Pokročilé operační systémy

Virtuální pamět a preemptivní víceúlohový systém

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Virtuální paměť

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

Výpadek stránky

- přístupu na neplatnou stránku, dojde k vyvolání výjimky č. 14
- typicky nenastavený bit present v tabulce stránek, nedostatečné oprávnění
- adresa, kde došlo k výpadku, je uložena do řídícího 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
- při změně tabulky stránky, je nutné zneplatnit obsah TLB (instrukce `invlpg`)

Správa rámců

- na správu volných rámců nejsou kladeny větší nároky
- lze vytvoři 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

Rozložení paměti

- 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`

Časovač (Programmable Interval Timer)

- jednoduché zařízení schopné generovat přerušení v daných intervalech
- nastaví se hodnota počitadla 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 (jednorázová režim, opakovaný, . . .)

Přepínání procesů

Možnosti

- nepreemptivní – proces se sám vzdá procesoru (např. **systémové volání** `sched_yield`)
- nepreemptivní v jádře – proces sám předá řízení plánovače (**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

Přepínání procesů

- 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 (přerušení, systémové volání, volání)

dílčí technické komplikace

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

Co chybí?

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

Úkol k procvičení

- Implementujte v uživatelském prostoru knihovnu pro vlákna.
- Nápovědy:
  - 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

- 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

Zkouška

Úkol č. 1: IPC

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

Úkol č. 2: FAT

- Implementuje souborový systém FAT (verze dle vlastní volby), viz `https://academy.cba.mit.edu/classes/networking_communications/SD/FAT.pdf`.
- Nemusí být součástí ukázkového operačního systému a může být i ve vyšším programovacím jazyce než C.
- Měl by minimálně podporovat vytváření, čtení, zápis souborů, práci s adresáři.