



Pokročilé operační systémy

Virtuální paměť a preemptivní multitasking

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci



- mapuje stránky logického adresního prostoru na fyzické rámce
- dvouúrovňová tabulka stránek
 - adresář tabulek stránek (page directory), který odkazuje na
 - adresář stránek (page table)
- page directory i table mají velikosti jedné stránky (4 kB) a obsahují 1024 záznamů
- horních 20 bitů představuje odkaz na rámec ve fyzické paměti
- spodních 12 bitů další příznaky (přítomnost stránky, oprávnění pro práci, tj. zápis, přístup z ring3, a další)
- aktuální mapování paměti je určeno řídicím registrem CR3 (odkaz na adresář tabulek stránek)
- stránkování musí být povoleno v registru CR0 (nejvyšší bit)



- při přístupu na neplatnou stránku dojde k vyvolání výjimky č. 14 (výpadek stránky, page fault)
- typicky nenastavený bit *present* v tabulce stránek nebo nedostatečné oprávnění
- adresa, kde došlo k výpadku, je uložena do řídicího registru CR2
- na zásobník je navíc uloženo chybové slovo indikující důvod výpadku
- v obslužné rutině by mělo dojít k přenastavení stránkovací tabulky nebo k adekvátní akci (ukončení procesu)
- po návratu z obsluhy přerušení je opětovně provedena instrukce, která vyvolala výpadek stránky
- při změně tabulky stránky, je nutné zneplatnit obsah TLB (instrukce `invlpg`)



- na správu volných rámců nejsou kladeny větší nároky
- lze vytvořit zásobník rámců, které nejsou přiřazeny žádnému procesu a z něj odebírat/přidávat
- jednoduché na implementaci (mírně neúsporné)
- alternativně lze použít bitmapu (úspornější, ale náročnější na vyhledávání)
- alternativně lze oba přístupy kombinovat



- paměť jádra (v našem případě)
 - do adresy 0x00200000
 - mapování stránek na identické (co do adres) rámce
 - stejné u všech procesů
 - systémové stránky
 - stránkovací tabulky alokovány na haldě jádra
- paměť uživatelského prostoru (v našem případě)
 - zbylý logický adresní prostor
 - rámce přiřazovány podle přístupů k jednotlivým stránkám (nerespektuje systémové volání brk)

další možnosti

- rozdělení paměťového prostoru na dvě poloviny uživatelský prostor/jádro (upperhalf kernel)
- Linux od 0xc0000000



- jednoduché zařízení schopné generovat přerušení v daných intervalech
- nastaví se hodnota počítadla a při každém tiku je tato hodnota snížena o 1
- při nastavení hodnoty na nula dojde k přerušení (č. 1, resp. 0x20)
- na x86 tři samostatné kanály
- různé režimy práce (jednorázový režim, opakovaný, ...)

Možnosti

- nepreemptivní – proces se sám vzdá procesoru (např. **systemové volání** `sched_yield`)
- nepreemptivní v jádře – proces sám předá řízení plánovači (**zavoláním** odpovídající funkce)
- preemptivní – důsledek doběhnutí časovače (**přerušení**)

Společné vlastnosti

- je potřeba uložit kontext (stav) procesoru
- u všech tří variant jsme v jádře (máme aktivní jaderný zásobník)
- část informací je již na zásobníku (původní `cs:eip`, `esp`, ...)
- stačí uložit zbývající registry (např. `pushad`)
- pro každý proces nutné alokovat samostatný jaderný zásobník
- lze použít haldu jádra



- při přepínání procesů je nutné
 - 1 přepnout jaderný zásobník
 - 2 přenastavit stránkovací tabulku (jsme v jádře a stránky jádra jsou sdílené, jednoduché)
 - 3 aktivovat časovač (pro preemptivní multitasking)
 - 4 provést návrat podle toho, jak přepnutí došlo (přerušeni, systémové volání, volání)

díličí technické komplikace

- načtení kódu do správné oblasti paměti
- předávání argumentů spouštěnému programu (nutné přes jádro)



- preemptivní jádro
 - proces v jádře nelze přerušit
 - nemusíme řešit zamykání
 - problém s delé trvajícími I/O operacemi
- plnohodnotné blokuující procesy
- jaderná vlákna
- virtuální souborový systém (VFS)
- page cache pro souborový systém
- IPC



- Implementujte v uživatelském prostoru knihovnu pro vlákna.
- Nápoředy:
 - Každé vlákno musí mít svůj vlastní zásobník.
 - U funkce `thread_yield`, která přepíná vlákna, stačí zachovávat/uchovávat pouze callee-saved registry.
 - Uložení kontextu jde vyřešit jednoduchým kódem v assembleru.

- zkouška bude probíhat individuální formou
- spočívá v implementaci vybraného úkolu a diskuzi nad ním
- před zkouškou pošlete zdrojové kódy s řešením a stručnou textovou dokumentaci
- je možné odevzdat i dílčí řešení (např. s chybami)
- termín zkoušky bude dohodnut individuálně
- v případě potřeby mě můžete kontaktovat



Úkol: IPC

- Implementujte do ukázkového operačního systému mechanismus meziprocesové komunikace, který umožní přádavat mezi procesy zprávy pevné velikosti.
- Operace `send` (neblokující), operace `receive` (blokující).
- Další vlastnosti jsou otevřené.