Virtuální paměť

PB 152 ◇ Operační systémy

Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze: jaro 2017

Virtualizace paměti

- principy, základy stránkování na žádost, *Demand Paging*
- důvody pro virtualizaci
- procesy v prostředí s virtuální pamětí
- politiky práce se stránkami
- nahrazovací algoritmy

Používané pojmy

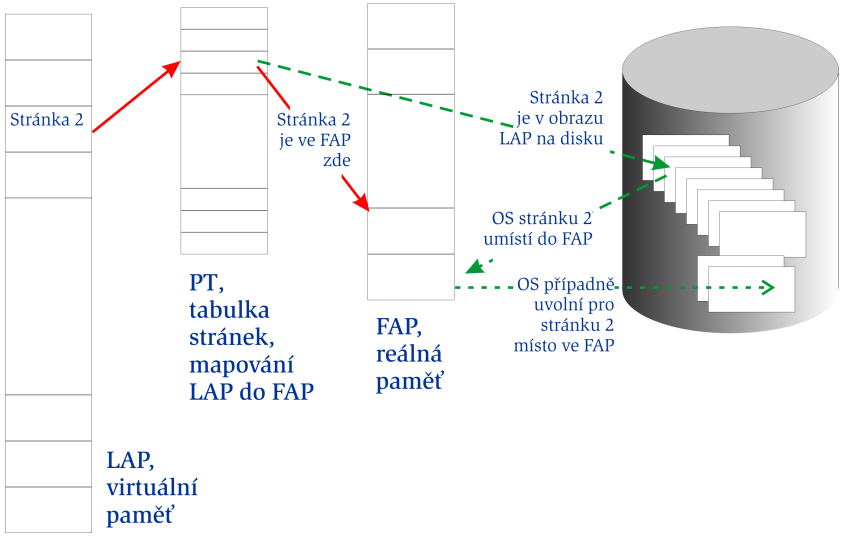
Virtual memory Logická paměť	A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses.The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations.
Virtual address Logická adresa	The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory.
Virtual address space	The virtual storage assigned to a process. Logický adresový prostor, LAP
Address space	The range of memory addresses available to a process. Fyzický adresový prostor, LAP
Real address	The address of a storage location in main memory. Fyzická adresa

- převažující model virtualizace – stránkování na žádost

Virtuální paměť, základní principy

- separace LAP a FAP
  - ✓ LAP – virtuální paměť, manipulovaná po stránkách (část LAP procesu je zobrazena v reálné paměti, celý LAP procesu je zobrazený na disku)
  - ✓ FAP – reálná paměť, spravovaná po rámcích
- ve FAP se musí nacházet alespoň stránky programu potřebné pro bezprostřední řízení procesu
- stránky procesu umístěné ve FAP nemusí být umístěné v sousedních rámcích FAP
- soudobé architektury – LAP je (podstatně) větší než FAP
- adresové prostory ve FAP lze (snadno) sdílet
- procesy lze tudíž vytvářet efektivněji
- techniky implementace virtuální paměti
  - ✓ stránkování na žádost, *Demand Paging*, převažující model
  - ✓ segmentování na žádost, *Demand Segmentation*

LAP je (podstatně) větší než FAP



Stránkování / segmentace, prostá varianta / na žádost

Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Main memory partitioned into small fixed-size chunks called frames	Main memory partitioned into small fixed-size chunks called frames	Main memory not partitioned	Main memory not partitioned
Program broken into pages by the compiler or memory management system	Program broken into pages by the compiler or memory management system	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)
Internal fragmentation within frames	Internal fragmentation within frames	No internal fragmentation	No internal fragmentation
No external fragmentation	No external fragmentation	External fragmentation	External fragmentation
Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a segment table for each process showing the load address and length of each segment	Operating system must maintain a segment table for each process showing the load address and length of each segment

Stránkování / segmentace, prostá varianta / na žádost

Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Operating system must maintain a free frame list	Operating system must maintain a free frame list	Operating system must maintain a list of free holes in main memory	Operating system must maintain a list of free holes in main memory
Processor uses page number, offset to calculate absolute address	Processor uses page number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address
All the pages of a process must be in main memory for process to run, unless overlays are used	Not all pages of a process need be in main memory frames for the process to run. Pages may be read in as needed	All the segments of a process must be in main memory for process to run, unless overlays are used	Not all segments of a process need be in main memory for the process to run. Segments may be read in as needed
	Reading a page into main memory may require writing a page out to disk		Reading a segment into main memory may require writing one or more segments out to disk

Stránkování (P:) versus segmentování (S:)

- P: FAP je 1-dimensionální,
  alokovaný po dílech – rámcích – pevné délky
  S: FAP je 1-dimensionální,
  alokovaný po dílech proměnné délky
- P: LAP je 1-dimensionální,
  dělí se na stránky definicí délky stránky v CPU
  S: LAP je 2-dimensionální (segmenty x buňky v segmentu,
  segmenty LAP vymezuje konstrukce adresy v instrukci
- P: Vnitřní fragmentace v rámcích, žádná externí fragmentace
  S: Žádná vnitřní fragmentace, možnost externí fragmentace

Stránkování (P:) versus segmentování (S:)

- P: OS udržuje tabulku stránek s určením rámce pro každou stránku
  S: OS udržuje tabulku segmentů s bází a délkou každého segmentu
- P: OS udržuje tabulku rámců definující stav každého rámce
  S: OS udržuje seznam volných oblastí FAP
- P: LAP → FAP se řeší dynamicky při běhu mikroporgramem podle tabulky stránek, PT
  S: LAP → FAP se řeší dynamicky při běhu mikroporgramem podle tabulky segmentů, ST

Stránkování (P:) versus segmentování (S:)

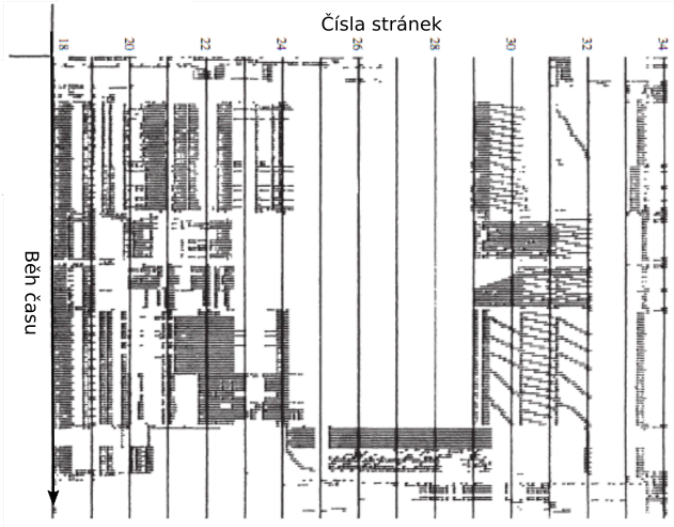
- P: Stránky procesu se zavádějí do reálné paměti podle potřeby
  S: Segmenty procesu se zavádějí do reálné paměti podle potřeby
- P: Zavedení stránky do reálné paměti může vyžádat výpis některé stránky na disk
  S: Zavedení segmentu do reálné paměti může vyžádat výpis jednoho nebo několika segmentů na disk

Důvody pro virtualizaci paměti

stránkováním/segmentací na žádost

- překlad adres LAP do adres FAP je třeba dělat co nejpozději, nejlépe dynamicky, při běhu procesu
- přidělování souvislých oblastí FAP procesu způsobuje vnější fragmentaci, stránkování tuto fragmentaci eliminuje
- stupeň multitaskingu je třeba udržovat co nejvyšší, rostou tím ale nároky procesů na prostor ve FAP
- existují procesy s paměťovými nároky přesahujícími dostupnou (možnou) velikost FAP
- při běhu procesu se nemůže ve FAP najednou uchovávat celý program a všechna potřebná data, zavádějí se dynamicky

Princip časoprostorové lokality procesu



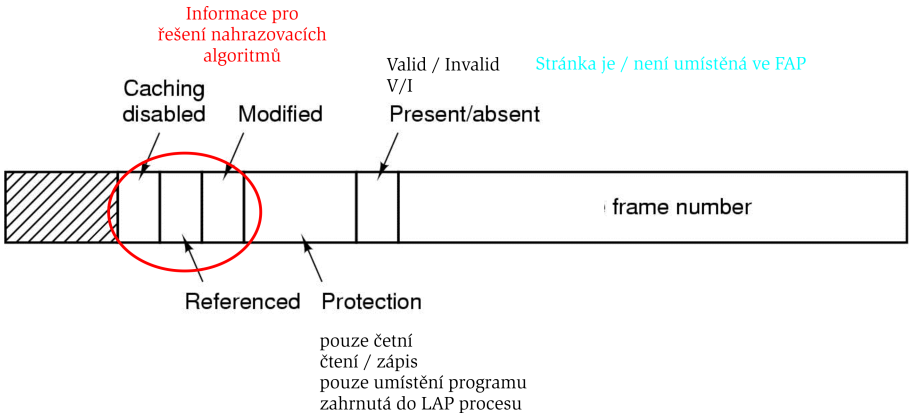
Princip časoprostorové lokality chování procesu

- odkazy na instrukce programu a na data mají tendenci tvořit shluky – časová lokalita a prostorová lokalita
  - ✓ provádění programu má, s výjimkou skoků a volání podprogramů, sekvenční charakter
  - ✓ programy mají tendenci zůstávat po jistou dobu v prostoru několika málo procedur, nelze jen volat a vracet se
  - ✓ většina iterativních konstruktů představuje malý počet často opakovaných instrukcí,
  - ✓ často zpracovávanou datovou strukturou je pole dat nebo posloupnost záznamů
- lze tudíž dělat inteligentní odhady o částech programu/dat potřebných v nejbližší budoucnosti ve FAP
- hlavní paměť (FAP) se může naplnit
  - ✓ něco umístit do FAP pak znamená nejdříve něco jiného z FAP odstranit

Běh procesů ve virtuální paměti

- respektuje se plná separace LAP a FAP
- velikost LAP je podstatně větší než velikost (dostupného) FAP
- OS při startu procesu zavede do FAP pouze malou část programu (LAP) s místem kam se iniciálně předává řízení
- po té dochází k dynamickým výměnám částí LAP ve FAP
  - ✓ po stránkách / po segmentech
  - ✓ ,,na žádost'', tj. až je jejich obsah referencovaný
- pro překlad logické adresy na fyzickou adresu se používá tabulka stránek/segmentů, *PT/ST*, *Page/Segment Table*
- každá položka v PT / ST obsahuje bit indikující ukončené zavedení odpovídající stránky (segmentu) do FAP – bit valid/invalid, V/I, present, . . .

Typický obsah položky PT při virtualizaci

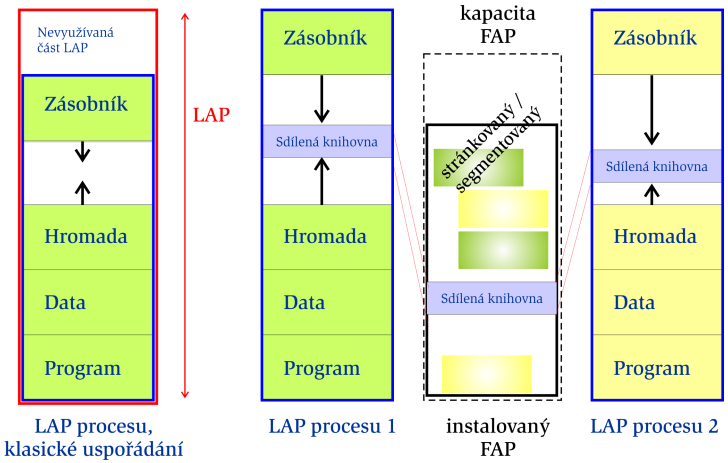


Běh procesů ve virtuální paměti

- Část (LAP) procesu umístěnou ve FAP nazýváme rezidentní množinou (*resident set*)
- Odkaz mimo rezidentní množinu způsobuje přerušení výpadkem stránky/segmentu (*page/segment fault*)
  - ✓ OS označí takový proces za čekající (na stránku/segment ve FAP)
  - ✓ OS spustí I/O operace a provede operace správy paměti nutné pro zavedení odkazované části LAP do FAP
  - ✓ mezitím, během jejího zavádění, běží jiný proces (jiné procesy)
  - ✓ po zavedení odkazované části se generuje I/O přerušení
  - ✓ OS čekající proces přemístí mezi připravené procesy
  - ✓ překlad adresy LAP na adresu FAP se dělá ,,indexováním'', transformacemi pomocí hodnot zapsaných do tabulky PT/ST vykonávanými pomocí hardware CPU (mikroprogramově)

LAP, LAP procesu

do FAP se „per partes" umísťuje pouze část LAP, kterou proces skutečně využívá



Virtuální paměť – vlastnosti, rekapitulace

- Ve FAP lze udržovat více procesů najednou, ve FAP nemusí být zobrazené celé LAP procesů
  - ✓ jednotlivé procesy okamžitě požadují méně prostoru ve FAP
  - ✓ čím více je procesů ve FAP, tím je větší pravděpodobnost, že stále bude alespoň jeden připravený – schopný běhu
  - ✓ provádí se méně IO operací, zavádí se pouze části LAP
  - ✓ dosahuje se rychlejších do reakcí i při více uživatelích
- Lze realizovat procesy požadující v sumě více paměti než je kapacita FAP
  - ✓ aniž se o řešení tohoto problému stará programátor / kompilátor
- jak se řeší uložení obrazu LAP (virtuální paměti) v externí paměti ?
  - ✓ nepoužívá se standardní systém souborů OS
  - ✓ používá se singulární metoda organizace souboru – větší bloky, speciální vyhledávání, . . .

Virtuální paměť – vlastnosti, rekapitulace

- Stránkování / segmentaci (prostou i na žádost) musí podporovat hardware správy paměti
  - ✓ žádost – dynamicky kontextově generovaná indikace nedostatku (tj. nepřítomnosti adresovaného objektu ve FAP) při získávání příští instrukce nebo při referenci dat
- OS musí být schopen organizovat tok stránek/segmentů mezi vnitřní a vnější pamětí

Příklady rozměrů stránek

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit words
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 Kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
IBM POWER	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes
Intel Itanium	**4 Kbytes to 256 Mbytes**
Intel core i7	**4 Kbytes to 1 Gbyte**

Fetch Policy – kdy stránku zavádět?

- Dvě politiky
- stránkování na žádost (*Demand paging*)
  - ✓ stránka se zobrazuje do FAP při odkazu na ni, pokud ve FAP není již zobrazena
  - ✓ způsobuje počáteční shluky výpadků stránek
- předstránkování (*Prepaging*)
  - ✓ umístění sousedních stránek v LAP v obrazu LAP na vnější paměti bývá rovněž blízké
  - ✓ platí princip časo-prostorové lokality, sousední stránky v LAP se s velkou pravděpodobnosti používají v blízkém časoprostoru
  - ✓ zavádí se více sousedních stránek než se žádá
  - ✓ předstránkování je vhodná politika zvláště při inicializaci procesu

Stránkování na žádost

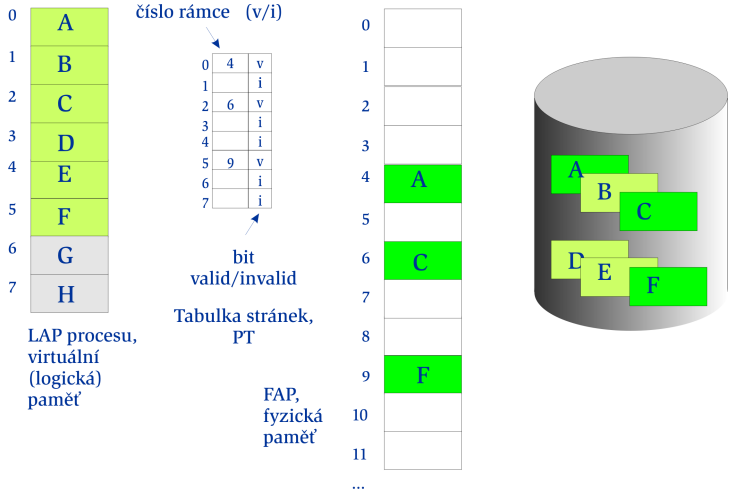
- Stránka se zavádí do FAP když je (pravděpodobně) potřebná
- přínosy
  - ✓ provádí se méně IO operací
  - ✓ vznikají menší požadavky na paměť
  - ✓ rychlejší interakce – může pracovat více uživatelů
- Kdy je stránka potřebná?
  - ✓ byla odkazovaná, referencovaná

Stránkování na žádost

- Kdy je reference stránky legální ?
  - ✓ legální reference stránky
    - – nachází se ve FAP, procesu / CPU je dostupná
    - – logická adresa se mikroprogramem přeloží na fyzickou adresu
  - ✓ nelegální reference stránky může být dvojího druhu
    - – chybná reference stránky, porušení ochran,
      odkaz na v procesu nedefinovaný LAP, . . . → *abort* procesu
    - – reference stránky procesu neumístěné ve FAP
      její zavedení do FAP zprostředkovává OS

Stránkování na žádost, bit Valid/Invalid

všechny stránky nemusí být ve FAP současně



Stránkování na žádost, bit Valid/Invalid, výpadek stránky

- S každým řádkem PT je spojen bit valid/invalid, V/I
  - ✓ 1: stránka je ve FAP,
  - ✓ 0: stránka není ve FAP
- je-li při překladu logické adresy na fyzickou adresu V/I = 0, generuje se přerušení typu výpadek stránky, *page fault*
  - ✓ OS si zjistí
    - jedná se chybnou referenci → OS spustí abort procesu
    - jedná se o referenci stránky procesu chybějící ve FAP → OS spustí její zavedení
      * proces umístí mezi čekající
      * nalezne prázdný rámec ve FAP
      * zavede z obrazu LAP ve vnější paměti chybějící stránku do tohoto rámce
      * zkoriguje PT, nastaví v ní číslo rámce bit valid (= 1)
      * proces čekající na stránku umístí mezi připravené procesy tak, aby se zopakovalo provedení instrukce, která způsobila výpadek stránky

Placement policy – kam stránku zavádět?

- kam stránku / segment umístit ve FAP?
- stránkování
  - ✓ nemá význam, pro umístění stránky se volí libovolný volný rámec
- segmentace
  - ✓ politiky first-fit, best-fit, worst-fit, next-fit, ...
- kombinace segmentování + stránkování, stránkované segmety
  - ✓ nemá význam, pro umístění stránky se volí libovolný volný rámec

Stránkování na žádost, výpadek stránky, není volný rámec

- Náhrada stránky v rámci požadovanou stránkou
  - ✓ oběť – stránka ve FAP dále už (pravděpodobně) nepotřebná
  - ✓ pokud se do ní nezapsalo, její kopie na disku je, bez problému se přepíše ve FAP
  - ✓ pokud se do ní zapsalo, nejprve se přepíše její obraz v obrazu LAP ve vnější paměti
- Je potřebný algoritmus hledání oběti a řešení náhrady
- kritérium optimality
  - ✓ optimální algoritmus způsobuje nejméně výpadků stránek
- Některé stránky mohou být do FAP za život procesu zaváděny opakovaně
- Některé rámce jsou dočasně chráněné před uvolněním (I/O buffery, ...)

Replacement Policy – kterou stránku nahradit?

- Politika nahrazování
  - ✓ uplatňuje se v okamžiku, kdy je reálná paměť (FAP) plná
  - ✓ není dostupný neobsazený rámec
- Kterou stránku ,,obětovat" a vyhodit z FAP?
  - ✓ také Politika výběru oběti řízená algoritmem pro výběr oběti
- určení oběti
  - ✓ ideální oběť – stránka, která již nebude odkazovaná, :-((
  - ✓ volbu mj. ovlivňuje politika přidělování prostoru ve FAP procesu
  - ✓ intro- množina možných obětí
    - oběti lze hledat jen mezi stránkami procesu, který vyvolal výpadek stránky
  - ✓ extra- množina možných obětí
    - oběti lze hledat i mezi stránkami libovolných existujících procesů

Algoritmus nahrazování stránek

Řeší OS po přerušení výpadkem stránky:

1. Proces dej spát, dokud se nezíská referencovaná stránka
2. Nalezni umístění požadované stránky na disku
3. Nalezni volný rámec:
   – existuje – použije se,
   – neexistuje – start výběru oběti a uvolnění rámce
4. Načti požadovanou stránku do volného / uvolněného rámce
5. Koriguj tabulku stránek a tabulku rámců
6. Restart procesu (via dispečer) od adresy instrukce, která způsobila výpadek stránky

Výkon stránkování na žádost

- Skutečná doba přístupu do paměti, *Effective Access Time* (*EAT*)
  - ✓ p – pravděpodobnost výpadku stránky, pak *EAT* =
    (1 – p) x *doba přístupu do paměti* + 200 ns
    p x (*režie přerušení výpadkem stránky* + 1–100 μs
    [*doba výpisu stránky*] + cca 8 ms
    doba načtení stránky + cca 8 ms
    režie restartu instrukce) 1–100 μs
- tempo výpadků stránek určuje pravděpodobnost výpadku stránky $0 \leq p \leq 1.0$
  - ✓ když $p = 0$, pak k výpadkům nedochází, LAP procesu se nachází v FAP
  - ✓ když $p = 1$, každá reference způsobí výpadek
- na disk se vypisují oběti jen když byly ve FAP modifikovány
  - ✓ v PT se udržuje *modify* (*dirty*) *bit*
  - ✓ = 1 při zápisu do stránky, nastaví ho příslušný mikroprogram

Výkon stránkování na žádost, ilustrativní příklad

- *EAT* = (1 – p) x 200 + p x (8 000 000) =
  = 7 999 800 p + 200
  - ✓ pokud výpadek stránky nastane 1-krát za 1000 referencí, EAT = 8,2 μs
  - ✓ přičemž doba přístupu do paměti = 0,2 μs
  - ✓ výkon počítače 40-krát klesl
- nemá-li se výkon snížit víc než o 10%:
  - ✓ $220 > 200 + 7\,999\,800p$, tj. $p < 0,000\,002\,5$
  - ✓ výpadek stránky nesmí nastat častěji než 1-krát za 399 990 referencí

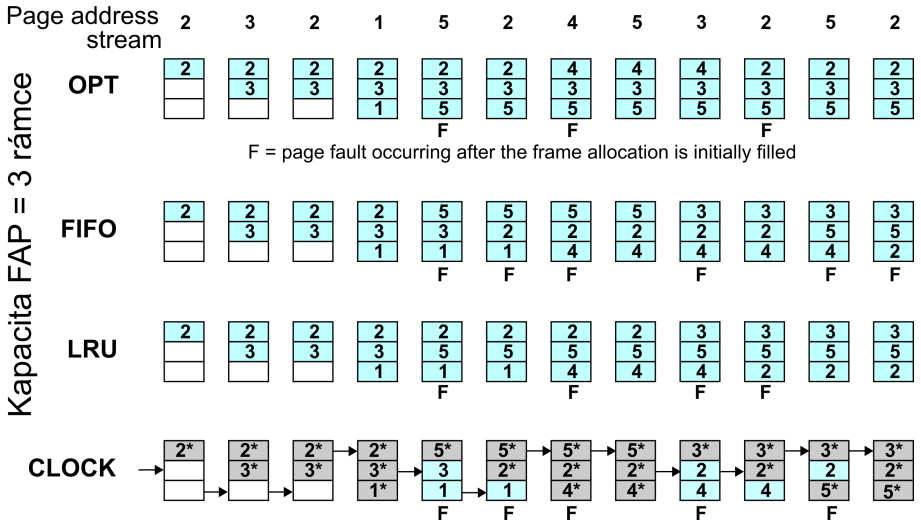
Algoritmy výběru oběti

- Optimal (OPT)
  - ✓ obětí je stránka s nejzazší referencí
- First-in-first-out (FIFO)
  - ✓ cyklický výběr rámců s obětí
- Least recently used (LRU)
  - ✓ obětí je stránka nejdéle nereferencovaná
- Clock, příp. „druhá šance“
  - ✓ cyklický výběr rámců s obětí s možností získat „život“ dočasně opravňující vybranou oběť po jistou dobu ve FAP přežít
  - ✓ implementace: FIFO + dodatečné nástroje
  - ✓ aproximace algoritmu LRU

Algoritmy výběru oběti

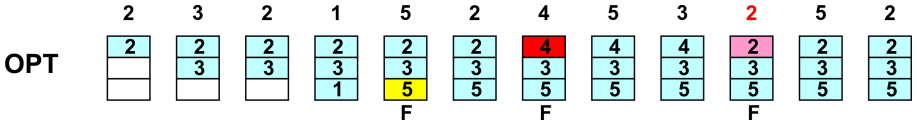
- Kritérium optimality – dosažení nejmenší možné pravděpodobnosti výpadku stránky
- Při ilustraci algoritmů použijeme shodnou posloupnost referencí stránek – 2 3 2 1 5 2 4 5 3 2 5 2
- Použitý FAP bude obsahovat 3 rámce
- Obecná pravda – se vzrůstem počtu rámců přidělených procesu počet výpadků procesem generovaných výpadků stránek klesá
  - ✓ celý program vč. dat ve FAP znamená, že se negeneruje žádný výpadek stránky

Příklad obsazování FAP probíranými nahrazovacími algoritmy



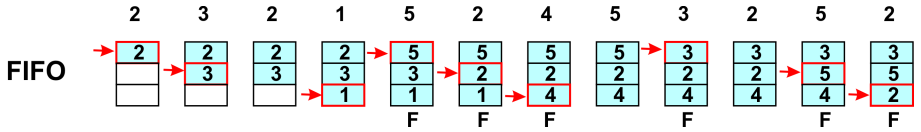
OPT, optimální nahrazování

- oběť, musí platit = příští odkaz na oběť je nejpozdější odkaz ze všech následných odkazů na stránky
- algoritmus OPT generuje nejmenší počet výpadků stránek
- algoritmus OPT je obecně neimplementovatelný, neznáme budoucnost, používá se jako srovnávací normál



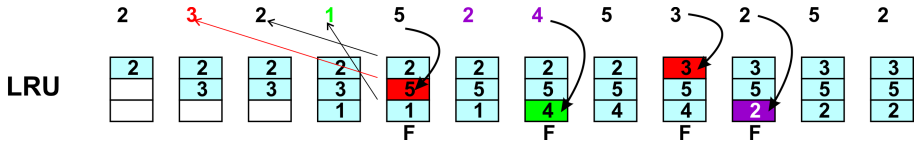
FIFO nahrazování

- oběť = stránka nejdéle zobrazená ve FAP
- často používané stránky mnohdy jsou ty nejstarší stránky
  - ✓ pravděpodobnost brzkého výpadku takové stránky je vysoká
- jednoduchá implementace
  - ✓ Rámce přidělené procesu jsou organizovány do cyklického bufferu, když je buffer plný, nahrazuje se nejstarší stránka



LRU nahrazování, LRU – Least Receantly Used

- oběť = ve FAP nejdéle neodkazovaná stránka
- princip lokality: pravděpodobnost odkazu na ni je malá
- i když má kvalitu blízkou optimální strategii,
  používá se zřídka,
  z důvodu obtížné a neefektivní implementace



LRU nahrazování, LRU – Least Receantly Used

- triviální implementace politiky LRU
  - ✓ V položkách PT se udržují (HW) čítače běhu času (+1 po Δ t)
  - ✓ Každá referencovaná stránka se v položce PT označí nulovou dobou od posledního referencování
  - ✓ obětí je stránka s nejdelší dobou nereferencování
  - ✓ negativa:
    - konstrukčně velmi drahé řešení
    - čítač má konečnou kapacitu – problém jeho přetečení
- náhrada řízená hodinami
  - ✓ ke každé stránce (rámci) je přiřazen jeden registr
  - ✓ při referenci stránky v rámci se do registru okopírují hodiny
  - ✓ Při hledání oběti se vybírá nejdéle nereferencovaná stránka

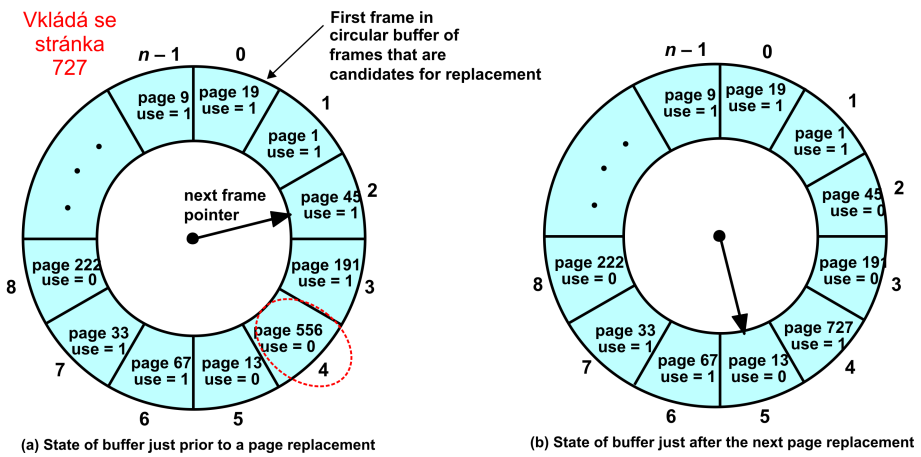
Algoritmy aproximující LRU

- bit reference
  - ✓ spojen s každou stránkou v PT, počátečně (inicializuje OS) = 0
  - ✓ když je stránka referencovaná, = 1 (mikroprogramově)
- historie referencí pomocí bitu reference
  - ✓ slabika spojená s každou stránkou v PT,
  - ✓ počátečně (inicializuje OS) = 00000000_2
  - ✓ bit 0 je bit reference
  - ✓ obsah slabiky se periodicky posouvá vlevo
  - ✓ obsah slabiky ilustruje 8 kroků historie
    - zda v periodě odpovídající kroku byla stránka referencovaná
    - 00000000_2 po 8 krocích – stránka nebyla vůbec referencovaná
    - 11111111_2 po 8 krocích – stránka byla referencovaná v každém kroku

Algoritmy aproximující LRU

- Druhá šance
  - ✓ Využívá opět *bit reference*
  - ✓ základ algoritmu – náhrada hodinami (časovým pořadím, FIFO)
  - ✓ Je-li stránka vybraná jako oběť podle principu hodin/FIFO a má-li bit reference = 1, pak:
    - bit reference se nastaví na 0
    - stránka se nechá v paměti
    - hledá se další oběť (podle hodin/FIFO) a podle stejných pravidel
  - ✓ de facto FIFO, z výběru oběti se vynechává alespoň jednou odkázaná stránka od posledního výběru

Algoritmus ,,Druhá šance"

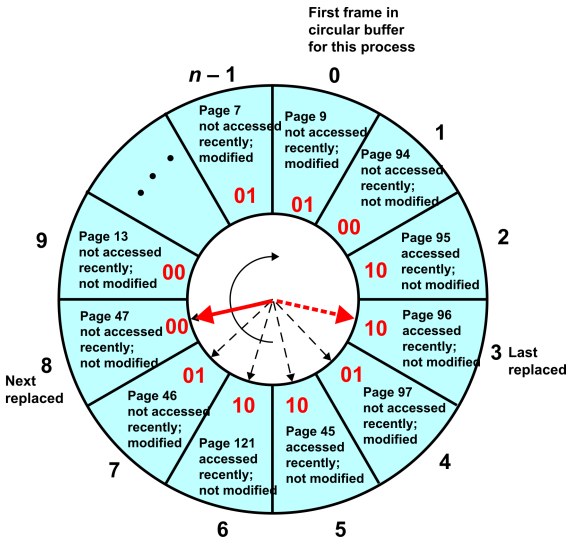


Algoritmy aproximující LRU

- modifikovaná (vylepšená) ,,Druhá šance"
  - ✓ *bit reference* je v PT doplněný o *bit modifikace*
  - ✓ pořadí výběru (cyklicky, počínaje tam, kdy výběr naposledy skončil):
    1. 00, nereferencovaná, nemodifikovaná – stačí ji v rámci přepsat a pokud hledání selže, hledá se
    2. 10, referencovaná, nemodifikovaná – stačí ji v rámci přepsat

       nahozené bity reference u vynechávaných rámců se nulují, všechny prohlížené rámce budou mít bit reference 0 a pokud hledání selže, opakuje se krok 1, příp. i krok 2
  - ✓ dokud není modifikovaná stránka vypsaná na disk, nelze ji určit jako oběť
  - ✓ používala dřívější verze *Macintosh virtual memory scheme*

Algoritmus ,,Modifikovaná druhá šance"



Další experimentální algoritmy – čítací algoritmy

- udržuje se čítače počtů referencí udělaných na každou stránku
- Algoritmus LFU, *Least Frequently Used*:
  - ✓ aktivní stránka je hodně referencovaná
  - ✓ obětí je stránka s nejmenší hodnotou čítače, je málo referencovaná
  - ✓ problém:
    - inicializační část programu bude mockrát použita a pak už vůbec ne, ale bude dlouho setrvávat v reálné paměti
  - ✓ řešení: čítač lze periodicky posouvat doprava (snižovat jeho hodnotu)
- Algoritmus MFU, *Most Frequently Used*:
  - ✓ obětí je stránka s největší hodnotou čítače, možná je ve FAP už ,,dlouho"
  - ✓ stránka s nejmenším čítačem asi byla právě umístěna do paměti a nestačila být ještě referencovaná, tak ji nevyhazujme

Přidělování počtu rámců procesům a politiky výběru obětí

- Každý proces potřebuje jistý minimální počet stránek ve FAP
- Příklad
  - ✓ IBM 370 – potřeboval 6 stránek pro zvládnutí instrukce MOVE:
  - ✓ instrukce měla délku 6 B
    mohla ležet až ve 2 stránkách, 2 stránky pro *odkud*, 2 stránky pro *kam*
- hlavní principy přidělování rámců procesům
  - ✓ pevné přidělování počtu rámců procesům
    – stejný počet nebo proporcionální počty (velikostem LAP procesů)
  - ✓ prioritní přidělování – podle (dynamické) priority procesu
- hlavní politiky výběru oběti
  - ✓ lokální výběr, ze stránek procesu, který způsobil výpadek stránky
  - ✓ globální výběr, ze stránek kteréhokoliv procesu

Přidělování počtu rámců procesům a politiky výběru obětí

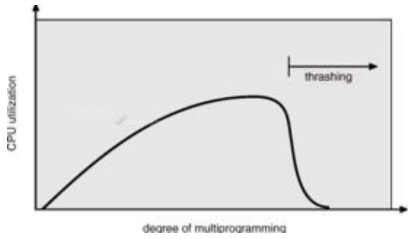
- pevný počet rámců, lokální náhrady
  - ✓ alokace počtu rámců podle typu aplikace
  - ✓ důsledek podhodnocení / nadhodnocení –
    poddimenzování – zvýší se frekvence výpadků stránek
    předimenzování – sníží se možný stupeň multiprogramování
- pevný počet rámců, globální náhrady
  - ✓ nerealizovatelné, nesmyslné řešení
- proměnný počet rámců, globální náhrady
  - ✓ nejsnazší implementace, klasická politika v mnoha OS, Unix SVR4, ...
  - ✓ nebezpečí ,,výprasku", Thrashing,
    mnoho procesů s malým počtem přidělených rámců,
    mnoho výpadků, čekání na I/O na disky. roste multitasking, ...
- proměnný počet rámců, lokální náhrady
  - ✓ tzv. pracovní množiny, později detaily

Politika čištění stránek

- protipól ,,fetch"politiky
- čištění = vypisování modifikovaných stránek na disk
- čisté stránky = shodné s jejich obrazy na disku
- varianty
  - ✓ čištění na žádost –
    když byla modifikovaná stránka vybrána jako oběť
  - ✓ precleaning –
    systematický výpis modifikovaných stránek,
    lze řešit dávkově na pozadí běhu procesů

Výprask, Thrashing

- Jestliže proces nemá v paměti dost stránek, generuje výpadky stránek velmi rychle a tudíž
  - ✓ dochází k nízkému využívání CPU aplikacemi
  - ✓ OS má dojem, že má zvýšit stupeň multiprogramování,
    protože stále se čeká na konec I/O operace
  - ✓ a tak se dodávají další procesy do systému, ...
- Thrashing – procesor nedělá nic jiného než výměny stránek

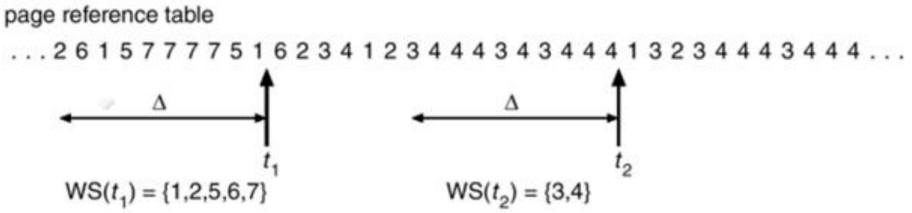


Model pracovní množiny, Working-Set

- d – okno pracovní množiny,
  zvolený pevný počet referencí stránek, např. 10 000
- WS_i (*working set* i), pracovní množina procesu P_i,
  množina referencovaných stránek v posledním oknu d
  (pracovní množina se v čase mění)
  - ✓ pracovní množina je aproximace časo-prostorové lokality procesu
  - ✓ bude-li d malé, nepostihne celou potřebnou pracovní množinu
  - ✓ bude-li d velké, postihne více než potřebnou pracovní množinu
- Pokud suma všech WS_i převýší kapacitu dostupné reálné paměti, vzniká ,,výprask"
- Ochrana před vznikem ,,výprasku"
  - ✓ např. jeden proces se pozastaví

Model pracovní množiny, Working-Set, 2

- Okno $\Delta = 10$
- Pracovní množiny v okamžicích t_1 a t_2

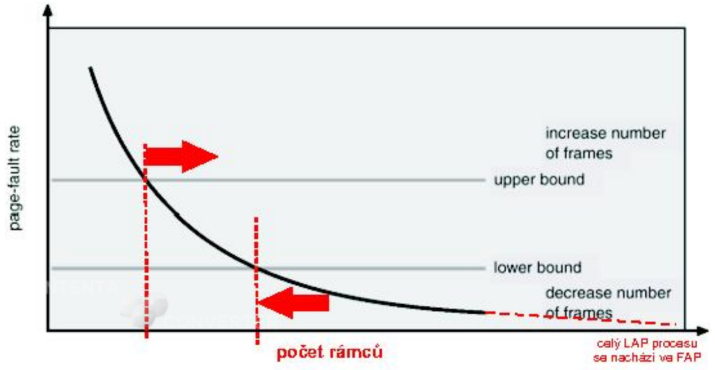


Sledování Working Set

- Aproximace pomocí intervalového časovače a bitu reference
- Příklad:
  - ✓ Nechť okno pracovní množiny $d = 10\,000$ referencí
  - ✓ Nechť časovač přerušuje po 5 000 časových jednotkách
  - ✓ V paměti se v PT udržují o každé stránce 2 bity (buffer a reference)
  - ✓ Kdykoliv časovač přeruší,
    okopíruje se bit reference do bufferu a nastaví se na 0
  - ✓ Kdykoliv se referencuje daná stránka,
    bit reference se mikroprogramově nastaví na 1
  - ✓ Je-li alespoň jeden z obou bitů = 1, stránka je ve WS
- Zpřesnění měření jen za cenu vyšší režie
  - ✓ např. 10 bitů místo 2 a přerušování po 1 000 časových jednotkách

Frekvence výpadků stránek

- Akceptovatelná frekvence výpadků stránek
  - ✓ je-li frekvence výpadků nízká, procesu lze rámce rámce odebírat
    – oběti se vybírají mezi jeho stránkami
  - ✓ je-li frekvence výpadků vysoká, proces potřebuje získat další rámce



Problém velikosti PT, řešení Pentium

- 2-úrovňové hierarchické stránkování
- PT se umísťuje ve virtuální paměti a stránkuje se
- formáty:
  - ✓ číslo stránky je děleno do dvou čísel p1 a p2
  - ✓ p1 indexuje adresář umístěný v hlavní paměti s odkazy na stránky obsahující vlastní tabulky stránek
  - ✓ adresář se neodkládá na disk
  - ✓ tabulky stránek (2. úrovně) lze odkládat

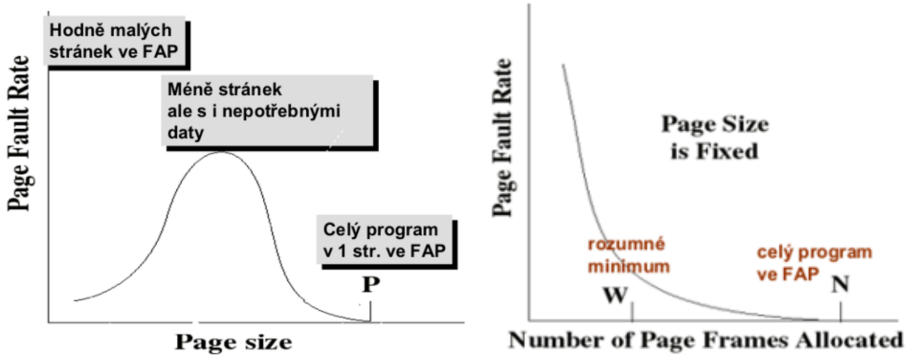
Problém velikostí stránek

- Velmi malý rozměr stránky
  - ✓ ve stránkách není nic, co není nutné z hlediska časové a prostorové lokality
  - ✓ malý počet výpadků stránek
- čím je stránka menší
  - ✓ tím je menší vnitřní fragmentace, ale diskové operace jsou méně efektivní
  - ✓ tím je větší počet stránek, takže je delší PT
- pokud se PT umístí přímo ve FAP, způsobuje velké čerpání prostoru FAP
- umístění PT ve virtuální paměti způsobuje dvojnásobný počet přerušení ,,page fault"

Problém velikostí stránek, 2

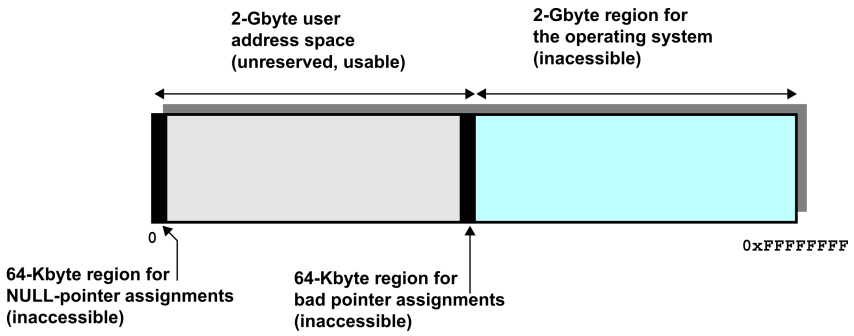
- čím je délka stránky větší
  - ✓ tím je ve stránce je obsaženo více dat než je nutné z hlediska časové a prostorové lokality
  - ✓ a protože FAP má konečnou kapacitu, vzniká větší počet výpadků stránek
- Pokud délka stránky se rovná délce programu, vše je ve FAP, není potřeba žádná virtualizace
- Obvyklé délky stránek – 1 KB až 4 KB
- Některé procesory podporují možnost používání několika rozměrů stránek, např. Pentium - 4KB nebo 4MB

Problém velikostí stránek, 3



Windows – správa paměti, LAP (virtuální adresový prostor)

- 32-bitová varianta



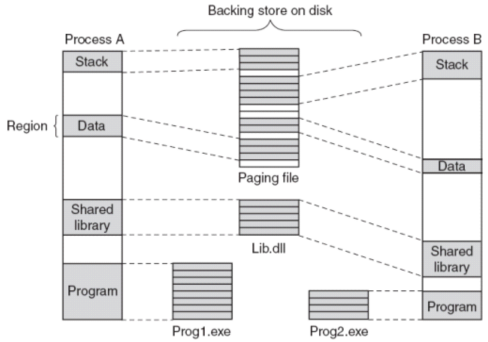
- 64-bitová varianta – uživatelská oblast 8 TB, celkem 16 TB

Windows – Správa paměti, LAP (Virtuální adresový prostor)

- Horních a dolních 64 KB LAP se do FAP nezobrazuje
- 64 KB až 2 GB: program a data uživatele
- 2 GB až 4 GB –64 KB virtuální paměť jádra OS obsahující
  - ✓ program, data, stránkovací tabulkuy, banky stránek, . . .
- Virtuální paměť jádra OS je ve FAP sdílená všemi procesy a je dostupná pouze při běhu v režimu jádra
- Na procesorech x86 a x64 je virtuální paměť stránkovaná
  - ✓ délka stránky 4 KB, nepoužívá se segmentace
- Stránky LAP zahrnuté do procesu – *committed pages*
- Stránky LAP nezahrnuté do procesu – *invalid pages*

Windows – Správa paměti, obraz LAP na disku

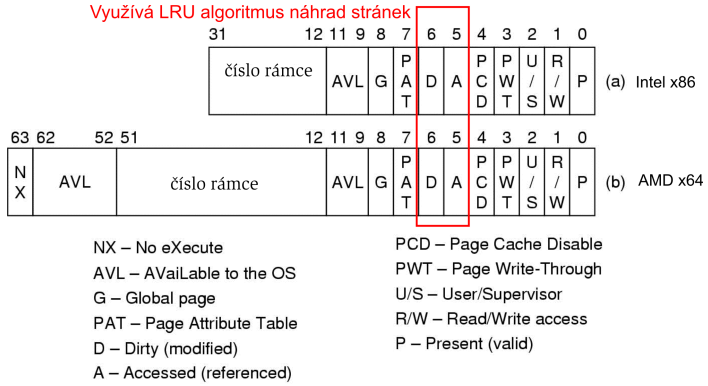
- Dynamicky používané zahrnuté stránky jsou uchovávané na disku v spec. souborech – pagefiles
- Texty programů jsou uchovávané na disku v konkrétních souborech



Windows – Správa paměti, práce s rámci

- Jakmile vlákno stránkováním na žádost umístí stránku do volného rámce FAP, rámec se stává pohotovostní (*standby*)
  - ✓ obsahuje nemodifikovanou stránku, kterou lze případně pouze přepsat ve FAP jinou stránkou
- Jakmile vlákno stránku v rámci FAP modifikuje, rámec se stává modifikovaným rámcem
  - ✓ obsahuje modifikovanou stránku, kterou je nutné při přepisu jinou stránkou nejprve vypsat do obrazu LAP na disku
- Správa paměti udržuje 5 seznamů rámců
  - ✓ volné rámce, *free*, nealokované, lze do nich kdykoliv umístit stránku
  - ✓ čisté rámce, *zeroed*, volné rámce, obsahující pouze 0
  - ✓ pohotovostní rámce, *standby*, alokované nemodifikované rámce
  - ✓ modifikované rámce, *modified*, alokované modifikované rámce
  - ✓ rámce vykazující chybu paměti, *bad*

Windows – Záznam v tabulce stránek, *page table entry* (PTE)

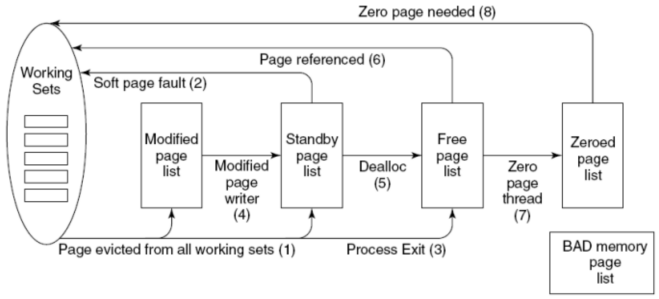


- Algorimus náhrad stránek – LRU, *Least Recently Used*
  - ✓ stránky, které se nejdéle nepoužily se pravděpod. nebudou používat
  - ✓ Bit A nastavuje hradware při zpřístupnění stránky ve FAP
  - ✓ Bit D nastavuje hardware při modifikaci stránky ve FAP, nemodifikovaná stránka se nemusí vypisovat při náhradě na disk

Windows – Výpadek stránky, pracovní množina stránek

- Vlákno referencuje
  - ✓ nezahrnutou stránku – chyba programu, abort
  - ✓ zahrnutou stránku chybně – chyba programu, abort (zápis do RO stránky, . . .)
  - ✓ zahrnutou stránku, která dosud není mapovaná do FAP – výpadek stránky, *Page Fault*
- Pro každý proces se udržuje ve FAP jistý počet stránek – *Working set*, WD, pracovní množina
  - ✓ WS má definované minimum a maximum velikosti (např. 20 – 345 stránek)
- proces začíná s minimální WS, WS se může zvětšovat až do maxima
  - ✓ pokud je volno ve FAP, maximum WS se může překročit

Windows – Algoritmus náhrad stránek, práce s rámci



- Manažer WS periodicky prohlíží všechny WS
  - ✓ Je hodně volné paměti FAP – počítá stáří stránek resetováním bitu A
  - ✓ FAP je téměř vyčerpaný – fixuje WS a nejstarší stránky nahrazuje novými, jsou-li žádané
  - ✓ ve FAP chybí místo – WS se zmenšují odstraňováním nejstarších stránek z FAP

Windows – Algoritmus náhrad stránek, práce s rámci

- Odstraňování stránek z FAP
  - ✓ když se procesu odstraňuje stránka z WS (1) její rámec se dává mezi modifikované či pohotovostní rámce
    - tyto stránky zůstávají ve FAP a na případnou následnou žádost se vrací do WS (2)
  - ✓ když proces končí (3) jeho rámce se dávají do seznamu volných rámců
  - ✓ modifikované rámce manažer paměti periodicky vypisuje na disk a zařazuje mezi pohotovostní (4)
  - ✓ proces může stránky v pohotovostních rámcích kdykoliv zrušit (5)
  - ✓ Kdykoli ,,není co dělat", volné rámce čistič rámců čistí, nuluje (7)
- nově požadované stránky se umísťují
  - ✓ do volných rámců (6)
  - ✓ pokud to vyžaduje bezpečnostní politika, pak do čistých rámců (8)