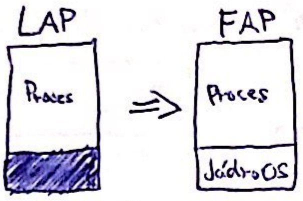
39. VIRTUALIZACE PAMĚTI, STÁNKOVÁNÍ A NAHRAZOVÁNÍ

- procesy pracují nad svým virtuálním adresním prostorem který mají při svém vytvoření alokovaný (mají jej celý pro sebe a nemusí řešit vzájemné vyloučení)
- tento prostor může být teoreticky nekonečný a obsahuje zásobník, kód programu atd..
  => jedná se o Logický adresný prostor (LAP)
- systém reálně obsluhuje pouze fyzickou paměť jako třeba RAM a tomu se říká Fyzický adresný prostor (FAP)
- pokud proces přistoupí na nějaké místo svého LAP a chce nad ním vykonávat operace tak se tento kus LAP nebo celý LAP musí nějakým způsobem namapovat do toho FAP protože CPU zná pouze FAP
- jsou různé způsoby jak je LAP organizován

Organizace LAP

1) Jeden souvislý úsek paměti

- LAP = FAP
- LAP je jeden souvislý a nijak nedělitelný úsek paměti který se celý mapuje 1:1 na FAP -> každá adresa v LAP = adresa v FAP a nemusí se dělat složité mapování
- část paměti má jádro OS a zbytek má ten jeden program -> lze pouze v mono-programových systémech (MSDOS)
- není nutné řešit ochranu paměti před jinými procesy, je pouze nutné chránit paměť jádra



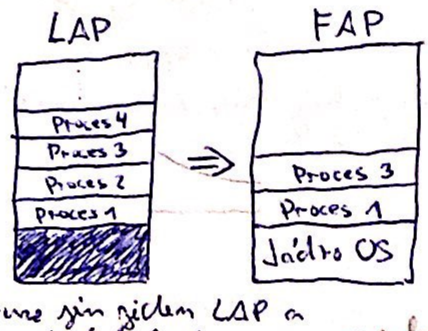
- v podstatě to => není ani mapování protože to je 1:1 a jedna adresa v LAP je ta samá adresa v FAP
- je jenom jeden LAP

2) Společný adresný prostor

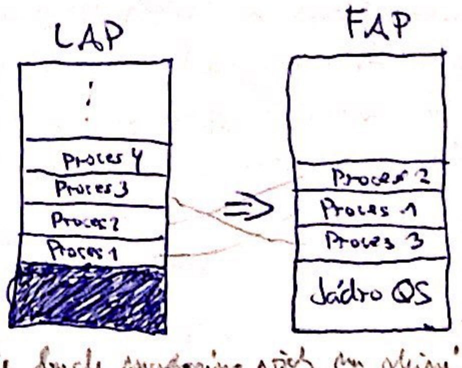
- je opět pouze jeden LAP (jako v předchozím bodě) ale zde už se nejedná o monoprogramové systémy a je zde více procesů a ty spolu ten adresný prostor sdílí
- jejich adresný prostor (LAP) je oddělen pouze offsety
- je nutné řešit ochranu jak paměti jádra OS tak ochranu paměti procesů aby si do ní nezasahovaly "nelezly"

=> všechny procesy mají jeden společný adresný prostor / nevím jestli to platí

- proces nemusí být vždy mapován na stejnou adresu v FAP (oproti předchozímu bodu) a tak se na to nesmí spoléhat



(a jindy pouze

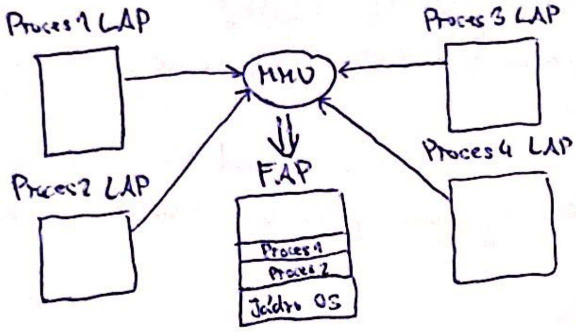


) / nevím jestli to platí

- pouze jeden LAP a mapování 1:1
- nelze spoléhat na to, že bude namapováno vždy na stejné místo

3) Oddělený adresný prostor

- každý proces má kompletní svůj vlastní adresný prostor odpovídající tomu FAP
- každý proces svůj LAP a nemusí se tedy řešit ochrana paměti procesů v rámci jejich LAP – nemohou si je přepisovat
- musí se ale provádět mapování všech LAP na FAP – je to složitější než u předchozích bodů
- mapování provádí MMU (Memory Mapping Unit)



- mapují se části LAP na které proces přistupuje do FAP a ostatní mohou být třeba na HDD ve swapu
- FAP je třeba RAM
- proces se nemusí znova spoléhat, že bude mapováno pořád na stejné místo v FAP

↳ je více LAP a tak to není mapování 1:1

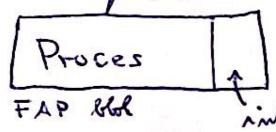
- jak mapovat LAP na FAP?

Mapování LAP na FAP – jak to ta MMU dělá

– dělá se to pomocí těchto 3. variant

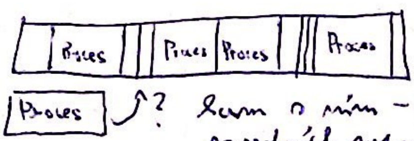
1) Bloky pevné délky

- procesy mají danou pevnou délku
- bloky FAP mají danou také pevnou délku
- procesy si vybírají bloky FAP podle velikosti a u jednotlivých bloků se tvoří fronty procesů – procesu se přidělí nejmenší dostatečný blok
- procesy mají tedy také pevně danou velikost
- je zde interní fragmentace v rámci FAP bloku – blok je větší než proces



2) Bloky proměnné délky

- FAP mění velikost svých bloků dynamicky tak aby seděly procesům
- bloky se dělí na menší když přijde menší proces a spojují na větší když přijde větší
- problémem je externí fragmentace – zbývá dostatek paměti po tom dynamickém dělení a spojování ale ta paměť není souvislá a tak je nelze využít když přijde další proces
- řešení
  - spojování sousedních, ale nesousední nespojíme a problém zůstává :(
  - setřásání – setřese se a spojí



tam se nevejde – pouhé spojení sousedních nepomůže ⇒ setřesení

1) i 2) ⇒ 1 proces = 1 blok

3) Segmentace

- 1 program/proces se rozdělí na více bloků – 1 blok je kód, druhý zásobník, další data, ...
- je dána adresa segmentu která určuje logickou adresu (LAP) a offset v ní
- tabulka segmentů – obsahuje fyzickou adresu segmentu a limit – tedy max. možnou velikost segmentu
- pořád je problém s externí fragmentací
- tabulka je buď
  - globální – společná pro všechny procesy
  - lokální – pro každý proces zvlášť

4) Stránkování

5) Segmentace se stránkováním – jednotlivé segmenty jsou stránkovány

Stránkování

- logický adresový prostor procesů (LAP) je rozčleněn na stránky
- fyzický adresový prostor je rozdělen na rámce
- rámce a stránky mají stejnou velikost
- tabulka stránek obsahuje mapování stránek na rámce
- logická adresa obsahuje číslo stránky a offset v ní
- tabulka stránek obsahuje (je) asociativní pole kde klíčem je číslo stránky a hodnotou je struktura obsahující číslo rámce, a flagy jako příznak platnosti byla stránka nedávno použita (pro nahrazovací algoritmy) nebo zda je v paměti nebo je odložena na swapu atd.

- rychlost – přístup se dívat do tabulky stránek a překládat logickou adresu (stránku) na fyzickou adresu (rámec) je neefektivní
  ⇒ používá se TLB (Translation Look-Aside Buffer) což je rychlá cache která uchovává poslední mapování
  - při přepnutí kontextu se musí vyprázdnit – jiný proces bude mít jiné mapování
- velikost – moc velká tabulka stránek zabírá moc místa v paměti
  ⇒ používají se víceúrovňové tabulky stránek
  - logická adresa obsahuje indexy do jednotlivých úrovní tabulky stránek
  - podíváme se do 1. úrovně na daný index z LA a tam najdeme bázi tabulky 2. úrovně a podíváme se na index do tabulky druhé úrovně z LA atd...
  - v té nejnižší úrovni je to číslo rámce

- jestliže se použije stránka která není v paměti ale je na swapu na HDD tak nastane **výpadek stránky**
  - a cílová stránka se musí zavést do paměti a pokud v ní není místo tak se musí nějaká stávající vyhodit ⇒ dělají nahrazovací algoritmy
- stránkovací algoritmus určí kdy se která stránka zavede do paměti a která se odebere (pomocí nahrazovacího algoritmu) tak aby nedocházelo k výpadkům a nebo co nejméně
- zavádění stránek procesu:
  - buď všechny – celý LAP procesu (nepoužívá se)
  - nebo jen ta jedna použitá (nepoužívá se)
  - ta použitá a její okolí – je pravděpodobné že další stránky bude proces chtít z toho okolí – cyklus, pole,...
- stránka co se nahrazuje se odloží na swap na HDD

Nahrazovací algoritmy

- rozhoduje jaká stránka bude odebrána z paměti (FAP) a odložena na swap při zavádění nové stránky – při výpadku stránky
- lokální – kterou nahradit vybere jen ze stránek toho procesu který měl výpadek stránky
- globální – kterou nahradit vybírá ze všech stránek v paměti
- pevný počet rámců – rámců je ve FAP pořád stejně a tak se na FAP může mapovat pořád stejné stránek
- proměnný počet rámců – počet stránek které mohou být mapovány se chvíli od chvíle liší
- časová lokalita – stránky chtějí se opakovat (cyklus)
- prostorová lokalita – stránky chtějí jsou blízko sebe (pole,..)

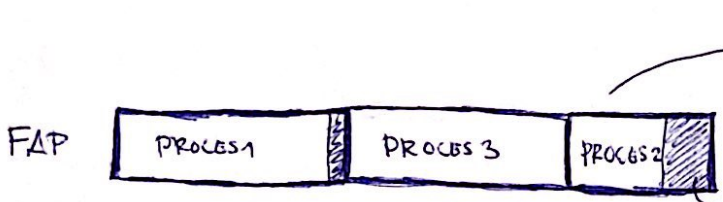
Nahrazovací alg. s pevným počtem rámců

- Optimální
  - nedocházelo by vůbec k výpadkům ale musel by znát budoucnost a to je nereálné
  - vybíral by stránku pro nahrazení která bude nejdéle nepoužitá v budoucnu
- FIFO
  - odebere tu co tam je nejdéle
  - možná je tam ale nejdéle protože se furt používá (často) :(
- LIFO
  - odebere poslední vloženou
  - u časové lokality a cyklů bude třeba často docházet k výpadkům :(
- LRU (Least Recently Used)
  - vybírá nejdéle nepoužitou
  - používá se zásobník a když se použije tak se dá nahoru a odebírá se zespoda
- LFU (Least Frequently Used)
  - odebere stránka co je používána nejméně často – počítadlo použití
  - může odebrat stránku která se právě navedla a bude se teď používat často :(
- SECOND CHANCE
  - poprvé se označí stránka že se má odebrat tak se jí pouze nastaví flag a až když se má znovu příště podruhé tak se odebere – dostává druhou šanci – používá se FIFO a při nastavení flagu se dá nazpět – nahoru
- CLOCK
  - druhá šance ale fronta je cyklická

Nahrazovací algoritmy s proměnným počtem rámců

- VMIN – jako ten optimální – zná budoucnost které se nebude používat – nereálné
- WS (Working Set)
  - v paměti je množina stránek jenom, které byly použity v nějakém intervalu a vždy při použití nějaké stránky se ta množina musí aktualizovat
- Page Fault Frequency
  - jako WS
  - množina stránek se zvětšuje když jsou časté výpadky – je jich tam víc
  - množina stránek se zmenšuje když nedochází moc k výpadkům – odebere se ta která se od předchozího výpadku nepoužila

BLOKY PEVNÉ DÉLKY



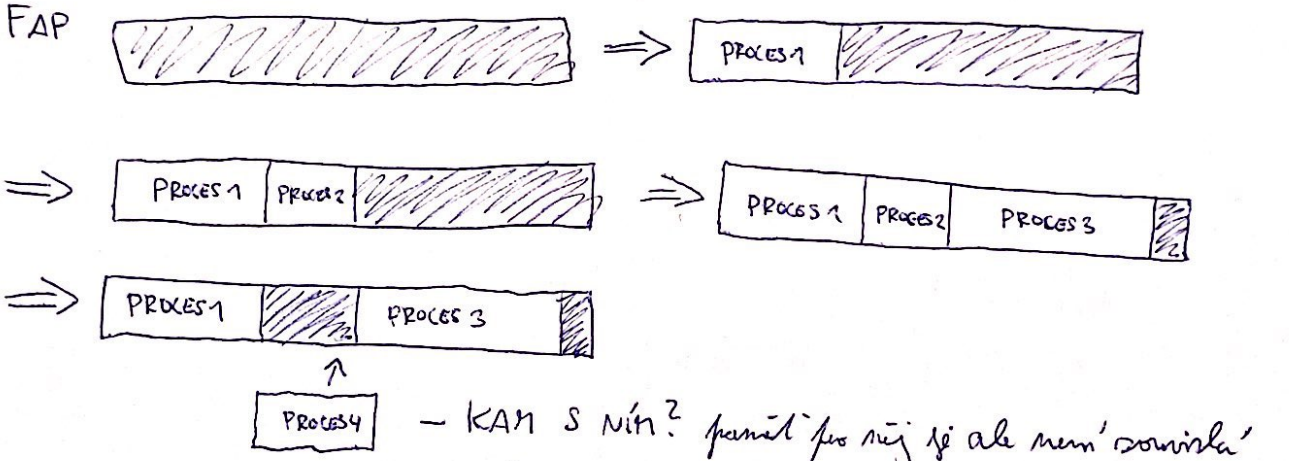
→ EXTERNÍ FRAGMENTACE NENÍ
– celý FAP je rozdělen na bloky a mezi nimi není místo

→ PRÁZDNÁ MÍSTA UVNITŘ BLOKŮ
⇒ INTERNÍ FRAGMENTACE

BLOKY PEVNÉ DÉLKY
– každý blok může mít jinou délku
ale není možné ji měnit

BLOKY PROMĚNNÉ DÉLKY

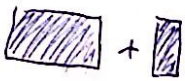
– FAP má bloky dynamicky alokované



– KAM S NÍM? paměť pro něj je ale není souvislá

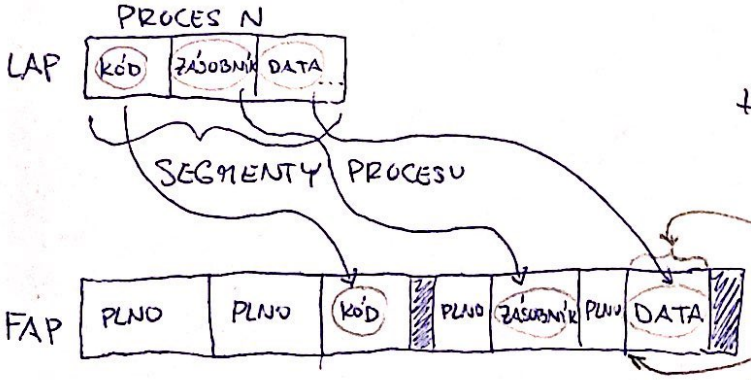
⇒ EXTERNÍ FRAGMENTACE

řešení – sbírání
+
spojování sousedních volných



– interní fragmentace není – bloky jsou na míru

SEGMENTACE



+ TABULKA SEGMENTŮ

obsahuje fyzickou adresu segmentu a limit (blok může rozšířit)

TS < GLOBÁLNÍ – všechny procesy společná
LOKÁLNÍ – každý proces svoje

– proces segmentován a po segmentech uložen do paměti

– EXTERNÍ FRAGMENTACE existuje