ページの制御

- ページングでは、使うページはページフレームを、そうで無いものはそれなりに
- プロセスの持つ仮想アドレス空間の中身
  - 物理ページ（ページフレーム）のあるページ
  - 補助記憶に退避されたページ
  - 割り当てなしのページ
- 物理ページは即座にアクセス可能
- 使えるページの準備

1. ページフレームの確保
   空きページがなければ、適当なページを退避（ページアウト）
2. 退避ページがある場合は、
   補助記憶から読み出し（ページイン）
3. 新たな割り当てならばページフレーム内を初期化
4. ページテーブルを更新

- ページ作成のタイミング
- ページフレームの割り当てがないページに対して
- デマンドページング：アクセスが生じたときに逐次
- プリページング：アクセスが生じそうなものを事前に
- アクセスが生じたときの合図は割り込みで
  →ページフォールト
- ページの置き換え
- 使うページ全てを事前に予見するのは不可能
- デマンドページングを基本、ある程度プリページング
- コンピュータの動作の性質を利用する
- 局所性（可能性が高い）
- 周りのデータが近い将来に使われる
- 同じ時期に周りのデータが使われる

メモリの応答時間

- ページフォールトの発生
  →ページ入れ替えによる時間ロス
- メモリの体感的な応答時間が大きくなる
- 体感的な平均メモリ応答時間E=(1−P)・M + P・R
  - ページフォールトの確率 P
  - ページの置換えにかかる時間 R
  - 実際の物理メモリの応答時間 M
- 一般にM<<Rなので、Pが大きいとメモリ応答は格段に悪くなる
- 使うページの予測
- E=(1−P)・M + P・R
- E を減らすには、RかPを小さくする
- Rの縮小は物理的な対応
  - 高速なHDDとかSSDを使う
- Pの縮小は
  - 物理メモリを増やす
  - すぐ使いそうなページをうまく残す→ソフトウェア的な対応
- 最適なページ置き換え
- 使わなさそうなページをページアウト
- ページアウトするページを決める手順
  →ページ置換えアルゴリズム
- 最適なアルゴリズム
  →実現はほぼ不可能（部分的には可能）
  - 挙動の全てが記録できていて
  - 挙動が完全に再現される場合
  - 将来使用するページを元に、
    最も参照されないページからページアウト

置換えアルゴリズム

- **FIFO**：First In, First Out
  →確保の古い順
- **NRU**：Not Recently Used
  →参照、変更ビットを確認。
  一定間隔で全てリセット
- **Second Chance**
  →FIFOの改良。参照があれば猶予してキューに入れ直し
- **Clock**
  →ページの参照リストを巡回してチェック。参照がなければページアウト。参照があればリセット（猶予）
- **LRU**：Least Recently Used
  →参照された時期をリスト所持。
  古い参照から
- **NFU**：Not Frequently Used
  →参照された回数をカウント。
  回数が少ないものから
- **Aging**
  →ビット列の最上位に参照ビットを。時間が経つごとに右シフトして値を小さく。ビット列の値が小さいものから

ワーキングセットモデル

- アクセスの局所性
- データやプログラムの特徴（傾向）
  - 連続して配置されている
  - 種類別に分けて配置されている
  - 繰り返し処理される
- 参照先はプログラムカウンタかアドレスレジスタに指定される
- メモリ領域の周辺が近い将来に参照される（時間的局所性）
- 同じ時期にメモリ領域の周辺が参照される（空間的局所性）
- ワーキングセットモデル
- 現在各プロセスが使用しているページの集合→ワーキングセット
- ワーキングセットの一部が物理メモリ上にない→アクセスするとページフォールトが発生
- ページフォールトが頻発するとメモリへのアクセス速度が激減→処理速度も激減（スラッシング）
- ワーキングセットを全て物理メモリにあれば問題はないが・・・
- マルチタスクではプロセス切り替えの際にワーキングセットも大きく変化→スラッシングが一時的に発生
- ワーキングセットをメモリ上に維持しようとするページの割当て手法→ワーキングセットモデル
- 予めプロセス毎のワーキングセットの遷移を保持→実行時にその実績に応じたページを割り当てる（プリページング）
- 使用されないページを予測して置き換え対象ページにすることも可能
- ただし、管理コストは大きいので疑似的な手法が必要
- 利用時期が古いものはセットから外れている→WSClock