受講にあたっての注意事項

本講義の全部あるいは一部の録画・録音および複製ならびに再配布を、厳に禁じます。

大阪大学 基礎工学部
SCHOOL OF ENGINEERING SCIENCE

オペレーティングシステム

3章 メモリ管理

3.2節 仮想メモリーメインメモリの隠ぺいー

3.2.3 マッピング

大阪大学大学院情報科学研究科
村田正幸
murata@ist.osaka-u.ac.jp
http://www.anarg.jp/

3.2 仮想メモリーメインメモリの隠ぺい一
3.2.3 マッピング

[1] アドレス変換テーブル

- 仮想アドレス*V*から実アドレス*R*へのアドレス変換
  - 仮想アドレス*V*によってアドレス変換テーブルを引き、その内容(エントリ、事項)である実アドレス*R*を得る
  - アドレス変換テーブルを引く操作が当該仮想アドレスが実メモリ上にあるかないかのチェック操作も兼ねる
- メインメモリ上に構成
  - マッピングの変更ごとに動的な書き換えが必要となる
  - サイズが大きい
- 以下の操作は高速処理が要件となるので、通常は、動的アドレス変換機構 (DAT) によって実現。すなわち、
  - 「当該仮想アドレスが実メモリ上にあるかないかのチェック」操作
  - 「アドレス変換テーブルを引く」操作

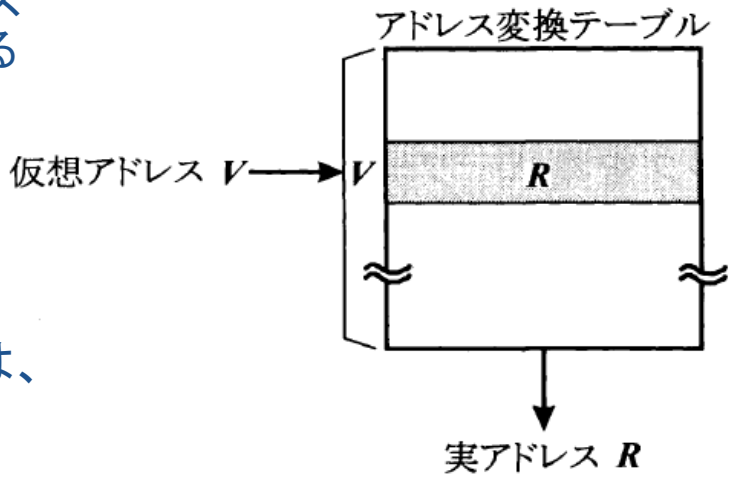


図 3.12 アドレス変換

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[2] マッピングの分類

- 仮想メモリ⇔実メモリ(メインメモリ)間のマッピング
  - 仮想メモリの一部(コピー)を実メモリ(メインメモリ)領域へ割り付ける機能
- 単位
  - 仮想メモリのうちの参照局所性の高い部分の実メモリ(メインメモリ)領域への割り付け単位
  - 実メモリ(メインメモリ)⇔バックアップメモリ(ファイル装置)間のデータ転送単位
- 分類

(A) ページング(paging)

- ページという一定サイズに固定したブロック(単位)でマッピングする
- プログラム(データ、プロセス)を、それらがもつ「論理的な意味」は無視して、機械的また物理的に、固定長(一定)サイズのページ単位に区切る
- 固定長領域割り付けの例

(B) セグメンテーション(segmentation)

- 1 個のプログラムやプロセスあるいは一連(一群)のデータといった「論理的な意味」を持つブロック(セグメントという)でマッピング
- セグメントは、プログラム(プロセス)やデータブロックという「論理的な意味」を考慮した論理的な単位で構成するので、そのサイズは可変
- 可変長領域割り付けの例

(C) ページセグメンテーション(paged segmentation)

- (A)と(B)を融合したマッピング

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[3] ページング方式

- 仮想アドレス空間も実アドレス空間も一定かつ固定長のページサイズで分割
- ページ単位で、仮想アドレス空間上のページ（仮想ページまたは論理ページ）と実アドレス空間上のページ（実ページまたは物理ページ）とをマッピングする
- 仮想アドレス空間には仮想ページごとに仮想ページ番号を、実アドレス空間には実ページごとに実ページ番号を、それぞれ振る
- 仮想ページと実ページのマッピングはページテーブルと呼ぶアドレス変換テーブルに記述しておく
  - 仮想ページから実ページへのアドレス変換はページテーブルを引く操作となる

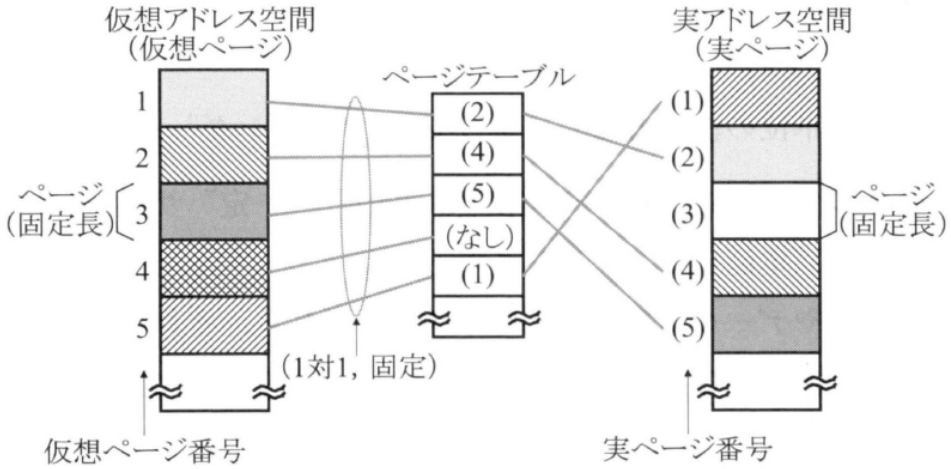


図 3.14 ページングによるマッピング（例）

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[3] ページング方式

- 長所
  1. マッピング単位が一定かつ固定長のページ

     →メインメモリ（実メモリ）やファイル装置（バックアップメモリ）の管理が簡単

     →メインメモリでの外部フラグメンテーションの発生はほとんどなくなり、メインメモリの使用効率は常時良好
  2. メインメモリ（実メモリ）サイズより大きなプロセス（プログラムやデータ）も、実メモリ上にマッピングできる

     ←複数ページに分割して一部の必要なあるいは使用する仮想ページだけを動的に入れ替え

3.2 仮想メモリーメインメモリの隠ぺい―
3.2.3 マッピング

[3] ページング方式

- 短所

3. プロセス(プログラムやデータ)がもつ「論理的な意味」を無視して、機械的また物理的にページに分割して、かつ、それらを不連続で、独立にマッピングする

→論理的な意味や論理的単位そのものが備えている参照局所性を活用することが困難になる

→元の論理的な意味や論理的単位で備えていた参照局所性も分割されてしまうので、その参照局所性を適用あるいは利用する方法は格段に複雑になる

←実アドレス空間上でのプロセスの属性管理やリンク(関係付け、連係)などの論理的な意味や論理的単位を無視して、それらを機械的また物理的に分割するため

4. ページサイズが固定であり、また、ページ単位でマッピングする

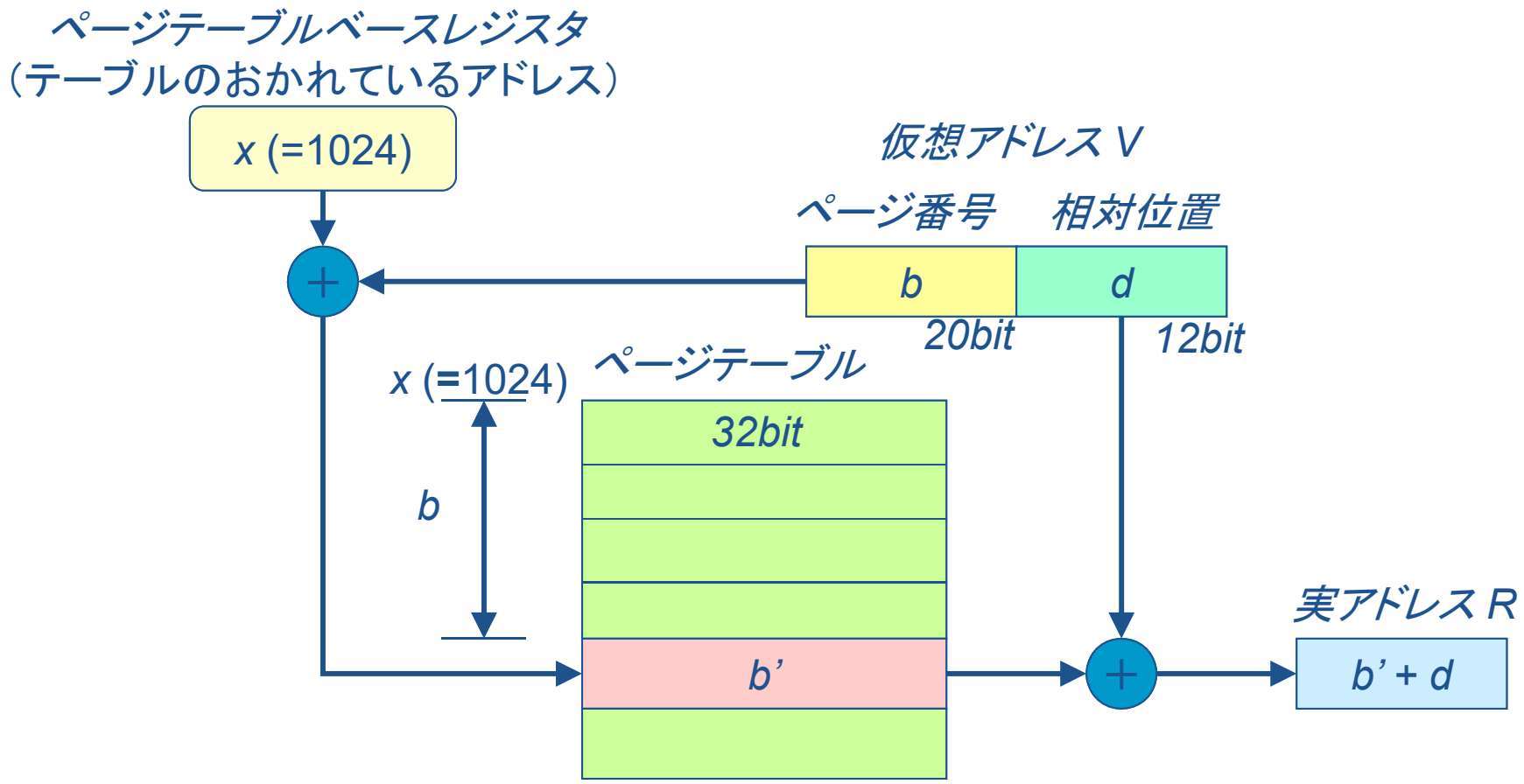
→内部フラグメンテーションが発生しやすい

→特に、ページサイズよりも格段に小さいプロセスをマッピングした場合、そのページ内に残っている大きな未使用領域を他のプロセスにマッピングできないという無駄が生じる

ページテーブルの構成

- メインメモリをページ枠 (Page Frame) に分割
  - ページ枠への割り当てはメインメモリ上のページテーブルで管理
  - ページ(ページ枠)の大きさ
    - ページが小さい→テーブルが大きくなって、検索時間が増大
    - ページが大きい→無駄な領域(内部フラグメンテーション)が大きくなる
  - ページサイズは4KBや8KBにするのが一般的
- ページテーブルの構成
  - 32 bit 仮想アドレス⇒20 bit ページ番号, 12 bitオフセット⇒1エントリ 32 bit (実ページ番号＋属性)
  - 2^{20} x 4 x n = 4 x *n* MB (*n*: 仮想アドレス空間の数≒同時に存在するプロセス数)

動的アドレス変換（DAT）

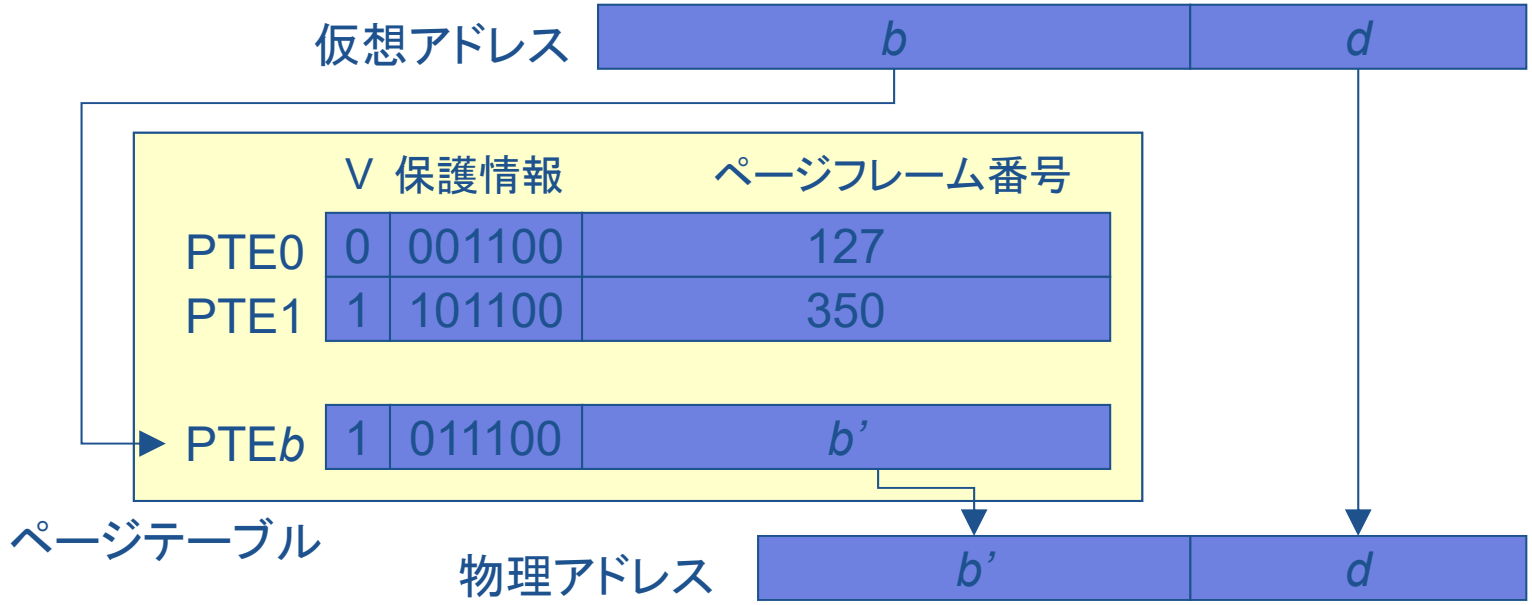


実アドレス＝アドレス変換テーブル[ページ番号]＋相対位置

UNIXにおけるアドレス変換機構

– 仮想アドレス(上位 *b*)へのアクセスがあった時、以下を実行する

1. テキストセグメント用ページテーブルの該当エントリ(PTE)を参照する
2. PTEのVビット(ページが有効か)を参照する
   - 0の時:実メモリに割り付けられていない
     – ページフォールト割り込みによって、ページ内容を読み出す
3. 保護ビットを参照して、アクセスの可否判断
   - 許可されていれば、ページフレーム番号 *b'* に置き換える



3.2 仮想メモリーメインメモリの隠ぺい―
3.2.3 マッピング

[4] セグメンテーション方式

- セグメント単位で、仮想アドレス空間上のセグメント（仮想セグメントまたは論理セグメント）と実アドレス空間上のセグメント（実セグメントまたは物理セグメント）とをマッピング
- 仮想アドレス空間には仮想セグメントごとに仮想セグメント番号を、実アドレス空間には実セグメントごとに実セグメント番号をそれぞれ振る
- 仮想セグメントと実セグメントのマッピングはセグメントテーブルと呼ぶアドレス変換テーブルに記述
  - 仮想セグメントから実セグメントへのアドレス変換はセグメントテーブルを引く操作となる

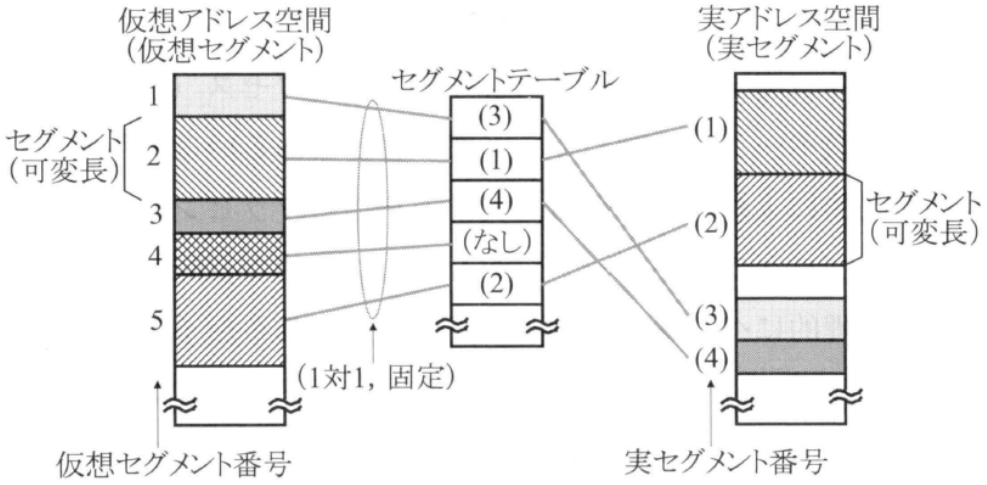


図 3.15 セグメンテーションによるマッピング（例）

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[4] セグメンテーション方式

- 長所

(1) セグメントは「論理的な意味」をもつ単位であるので、セグメントテーブルに記述しておく「セグメントの属性」をアクセス制御時に使用できる。

- セグメントの属性の例

(a)アクセス権
(b)命令かデータかの種別
(c)データ型;
(d)他のセグメントとの関係
(e)共用か非共用かの区別

(2) セグメント単位でのマッピングとなるので、メインメモリ内に領域を確保できれば、内部フラグメンテーションは発生しない。もちろん、外部フラグメンテーションは発生する。

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[4] セグメンテーション方式

- 短所

(3)セグメントサイズは可変かつ自由

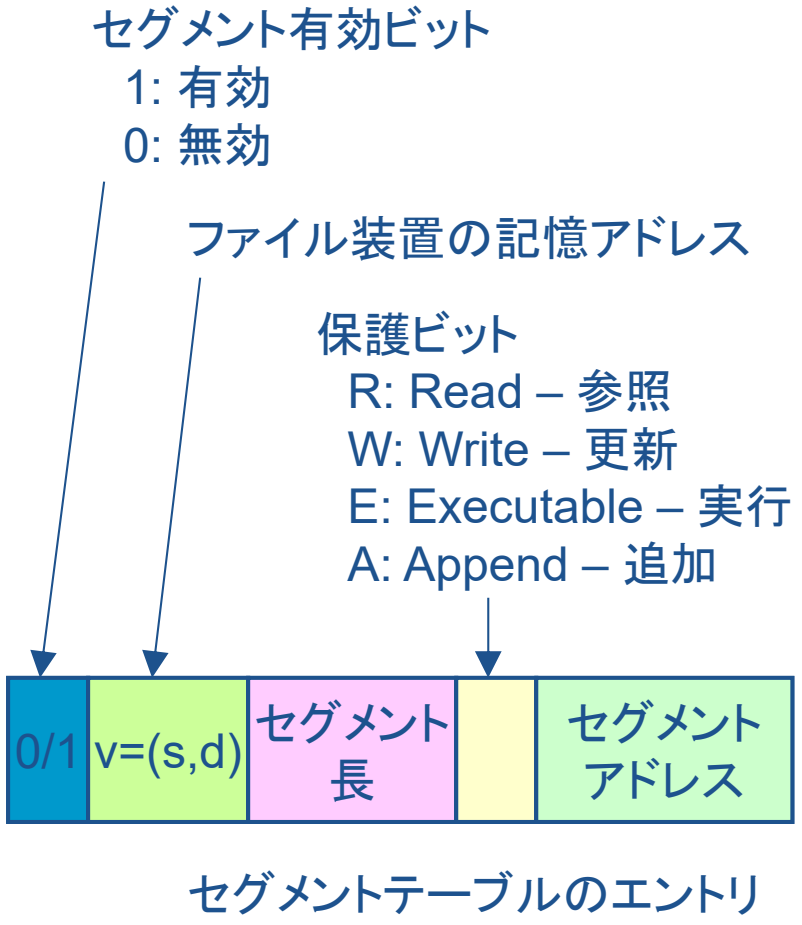
- メインメモリ（実メモリ）やファイル装置（バックアップメモリ）での領域管理、および、実メモリと仮想メモリとのマッピングの管理は非常に複雑
- メインメモリ（実メモリ）での外部フラグメンテーションが発生しやすい

→それが顕著になると、メインメモリの利用効率は格段に悪くなる

(4)メインメモリサイズより大きな仮想アドレス空間をセグメントとしてマツピングできないので、その場合に長所の(1)(2)を喪失してしまう

セグメンテーション方式における アクセス制御の例

- プログラムは本来、論理的な単位（セグメント）の集合
  - 手続き、関数、配列
- セグメンテーション方式
  - プログラムのアドレス空間をセグメントに分割する
  - セグメント内の要素
    - セグメント先頭からのオフセットで識別
    - 論理アドレス＝＜セグメント、オフセット＞
  - 詳細なアクセス制御が可能
    - アクセス制御：セグメントの参照、更新、実行、追加など
    - 1つのセグメントは同じ保護モードでよい
    - アクセス制御の記述データ量が少しですむ



第10回（5月14日2時限）ミニレポート課題

仮想記憶方式に関する以下の説明文において、( )内に適切な語を埋めよ。

ページング方式では、アドレス空間を(　a　)長のページサイズで分割し、ページ単位で仮想アドレス空間上のページを(　b　)空間上のページにマッピングする。仮想ページと実ページのマッピングは(　c　)を用いて行う。ページング方式の長所として、実メモリやファイル装置の管理が簡単になることの他、実メモリでの(　d　)フラグメンテーションの発生がほとんどなくなることなどが挙げられる。一方、短所として、(　e　)フラグメンテーションが発生しやすいことが挙げられる。

一方、セグメンテーション方式では、セグメント単位で仮想アドレス空間上のセグメントを(　b　)空間上のセグメントにマッピングし、仮想セグメントと実セグメントのマッピングは(　f　)を用いて行う。セグメントは論理的な意味を持つので、セグメント属性を用いて実行可能か書き換え可能かなどを制御する(　g　)に利用できる。また、セグメント単位でのマッピングとなるので、実メモリ内に領域を確保できれば(　h　)フラグメンテーションは発生しない。

締切：5月19日（水）
CLEで提出

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

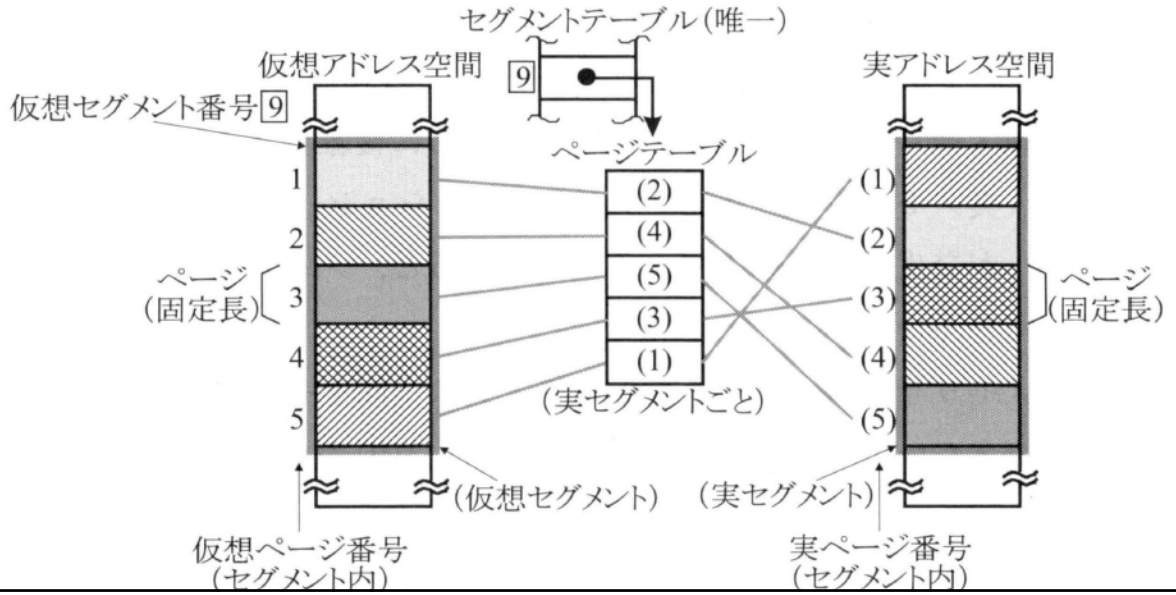
[5] ページセグメンテーション

- 2種類のマッピングを行う
  1. 全体では、セグメント単位（セグメンテーション）
  2. 各セグメント内では、ページ単位（ページング）
- 仮想アドレス空間も実アドレス空間も一定かつ固定長のページサイズで分割する
  1. セグメント単位で、仮想セグメントと実セグメントとをマッピングする

     セグメントはひとまとまりであるので、その全体サイズは可変かつ自由

     仮想セグメントと実セグメントのマッピングはセグメントテープルに記述してある
  2. ページ単位で、仮想セグメントと実セグメントのそれぞれのセグメント内のページどうし（仮想ページと実ページ）をマッピングする

     仮想ページと実ページのマッピングは各実セグメントごとに備えるページテーブルに記述してある



3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[5] ページセグメンテーション（続き）

- 仮想セグメントをページ単位で分割して、それらの各ページを実セグメント内のページに独立して不連続にページングでマッピングするセグメンテーション
- ページセグメンテーションにおけるアドレス変換の方法
  1. セグメントテーブルを引くことによって、仮想セグメント番号から実セグメントごとに備えるページテーブルのアドレスを得る
  2. 1で得たページテーブルを引くことによって、仮想ページ番号から実ページ番号を得る
- OSによるメモリ管理の観点から見たページセグメンテーションの特徴

  長所

  (1)ページングとセグメンテーシヨンの両方式の長所を融合してもつ

  最初のマッピングは論理的なまとまりのセグメント単位で行うので、そのセグメントの「論理的意味」を活用できる

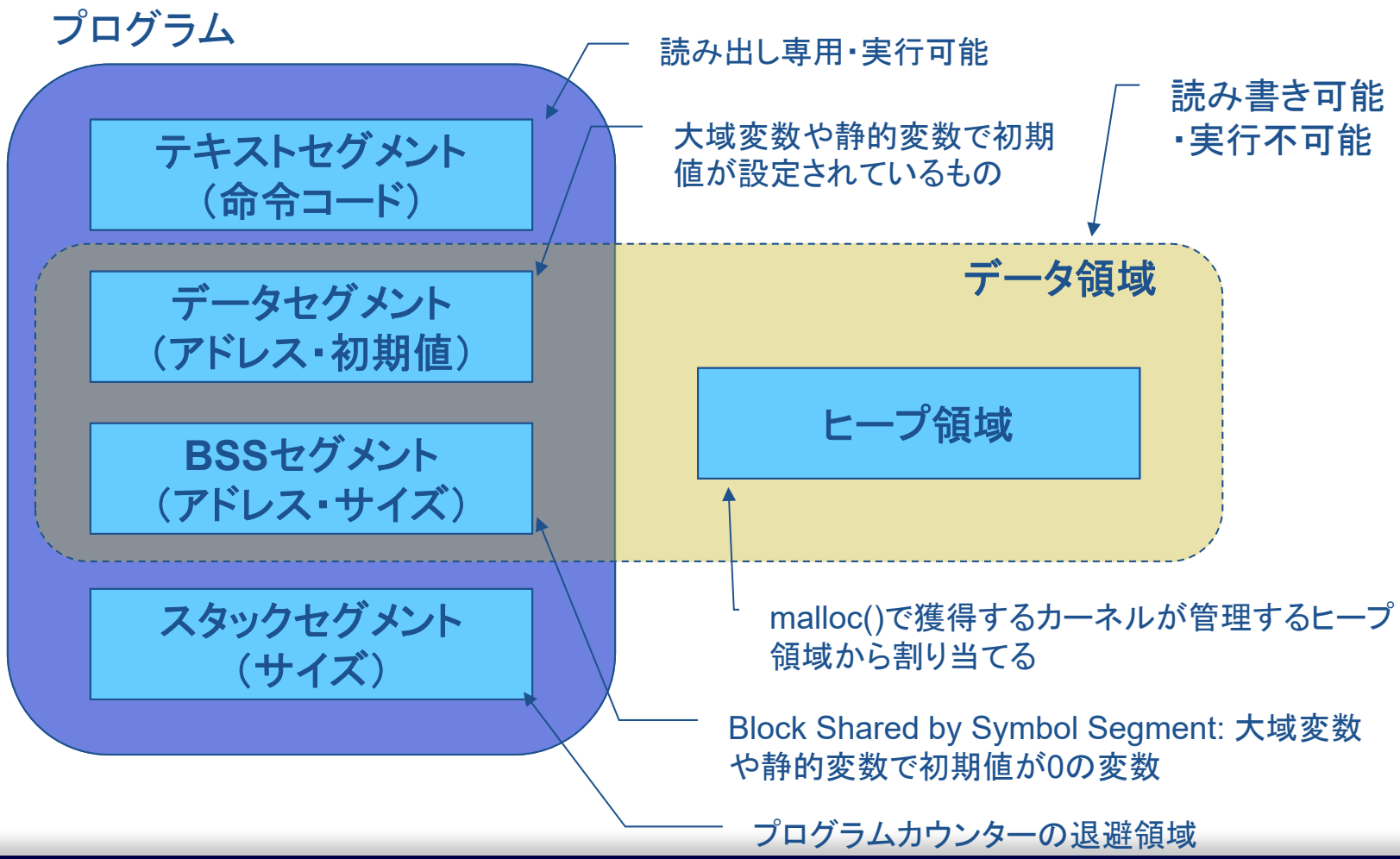
  セグメントのマッピングの決定後は通常のページングによるので、実メモリの管理が簡単で、利用効率も優れている

  短所

  (2) 1個のセグメントテーブルと実セグメント総数分のページテーブル（どれもすべてが可変長）の2種類のアドレス変換テーブルが必要となる

  テーブル全体が占める領域（空間）、および、2段階のアドレス変換すなわちテーブル検索にかかる時間、のそれぞれが大きくなる

C言語のセグメント


プログラム
テキストセグメント
（命令コード）
データセグメント
（アドレス・初期値）
BSSセグメント
（アドレス・サイズ）
スタックセグメント
（サイズ）
読み出し専用・実行可能
大域変数や静的変数で初期
値が設定されているもの
読み書き可能
・実行不可能
データ領域
ヒープ領域
malloc()で獲得するカーネルが管理するヒープ
領域から割り当てる
Block Shared by Symbol Segment: 大域変数
や静的変数で初期値が0の変数
プログラムカウンターの退避領域

3.2 仮想メモリーメインメモリの隠ぺいー
3.2.3 マッピング

[6] [まとめ】仮想メモリへのプロセス割り付けの実際

- ユーザプロセス領域ごとに、それを1個の独立した仮想アドレス空間とする
  - ユーザプロセス領域を構成するコード、データ、ヒープ、スタックフレームの各領域はそれぞれ独立したセグメントとする
  - 実メモリ(メインメモリ)へマッピングする、すなわち割り付けるユーザプロセス領域(セグメントごと)のために、バックアップメモリ(ファイル装置)上に、スワップ領域という一定サイズの領域をあらかじめ確保しておく
  - OS自身は、仮想メモリ(ブロック置換)の対象外とし、実メモリ(メインメモリ)に一定サイズの領域をOS自身で直接確保し、その領域に常駐する
- プロセッサが実行するプログラム(命令やデータ)は、実行時に、OSがプロセスとしてメインメモリに割り付ける
  - プロセス割り付けにおいて、仮想メモリ機構がブロック置換によって、必要とするプロセスをファイル装置から読み出してメインメモリに置く(割り付ける)
  - 不要と判定できる(参照局所性が低い)プロセスは、仮想メモリ機構がブロック置換によってメインメモリからファイル装置へ追い出す

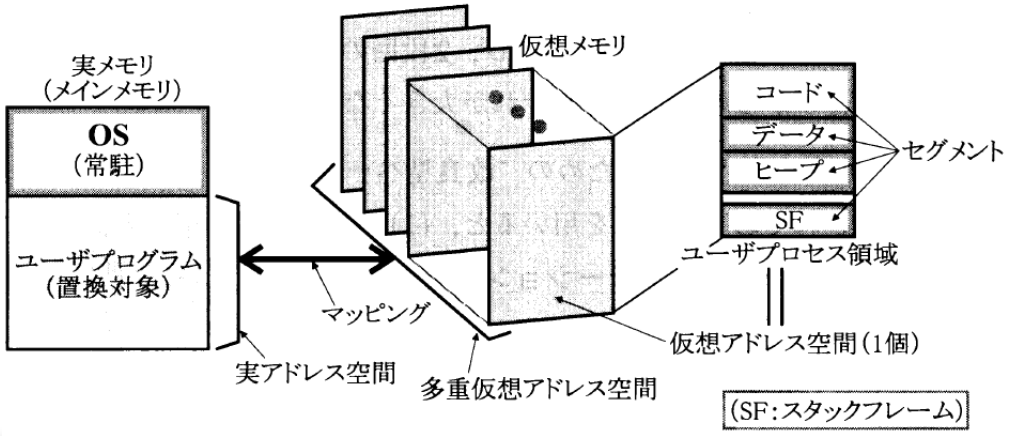


図 3.17 仮想メモリへのプロセス割り付け