BLGM 343 – DENEY 6*
MESAJ KUYRUKLARI

Amaçlar

1. UNIX benzeri işletim sistemlerinde, işlemler arası veri alış verişinin kavranması
2. Mesaj kuyrukları hakkında çalışma yapılması

Ön Bilgi

Mesaj kuyruğu, Unix benzeri işletim sistemlerinde kullanılan işlemler arası bir iletişim mekanizmasıdır. Mesaj kuyruğu herhangi bir işlem tarafından istenilen bir anahtar atanarak oluşturulabilir. Oluşturulan kuyruk sistem çekirdeği tarafından tutulur. İsteyen her işlem (yetkiler göz önünde bulundurularak) kuyruk anahtarını kullanarak, herhangi bir mesaj kuyruğuna erişebilir. Böylece iletişim kuracak olan işlemlerin bu anahtar değerini bilmeleri yeterlidir. Ayrıca, kuyrukta bulunan her mesajın bir veri tipi ve içeriği (veri) vardır. Böylece, mesajı okuyan işlemler, mesajın içeriğinde bulunan veri tipini tespit edebilirler. Tünellerle karşılaştırıldığında, mesaj kuyrukları daha fazla özellik ve serbestlik sunmaktadırlar. Ayrıca paylaşılan bellek aracılığıyla veri transferine göre, yetkilendirme sistemi bulunduğu için daha güvenlidirler.

Mesaj kuyruğu, dosyalar gibi erişim haklarına sahiptir, ve kullanılabilmesi için open benzeri bir fonksiyon olan **msgget()** çağrısıyla açılması/yaratılması gerekmektedir. **msgget()** sistem çağrısı ikiparametreye ihtiyaç duymaktadır. Birinci parametre, kuyruğun anahtarı (tam sayı bir değer), ikinci parametre ise seçeneklerdir. Eğer yeni bir mesaj kuyruğu oluşturuluyorsa, IPC_CREAT değeriyle dosya yetkisi birleştirilerek seçenek olarak gönderilmelidir. Örneğin: IPC_CREAT | 0666. Eğer mesaj kuyruğu mevcutsa seçenek olarak 0 değeri verilebilir. **msgget** sistem çağrısından dönen değer eğer -1 ise hata vardır, eğer pozitif (veya 0) bir değer verilirse, bu değer mesaj kuyruk tanımlayıcısıdır. Kullanımı aşağıdaki gibidir:

```
tanimlayici = msgget(anahtar, secenekler);
```

msgsend() sistem çağırısıyla, bir mesaj kuyruğa eklenebilir. Kuyruktan bir mesaj almak için ise **msgrcv()** sistem çağrısı kullanılabilir. **msgctl()** sistem çağrısı kullanılarak bir mesaj kuyruğu kapatılabilir.

* BLGM 343 dersi için Güz 2013/2014 döneminde Gürcü Öz ve Cem Kalyoncu tarafından hazırlanmıştır

Sistemdeki mesaj kuyrukları hakkında bilgi almak için **ipcs** komutunu, bir mesaj kuyruğunu silmek için ise **ipcrm** komutunu kullanabilirsiniz.

Deneyler

1. Linux hesabınıza giriş yaparak gerekli dosyayı bölüm sitesinden indiriniz. İndirdiğiniz arşivdeki dosyaları, daha sonra kolayca bulabileceğiniz şekilde bir dizine yerleştiriniz. Daha sonra sağlanmış olan kaynak kodunu derleyiniz.

2. **sender.c** programını 5-10 kere farklı bilgilerle çalıştırınız. Bu girdiğiniz bilgileri defterinize not ediniz.

3. Bu sefer, **receiver.c** programını çalıştırarak daha önce kullandığınız anahtar ve mesaj tiplerini vererek elde ettiğiniz değerleri gösteriniz.

4. ilk programı değiştirerek her 3 saniyede bir aynı mesajı eklemesini sağlayınız. İkinci program ise sürekli bir döngü içerisinde çalışarak elde ettiği tüm mesajları ekrana yazdırsın.

Programlarınızın sonlanması için bir mekanizma ekleyiniz. Örneğin programlar 5 mesaj işledikten sonra durabilirler.

5. ilk programı değiştirerek her 3 saniyede bir klavyeden mesaj eklenmesini sağlayınız. İkinci program ise sürekli bir döngü içerisinde çalışarak elde ettiği tüm mesajları ekrana yazdırsın.

6. Son olarak, adım 5'e ek olarak ikinci programın kuyruğa gönderilmiş tipi belli bir mesajı ekrana yazdırmasını sağlayınız.

Not: Eğer **msgrcv()** sistem çağrısında, tipi bilgisi 0 olursa, bu çağrı kuyruktaki sıradaki

mesajı okur.

Not: sleep() sistem çağrısının kullanarak programınızın beklemesini sağlayabilirsiniz.

Kaynakça

1. Curry, D., UNIX Systems Progr. for SVR4, O'Reilly & Assoc., 1995, pp. 353 - 363.

2. Haviland, K., et al, UNIX System Programming, 2nd ed., 1999, pp. 151 - 171.

Sorular

1. Mesaj kuyruklarının kullanımı için gerekli olan sistem çağrılarını listeleyiniz.

2. Mesaj kuyruğuyla ilgili veri yapıları nelerdir?

3. Mesaj kuyruğuna gönderilen bir mesajda ne gibi bilgiler olmalıdır?

4. Mesaj kuyruğu ile iletişim kuracak olan işlemlerin bilmeleri gereken bilgiler nelerdir?

5. Bir mesaj kuyruğunun kullanılabilmesi için işlemlerin bir birleriyle ilgili olaması gerekmekte midir?

6. İki işlemin bir mesaj kuyruğu kullanımı için yapması gerekenleri anlatınız.

Ek: 1. Program kodları

receiver.c

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

#define MESAJUZUNLUGU 1024

struct msgbf {
    long mtype;
    char mtext [MESAJUZUNLUGU];

};

int main (int argc, char *argv[]) {
      int kuyrukno, v; int i;
      struct msgbf mess;
      if (argc != 3) {
            printf ("Kullanım: <anahtar> <tip>\n");
            exit (1);
      }
      /* Mesaj kuyruğunu aç */
      kuyrukno = msgget ((key_t) atoi (argv[1]), 0);
      if (kuyrukno == -1) {
            perror("Mesaj kuyruğu açılamıyor");
            exit (1);

      }
      /* Berlitilen tipteki mesajları kuyruktan oku */

                  v = msgrcv(kuyrukno,(struct msgbuf *)&mess, 100,
                  atoi(argv[2]), IPC_NOWAIT);
            if (v < 0){
                  if (errno == ENOMSG) {
                        printf("İstenen tipte mesaj kuyrukta yok!\n");
                        exit(2);

                  }
                  else {
                        perror("Kuyruktan okunamıyor");

                  }
            }
```

```
            else printf ("[%d]  %s\n", mess.mtype, mess.mtext);


      if (msgctl(kuyrukno,IPC_RMID, 0) <0)
            perror("Mesaj kuyruğu yok edilemiyor");
      else

            printf("%ld anahtarına sahip mesaj kuyruğu yok edildi!\n",

            atoi(argv[1]));
      exit(0);
}
```

sender.c

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

#define MESAJUZUNLUGU 1024

struct msgbf {
    long mtype;
    char mtext [MESAJUZUNLUGU];

};

int main (int argc, char *argv[]) {

      int kuyrukno, v;
      int i;
      struct msgbf mess;
      if (argc != 4) {
      printf ("Kullanım: <anahtar> <tip> <yazı>\n");
      exit(1);

      }
      /* Mesaj kuyruğunu yaratma */
      kuyrukno = msgget ((key_t) atoi (argv[1]), IPC_CREAT | 0666);
      if (kuyrukno == -1) {
            perror("Kuyruk oluşturulamıyor");
            exit(1);

      }

      /* Parametreleri kullanarak mesajı hazırla */
      mess.mtype = atoi (argv[2]);
      strcpy (mess.mtext, argv[3]);
      printf("Gönderiliyor...\n");

      /* Mesajı kuyruğa yaz */

            v = msgsnd(kuyrukno, (struct msgbuf *)&mess,
            strlen(argv[3]) + 1, 0);

            if (v < 0)
                  perror("Kuyruğa yazma hatası");
      printf("Gönderici görevini tamamladı\n");
      exit(0);
```

```
}
```