Übung zu Betriebssystemtechnik

Aufgabe 7: Copy-on-Write

09. Juli 2025

Dustin Nguyen, Maximilian Ott & Phillip Raffeck

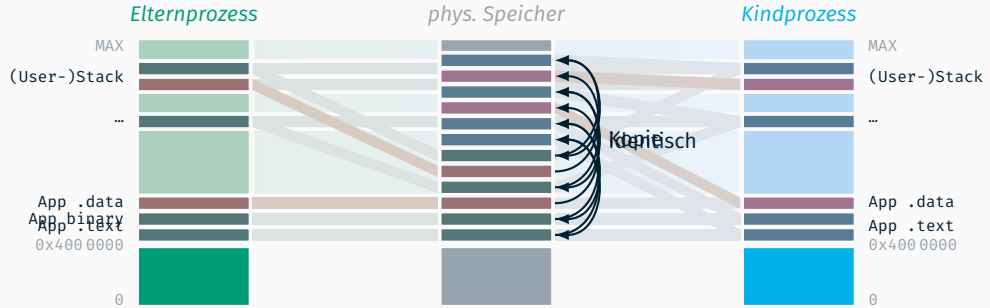
Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Technische Fakultät

StuBSmi soll durch das Vermeiden von unnötigen Kopieroperationen ressourcenschonender werden

Idee: Mitbenutzung identischer Seiten



Beispiel aus Aufgabe 5 (aber mit Seitengranularität)

- `fork()` dupliziert den aktuellen Prozess
- Kindprozess mit (tiefer) Kopie des virt. Speichers

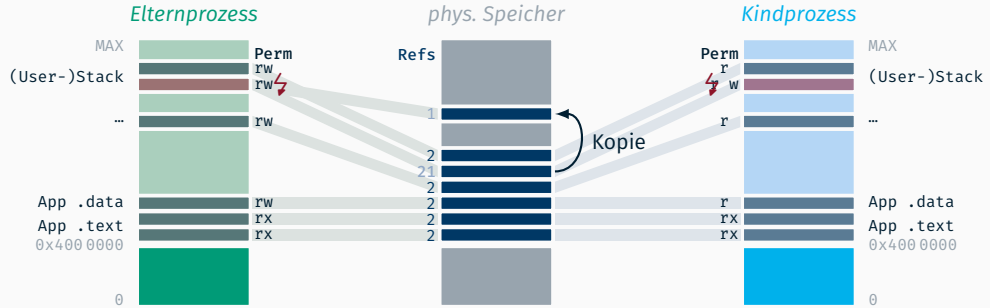
Bei der Ausführung wird jedoch nur auf einen Teil schreibend zugegriffen

- *im Beispiel:* Datensegment der App und Stapel
- viele kopierte Seiten bleiben weiterhin identisch

Wieso nicht gleich identische Seiten mitbenutzen?

Copy-on-Write

Copy-on-Write am Beispiel `fork()`



`fork()` mit (flacher) Kopie

- Referenz auf die gleiche phys. Seite im Kindprozess
- für alle beteiligte phys. Seiten werden die Referenzen gezählt
- den Prozessen wird Schreibzugriff auf die Seiten entzogen (ursprüngliche Zugriffsberechtigung muss aber gemerkt werden!)

Bei einem Schreibzugriff...

1. gibt es einen Seitenfehler → Sprung in `pagefault_handler`

Weitere Einsatzmöglichkeiten

Copy-on-Write kann in **StuBSmi** für alle Kopieroperationen verwendet werden, solange

- diese im Userspace ist
- Quell- & Zielseite die gleiche Ausrichtung haben
- sie eine (oder mehrere) volle Seite(n) umfasst

→ auch für IPC (`send-recv-reply`) nutzbar

- ggf. passende Ausrichtung erzwingen (→ GCC `aligned`-Attribut)
- Kombination aus Copy-on-Write und klassischem `memcpy`
  → Inhalte jenseits der Puffergrenze dürfen nicht kopiert werden!
- transparente Implementierung bei gutem Software-Engineering
  - generische `copy`-Funktion, welche anhand obiger Kritierien bei jeder Seite zwischen flacher und tiefer Kopie wählt
  - keine Anpassung an IPC (oder `fork`) Code notwendig

Umsetzung Referenzzähler

Referenzzähler

Für jede Seite im physikalischen Speicher (ab 64 MiB) wird potenziell ein Referenzzähler gebraucht.

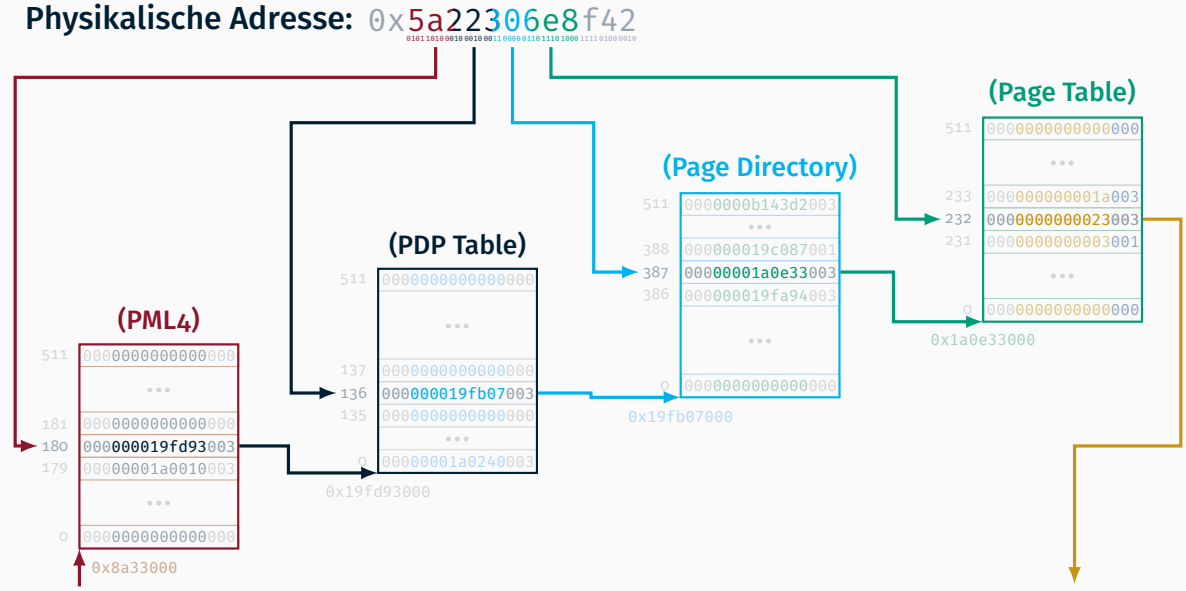
In **StuBSmi** gelten dabei zur Vereinfachung folgende Einschränkungen:

- Kompatibilität bis 256 TiB physikalischer Speicher (48 Bit)
- Unterstützung für bis zu einer Billion Referenzen pro Seite (40 Bit)

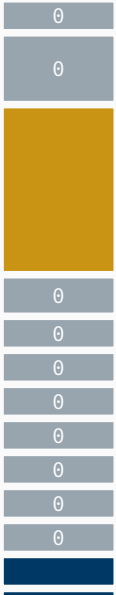
→ **Verwendung einer Schattenseitentabelle (shadow page table)**

- bestehende Implementierung (aus Aufgabe 3) wiederverwenden
- nun aber für physikalische (statt virtuelle) Seitenadressen
- in der *Page Table* (unterste Ebene) Referenzzähler statt der Zielseitenadresse speichern

Schattenseitentabelle für Referenzzähler am Beispiel



Eintrag in der Schattenseitentabelle (Shadow Page-Table)



Execute Disable

ignoriert

~~**Physikalische Adresse** der 4 KiB Zielseite~~

Referenzzähler für die korrespondierende physikalische Seite

ignoriert

Global

Page Size

Dirty

Accessed

Page-Level Cache Disable

Page-Level Write Through

User Mode

Writeable: nur lesender (0) oder auch schreibender (1) Zugriff

Present: Eintrag aktiv (1) oder inaktiv (0)

Fragen?

Fast geschafft – letzte Aufgabe!

- Donnerstag, 18. Juli: Zusammenfassung, Besprechung der Evaluation
- Freitag, 19. Juli: Letzte Rechnerübung