Správa primárnej pamäte – pokračovanie

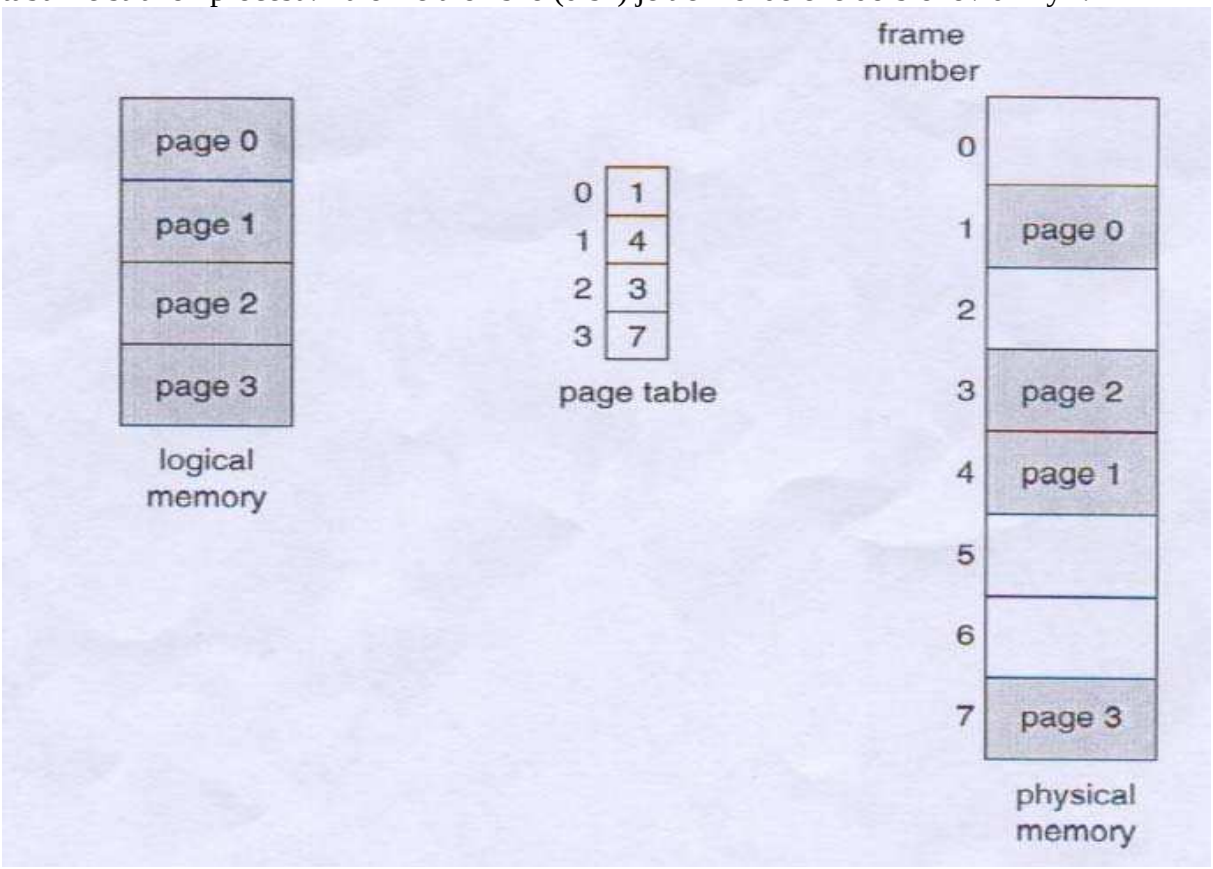
Stránkovanie

Stránkovanie je schéma riadenia pamäte, ktorá povoľuje, aby fyzický adresový priestor procesu bol nesúvislý. Stránkovanie je bežne používané v mnohých operačných systémoch.

Základy stránkovania

Fyzická pamäť je rozdelená do blokov pevnej dĺžky nazývaných **rámce**. Logická pamäť je tiež rozdelená do blokov rovnakej veľkosti nazývaných **stránky**. Dĺžku rámca a stránky označme ***r.***

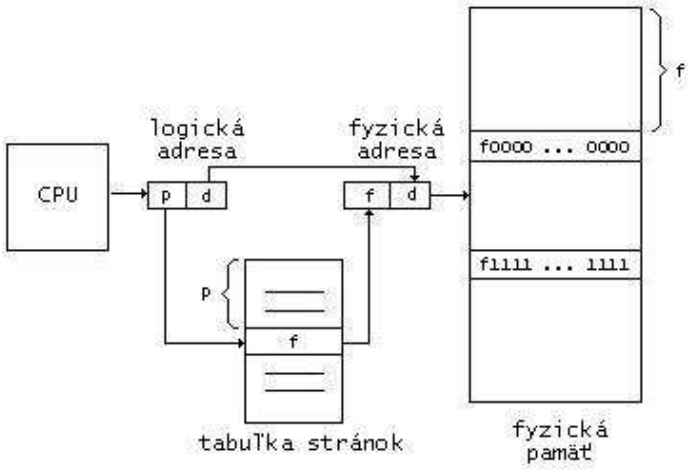
Keď proces má byť vykonaný, jeho stránky sú zavedené do nejakých voľných pamäťových rámcov zo záložného úložiska. Prehľad o priradení rámcov stránkam procesu je držaný v tabuľke stránok procesu. Záložné úložisko (disk) je tiež rozdelené do blokov dĺžky *r*.



Stránkovací hardvér

Hardvérová podpora pre stránkovanie je ilustrovaná na nasledujúcom obrázku. Každá logická adresa generovaná CPU je rozdelená na dve časti: **číslo stránky** (*Page*) ***p*** a **posunutie stránky** (*Offset*) ***d***.

Číslo stránky je použité ako index v **tabuľke stránok**. Tabuľka stránok obsahuje **číslo rámu *f*** každej stránky. Toto číslo rámu *f* je spojené (konkatenácia reťazcov) s ofsetom stránky *d* na definovanie fyzickej pamäťovej adresy, ktorá je odoslaná pamäťovej jednotke.

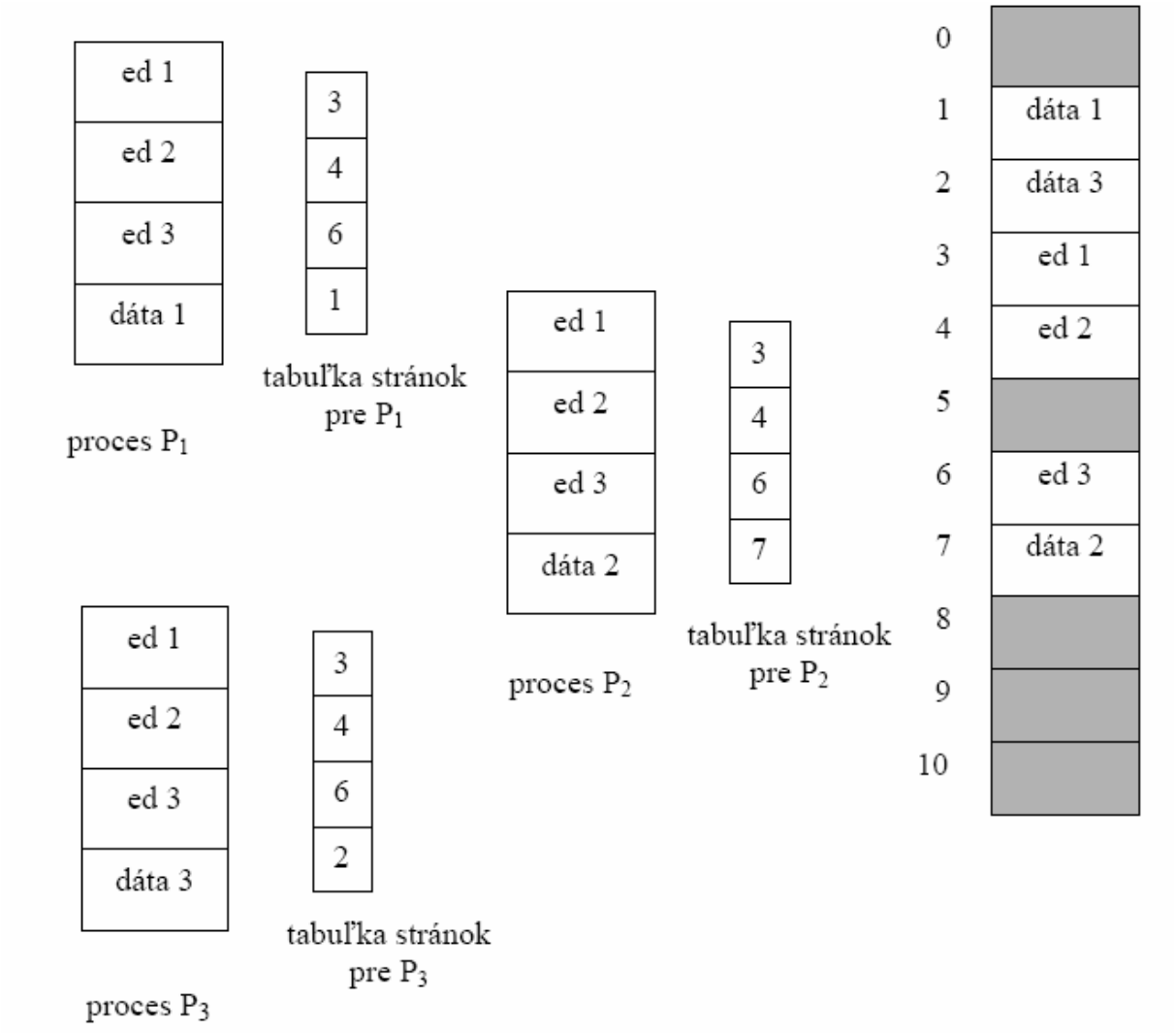


Veľkosť stránky *r* je definovaná hardvérom. Veľkosť stránky je typicky mocninou 2, pohybuje sa medzi 512 bajtmi a 16 MB na stránku v závislosti od architektúry počítača.

Fragmentácia
Keď používame schému stránkovania nemáme externú fragmentáciu. Avšak, môžeme mať internú fragmentáciu. *Posledný* pridelený rámec nemusí byť úplne plný. V priemernom prípade očakávame vnútornú fragmentáciu pol stránky pre proces. V najhoršom prípade bude vnútorná fragmentácia skoro celý rámec (na poslednej stránke bude len jeden bajt).

Veľkosť stránky
Táto úvaha o fragmentácii naznačuje, že sú vhodné malé veľkosti stránok. Na druhej strane je želateľné mať malé tabuľky stránok a teda veľké veľkosti stránok. Treba vybrať kompromis. Dnes je veľkosť stránok zvyčajne medzi 4 KB a 8 KB a niektoré systémy poskytujú ešte väčšie stránky. Niektoré CPU dokonca podporujú viac veľkostí stránok.

Zdieľané stránky
Ďalšia výhoda stránkovania je možnosť *zdieľania* spoločného kódu. Táto okolnosť je dôležitá hlavne v time–sharing prostredí. Uvážte systém, ktorý podporuje 40 užívateľov, z ktorých každý spustí textový editor. Ak textový editor pozostáva zo 150 KB kódu a 50 KB dátového priestoru, potrebovali by sme 8000 KB na podporu 40 užívateľov. Ale, ak kód je **reentrantný**, dá sa zdieľať.
Reentrantný kód (alebo **pravý kód**) je kód, ktorý nemôže modifikovať sám seba. Ak je kód reentrantný, počas vykonávania sa nikdy nemení. Teda dva alebo viac procesov môže vykonávať ten istý kód naraz. Každý proces má vlastnú kópiu registrov a dátové úložisko k držaniu dát pri vykonávaní procesu. Dáta pre dva rôzne procesy sa budú samozrejme navzájom odlišovať.
Vo fyzickej pamäti potrebujeme držať iba jednu kópiu editora. Každá užívateľská tabuľka stránok mapuje na tú istú fyzickú kópiu editora, ale dátové stránky sa mapujú do rôznych rámcov.
A tak, pre 40 užívateľov potrebujeme len jednu kópiu editora (150 KB) plus 40 kópií 50 KB dátového priestoru na užívateľa. Potrebujeme celkovo 2150 KB namiesto 8000 KB — významná úspora.
Ďalšie často využívané programy sa tiež dajú zdieľať — kompilátory, systémy okien, run–time knižnice, databázové systémy a tak ďalej. Aby sa kód dal zdieľať, musí byť reentrantný.



Segmentácia

Dôležitý aspekt správy pamäte, ktorý sa stal nevyhnutným pri stránkovaní, je oddelenie užívateľského pohľadu na pamäť a skutočnej fyzickej pamäte. Užívateľský pohľad na pamäť nie je rovnaký ako skutočná fyzická pamäť.
Tento „estetický“ nedostatok odstraňuje segmentovanie.

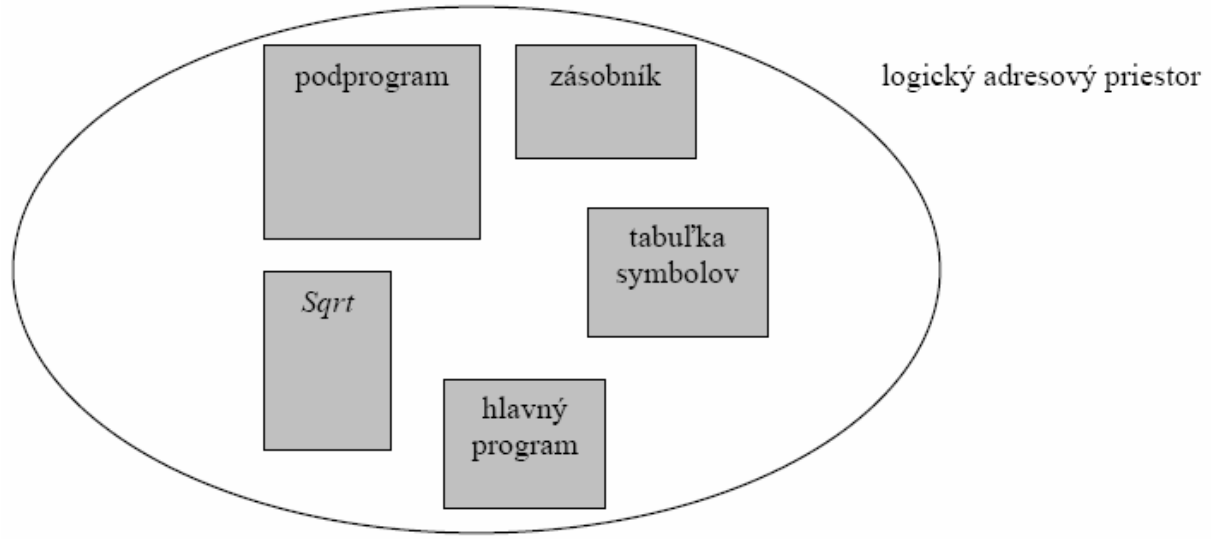
Základy segmentácie

Predstavujú si užívatelia pamäť ako lineárne pole bajtov, kde niektoré bajty obsahujú inštrukcie a iné dáta? Väčšina ľudí by povedala, že nie. Užívatelia skôr uprednostňujú pozerať sa na pamäť, jako na kolekciu rôzne veľkých segmentov bez nutnosti ich zoradenia. Zamyslite sa nad tým, ako rozmýšľate o programe, keď ho píšete. Predstavujete si ho ako hlavný program s množinou podprogramov, procedúr, funkcií alebo modulov. Môžu tam tiež byť rozličné dátové štruktúry: tabuľky, polia, zásobníky, premenné a tak ďalej. Na každý z týchto modulov alebo dátových prvkov odkazujete pomocou mena. Hovoríte o „funkcii *Sqrt*”, „hlavnom programe”, bez toho, že by ste sa starali o adresy v pamäti, ktoré tieto elementy zaberajú. Nezaujímate sa o to, či hlavný program je uložený pred alebo za funkciou *Sqrt*. Každý z týchto segmentov má rôznu dĺžku. Elementy vnútri segmentu sú identifikované podľa svojho ofsetu od začiatku segmentu: Prvý príkaz v programe, sedemnásta položka v tabuľke symbolov, piata inštrukcia funkcie *Sqrt* a tak ďalej.

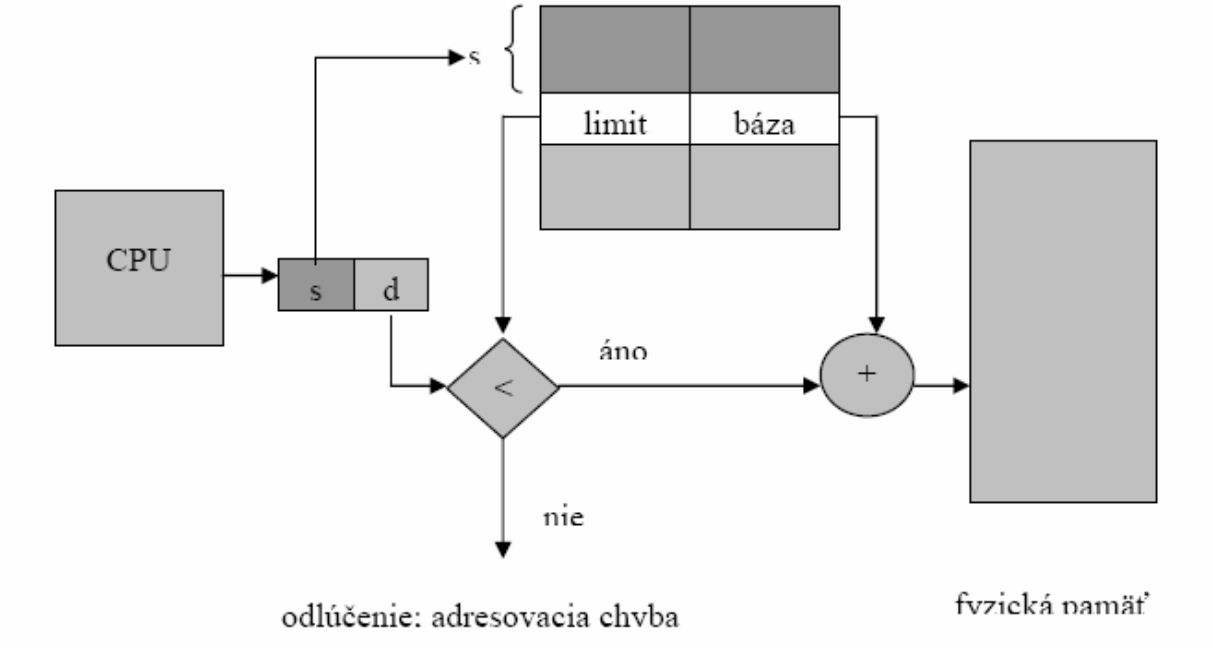
Segmentácia je schéma riadenia pamäte, ktorá podporuje tento užívateľský pohľad na pamäť. Logický adresový priestor je kolekcia segmentov. Každý segment má meno a dĺžku. Adresu určujú meno segmentu a ofset vnútri segmentu. Užívateľ preto specifikuje každú adresu pomocou divoch veličín: mena segmentu a ofsetu.
Pre jednoduchost implementácie sú segmenty očíslované a odkazuje sa na ne skôr pomocou čísla než mena. Logická adresa teda pozostáva z dvojice

<číslo segmentu, ofset>.

Kompilátor automaticky vytvoří segmenty podľa požiadaviek v zdrojovom programe.



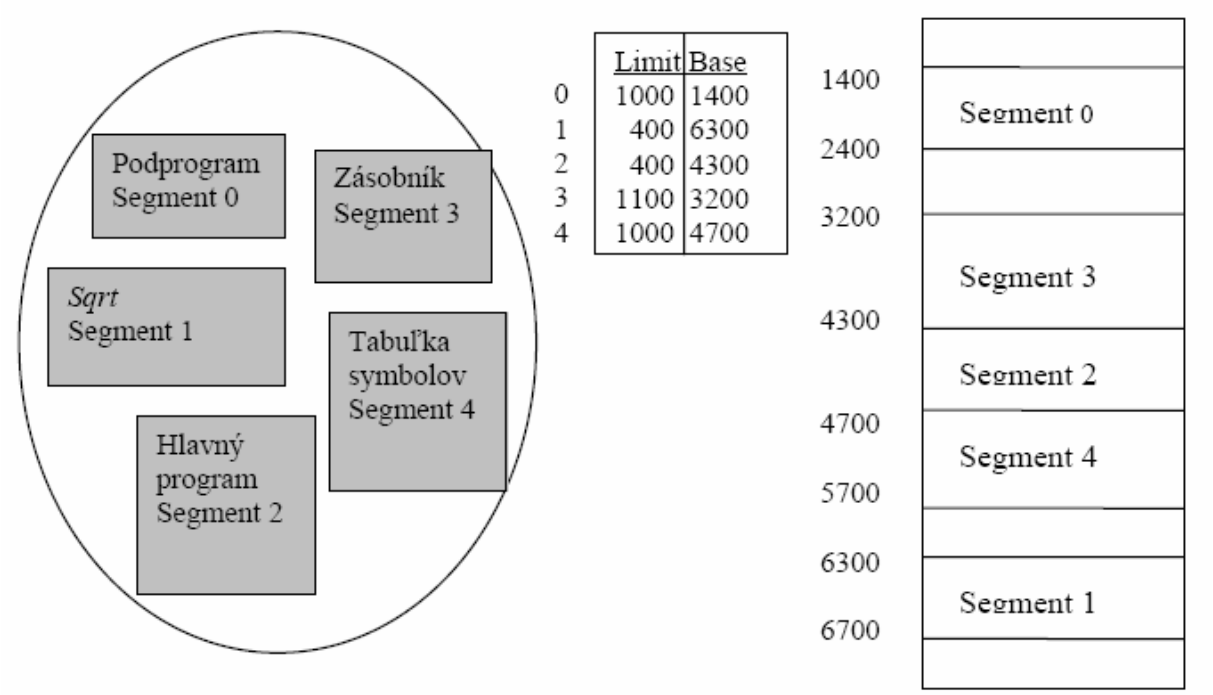
Hoci užívateľ teraz môže odkazovať na objekty v paměti cez dvojrozmerné adresy, skutočná fyzická pamäť je samozrejme jednorozmerná postupnost bajtov. Preto musíme definovať mapovanie dvojrozmerných adries definovaných užívateľom na jednorozmerné fyzické adresy.
Mapovanie je ovplyvnené tabuľkou segmentov. Každá položka tabuľky segmentov má *bázu* segmentu a *limit* segmentu. Báza segmentu obsahuje začiatočnú fyzickú adresu, na ktorej sa segment v pamäti nachádza, zatiaľ čo limit segmentu špecifikuje jeho dĺžku.



Logická adresa pozostáva z divoch častí: z čísla segmentu s a ofsetu d. Číslo segmentu sa používa jako index v tabuľke segmentov. Ofset logickém adresy d musí byť medzi 0 a

limitom segmentu. Ak je tento ofset platný (teda je menší než limit), pripočíta sa k báze segmentu, čím vznikne adresa želaného bajtu vo fyzickej pamäti.

Príklad:



Zdieľanie
Výhodou segmentácie je *zdieľanie* kódu a dát. Je jednoduchšie jako pri stránkovaní.

Fragmentácia
Segmentácia môže spôsobiť vonkajšiu fragmentáciu, ak všetky voľné pamä ťové bloky sú príliš malé na to, aby sa do nich uložil segment. V tomto prípade proces môže počkať, kým sa neuvoľní viac pamäte, alebo kým kompakcia nevytvorí väčšiu dieru.

Virtuálna pamäť

Táto technika umožňuje beh procesov, ktorých LAP (logický adresový priestor) nie je celý vo FAP (fyzickom adresovom priestore), t.j. celý proces ne je zavedený v operačnej paměti. Jedna z možných implementácií je stránkovanie na žiadosť, ktoré tu stručne a zjednodušene popíšeme.

Stránkovanie na žiadosť
Základom je stránkovanie alebo segmentované stránkovanie. Každý rámec má priradené miesto aj na disku — v tzv. odkladacej oblasti. Nie každá stránka procesu musí byť v paměti, niektoré môžu byť na disku.
Čo sa ale stane, ak sa proces snaží pristúpiť k stránke, ktorá je odložená? Nastane tzv. výpadok stránky- proces prejde do stavu čakajúci a operačný systém prenesie stránku do pamäti.
V podstate môže začať svoj beh tak, že len jeho „úvodná stránka“ (začiatok programu) je v pamäti, ostatné sú odložené. Po prenesení chýbajúcej stránky do pamäti, proces prejde do stavu pripravený a eventuálne sa dostane k procesoru.

Pri tejto schéme súčet veľkostí LAP procesov v systéme môže prekročiť veľkosť FAP.

Zdroj:
Studenovsky: Operacne systemy