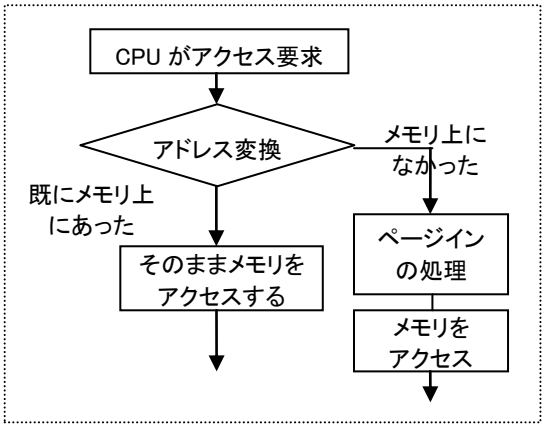
OS 第10回予習資料　　第10回の授業までに空欄を埋めて予習　　質問は事前＋当日受付けます

第10回　デマンドページングの置き換えアルゴリズム、　ファイルシステム

（前回） 物理メモリ（メインメモリ）は最初は空で　　⇒ （今回） では、どんどんページインで持ってきたら
　必要になったページを HD からページインする　　あふれてしまうではないか？
　　前提は、 物理メモリ容量 ＜ 仮想メモリ容量

10－1. （復習）デマンドページングの仕組を説明してみよう




デマンドページングの場合は、メモリのイメージはメインメモリ上ではなく、（　　）に置かれている。
このイメージと、メインメモリと、両方とも同じ大きさの（　　）に分割されている。
CPUがイメージ上の欲しい部分に（＝仮想アドレスで指定して）アクセスしようとすると、メインメモリ上には無いので、（　　）の処理をする。
この処理の手順は、①欲しい（　　）が HD 上のどこにあるかを求め、②HD からメインメモリの（　　）場所へ転送し、③最後にマッピングが正しくなるように（　　）。
このとき、②でメインメモリに空きがない場合、すでに使われているスペースを「追い出し」て空きを作る。ここで、誰を追い出すをか決めなければならない。 それがページ置き換えのアルゴリズムである。

10－2. ページ置き換えのアルゴリズムを比較してみよう

置き換えアルゴリズムは、（　　）を決めます。このとき、すぐあとに利用する仮想ページを追い出すと、その追い出した仮想ページをまたアクセスすることになり、（追い出したので）メインメモリ上に無いので、（　　）と判定され、ページインしなければなりません。 ミスの回数が増え、（　　）が下がり、その結果ページアクセス時間の期待値は（　　）します。
つまり、置き換えアルゴリズムによって、ページアクセス時間の期待値が増減します。 なるべくアクセス時間が小さい方が、早くメモリを読み書きできるので、コンピュータの処理性能は高くなります。

スライドでは、下記の 3 つのアルゴリズムを紹介しています。

| 名称 | | | |
|---|---|---|---|
| 動作原理 | | | |

このほかに、ランダムに選ぶ、というのも考えられます。

期待される性能(トータルしてページミス率が小さい/ヒット率が大きい)順に順位を付けると

最大(　　) ＞ まずまず(　　) ＞ (　　) ＞ (ランダム)

の順になるでしょう。(ランダムの性能は微妙かも知れませんが)

実際には、性能(ページミス率/ヒット率)は、メモリへのアクセスの(アドレス)パターンに依存します。 たとえば、仮想アドレス空間の中をランダムにアクセスする(ピョンピョン跳ね回る)と、アクセスの(　　)性が成り立たず、ミスを繰り返すことになりますが、アクセスがランダムであればページの追い出しアルゴリズムにいくら工夫をしても、効果はありません(平均で見て)。 しかし、通常のプログラム実行では、アクセスの(　　)性が成り立っているので、最近参照したページは再び参照される可能性が(　　)と言えます。 従って、最近参照したページはなるべく(　　)ようにし、追い出すページは(どのような　　)ページを選ぶ方が、ミスが少なくなると考えられます。 LRU はそれを実現しようとする方式です。

アクセスするページ番号の「列」を与えて、その時に仮想ページがどのようにメインメモリ上に持ってこられるか/追い出されるかを、追いかけてみること(シミュレーション)が行われます。スライドの例をまねして、動きを追いかけてみましょう。

＜条件＞ 参照列(アクセスする仮想ページ番号) 0 1 2 3 4 0 1 2 5 0 1 2 3 4 5　メインメモリ(物理ページ)の量＝4

＜書き方＞ 表の行「物理 0～3」は物理ページを指定しているので、その物理ページに入れる仮想ページ番号を記入

「フォルト」は、その仮想ページを持ってくるときにページフォルトが起こったら×、起こらなければ空欄

＜まずは、FIFO で書いてみよう＞

| 参照列 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 5 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 物理 0 | | | | | | | | | | | | | | | |
| 物理 1 | | | | | | | | | | | | | | | |
| 物理 2 | | | | | | | | | | | | | | | |
| 物理 3 | | | | | | | | | | | | | | | |
| フォルト | | | | | | | | | | | | | | | |

＜同様に、LRU で書いてみよう＞

| 参照列 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 5 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 物理 0 | | | | | | | | | | | | | | | |
| 物理 1 | | | | | | | | | | | | | | | |
| 物理 2 | | | | | | | | | | | | | | | |
| 物理 3 | | | | | | | | | | | | | | | |
| フォルト | | | | | | | | | | | | | | | |

＜ついでに、LRU で、物理ページが 5 ページある場合を試してみよう。何が変わるか？＞

| 参照列 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 5 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 物理 0 | | | | | | | | | | | | | | | |
| 物理 1 | | | | | | | | | | | | | | | |
| 物理 2 | | | | | | | | | | | | | | | |
| 物理 3 | | | | | | | | | | | | | | | |
| 物理 4 | | | | | | | | | | | | | | | |
| フォルト | | | | | | | | | | | | | | | |

＜参照列全体を見渡して OPT になるような追い出しを考えてみよう（スライド参照）＞

| 参照列 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 5 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 物理 0 | | | | | | | | | | | | | | | |
| 物理 1 | | | | | | | | | | | | | | | |
| 物理 2 | | | | | | | | | | | | | | | |
| 物理 3 | | | | | | | | | | | | | | | |
| フォルト | | | | | | | | | | | | | | | |

FIFO と LRU の動作を、自分でよく比較してみてください。

10－3　ワーキングセットの考え方

ワーキングセットとは？　（教科書 107 ページ）

ワーキングセットの変化

ある時刻 t の直前の幅 w の時間［t−w, t］を考えます。 この時間［t−w, t］内での仮想アクセスアドレスの集合を W(t,w) とします。この W がワーキングセットに当たります。

ワーキングセット W は時間により変化します（t の関数）。時間による大きさ（要素数）や位置の変化は、プログラムがどんなメモリアクセスをするかによって違ってきます。（教科書 p108 の図 6.5）

たとえば、仮想メモリ空間上を広く跳び回るプログラムの場合、W の大きさ自体が大きくなります。 また、プログラムが進行するに従ってアクセスする位置が変わっていくプログラムの場合、W は時間の進行に従って位置が変わりますが、ある時点での大きさはそれほど大きくないかもしれません。

デマンドページングでは、ワーキングセットがメインメモリ（物理メモリ）の中に入ってしまえば、その時点ではほとんどページミスが起こらないことになります。ワーキングセットが時間とともに移動するでしょうから、その移動の分だけがページミスとなって新たに物理メモリに取り込まれる、その時にページ追い出しも起こるでしょうが、追い出したページをすぐに再度利用することは起こらない、という形になるでしょう。他方、ワーキングセットより物理メモリが小さければ、常時ページミスとそれによる追い出しが起こり、その場合はすぐに必要になるページを追い出すことも起こり得ます。

実際には、走っているプロセス（プログラム）が複数なので、ワーキングセットも複数あることになり、その合計の分量が物理メモリに収まらなければなりません。

10－4　スラッシング

スラッシングとはどういう現象ですか？

スラッシングは、どういう時に発生するのですか？

スラッシングの対策を2つ挙げてください。

①

②

10－5　ページテーブルの「アクセスビット」と「ダーティビット」

a. ページテーブルのアクセスビット

ページテーブルとは何でしたか？ ……

……

ページテーブルのページエントリごとに、「アクセスビット」 を用意してあります。

アクセスビットは、そのページに対するアクセス（読み・書きのアクセス）があったら、1を立てます。（これはアクセスがページテーブルを通過するたびに、ハードウェアでビットを立てます。）

一定時間 T ごとに、（ページテーブル上にある）アクセスビットをチェックし、更に必ずすべて0にクリアします。（これはソフトウェアで行います。）

チェック時にもしあるページ X のアクセスビットが

1であれば、過去 T の間に（　　）ことが分かります。

0であれば、過去 T の間に（　　）ことが分かります。

これにより、疑似的に LRU を実現します。 いちばん単純なやり方としては、もし過去 T の間にアクセスが無ければ、このページは最近（　　）と判断し、LRU の追い出しの対象とします。

b.　ページテーブルのダーティビット

アクセスビットと同様に、ダーティビット（汚れていますビット）を用意してあります。

ダーティビットは、そのページに書き込みアクセスがあったら、1を立てます。（アクセスビットと同様に、ハードウェアでビットを立てます。）

これは、ページを追い出すときに、そのページをハードディスクに書きだすかどうかを決めるときに使います。

仮想ページの情報が、もし、ハードディスクからメインメモリに持ってきた時点から追い出す時点までの間に、1 回も書込みされなかったとすると、追い出しの時点でもハードディスク上の内容とメインメモリ上の内容は（＿＿＿＿）はずです。

したがって、追い出しの時に、わざわざ（時間をかけて）ハードディスクに書き戻す必要は無いことになります。

もし、1 回でも書き込み操作があれば、メインメモリのページの内容をハードディスクに書き戻さなければなりません。

この、当該ページに書き込みがあったかなかったかの判定をするために、（＿＿＿＿＿＿＿＿）を使います。

～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～

10－6　ファイルシステム

ファイルシステムの役割は　①（　　）　②（　　）　③（　　）。

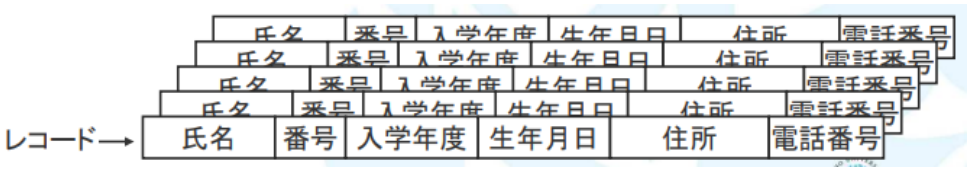
ファイルの内容は、

①構造を持たない場合　⇒　（　　）の連なり　⇒　解釈は（　　）

②構造を持つ場合　⇒　たとえば、「レコード」を単位とし、その中の「フィールド」ごとに何が入るかを決めている

(a) OS がサポートする場合や

(b) データベースシステムがサポートする場合がある




ファイル操作の抽象化：

機器や媒体によらず、操作を共通化する。（記憶デバイスだけではなく、入出力デバイスも（なるべく）共通にする）

次ページへ続く

4 つの基本操作

①(　　) 何をするか(　　)

②(　　) 何をするか(　　)

③(　　) 何をするか(　　)

④(　　) 何をするか(　　)

この4つで足りないこともある。 たとえばファイル上のレコードの「早送り」「巻き戻し」「第xxブロックへのアクセス」など。時間があれば、これらに対してどう対処しているか調べてみると面白かろう。

「アクセス方式」は、アクセスの仕方の違いで、よく出てくるのが

①ファイルを先頭から(　　に)読む (　　)と

②ファイルの任意のブロックを指定して読める (　　)である。

①のイメージは (　　)のようなものであり、ユーザのできる操作は(原則として)先頭から読むことだけである。 任意のブロックを読むためには、(　　)をした後、(　　)しなければならない。

②のイメージは (　　)のようなものであり、ユーザはブロック(　　)を指定して読む。

ハードウェアの構造とアクセス方式の関係は、次のように言える。

ハードウェアが本質的に (　　)アクセス方式の例として、磁気テープを考える。 磁気テープの使い方は音楽カセットテープや VHS ビデオテープなどと同じで、先頭から順に読めるが、任意の場所を(同じアクセス時間で)読むことはできない。(厳密には早送りができるが、早送りを指定場所で止めることは(読んでみないと)できないこと、どちらにしても早送りの時間がかかること、がある。)

このような磁気テープを使って、ハードディスクのような (　　)アクセス方式を実現することを考える。 ユーザの指定した(任意の)ブロックをアクセスするためには、①テープを巻き戻して、②先頭から読んでいって(空読みと呼ぶ)ちょうど欲しいブロックの所まで到達したらデータを読み出す、という手順を行う必要がある。 これは、できないわけではないが、時間がかかる。 つまり、磁気テープは(　　)アクセス方式は、実現できるが、向いていない。

逆に、ハードウェアが本質的に (　　)アクセス方式である、ハードディスクはどうか。 ハードディスクは、アームの移動とディスクの回転時間はかかるが、任意のブロックに非常に短時間でアクセスできる。 これを使って、磁気テープのような(　　)アクセス方式を行うと、どうなるか？ ブロック番号の順にアクセスすることは、(任意のブロックにほぼ同じ短時間でアクセスできるのだから)全く問題がない。 つまり、ハードディスクは(　　)アクセス方式も(　　)アクセス方式も、どちらも問題なく実現できる。

まとめると、ハードウェアの構造とアクセス方式の組合せは、

| ハードウェア | (　　)アクセス方式 | (　　)アクセス方式 |
|---|---|---|
| 磁気テープなど順次アクセス | | |
| ハードディスクなどランダムアクセス | | |