

Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

VIII. Adressraummodelle: Mehradressraumsysteme

Wolfgang Schröder-Preikschat

14. Juni 2022



Einleitung

- Mehradressraumsystem
- Virtualität

Exklusionsmodell

- Prinzip

- Interaktion

 - Fenstertechniken
 - Spezialbefehle

Inklusionsmodell

- Prinzip

- Interaktion

 - Schutzgatter

Zusammenfassung



- Mehradressraumkonzept
 - Betriebssystem und Maschinenprogramme erhalten **private Adressräume**
 - die gleichsam Schutzdomänen manifestieren
 - Grundlage dafür ist die **Mehrfachnutzung** desselben realen Adressbereichs
 - für einen N -Bit Rechner gilt für diesen Adressbereich $A = [0, 2^N - 1]$
 - Adressraum als Entität, der eine gewisse **Virtualität** zugeschrieben wurde
 - d.h., die nicht physisch, aber in ihrer Funktionalität/Wirkung vorhanden ist
 - Technik ist die **Individualisierung** eines bestimmten Adressbereichs:
 - i A komplett *oder*
 - ii der obere oder untere Teilbereich von $A \rightsquigarrow \text{StuBS}_{BST}$
- **horizontale Isolation** einerseits, **vertikale Isolation** andererseits
 - horizontal** ■ innerhalb der Maschinenprogrammebene
 - Abkapselung des Maschinenprogramms als Prozessexemplar
 - vertikal** ■ Maschinenprogramm- und Betriebssystemebene übergreifend
 - Abkapselung von Prozessexemplar und Betriebssystem
- implizite Mitbenutzung (*sharing*) einer ganzen „Länderei“ (*user land*)
 - zur Beschleunigung von Interaktionen mit dem Betriebssystem



Virtualität von Adressen

Die Eigenschaft einer Sache, nicht in der Form zu existieren, in der sie zu existieren scheint, aber in ihrem Wesen oder ihrer Wirkung einer in dieser Form existierenden Sache zu gleichen.¹

- Bereiche des (begrenzten) physischen Adressraums individualisieren
 - i **Vervielfachung** von $A = [0, 2^N - 1]$
 - komplett A für Betriebssystem und allen Maschinenprogrammen, jeweils
 - ii **Partitionierung** von $A_t = [0, 2^N - 1]$, Vervielfachung von $A_p \subset A_t$
 - komplett A_t (total) für das Betriebssystem
 - komplett A_p (partiell) für alle Maschinenprogramme, jeweils
- Adressen dieser Bereiche sind nicht wirklich (physisch), wohl aber in ihrer Funktionalität vorhanden
 - hinter jeder dieser Adresse steht eine speicherabbildbare Entität
 - sie referenzieren Entitäten der Programmtexte (d.h., Befehle) oder -daten

↪ **Segmentierung** und/oder **Seitennummerierung** von Adressräumen

¹<http://de.wikipedia.org/wiki/Virtualität>

Einleitung

Mehradressraumsystem

Virtualität

Exklusionsmodell

Prinzip

Interaktion

Fenstertechniken

Spezialbefehle

Inklusionsmodell

Prinzip

Interaktion

Schutzgatter

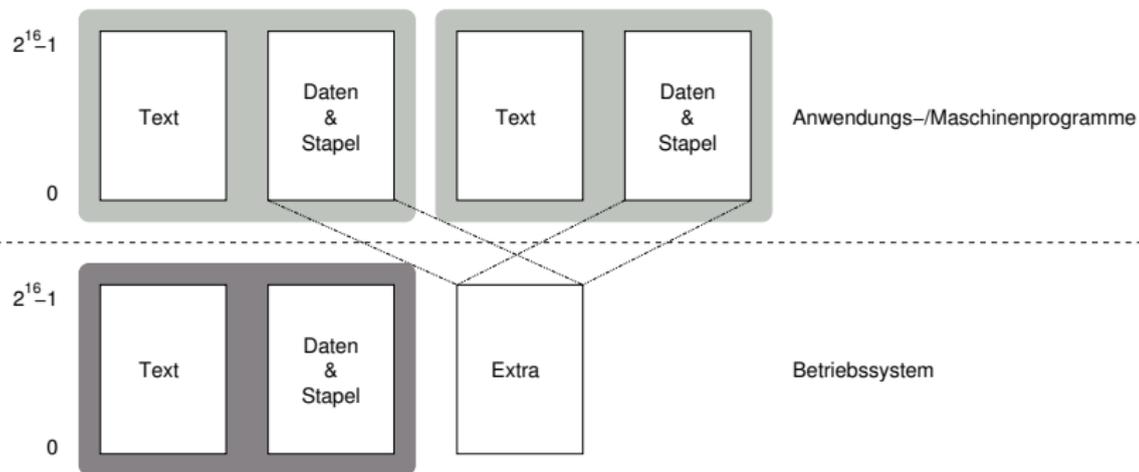
Zusammenfassung



Illusion von einem eigenen physischen Adressraum für Betriebssystem und Maschinenprogramme \leadsto **Exklusion**

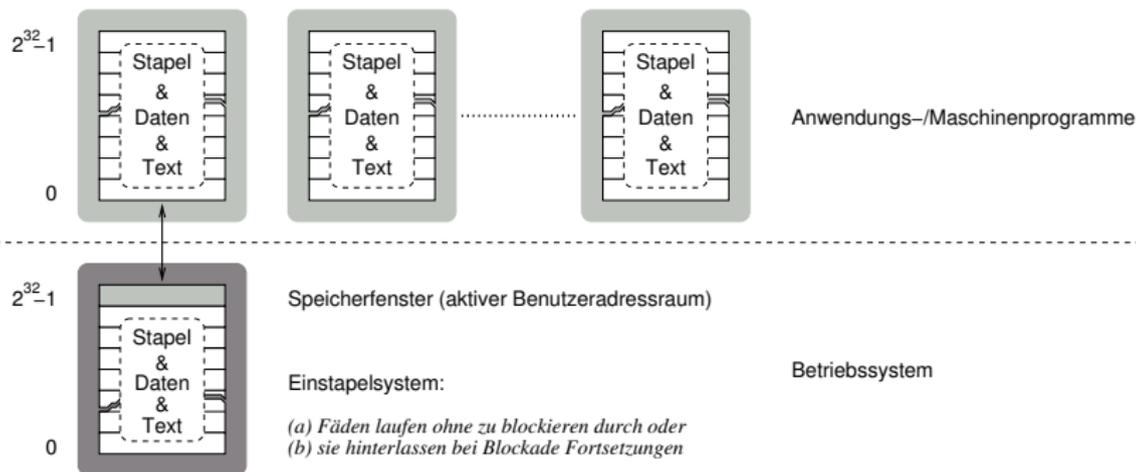
- Vervielfachung des Adressbereichs $A = [0, 2^N - 1]$
 - wobei N bestimmt ist durch die reale Adressbreite des Prozessors
 - evolutionär betrachtet galt/gilt z.B. für $N = 16, 20, 24, 31, 32, 48, 64$
- die MMU² verhindert ein Ausbrechen von Prozessen aus A
 - dies gilt für alle durch das Betriebssystem verwalteten Prozesse, *also*
 - sowohl für Maschinenprogramme als auch für das Betriebssystem selbst
- Informationstausch zwischen Programmen mittels **Maschinenbefehle**
 - des Betriebssystems für die Maschinenprogramme *und* \mapsto Ebene₃
 - Systemaufrufe zur Interprozesskommunikation oder Adressbereichsabbildung
 - der CPU für die Betriebssystemprogramme \mapsto Ebene₂
 - privilegierte Befehle zum Lese-/Schreibzugriff auf den Benutzeradressraum
- im Vordergrund steht die **strikte Isolation** von ganzen Adressräumen

²Ebenso eine MPU (*memory protection unit*).



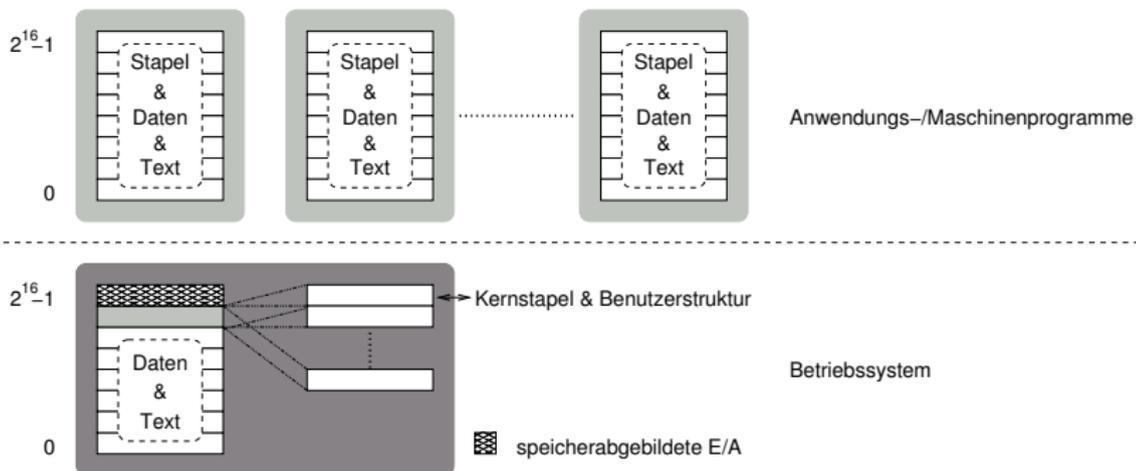
- horizontaler und vertikaler Informationsaustausch
 - horizontal ■ Interprozesskommunikation (Nachrichtenversenden)
 - vertikal ■ Zugriffe auf den Benutzeradressraum mittels Extrasegment
- Beispiele von (mikrokernbasierten) Betriebssystemen der Art:
 - QNX ■ ereignisbasiert, vgl. auch [8, 3]
 - AX ■ ereignis-/prozedurbasiert, QNX-kompatibel [9]





- horizontaler und vertikaler Informationsaustausch
 - horizontal
 - Interprozesskommunikation (Nachrichtenversenden)
 - seitenbasierte Mitbenutzung
 - vertikal
 - Zugriffe auf den Benutzeradressraum mittels Speicherfenster
- prominentes Beispiel eines Betriebssystems der Art:
 - Darwin
 - Hybridkernansatz, ereignis-/prozedurbasiert, 512 MiB Fenster





- horizontaler und vertikaler Informationsaustausch
 - horizontal ■ Interprozesskommunikation (Nachrichtenversenden, *pipe*)
 - seitenbasierte Mitbenutzung in Inkrementen von 64 Bytes
 - vertikal ■ Zugriffe auf den Benutzeradressraum mittels Spezialbefehle
- prominentes Beispiel eines (monolithischen) Betriebssystems der Art:
 - UNIX ■ Version 6 [6, 5], prozedurbasiert



- **adressraumübergreifender Zugriff** durch **Spezialbefehle** der CPU
 - `mfp` `ea` ■ *move from previous instruction space*
 - das von `ea` gelesene Datum wird auf den Kernstapel abgelegt
 - `mtp` `ea` ■ *move to previous instruction space*
 - das an `ea` zu schreibende Datum wird vom Kernstapel geholt
 - ↪ `ea` ist *effektive Adresse* im Adressraum vor Kernaktivierung
- **ungültige Benutzeradressen** können diese Befehle scheitern lassen !!!
 - der Kern muss daraufhin seinerseits sein mögliches Scheitern abwenden
- im MMU-Fall wird die Prozessorbetriebsart verfolgt (vgl. [1, S. 2-4]):
 - `00` ■ *kernel mode* \leadsto privilegiert, sicher, vertrauenswürdig
 - `11` ■ *user mode* \leadsto unprivilegiert, unsicher, zweifelhaft
- dafür vorgesehene Bitfelder im Prozessorstatuswort:
 - `15-14` ■ *current mode*
 - `13-12` ■ *previous mode*, vor der letzten Unterbrechung (*trap*, *interrupt*)
- je Betriebsart 8 Seitenadressregister, Umschaltung bei Moduswechsel



Vorbeugungsmaßnahmen, um ein mögliches Scheitern des Kerns – und damit einen **Systemabsturz** (*crash*) – abzuwenden

- sind betriebssystemabhängig & gehen optimistisch/pessimistisch vor
 - optimistisch**
 - Annahme: *ea* ist eine eher gültige Benutzeradresse
 - den Kern auf mögliche Unterbrechung (*trap*) vorbereiten
 - im Ausnahmefall die Ausführung des Kerns fortsetzen
 - pessimistisch**
 - Annahme: *ea* ist eine eher ungültige Benutzeradresse
 - diese vor Verwendung mit dem Spezialbefehl überprüfen
 - den Ausnahmefall im Kern nicht auftreten lassen
- im Falle eines voll verdrängbaren Kerns offenbart der pessimistische Ansatz eine **Laufgefahr** (*race hazard*)
 - angenommen, die Prüfung ergab die Gültigkeit der Benutzeradresse *ea*
 - vor Verwendung von *ea* im Kern wird der Prozess aber verdrängt
 - asynchron dazu verändert sich die Adressraumbelegung des Prozesses
 - so dass Benutzeradresse *ea* bei Prozesswiederaufnahme ungültig ist

↪ **beachte:** ein Betriebssystem darf Benutzerprogramme nicht „benutzen“



```
1 gword: ; note: r1 holds read address
2   mov PS, -(sp) ; save processor status word
3   bis $340, PS ; disable interrupts
4   mov nofault, -(sp) ; save kernel detour pointer
5   mov $err, nofault ; setup detour for trap handler
6   mfpi (r1) ; transfer data onto kernel stack
7   mov (sp)+, r0 ; receive data just read
8   br 1f ; prepare for return...
9 pword: ; note: r1 holds write address
10  mov PS, -(sp) ; save processor status word
11  bis $340, PS ; disable interrupts
12  mov nofault, -(sp) ; save kernel detour pointer
13  mov $err, nofault ; setup detour for trap handler
14  mov r0, -(sp) ; send data to be written
15  mtpi (r1) ; transfer data from kernel stack
16 1:
17  mov (sp)+, nofault ; restore kernel detour pointer
18  mov (sp)+, PS ; restore processor status word
19  rts pc
```



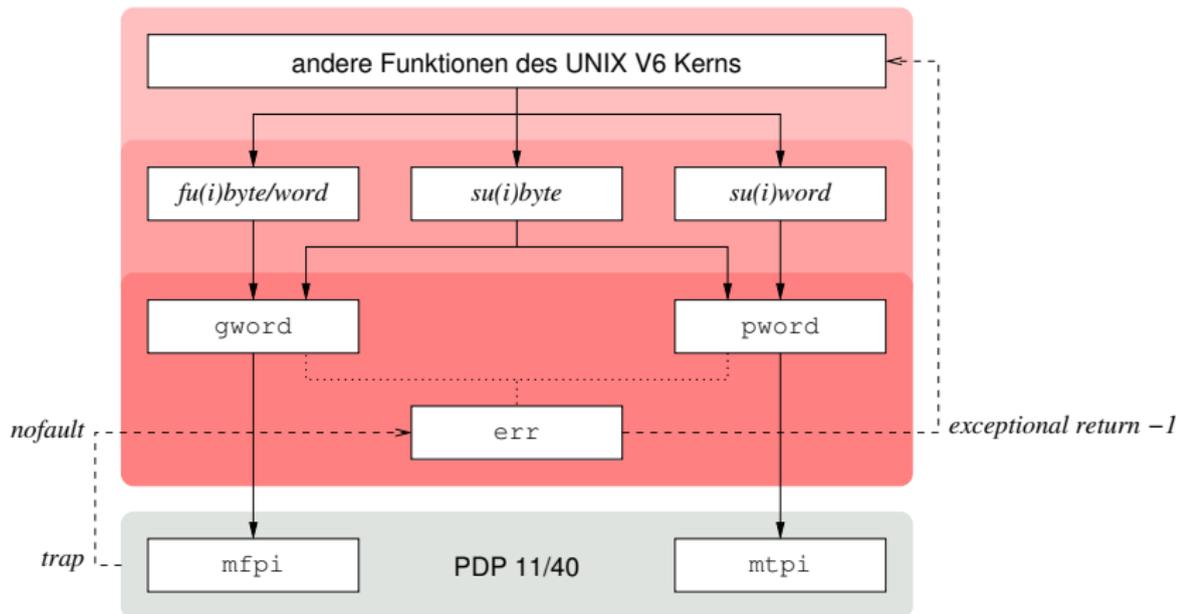
- Abfangen der Ausnahmesituation (*trap*): *mfp*/*mtp* ist gescheitert

```
1 trap:
2     mov    PS, -4(sp)      ; magic: needed for code 1f
3     tst    nofault        ; check for kernel detour
4     beq   1f              ; none, continue
5     mov   $1, SSR0        ; else, reinitialize MMU
6     mov   nofault, (sp)   ; overwrite return address and
7     rtt                                ; return from trap
8 1:                                ; fall into normal trap...
```

- „*escape exception*“ [2] für *gword*/*pword*-rufende Programme

```
1 err:
2     mov   (sp)+, nofault  ; restore kernel detour pointer
3     mov   (sp)+, PS      ; restore processor status word
4     tst   (sp)+          ; forget g/pword return address
5     mov   $-1, r0        ; indicate failure of operation
6     rts   pc             ; return from caller of g/pword
```





- im Ausnahmefall werden die *fetch/store*-Operationen abgebrochen
 - sie bilden zusammen mit *gword*, *pword* und *err* ein Modul [7]
 - denn: alle teilen sich dasselbe Wissen über Stapelaufbau/Ereignisabfolge



Einleitung

Mehradressraumsystem

Virtualität

Exklusionsmodell

Prinzip

Interaktion

Fenstertechniken

Spezialbefehle

Inklusionsmodell

Prinzip

Interaktion

Schutzgatter

Zusammenfassung

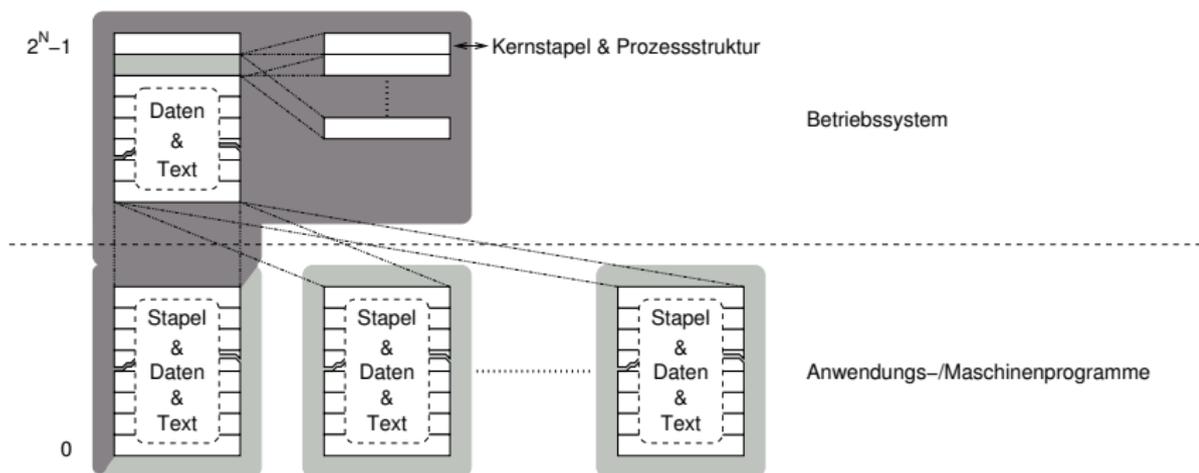


Partiell private Adressräume

Illusion von einem eigenen physischen Adressraum bzw. -bereich für die Maschinenprogramme \leadsto **Inklusion** des Betriebssystem(kern)s

- Vervielfachung des Adressbereichs $A_p \subset A_t$
 - A_t ist der dem Betriebssystem *total* zugeordnete Adressbereich
 - existiert einfach, aber mit A_p als integrierten variablen (mehrfachen) Anteil
 - A_p ist der einem Maschinenprogramm in A_t *partiell* zugeordnete Bereich
 - existiert mehrfach, einmal für jedes Anwendungs- bzw. Maschinenprogramm
- der Benutzeradressraum ist ein Teil des Betriebssystemadressraums
 - die MMU verhindert ein Ausbrechen von Prozessen aus A_p und A_t , nicht jedoch deren Eindringen heraus aus $A_t - A_p$ und hinein in A_p
 - bedingter Schreibschutz von A_p für A_t dämmt **Betriebssystemfehler** ein
 - dabei erstreckt sich A_p über den oberen oder unteren Bereich von A_t
 - nur Prozesswechsel³ zwischen A_p bedingen das Umschalten der MMU
- im Vordergrund steht **Performanz** (Geschwindigkeit, Rauschen)
 - Systemaufrufe implizieren **keinen Adressraumwechsel** mehr

³Genauer: Der Wechsel zwischen schwergewichtigen Prozessen.

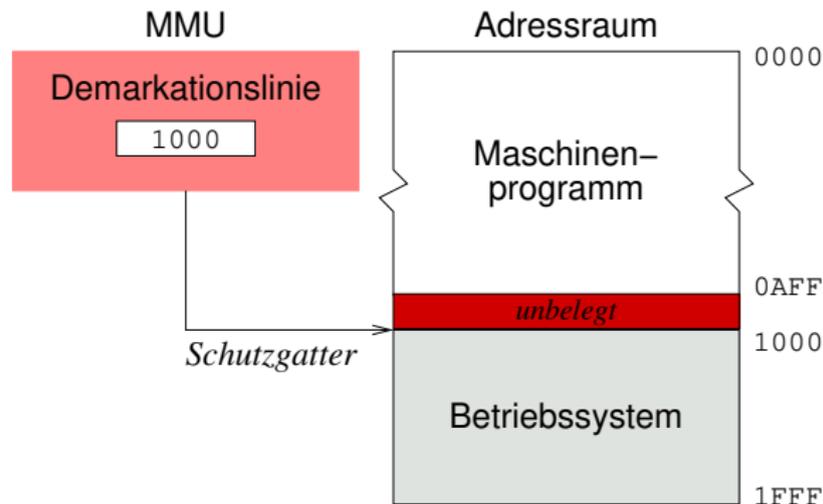


- horizontaler und vertikaler Informationsaustausch
 - horizontal ■ Interprozesskommunikation, Segment-/Seitenmitbenutzung
 - vertikal ■ **speicherabgebildeter Zugriff** auf den Benutzeradressraum
- prominente Beispiele von Betriebssystemen der Art:
 - Linux ■ monolithisch, prozedurbasiert
 - NT ■ Hybridkernansatz, prozess-/prozedurbasiert
 - macOS ■ ab Mac OS X 10.15 (Catalina): 32 \rightsquigarrow 64-Bit Technologie

Inklusion des Benutzeradressraums in den Betriebssystemadressraum ist nur bei hinreichend großem N ein sinnvoller Ansatz

- das Modell wurde attraktiv mit Adressbreiten von $N \geq 30$ Bits
 - also für reale Adressbereiche ab 1 GiB Speicherumfang
- typische Aufteilung von $A = [0, 2^{32} - 1] = 4$ GiB:
 - gleich** ■ 2 GiB jeweils für Benutzer- und Betriebssystemadressraum
 - NT
 - ungleich** ■ 3 GiB Benutzer- und 1 GiB Betriebssystemadressraum
 - Linux, NT (Enterprise Edition)
 - steht und fällt mit der Größe von Benutzerprogrammen/-prozessen
- die Folge einer solchen Aufteilung ist, dass die Benutzerprozesse dem Betriebssystem ein **stärkeres Vertrauen** schenken müssen
 - Schreibschutz auf A_p legen und nur bei Bedarf zurücknehmen/lockern
 - sonst sind Zeigerfehler in $A_t - A_p$ verheerend für Programme in A_p





■ Arbeitsmodi

- nur im unprivilegierten Modus ist das Schutzgatter aktiv
- nur im privilegierten Modus sind Zugriffe auf Bereiche jenseits der Demarkationslinie erlaubt

Reminiszenz (vgl. [4, S. 15])

Eine alte, bekannte Technik, die nach wie vor hohe Bedeutung genießt.



Einleitung

- Mehradressraumsystem
- Virtualität

Exklusionsmodell

- Prinzip
- Interaktion
 - Fenstertechniken
 - Spezialbefehle

Inklusionsmodell

- Prinzip
- Interaktion
 - Schutzgatter

Zusammenfassung



- private Adressräume
 - Systemaufrufe allein rufen bereits Adressraumwechsel hervor
 - vertikaler Informationsaustausch ist nur indirekt möglich
 - + schränken den effektiv zur Verfügung stehenden Adressbereich nicht ein
 - + stellen geringere Anforderungen an die Korrektheit des Betriebssystems

 - partiell private Adressräume
 - + Systemaufrufe gehen ohne Adressraumwechsel einher
 - + vertikaler Informationsaustausch ist direkt möglich
 - verkleinern den effektiv zur Verfügung stehenden Adressbereich
 - stellen höhere Anforderungen an die Korrektheit des Betriebssystems
- ↪ **Zielkonflikt** (*tradeoff*)
- keiner der beiden Ansätze ist *per se* ein Vorteil einzuräumen. . .



- Mehradressraumsysteme
 - implementieren (total/partiell) private Adressräume
 - total – sowohl für Maschinenprogramme als auch Betriebssystem
 - partiell – nur für die Maschinenprogramme untereinander
 - Informationsaustausch zwischen Betriebssystem- & Benutzeradressraum:
 - fensterbasiert, bedarfsorientierte Einblendung von Adressraumabschnitten
 - spezialbefehlbasiert, selektives Kopieren von Maschinenwörtern
 - adressraumgeteilt, direkter Zugriff von kompletten Benutzeradressraum
 - Isolation von Adressräumen in zweierlei Hinsicht
 - horizontale Isolation innerhalb der Maschinenprogrammebene
 - Abkapselung des Maschinenprogramms als Prozessexemplar
 - Wechsel zwischen den Prozessexemplaren impliziert Adressraumwechsel
 - vertikale Isolation des Betriebssystems vom Maschinenprogramm
 - Abkapselung von Prozessexemplar und Betriebssystem
 - Wechsel zwischen den Ebenen impliziert keinen Adressraumwechsel
- ↪ Aufweichen der Privatsphäre, wenn Performanz vor Schutz gehen soll



- [1] DIGITAL EQUIPMENT CORPORATION (Hrsg.):
PDP-11/40 Processor Handbook.
Maynard, MA, USA: Digital Equipment Corporation, 1972.
(D-09-30)
- [2] GOODENOUGH, J. B.:
Exception Handling: Issues and a Proposed Notation.
In: *Communications of the ACM* 18 (1975), Nr. 12, S. 683–696
- [3] HILDEBRAND, D. :
An Architectural Overview of QNX.
In: *Proceedings of the USENIX Workshop on Micro-kernels and Other Kernel Architectures (USENIX Microkernels)* USENIX Association, 1992. –
ISBN 1–880446–42–1, S. 113–126
- [4] KLEINÖDER, J. ; SCHRÖDER-PREIKSCHAT, W. :
Stapelverarbeitung.
In: LEHRSTUHL INFORMATIK 4 (Hrsg.): *Systemprogrammierung*.
FAU Erlangen-Nürnberg, 2015 (Vorlesungsfolien), Kapitel 7.1



- [5] LIONS, J. :
A Commentary on the Sixth Edition UNIX Operating System.
The University of New South Wales, Department of Computer Science, Australia :
<http://www.lemis.com/grog/Documentation/Lions>, 1977
- [6] LIONS, J. :
UNIX Operating System Source Code, Level Six.
The University of New South Wales, Department of Computer Science, Australia :
<http://v6.cuzuco.com>, Jun. 1977
- [7] PARNAS, D. L.:
On the Criteria to be used in Decomposing Systems into Modules.
In: *Communications of the ACM* 15 (1972), Dez., Nr. 12, S. 1053–1058
- [8] QUANTUM SOFTWARE SYSTEMS LTD. (Hrsg.):
QNX Operating System User's Manual.
Version 2.0.
Toronto, Canada: Quantum Software Systems Ltd., 1984
- [9] SCHRÖDER, W. :
Eine Familie von UNIX-ähnlichen Betriebssystemen – Anwendung von Prozessen und des Nachrichtenübermittlungskonzeptes beim strukturierten Betriebssystementwurf,
Technische Universität Berlin, Diss., Dez. 1986

