Konzepte von Betriebssystemkomponenten

Referat am 24.11.2003

Thema:

Adressräume, Page Faults, Demand Paging, Copy on Write

Referent: Johannes Werner

Gliederung

- Adressräume
- Page Faults
- Demand Paging
- Copy on write
- Umsetzung in Linux
- Fragen

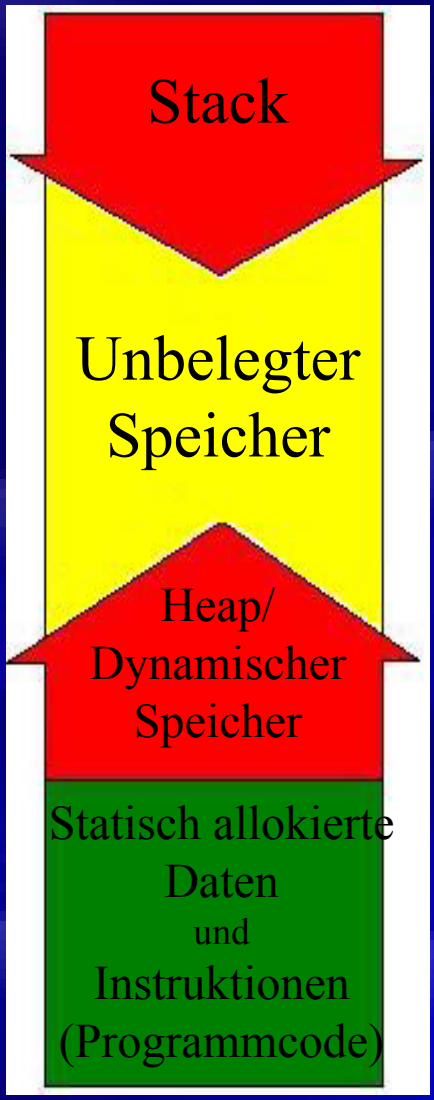
Was ist ein Adressraum?

- besteht aus allen logischen Adressen eines Prozesses
- jeder Prozess besitzt einen eigenen Adressraum
  (außer: geteilter (shared) Speicher)
- Idealvorstellung:
  - Maximale Größe (ab i386: 4GB)
  - Vollständig adressierbar
    (ungeachtet der tatsächlichen Speicher-Auslastung)
  - exklusive Benutzung durch einen Kontrollfluß
  - isoliert von anderen Adressräumen

Adressräume:

Typische Nutzung eines Adressraums: (Idealvorstellung)

- kleiner Programmbereich
- Abstand zwischen Heap und Stack sehr groß
- Keine nicht benutzbaren Teile, wie
  - E/A Bereiche
  - nicht vorhandene Adressbereiche ->(Hardware)
  - Fremdprozesse



Realer (Hardware) Adressraum:

Adressräume:

Das Betriebssystem muss einem Prozess auf verschiedenen Systemen dieselben Ressourcen zur Verfügung stellen.

Wie ist dieses Problem lösbar?

- Prozesse benutzen nur Anfang und Ende des Adressraumes (Daten, Heap, Stack)
- Logische (Virtuelle) Adressierung

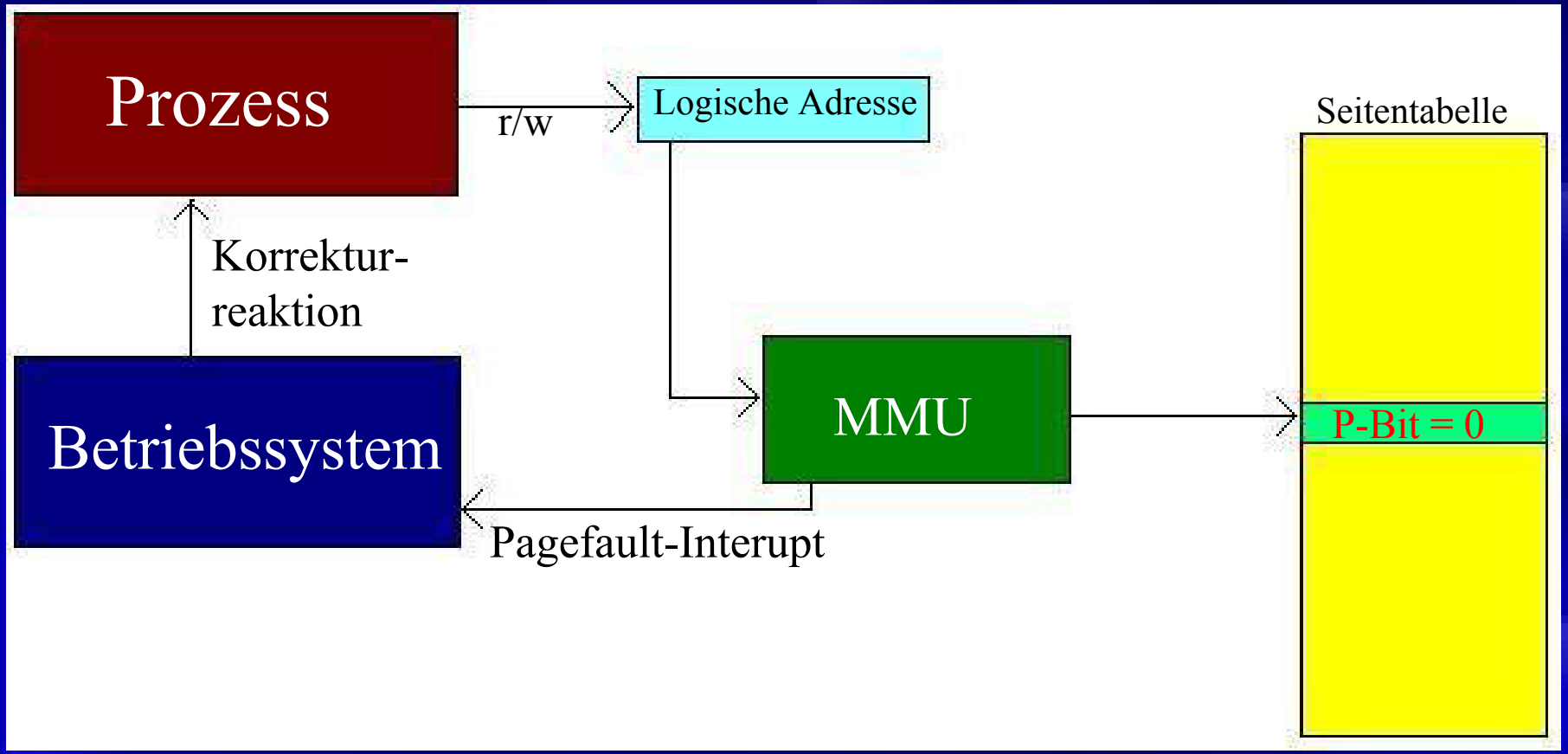
  d.h. Abbildung des Logischen auf den Physikalischen Adressraum

Adressräume:

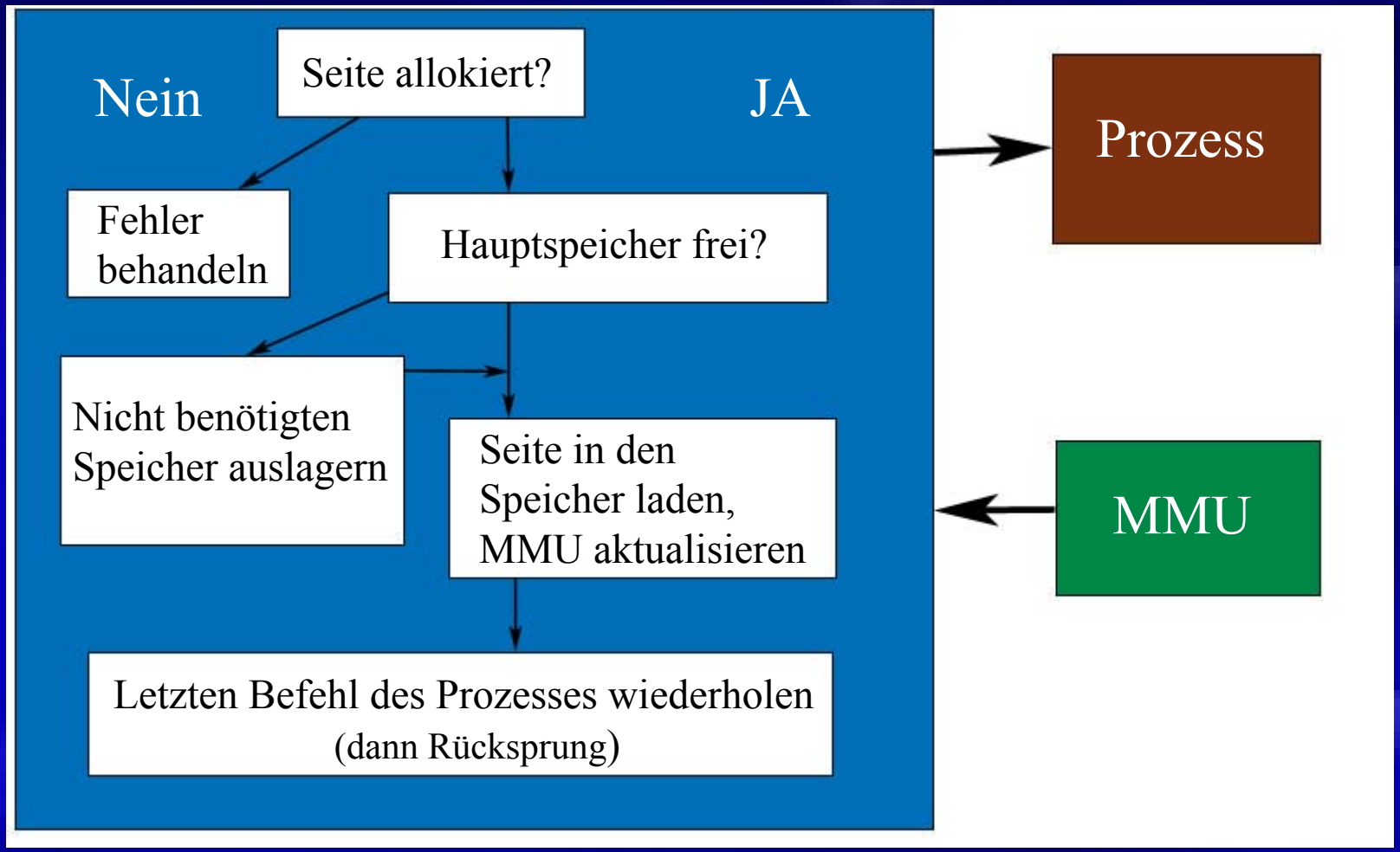
Ziele bei der Umsetzung von Adressräumen:

- Unterstützung mehrerer Adressräume
  - Bereitstellung von Stacks für die einzelnen Prozesse
- Schutz
  - des Betriebssystems vor Prozessen
  - der Prozesse voreinander
- mehr nutzbarer Speicher als real vorhanden
  - z.B. durch Speicherauslagerung

Wann tritt ein Seitenfehler auf?



Behandlung von Seitenfehlern:



Seitenwechsel auf Anfrage:

- Seiten erst laden, wenn sie gebraucht werden

Vorteile:

- Weniger E/A- Transfers benötigt
- Speichereffizienter
- schnellere Antwort
- mehr Prozesse

Copy on Write

Problemstellung:

- Häufiges Erzeugen nahezu gleicher Prozesse
  - Programmcode
  - Stack
  - Daten
- teuer Speicherseiten zu kopieren
- Kopie meistens unnötig
  - z.B. fork(), exec() – Fall
  - Daten werden nicht geändert / benutzt

Wie kann man das effizienter gestalten?

Copy on Write

Lösungsansatz:

- Prozesse teilen zunächst Speicher
- bei Schreibzugriff eines Prozesses Speicher duplizieren

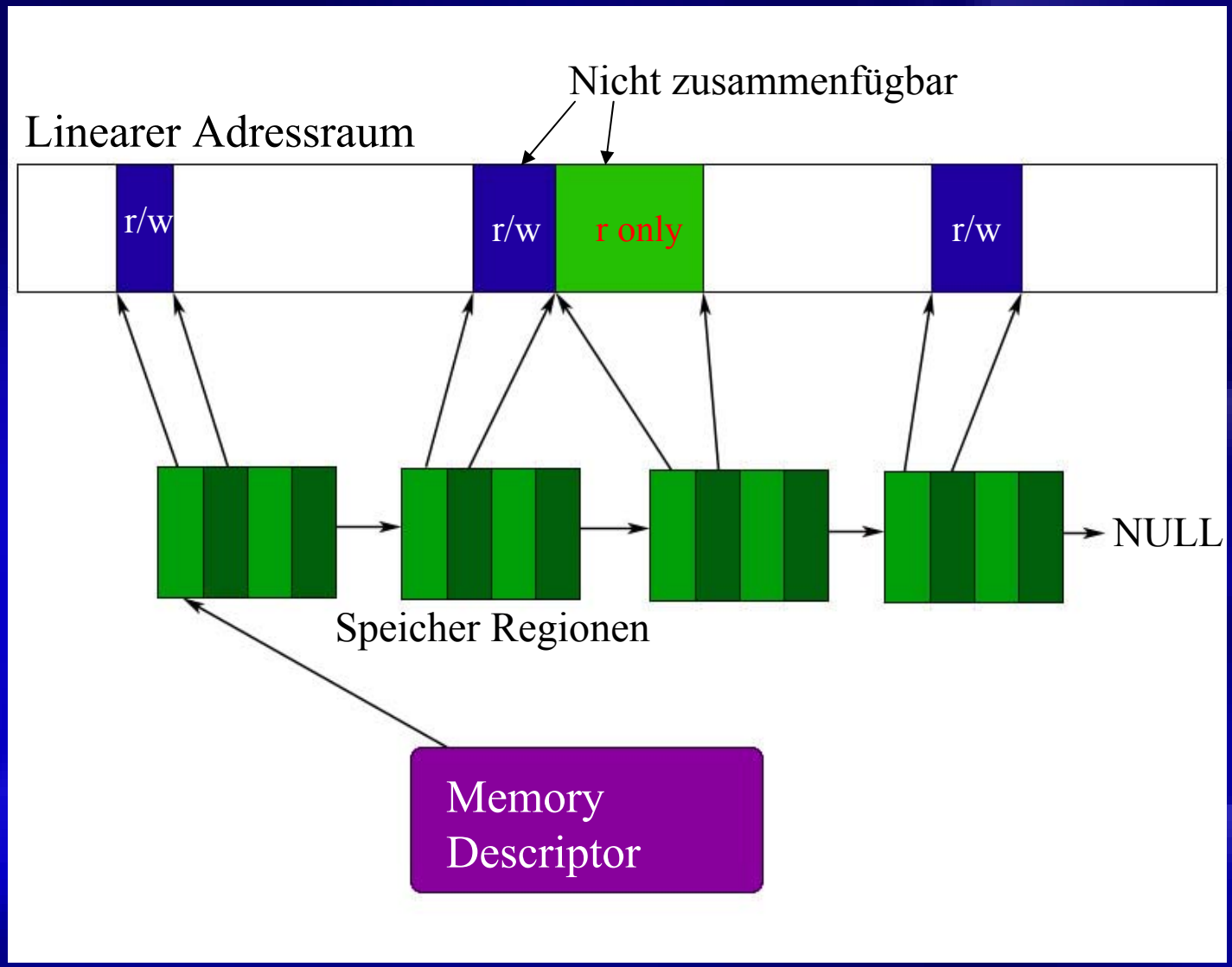
Process Memory Descriptor:

- jeder Prozess hat einen eigenen PMD
- enthält alle Adressräume eines Prozesses
- Typ: mm_struct
- Enthält z.B:
  - rrs: Anzahl der zugeordneten Seiten
  - mmap: Zeiger auf die erste Speicherregion

Speicherregionen:

- Repräsentiert durch Typ vm_area_struct
- Enthält z.B:
  - Zeiger auf den besitzenden PMD
  - Zeiger auf nächste SR (vm_next)
- Kernel fügt benachbarte SR zusammen, wenn möglich
- Verwaltung als Liste
- Bei größen Listen (>32 SR): RB-Trees

Anwendung unter Linux: Adressräume



Anwendung unter Linux: Page Faults

Tritt auf, wenn:

- der Stack wächst über Seitengrenze
- mehr Dynamischer Speicher angefordert wird
- Programmierfehler (ungültige Referenz)
- Seite nicht im Hauptspeicher

Arten von Seitenfehlern:

- Unterscheiden zwischen „guten“ und „bösen“ Speicherzugriffen
- Gute:
  - Zugriff auf ausgelagerte Seite (I)
  - Zugriff auf nie benutzte Seite (II)
    (vgl. Demand Paging)
  - Vergrößerung des Stack über Seitengröße (III)
- Böse:
  - Zugriff auf nicht allokierte Seite
  - Zugriff mit falscher Berechtigung (*)
  - Zugriff auf NULL

Anwendung unter Linux: Page Faults

Behandlung von Seitenfehlern:

- Böse:
  - (*): COW? Ja: cow behandeln
  - Sonst: Prozess terminieren
- Gute:

  zuerst freien Speicher suchen, wenn nötig belegten auslagern (einplanen), dann bei
  - (I): Seitentransfer einplanen
  - (II): Seite freigeben
  - (III): Seite an Stack anfügen

  letzten Befehl wiederholen und in den Prozess zurückspringen

Anwendung unter Linux: Demand Paging

- Prozess erhält auf Speicheranfrage nur logischen Speicher, keinen Physikalischen
- Speicher wird erst allokiert, wenn der Prozess darauf zugreift

Anwendung unter Linux: Copy on Write

Erzeugung eines neuen Prozesses erzeugt keinen neuen Adressraum

- Elternadressraum wird geteilt
- alle Seiten werden „Read only“ markiert
- Eltern o. Kind Prozess schreibt ->Pagefault
- Pagfault_handler erzeugt eine Kopie für den schreibenden Prozess

Zusammenfassung

- Verwaltung von Speicher Schutzfunktionen
- Geringe Speicherverschwendung
- Aufteilung des gesamten Systemspeichers
- Bereitstellung eines transparenten Interface

Fragen?