Signály

Vznik signálu

Generovanie signálov

Maskovanie signálov

Spracovanie signálov

Vznik signálu

- **Signál** je asynchrónne evidovanie udalosti, resp. evidovanie asynchrónnej udalosti v systéme.
- Signál je zaslaný procesu (práve bežiacemu, resp. inému) ak vznikne udalosť generujúca signál.
- Vybrané udalosti generujúce signál:

1. hardvérové a softvérové chyby v systéme (delenie nulou, chybná inštrukcia,....)
2. expirácia časovača
3. rôzne spôsoby ukončenia procesov
4. pozastavenie / opätovné spustenie procesu
5. zrušenie procesu z terminálu
6. “umelé” generovanie signálov pre synchronizáciu procesov a ovládanie behu procesov

Signály synchronizácie procesov

- Signály, ktoré sa účelovo používajú na synchronizáciu behu procesov z algoritmického hľadiska:

  SIGUSR1, SIGUSR2, SIGSTOP, SIGCONT, SIGTERM, SIGKILL,

  alebo na oznámenie expirácie časových intervalov SIGALRM

Reakcia procesu na signály

- Aktívny proces môže reagovať na signál jemu určený nasledovne:

1. Proces, **predpripravením masky signálov** , zamedzí prijatie signálu – zablokuje jeho vstup do procesu. Signál zostáva zapísaný v dátovej oblasti procesu. Signál zostane v stave „nevybavený – pending“ pokiaľ:

   proces nezanikne

   proces nezmení nastavenie masky

   proces nastaví ignorovanie signálu

- Signály SIGSTOP a SIGKILL nemôžu byť nikdy blokované

Reakcia procesu na signály

2. Proces **nie je špeciálne pripravený na vznik** (príchod) signálu. Signál vstúpi do procesu a následne sa vykoná aktivita, ktorá je pre daný signál v systéme prednastavená (defaultná). Príklad je spracovanie signálu SIGFPE (floating point exception)
3. Proces **je pripravený na prijatie** signálu

- jeho ignorovaním
- obslužnou (odchytávajúcou) funkciou, ktorá danú situáciu spracuje

Nie všetky signály sa dajú ignorovať,

(SIGSTOP a SIGKILL nemôžu byť nikdy ignorované) alebo odchytiť)

Klasifikácia spracovania signálov

- Existujú dve triedy spracovania signálov závisiace od rôznych Unixovských implementácií:

  -nespoľahlivé

  -spoľahlivé

  Nespoľahlivé signály sú tie, ktorých handler po prvom volaní nezostane nainštalovaný. Tieto signály musia byť v samotnom handlery reinštalované.

  Signály sú „nespoľahlivé“ preto lebo operácie zachytenia signálu a reinštalácie handlera sú neatomické - signál môže vzniknúť prv než sa inštalácia znovu vykoná a preto môže byť stratený

Klasifikácia spracovania signálov

- Spoľahlivé spracovanie signálov je také, pri ktorom zostáva handler stále nainštalovaný

  - reinštalácia handlera je nie potrebná

  - signál sa nemôže stratiť bez spracovania handlerom

Rôzne implementácie funkcií signálov

- SVR4

  Obsahuje funkcie: *signal*, *sigset*, *sighold*, *sigrelse*, *sigignore* a *sigpause*.

  Funkcia *signal* je nespoľahlivá obsluha, ostatné funkcie zabezpečujú automatickú reinštaláciu handlera, systémové volania sa po prerušení nereštartujú.
- BSD

  Funkcie: *signal*, *sigvec*, *sigblock*, *sigsetmask* a *sigpause*.

  Všetky funkcie poskytujú spoľahlivé signály s reštartovaním systémových funkcií.
- POSIX1

  Funkcie *sigaction*, *sigprocmask*, *sigpending* a *sigsuspend*.

  Všetky funkcie poskytujú spoľahlivé signály. Reštarovanie systémových volaní je pre POSIX nie definované.

Rôzne implementácie funkcií signálov

- Najbezpečnjší spôsob spracovania signálov aj z hľadiska portability programu je použitie funkcie *sigaction*.
- Pre SVR4 a BSD s flagom SA_RESTART v struct sigaction
- Ak chceme použiť v Linuxe funkciu *signal* so sémantikou pre BSD je treba kompilovať s voľbou

  -I/usr/include/bsd -lbsd

GENEROVANIE SIGNÁLOV

Funkcie spôsobujúce vznik signálu:

- raise (int signal) (3C)
- abort() (3C) generuje SIGABRT (nemaskovatelný) stdlib.h
- unsigned alarm (unsigned sec) (2) unistd.h

  vznik signálu SIGALRM resetuje všetky nastavené alarmy,

  sec= 0 , zruší nastavenie alarmu

  zvyšok nezrušeného alarmu sa zdedí cez exec()
- kill (pid_t pid, int signal)

  ak pid > 0 signál pre definovaný proces

  pid = 0 pre všetky procesy z rovnakej process group ID

  pid = -1 pre všetky procesy, ktorých real user ID je rovný effective user ID zasielajúceho procesu .....
- sigsend (.....) s podobným významom, ako kill

Ako proces zistí, že dostal signál

- Len aktívny proces môže zistiť či mu bol doručený signál
- Vždy po návrate z volania systémovej funkcie proces zisťuje či existujú nejaké signály, ktoré mu boli zaslané
- proces môže na signál čakať svojim suspendovaním (stav waiting) pokiaľ signál nepríde

MASKOVANIE SIGNÁLOV

Vytvorenie premennej, ktorá bude použitá ako maska

- Funkcie sú deklarované v signal .h

  sigemptyset (sigset_t * set) všetky signály povolené

  sigfillset (sigset_t * set) všetky signály zakázané

  sigaddset (sigset_t * set, int signo) pridanie zakázaného signálu

  sigdellset (sigset_t * set, int signo) uvolnenie signálu

  sigismember (sigset_t * set, int signo) test
- sigset_t je štruktúra

MASKOVANIE SIGNÁLOV

Nastavenie masky pre proces

sigprocmask (int how, sigset_t * set, sigset_t * oset)

kde how je

- SIG_BLOCK - nastavenie blokovania je rozšírené o set
- SIG_UNBLOCK - set je vybratý z aktuálnej masky
- SIG_SETMASK - nastaví sa maska podľa set

MASKOVANIE SIGNÁLOV

- Jednorázové nastavenie masky a suspendovanie procesu, pokiaľ nepríde povolený signál

  sigsuspend (sigset_t * set)
- Test na existenciu nevybaveného signálu

  sigpending (sigset_t * set)

  Výsledok testu sa uchová v set
- Jeden zo pôsobov blokovania procesu , ktorý bude odblokovaný príchodom signálu

  sigpause()

OŠETRENIE SIGNÁLU

- JEDNORÁZOVÉ ošetrenie signálu

  signal (int signo, meno_obsluznej_funkcie)

  meno_obsluznej_funkcie :

  SIG_IGN - ignorovanie signálu

  SIG_DFL - špecifikuje defaultové spracovanie signálu

  smerník na funkciu, korá sa po vzniknutí signálu vykoná. Jej argumentom je číslo signálu.

  void meno_ obsluznej_funkcie (int sig)

OŠETRENIE SIGNÁLU

- TRVALÉ PRESMEROVANIE ošetrenia signálu.

  sigaction (int signo, struct sigaction * act, struct sigaction * oact), kde

```
struct sigaction {
                sigset_t   sa_mask;
                int        sa_flags;
                void       sa_handler;
    /*podla nastavenia sa_flags
      buď smerník na obslužnú funkciu typu
       void  meno (int)
      alebo smerník na obslužnú funkciu typu:
       void   meno (int,  siginfo_t  *,  struct ucontext  *)*/
}
```

OŠETRENIE SIGNÁLU

- sa_mask - maska signálov, ktorá bude platná počas behu funkcie obsluhujúcej signal

  (+ k nej bude automaticky priradený obsluhovaný signál !?)
- sa_flags - hodnoty pre rôzne varianty chovania sa procesu ku vzniknutému signálu:

  SA_NODEFER - obsluhobaný signál nebude pridaný k maske

  SA_NOCLDWAIT - ignoruj SIGCHLD (nebudú vznikať zombie)

  SA_SIGINFO - bude použítá obslužná f. druhého typu („alebo“) štruktúru siginfo vytvorí systém

OŠETRENIE SIGNÁLU

- typedef struct siginfo {
  si_signo - obsluhovaný signál
  si_code - pre rôzne typy signálov bližšia špecifikácia dôvodu vzniku
  si_errno - číslo chyby podľa errno.h
  si_pid - ID procesu zasielajúceho signal pre sigqueue()
  si_value - hodnota signálu
  .....
} siginfo_t;

HODNOTY SA_FLAGS V STRUCT SIGACTION

- SA_NOCLDSTOP – po skončení detského procesu sa nevyšle SIGCHLD
- SA_RESTART – po prerušení handlerom sa niektoré systémové volania reštartujú
- SA_NOMASK – maska signálov sa nepoužije, počas výkonu handlera sú signály blokované
- SA_ONESHOT – zrušenie handlera po jeho vykonaní
- SA_INTERRUPT – v Linuxe sa nepoužíva
- SA_STACK

Ktoré procesy a komu môžu zasielať signály

- Jadro a proces superuser môžu zasielať signál každému procesu
- užívateľský proces môže zasielať signál len procesom toho istého užívateľa rovnaké uid a gid