Betriebssysteme

Hauptspeicherverwaltung

Virtueller Speicher

- Jeder Prozess erhält einen eigenen virtuellen (logischen) Adressraum
- Der virtuelle Speicher wird auf den Sekundärspeicher abgebildet
- Programm- und Datenbereiche sind nicht durch die Größe des realen Hauptspeichers begrenzt
- Ausführung von mehreren Programmen, deren Gesamtgröße die Größe des realen Hauptspeichers überschreiten, ist möglich

Virtuelle Speicherverwaltung

- Der reale/physikalische Hauptspeicher enthält in der Regel nur die augenblicklich benötigten Programmabschnitte und Daten
- Zur Zeit nicht benötigte Programmabschnitte und Daten können durch das Betriebssystem auf den Sekundärspeicher ausgelagert werden
- Erst im Mehrprogrammbetrieb effizient nutzbar.

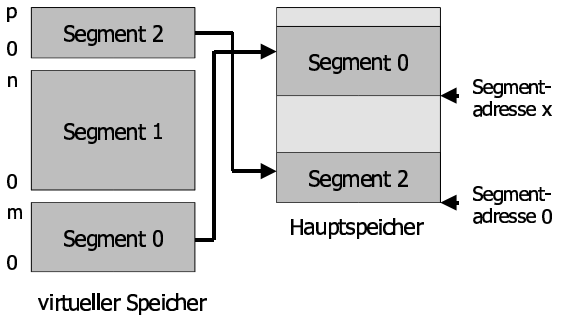
Mechanismen der virtuellen Speicherverwaltung

- Hardware-Unterstützung durch Memory-Management-Unit (MMU)
- Speicherschutz:
  - Isolierung der Adressräume
  - Überprüfung der Zugriffsarten
- Grundlegende Abbildungsmechanismen:
  - Segmentierung
  - Seitenadressierung (paging)

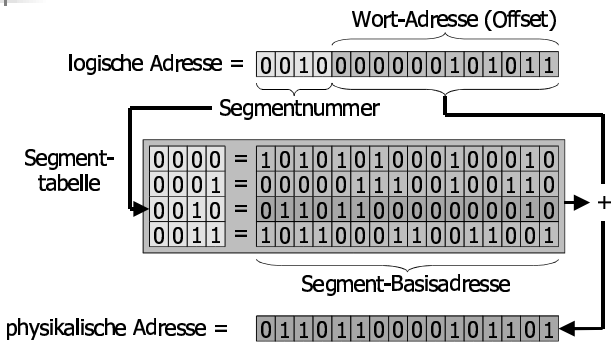
Segmentierung

- Unterteilung des logischen Adressraums in Segmente:
  - Abschnitte variabler Größe
  - Aufteilung gemäß den zusammengehörenden Einheiten des Programms (Unterprogramme, Datenbereiche usw.)
  - typische Größe: 256 Byte – 64 KB

Abbildung der Segmente auf den Hauptspeicher



Logische und reale Adressen bei Segmentierung

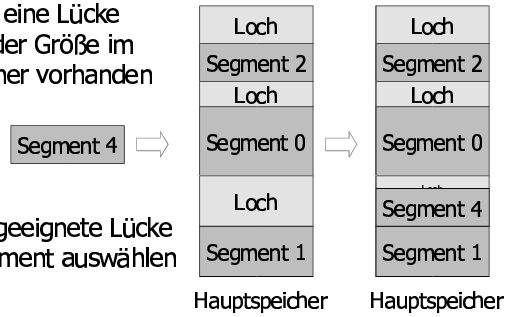


Segmenttabellen

- Tabelle mit Informationen zu allen Segmenten eines Prozesses:
  - Segmentnummer
  - Segment-Basisadresse
  - Längenangabe (für Bereichsschutz)
  - Zugriffsattribute (für Zugriffsschutz)
  - Markierung geladen/nicht geladen
  - Markierung verändert/nicht verändert (dirty tag)

Einlagern benötigter Segmente: Fall 1

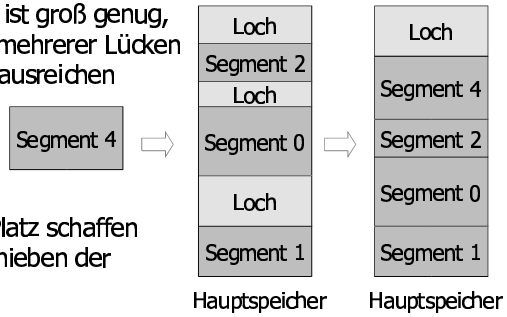
Mindestens eine Lücke ausreichender Größe im Hauptspeicher vorhanden



Vorgehen: geeignete Lücke für das Segment auswählen

Einlagern benötigter Segmente: Fall 2

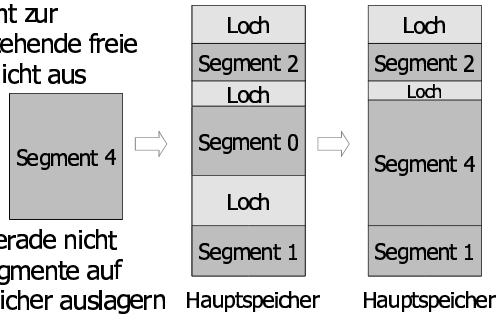
Keine Lücke ist groß genug, die Summe mehrerer Lücken würde aber ausreichen



Vorgehen: Platz schaffen durch Verschieben der Segmente

Einlagern benötigter Segmente: Fall 3

Der insgesamt zur Verfügung stehende freie Platz reicht nicht aus

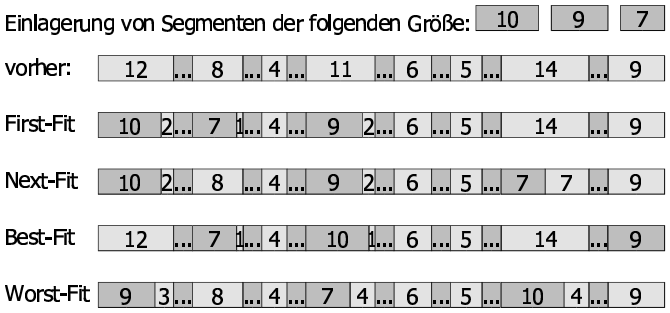


Vorgehen: gerade nicht benötigte Segmente auf Sekundärspeicher auslagern

Belegungsstrategien

- **First-Fit:** Wähle die erste, ausreichend große Lücke
  - schnell, aber wiederholtes Durchsuchen der Restlöcher
- **Next-Fit:** Beginne bei der zuletzt gefüllten Lücke und wähle die nächste, ausreichend große Lücke
  - schnell, wiederholtes Durchsuchen der Restlöcher nicht notwendig
- **Best-Fit:** Wähle die kleinste, noch ausreichend große Lücke
  - langsamer, erhält große Restlöcher, erzeugt kleine Restlöcher
- **Worst-Fit:** Wähle die größte vorhandene Lücke
  - langsamer, vermeidet kleine Restlöcher

Belegungsbeispiel



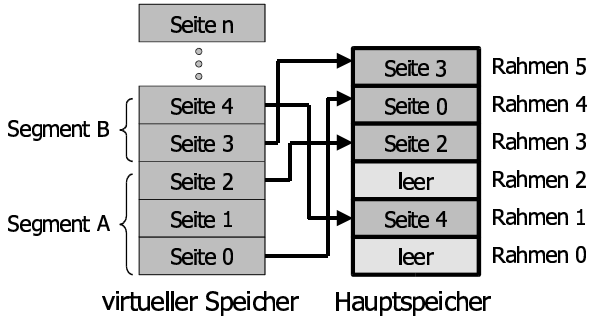
Vor- und Nachteile der Segmentierung

- Vorteile
  - Segmente entsprechen zusammengehörigen Einheiten, denen spezifische Merkmale zugeordnet werden können
  - Überlappende Segmente möglich (shared code)
  - Segmentlänge dynamisch änderbar
- Nachteile
  - Aufgrund der variablen Segmentlänge aufwendige Freispeicherverwaltung notwendig
  - Segmente müssen immer vollständig ein- oder ausgelagert werden
  - Speicherzersplitterung (Fragmentierung) möglich

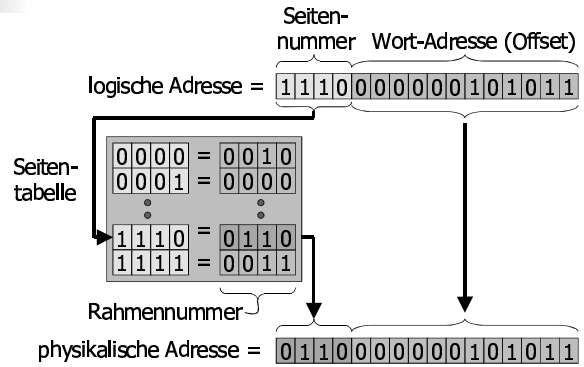
Seitenadressierung (paging)

- Unterteilung des logischen Adressraums in Seiten:
  - Abschnitte gleicher Größe (Seiten/pages)
  - Unterteilung des physikalischen Adressraums in korrespondierende Abschnitte (Rahmen/frames)
  - Aufteilung gemäß physikalischer Gesichtspunkte (leichte Adressierung, keine Fragmentierung)
  - typische Seiten-/Rahmen-Größe: 512 Byte – 8 KB

Zuordnung von Seiten zu Rahmen



Logische und reale Adressen bei Seitenadressierung



Seitentabellen

- Tabelle mit Informationen zu allen Seiten eines Prozesses:
  - Seitennummer
  - zugehörige Rahmennummer
  - Zugriffsattribute (für Zugriffsschutz)
  - Markierung geladen/nicht geladen
  - Markierung verändert/nicht verändert (dirty tag/modified bit)

Gemeinsam genutzter Speicher (shared memory)

- Dynamische, kontrollierte, gemeinsame Nutzung von Speicherbereichen
- Abbildung eines physikalischen Speicherbereichs in mehrere virtuelle Bereiche
- Notwendig: Freigabemechanismus, Löschschutz und Synchronisationsverfahren
- Beispiele:
  - Mehrfaches, gleichzeitiges Ausführen desselben Programms
  - Benutzung derselben Programm-Bibliotheken
  - Gleiche Datenbereiche (z. B. Parameter zur Fensterdarstellung)

Optimale Seitengröße

- Sei h die Hauptspeichergröße und s die Seitengröße
- Annahmen:
  - Die Prozessgröße ist gleichverteilt
  - Pro Prozess wird eine einstufige Seitentabelle mit $\lceil h / s \rceil$ Einträgen verwendet
  - Ein Eintrag in der Seitentabelle beansprucht ein Wort
- Folgerungen:
  - Der mittlere Verschnitt beträgt s / 2 Worte
  - Der mittlere Verlust V beträgt $(h / s + s / 2)$ Worte $\equiv h f_v$
  - Die optimale Seitengröße s_{opt} beträgt $\sqrt{2h}$ Worte
  - Der Verlustfaktor f_v beträgt in diesem Fall $2 / s_{opt}$

Seitenersetzungsstrategien

- **Problem**: Alle Rahmen im Hauptspeicher sind belegt, es muss aber eine neue Seite eingelagert werden → Auslagerung notwendig
- **Ziel**: Auswahl der auszulagernden Seiten, so dass (langfristig gesehen) möglichst wenig Auslagerungen notwendig werden
- Häufig verwendete Strategien:
  - First-In-First-Out-Strategien (FIFO)
  - Not-Recently-Used-Strategie (NRU)
  - Least-Recently-Used-Strategie (LRU)
  - Least-Frequently-Used-Strategie (LFU)

Die optimale Strategie

Ersetze die Seite, die am spätesten in der Zukunft benötigt wird (Belady, 1966)

Referenzfolge: 0, 1, 2, 3, 1, 4, 5, 0, 2, 6, 4, 5, 6, 3, 5, ...

RAM	0	0	0	0	0	0	0	0	2	6	6	6	6
RAM		1	1	1	1	4	4	4	4	4	4	4	4
RAM			2	3	3	3	5	5	5	5	5	5	5
DISK				2	2	2	2	2	0	0	0	0	0
DISK						1	1	1	1	1	1	1	1
DISK							3	3	3	3	3	3	3
DISK										2	2	2	2

FIFO-Strategien

Ersetze die Seite, die bereits am längsten im Hauptspeicher steht.
Abwandlung: Falls die Seite ein gesetztes Referenced-Bit besitzt, behandle sie, als wäre sie neu eingelagert worden (second chance)

Referenzfolge: 0, 1, 2, 3, 1, 4, 5, 0, 2, 6, 4, 5, 6, 3, 5, ...

RAM	0	0	0	3	3	3	3	0	0	0	4	4	4
RAM		1	1	1	1	4	4	4	2	2	2	5	5
RAM			2	2	2	2	5	5	5	6	6	6	6
DISK				0	0	0	0	3	3	3	3	3	3
DISK						1	1	1	1	1	1	1	1
DISK							2	2	4	4	0	0	0
DISK										5	5	2	2

NRU-Strategie

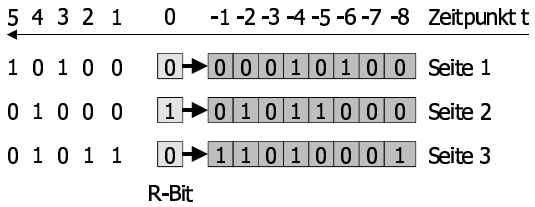
Ersetze eine Seite, die in letzter Zeit nicht benutzt wurde. Berücksichtige das Referenced- und das Modified-Bit. Ersetze in der Reihenfolge: R=0 und M=0, R=0 und M=1, R=1 und M=0, R=1 und M=1.

Referenzfolge: 0, 1, 2, 3, 1, 4, 5, 0, 2, 6, 4, 5, 6, 3, 5, ...
(keine Seite wird modifiziert, R wird nach 2 Takten zurückgesetzt)

RAM	0	0	0	3	3	3	5	5	5	6	6	6	6
RAM		1	1	1	1	1	1	0	0	0	4	4	4
RAM			2	2	2	4	4	4	2	2	2	5	5
DISK				0	0	0	0	1	1	1	1	1	1
DISK						2	2	2	4	4	0	0	0
DISK							3	3	3	3	3	3	3
DISK										5	5	2	2

LRU-Strategie

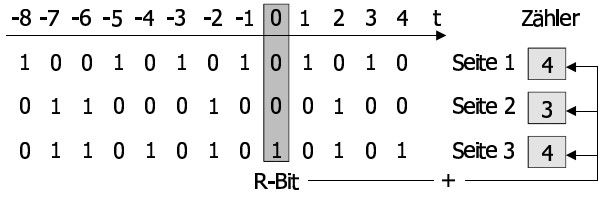
Ersetze die Seite, die am längsten nicht mehr benutzt wurde.
Realisierung: - Zeitstempel (ersetze älteste Seite mit R=0)
- Schieberegister (ersetze Seite mit kleinsten Wert)



Realisierung der LRU-Strategie mit Schieberegistern

LFU-Strategie

Ersetze die Seite, die bisher am seltensten benutzt wurde
Realisierung: Zähler, der bei gesetztem R-Bit erhöht wird
Varianten: Einbau von Alterungsmechanismen, z. B. periodisches Rücksetzen des Zählers oder Verschieben des Werts nach rechts



Wichtige Begriffe

- Arbeitsmenge (working set):
  Menge der Seiten, die im Hauptspeicher vorhanden sein müssen, damit ein Prozess (innerhalb eines bestimmten Zeitfensters) effizient arbeiten kann
- Seitenfehler (page fault):
  Zugriff auf eine nicht im Hauptspeicher befindliche Seite. Ein Seitenfehler führt zur Unterbrechung des Prozesses und der Aktivierung des Betriebssystems
- Seitenflattern (thrashing): Das Betriebssystem muss aufgrund einer Überlastung rechenwilliger Prozesse ständig Seiten ein- und auslagern. Die Ausführung der Prozesse kommt praktisch zum Stillstand.

Vor- und Nachteile der Seitenersetzung

- Vorteile
  - Nur die aktuell benötigten Teile des Prozesses müssen im Hauptspeicher vorhanden sein
  - einfache Freispeicherverwaltung
  - keine Fragmentierung
  - Überlappende Segmente möglich (shared code)
- Nachteile
  - Spezifische Merkmale (von Segmenten) müssen bei allen korrespondierenden Seiten eingetragen werden
  - Seitentabellen sind wesentlich größer als Segmenttabellen
  - Platzverlust durch nicht vollständig belegte Seiten (Verschnitt)