UNIX de dosyalar için temel sistem çağrıları

Not: eklenmesi gereken kütüphane dosyası #include <fcntl.h>

Sistem çağrısı ismi	Kullanım amacı	Dönen değerler
open	Bir dosyayı okuma amaçlı yazma amaçlı veya her iki amaç için açar.	Dosya tanımlayıcı veya -1
creat	Yeni (boş) bir dosya açar.	Dosya tanımlayıcı veya -1
read	Açılan dosyadan okur arabelleğe(buffer) yazar.	Okunan gerçek karakter sayısı veya -1
write	Veriyi(data) arabellekten dosyaya yazar.	Yazılan gerçek karakter sayısı veya -1
close	Önceden açılan(veya yaratılan) dosyayı kapatır.	**0** (sıfır) veya -1
unlink	Dosyayı ortadan kaldırır, siler	**0(sıfır) veya -1**

Dosyalar ile ilgili system çağrıları içeren bir C programının yapısı

```
#include <fcntl.h>          ←!!!!!!!!!!!!!!!!!!!
  /*  Diğer gerekli #include  satırları   */

int main()
{
      /*  Dosya sistemi ile ilgili system çağrıları  */
}
```

VEYA

```
#include <fcntl.h>          ←  !!!!!!!!!!!!!!!!!!
  /*  Diğer gerekli  #include  satırları  */

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

{
      /*  Dosya sistemi ile ilgili system çağrıları  */
}
```

open()

```
#include <stdio.h>     /* perror() için  */
#include <fcntl.h>

char  *isim = "/tmp/abc" ;
main()
{  int  fd ;

   . . .        veya:  fd = open("/tmp/abc",O_WRONLY);

   fd = open(isim, O_WRONLY) ;

   if  (  fd < 0 )
     {  perror( isim ) ;

        exit( 1 ) ;
     }
. . . . .
}
```

diğer etiketler(flags):
O_RDONLY
O_RDWR

isim değişkeninin değerini ve sistemin verdiği hata mesajını yazdırır

Örneğin:

/tmp/abc: Dosya mevcut değil

open() sistem çağrısının yeni bir dosya yaratmada kullanımı

```
fd =open("dosya_x", O_WRONLY|O_CREAT|O_APPEND, 0644);
```

İşleyiş:

1. Eğer dosya "dosya_x" bulunduğum dizinde varsa, yazma amaçlı(ekleme modunda) açılır.

2. Eğer "dosya_x" yoksa 0644 izinleri ile yaratılır.

creat() çağrısının kullanımı

```
fd = creat("yeni_dosya", 0644);
```

"yeni_dosya", 0644 izinleri ile yaratılır.

read(), write() sistem çağrıları

```
dönen_sayı = read( fd, buffer, sayı) ;
dönen_sayı = write( fd, buffer, sayı) ;
```

Örnek:

```
#include <stdio.h>          /* perror() için  */
#include <fcntl.h>

int main()
{  int    fd ;  int   okunan_sayı  ;
    char  buf[512]  ;
    .  .  .  .  .
    fd = open("<yol_ismi>", O_RDONLY ) ;
    .  .  .  .  .
    okunan_miktar = read( fd, buf, 512) ;
    if ( okunan_miktar < 0 )
      { perror("<yol_ismi>") ;
        exit(1) ;
      }
   if ( okunan_miktar == 0 )
      { printf("Dosyada daha fazla veri yok\n") ;
        exit(0) ;
      }
}
```

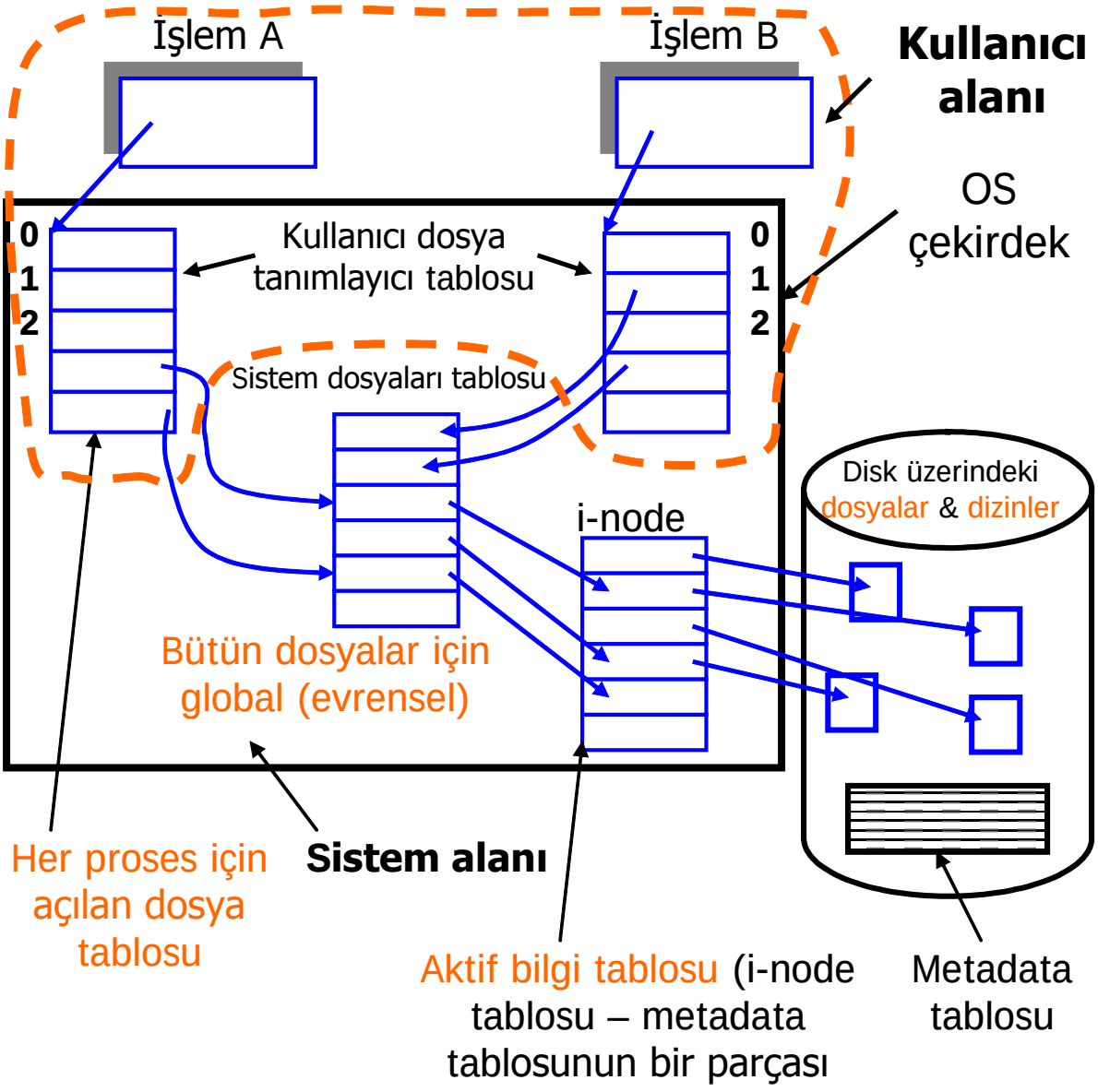
Bloklar halinde bir dosyayı kopyalama (kısmi bir program)

```
#include <fcntl.h>

char *kaynak = “dosya_x”;        /* bana ait dizinde */
char *kopya = “/tmp/kopya” ;

main()
{
   int k_tanımlayıcı, y_tanımlayıcı, okunan miktar ;
   char buffer[100] ;  /* byte blokları için geçici bellek */
   k_tanımlayıcı = open(kaynak, O_RDONLY) ;
   y_tanımlayıcı = creat(kopya, 0644) ;

/*  kopyalama  */

while ( (okunan_miktar = read(k_tanımlayıcı, buffer,
100)) > 0 )
   {  write(y_tanımlayıcı, buffer, okunan_miktar) ; }

close(k_tanımlayıcı) ;
close(y_tanımlayıcı);
}
```

veya

```
  . . .
okunan_miktar = 1 ;
for ( ; okunan_miktar>0 ; )
{
    okunan_miktar = read(k_tanımlayıcı, buffer, 100) ;
    write(y_tanımlayıcı, buffer, okunan_miktar) ;
}
```

UNIX de işlemler ve dosyalar



Metadata: Bir başka veri hakkındaki veriler.

open() sistem çağrısı için örnekler

```
fd1 = open("data", O_RDONLY) ;
fd2 = open("info", O_RDWR) ;
fd3 = open("data", O_WRONLY) ;
```

Proses A (kullanıcı prosesi)

Sonuç:

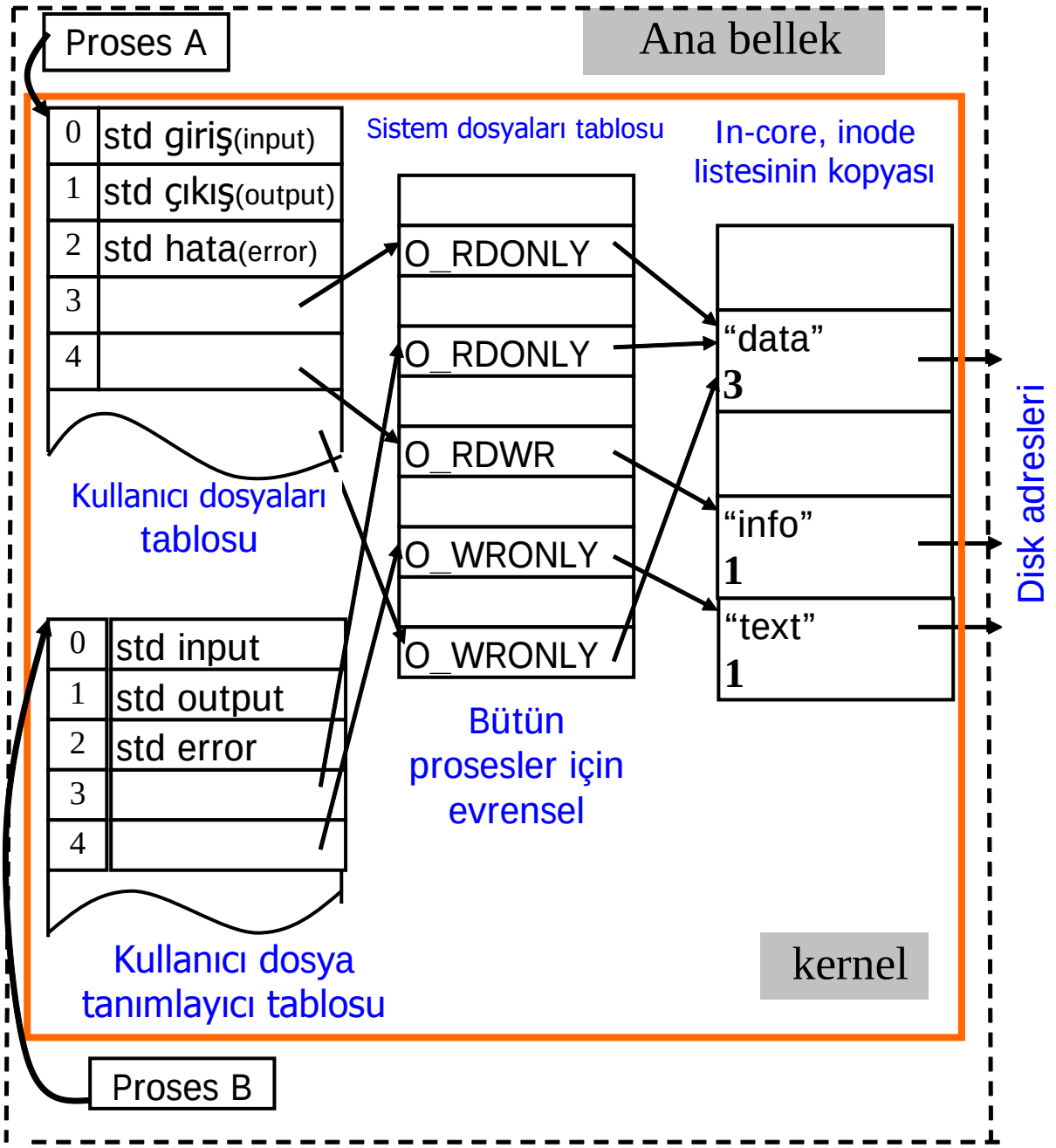
fd1 = 3, fd2 = 4 , fd3 = 5 (proses A içinde)

```
fd1 = open("data", O_RDONLY) ;
fd2 = open("text", O_WRONLY) ;
```

Proses B (kullanıcı prosesi)

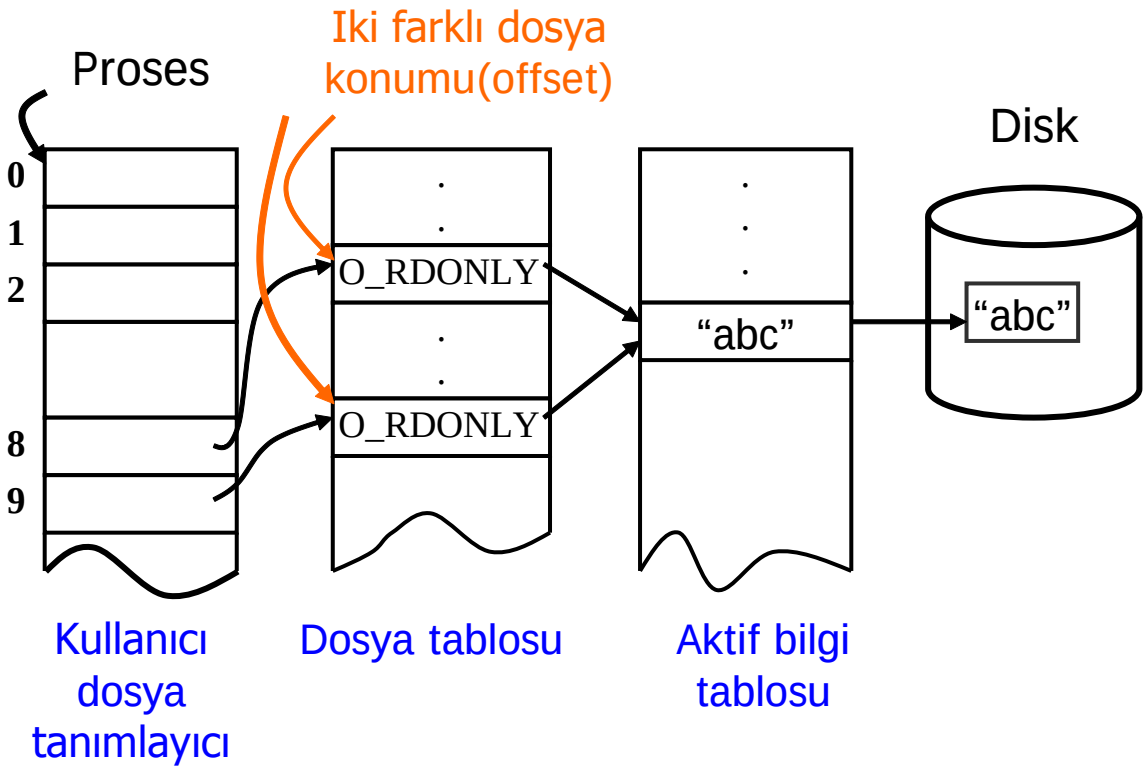
Sonuç:

fd1 = 3 , fd2 = 4 (proses B içinde)


Proses A
Ana bellek
0 std giriş(input)
1 std çıkış(output)
2 std hata(error)
3
4
Kullanıcı dosyaları tablosu
Sistem dosyaları tablosu
O_RDONLY
O_RDONLY
O_RDWR
O_WRONLY
O_WRONLY
Bütün prosesler için evrensel
In-core, inode listesinin kopyası
"data"
3
"info"
1
"text"
1
Disk adresleri
0 std input
1 std output
2 std error
3
4
Kullanıcı dosya tanımlayıcı tablosu
kernel
Proses B

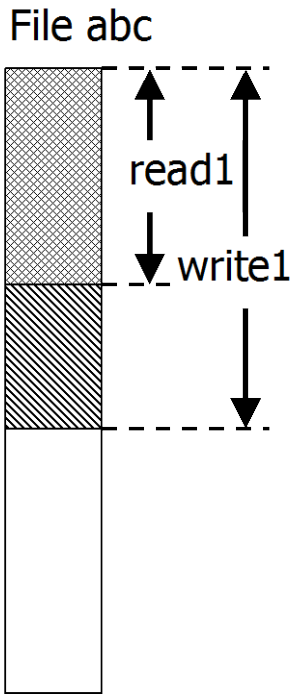
Bir dosyadan iki dosya tanımlayıcı ile okuma

```
#include <fcntl.h>
main()
{  int  fd1, fd2  ;
   char  buf1[512], buf2[512] ;
   . . . . . .
   fd1 = open("abc", O_RDONLY ) ;
   fd2 = open("abc", O_RDONLY ) ;
   read( fd1, buf1, 512 ) ;
   read( fd2, buf2, 512 ) ;
   . . . . .
}
```



İki proses tarafından bir dosyadan okuma ve yazma

```
#include <fcntl.h>
main()                  /* proses A */
{  int  fd  ;
    char  buf[300] ;
    fd = open("abc", O_RDONLY) ;
    read( fd, buf, 300 ) ;     /* 1 */
    read( fd, buf, 300 ) ;     /* 2 */
}
```



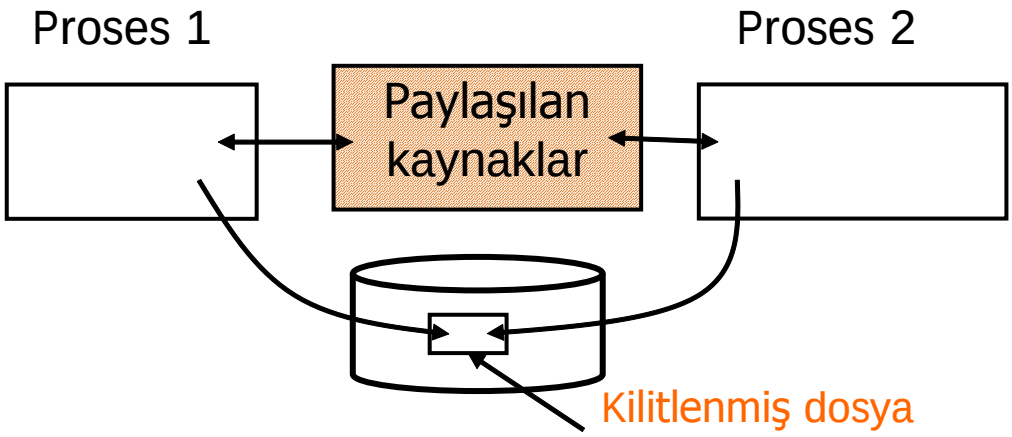

```
#include <fcntl.h>
main()                  /* proses B */
{  int  fd, i ;
    char  buf[512] ;

    for ( i = 0  ;  i < 512 ; i++ )
      buf[i] = 'C'  ;
    fd = open("abc", O_WRONLY) ;
    write( fd, buf, 512) ;  /*  1  */
    write( fd, buf, 512) ;  /*  2  */
}
```

Olası okuma ve yazma sıraları:

read1, read2, write1, write2 ;
read1, write1, read2, write2 ;
. . .

} /* 6 sequences */

İki veya daha fazla prosesin kilitlenmiş bir dosyayı kullanımı

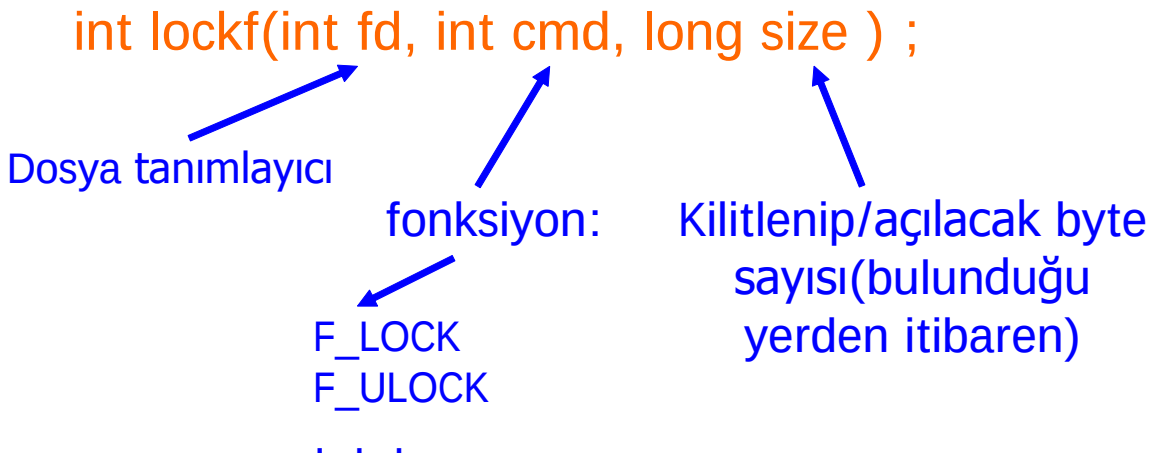



```
#include <errno.h>
#include <fcntl.h>
      /*  diğer dahil edilen dosyalar  */
#define  LOCKF    "/tmp/abc"

main()
{  int  fd ;
```

Kimseye müsade etme

```
    while ( (fd = creat( LOCKF, 0 ))  ==-1 )  sleep(1)  ;
    close( fd ) ;

    /* paylaşılan dosyayı kullan  */

    unlink( LOCKF ) ;     /* dosyayı sil  */
}
```

Bir dosyayı kilitlemek için kütüphane fonksiyonu *lockf()* in kullanımı




```
#include <fcntl.h>
. . . . . .
main()                                  /*dosya halehazırda vardır */
{  int  fd ;   int  i  ;

    fd = open("abc", O_RDWR) ;
                                        /* tüm dosyayı kilitle */

   i = lockf( fd, F_LOCK, 0 ) ;
    if ( i == -1 )  { perror( ...) ;  exit( 1 ) ;  }

            /* kilitlenen dosyayı kullan  */

    lockf( fd, F_ULOCK, 0 ) ;           /* dosyayı açma  */
}
```