Bilgisayar İşletim sistemleri Uygulama V

Unix'de Semafor İşlemleri

Semafor İşlemleri

- Unix'te semafor işlemlerinde kullanılacak başlık dosyaları
  - sys/ipc.h
  - sys/sem.h
  - sys/types.h
- Semafor yaratma
  - int semget(key_t key, int nsems, int semflg);
    - Semflg : IPC_CREAT|0700

Semafor İşlemleri

- Semafor üzerinde işlem gerçekleme

  int semop(int semid, struct sembuf *sops, unsigned nsops);

  struct sembuf

  ```
  {
     unsigned short sem_num;  /* numaralama 0'dan başlar*/
     short sem_op;
     short sem_flg;
  };
  ```

  – sem_flg:
    - SEM_UNDO: process sonlanınca işlemi geri al
    - IPC_NOWAIT: Eksiltemeyince hata ver ve dön
  – sem_op:
    - =0 : sıfır olmasını bekle (Okuma Hakkı olmalı)
    - ≠0: değer semafor değerine eklenir(çıkarılır) (Değiştirme hakkı olmalı)

Semafor İşlemleri

- Değer Kontrolü

  int semctl(int semid, int semnum, int cmd, arg);

  – cmd
    - IPC_RMID
    - GETVAL
    - SETVAL
    - SETALL
    - GETALL

Semafor İşlemleri

- Eksiltme İşlemi

```
void sem_wait(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=(-1*val);
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}
```

Semafor İşlemleri

- Arttırma İşlemi

```
void sem_signal(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=val;
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}
```

Örnek 1
Semafor İşlemleri

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define SEMKEY 8

int sonsem;

void signal12(void)
{}
```

```
void sem_signal(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=val;
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}

void sem_wait(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=(-1*val);
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}
```

Örnek 1

```
int main(void)
{
   int f=1, i;
   int cocuklar[10];


   signal(12,(void *) signal12);

   for (i=0; i<10; i++)
   {
      if (f>0)
         f=fork();
      if (f==-1)
      {
         printf("fork error....\n");
         exit(1);
      }
      if (f==0)
         break;
      else
         cocuklar[i]=f;
   }
```

Örnek 1

```
if (f>0) /*anne */
   {
      sonsem=semget(SEMKEY, 1, 0700|IPC_CREAT);
      semctl(sonsem, 0, SETVAL,0);

      sleep(1);

      for (i=0; i<10; i++)
         kill(cocuklar[i], 12);

      sem_wait(sonsem,10);

      printf("Tum cocuklar oldu\n");
      semctl(sonsem,0,IPC_RMID,0);
   }
```

Örnek 1

```
else /*cocuk */
   {
      pause();
      sonsem=semget(SEMKEY, 1,0);
      printf("Ben %d. sirada yaratilan cocugum.
   Kimligim=%d\n",i, getpid());
      printf("  su an .... %d\n",
   semctl(sonsem,0,GETVAL,0));
      sem_signal(sonsem,1);
   }
   return(0);
}
```

Örnek 2
Hatalı Kullanım ve Ölümcül Kilitlenme

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <sys/sem.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>

#define KEYSEM1 1
#define KEYSEM2 2
#define KEYSEM3 3

void sinyal12(void)
{}
```

```
void sem_signal(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=val;
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}

void sem_wait(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=(-1*val);
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}
```

Örnek 2

```
int main (void)
{
   int sem1,
   sem2,
   sonsem,
   c[2],
   f,
   i,
   siram;

   signal(12, (void *) sinyal12);
```

```
for (i=0; i<2; i++)
  {
     f=fork();
     if (f==0)
        break;
     else
        c[i]=f;
  }
  if (f==-1)
  {
     printf("FORK hata....\n");
     exit(1);
  }
```

Örnek 2

```
if (f!=0)
  {
     printf("Anne kaynaklari yaratmaya basliyor....\n");
     sem1=semget(KEYSEM1, 1, 0700|IPC_CREAT);
     semctl(sem1, 0, SETVAL,1);

     sem2=semget(KEYSEM2,1,0700|IPC_CREAT);
     semctl(sem2,0,SETVAL,1);

     sonsem=semget(KEYSEM3,1,0700|IPC_CREAT);
     semctl(sonsem,0,SETVAL,0);

     sleep(2);
     printf("Anne cocuklari baslatiyor .......\n");
     for (i=0; i<2; i++)
        kill(c[i],12);

     sem_wait(sonsem,2);

     printf("Anne: Cocuklarin isi bitti, kaynaklar iade ediliyor...\n");
     semctl(sonsem,0,IPC_RMID,0);
     semctl(sem1,0,IPC_RMID,0);
     semctl(sem2,0,IPC_RMID,0);
     exit(0);
  }
```

Örnek 2

```
else
  {
     siram=i;
     printf("cocuk %d anneden haber bekliyor ....\n", siram);
     pause();
     sem1=semget(KEYSEM1,1,0);
     sem2=semget(KEYSEM2,1,0);
     sonsem=semget(KEYSEM3,1,0);
     printf("cocuk %d anneden haber aldi, basliyor ....\n", siram);
     if (siram==0)
     {
        printf("cocuk %d: sem1 eksiltiyorum.\n", siram);
        sem_wait(sem1,1);
        sleep(1);
        printf("cocuk %d: sem1 tamam, sem2 eksiltiyorum.\n", siram);
        sem_wait(sem2,1);
        printf("cocuk %d: kritik bolgemdeyim.\n", siram);
        sleep(5);    /* K.B. islemleri */
        sem_signal(sem2,1);
        sem_signal(sem1,1);
        sem_signal(sonsem,1);
     }
```

Örnek 2

```
else if (siram==1)
      {
         printf("    cocuk %d: sem2 eksiltiyorum.\n", siram);
         sem_wait(sem2,1);
         sleep(1);
         printf("    cocuk %d: sem2 tamam, sem1 eksiltiyorum.\n", siram);
         sem_wait(sem1,1);
         printf("    cocuk %d: kritik bolgemdeyim.\n", siram);
         sleep(5);    /* K.B. islemleri */
         sem_signal(sem1,1);
         sem_signal(sem2,1);
         sem_signal(sonsem,1);
      }
   }
   return 0;
}
```

Örnek 3
Ölümcül Kilitlenmenin Giderilmesi

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <sys/sem.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>

#define KEYSEM1 1
#define KEYSEM2 2
#define KEYSEM3 3

void sinyal12(void)
{}
```

```
void sem_signal2(int semid, int val, int
     nsems)
{
   struct sembuf semafor[2];
   int i;


   for (i=0; i<nsems; i++)
   {
      semafor[i].sem_op=val;
      semafor[i].sem_flg=1;
      semafor[i].sem_num=i;
   }

   /*bir semafor seti üzerinde aynı anda iki
     işlem yapılıyor*/
   semop(semid, semafor, 2);

   for (i=0; i<nsems; i++)
   {
      printf("signal : %d su an .... %d\n", i,
     semctl(semid,i,GETVAL,0));
   }

}
```

Örnek 3

```
void sem_wait2(int semid, int val, int nsems)
{
   struct sembuf semafor[2];
   int i;

   for (i=0; i<nsems; i++)
   {
      semafor[i].sem_op=(-1*val);
      semafor[i].sem_flg=1;
      semafor[i].sem_num=i;
   }

   /*bir semafor seti üzerinde aynı anda iki
     işlem yapılıyor*/
   semop(semid, semafor,2);

   for (i=0; i<nsems; i++)
   {
      printf("wait   : %d su an .... %d\n", i,
     semctl(semid,i,GETVAL,0));
   }
}
```

```
void sem_signal(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=val;
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}

void sem_wait(int semid, int val)
{
   struct sembuf semafor;

   semafor.sem_num=0;
   semafor.sem_op=(-1*val);
   semafor.sem_flg=1;
   semop(semid, &semafor,1);
}
```

Örnek 3

```
int main (void)
{
   int sem,
   sonsem,
   c[2],
   f,
   i,
   siram;

   signal(12, (void *) sinyal12);


   for (i=0; i<2; i++)
   {
      f=fork();
      if (f==0)
         break;
      else
         c[i]=f;
   }
   if (f==-1)
   {
      printf("FORK hata....\n");
      exit(1);
   }
```

Örnek 3

```
if (f!=0)
  {
     printf("Anne kaynaklari yaratmaya basliyor....\n");
     sem=semget(KEYSEM1, 2, 0700|IPC_CREAT);
     semctl(sem, 0, SETVAL,1);
     semctl(sem, 1, SETVAL,1);

     sonsem=semget(KEYSEM3,1,0700|IPC_CREAT);
     semctl(sonsem,0,SETVAL,0);

     sleep(2);
     printf("Anne cocuklari baslatiyor .......\n");
     for (i=0; i<2; i++)
        kill(c[i],12);

     sem_wait(sonsem,2);

     printf("Anne: Cocuklarin isi bitti, kaynaklar iade ediliyor...\n");
     semctl(sonsem,0,IPC_RMID,0);
     semctl(sem,0,IPC_RMID,0);
     exit(0);
  }
```

Örnek 3

```
else
  {
     siram=i;
     printf("cocuk %d anneden haber bekliyor ....\n", siram);
     pause();
     sem=semget(KEYSEM1,2,0);
     sonsem=semget(KEYSEM3,1,0);
     printf("cocuk %d anneden haber aldi, basliyor ....\n", siram);

     printf("cocuk %d: sem eksiltiyorum.\n", siram);
     sem_wait2(sem,1,2);
     printf("cocuk %d: kritik bolgemdeyim.\n", siram);
     sleep(5);    /* K.B. islemleri */
     sem_signal2(sem,1,2);
     sem_signal(sonsem,1);
  }
  return 0;
}
```

- Her iki çocuk da, ilgili semafor grubundaki iki semaforu birden eksiltip arttırır
  - Semafor işlemlerinin başladığı bloğa bir çocuk girdiği zaman, diğer çocuk bekler
    - Ölümcül Kilitlenme Olasılığı Yok...