Sistemsko programiranje

Međuprocesna komunikacija II

Katedra za računarstvo
Elektronski fakultet u Nišu

Prof. dr Dragan Stojanović
Prof. dr Aleksandar Stanimirović
Prof. dr Bratislav Predić

Sadržaj

- Signali
- Redovi poruka

Sadržaj

- Signali
- Redovi poruka

Signali

Pojam

- Signali su mehanizam koji se koristi kako bi se proces obavestio da se u sistemu desio neki asinhroni događaj.
- Signale najčešće šalje sam operativni sistem kako bi procese obavestio o pojavi grešaka ili izuzetaka.
- Procesi takođe mogu da koriste mehanizam signala za međusobnu komunikaciju.
- Proces može da šalje signale drugim procesima ukoliko ima privilegije za to.

Obrada

- Po prijemu signala proces ga može obraditi na jedan od tri načina:
  - primljeni signal se ignoriše
  - koristi se podrazumevana obrada signala
  - proces definiše funkciju za obradu signala

Signali

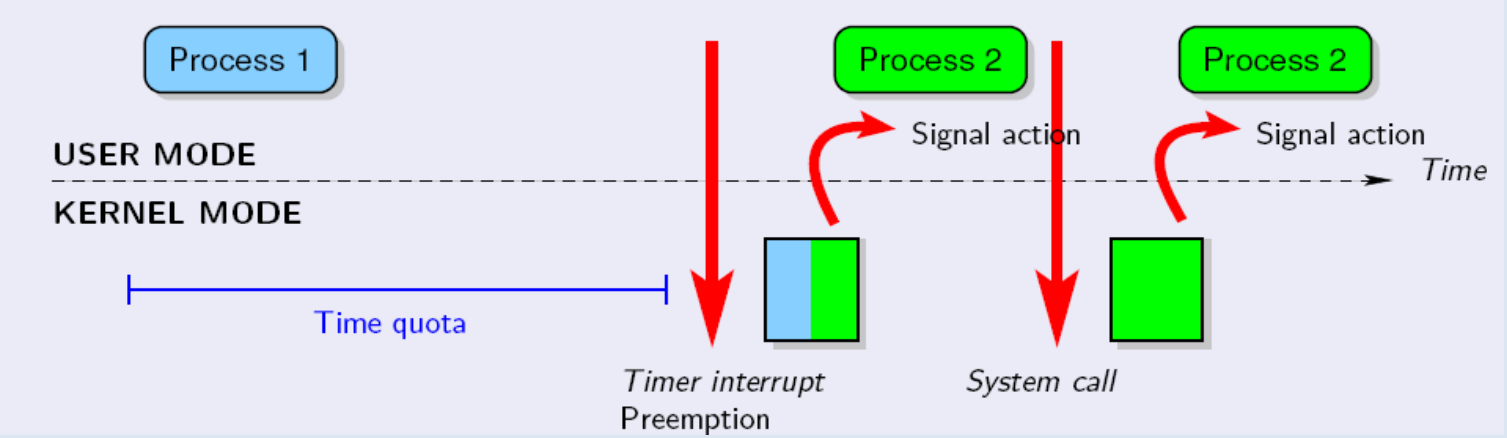
Obrada

- Signali su asinhroni mehanizam:
  - asinhroni slanje i prijem signala
  - signal se može poslati bez obzira da li se proces kome se šalje signal izvršava ili ne
  - prijem signala može prekinuti izvršavanje procesa u bilo kojoj tački
- Funkcija za obradu signala se poziva nakon pristizanja signala i izvršava se u korisničkom režimu.
- Signal čeka sve dok ne dođe red na njegovu obradu.

Signali

Detekcija signala (signal catching)

- Signal se detektuje prilikom prelaska iz režima jezgra u korisnički režim:
  - po završetku sistemskog poziva
  - context switching (promena aktivnog procesa)



Signali

Slanje signala

```
#include <signal.h>
#include <sys/types.h>
int kill (pid_t pid, int sig);
```

Semantika

- Sistemski poziv koji šalje signal specificiranom procesu ili grupi procesa.
- Prvi argument **pid** definiše procese kojima se signal šalje:
  - pid > 0 – signal se šalje procesu sa zadatim identifikatorom
  - pid == 0 – signal se šalje svim procesima koji pripadaju istoj grupi kao i proces koji šalje signal
  - pid < -1 – signal se šalje svim procesima koji pripadaju grupi –**pid**
  - pid == -1 – signal se šalje svim procesima kojima tekući proces može da pošalje signal isključujući njega samog i proces **init(1)**

Signali

Semantika

- Drugi argument **sig** definiše signal koji se šalje:
  - SIGHUP – prekinuta komunikacija terminalom
  - SIGINT – procesu se šalje signal Ctrl + C
  - SIGQUIT – procesu se šalje signal Ctrl + \
  - SIGKILL – signal koji prekida proces i koji se ne može blokirati
  - SIGBUS/SIGSEGV – greška na magistrali odnosno greška pri segmentaciji
  - SIGPIPE – write opoeracija nad datavodom iz koga nijedan proces ne čita podatke
  - SIGALRM – signal alarma
  - SIGTERM – signal za terminaciju procesa
  - SIGSTOP – signal koji suspenduje izvršavanje procesa
  - SIGTSTP- izvršavanje procesa u pozadini (Ctrl + Z)
  - SIGCONT – proces nastavlja izvršavanje nakon suspendovanja
  - SIGCHLD – proces dete je završilo sa izvršavanjem
  - SIGUSR1/SIGUSR2 – korisnički definisani signali

Signali

Semantika

- Ukoliko drugi argument ima vrednost 0 signal se ne šalje ali se vrši provera grešaka.
- Proces može da šalje signale procesima koje je kreirao isti korisnik ili koje su kreirali korisnici koji pripadaju istoj grupi.
- Procese koje kreira korisnik root mogu da šalje signale svim procesima.
- Funkcija kill vraća 0 ukoliko je signal uspešno poslat ondnosno -1 ukoliko je došlo do greške.

Signali

Obrada signala

```
#include <signal.h>
void (*signal(int signum, void(*handler)(int)))(int);
```

Semantika

- Sistemski poziv koji definiše kako će proces obrađivati određeni signal.
- Prvi argument **signum** predstavlja identifikator signala za koji se definiše način obrade.
- Drugi argument **handler** definiše način obrade:
  - SIG_DFL – podrazumevana obrada signala
  - SIG_IGN – signal se ignoriše
  - pokazivač na funkciju koja se poziva po pristizanju signala
- Sistemski poziv vraća pokazivač na funkciju koja je prethodno bila zadužena za obradu signala u slučaju uspeha odnosno SIG_ERR u slučaju greške.

Signali

Obrada signala

```
#include <unistd.h>
int pause();
```

Semantika

- Sistemski poziv koji suspenduje izvršavanje procesa sve do prijema nekog signala (bez obzira da li se po prijemu signala poziva neka funkcija za obradu signala ili se proces prekida).
- Ukoliko se signal ignoriše neće se nastaviti izvršavanje procesa.
- Funkcija uvek vraća -1.

Signali

Obrada signala

```
#include <unistd.h>
int alarm(unsigned int seconds);
```

Semantika

- Sistemski poziv koji tekućem procesu šalje signal SIGALRM posle specificiranog broja sekundi.
- Podrazumevana obrada za signal SIGALRM je prekid izvršavanja procesa.

Signali

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

#define N 10

void  do_nothing()
{
    return;
 }

int main()
{
  signal(SIGALRM, do_nothing);

  alarm(N);

  pause();

  signal(SIGALRM, SIG_DFL)
}
```

Funkcija koja se poziva za slučaj pojave signala.

U slučaju pojave signala SIGALRM poziva se funkcija do_nothing.

Zahteva se slanje alarma nakon isteka N sekundi.

Pauzira se izvršavanje procesa do pojave prvog signala.

Vraća se podrazumevana obrada signala SIGALRM.

Signali

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

#define K 30
int pid;

void kraj();
int main(int argc, char *argv[])
{
  signal(SIGALRM, kraj);
  pid = fork();
  if (pid == 0)
  {
    execle("charcount", "charcount",NULL, NULL);
  }
   else
  {
    alarm(10);
    pause();
  }
}
```

```c
void kraj()
{
  printf("Isteklo vreme\n");
  kill(pid, SIGKILL);
}


/*datoteka charcount.c*/
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
  FILE *f;
  char c;

  f = fopen("proba", "r");

   /*brojanje karaktera*/
  while(c=fgetc(f));
}
```

Sadržaj

- Signali
- Redovi poruka

Redovi poruka

Pojam

- Red poruka (message queue) obezbeđuje mehanizam za struktuiranu, loss-less (bez gubitaka) komunikaciju između procesa.
- Prilikom komunikacije korišćenjem redova poruka moguće je definisati prioritete.
- Red poruka predstavlja lančanu listu koja se održava u jezgru operativnog sistema nezavisno od procesa.
- Red poruka obezbeđuje struktuiranu komunikaciju odnosno očuvane su granice između poruka (za razliku od datavoda gde se podaci tretiraju kao niz bajtova).
-

Redovi poruka

Kreiranje reda poruka

```
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/msg.h>
int msgget(key_t key, int msgflg);
```

Semantika

- Sistemski poziv koji kreira novi red poruka.
- Prvi argument **key** predstavlja jedinstveni identifikator reda poruka na nivou čitavog sistema. Mora biti poznat svim procesima koji žele da koriste određeni red poruka.

Redovi poruka

Semantika

- Vrednost drugog argumenta **msgflg** se definiše kao rezultat OR operacije nad različitim vrednostima i određuje :
  - prava pristupa redu poruka (koristićemo vrednost 0666 koja svim korisnicima dodeljuje sve privilegije nad red poruka)
  - mod kreiranja reda poruka. Najčešća vrednost je:
    - IPC_CREAT – sistemski poziv kreira red poruka ukoliko on već ne postoji u sistemu.
- Sistemski poziv vraća celobrojni identifikator (referencu) reda poruka a u slučaju greške vraća (-1). Dobijeni identifikator (referenca) je važeći samo kod procesa koji je izvršio sistemski poziv i kod njegove dece.
- Ukoliko u sistemu ne postoji red poruka sa zadatim identifikatorom a specificiran je flag IPC_CREAT kreira se novi red poruka.
- Ukoliko u sistemu već postoji red poruka sa zadatim identifikatorom ne kreira se novi semafor već se samo vraća referenca na postojeći red poruka.

Redovi poruka

Slanje i primanje poruka

```
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/msg.h>
int msgsnd(int msgid, struct msgbuf * msgp, size_t msgsz, int msgflg);
int msgrcv(int msgid, struct msgbuf * msgp, size_t msgsz, int msgtyp, int msgflg);
```

Semantika

- Prvi argument **msgid** predstavlja identifikator (referencu) reda poruka koji je dobijen pozivom funkcije msgget.
- Drugi argument **msgp** predstavlja pokazivač na bafer koji sadrži poruku koja se šalje ili u koji će biti smeštena poruka koja se prima.
- Treći argument **msgsz** predstavlja veličinu bafera poruke u bajtovima.

Redovi poruka

Semantika

- Argument **msgflg** definiše mod slanja/prijema poruke a vrednost 0 označava prihvatanje podrazumevanog moda rada.
- Argument **msgtyp** kod funkcije za prijem je nenegativan ceo broj koji definiše tipove poruka koje se čitaju iz reda:
  - msgtyp == 0 – čitaju se sve poruke bez obzira na tip
  - msgtyp > 0 – čitaju se samo poruke specificiranog tipa
- Čitanjem se poruka uklanja iz reda poruka.

Redovi poruka

Poruka

```
struct msgbuf {
  long    mtype;      /* tip poruke, mora biti  > 0 */
  char    mtext[1];   /* tekst poruke */
};
```

- Polje **mtext** je niz (ili nek adruga struktura) koji definiše sadržaj poruke.
- Dozvoljene su poruke nulte veličine koje ne sadrže polje **mtext**.
- Polje **mtype** mora je pozitivan ceo broj koji omogućava procesu koji prima poruke da selektuje samo poruke od interesa.
- U funkcijama za slanje i prijem poruka argument **msgsz** definiše maksimalnu veličinu polja **mtext** u ovoj strukturi.

Redovi poruka

Kontrola reda poruka

```
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/msg.h>
int msgctl (int msqid, int cmd, struct msgid_ds buf);
```

Semantika

- Prvi argument **msgmid** predstavlja identifikator (referencu) reda poruka koji je dobijen pozivom funkcije msget.
- Drugi argument **cmd** definiše operaciju koju treba izvršiti nad semaforom. Za brisanje reda poruka se koristi operacija IPC_RMID.

Redovi poruka

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/msg.h>

#define QUEUE_KEY 10201
int main(int argc, char *argv[])
{
  int msqid;
  char msg[] = "Proba";

  msqid = msgget(QUEUE_KEY, 0666 | IPC_CREAT);
  if (msqid == -1)
    /*Obrada greske*/

  struct msgbuf buf;
  buf.mtype = 1;
 strcpy(buf.mtext, msg);

  if (msgsnd(msqid, &buf, 6, 0) == -1)
    /*Obrada greske*/
  else
   printf("Poruka poslat");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/msg.h>

#define QUEUE_KEY 10201
int main(int argc, char *argv[])
{
  int msqid;
  int msqlen = 10;
  char msg[];

  msqid = msgget(QUEUE_KEY, 0666 | IPC_CREAT);
  if (msqid == -1)
    /*Obrada greske*/

  struct msgbuf buf;

  if (msgrcv(msqid, &buf, msglen, 0, 0) == -1)
    /*Obrada greske*/
  else
   {
      strcpy(msg, buf.mtext, msglen);
      printf("Poruka poslat");
   }
}
```