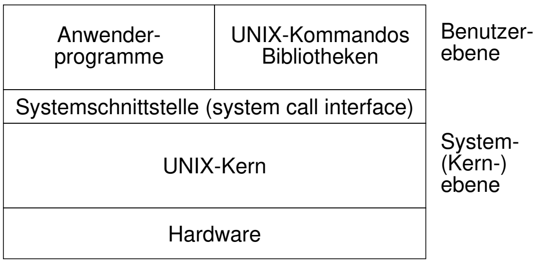
4 Cache-Speicher

C. Schimmel: "UNIX Systems for Modern Architectures", Addison-Wesley Publishing Company, 1994.

4.1 Überblick über relevante UNIX-Interna

❑ Grobstruktur

Benutzerebene - Systemebene



❑ Programm

Enthält auszuführende Befehlsfolge

❑ Prozeß

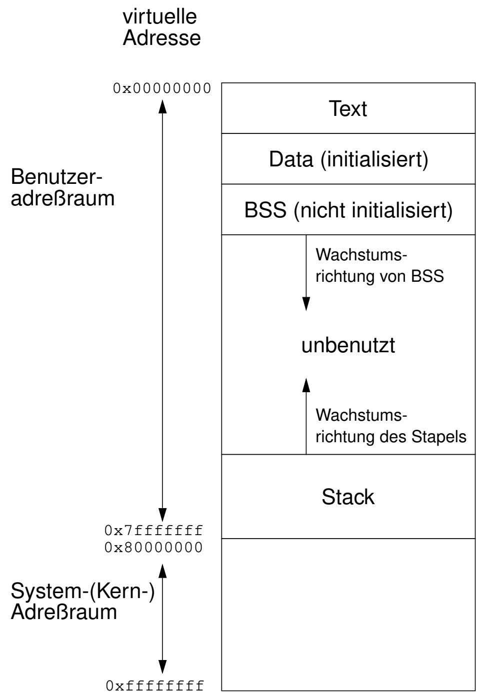
Ausführung eines Programms,
besteht aus Programm, Datenbereich, Hardwarezustand und Adreßraum

❑ Thread

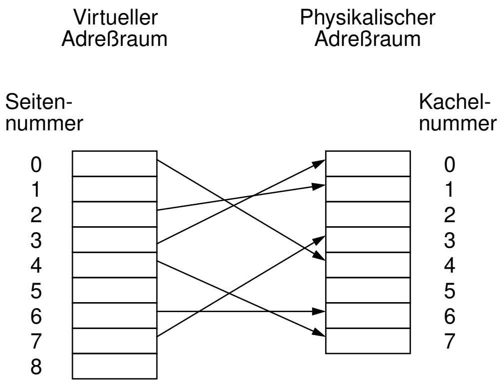
Sequentielle Befehlsausführung in einem Prozeß,
besteht aus Programm, Datenbereich, Hardwarezustand

❑ Der Adreßraum von Prozessen

Jeder Prozeß besitzt eigenen Adreßraum



❑ Adreßraumabbildung



❑ Kontext-Umschaltung

Sichern und Neueinstellen

- der Prozessorregister
- des Befehlszählers
- des Kellerzeigers
- der Adreßraumabbildung

❑ Thread-Umschaltung

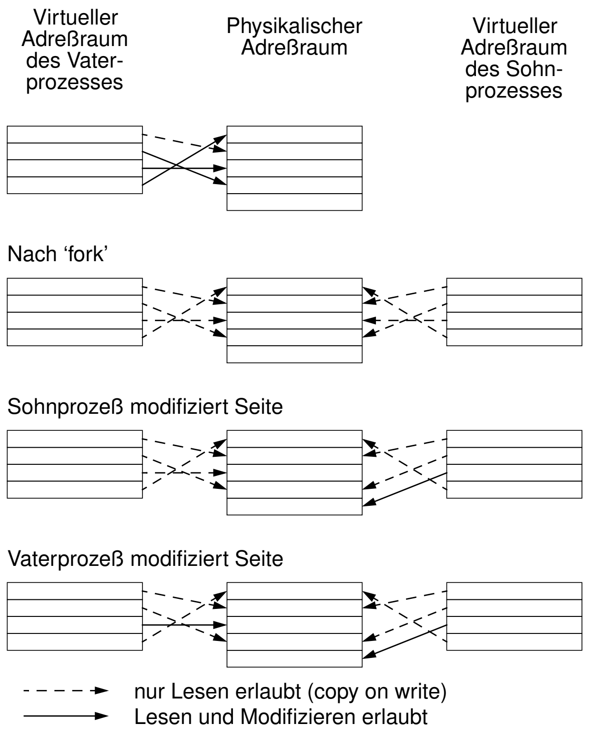
Sichern und Neueinstellen

- der Prozessorregister
- des Befehlszählers
- des Kellerzeigers

aber Beibehaltung der Adreßraumabbildung

❏ Systemaufrufe zur Speicher- und Prozeßverwaltung

- Der ‘fork’-Aufruf unter Benutzung von ‘copy on write’



- Der ‘exec’-Aufruf

  Ändert den Programmteil eines Prozesses, die Prozessorregister, den Befehlszähler und die Adreßraumabbildung.

  behält den Zustand bzgl. der meisten Unix-Dienste bei, wie geöffnete Dateien, ‘home directory’, ...

- Der ‘exit’-Aufruf

  Bewirkt zusammen mit der Tilgung eines Prozesses die Freigabe seines Adreßraums und das Abkoppeln von Unix-Diensten, z. B. durch Schließen seiner geöffneten Dateien.

  Überführung in den Zombie-Zustand

- Der ‘sbrk’- und ‘brk’-Aufruf

  Verändern die Größe des BSS-Segmentes.

  sbreak: Parameter gibt Veränderung der Segmentlänge an

  brk: Parameter gibt die kleinste, von BSS nicht mehr benutzte virtuelle Adresse an

- Gemeinsamer Speicher (shared memory)

  Einrichtung, indem in mehreren Adreßraum-abbildungen Verweise auf die gleiche Kachel eingetragen werden, möglicherweise unter unterschiedlichen virtuellen Adressen.

  Wird vor allem zur schnellen Kommunikation zwischen Prozessen genutzt.

  Systemaufrufe:

  shared memory operations

```
char *shmat(int shmid, char *shmaddr,
            int shmflg)
int shmdt(char *shmaddr)
```

  shared memory control operations

```
int shmctl (int shmid, int cmd,
            struct shmid_ds *buf)
```

  get shared memory segment identifier

```
int shmget(key_t key, int size, int shmflg)
```

- E/A-Operationen

  gepuffert - ungepuffert

- ‘mapped files’

  Verwendbar wie Felder eines Prozesses

  Wirken logisch ähnlich ‘shared memory’