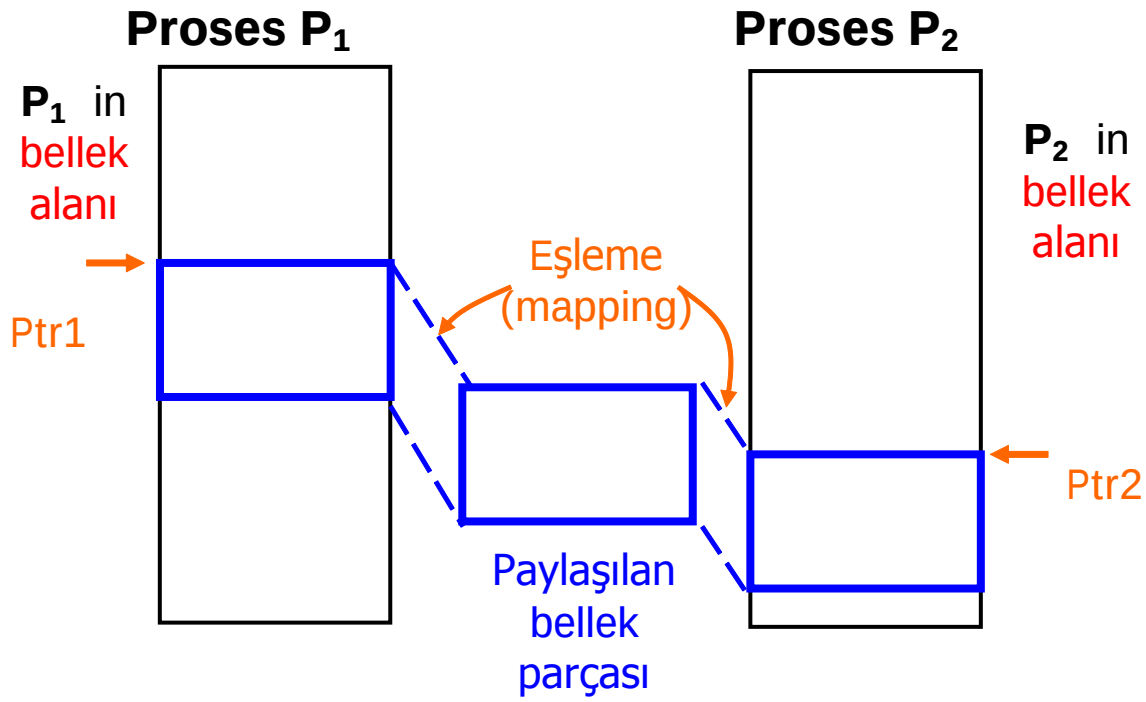
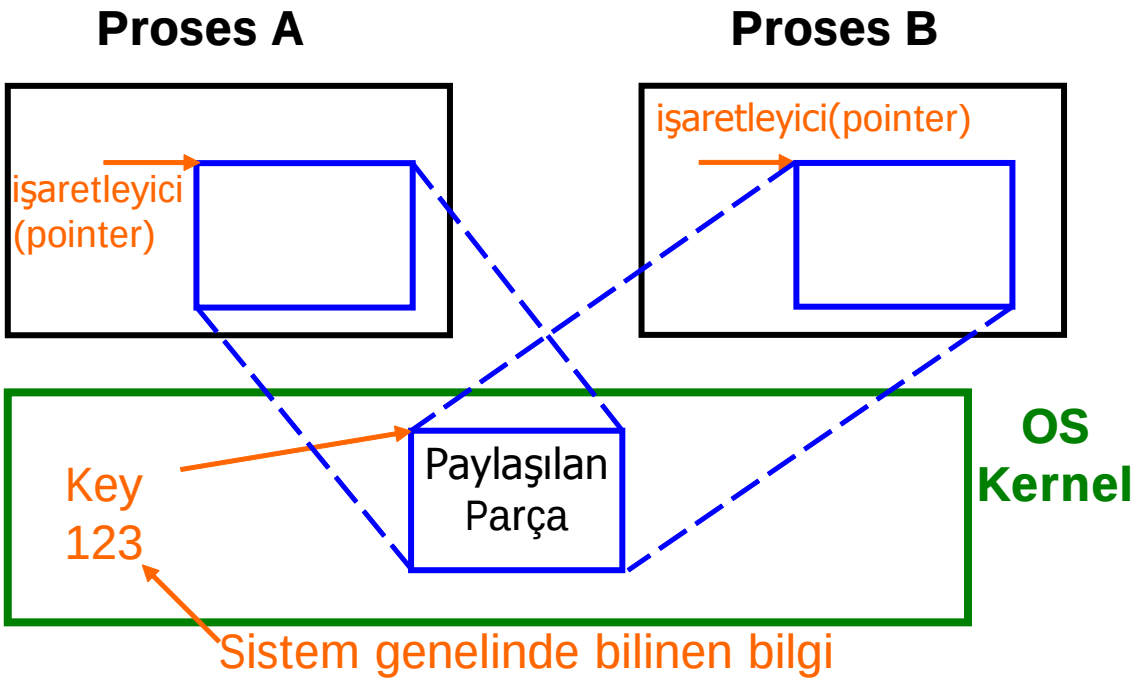
Paylaşılan bellek(shared memory) kavramını anlama



Normalde, UNIX bir kullanıcı prosesinin başka bir prosesin bellek alanında bulunan herhangi bir veriye ulaşmasına izin vermez.

UNIX de bir IPC mekanizması olarak paylaşılan bellek



Paylaşılan belleğin yaratılma ve kullanılma adımları:

1. Proseslerden biri paylaşılan belleği yaratır. Diğer prosesler ise yaratılan bellek parçasının ID sini alır.
2. Her bir proses, ayni anahtarı (key) kullanarak kendisini yaratılan bellek parçasına bağlar.
3. Artık her bir proses paylaşılan bellek parçasına yazabilir ve okuyabilir . Olası yarış durumunu (race condition) önlemek için ortak bellek kullanılırken prosesler koordine edilmelidirler.
4. Her proses paylaşılan bellek parçasından ayrılır (işi bittikten sonra).

Paylaşılan bellekler için sistem çağrıları

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

1. Paylaşılan bellek yaratma veya ID sini alma

Sayısal anahtar int tipinde

Parçanın boyutu, bytes olarak, int tipinde

```
id = shmget ( key, size, flag ) ;
```

int

IPC_CREAT | 0666
Yeni bellek parçası yaratmak için rw-rw-rw- izinleri ile veya **0** olan parçanın ID sini almak için

Paylaşılan bellek ayrıldı fakat eşleşme yapılmadı

2. Var olan bellek parçasını prosesin belleğine iliştirme (veya eşleştirme) bellek parçasının işaretleyicisini alma

shmget() ten dönen değer

Bağlanacak adres (genellikle **0** dır, sistemin seçmesi istenir)

```
ptr = shmat ( id, addr, flag ) ;
```

Paylaşılan bellek parçasının başlangıç adresi

Genellikle **0**, veya SHM_RDONLY

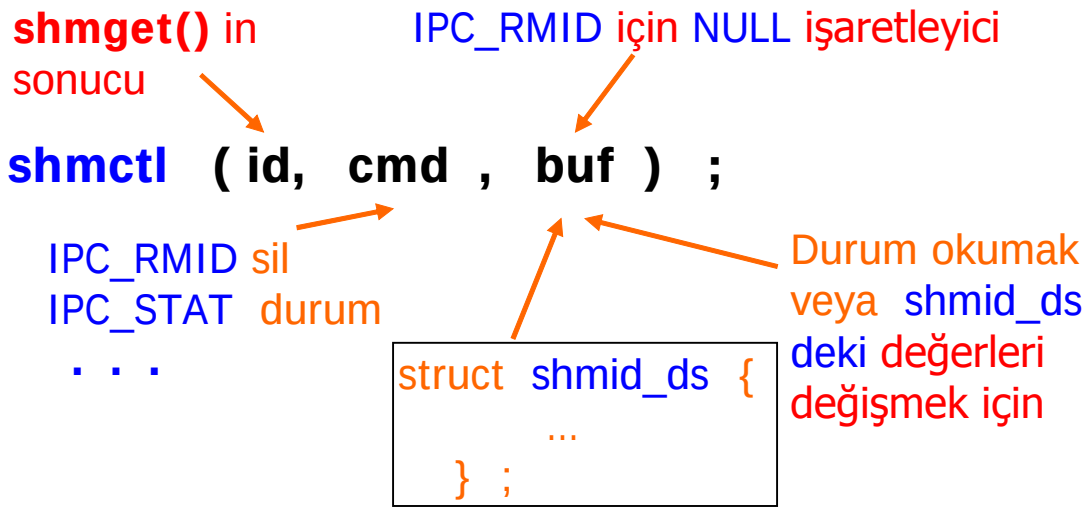
Bu işlemle paylaşılan bellek prosese bağlanmış olur

3. Paylaşılan bellek parçasını proses ten ayırmak

shmat() ın dönen değeri

```
shmdt ( ptr ) ;
```

4. Paylaşılan bellek parçası üzerinde kontrol işlemleri



Paylaşılan belleğin özellikleri

(a) + Verimlilik(Efficiency) (tünellerde olduğu gibi verinin ara kopyalaması yok)

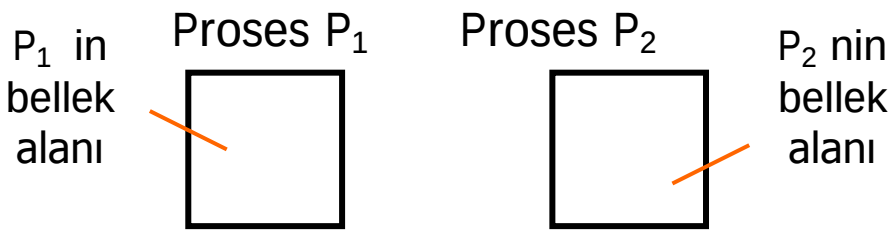
(b) + Rastgele giriş(Random access) (tünellerde sıralı byte akışı olur)

(c) + IPC nin çoktan-çoğa(Many-to-many) mekanizması: bir çok proses ayni bellek parçasına bağlanabilir (tüneller IPC nin bire–bir(one-to-one) mekanizmasıdır)

(d) – Senkronizasyon sağlama özelliği yoktur (tüneller senkronizasyon sağlar)

Paylaşılan bellek üzerindeki operasyonlar

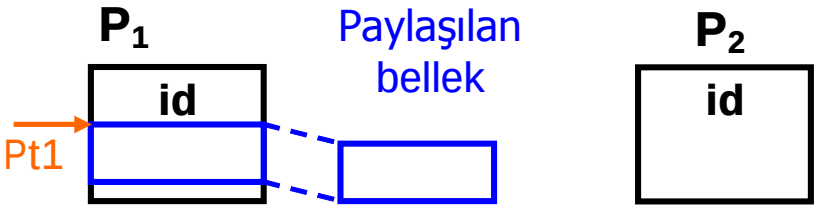
1. Başlangıç durumu



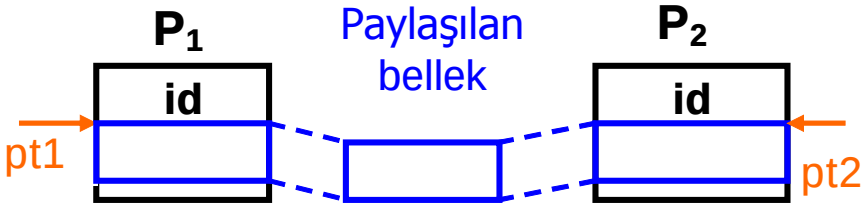
2. P_1 id=shmget()' i çalıştırdıktan sonra

3. P_2 id=shmget()'i çalıştırdıktan

4. P_1 pt1=shmat()'ı çalıştırdıktan sonra



5. P_2 pt2=shmat()'ı çalıştırdıktan sonra



Artık her iki proses de paylaşılan belleği kendi belleklerinin bir parçasıymış gibi kullanabilirler.

Paylaşılan belleğin kullanımına bir örnek

Proses A

```
#include ...
#define  MSIZ  27
main()
{  char  c ;  int  shmid  ;
    key_t  key = 123 ;
    char  *shm,  *s ;

    if ( (shmid = shmget(
key,MSIZ,IPC_CREAT | 0666) )
< 0 )
   {perror(...); exit(1) ; }

    if ( (shm = shmat( shmid,
NULL, 0)) == (char *) –1 )
    (perror(...) ; exit(1) ;  }

/*Veriyi paylaşılan belleğe koy */
   s = shm ;
   for (c = 'a' ; c <= 'z' ; c++ )
        *s++ = c ;
    *s = '\0'  ;

/*Proses B ilk karakteri
değiştirene kadar bekle */

while(*shm != '*' ) sleep( 1 );

  shmdt( shm ) ;
    exit( 0 ) ;
}
```

Proses B

```
#include ...
#define  MSIZ  27
main()
{  int  shmid  ;
    key_t  key = 123 ;
    char  *shm,  *s ;

  if ( (shmid = shmget(
key,MSIZ, 0 )) < 0 )
                or 0
       {perror(...) ; exit(1) ; }

if ( (shm = shmat( shmid,
NULL, 0)) == (char *) –1 )
    (perror(...) ; exit(1) ;  }

/* paylaşılan bellekten veri al  */

 for (s=shm ; *s != '\0' ; s++ )
      putchar( *s ) ;  / *
göstermek için */
    putchar( '\n') ;

/* Bellek parçasındaki ilk
karakteri değiştir */
    *shm = '*' ;
   shmdt ( shm ) ; /* ayrıl */
    exit( 0 ) ;
}
```