

## 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



## 第9回(5月14日1時限)ミニレポート課題 解答例

- 空間的参照局所性、時間的参照局所性のあるプログラムの例をC言語を用いて、それぞれ示せ。

```
int i, s[64], d[64];  
for (i=0; i<64; i++) d[i] = s[i];
```



## 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 **固定**)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 **実アドレス**)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 **ページテーブル**)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その他、実メモリでの(d **外部**)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 **内部**)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 **実アドレス**)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 **セグメントテーブル**)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 **アクセス制御**)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 **内部**)フラグメンテーションは発生しない。

# 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

#### 3.2.4 置換え 3.2.5 メモリ保護



大阪大学大学院情報科学研究科  
村田正幸

[murata@ist.osaka-u.ac.jp](mailto: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

### 3.2.4 置き換え

#### [1]【まとめ】ページフォールトとブロック置換(再掲)

- ページングやページセグメンテーションというページを単位とするマッピング
  - アクセスを要求した仮想ページがメインメモリ(実メモリ)上にない、すなわちページフォールトである場合に生じる割り込みがページフォールト割り込み
  - ページフォールト割り込みは、「アドレス変換時に、ページテーブルにアクセス対象の仮想ページの登録がない、すなわち、ページフォールトが生じる」場合に発生
- ページフォールト割り込みによって、ページフォールトという通知を受けたOSは、その割り込み処理において、以下のブロック置換(ページングやページセグメンテーションの場合はページ置換)を実行する
  1. ファイル装置(仮想メモリのバックアップメモリ)に格納してある当該仮想ページをメインメモリ(実メモリ)の不要な実ページと置換(スワップ)する
  2. ページテーブルを書き換える
- ページ置換はページ単位で実行
  - (A) ページアウト (page-out): メインメモリからのスワップアウト
  - (B) ページイン (page-in): メインメモリへのスワップイン



## 3.2 仮想メモリーメインメモリの隠ぺいー

### 3.2.4 置き換え

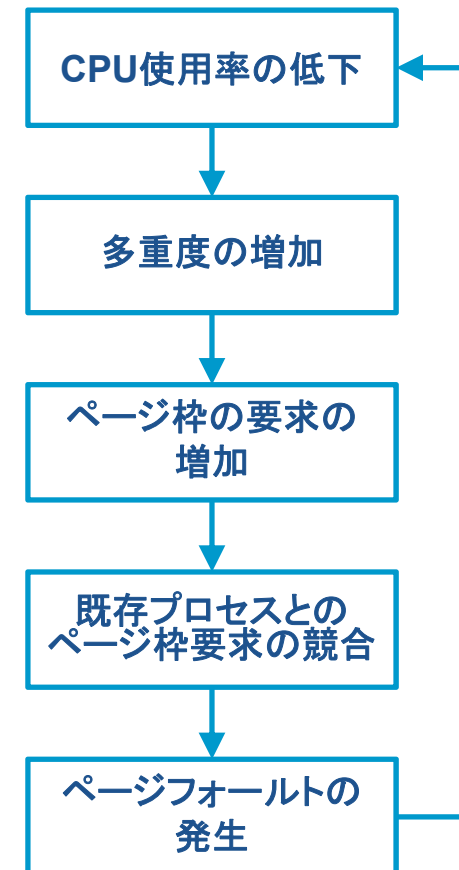
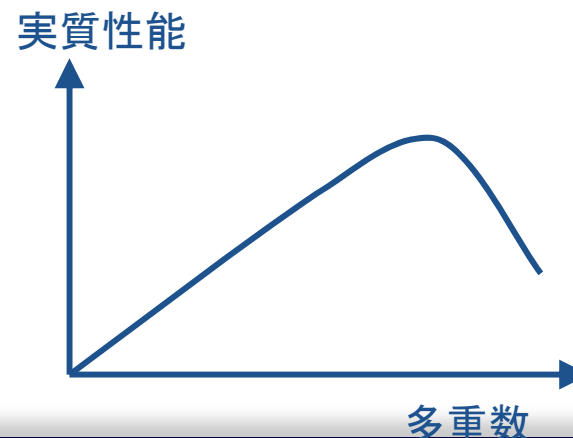
#### [2] ページ置換アルゴリズム

- 使用頻度すなわち参照局所性が高いページをメインメモリ(実メモリ)で保持する
  - 使用頻度、すなわち参照局所性が低い実ページをスワップアウトすることが有効となる
  - ページ置換アルゴリズムの目標は、参照局所性が低いページを決定すること
- ページフォルトが発生すると、OSはまずページ置換アルゴリズムによって、スワップアウトするページを決定
  - ページ置換アルゴリズムが不適切であると、適度なサイズの実メモリ領域が確保できていてもスラッシングを引き起こしてしまう
- ページ置換アルゴリズムの評価指標
  - (a) ページのライフタイム: ページフォルト間の平均時間間隔すなわちページの実メモリ上の平均滞在時間
  - (b) ページフォルト率: ライフタイムの逆数



# スラッシング

- スラッシング
  - 仮想メモリ機構では、実メモリ容量が小さすぎると、実メモリ上での参照局所性の利用が不可能となり、ブロック置換が頻発する
  - CPUがOS機能であるブロック置換管理にかかりきりとなって、ユーザプロセスおよびブロック置換管理以外のOS機能の実行が実質上不可能となること
  - スラッシングはオーバーヘッドであるので、仮想メモリの効果を生かすためには、十分な容量の実メモリを装備かつ確保して、スラッシングの発生を防止する必要がある
- マルチプロセス環境において右のような悪循環に陥る





## 3.2 仮想メモリーメインメモリの隠ぺいー

### 3.2.4 置き換え

#### [3] 代表的なページ置換アルゴリズム

##### 1. FIFO (First In First Out) : 到着順ページ置換えアルゴリズム

- メインメモリに一番最初にすなわち最古にスワップインしているページを最優先のスワップアウト対象とする
- ページ置換機構の実装コストはLRUとランダムとの中間
- アクセスや参照はチェックしないので、参照局所性の反映度はLRUよりも低い

##### 2. LRU (Least Recently Used): 最長不使用ページ置換えアルゴリズム

- 最後のアクセス時刻が最古であるページを最優先のスワップアウト対象とする
- 長所: 時間的参照局所性を反映しているので、ページフォールト率は低くなる
- 短所: アクセス履歴を記録し比較するページ置換管理機構は複雑で、ページ置換時間は長くなる

##### 3. LFU (Least Frequently Used): 最低使用頻度順ページ置換えアルゴリズム

- メインメモリに読み込まれているページの中で、もっとも使用頻度の少ないページをスワップアウト対象とする
- 長所: 時間的空間的参照局所性を反映しているので、ページフォールト率は低くなる
- 短所: アクセス履歴を記録し、比較するページ置換管理機構が複雑になる

##### 4. ワーキングセット (working-set)

- ワーキングセットではないページを優先すべきスワップアウト対象とする
- ワーキングセットではないページのうちから、他のアルゴリズムによってスワップアウト対象を決定する





## 3.2 仮想メモリーメインメモリの隠ぺいー

### 3.2.4 置き換え

#### [3] 代表的なページ置換アルゴリズム

##### 5. ランダム (random)

- 任意あるいは無作為にスワップアウトするページを決定する
- 長所: 簡単に実現あるいは実装できる
- 短所: 参照局所性をまったく考慮していない、すなわち、アルゴリズム (戦略) とはいえない

##### 6. OPT (MIN): 最適アルゴリズム

- 次回に参照されるのが最も遠い未来であるページをスワップアウト対象とする
- 実行してみないとわからないので、実用的ではない

#### 他にも種々のページ置換アルゴリズムがある



- 実際には、ページ置換アルゴリズムよりも、アクセスや参照の対象であるプログラムやデータブロックそのものが示す参照局所性がページフォールト率を左右することも少なくない
- どのアクセスや参照の対象 (プログラムやデータ) に対してもページフォールト率が低くなるような最適なページ置換アルゴリズムの設計や選定は困難



# ページ置換えアルゴリズム:FIFO

- メインメモリ上の最も古いページを置き換える
- 実装
  - メインメモリ上のすべてのページに対するFIFOキュー
  - 置換えが必要になると、キューの先頭のページを選び、ページアウト
  - ページイン⇒キューの最後につなぐ

ページフォールト	p	p	p	p	p	p	p			p	p	
参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容 (FIFOキュー)	0	0	0	1	2	3	0	0	0	1	4	4
		1	1	2	3	0	1	1	1	4	2	2
			2	3	0	1	4	4	4	2	3	3



参照ストリング：仮想アドレス空間の特定のアドレス参照列



## 演習問題3.2

- 置き換えアルゴリズムとしてFIFOを考え、参照列が0 1 2 3 0 1 4 0 1 2 3 4 のとき、ページ枠数が1、2、4、5の場合はどうなるか？ ページフォールト数を求め、ページ枠数とページフォールト回数の対応表を作成せよ

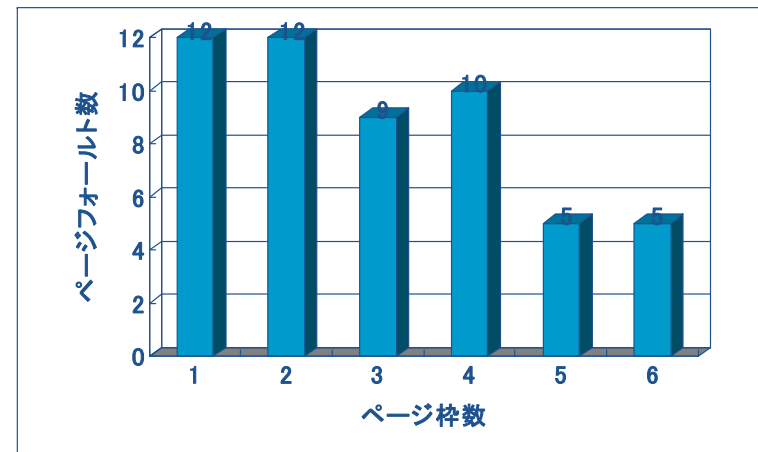


# Beladyの異常 (Belady's anomaly)

- メモリを増強してページ枠を4に増やすとフォールト率がかえって増加する
  - FIFO異常

ページ フォールト	p				p							
参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容 (FIFOキュー)	0	0	0	0	0	0	1	2	3	4	0	1
		1	1	1	1	1	2	3	4	0	1	2
			2	2	2	2	3	4	0	1	2	3
				3	3	3	4	0	1	2	3	4

- Beladyの異常を起こさないアルゴリズム
  - 「スタックアルゴリズム」と呼ばれるクラス
  - OPT、LRUなど





# ページ置換えアルゴリズム: OPT

- 別名MIN: 最小のページフォルト率
- 選択するページ
  - 次回に参照されるのが最も遠い未来である
  - もちろん実用的ではない
  - 他のアルゴリズムの性能を評価するための比較対象

ページフォルト

p p p p p p p

参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容	0	1	2	3	3	3	4	4	4	4	4	4
		0	1	1	1	1	1	1	1	2	3	3
			0	0	0	0	0	0	0	0	0	0



# ページ置換えアルゴリズム: LRU

- メインメモリにあるページのうち最も長い間参照されていないページを選択
  - 実際に利用される例が多く、よいと考えられている
- ただし、ハードウェアによる支援が必要
  - スタック: 参照されたページ番号をスタックの最上部に入れる。スタックの最下部がもっとも古いものになり、そのページを選択する。

ページフォールト

p p p p p p p p p p

参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容 (スタック)	0	1	2	3	0	1	4	0	1	2	3	4
		0	1	2	3	0	1	4	0	1	2	3
			0	1	2	3	0	1	4	0	1	2



## 第11回(5月21日1時限)ミニレポート課題

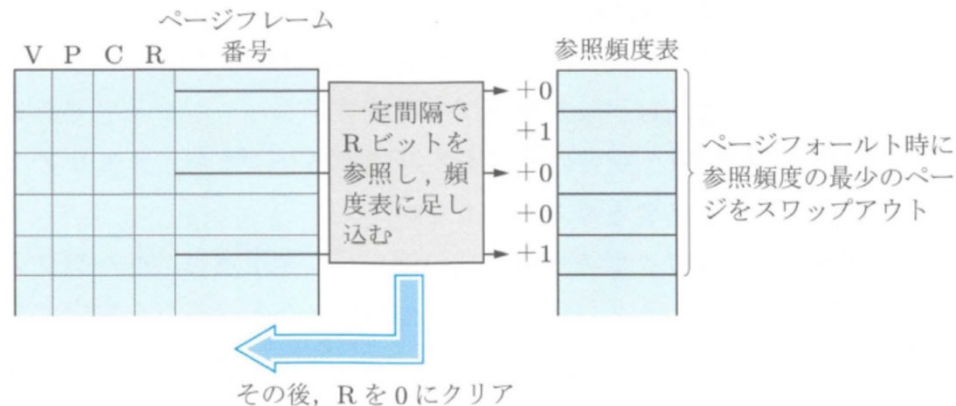
1. LRUアルゴリズムの動作例を、参照ストリングを“0 1 2 3 4 5 1 0 3 2 0 4 5 1”、ページ枠を4とした場合について、LRUスタックの変化の様子を示せ。また、実メモリ上のページ枠の内容の変化の様子を別に示せ。
2. 次にアクセスされる確率をもっとも低いページを選択する近似的な手法としてLRUアルゴリズムが用いられる理由を説明せよ。
3. LRUアルゴリズムを正確に実装することが困難な理由を説明せよ。

締切: 5月26日(水)  
CLEで提出



# LRUアルゴリズムの近似的実装

- 参照ビットにより、近似的にLRUを実現する
  - ページテーブルに参照ビット (reference bit) R とよばれる新たなビットを導入する。
  - このビットは、ページフォールトのたびに0にクリアされる。
  - 当該ページがアクセスされた時点で1にセットされる。
  - Rビットは、ページフォールトからつぎのページフォールトまでの間にページが参照されたかどうかを示す。
- さらに、各ページごとにページの参照頻度を示すエントリーを用意することも考えられる
  - ページフォールト時に、参照ビットをチェックし、参照ビットがセットされている場合は、テーブル内の参照頻度を増加させ、もっとも参照頻度の低いページをページアウトの対象とする。
  - ページフォールト時でなく、定期的に(適当な間隔で)、テーブル内の参照頻度を増加させることも考えられる。このほうが、よりLRUに近くなる。







# スタックアルゴリズム

- スタックアルゴリズム
  - 参照ストリングの各要素において、  
[ページ枠 $p$ 個の場合メインメモリに置かれるページ集合]  $\subset$  [ $p+1$ 個の場合のページ集合]
  - 右はLRUで3ページ枠、4ページ枠の例
- メインメモリを増やすとページフォールト率は下がる。問題は、良いとされているLRUがFIFOより悪いこと(3ページ枠のとき)
- どの場合にも通用する最適解はない
  - ある与えられたメインメモリサイズに対して最適なアルゴリズムはある
  - メインメモリサイズを変化させると最適なアルゴリズムは変わる
  - ただし、どんなメインメモリサイズに対しても最適なものはないが、「だいたいよい」ものがある

ページフォールト

p p p p p p p p p p

参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容 (スタック)	0	1	2	3	0	1	4	0	1	2	3	4
		0	1	2	3	0	1	4	0	1	2	3
			0	1	2	3	0	1	4	0	1	2

↑↑↑↑↑↑↑↑↑↑  
↓p ↓p ↓p ↓p ↓p ↓p ↓p ↓p ↓p ↓p

ページフォールト

参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容 (スタック)	0	1	2	3	0	1	4	0	1	2	3	4
		0	1	2	3	0	1	4	0	1	2	3
			0	1	2	3	3	3	4	0	1	



# ページ置換アルゴリズムの評価方法

- 置換えアルゴリズムの評価方法
  - 仮想アドレス空間の特定のアドレス参照列(参照ストリング)を用いてアルゴリズムを評価
  - 参照ストリングの生成
    - 乱数発生器
    - 既存のプログラムの実行時のトレースデータ
- 評価プログラム
  - 入力
    - 参照ストリング
    - ページ枠数
  - 出力
    - 置換え要・不要の列、置換えるページ番号の列⇒ページフォールト率



## 参照局所性の実際(再掲)

- type 1: write(赤)
- type 2: read(青)
- type 3: execute(緑)

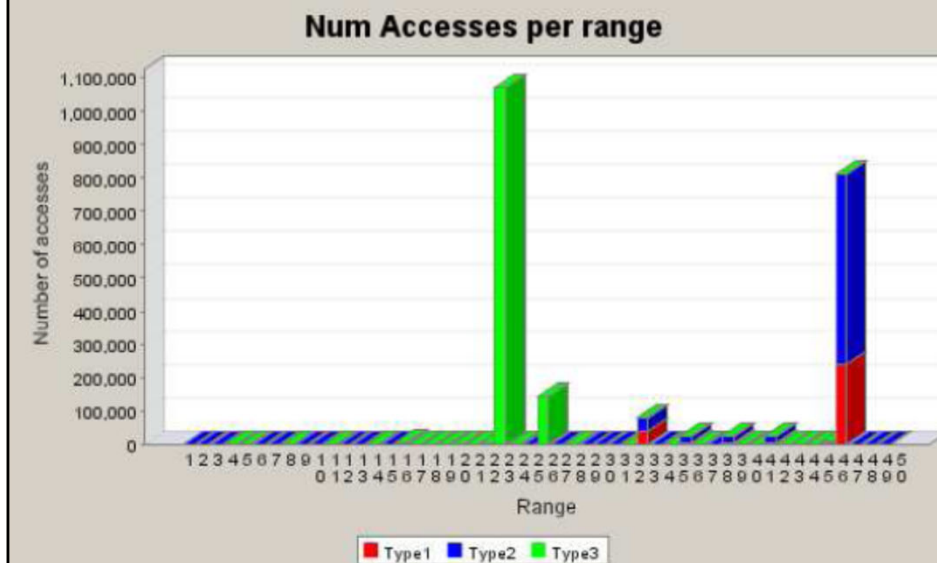


Fig. 2. Access pattern of m-ft

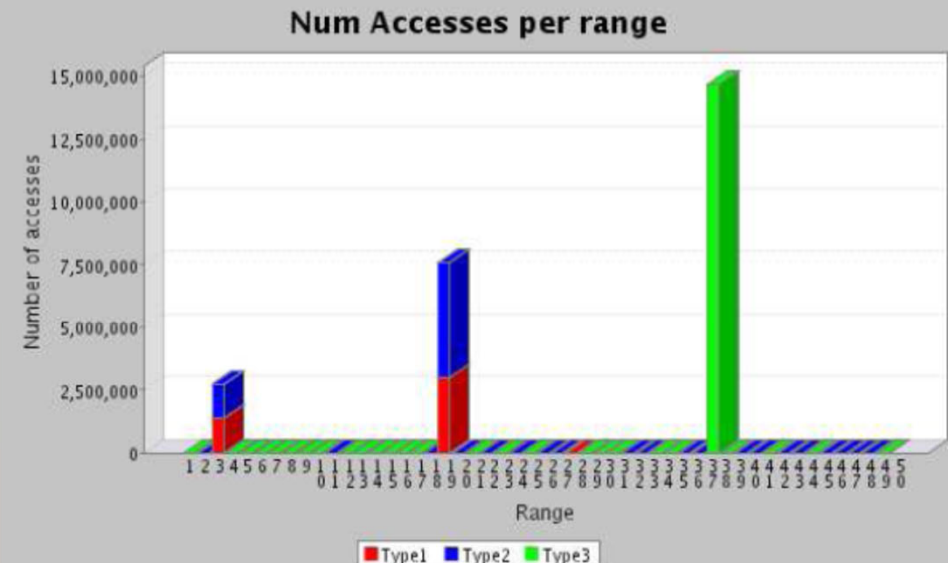


Fig. 1. Access pattern of tower of Hanoi program



## 演習問題3.3

- 以下のプログラムを実行した時の参照ストリングを求めよ。

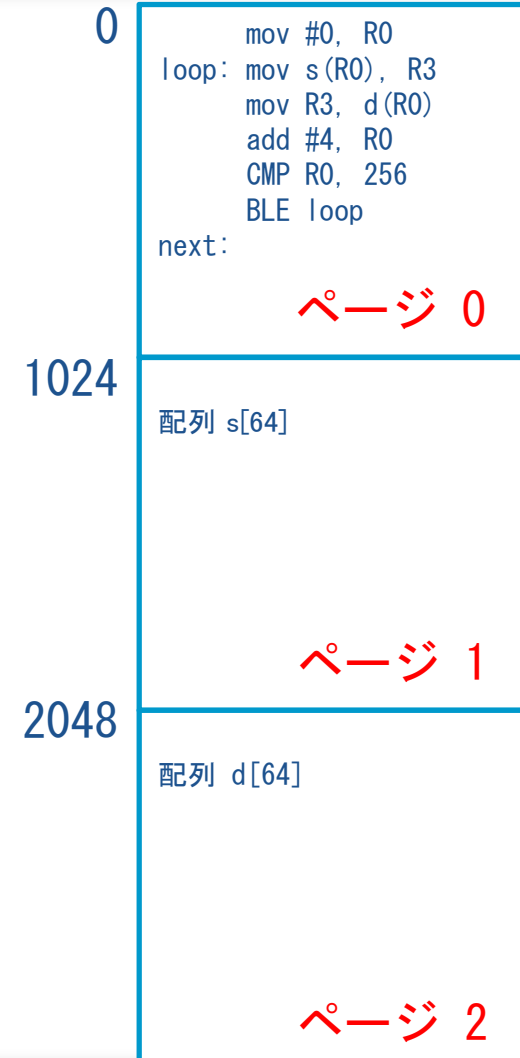
```
int s[64], d[64];  
register int i;  
for (i=0; i<64; i++) d[i] = s[i];
```

- ただし、プログラムはページ0、配列sはページ1、配列dはページ2に割り当てられるものとせよ。
- 同じページへの繰り返し参照は1つにまとめて書いてよい
  - 11100221⇒1021



## 演習問題3.3ヒント

```
    mov #0, R0
loop: mov s(R0), R3
      mov R3, d(R0)
      add #4, R0
      CMP R0, 256
      BLE loop
next:
```





# ページ置換えアルゴリズムの実際

- A. Aggrawal, “Page Replacement Algorithm Simulator”
  - プログラムの実行時のトレースデータを使用
- LRUはよいが、その近似アルゴリズムCLK(参照ビットを用いる方法)はさほどでもない(左図)
- いつも理屈どおりとは限らない(右図)

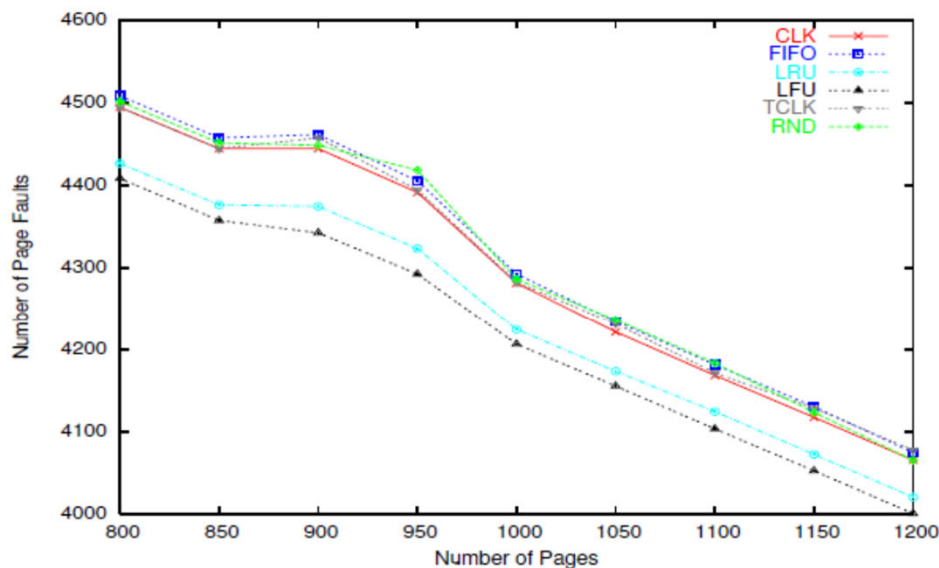


Fig. 9. Performance of page replacement algorithms on m-sor program

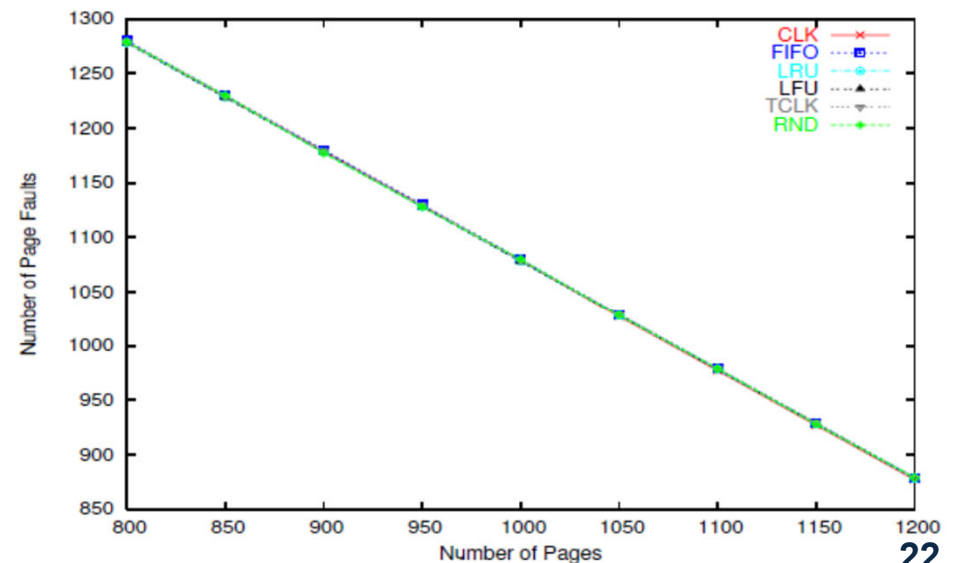


Fig. 13. Performance of page replacement algorithms on sacc program



# ワーキングセットモデル

- Denningが提案したプロセスの振る舞いのモデル
- ワーキングセット: プロセスの実行の各時点で、プロセスが活発に参照するページの集合
  - ワーキングセットがメインメモリに在ればよい
  - ない場合はページフォルトが頻発する: スラッシング
- ワーキングセット  $W(t, w) := [t-w, t]$  の間にプロセスによって参照されるページ集合
  - ここでの時間は仮想時間→他のプロセスが実行される時間を含めない
  - 変数  $w$ : ウィンドウサイズ
- ワーキングセット法: ワーキングセットをメモリにできるだけ保持する技法→マルチプログラミング環境で有用になる

## $w=5$ の場合

ページフォルト	p	p	p	p	p	p	p		p	p	p	
参照ストリング	0	1	2	3	0	1	4	0	1	2	3	4
	0	1	2	3	0	1	4	0	1	2	3	4
ページ枠の内容		0	1	2	3	0	1	4	0	1	2	3
			0	1	2	3	0	1	4	0	1	2



## 3.2 仮想メモリーメインメモリの隠ぺい

### 3.2.4 置き換え

#### [4] ページ置換のタイミング

- いつ、どのタイミングでページを仮想アドレス空間から実アドレス空間に読み込む(ページイン)か？

##### (A) デマンドページング(要求時ページング):

- プログラム(実行中のプロセス)自身が実行時(動的)にアクセスや参照を要求(デマンド)するページをそのたびに読み込む。
- OS機能をハードウェア機構が支援する、つまり、OSとハードウェアの機能分担する動的ページング
- 長所
  - (1) 必要なすなわち要求するページだけを読み込む点では無駄がない
- 短所
  - (2) プロセス実行の開始時、すなわち、初めての実行中状態への選移時にページフォールトが集中発生するので、これによるページ置換処理の間は、先に読み込まれるページを使用するプロセスは未実行で実行可能状態のまま待つという時間的かつ空間的な冗長性が存在する
- 適切なページ置換アルゴリズムの選択が重要な要件となる





## 3.2 仮想メモリーメインメモリの隠ぺいー

### 3.2.4 置き換え

#### [4] ページ置換のタイミング(続き)

##### (B)プリページング(先行ページング、予測ページング):

- OSがあらかじめ「アクセスや参照がある」と予測するページ(通常は複数個)を実行前(静的)にまとめて読み込む
- OS機能をコンパイラ機能が支援する
- OSとコンパイラの機能分担による静的ページング
- 長所
  - (1) 予測の的中率が高ければ、複数ページを一度に読み込むブロック転送が適用できるので、読み出し回数とページ当たりの平均読み出し時間は減り、高速化が達成できる;
- 短所
  - (2) 不要なページも読み込んでしまう可能性がある
  - (3) 予測の的中率が低いと、予測に要するオーバーヘッドが顕在化する。予測に要するオーバーヘッドを上回る(ページ置換そのものの)高速化が必須の要件となる
- **現代では、メインメモリの実装コストが低くなって、大容量のメインメモリ(実メモリ)を実装できるのが普通**
  - 大容量のメインメモリを実装していれば、余分で冗長なページも読み込んでおくことができる
  - 予測がはずれる、すなわち「読み込んだページを使用しない」場合の影響は少ない
  - (B)のプリページングを併用するOSが多い



## 3.2 仮想メモリーメインメモリの隠ぺいー

### 3.2.5 メモリ保護

#### [1] OSによるメモリ保護機能

- メインメモリ領域に保持するプロセス(プログラム)の個々について、アクセス権にしたがって、相互に保護し合う
- OSのメモリ保護機能はOSがアクセス権を管理することによって実現
- ユーザプロセス(ユーザプログラム)の実行中にアクセス権に違反する不正アクセスがあれば(メモリ保護違反)、それを要因とする割り込みが発生する
- OSは、メモリ保護違反による割り込みを受け付けると、ユーザプログラムの実行(プロセッサ状態はユーザ状態)からOSの実行(プロセッサ状態はカーネル状態)に切り替えて、メモリ保護違反に対する割り込み処理を行う

#### 定義3.6 (アクセス権)

- 当該プロセスが、他プロセスに対して、あるアクセス形態を許可するか禁止するかに関する取り決めをアクセス権という
- アクセス形態の例: 読み出し、書き込み、(命令としての)実行など。
- アクセス権はプロセス(プログラム)ごとに設定