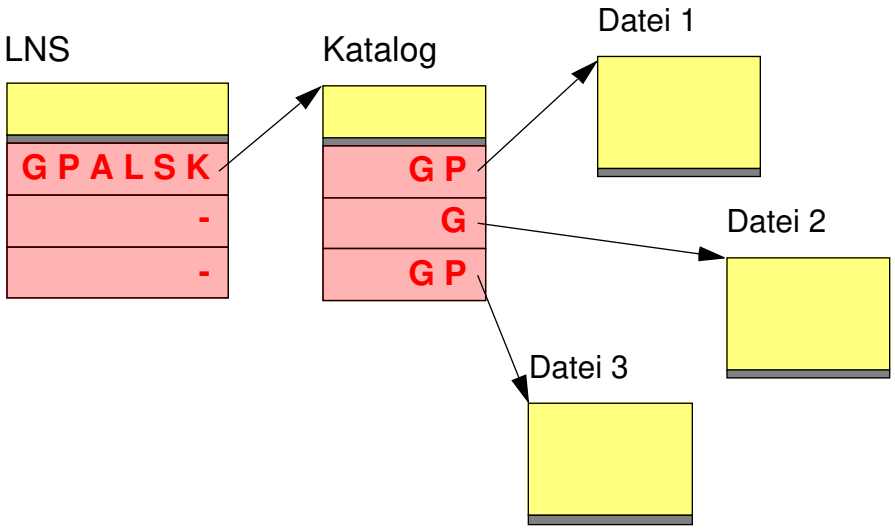
# 1 Beispiel: Hydra (5)

- C-List
  - ◆ Operationen:
    - *Load:* kopieren einer Capability aus der C-List eines Objekts in die C-List des LNS
    - *Store:* kopieren einer Capability aus der C-List des LNS in die C-List eines Objekts (dabei können Rechte maskiert werden)
    - *Append:* anfügen einer Capability in die C-List eines Objekts
    - *Delete:* löschen einer Capability aus der C-List eines Objekts
  - ◆ Rechte:
    - ***L****OADRTS:* erlaubt Aufruf von *Load*
    - ***S****TORTS:* erlaubt Aufruf von *Store*
    - ***A****PPRTS:* erlaubt Aufruf von *Append*
    - ***K****ILLRTS:* erlaubt Aufruf von *Delete*


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

# 1 Beispiel: Hydra (6)

- Implementierung von Katalogen:
  - *Load* erlaubt das Auflösen von Namen (Aufrufer bekommt die Capability)
  - *Store* und *Append* erlauben das Hinzufügen von Dateien zum Katalog
  - *Delete* erlaubt das Austragen von Dateien aus dem Katalog

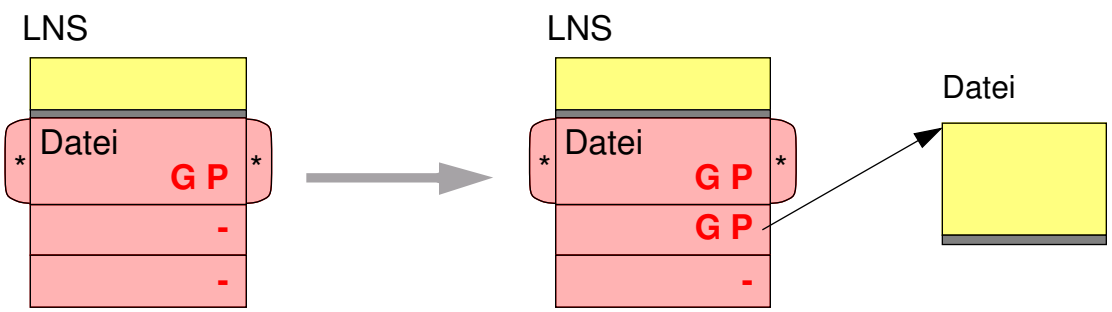




Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

# 1 Beispiel: Hydra (7)

- Objekterzeugung über Erzeugungsschablonen *(Creation Templates)*
  - Erzeugungsschablone enthält den Typ des neu zu erzeugenden Objektes und eine Rechtemaske
  - nur die in der Maske angeschalteten Rechte werden dem Aufrufer in einer neuen Capability gegeben

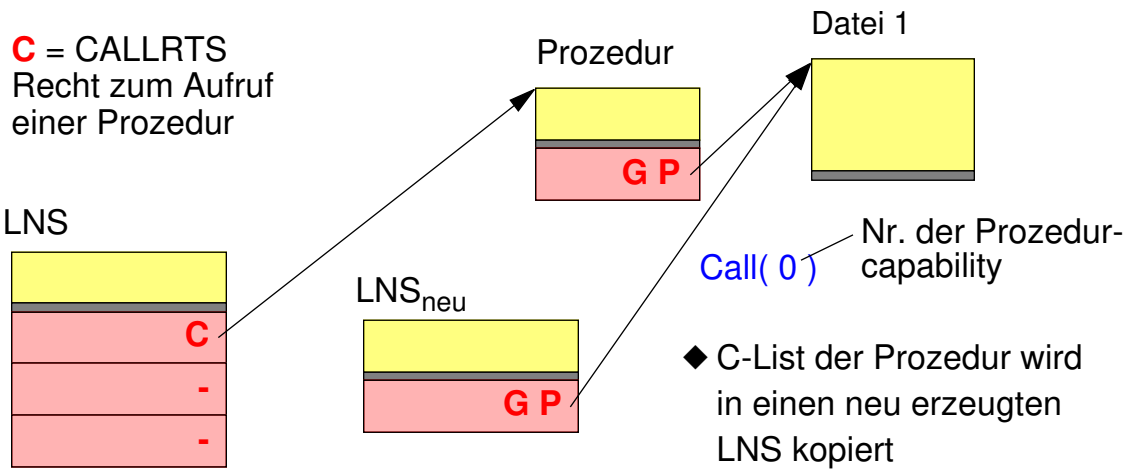




Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

# 2 Hydra Prozeduraufruf

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

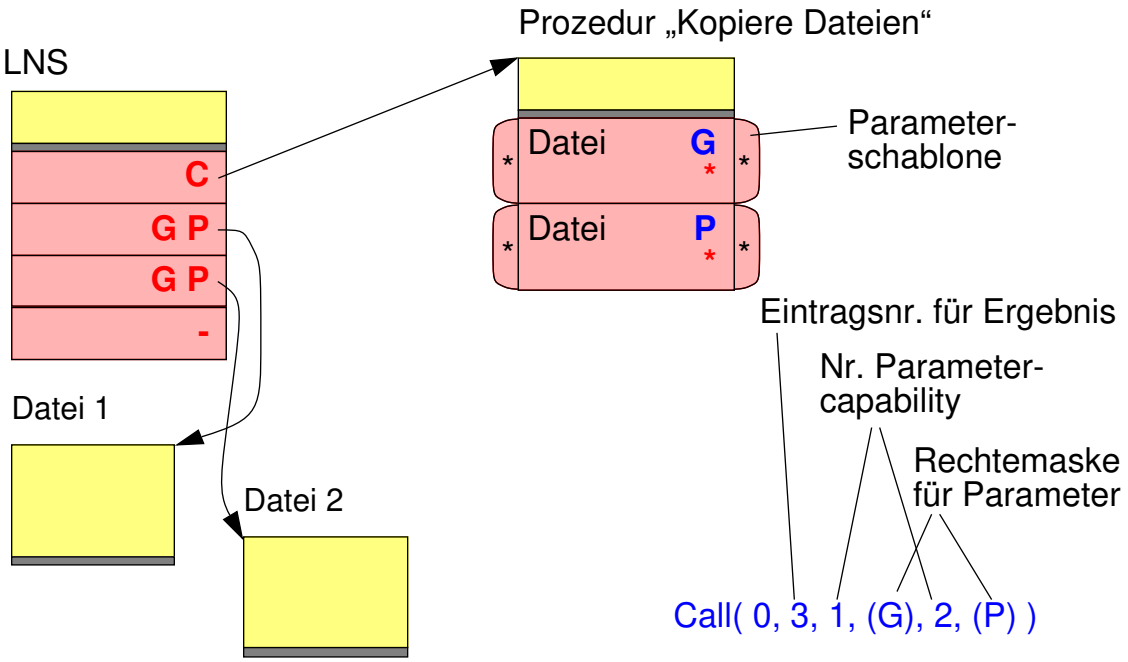




Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

# 2 Hydra Prozeduraufruf (2)

- 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.