SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE
PREDDIPLOMSKI STUDIJ RAČUNARSTVA

OPERACIJSKI SUSTAVI

VJEŽBE

Nastavnik: prof.dr.sc. Sven Gotovac
gotovac@fesb.hr

Asistent: Željko Šeremet
zeljko.seremet@fsre.sum.ba

MOSTAR, SVIBANJ 2024.

Sinkronizacija procesa semaforima

Sadržaj

- Semafori u UNIX-u
- Dobavljanje semafora
- Postavljanje početne vrijednosti semafora
- Uništavanje semafora
- Operacije sa semaforima
- Pomoćne funkcije za rad sa semaforima

Uvod (1)

- Sinkronizacija procesa se uglavnom obavlja semaforima.
- Ostali mehanizmi (čija primarna namjena nije sinkronizacija, poput redova poruka, cjevovoda, i sl.) rijetko se koriste za sinkronizaciju procesa.
- Semafori se također mogu koristiti i kod dretvi (ima i posebnih semafora za dretve), ali se kod dretvi najčešće koriste mehanizmi međusobnog isključivanja korištenjem tzv. mutexa te uvjetnih varijabli.

Uvod (2)

- Semafor je mehanizam koji sprečava da dva ili više procesa pristupaju zajedničkom sredstvu istovremeno.
- Binarni semafor ima dva stanja: propusno i nepropusno.
- Opći semafor ima beskonačan broj stanja (ili barem vrlo velik). To je brojač koji se smanjuje za jedan kada se zahtijeva semafor, a povećava se za jedan kada se oslobađa. Ako je na nuli, a proces zahtijeva semafor, tada taj proces mora čekati dok drugi proces ne poveća vrijednost semafora.
- Semafor u UNIX-u (System V) ne može imati negativnu vrijednost iako su teoretski ostvarivi i semafori kod kojih bi bile dozvoljene i negativne vrijednosti. Upotreba semafora se obično razmatra kroz dvije jednostavne operacije: "dohvati" i "otpusti" (*acquire* i *release*) ili "čekaj" i "postavi" (*wait* - *signal*, odnosno, izvorno iz nizozemskog: P i V).

Semafori u UNIX-u

- Pozivi za korištenje semafora u UNIX-u su daleko od jednostavnosti operacija dohvati i otpusti.
- Dapače, to su izuzetno složeni mehanizmi (vidi: **man semget**, **man semop**, **man semctl**).
- Zbog toga ćemo se ograničiti samo na dio funkcionalnosti poziva za rad sa semaforima koji je dovoljan za većinu primjena.

Dobavljanje semafora (1)

- Semafori se dobavljaju sustavskim pozivom *semget* slično kao i segment zajedničkog spremnika:

  int semget(key_t key, int nsems, int flags) ;
- Ovaj sustavski poziv dohvaća skup semafora koji se mogu kontrolirati svi odjednom ili stvara novi skup semafora. Novi skup sa ukupno *nsem* semafora će biti stvoren ako se za ključ upotrijebi IPC_PRIVATE.
- U devet najnižih bitova *flags* se stavljaju dozvole pristupa. *semget* vraća identifikacijski broj skupa semafora ili -1 u slučaju greške. Redni brojevi semafora u skupu počinju od nule.

Dobavljanje semafora (2)

- Zbog jednostavnosti bilo bi dobro imati uvijek samo po jedan semafor u skupu.
- Međutim, kao i kod zajedničkog spremnika, ukupan broj skupova semafora u sustavu je ograničen.
- Zbog velikog broja korisnika taj je broj lako premašiti, pa je poželjno sve potrebne semafore uvijek dobavljati odjednom, tj. u jednom skupu.

Postavljanje početne vrijednosti semafora (1)

- Kada se stvara novi skup semafora, vrijednosti svih semafora u skupu se postavljaju na 0. Ova vrijednost se može promijeniti sustavskim pozivom *semctl*, ali to je samo jedna od brojnih operacija koje on obavlja:

```
union semun {
int val;
struct semid_ds *buf;
unsigned short *array;};
int semctl(int semid, int semnum, int cmd, union
semun *arg) ; //Solaris
int semctl(int semid, int semnum, int cmd, union
semun arg) ;  //Ostali, više/manje
```

Postavljanje početne vrijednosti semafora (2)

- *semid* je uvijek identifikacijski broj skupa semafora. Vrijednost semafora se može postaviti na dva načina:
  - Ako je *semnum* redni broj semafora, *cmd* SETVAL, a *arg* nenegativan cijeli broj (*arg.val*), postavlja se vrijednost određenog semafora.
  - Ako je *cmd* SETALL, a *arg* kazaljka na niz kratkih cijelih brojeva bez predznaka (*arg.array*), postavljaju se svi semafori u skupu na vrijednosti iz danog niza.
- *semctl* pozvan na ovaj način vraća 0 ako je sve u redu ili -1 u slučaju greške (na primjer, vrijednost na koju treba postaviti semafor je prevelika ili negativna).

Uništavanje semafora

- Skup semafora treba uništiti nakon upotrebe jer u protivnom ostaje kao trajno zauzeto sredstvo u sustavu.
- Uništavanje semafora također se provodi sustavskim pozivom *semctl*.
- *semid* je identifikacijski broj skupa semafora, a *cmd* treba biti IPC_RMID. Ostali argumenti nisu bitni.

Operacije sa semaforima (1)

```
struct sembuf {
short sem_num;
short sem_op;
short sem_flg;
        };
int semop(int semid, struct sembuf (*sops)[],
  int nsops) ;
```

- *semop* se može upotrijebiti za nedjeljivo izvođenje niza operacija na skupu semafora. Međutim, najčešće je sasvim dovoljno izvesti jednu operaciju u jednom pozivu.

Operacije sa semaforima (2)

- U tom slučaju, *semid* je identifikacijski broj skupa semafora, *nsops* je 1, a *sops* je kazaljka na strukturu koja opisuje traženu operaciju. Unutar te strukture, *sem_num* je redni broj semafora u skupu, *sem_op* je tražena operacija koju dodatno opisuje i *sem_flg*.
- Za uobičajeno korištenje semafora dovoljne su dvije vrste operacija: povećavanje vrijednosti semafora (*sem_op* je pozitivan) i smanjivanje vrijednosti semafora (*sem_op* je negativan). U oba slučaja, *sem_flg* treba biti 0. *semop* vraća rezultat 0 ako je sve u redu, ili -1 u slučaju greške.

Operacije sa semaforima (3)

- Smanjivanje vrijednosti semafora odgovara operaciji "čekaj". Ukoliko je vrijednost semafora moguće umanjiti za zadani broj, a da ne postane negativna, vrijednost se umanjuje, a proces nastavlja rad. U suprotnom slučaju, proces čeka dok vrijednost semafora ne postane dovoljno velika da se umanjivanje može provesti.
- Povećavanje vrijednosti odgovara operaciji "postavi". Vrijednost semafora se povećava za zadani broj, a proces nastavlja rad. Proces zaustavljen u redu semafora se oslobađa ako je ovim povećanjem vrijednost semafora postala dovoljno velika da je on može umanjiti, a da ipak ne postane negativna.
- **Primjer_sema_proc.rar**

Zadaća 4.

- Modelirati vrtuljak (ringišpil) s dva tipa dretvi/procesa: dretvama/procesima *posjetitelj* (koje predstavljaju posjetitelje koji žele na vožnju) te dretvom/procesom *vrtuljak*. Dretvama/procesima *posjetitelj* se ne smije dozvoliti ukrcati na vrtuljak kada više nema praznih mjesta (kojih je ukupno *N*) te prije nego li svi prethodni posjetitelji siđu. Vrtuljak se može pokrenuti tek kada je pun. Za sinkronizaciju koristiti opće semafore i dodatne varijable.

Upute

```
Dretva posjetitelj() {
   ...
   sjedi;
   ...
   ustani; // ili sidji
   ...
}

Dretva vrtuljak() {
   dok je(1) {
      ...
      pokreni vrtuljak;
      zaustavi vrtuljak;
      ...
   }
}
```

Literatura

- Korisni linkovi:
- http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/7_Deadlocks.html
- http://www.cs.cf.ac.uk/Dave/C/node26.html
- http://www.classes.cs.uchicago.edu/archive/2008/fall/51081-1/LabFAQ/lab5/Semaphores.html

Kraj