Univerzita Hradec Králové
Fakulta informatiky a managementu

Signály

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

Signály

- Jedná se o nejstarší metody komunikace mezi jádrem a procesem, a mezi samotnými procesy.
  - Princip
    - proces vykonává určitou činnost – přijde mu signál – přeruší původní činnost – obslouží signál – proces může pokračovat dál v práci
- Rozdíl oproti přerušení a speciálním instrukcím
  - Přerušení
    - legacy způsob přechodu User-space procesu – Kernel-space
    - HW přerušuje kernel
  - Speciální instrukce
    - doporučený způsob přechodu User-space procesu – Kernel-space

Druhy signálů

- Dělení podle implementace:
  - obyčejné signály
  - real-time signály
- Obyčejné signály
  - jedná se o bity v masce signálu. V příchozím signálu se odpovídající bit (určený číslem signálu) nastaví na jedničku. U zpracovávaného signálu se bit vynuluje.
  - neprojeví se jeho vícenásobné doručení před jeho obsloužením

Druhy signálů

- Real-time signály
  - nazývané spolehlivé signály, používají frontu – zaručeno, že žádný signál není ztracen,
  - doručen tolikrát, kolikrát byl poslán
  - real-time signály používají čísla od 32 výše,
  - využívá se např.pro komunikaci mezi vlákny,

Druhy signálů

- Jako další možné dělení se používá možnost předefinování reakce na daný signál.
  - Na většinu to možné je.
- Výjimku tvoří:
  - SIGKILL – okamžité ukončení procesu
  - SIGSTOP – zastavení procesu
- Proč?
  - Stačí nekonečný cyklus v main metodě a dopady jsou zřejmé …..

Druhy signálů

- Dělení podle posílání signálů:
  - Synchronní
  - Asynchronní
- U většiny signálů se vyskytují oba způsoby.
- U synchronních signálů přesně víme, kdy ho proces obdrží.
- Asynchronní signál může přijít kdykoli za běhu procesu a reakce na něj by tomu měla být přizpůsobena.

Posílání a doručování signálů

- Existují dvě skupiny signálů:
  - signály posílané zásadně jádrem
  - signály posílané uživatelskými procesy
- Základní metoda pro poslání signálu je využitím funkce kill().
  - ta umožňuje poslat signál jednomu procesu nebo všem procesům ve skupině.(kromě init).
- Existuje funkce raise(), která posílá signál stejnému procesu, který ji zavolal.
  - Proč metoda kill()
    - historický důvod – kompatibilita z UNIX systémy|dříve jediný signál – ukonči proces (dnes SIGKILL) | volání signal() a handler() by dnes dávalo lepší smysl

FIM UHK

Dostupné signály kill -l

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-75 58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Posílání a doručování signálů

- Další speciální funkcí je sigqueue(), která je využívaná pro real-time signály a informuje zda byl signál vložen do fronty.
- Často se využívá funkce pthread_kill(), která slouží k poslání signálu určitému vláknu – lze použít jen v rámci jednoho procesu.
- Lze zaslat signál i vláknům nebo skupině vláken různých procesů, k čemuž se využívá funkce tkill() a tgkill(). Primárně určeny pro použití v knihovnách.

Posílání a doručování signálů

- Posílaný signál je dříve nebo později doručen, pokud je komu.
- Blokované signály považujeme za doručené až v okamžiku jejich odblokování.
- Signál poslaný procesu jako celku, je považován za doručený až je přijatý celým procesem.
- Je-li signál poslán jen konkrétnímu vláknu, pak může být doručen jen jemu.

Reakce na signál

- Záleží na kde se proces při doručení signálu nachází v uživatelském režimu nebo v jádře.
- Pokud se nachází v jádře, pak:
  - Vrátí se do uživatelského prostoru
  - Vykoná obslužnou rutinu
  - Návrat do jádra – odtud vzápětí návrat do uživatelského prostoru s návratovou hodnotou definovanou přerušeným voláním

Reakce na signál

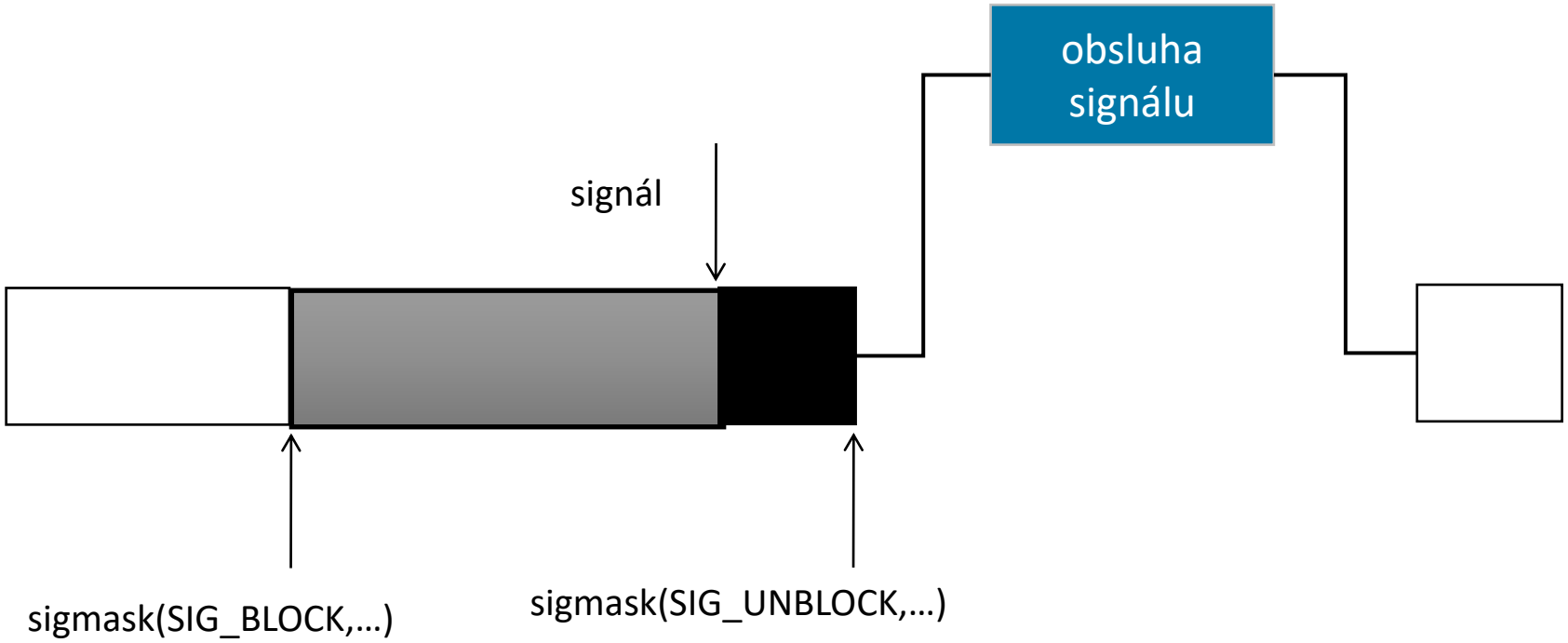
- Záleží na kde se proces při doručení signálu nachází v uživatelském režimu nebo v jádře.
- Pokud se nachází v uživatelského prostoru, pak:
  - Bude donucen vstoupit do jádra
    - V rámci téhož procesu je zajištěno přeplánováním procesu – po přepnutí kontextu se již nevrátí do uživatelského prostoru
  - Obsloužení signálu jako v předchozím příkladu
  - Návrat do jádra – úklid dat pro obsluhu signálu – přepnutí do už. Prostoru proces pokračuje tam, kde přestal.

Reakce na signál

- Reakce na signál:
  - výchozí reakce
    - nezměníme-li předem reakci na signál, použije se ta, která je pro daný signál výchozí.
  - ignorování
    - na příchozí signál se nebude nijak reagovat
  - obslužná rutina
    - určí se funkce, která bude pro obsluhu zavolána
  - synchronní zpracování
    - jedná se o zvláštní případ, ve skutečnosti jde jen o to, že se v jádře čeká na příchod signálu a následně je vráceno jeho číslo.

Blokace signálu

= Existují případy, kdy nechceme, aby signál byl doručen okamžitě, ale až v určitou chvíli. Tento požadavek se řeší tzv. blokováním signálu.



Blokace signálu

- K manipulaci s blokacemi signálu se využívá jejich bitová maska (datový typ sigset_t) a sada funkcí.
- Funkce sigemptyset() masku signálu vynuluje – nebude tedy obsahovat žádné signály.
- Funkce sigfillset() ji naopak naplní, sigaddset() přidá signál do masky, sigdelset() signál z masky odstraní. Pro kontorlu, zda je určitý signál v masce používáme sigismember().

= Reakce na signál

```
void sig_handler(int signo) {
     if (signo == SIGINT)
          printf("received SIGINT\n");
}

int main(void) {
     signal(SIGINT, sig_handler); //Obsluha signálu SIGINT
     while(1)
          sleep(1);
     return 0;
}
```

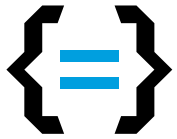
Ukázka programu

= Pokročilá reakce na signál

```
void pSigHandler(int signo) {
      if (signo == SIGTSTP) {
            printf("TSTP");
      }
}

int main(void) {
  struct sigaction psa;
  psa.sa_handler = pSigHandler;
  sigaction(SIGTSTP, &psa, NULL);

  for(;;) {sleep(1);}
  return 0;
}
```



Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost

Další téma: Souborové operace