Speicherzuteilungsstrategien: Grundlegende Überlegungen (1/2)

- Je **weniger Platz für den einzelnen Prozess** zur Verfügung steht, desto **mehr Prozesse** können im Hauptspeicher resident sein
  (➔ Seitenzahl pro Prozess minimieren!)
- Stehen **einem Prozess zu wenig Seiten** zur Verfügung, dann wird die **Seitenfehlerrate** trotz Lokalitätsprinzip sehr hoch sein
  (➔ Seitenzahl pro Prozess maximieren!)
- **Über eine bestimmte Größe** hinaus wird sich zusätzlicher Speicher nur **geringfügig** auf die Seitenfehlerrate auswirken
  (➔ Seitenzahl pro Prozess optimieren!)

Speicherzuteilungsstrategien: Grundlegende Überlegungen (2/2)

- Verteilung der freien Hauptspeicherseiten auf die existierenden Prozesse:
  - Einem Prozess wird eine **feste Anzahl** von Hauptspeicherseiten zugebilligt.
  - Einem Prozess werden während seiner Ausführung abhängig von seinem Verhalten (Seitenfehlerrate) eine **variable Anzahl** von Hauptspeicherseiten zur Verfügung gestellt.
- Strategien nach Auftreten von Seitenfehlern :
  - Bei **lokalen Strategien** muss eine Seite des aktiven Prozesses ersetzt werden.
  - Bei **globalen Strategien** sind alle Seiten im Hauptspeicher Kandidaten für die Ersetzung.

fest variabel lokal global

Speicherzuteilung:
V1: Feste Seitenanzahl und lokale Strategie

lokal fest

- Einem Prozess steht für seine Ausführung eine feste Anzahl von Hauptspeicherseiten zur Verfügung.
- Tritt ein Seitenfehler auf, dann muss das Betriebssystem entscheiden, welche andere Seite dieses Prozesses ersetzt werden muss
  - Das wesentliche Problem liegt in der Festlegung der Seitenzahl (x % der gesamten Prozessgröße?)

Speicherzuteilung:
V2: Variable Seitenzahl und globale Strategie

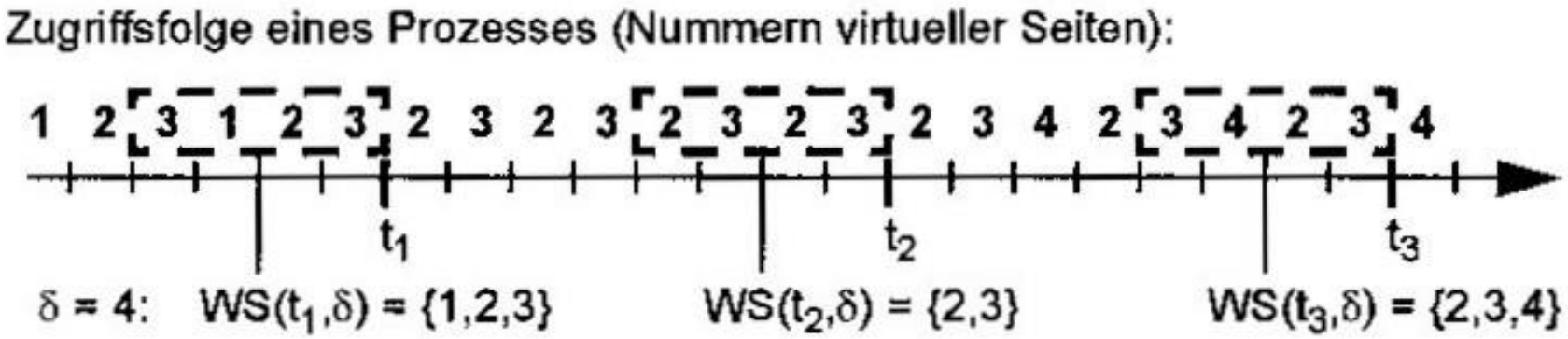
global variabel

- Den im Hauptspeicher befindlichen Prozessen stehen eine variable Anzahl von Seiten zur Verfügung.
- Seitenfehler → Zuweisung einer freien Seite, falls verfügbar
- Keine freien Seiten mehr verfügbar → Seite zur Ersetzung auswählen (beliebiger Prozess!)
- Zu viele Prozesse im Hauptspeicher → u.U. sind die zur Verfügung stehenden Speicherbereiche nicht mehr ausreichend und die Seitenfehlerrate wird sehr groß
- Jedes Programm, das zusätzliche Seiten benötigt, erhält diese auf Kosten von Seiten anderer Programme. Da diese aber ebenfalls noch benötigt werden, werden in immer schnellerer Folge weitere Seitenfehler erzeugt. (Seitenflattern, engl. „**Thrashing**“). Die Prozesse verbrauchen dann für das Seitenwechseln mehr Zeit als für ihre Ausführung.

V3: Variable Seitenzahl und lokale Strategie: Working Set-Strategie -> Working Set - Modell

lokal variabel

- Der **Working Set WS (t,δ)** eines Prozesses zur Zeit t ist die Menge aller Seiten, die er bei den letzten δ Speicherzugriffen angesprochen hat.
- Der Umfang des Working Set ist abhängig von δ. Im Extremfall umfasst der Working Set das gesamte Programm
- Je größer der Working Set, umso geringer die Seitenfehlerrate
- Beispiel:



V3: Variable Seitenzahl und lokale Strategie: Working Set-Strategie

lokal variabel

- Der Working Set jedes Prozesses wird beobachtet.
- Seitenfehler → Zuweisung einer freien Seite, falls verfügbar
- Periodisch wird die zugewiesene Seitenanzahl an den Working Set angepasst: diejenigen Seiten werden aus dem Speicher entfernt, die nicht mehr zum Working Set gehören
- Ein Prozess darf nur dann ausgeführt werden, wenn sein Working Set im Hauptspeicher resident ist

Mögliche Implementierung (grobe Annäherung):

- Periodisch werden diejenigen Seiten aus dem Speicher entfernt, deren Reference-Bit = 0 ist
  (kein Zugriff erfolgt seit letzter Überprüfung).
- Seiten mit Reference-Bit =1 (~ Working Set):
  Reference-Bit zurücksetzen!

Alternative / Ergänzung: Page-Fault Frequency (PFF) - Strategie

- Für die Seitenfehlerrate werden untere und obere Grenzen definiert.
- Seitenfehler → Ersetzung einer Prozess-eigenen Seite
- Wird die untere Grenze erreicht, dann werden dem Prozess Speicherseiten weggenommen
- Erreicht ein Prozess in Ausführung die obere Grenze, dann werden ihm – wenn möglich – neue Speicherseiten hinzugefügt. Sind keine weiteren Speicherseiten verfügbar, so muss der Prozess suspendiert und ausgelagert werden.

Aufgaben Abschnitt 4.2.c): Pagingstrategien

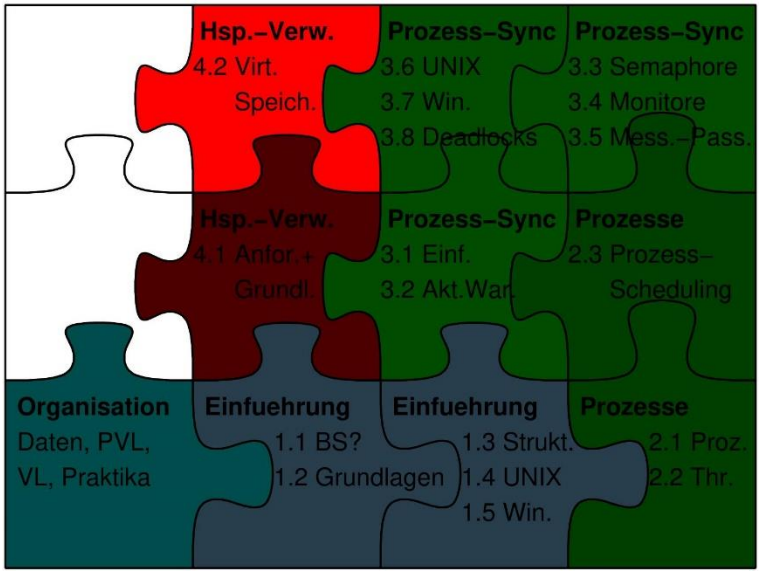
Quelle: Tanenbaum, Moderne Betriebssysteme, 3. Auflage, Kapitel 3, Aufgabe 31

31. Ein Computer stellt jedem Prozess einen Adressraum von 64 KB zur Verfügung, aufgeteilt in 4-KB-Seiten. Ein Programm hat 32.768 Byte Programmcode, 16.386 Byte Daten und 15.870 Byte Stack. Passt dieses Programm in den Adressraum? Würde es passen, wenn die Seiten 512 Byte groß wären? Denken Sie daran, dass eine Seite nicht zwei Teile von verschiedenen Segmenten enthalten kann.

Anmerkung: Aufgrund von Rechten liegen Programm-, Daten- und Stacksegment nicht auf gleichen Seiten.

Kapitel 4
Hauptspeicher-Verwaltung

1. Anforderungen und Grundlagen
2. Virtueller Speicher
   a) Einführung und Prinzipien
   b) Paging
   c) Pagingstrategien
   d) Unix / Windows
   e) Meltdown / Spectre



Virtueller Speicher in UNIX: Paging

- Speicherzuteilung: **Variable** Seitenzahl und **globale** Strategie
- Seitenfehler → Zuweisung einer freien Seite
- **Page Daemon** (Prozess-ID 2)
  - Wacht alle 250 ms auf
  - Falls Anzahl freier Seiten zu klein: Verwendet modifizierten Clock-Algorithmus, um **beliebigen** Prozessen Seiten zu entziehen
- **Swapper**
  - lagert Prozesse aus, falls zu viele Seitenfehler auftreten
  - lagert erst wieder ein, wenn genügend freie Seiten zur Verfügung stehen (Scheduling!)

Virtueller Speicher in Windows: Paging (1/2)

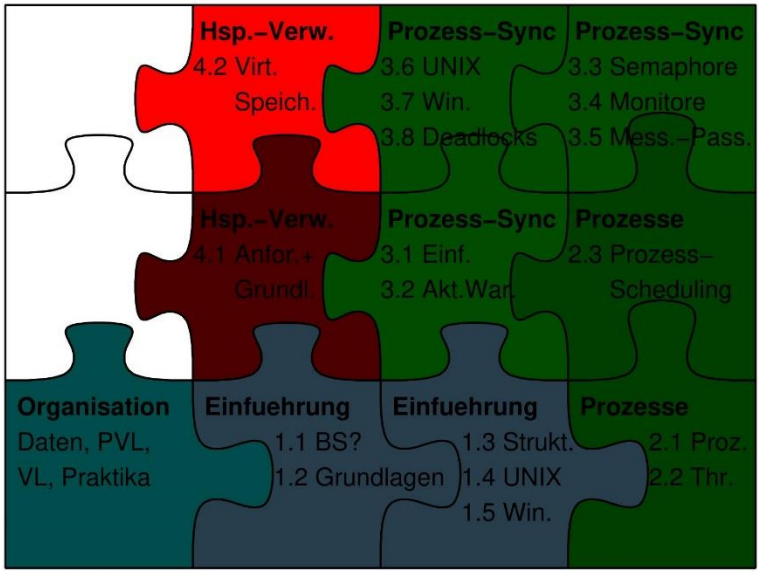
- Speicherzuteilung: **Variable** Seitenzahl und **lokale** Strategie
- **Eigene Working Set-Definition:** Alle Seiten des Prozesses, die sich momentan im Hauptspeicher befinden
- Ein Working-Set hat eine **minimale** und **maximale** Grenze (bei Start für alle Prozesse gleich)
- Seitenfehler:
  - **Working Set < Maximum** → **neue** Seite hinzufügen
  - sonst: Seite aus Working Set **ersetzen**! (lokal)
  - bei vielen Seitenfehlern: Maximum für den Prozess erhöhen (Kombination mit PFF-Strategie)

Virtueller Speicher in Windows: Paging (2/2)

- **Balance-Set-Manager**
  - Wacht einmal pro Sekunde auf
  - Falls Anzahl freier Seiten zu klein: Startet **Working-Set-Manager**, um Prozessen Seiten zu entziehen
- **Working-Set-Manager**
  - Untersucht alle Prozesse in definierter Reihenfolge (große Prozesse vor kleinen, Hintergrundprozesse vor Vordergrundprozessen)
  - Falls **Working Set < Minimum** oder bereits **viele Seitenfehler** → **ignorieren**
  - Sonst: Prozess-Seiten anhand von modifiziertem Clock-Algorithmus aus Hauptspeicher (Working Set) **entfernen**

Kapitel 4
Hauptspeicher-Verwaltung

1. Anforderungen und Grundlagen
2. Virtueller Speicher
   a) Einführung und Prinzipien
   b) Paging
   c) Pagingstrategien
   d) Unix / Windows
   e) Meltdown / Spectre



Virtueller Speicher: Sicherheitslücken

- Typische Angriffe: **Buffer Overruns** und **Buffer Underruns** (etwa in C/C++: Zugriff auf Strings ohne Längenbegrenzung):
  `for (i=0; userEingabe[i]!=0; i++) verarbeite(i);`
- → immer mit Längenbegrenzung (z.B. `strncpy` statt `strcpy`)!
- Immerhin: Sicherheitsfunktionen der Prozessoren
- **Aber:** Immer wieder Sicherheitslücken
  - Dabei auch genutzt: **Seitenkanalangriffe** (Side Channel Attacks)
- Veröffentlicht Januar 2018: **Meltdown + Spectre**
  - Seitenkanalangriff, nutzt Paging-Mechanismen, Cache, TLB
  - Grundlage bereits in alten Architekturen → sehr verbreitet!

Ende des 4. Kapitels: Was haben wir geschafft?

4. Hauptspeicher-Verwaltung

4.1 Anforderungen und Grundlagen

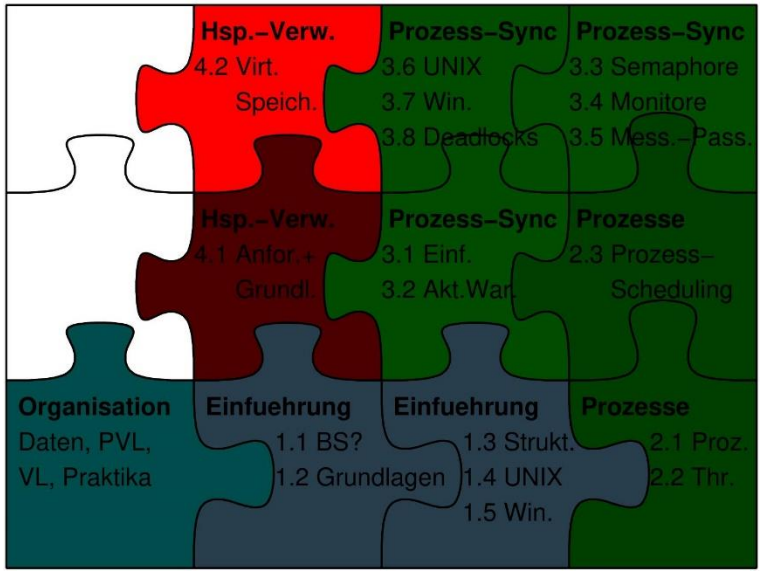
4.1.1 Anforderungen

4.1.2 Adressberechnung

4.1.3 Einfache Zuweisungs- und Freigabeverfahren

4.1.4 Implementierungsaspekte

4.2 Virtueller Speicher



Ende des 4. Kapitels:
Was haben wir geschafft?

4. Hauptspeicher-Verwaltung

4.1 Anforderungen und Grundlagen

4.2 Virtueller Speicher

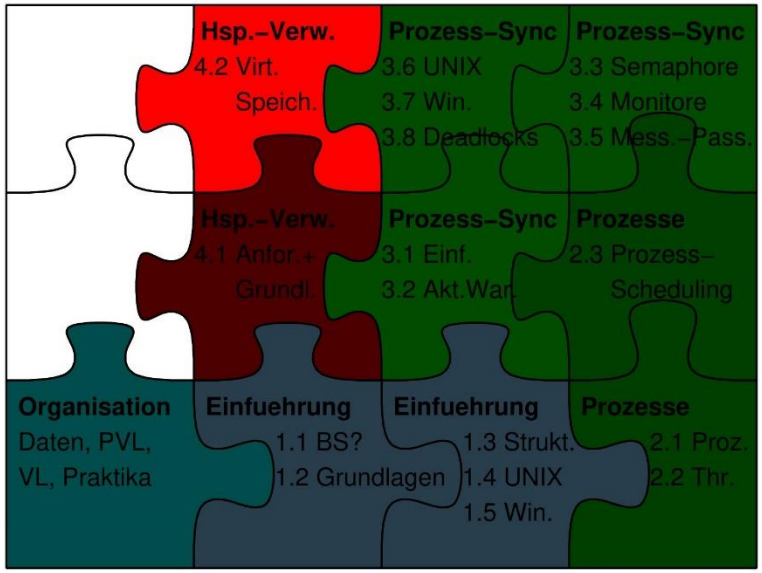
4.2.1 Einführung und Prinzipien

4.2.2 Paging

4.2.3 Pagingstrategien

4.2.4 Unix / Windows

4.2.5 Sicherheitslücken



Ende des 4. Kapitels: Welche Fragen können wir beantworten?

1. Nennen und beschreiben Sie kurz Anforderungen an Betriebssysteme!
2. Wo und von wem werden Adressen zwischen dem Programmcode und der Ausführung in der CPU angepasst?
3. Was sind virtuelle/logische und was physikalische/reale Adressen?
4. Beschreiben Sie kurz feste und dynamische Partitionierung! Was sind jeweils Vor- oder Nachteile? Was ist interne/externe Fragmentierung?
5. Beschreiben Sie kurz 3 Platzierungsverfahren! Was ist hinsichtlich Aufwand und was hinsichtlich Ergebnis das beste Verfahren?
6. Auf welche Arten konnen freie Speicherbereiche verwaltet werden?
7. Was sind “Swapping” und “dynamische Bibliotheken”?
8. Welche Eigenschaften hat idealer virtueller Speicher? Wie sieht eine Realisierung mit “Paging” aus?
9. Welche Nachteile haben einfache Seitentabellen? Beschreiben Sie drei Alternativen!
10. Was kann beim Zugriff eines Prozesses auf eine Speicheradresse alles passieren?
11. Was sind Ersetzungs- und Zuteilungsstrategien? Beschreiben Sie Bsp.!