Implementace režimu jádra a uživatelského režimu.

Thursday, May 30, 2013 8:27 AM

While transitioning from user mode to kernel mode, the processor makes a switch between the per-process-user-stack and the per-process-kernel-stack. Then the user-per-process stack segment selector and stack pointer is stored in the kernel stack and then the `eip` instruction pointer (return address at user mode) and other hardware registers are pushed on to the kernel stack
When the kernel has to return to user mode, the `trapret` code pops all values stored in the kernel stack back to the hardware registers.

From <http://stackoverflow.com/questions/9968028/returning-from-kernel-mode-to-user-mode>

Výpočet v módu jádro – v důsledku událostí:

Popsat rozdíl mezi uživatelským módem a režimem jádra.

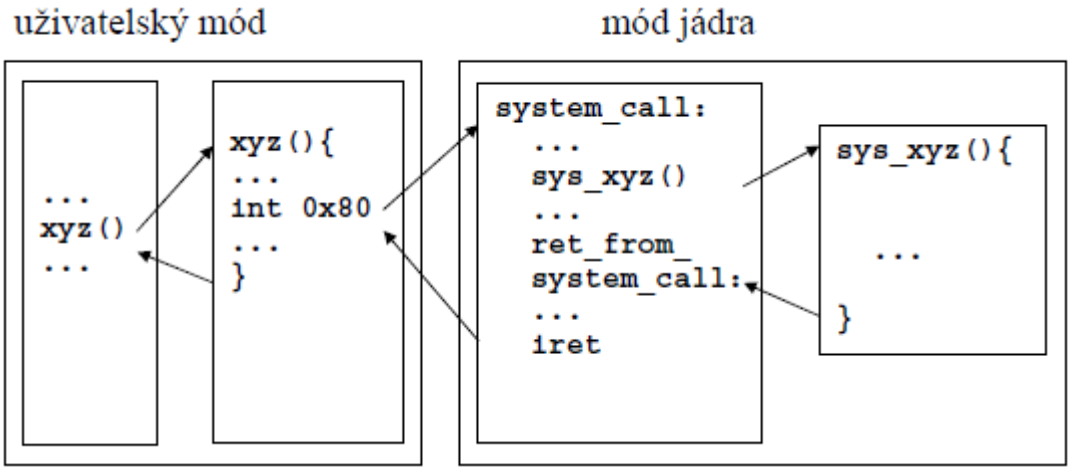
- Přerušení
- Výjimky
- Softwarové přerušení

Řízení se předá na proceduru pro ošetření odpovídající události. Část stavu přerušeného procesu potřebná pro jeho vykonávání po skončení obsloužení události (počítadlo instrukcí, PSW (**Processor Status Word**) se uloží do zásobníku jádra přerušeného procesu.

Systémové volání

V standardní knihovně jazyka C je pro každé systémové volání obálková procedura, řízení se předá software přerušením proceduře jádra, která se nazývá syscall(), system_call. Protože je jen jedna pro všechny služby, požadovaná služba je identifikována parametrem procedury, který se nazývá číslo systémového volání.

Linux



Aplikační program volá službu xyz, obálková procedura uloží číslo služby do registru **eax** před vykonáním **int 0x80**.

System_call

- Uloží obsah registrů (hardwarový kontext)
- Zavolá odpovídající funkci (v jazyce C)
- Ukončí se voláním ret_from_sys_call()

Výjimky se zpracují obdobně

- jsou synchronní s procesem (vznikají v důsledku událostí způsobených vykonáváním procesu)

- Procedury pro jejich zpracování mají obdobnou strukturu jako procedura pro systémová volání **systém_call**

Linux

Procedura pro zpracování výjimky

- Uloží obsah registrů
- Zpracuje výjimka (funkce v jazyku C)
  - Pošle signál procesu
  - Zpracuje žádost o stránku
- Ukončí se voláním funkce ret_from_exception()

Zpracování přerušení

Přerušení je obecně asynchronní vzhledem k přerušenému procesu – proces čeká na přenos dat, po dokončení přenosu je přerušen úplně jiný proces. Zpracování přerušení nesmí způsobit čekání, přerušený proces zůstává ve stavu běžící. Čas zpracování přerušení je započítán přerušenému procesu, při zpracování přerušení se tedy přistupuje do jeho záznamu **proc**.

Obsloužení přerušení:

- Uloží IRQ (Interrupt ReQuest) a obsah registrů
- Pošle potvrzení PIC (Programmable Interrupt Controller)
- Vykoná obslužní proceduru přerušení
- Ukončí se skokem na ret_from_intr()

Vzájemné vnoření systémového volání, výjimek a přerušení

Předpokládejme odladěné jádro

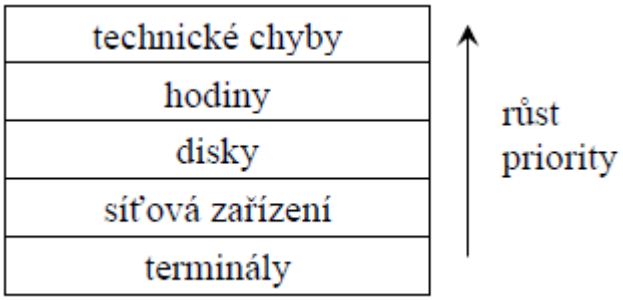
1. Zpracování systémového volání
   - Může vzniknout výjimka při žádosti o stránku (výpadek stránky)
   - Může dojít k přerušení
2. Zpracování výjimky
   - Může dojít k přerušení
3. Zpracování přerušení
   - Může dojít k přerušení

Při každém odkladu zpracování některé z uvedených událostí musíme nejdříve uložit odpovídající HW kontext.

- Vytváří se kontextové vrstvy zásobníku jádra přerušeného procesu
- Existuje globální zásobník přerušení

Prioritní schéma

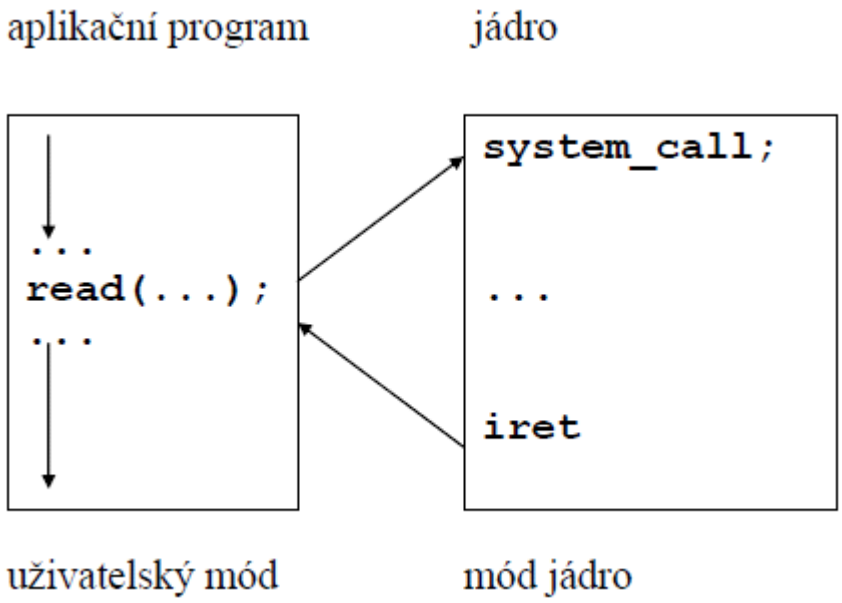
- Přerušení mají přiřazené prioritní úrovně (interrupt priority level)



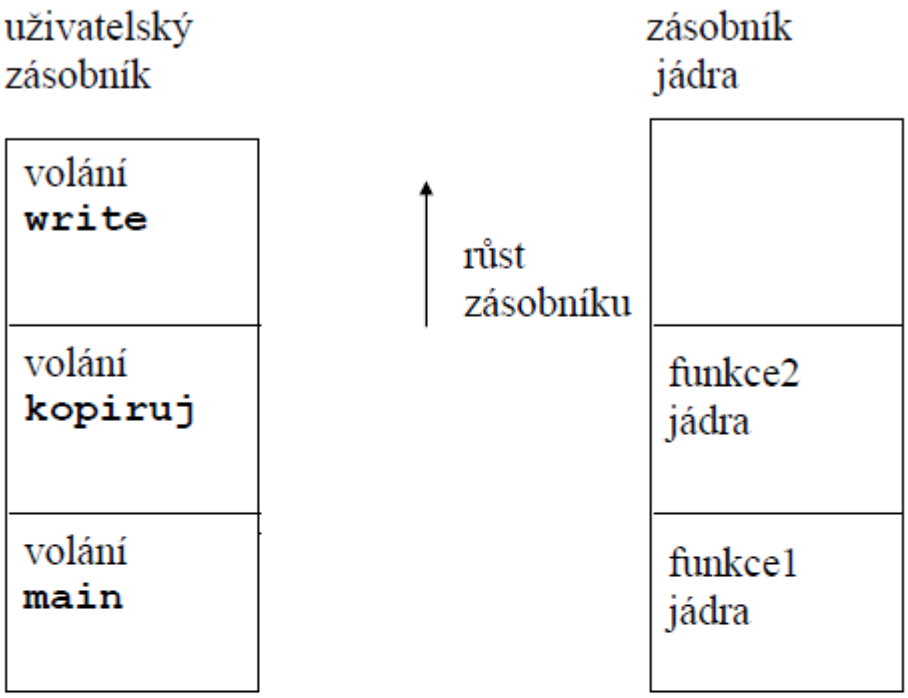
Ve stavovém registru procesoru je nastavena okamžitá prioritní úroveň zpracovávaného přerušení. Vznikne-li přerušení přerušení s nižší nebo stejnou prioritní úrovní, je uloženo a jeho obsluha je odložena.
Vznikne-li přerušení s vyšší prioritní úrovní, uloží se HW kontext, na tuto vyšší prioritní úroveň se nastaví hodnota okamžitého přerušení ve stavovém registru procesoru a přerušení se zpracuje.

Při skončení zpracování přerušení se z uloženého PSW obnoví okamžitá prioritní úroveň přerušení.

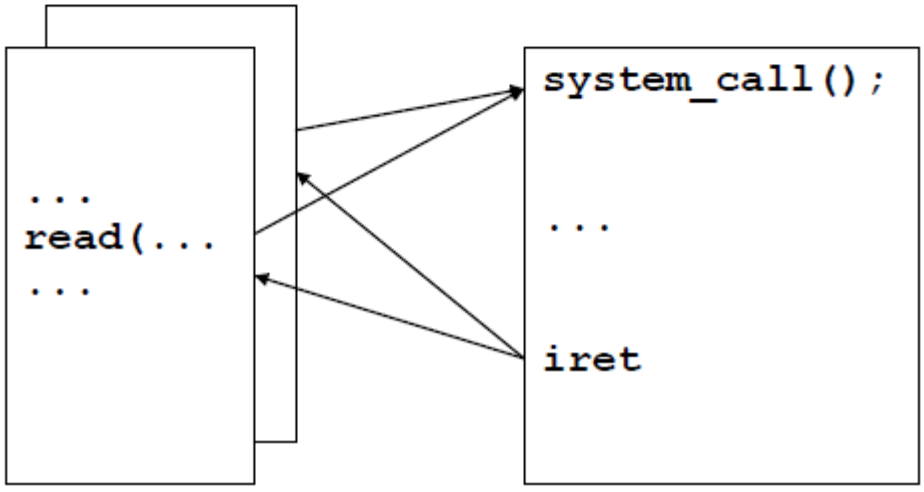
Proces a jádro



- uživatelský
- jádra



více procesů



Jádro je reentrantní – každý proces má svůj zásobník jádra, často v adresovém prostoru procesu – chráněný, spravovaný jádrem. Každý proces má položku v tabulce procesů: **proc** záznam a **u** (user) oblast.

Z PPR

Princip – MSDOS compatible, uniprocesor

V reálném režimu: Pomocí služby DOSu se nainstaluje handler přerušení 8, které generují hodiny. Když je volána obsluha přerušení, v zásobníku jsou uloženy registry CS, IP a Flags kódu, jehož vykonávání bylo přerušeno. Obsluha přerušení uloží stávající registry, plánovač vybere novou úlohu a zapíše do zásobníku její hodnoty uvedených registrů (CS, IP a Flags). Obsluha přerušení obnoví zbývající registry procesoru pro plánovačem vybranou úlohu. Provede se instrukce iret, kterou se spustí naplánovaný proces díky přepsání hodnot v zásobníku.

Z <https://d.docs.live.net/67cbdb2faf0647ea/Dokumenty/FAV/A11N0110P/Státní%20závěrečná%20zkouška/PPR/PPR_Karfik_v2.docx>

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>