Memory Management Server

Kai Beskers, Sven Janko

18. Juni 2009

SDI 1

Übersicht

Einführung

Aufgaben

- Addressräume strukturieren und verwalten
- Pagefaultbehandlung
- ELF-Images laden
- Shared Memory ermöglichen

Zielsetzung

- Wir besprechen nur Mechanismen, Strategien sind Sache einer konkreten Implementierung.

Einführung

Aufgaben

- Addressräume strukturieren und verwalten
- Pagefaultbehandlung
- ELF-Images laden
- Shared Memory ermöglichen

Zielsetzung

- Wir besprechen nur Mechanismen, Strategien sind Sache einer konkreten Implementierung.

Einführung

Aufgaben

- Addressräume strukturieren und verwalten
- Pagefaultbehandlung
- ELF-Images laden
- Shared Memory ermöglichen

Zielsetzung

- Wir besprechen nur Mechanismen, Strategien sind Sache einer konkreten Implementierung.

Einführung

Aufgaben

- Addressräume strukturieren und verwalten
- Pagefaultbehandlung
- ELF-Images laden
- Shared Memory ermöglichen

Zielsetzung

- Wir besprechen nur Mechanismen, Strategien sind Sache einer konkreten Implementierung.

Einführung

Aufgaben

- Addressräume strukturieren und verwalten
- Pagefaultbehandlung
- ELF-Images laden
- Shared Memory ermöglichen

Zielsetzung

- Wir besprechen nur Mechanismen, Strategien sind Sache einer konkreten Implementierung.

Der Startvorgang

- Roottask startet MM-Server
- MM-Server baut Datenstrukturen auf
- Roottask startet Taskservice
- Taskservice startet Hintergrundspeicherverwaltung
- Taskservice registriert Hintergrundspeicherverwaltung bei MM-Server

Der Startvorgang

- Roottask startet MM-Server
- MM-Server baut Datenstrukturen auf
- Roottask startet Taskservice
- Taskservice startet Hintergrundspeicherverwaltung
- Taskservice registriert Hintergrundspeicherverwaltung bei MM-Server

Der Startvorgang

- Roottask startet MM-Server
- MM-Server baut Datenstrukturen auf
- Roottask startet Taskservice
- Taskservice startet Hintergrundspeicherverwaltung
- Taskservice registriert Hintergrundspeicherverwaltung bei MM-Server

Der Startvorgang

- Roottask startet MM-Server
- MM-Server baut Datenstrukturen auf
- Roottask startet Taskservice
- Taskservice startet Hintergrundspeicherverwaltung
- Taskservice registriert Hintergrundspeicherverwaltung bei MM-Server

Der Startvorgang

- Roottask startet MM-Server
- MM-Server baut Datenstrukturen auf
- Roottask startet Taskservice
- Taskservice startet Hintergrundspeicherverwaltung
- Taskservice registriert Hintergrundspeicherverwaltung bei MM-Server

Das ELF-Loader Protokoll

- ELF-Image anfordern
- ELF-Image interpretieren und etablieren
- Stack allozieren
- TCBarea allozieren

Das ELF-Loader Protokoll

- ELF-Image anfordern
- ELF-Image interpretieren und etablieren
- Stack allozieren
- TCBarea allozieren

Das ELF-Loader Protokoll

- ELF-Image anfordern
- ELF-Image interpretieren und etablieren
- Stack allozieren
- TCBarea allozieren

Das ELF-Loader Protokoll

- ELF-Image anfordern
- ELF-Image interpretieren und etablieren
- Stack allozieren
- TCBarea allozieren

Seitentabelle

- Threadid $\longrightarrow$ Seitentabelle
- Referenzzähler in Seitentabelle
- Swap- und Seitentabelle pro Adressraum (inklusive Klassifizierung der Seiten)
- ...

Seitentabelle

- Threadid $\longrightarrow$ Seitentabelle
- Referenzzähler in Seitentabelle
- Swap- und Seitentabelle pro Adressraum (inklusive Klassifizierung der Seiten)
- ...

Seitentabelle

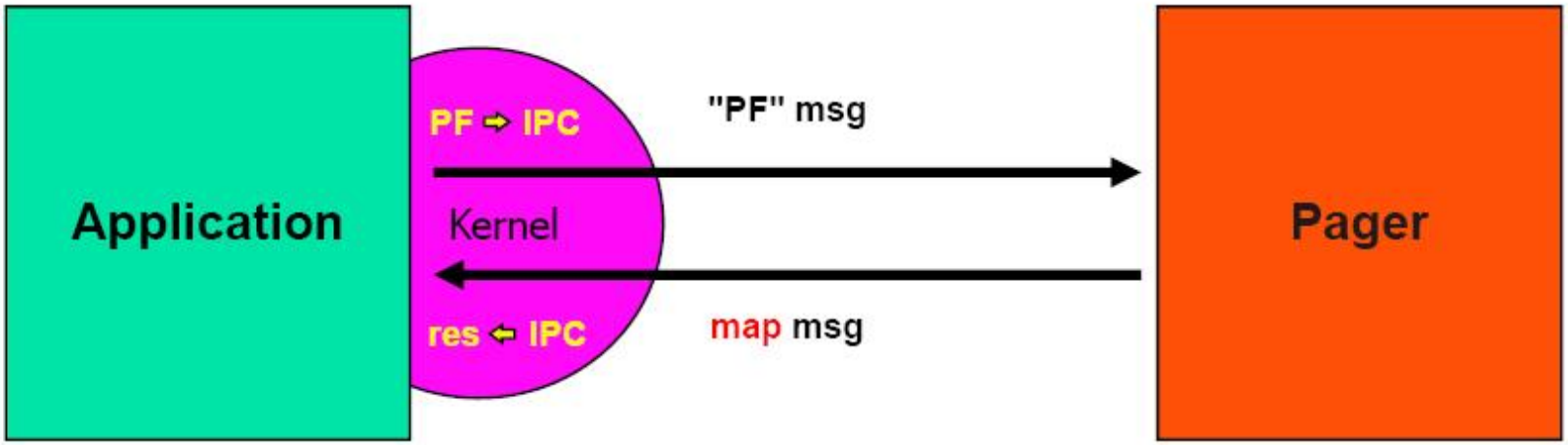
- Threadid $\longrightarrow$ Seitentabelle
- Referenzzähler in Seitentabelle
- Swap- und Seitentabelle pro Adressraum (inklusive Klassifizierung der Seiten)
- ...

Seitentabelle

- Threadid $\longrightarrow$ Seitentabelle
- Referenzzähler in Seitentabelle
- Swap- und Seitentabelle pro Adressraum (inklusive Klassifizierung der Seiten)
- ...

Freispeicherverwaltung: Boundary Tag

Wiederholung: Pagefault Protokoll



Speicherallozierung

- **malloc** (in L4_Word_t size, out L4_Word_t location)
- **malloc_priv** (in L4_Word_t size, in L4_Word_t address, out L4_Word_t location)
  - address: physische Speicheradresse bzgl. I/O
  - locking?
- **free** (in L4_Word_t size, in L4_Word_t location)

Speicherallozierung

- **malloc** (in L4_Word_t size, out L4_Word_t location)
- **malloc_priv** (in L4_Word_t size, in L4_Word_t address, out L4_Word_t location)
  - address: physische Speicheradresse bzgl. I/0
  - locking?
- **free** (in L4_Word_t size, in L4_Word_t location)

Speicherallozierung

- **malloc** (in L4_Word_t size, out L4_Word_t location)
- **malloc_priv** (in L4_Word_t size, in L4_Word_t address, out L4_Word_t location)
  - address: physische Speicheradresse bzgl. I/0
  - locking?
- **free** (in L4_Word_t size, in L4_Word_t location)

Speicherallozierung

- **malloc** (in L4_Word_t size, out L4_Word_t location)
- **malloc_priv** (in L4_Word_t size, in L4_Word_t address, out L4_Word_t location)
  - address: physische Speicheradresse bzgl. I/0
  - locking?
- **free** (in L4_Word_t size, in L4_Word_t location)

Speicherallozierung

- **malloc** (in L4_Word_t size, out L4_Word_t location)
- **malloc_priv** (in L4_Word_t size, in L4_Word_t address, out L4_Word_t location)
  - address: physische Speicheradresse bzgl. I/0
  - locking?
- **free** (in L4_Word_t size, in L4_Word_t location)

Seitenpinning

- **pin_priv** (in L4_Word_t size, in L4_Word_t address)
- **unpin_priv** (in L4_Word_t size, in L4_Word_t address)

Seitenpinning

- **pin_priv** (in L4_Word_t size, in L4_Word_t address)
- **unpin_priv** (in L4_Word_t size, in L4_Word_t address)

Shared Memory

- **map** (in L4_fpage_t page, in L4_Word_t offset, in L4_Word_t priv, in L4_ThreadId_t threadid)
- **unmap** (in L4_fpage_t page, in L4_Word_t offset, in L4_ThreadId_t threadid)
- Semantik wie bei der map-IPC

Shared Memory

- **map** (in L4_fpage_t page, in L4_Word_t offset, in L4_Word_t priv, in L4_ThreadId_t threadid)
- **unmap** (in L4_fpage_t page, in L4_Word_t offset, in L4_ThreadId_t threadid)
- Semantik wie bei der map-IPC

Shared Memory

- **map** (in L4_fpage_t page, in L4_Word_t offset, in L4_Word_t priv, in L4_ThreadId_t threadid)
- **unmap** (in L4_fpage_t page, in L4_Word_t offset, in L4_ThreadId_t threadid)
- Semantik wie bei der map-IPC

Funktionen: Server

- **register_backingstore** (in L4_ThreadId_t backingstore)
- **new_as** (in L4_ThreadId_t threadid, in L4_Word_t priv, in L4_Word_t file_handle, in L4_ThreadId_t fileserver, out L4_Word_t sp, out L4_Word_t ip, out L4_fpage_t TCBarea)
- **register_thread** (in L4_ThreadId_t threadid, in L4_ThreadId_t asid, in L4_Word_t priv, out L4_Word_t sp)
- **deregister_thread** (in L4_ThreadId_t threadid)

Funktionen: Server

- **register_backingstore** (in L4_ThreadId_t backingstore)
- **new_as** (in L4_ThreadId_t threadid, in L4_Word_t priv, in L4_Word_t file_handle, in L4_ThreadId_t fileserver, out L4_Word_t sp, out L4_Word_t ip, out L4_fpage_t TCBarea)
- **register_thread** (in L4_ThreadId_t threadid, in L4_ThreadId_t asid, in L4_Word_t priv, out L4_Word_t sp)
- **deregister_thread** (in L4_ThreadId_t threadid)

Funktionen: Server

- **register_backingstore** (in L4_ThreadId_t backingstore)
- **new_as** (in L4_ThreadId_t threadid, in L4_Word_t priv, in L4_Word_t file_handle, in L4_ThreadId_t fileserver, out L4_Word_t sp, out L4_Word_t ip, out L4_fpage_t TCBarea)
- **register_thread** (in L4_ThreadId_t threadid, in L4_ThreadId_t asid, in L4_Word_t priv, out L4_Word_t sp)
- **deregister_thread** (in L4_ThreadId_t threadid)

Funktionen: Server

- **register_backingstore** (in L4_ThreadId_t backingstore)
- **new_as** (in L4_ThreadId_t threadid, in L4_Word_t priv, in L4_Word_t file_handle, in L4_ThreadId_t fileserver, out L4_Word_t sp, out L4_Word_t ip, out L4_fpage_t TCBarea)
- **register_thread** (in L4_ThreadId_t threadid, in L4_ThreadId_t asid, in L4_Word_t priv, out L4_Word_t sp)
- **deregister_thread** (in L4_ThreadId_t threadid)

Ende

Gibt es Fragen?