



Univerzita Hradec Králové
Fakulta informatiky a managementu

Správa paměti

Mgr. Josef Jan Horálek, Ph.D. & Ing. Tomáš Svoboda, Ph.D.





- = Jeden z hlavních úkolů jádra systému – zajistit dostatek prostoru pro jádro a všechny procesy při omezené velikosti paměti a co nejmenším zdržením režijními operacemi
- = Hlavní úkoly:
 - = Převody mezi různými druhy adres
 - = Zpřístupnění paměti, která leží mimo adresní prostor
 - = Nízko-úrovňová alokace pro různé účely

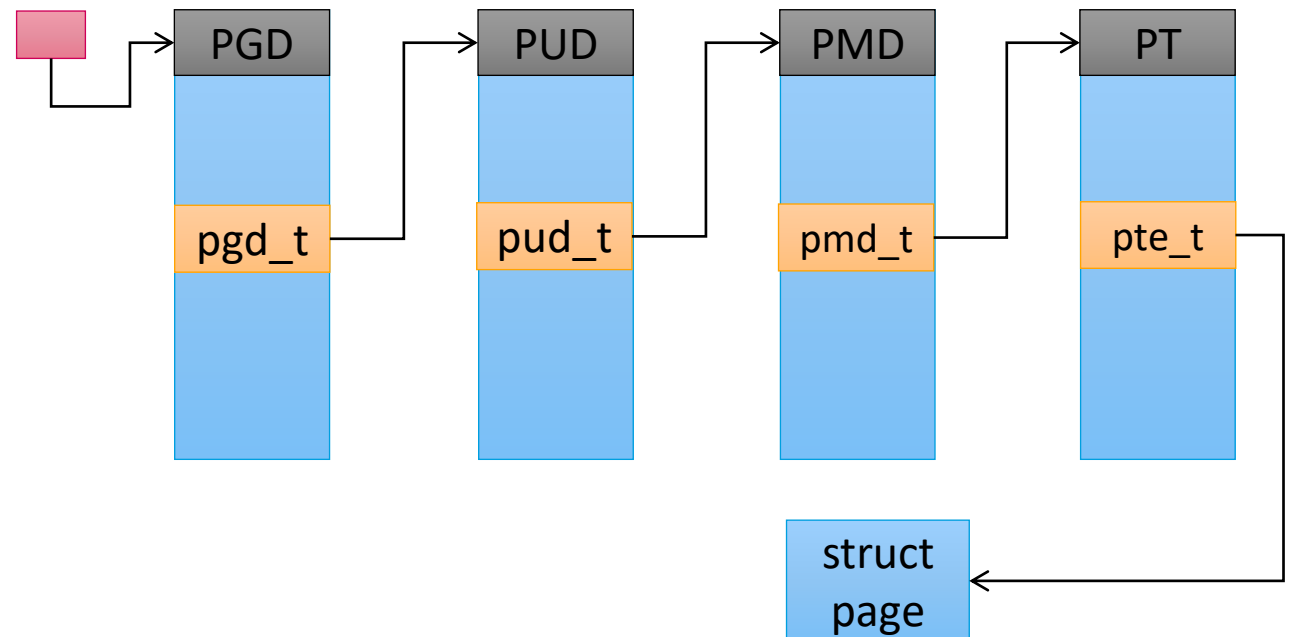


FIM UHK

- = Hlavní úkoly:
 - = Správa paměti a alokace pro jádro
 - = Správa paměti procesů, přidělování, mapování, sdílení
 - = Různé cache, přednačítání, zpožděný zápis
 - = Odkládání na disk, zpětné načítání
 - = Řešení nedostatku paměti



- = PGD – page global directory – globální adresář stránek
- = PUD – page upper directory – horní adresář stránek
- = PMD – page middle directory – střední adresář stránek
- = PT – page table – tabulka stránek





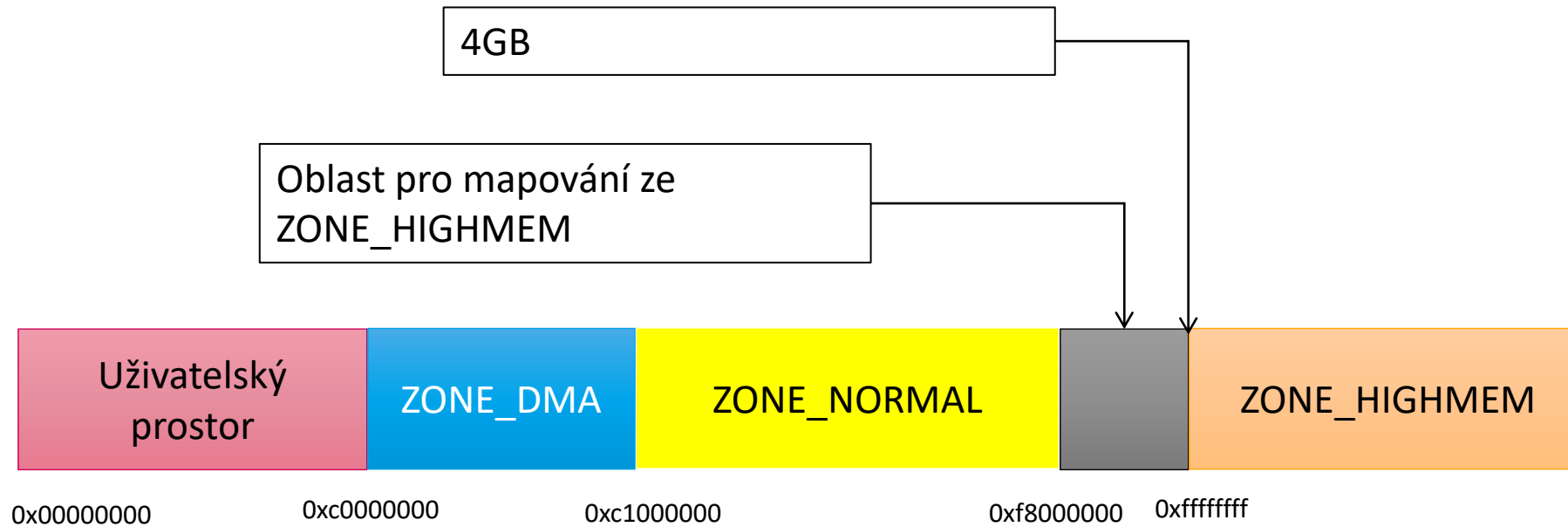
- = Z důvodů HW limitů je paměť rozdělena na zóny, kde každá má své vlastnosti předurčující její použití
- = Dělení na zóny je dáno architekturou – liší se u různých architektur – obecně:
- = Zóna pro DMA
- = Zóna pro normální použití
- = Zóna mimo adresní prostor

Zóna
DMA

Normální
zóna

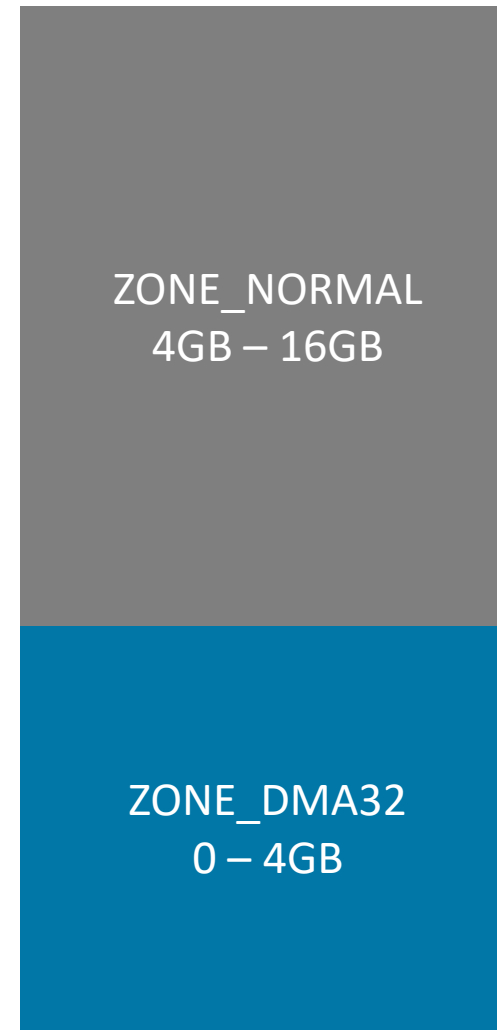
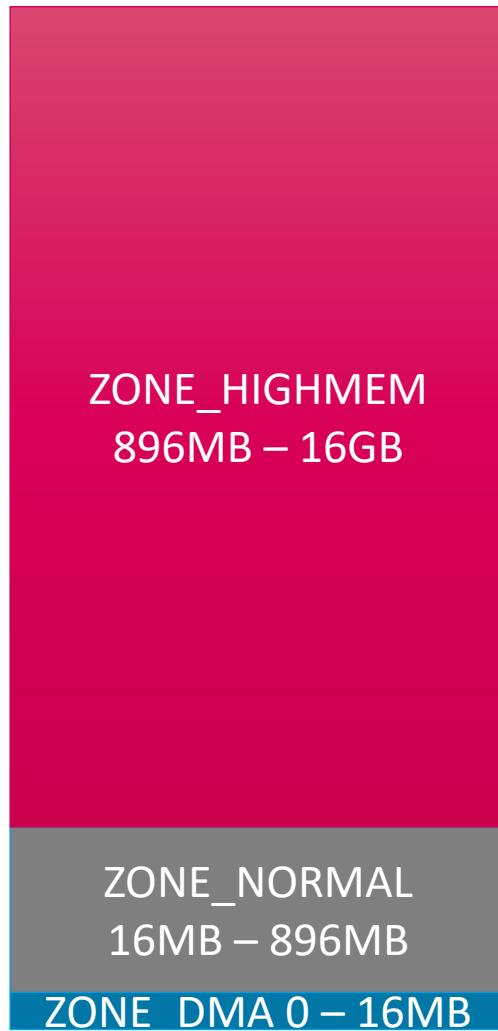
Zóna mimo adresovatelný prostor

= Nejčastější problémy dané HW limity 32bitové architektury x86





- = 64bitový adresní prostor - zjednodušení situace
- = Přímou lze adresovat oblast 64 EB – nevyužívá se tedy zóna horní paměti
- = Přičemž spodní 4GB vždy patří do zóny DMA32





- = Skupiny bloků od 1 do 2^n stránek; n obvykle 10
- = Vždy zdvojnásobuje další úrovně velikosti
- = Celkem 10 úrovní velikosti ve výchozím stavu
- = Volné bloky do seznamů (pro každou úroveň zvlášť)
- = Bloky typicky 4096B (4K)



- = Pravidla pro alokaci:
 - = zkus najít blok odpovídající velikosti – pokud existuje – přiděl ho
 - = zkus najít blok o úroveň větší velikosti – neexistuje-li pokračuj na další úroveň
 - = nalezený blok rozděl tak, aby se přidělila potřebná část a zbytek byl použitelný pro další zpracování
- = Při uvolňování se postupuje opačně. Lze-li sousední dva bloky sloučit učiní se tak. Postupuje se rekurzivně tak dlouho, dokud lze slučovat.



FIM UHK

Rezervované stránky

- = Normální alokace – není-li k dispozici volný blok paměti – volání se uspí – spuštěn mechanismus uvolnění paměti – uvolní dostatečnou velikost – alokace probuzena
- = Ne vždy možné – jádro udržuje některé stránky vyhrazené pro použití, kdy se alokace nesmí blokovat
- = Tato paměť se dělí mezi ZONE_NORMAL a ZONE_DMA resp ZONE_DMA32



FIM UHK

Cache pro stránky

- = Vylepšení výkonnosti při alokaci
- = Často se alokují jednotlivé stránky – využití buddy pomalé
- = Pro každý procesor v jádře vytvořeny cache – horká (hot cache) a studená (cold cache)

- = **_alloc_page()** základní alokační funkce volaná z **alloc_pages()**
- = Na začátek zkusí získat volné stránky pomocí **get_page_from_freelist()** – pokud ne – volán démon kswapd, aby uvolnil dostatečný počet stránek
- = Paralelně s démonem proveden druhý pokus o získání stránek **get_page_from_freelist()**
- = Pokud ne – proveden třetí pokus
- = Pokud dojde k selhání i nyní – zacyklení a pokus neustále opakován dokud se nedosáhne úspěchu
- = Automatická alokace – zde končí neúspěchem

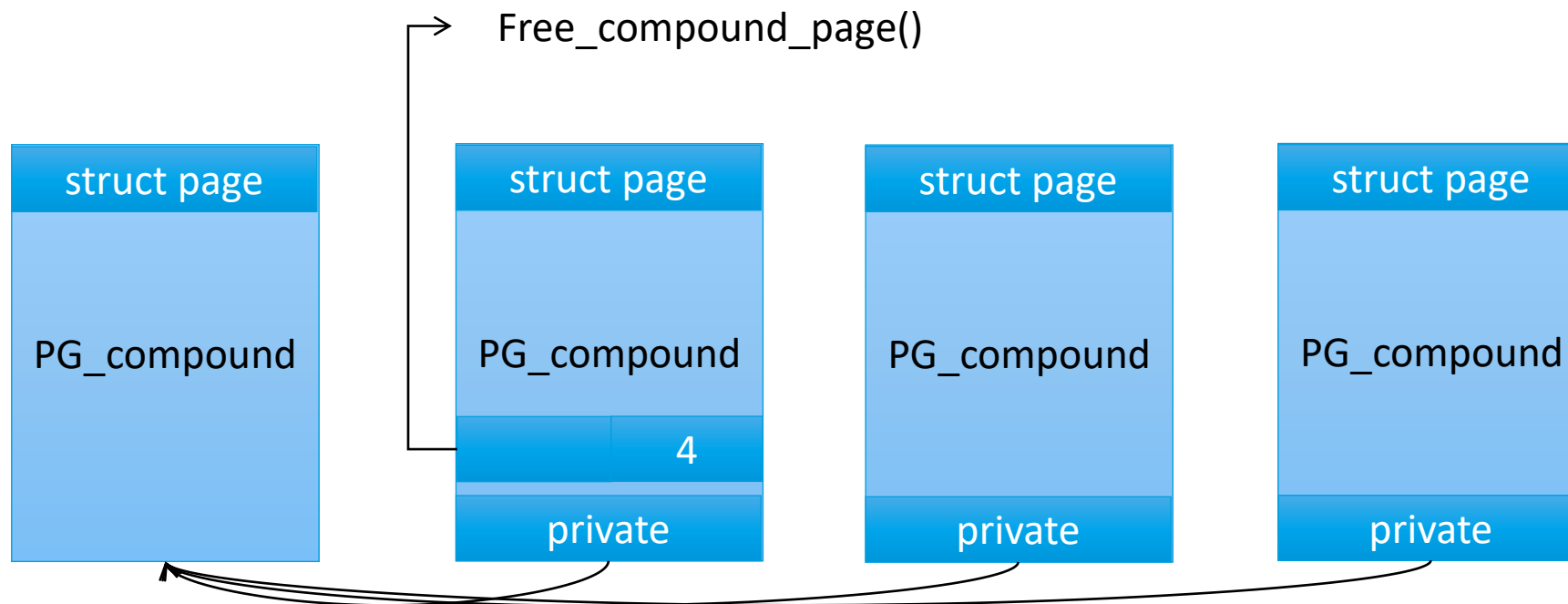


- = Normální alokace
- = dá prostor k běhu jiným procesům
- = pak synchronní uvolňování paměti (**try_to_free_pages()**)
- = nový pokus získat volné stránky
- = je-li neúspěšné a současně neuspělo ani synchronní uvolňování = out of memory (nedostatek paměti – funkcí **out_of_memory()**)



`get_page_from_freelist()`

- = Funkce prochází jednotlivé paměťové zóny
- = Podle použitého limitu rozhoduje o možnosti použití příslušné zóny
- = Lze-li zónu použít volá se pro ni `buffered_rmqueue()` pro získání volných stránek, které pracují přímo se zónovým alokátozem

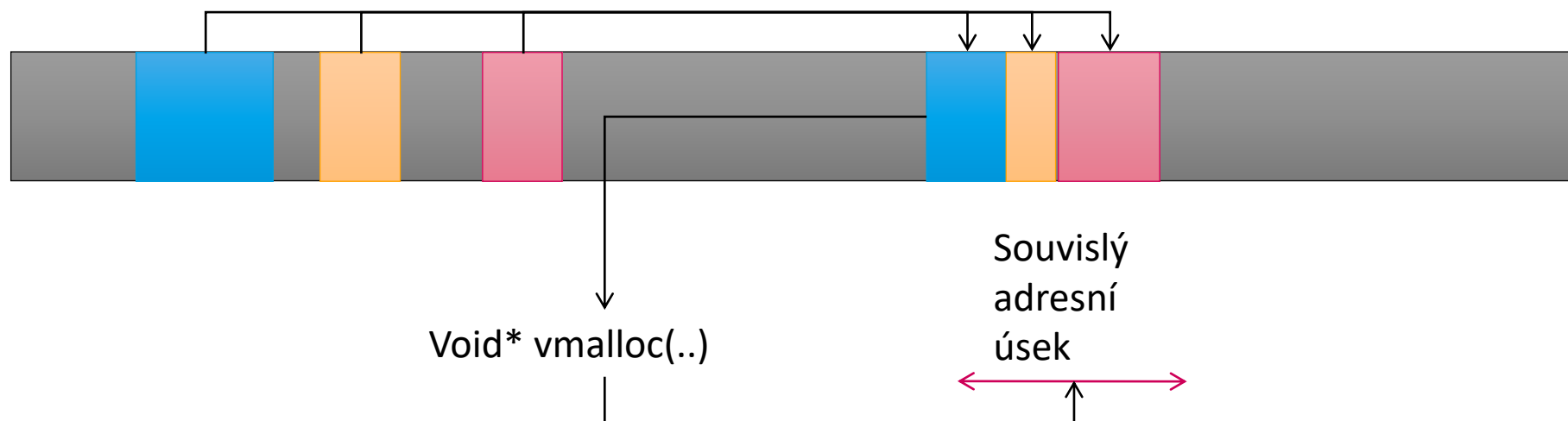
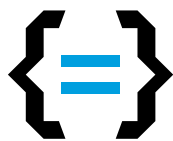


- = Jednoduchý proces na vyšší úrovni – složitější na úrovni algoritmu buddy
- = Funkce **_free_pages()** dekrementuje počítadlo referencí
- = testuje uvolnění stránky
- = volá **free_hot_page()**, která stránku vloží do hot cache aktuálního procesoru
- = pokud dosažena horní mez cache
- = funkce **free_pages_bulk()** nadbytečné stránky vrací do buddy seznamů

- = Není jednoduché nalézt souvislý úsek paměťových stránek
 - = alokace nesouvislých úseků a jejich namapování
- = Alokují se sada úseků a ty se namapují do virtuálního adresného prostoru jádra – získáme souvisle adresovatelný úsek – pak lze normálně pracovat
- = Funkce `get_vm_area_node()` hledá volnou oblast, kam bude paměť po alokaci namapována, v rozsahu adresného prostoru vyhrazeného pro tento účel



- = Oblast musí být dostatečně velká
 - = velikost + 1stránka navíc – ochrana
- = Vlastní alokace
 - = funkce `_vmalloc_area_node()`
 - = alokuje jednotlivé stránky.
- = Ukazatele alokovaných stránek ukládány do pole – použito při mapování



- = Alokovanou paměť je potřeba po použití uvolnit
- = Funkce **vfree()** – volá funkci **_vumap()**, která se stará o odmapování a uvolnění
- = Nejdříve dojde k odmapování a je-li potřeba postupně prochází všechny stránky a nastává dealokace

- = Metody řešení nedostatku fyzické paměti:
 - = Uvolnění aktuálně nevyužívaných stránek
 - = Aktivace mechanismu OOM killer – postupně ukončí tolik procesů, aby vyřešil situaci
- = Kategorie uvolňovaných stránek:
 - = Odložitelné
 - = Synchronizovatelné
 - = Přímo uvolnitelné



FIM UHK

Reverzní mapování

- = Používá se pro najití všech položek odkazujících na určitou stránku
- = V linuxovém jádře se používá objektové reverzní mapování a to buď anonymní nebo souborové

- = Nejlépe a nejrychleji řešit nedostatek paměti, zároveň nejméně omezovat jádro a procesy
- = Pravidla:
 - = Přednost mají nejdéle nepoužívané stránky
 - = Přednost mají stránky, které nikdo nepoužívá. Stránky, které nepoužívá žádný proces by se měly uvolňovat dříve než stránky použité v procesech
 - = Co nejvíc stránek procesů musí být uvolnitelný – všechny stránky kromě aktivních
 - = Pro uvolnění sdílených stránek se musí odmapovat všechny odkazující položky



FIM UHK

OOM killer

- = Situace, kdy se nedaří uvolnit žádnou další paměť, kterou je potřeba alokovat – násilné ukončení procesů
- = Začíná se výběrem vhodného kandidáta na ukončení
 - = hledá funkce `select_bad_process()`
 - = prochází všechny procesy a vybere ten s nejvyšším skóre (badness)



- = Proces swapoff
 - = systém si nemůže dovolit přijít o swap
 - = maximální skóre
 - = Alokovaná paměť procesu
 - = přímá hodnota
- = Počet potomků
 - = $\frac{1}{2}$ paměti každého potomka rodiči
- = Procesorový čas
 - = skóre dělíme druhou mocninou času procesoru v desítkách sekund



- = Reálný čas
 - = skóre se dělí čtvrtou odmocninou času běhu procesu v desítkám sekund
- = Hodnota nice
 - = je li >0, skóre se zdvojnásobí
- = Administrátorské procesy
 - = procesům s UID=0 nebo EUID=0 a procesům CAP_SYS_ADMIN se skóre dělí čtyřmi
- = Procesy s přímým přístupem k HW
 - = procesům WIO se skóre dělí 8



Výpočet badness() - ovlivňuje

- = Společné uzly
 - = pokud proces nemá s aktuálním procesem společné paměťové uzly, skóre se dělí osmi
- = Nastavená hodnota
 - = hodnota **oom_adj** říká, o kolik se má skóre posunout
 - = tedy hodnota 3 je posun o 3 bity
 - = osmina původního skóre



FIM UHK

OOM killer

- = Nenalezne-li funkce žádného kandidáta, je systém zastaven, protože nelze pokračovat dál.
- = Pokud vhodný proces nalezen
 - = volá se funkce **b**
 - = začne testem zda proces již není ukončován
 - = pokud ano nastaví se mu příznak **TIF_MEMDIE** a nic dalšího se neprovede



- = Než začne ukončovat přímo vybraný proces pokusí se ukončit některého z jeho potomků
- = Pokud úspěšné – nepokračuje
- = K ukončení používána funkce **oom_kill_task()**
- = ta ukončí určitý proces a současně i všechny ostatní procesy, které s ním sdílejí paměťový kontext.
- = Samostatné ukončení se provádí zasláním signálu SIGKILL



Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost

Další téma: Správa procesů a vláken

