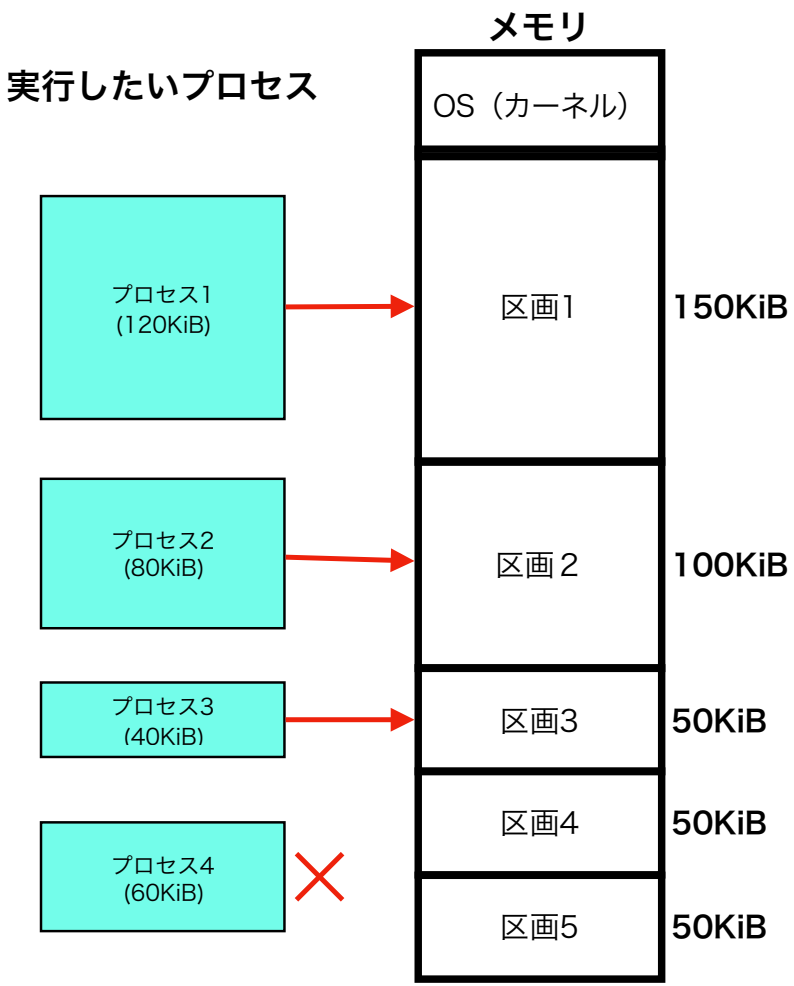
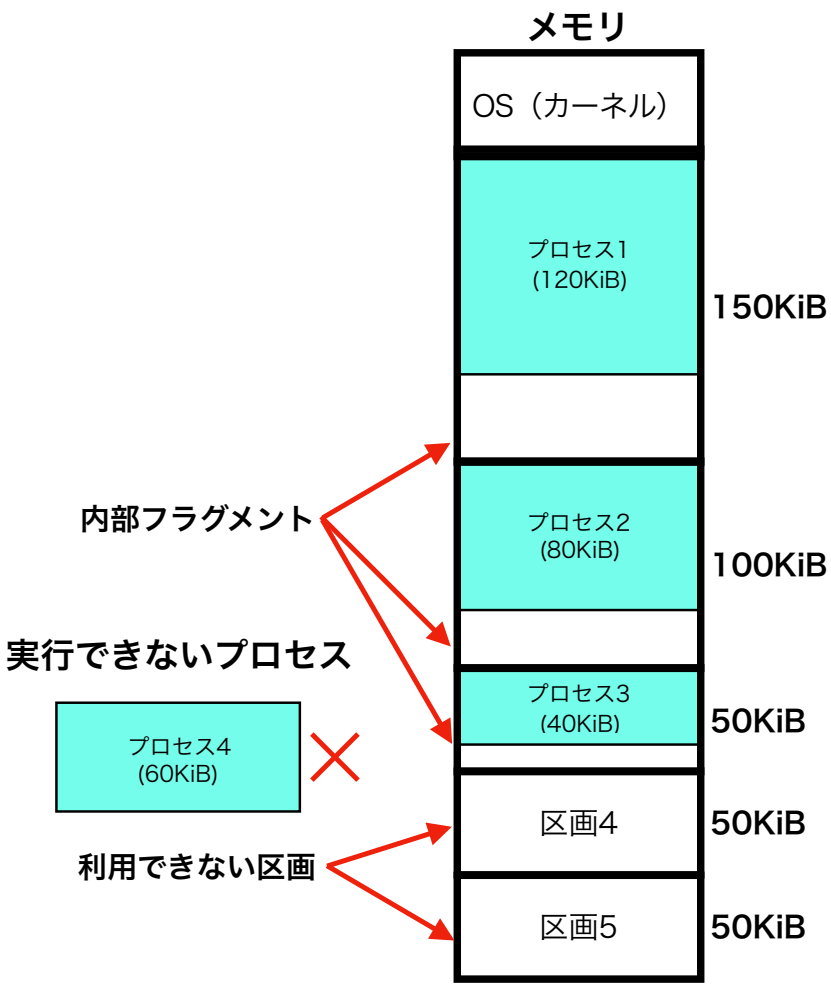
オペレーティングシステム
第９章 メモリ割付け方式

https://github.com/tctsigemura/OSTextBook

固定区画方式



(a) 区画を選択しプロセスをロード



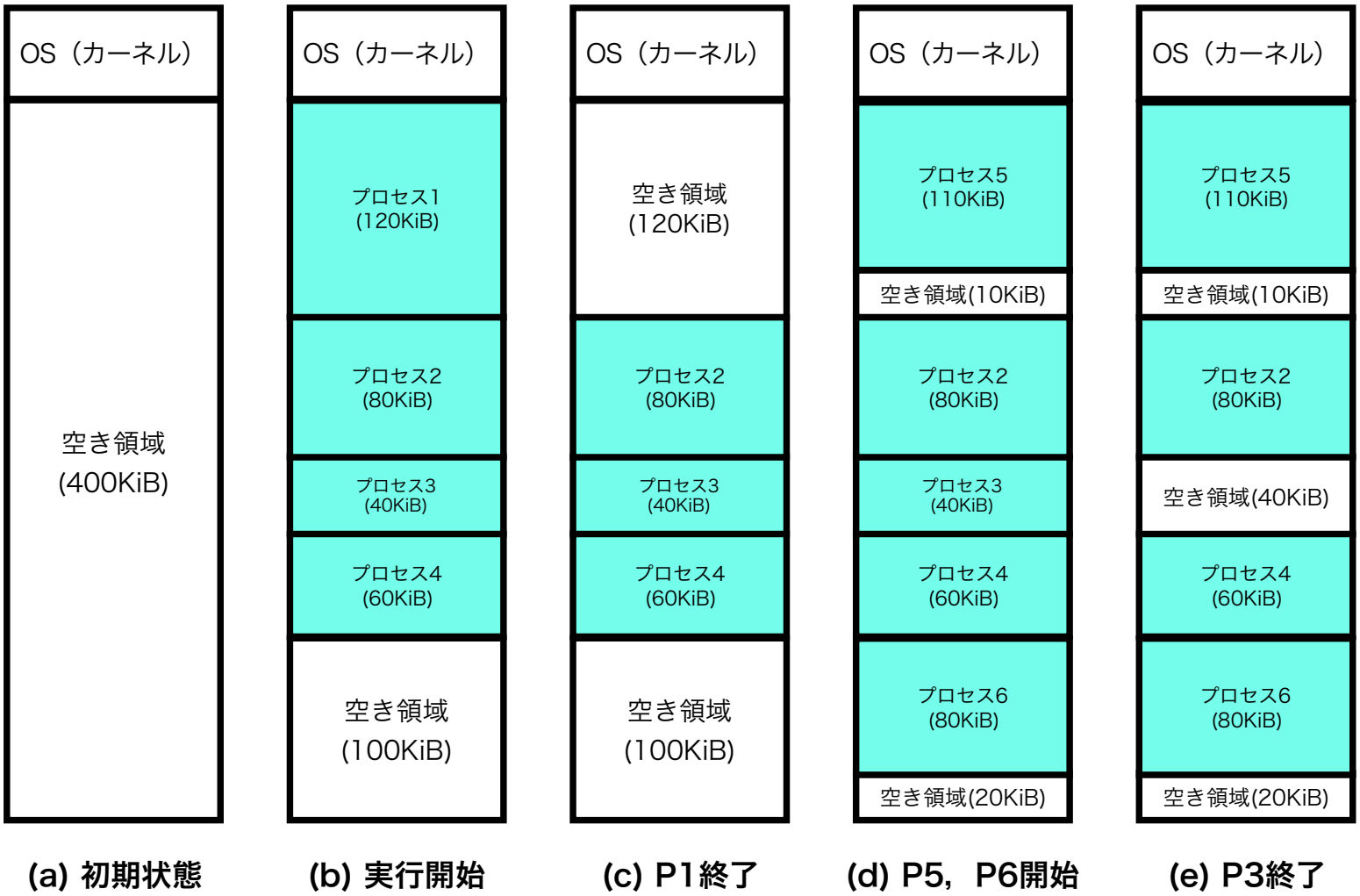
(b) プロセスを実行

予めメモリを大小数種類の区画に分割しておく.

固定区画方式の特徴

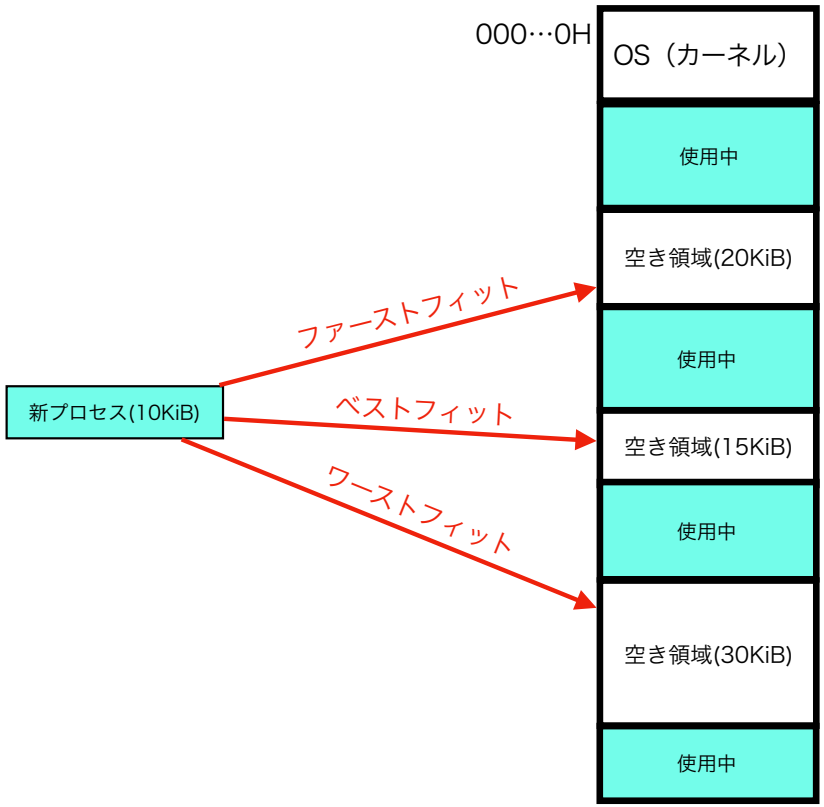
1. 空き領域の管理が容易である.
2. 領域内部に無駄な領域（**内部フラグメント**）が生じる.
3. 小さな領域が複数空いていても大きなプロセスは実行できない.
4. 実行可能なプロセスのサイズに強い制約がある.
   （図の例では，151KiB のプロセスは実行できない.）
5. 同時に実行できるプロセスの数に制約がある.
   （図の例では，同時に五つ以上のプロセスは実行できない.）

可変区画方式



必要に応じて空き領域から区画を作る．
外部フラグメントが生じる．

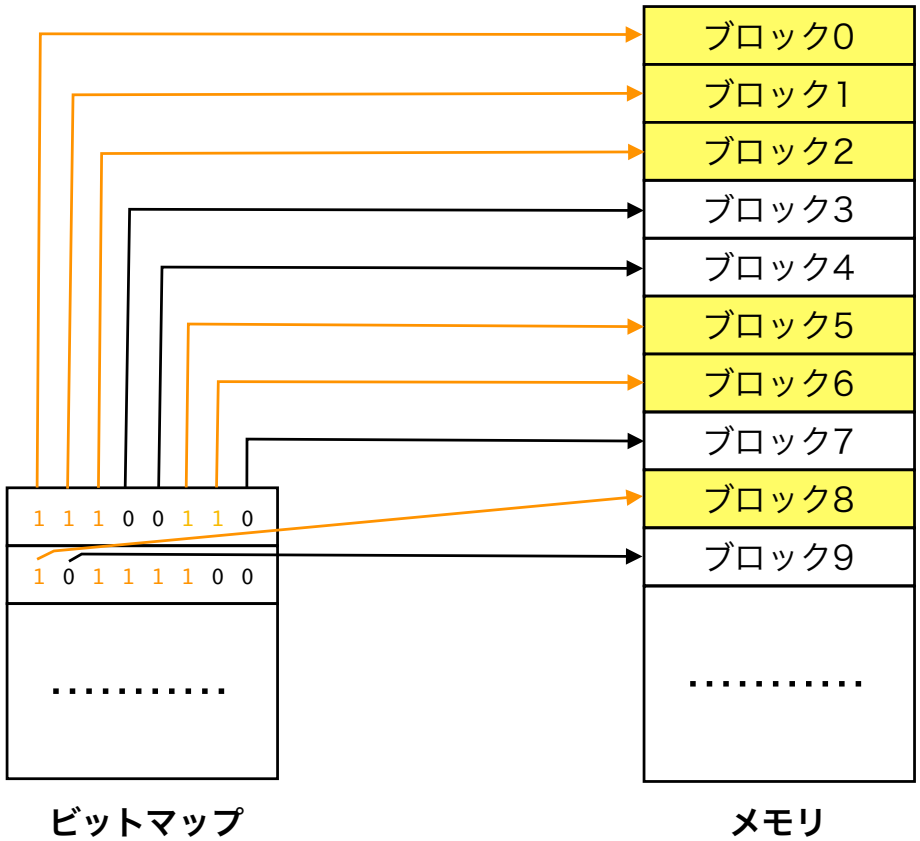
可変区画方式の領域選択方式



- ファーストフィット（first-fit）方式：アドレス順にさがす.
- ベストフィット（best-fit）方式：最小の領域を選択する.
- ワーストフィット（worst-fit）方式：最大の領域を選択する.

空き領域の管理方式（ビットマップ方式）

どこに利用可能な空き領域があるかビットマップで管理する．



- メモリを一定の大きさのブロックに分割する．
- ビットマップの 1 ビットが 1 ブロックに対応する．

ビットマップの大きさ

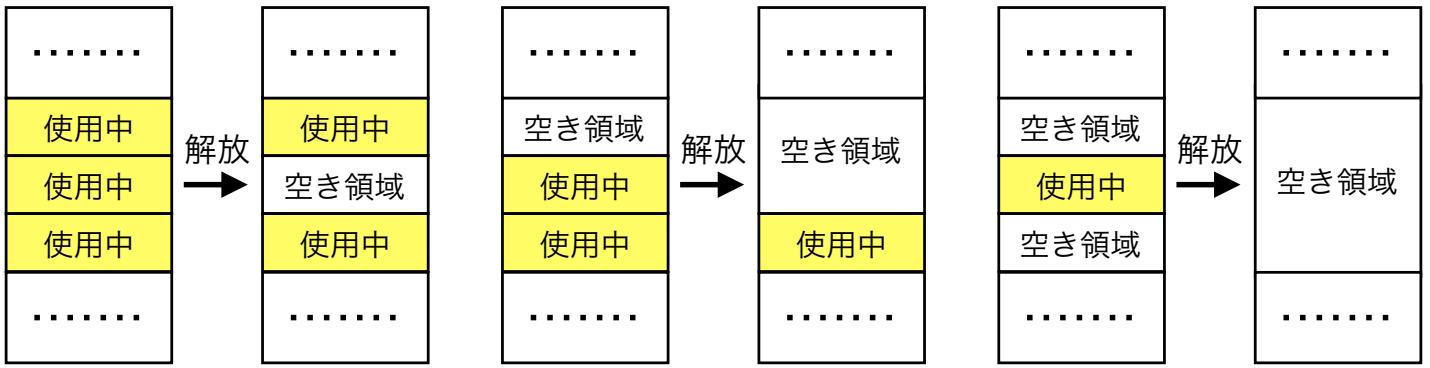
ビットマップの大きさを計算してみる.

- メモリサイズ：8GiB
- ブロックサイズ：4KiB
- ブロック数：$8GiB \div 4KiB = (8 \times 2^{30}) \div (4 \times 2^{10}) = 2 \times 2^{20} = 2Mi$
- ビットマップのサイズ：$(2 \times 2^{20}) \div 8 = 2^{18} = 256KiB$

無視できるほど小さくはない.
ビットマップを小さくするにはブロックサイズを大きくすればよい.
内部フラグメントが大きくなる.

空き領域の管理方式（リスト方式）

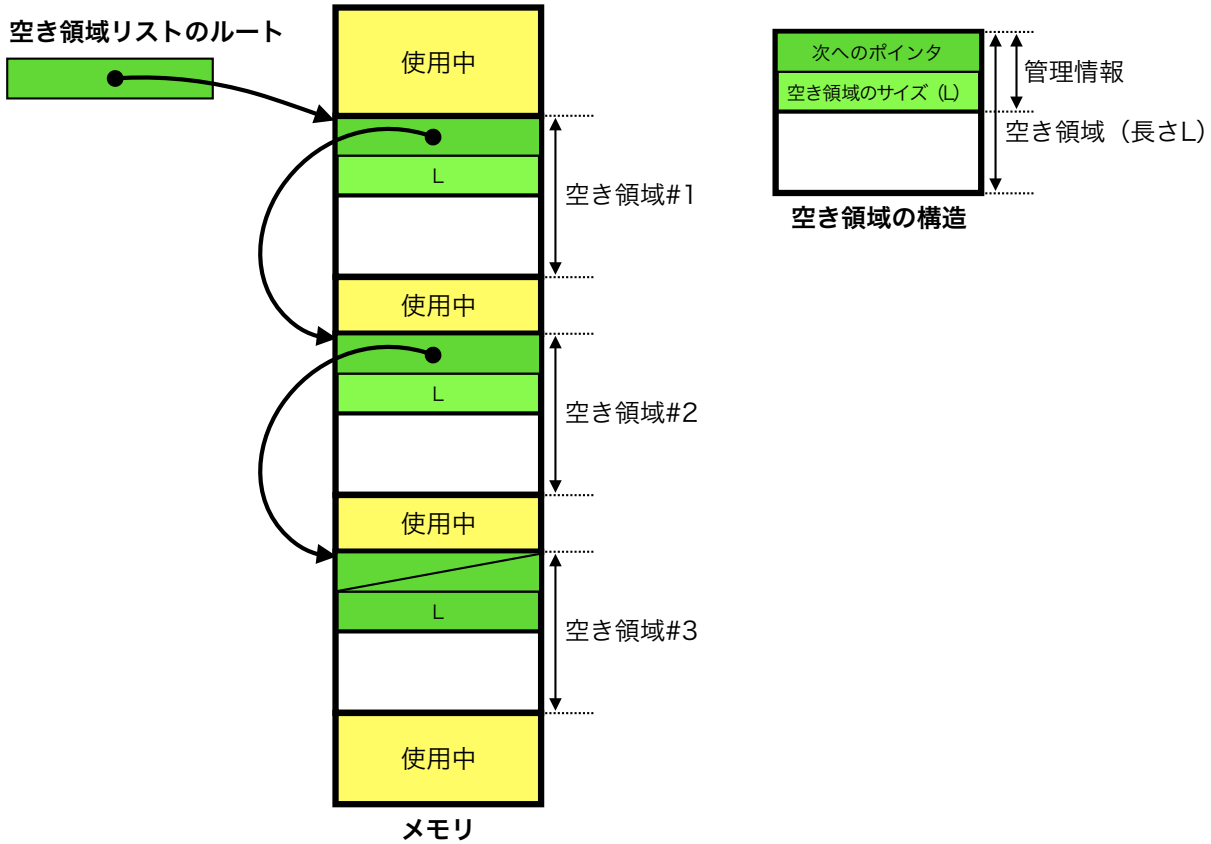
空き領域をリストにして管理する.
様々なサイズの空き領域が混在しても良い.



(a) 前後が使用中の場合　(b) 片側が空き領域の場合　(c) 両側が空き領域の場合

使用中の領域が解放されると隣接する空き領域と合体させる.

空き領域リストのデータ構造



- 空き領域の一部を管理データの格納に使用する.
- アドレス順のリストにして管理する.
- ファーストフィットの探索に都合が良い.
- 隣接領域との合体にも都合が良い.

練習問題

可変区画方式で管理される 100KiB の空き領域がある時，次の順序で領域の割付け解放を行った．ファーストフィット方式を用いた場合とベストフィット方式を用いた場合について，実行後のメモリマップを図示しなさい．

1. 30KiB の領域を割付け
2. 40KiB の領域を割付け
3. 20KiB の領域を割付け
4. 先程割付けた 40KiB の領域を解放
5. 10KiB の領域を割付け