I/O – Flexpages

Frank Réolon

24.September 2004

Aufbau des Vortrags

- Basisinformationen
- Speicherverwaltung mit Flexpages
- Übergang zu I/O – Flexpages
- Realisierung der I/O - Flexpages

1.Basisinformationen

- L4 (die 2.Generation)

Aufgaben:

-Schnelle Prozess Kommunikation

-Flexible Konzepte um disponiblen Speicher zu managen

-Kontrolle und Schutz der unterliegenden Hardware

-Lösen des Geschwindigkeitsproblem der 1.Generation

1.Basisinformationen

I/O – Ports

Steuerleitungen vom Prozessor zur Hardware.

I/O – Space

In x86-Architektur separater Adressbereich neben Hauptspeicher.
I/O – Space fasst alle Register, welche auf der Hardware liegen zusammen.

1.Basisinformationen

- σ_0

Anfänglicher, gesamter verfügbarer Speicher bei Systemstart.

- Pager

Auf Benutzerebene laufende Prozesse, welche sich um die Speicherverwaltung kümmern.

2.Speicherverwaltung mit Flexpages

Flexpage

- beschreibt ein Speicherobjekt in einem Adressraum
- Zusammensetzung:
  - hat eine gegebene Basisadresse
  - eine bestimmte Größe
- Besteht aus allen gültigen virtuellen Seiten dieser Region
- Aufgabe:
  - Verteilung der Adressbereiche

2.Speicherverwaltung mit Flexpages

- 3 Operationen auf Flexpages

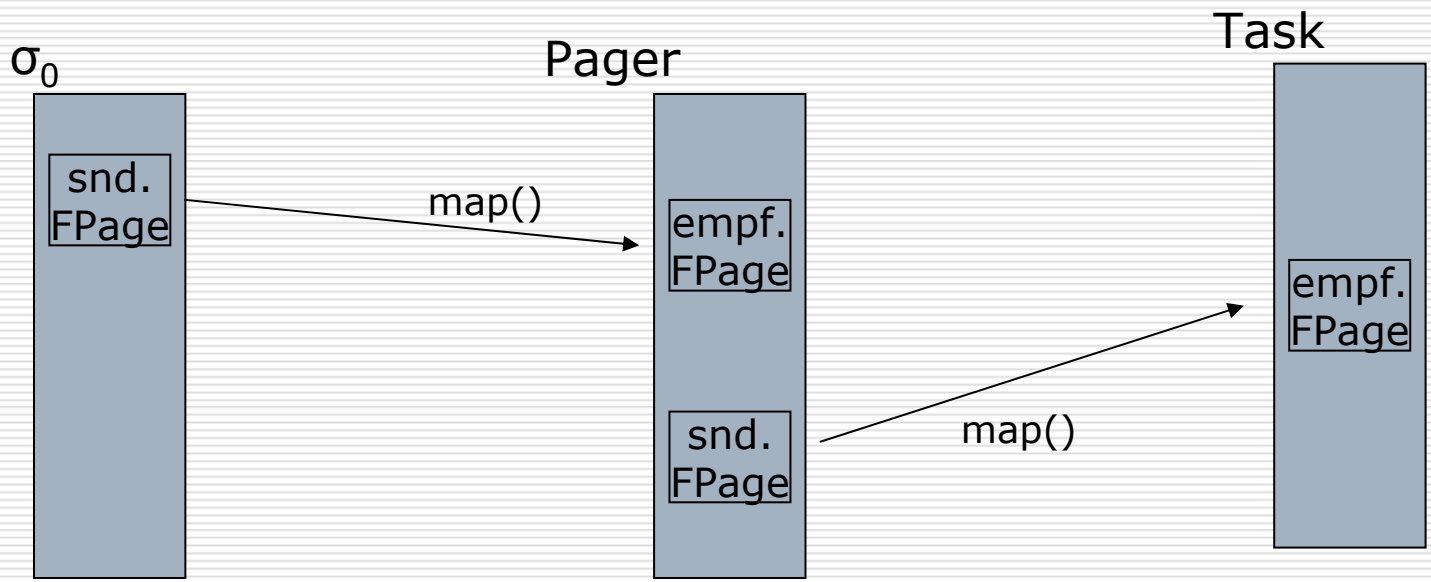
a) map: - bildet Speicherregion der Senderadresse in Empfängeradresse ab

b) grant: - selbe Vorgang wie bei map, aber zusätzlich verliert Sender Zugriffsrechte auf Speicherregion

c) unmap: - hebt alle Verbindungen einer Speicherregion zu der dazugehörigen Flexpage auf

2.Speicherverwaltung mit Flexpages

- Beispiel



3.Übergang zu I/O - Flexpages

I/O Instruction

- I/O – Ports sind jeweils individuell ansprechbar

- Prozessor kann sich Daten von I/O-Ports holen oder hin senden, dazu gibt es zwei Befehle:
  - in(port)
  - out(port)

3.Übergang zu I/O – Flexpages

I/O Schutzmechanismen

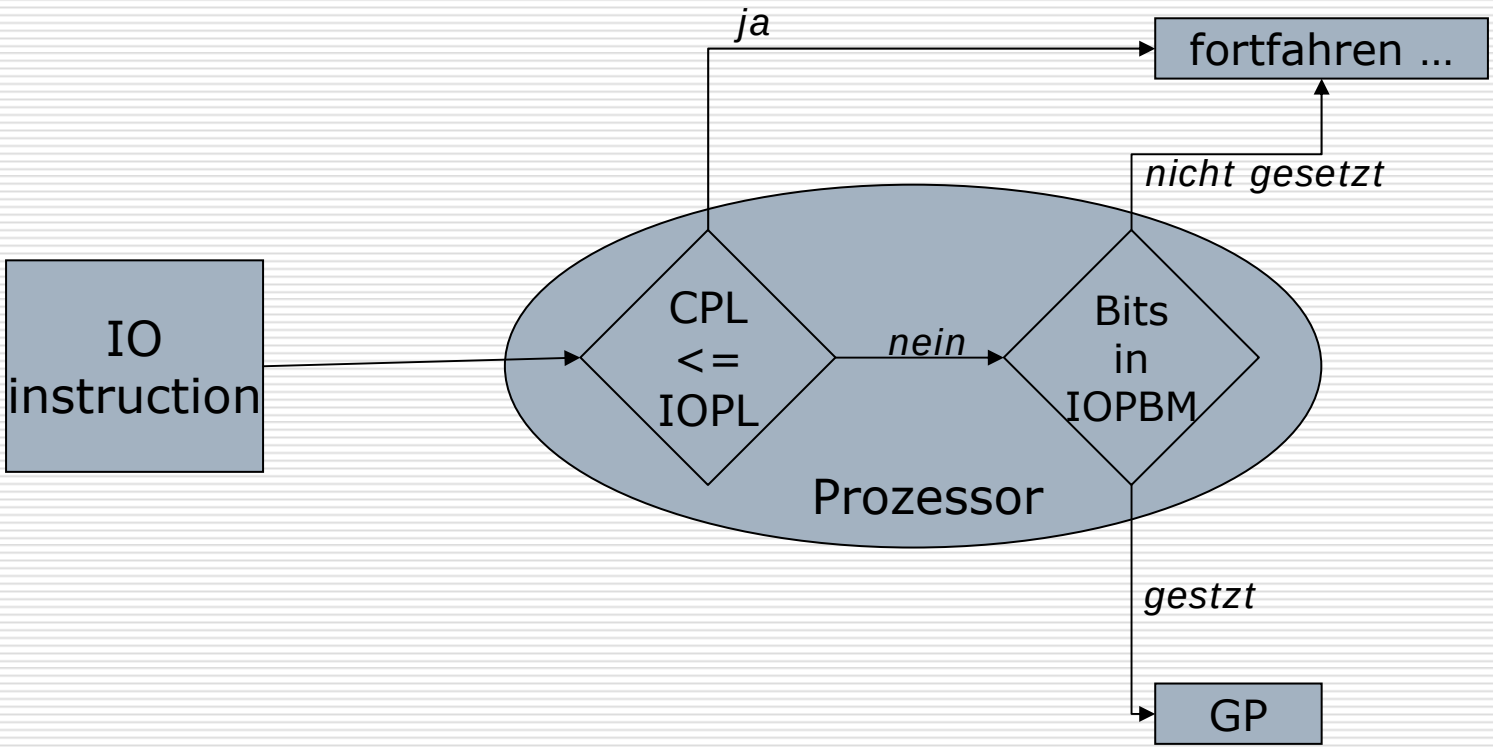
Zugriffsverletzung endet in eine GP (general protection exception).
2 Mechanismen werden genutzt:

a) I/O Privilege Levels (von Hardware bereit gestellt):
gewährt Zugriff auf alle Ports

CPL : current privilege level
IOPL: IO privilege level

b) I/O Permission Bitmap (IOPBM):
gewährt oder verbietet Zugriff auf einzelne Ports

3.Übergang zu I/O – Flexpages



3.Übergang zu I/O - Flexpages

- Einführung der I/O – Flexpages
  - Ziel: Analogie zu Speicherverwaltung herstellen
  - Definition: I/O – Flexpage ist ein Objekt mit variabler Größe, welches Bezug auf eine Auswahl von I/O – Ports nimmt.
  - I/O – Flexpage besitzt eine Basisadresse p
  - Größe einer I/O – Flexpage ist immer durch eine 2er – Potenz beschrieben.

3.Übergang zu I/O – Flexpages

- Operationen auf I/O – Flexpages

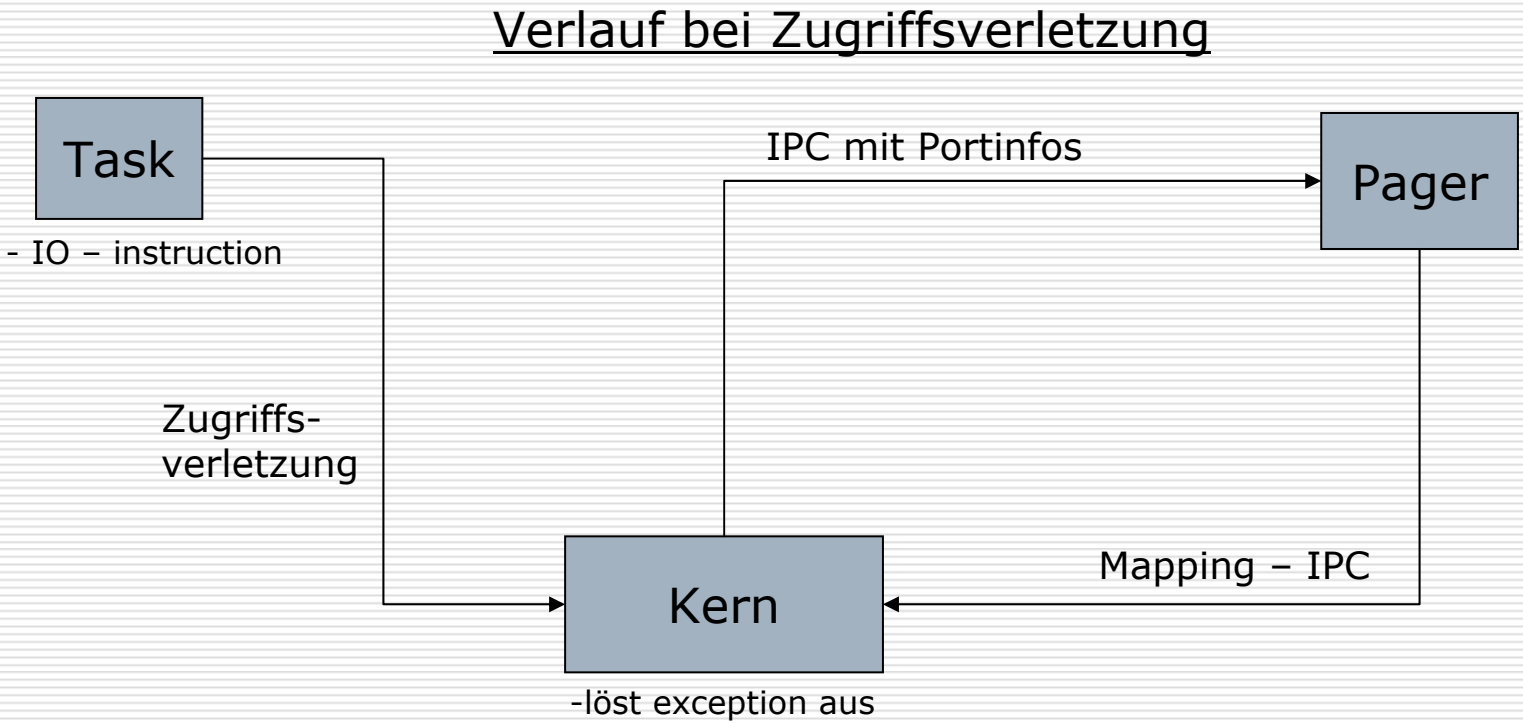
  - map: bildet Regionen des I/O – Space, welcher in I/O – Flexpage beschrieben ist, in den Empfänger ab
  - grant: selber Vorgang wie bei map, aber Sender verliert Zugriffsrechte auf I/O – Ports
  - unmap: hebt alle Verbindungen der I/O – Ports zu der I/O – Flexpage auf

3.Übergang zu I/O – Flexpages

- σ_0
  - Verteilung der I/O – Ports genauso wie virtueller Speicher:
    I/O – Space wird verteilt und kontrolliert von den initialen I/O – Pagern

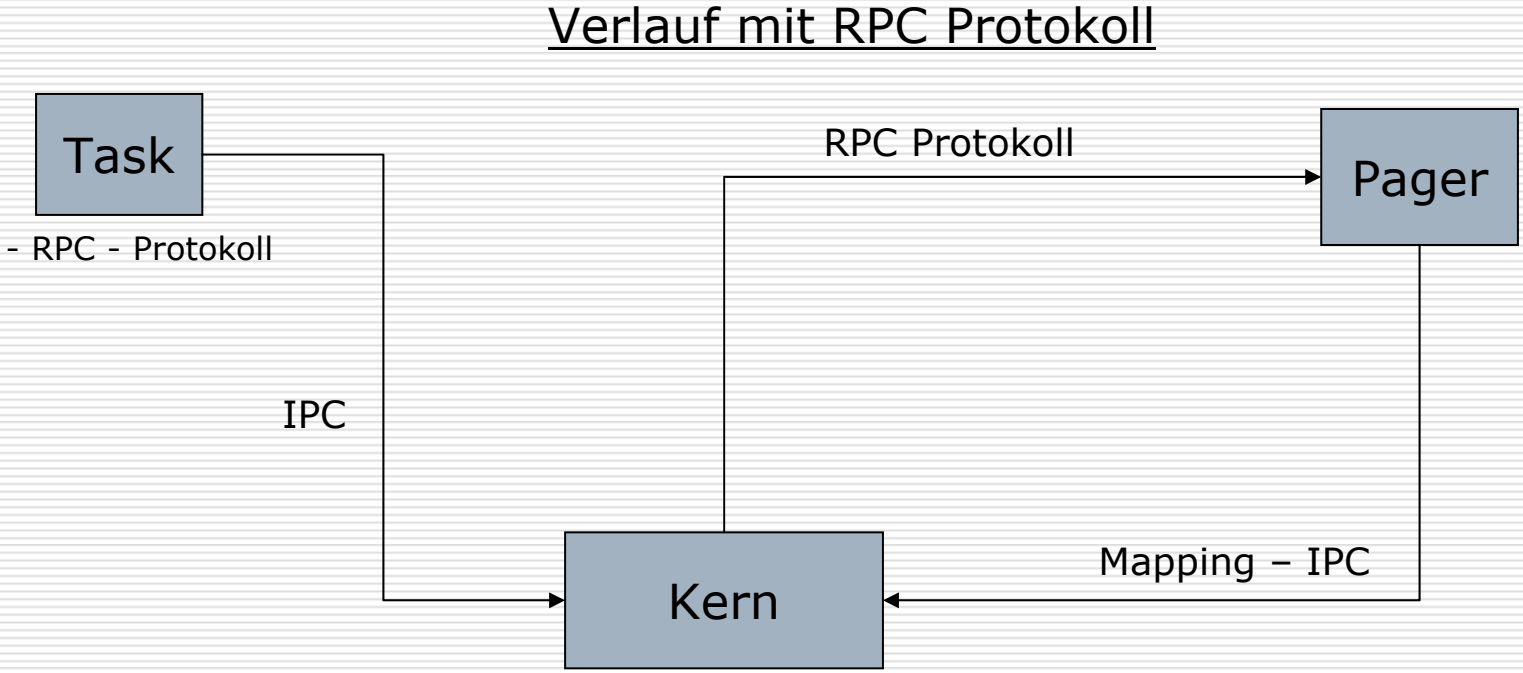
3.Übergang zu I/O – Flexpages

- Das RPC Protokoll für GP



3.Übergang zu I/O – Flexpages

- Das RPC Protokoll für GP



3.Übergang zu I/O – Flexpages

- Legacy Support
  - 2 Einstellungen möglich:

a) implicit I/O – Mapping
- grundsätzlich: Anwendung Zugriffsrechte auf alle IO – Ports
- um Zugriffe zu sperren: Adressbereich explizit unmappen

b) No implicit IO – Mapping
- jede Anwendung keinerlei Zugriffsrechte
- Zugriffe gewährt nur durch mapping bzw. granting

4.Realisierung der I/O – Flexpages

- I/O – Privilege Levels
  - Alle Anwendungen auf Benutzerebene haben IOPL von 0
  - Sonderfall σ_0 hat IOPL von 3
- IOPBM und das Mapping Konzept
  - Jede Anwendung hat IOPBM
  - Funktionalität von map, grant und unmap durch Ändern der verantwortlichen Bits in der IOPBM der sendenden und empfangenden Task realisiert
  - bei GP ermittelt µ-kernel den Grund dafür:
    - bei Lücke in der IO – permission
      -> senden einer IPC an Pager

4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - Motivation: Bereitstellen von hierarchischen gegliederten I/O – Adressbereichen
  - Baumartige Struktur:
    - jeder Knoten beschreibt I/O – Bereich
    - jede Kante beschreibt Mapping

4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - Datenstruktur: io – mapping – nodes

    io – mapping – node hat folgende Parameter:
    - taskID
    - low
    - high
    - depth
    - prev, next
    - prev_tasklist, next_tasklist
    - parent

4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - map:

MAP_IO_FPAGE(from, to, fpage)

1. in Task list von from wird nach relevanten Knoten gesucht
2. neuer Knoten wird kreiert, Bereich wird durch 1. bestimmt
3. frühere Mappings in to werden im relevanten Bereich aufgehoben
4. einfügen des Knotens in den I/O – Mapping – tree
5. letztendlich Freigegeben der Bits in der to – IOPBM

4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - grant:

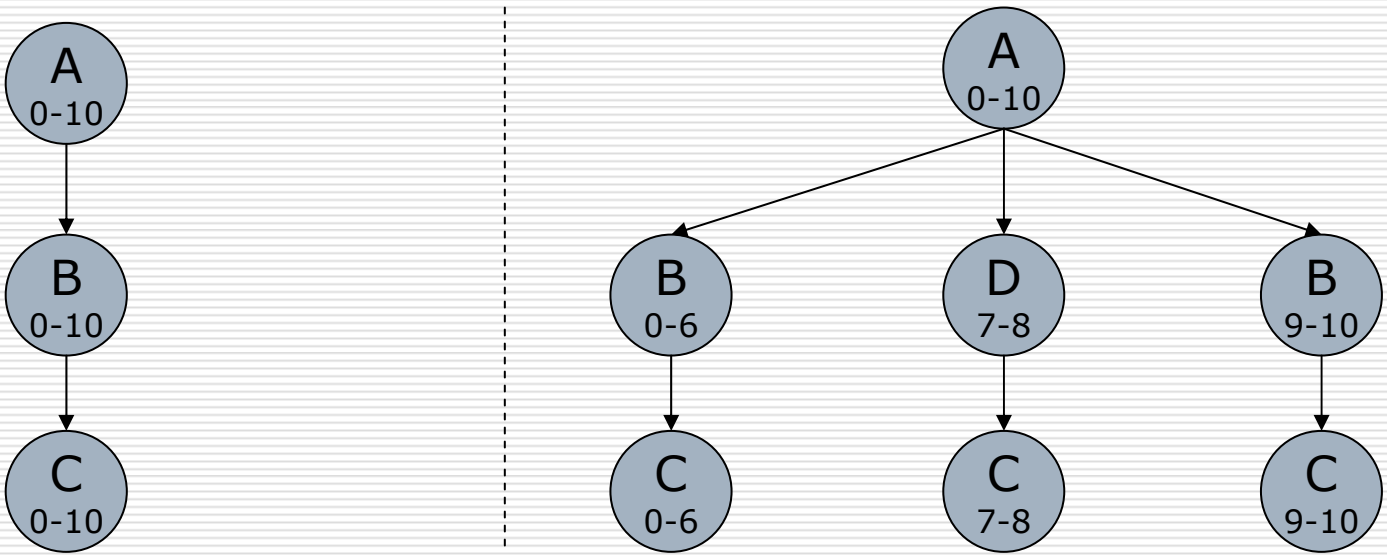
GRANT_IO_FPAGE(from, to, fpage)

1. in Task list von from wird nach relevanten Knoten gesucht
2. frühere Mappings in to werden im relevanten Bereich aufgehoben
3. Aufrufen von grant_subtree (from, to, fpage)
4. Mappings in from werden im relevanten Bereich aufgehoben
5. Updaten der from – und to – IOPBM

4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - Beispiel:

grant_subtree(B, D, 7-8)



4.Realisierung der I/O – Flexpages

- I/O – Mapping – Datenbank
  - unmap:

UNMAP_IO_FPAGE(task, fpage, mapmask)

1. in Task list von from wird nach relevanten Knoten gesucht
2. Gesamter Subtree wird unmappt, ist mapmask auf true wird auch der Vaterknoten ungemappt

I/O – Flexpages

- Ende