Univerzita Hradec Králové
Fakulta informatiky a managementu

Systémová volání

Mgr. Josef Horálek

FIM UHK

Systémová volání

- Systémová volání = volání jádra
- základní komunikace aplikačních programů s jádrem
- Tvůrce programu obvykle oddělen vrstvou standardní knihovny – nepotřebuje tedy znát přesné chování – je však dobré je pochopit

FIM UHK

Základní princip

- Jak vypadá vztah uživatelský proces – jádro ?
- Jedná se o dva rozdílné světy:
  - uživatelský proces běží ve svém adresním prostoru (instrukce i data) a vykonává kód programu
  - někdy potřebuje službu, kterou pro něj vykoná jen jádro – systémové volání

Základní princip

Příklad:

Zjištění a nastavení parametrů zařízení – funkce ioctl () .

```
int ioctl (int fd, unsigned long request,…);
```

Podle konkrétní hodnoty požadavku má volání dva či tři argumenty.

V jádře pak vypadá prototyp fce např. takto:

```
Long (*unlocked_ioctl)(struct file *filp,
  unsigned int cmd, unsigned long arg);
```

FIM UHK

Základní princip

= Můžeme nyní zanedbat odlišné datové typy a další detaily – pak se jedno volání funkce transformuje na jiné.

= V čem se tedy systémové volání liší od běžné funkce?

= V tom, že leží právě na hranici této transformace.

FIM UHK

Jak to funguje - například ioctl()

1. Uloží se aktuální obsah registrů
2. Do registrů se uloží číslo systémového volání a parametry
3. Předá se řízení jádru (přepnutí do režimu jádra a začne vykonávat jeho kód)
4. Podle čísla volání se vyhodnotí, jaká funkce v jádře se zavolá
5. Protože jde o volání spojené s otevřeným souborovým deskriptorem, najde se odpovídající objekt souboru

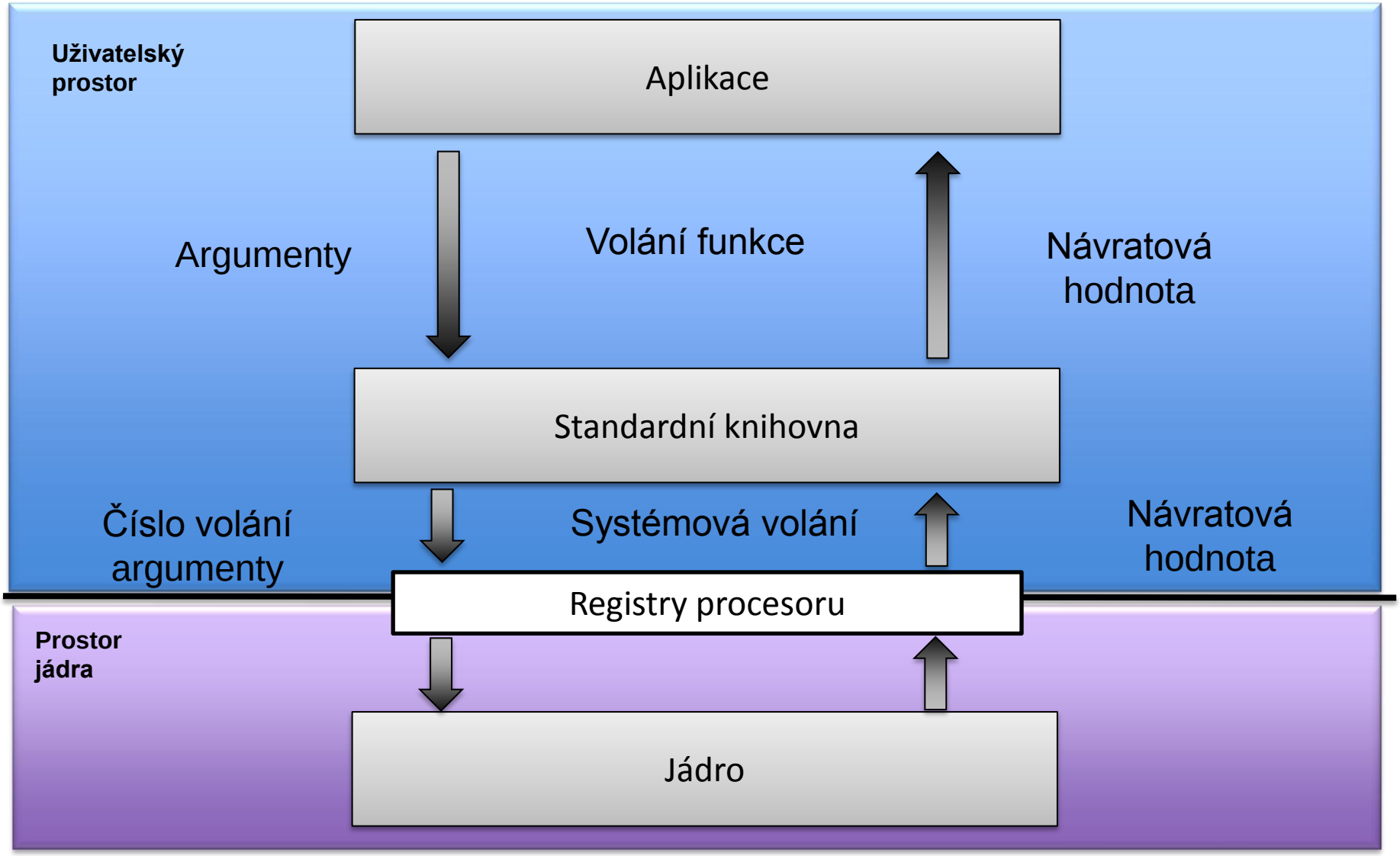
FIM UHK

Jak to funguje - například ioctl()

6. Podle objektu souboru se zavolá příslušná výkonná funkce v ovladači
7. Ve výkonné funkci se provede vše potřebné a na konci se vrátí výsledek
8. Do registru se uloží výsledek operace
9. Řízení se předá zpět programu resp. Standardní knihovně (uživatelský režim)
10. Výsledek se přenese z registru do paměti
11. Obnoví se uložený stav registrů

FIM UHK

Schéma systémového volání



FIM UHK

Do režimu jádra a zpět

- Jak vlastně probíhá předání řízení jádru a opětovné předání zpět?
- Dvě metody (platformě závislé):
  - speciální instrukce
  - přerušení

Přerušení

- Starší metoda (pro i386)
  - Široké použití
  - Pomalejší
    - Volající proces způsobí softwarové přerušení (platforma x86 nejčastěji 0x80)
    - Začne se obsluhovat, procesor přepnut do režimu jádra – provádí kód jádra
    - Po skončení je obnoven stav procesoru – přepnut do uživatelského režimu

FIM UHK

Speciální instrukce

- Nevyužívá přerušení
- Instrukce **sysente**r sama zajistí přepnutí do režimu jádra
- Nastaví registry (CS, EIP atd.) a předá řízení kódu jádra
- Na závěr se vykoná instrukce **sysexit**, která připraví registry pro návrat a přepne do uživatelského režimu

FIM UHK

Reakce na výsledek

- Systémové volání končí opuštěním jádra a předáním výsledku přes registr.
- Konkrétní hodnoty závisí na potřebách volající aplikace, existuje však několik základní možností:
  - Úspěšně zakončené volání, které nepřenáší zpět žádnou konkrétní hodnotu, vrací nulu (0)
  - Úspěšně zakončené volání, které přenáší návratovou hodnotu, vrací tuto hodnotu. (obecně jakákoli hodnota obvykle nezáporná)
  - Neúspěšně zakončené volání vrací zápornou hodnotu odpovídající kódu chyby

FIM UHK

Přerušené volání

- Často nastává situace, kdy je vykonáván kód uvnitř implementace systémového volání a v tu chvíli přijde signál, na který se musí reagovat.
- V toto chvíli mohou nastat tři základní situace:
  - Systémové volání je buď krátké, nebo se nachází již blízko svého konce, tedy nemusí již na nic čekat. Pak se volání normálně dokončí a vrátí standardní výsledek.

FIM UHK

Přerušené volání

- Volání bylo přerušeno na začátku, resp. V situaci, kdy buď nebyly v systému provedeny žádné změny, nebo byly provedeny bezpečně reverzibilní změny. Tato situace se řeší opuštěním jádra a vrácením hodnoty
  –ERESTARTSYS, -ERESTARTNOHAND,
  – ERESTARTNOINTR.
- K přerušení volání došlo v situaci, kdy již byly provedeny nevratné změny v systému, například byl do zařízení odeslán manipulační příkaz. Situace se řeší opuštěním jádra s návratovou hodnotu –EINTR. Nejdříve se samozřejmě musí systém dostat do konzistentního stavu.

FIM UHK

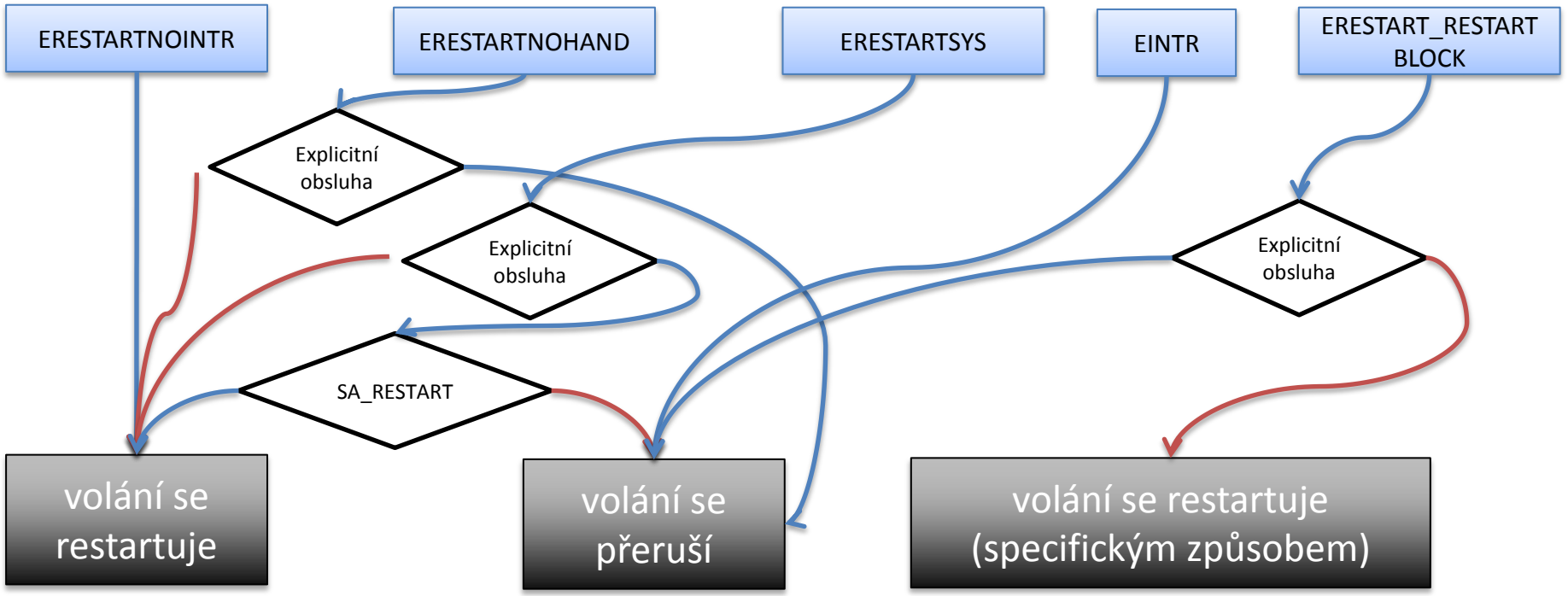
Přerušené volání

- ERESTARTSYS – volání bude opět restartováno; výjimkou je případ, kdy byl signál explicitně obsloužen a neměl nastaven příznak SA_RESTART. Tehdy nastane přerušení volání a návrat do programu
- ERESTARTNOHAND – volání bude restartováno vždy, když není signál explicitně obsloužen uživatelskou rutinou
- ERESTART_RESTARTBLOCK – podobná situace s použitím jiného volání

FIM UHK

Informace o modulech

- ERESTARTNOINTR – volání se restartuje za každých okolností
- EINTR – volání se vždy přeruší



FIM UHK

Krátce o oprávnění procesu

- Systémová volání obecně představují potenciální hrozbu pro běžící systém – volání, která něco nastavují nebo mění, mohou obecně do běžícího systému zasáhnout negativním způsobem na ostatní běžící procesy, uživatelská data nebo systém.

FIM UHK

Krátce o oprávnění procesu

- Pro oprávnění se využívá model založený na právech definovaných pro určitou třídu operací tvz. capabilities.
- Jako příklad můžeme uvést:
  - CAP_CHOWN - právo měnit vlastníka souboru
  - CAP_KILL – právo poslat cizímu porcesu signál
  - CAP_NET_BIND_SERVICE – právo používat privilegované porty
  - CAP_SYS_TIME – právo nastavit systémový čas
  - CAP_SYS_ADMIN – právo k různým operacím souvisejícím se správou systému
  - atd.

FIM UHK

Krátce o oprávnění procesu

- Jak to funguje?
  - Zjednodušeně můžeme říci, že se ovladač nebo jiná součást jádra dotáže, zda má proces potřebnou kvalifikaci.
- Jak proces kvalifikaci získá?
  - nefunguje dědění;předávání; uložení v atributech souborů
  - Cesta: prosec spuštěný s právy root zbaví všechny procesy kvalifikací kromě těch, které potřebuje.
  - Pak změní uživatele, pod nímž běží, se zachováním aktuálních kvalifikací.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost…