**Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungOrganisation des Hauptspeichers**

Bereiche fester Länge

Konzept

* Speicher wird in Bereiche fester Länge geteilt
* Können unterschiedlich lang sein

Contra

* Jedes Programm muss so geschrieben sein, dass es in einen Bereich passt
* nicht genutzte Speicher kann aufgrund der eingesetzten Verwaltungsstrukturen nicht für andere Speicheranforderungen genutzt werden

Alternative

* Speicher wird in (relativ) kleine Bereiche fester Länge aufgeteilt
* Programm belegt mehrere (aufeinanderfolgende) Bereiche
* Verlust durch interne Fragmentierung: durchschnittlich ein halber Bereich pro Programm

Bereiche variabler Länge

* Betriebssystem stellt Speicherbereich in der gewünschten Länge bereit

Contra

* Externe Fragmentierung (Insgesamt ausreichend freier Speicher vorhanden, jedoch nicht zusammenhängend) Ein Bild, das Text enthält.

  Automatisch generierte Beschreibung

Überlagerungsstruktur

* Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungWenn Programm größer als Hauptspeicher

Hauptspeicherzuweisung

Verwaltung der statischen Bereiche:

* Für jeden Bereich wird ein Bit benötigt, das angibt, ob er frei oder belegt ist ( ⇒ Core Map)
* Falls sich ein Programm über mehrere Bereiche erstreckt, muss man die Zuordnung verwalten

Einfache Speicherplatzzuteilung

* Programm wird einem statischen Bereich zugeordnet
* Voraussetzung: Programmlänge muss bekannt sein
* Strategie: Anwender gibt an welcher Bereich oder Auftragssteuerung entscheidet

Variabler aber komplexer

* Programm erstreckt sich über mehrere Bereiche fester Länge
* Voraussetzung: Programmlänge muss bekannt sein
* Strategie: Auftragsverwaltung muss eine entsprechende Menge zusammenhängender Bereiche finden, ggf. durch Verschieben von Programmen

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungVerwaltung der variablen Bereiche

* Verkettung der unbelegten Bereiche in einer Freiliste
* Bei der Rückgabe von Bereichen werden aneinandergrenzende Bereiche   
  verschmolzen

**Zuteilungsstrategien**

First-Fit  
Verwendet den ersten freien Bereich, der groß genug ist. Der nicht benötigte Rest des Bereichs wird wieder in die Freiliste eingekettet

Best-Fit  
Finde kleinsten Bereich, der gerade noch passt. Der nicht benötigte Rest des Bereichs wird in die Freiliste eingekettet

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Virtueller Speicher**

Modell des virtuellen Speichers

* Der Hauptspeicher wird durch einen Teil des Hintergrundspeichers ergänzt
* Folge: Das Programm muss nicht vollständig im Hauptspeicher liegen
* Programmadressen sind prozessindividuelle virtuelle Adressen, die in reale Hauptspeicheradressen oder ggf. solche des Hintergrundspeichers umgerechnet werden müssen

Adressumsetzung

* Adressumsetzung muss Hardware-unterstützt durchgeführt werden
* andernfalls wird die Gesamtsystemeffizienz stark beeinträchtigt
* Nachladbare Programmteile: Seiten (Pages; Bereiche fester Länge)

Seitentausch (Paging)

* Einteilung des Hauptspeichers in Kacheln (Page Frames) fester Größe

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungWie finde ich ein Speicherwort ?

* Interpretation einer vom Übersetzer erzeugten virtuellen Adresse
* Anzahl der Seiten: 2k - Seitengröße: 2m Worte
* Ziel bei der Adressumsetzung: Aus der Seitennummer P die Kachelnummer K bestimmen
* Vorgehensweise
  + Verwenden einer Seitentabelle
  + Konkatenieren der Kachelnummer K mit der Wortadresse W.
  + Hauptspeicheradresse = K,W

Aufbau eines Seitentabelleneintrages

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Schutzaspekt: Adressraumtrennung

* Jeder Prozess besitzt seine eigene Seitentabelle.
* Folge: Prozesse adressieren mit identischen virtuellen Adressen unterschiedliche physikalische Speicherbereiche

Effizienz

* Problem: Aus jedem Hauptspeicherzugriff werden zwei!
* Lösung: Hardware-Unterstützung Verwendung eines Adress-Cache als Teil der Memory Management Unit (MMU)

**Speicherverwaltung virtueller Systeme**

Speicherplatzzuteilung

Seitentausch

* Verwendung von Einheiten fester Länge
* Seitengröße klein ⇒ geringe interne Fragmentierung
* Seitengröße groß ⇒ effiziente Plattenzugriffe; kleinere Seitentabellen
* Seitengröße identisch zur Größe eines physikalischen Plattenblocks   
  ⇒ einfach beim Ein-/Auslagern

Verwaltungsstrategien für den Heap: First-Fit-, Best-Fit-Verfahren oder Buddy-System

Einlagerungsstrategien

Vorgeplanter Seitentausch

* Seiten im Voraus in den Hauptspeicher bringen, um Seitenfehler zu vermeiden
* Viele Nachteile
  + Prozessorzeit notwendig, um die Auswahl zu treffen
  + Zu teuer, wenn Seite nicht benötigt wird
  + Trefferrate muss extrem hoch sein

Seitentausch auf Verlangen (Demand Paging): wird fast immer verwendet

Auslagerungsstrategien

* Random Lagere eine zufällig ausgewählte Seite aus.

Implementierung: Ohne Datenstrukturen

* First In First Out (FIFO) Lagere die Seite aus, die sich schon am längsten im  
   Hauptspeicher befindet

Implementierung: über zyklische Kachelanzeiger

* Not Frequently Used (NFU) Lagere die Seite aus, die am seltensten benutzt wurde

Implementierung: Zähler für Zugriffe; niedrigster Zähler raus

* Least Recently Used (LRU) Lagere die Seite aus, auf die am längsten nicht mehr  
   zugegriffen wurde.

Implementierung: mittels Hardware-mäßig zu setzenden Referenz-Bits

* Optimal Replacement Lagere die Seite aus, für die der nächste Zugriff am  
   Weitesten in der Zukunft liegt

Implementierung: Nicht implementierbar

* Second Chance Jede Seite erhält eine zweite Chance, veränderte Seiten

sogar eine dritte!

Implementierung: Clock-Algorithmus: Neben Referenzbit und Change-Bit (beide  
 CPU) wird ein Auslagerbit (in Software) verwendet. Über  
 einen zyklischen Kachelzeiger werden nacheinander alle  
 Kacheln betrachtet.

Prozesse und virtueller Speicher