Sistem Programlama

Ders 7

Doç. Dr. Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Dosya belirteci kopyalama

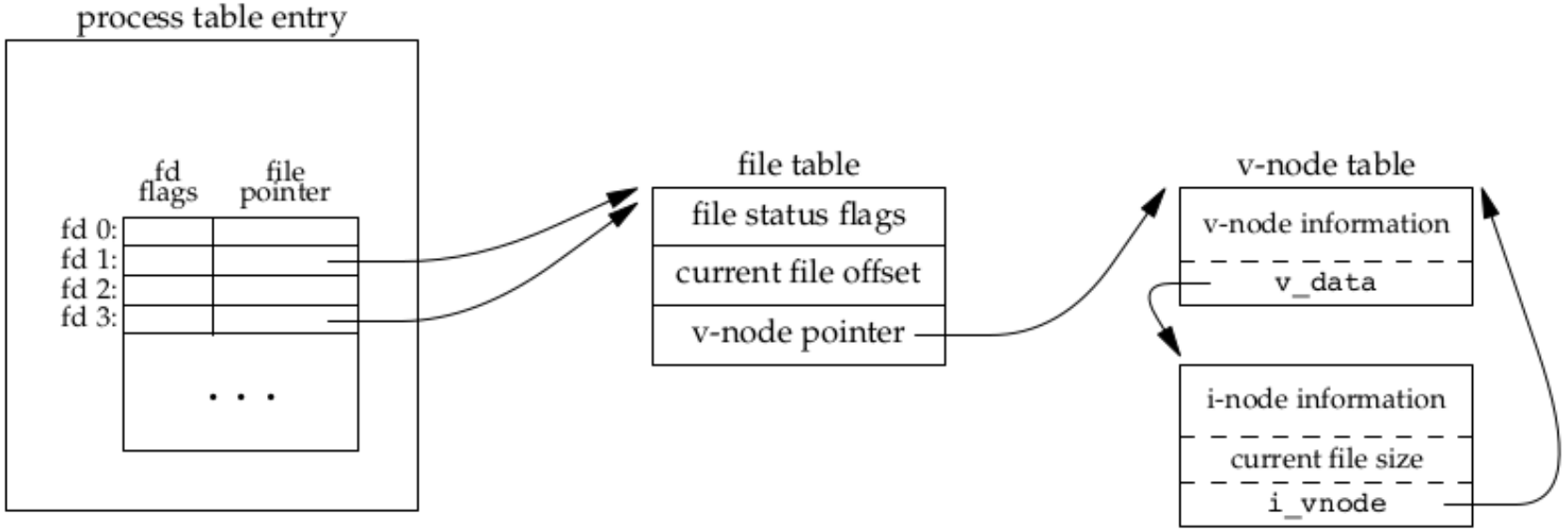
- Kullanılmakta olan bir dosya belirteci aşağıdaki fonksiyonlardan biriyle kopyalanabilir.

```
#include <unistd.h>

int dup(int oldd);

int dup2(int oldd, int newd);
```

Her iki fonksiyonda ok ise yeni dosya belirtecini döner, hata ise -1 döner.

- Kullanılmakta olan bir dosya belirteci dup(2) ile kopyalanabilir. dup2(2) dosya belirtecini belli bir değere kopyalar.
- dup(2) elde bulunan en düşük dosya belirteci numarasını döner.

Dosya belirteci kopyalama



sync, fsync ve fdatasync fonksiyonları

- Geleneksel olarak Unix sistemlerinde kernel önbelleğe sahiptir.
  - Disk ile ilgili yazma ve okuma işlemleri bu önbellekten geçer.
- Diske yazmak istediğimizde bu işlem verimliliği artırmak için anında yapılmaz, bunun yerine kernel önbelleğine yazılır ve bir zaman sonra toplu halde yapılır.
  - Bu yönteme geciktirilmiş yazma denir.
- Kernel bir süre sonra istenilen bütün yazma operasyonlarını gerçekleştirir.
  - Genellikle yazmayı önbellekte yer ihtiyacı olduğunda yapar.
- Disk ile önbellek arasında tutarlılığın sağlanması için sync, fsync ve fdatasync fonksiyonları kullanıma sunulmuştur.

sync, fsync, ve fdatasync fonksiyonları

```
#include <unistd.h>

int fsync(int filedes);

int fdatasync(int filedes);
```

Dönüş: OK ise 0, hata ise -1

```
void sync(void);
```

- sync üzerinde değişiklik yapılan bütün önbellek bloklarını yazılma sırasına sokar.
- fsync sadece bir dosya ile ilgili değişikliğe uğramış bloklarla ilgilenir ve diske yazma işlemi tamamlayıncaya kadar bekler.
- fdatasync dosyanın sadece veri kısmı ile ilgilenir.
- sync fonksiyonu sistem tarafından belli aralıklarla çağırılır (genellikle 10 saniyede bir).
  - Bu sayede kernel önbelleği düzenli olarak boşaltılır.

fcntl fonksiyonu

- Açık bir dosyanın özelliklerini değiştirmek için fcntl fonksyionu kullanılabilir.

```
#include <fcntl.h>

int fcntl(int filedes, int cmd, … /* int arg */ );
```

Dönüş: OK ise cmd değerine bağlı, hata ise -1

- fcntl fonksyionu 5 farklı amaç için kullanılabilir
  - Bir dosya belirtecini kopyalamak için (cmd = F_DUPFD)
  - Dosya belirteci bayraklarını almak/ayarlamak için (cmd = F_GETFD veya F_SETFD).
  - Dosya durum bayraklarını almak/ayarlamak için (cmd = F_GETFL veya F_SETFL)
  - Asenkron I/O sahipliğini almak/ayarlamak için (cmd = F_GETOWN veya F_SETOWN)
  - Kayıt kilidi almak/ayarlamak için (cmd = F_GETLK, F_SETLK veya F_SETLKW.

fcntl fonksiyonu

- fileflags.c.

  $ ./fileflags 0 < /dev/tty

  read only

  $ ./fileflags 1 > temp.foo

  $ cat temp.foo

  write only

ioctl fonksiyonu

- ioctnl fonksiyonu birçok farklı durumda kullanılan çok amaçlı bir fonksiyondur.
- Özellikle farklı araç sürücüleri kendi ioctl komutlarını tanımlayabilirler.

```
#include <unistd.h>            /* System V*/
#include <sys/ioctl.h>            /* BSD ve Linux */
#include <stropts.h>              /* XSI Streams */
int ioctl(int filedes, int request, …);
```

Dönüş: OK ise isteğe bağlı değer, hata ise -1.

Category	Constant names	Header	Number of ioctls
disk labels	DIOxxx	<sys/disklabel.h>	4
file I/O	FIOxxx	<sys/filio.h>	14
mag tape I/O	MTIOxxx	<sys/mtio.h>	11
socket I/O	SIOxxx	<sys/sockio.h>	73
terminal I/O	TIOxxx	<sys/ttycom.h>	43

Figure 3.15 Common FreeBSD `ioctl` operations

Dosyalar ve klasörler

- Önceki bölümde dosya I/O işlemleri için kullanıdığımız temel fonksiyonları gördük.
- Bu bölümde dosya sisteminin ve dosyaların özelliklerini inceleyeceğiz.
- stat fonksiyonu ve bu fonksiyon ile elde ettiğimiz stat yapısının üyelerini ve bu üyelerin anlamlarını inceleyeceğiz.
- Dosya özelliklerini değiştirmek için kullanılan fonksiyonları göreceğiz
  - Örneğin dosyanın sahibini veya dosya üzerindeki izinleri değiştireceğiz.
- Unix dosya sisteminin detaylarını ve sembolik link kullanımını inceleyeceğiz.
- Klasör işlemleri için kullanılan fonksiyonları göreceğiz.

stat, fstat ve lstat fonksiyonları

- stat, fstat ve lstat fonksiyonları belli bir dosya hakkında bilgi verir.

```
#include <sys/stat.h>

int stat(const char *restrict pathname, struct stat *restrict buf);

int fstat(int filedes, struct stat *buf);

int lstat(const char *restrict pathname, struct stat *restrict buf);
```

Dönüş: OK ise 0, hata ise -1.

- stat fonksiyonu yoladı verilen dosyanın bilgilerini döner.
- fstat fonksiyonu dosya belirteci verilen dosyanın bilgilerini döner.
- lstat fonksiyonu sembolik linkleri takip eder.

stat, fstat ve lstat fonksiyonları

- Stat yapısının içeriği sistemden sisteme değişebilmekle birlikte genellikle aşağıdaki bilgileri içerir.

```
struct stat {
  mode_t           st_mode;    /* file type & mode (permissions) */
  ino_t            st_ino;     /* i-node number (serial number) */
  dev_t            st_dev;     /* device number (file system) */
  dev_t            st_rdev;    /* device number for special files */
  nlink_t          st_nlink;   /* number of links */
  uid_t            st_uid;     /* user ID of owner */
  gid_t            st_gid;     /* group ID of owner */
  off_t            st_size;    /* size in bytes, for regular files */
  struct timespec  st_atim;    /* time of last access */
  struct timespec  st_mtim;    /* time of last modification */
  struct timespec  st_ctim;    /* time of last file status change */
  blksize_t        st_blksize; /* best I/O block size */
  blkcnt_t         st_blocks;  /* number of disk blocks allocated */
};
```

- Her bilgi için temel veri tipi tanımlanmıştır.
- Çoklukla kullandığımız ls -l komutu bu yapı içerisinde saklanan bilgileri listeler.

Dosya tipleri

- Normal dosya
  - En yaygın dosya tipi. Dosyanın anlamdırılması kullanan uygulamaya bırakılmıştır (metin veya binary).
- Klasör dosyası
  - Diğer dosyaların isimlerini ve dosyaların özelliklerine işaretçiler içerir. Klasör içeriği her işlem tarafından okunabilir. Değişiklik ise kernel üzerinden mümkündür.
- Blok özel dosyalar
  - I/O amaçlı belli büyüklükte birimler ile işlem yapmaya izin veren dosya tipi (Disk dürücüleri için kullanılır).
- Karakter özel dosyalar
  - I/O amaçlı farklı büyüklükte birimler ile önbelleksiz işlem yapmaya olanak veren dosya tipidir.
  - Sisteme bağlı araçlar ya blok özel yada karakter özel dosyalar üzerinden sisteme bağlanır.

Dosya tipleri

- FIFO
  - Işlemler arası iletişimi sağlamak için kullanılan dosya tipi
- Socket
  - Ağ üzerinden haberleşen işlemler arası haberleşmeyi sağlamak için kullanılan dosya tipi.
- Sembolik link
  - Bir başka dosyaya işaret eden dosya tipi.
- Dosya tipleri stat yapısının st_mode alanında saklanır. Aşağıda belirtilen makroları kullanarak bir dosyanın tipini öğrenebiliriz.

Dosya tipleri

Macro	Type of file
S_ISREG()	regular file
S_ISDIR()	directory file
S_ISCHR()	character special file
S_ISBLK()	block special file
S_ISFIFO()	pipe or FIFO
S_ISLNK()	symbolic link
S_ISSOCK()	socket

Stat yapısının st_mode alanı bu makrolara gönderilir.
filetype.c
$ ./filetype /etc/passwd /etc /dev/log /dev/tty

/etc/passwd: regular
/etc: directory
/dev/log: symbolic link
/dev/tty: character special

Dosya tipleri

- Unix sisteminde baskın olan dosya tipi normal dosyalardır.
  - Aşağıda bir Linux sisteminde dosya tipleri sayıları ve sistemdeki oranları gösterilmiştir.

File type	Count	Percentage
regular file	415,803	79.77 %
directory	62,197	11.93
symbolic link	40,018	8.25
character special	155	0.03
block special	47	0.01
socket	45	0.01
FIFO	0	0.00

Kullanıcı ve grup numaraları

- Her işleme ait altı veya daha fazla numara bulunur.

Gerçek kullanıcı no Gerçek grup no	İşlemin sahibini tanımlar
Efektif kullanıcı no Efektif grup no Tamamlayıcı grup no	Dosya üzerindeki izin kontrolü için kullanılır.
Kayıtlı kullanıcı no kayıtlı grup no	exec fonksiyonları tarafından kaydedilir.

Gerçek kullanıcı numarası ve gerçek grup numarası işlemin sahibini tanımlar
Bu bilgiler parola dosyasından alınır
Efektif kullanıcı numarası, efektif grup numarası ve tamamlayıcı grup numarası dosya izin kontrolleri için kullanılır.
Kayıtlı kullanıcı no ve kayıtlı grup no, başka bir program çalıştırıldığında kopyalanır.

Kullanıcı ve grup numaraları

- Normal kullanımda efektif kullanıcı numarası gerçek kullanıcı numarasına, efektif grup numarası ise gerçek grup numarasına eşittir.
  - Yani dosyayı kim çalıştırmışsa oluşan işlemin sahibi o kişi olur.
- Ancak istenildiğinde farklı kullanımlar mevcuttur.
- Her dosyanın sahibi ve grup sahibi vardır. stat yapısındaki st_uid alanı dosyanın sahibini, st_gid ise dosyanın grup sahibini belirtir.
- set-user-id bit: Bu bit değeri ayarlanmışsa (değeri 1 ise) dosya çalıştırıldığında dosyanın sahibi oluşan işlemin efektif kullanıcı numarası olur.
- set-group-id bit: Bu bit değeri ayarlanmışsa, dosya çalıştırıldığında dosyanın grup numarası oluşan işlemin efektif grup numarası olur.
- passwd, set-user-id değeri ayarlanmış bir programdır.

Dosya erişim hakları

- stat yapısındaki st_mode değeri dosya erişim haklarına ait bit değerlerini içerir.
- Her dosya için 9 farklı bir değeri vardır ve bu değerler üç farklı kategoriye ayrılmıştır.
- Bu sabit isimler <sys/stat.h> dosyasında tanımlanmıştır

st_mode mask	Meaning
S_IRUSR	user-read
S_IWUSR	user-write
S_IXUSR	user-execute
S_IRGRP	group-read
S_IWGRP	group-write
S_IXGRP	group-execute
S_IROTH	other-read
S_IWOTH	other-write
S_IXOTH	other-execute

Dosya erişim hakları

- İlk üç satırdaki user terimi dosyanın sahibini kastetmektedir.
  - chmod komutu kullanarak izin haklarını değiştirdiğimizde dosya sahibi için u, grup hakları için g ve diğer kişilerin hakları için o kullanacağız.
- Dosya erişimi ile ilgili bazı kurallar:
  - Bir dosyayı açabilmek için, dosya yoladında bulunan bütün klasörlerde execute (çalıştır) hakkı bulunmalıdır.
  - Dosyayı O_RDONLY veya O_RDWR olarak açmak için read (okuma) izninin olması gerekir.
  - Dosyayı O_WRONLY veya O_RDWR olarak açmak için write (yazma) izninin olması gerekir.
  - O_TRUNC ile açmak için write izni olmalıdır.
  - Yeni bir dosya yaratabilmek için bulunan klasörde write ve execute izni olmalıdır.

Dosya erişim hakları

- Dosya erişim hakları konusunda bazı kurallar
  - Bir dosyayı silebilmek için dosyanın bulunduğu klasörde write ve execute hakları olmalıdır.
    - Dosyanın üzerindeki haklar önemli değildir.
  - Bir dosyayı çalıştırmak için, dosya üzerinde execute hakkı olmalıdır.
- Kernel bir dosya açılmaya yaratılmaya veya silinmeye çalışıldığında dosya erişim testleri yapar. Bu testler aşağıdaki bilgilere dayanır:
  - Dosyanın sahibi (st_uid ve st_gid).
  - Işlemin efektif kullanıcı numarası ve efektif grup numarası.
  - Eğer desteklenyorsa, işlemin tamamlayıcı grup numarası.

Dosya erişim hakları

- Kernel tarafından yapılan testler şunlardır.
  - Eğer effective-uid == 0 (yönetici), hak tanınır.
  - Eğer effective-uid == st_uid ise
    - Alakalı bit ayarlanmışsa hak tanınır.
    - Aksi halde reddedilir.
  - Eğer effective-gid == st_gid ise
    - Alakalı bit ayarlanmışsa hak tanınır.
    - Aksi halde reddedilir.
  - Eğer alakalı diğer kullanıcı bit ayarlanmışsa hak tanınır, aksi halde reddedilir.
  - Bu testler sıra ile yapılır.

Yeni oluşturulan dosyaların sahibi

- Yeni oluşturulan dosyanın sahibi oluşturan işlemin efektif kullanıcı numarası olarak ayarlanır.
- POSIX.1 standartlarına göre yeni oluşturulan dosyanın grup numarası aşağıda belirtilen şekillerde ayarlanabilir.
  - Yeni oluşturulan dosyanın grup numarası, oluşturan işlemin efektif grup numarası olarak ayarlanabilir.
  - Yeni oluşturulan dosyanın grup numarası, dosyanın oluşturulduğu klasörün grup numarası olarak ayalanabilir.
    - Bu opsiyon kullanılırsa aynı klasör altında oluşturulan dosya ve klasörlerin aynı gruba ait olması mümkün olur.
- Klasör sahipliği konusundaki kurallar aynıdır.

access fonksiyonu

- Access fonksiyonu kullanılarak bir dosyaya erişim hakları kontrol edilebilir.
- Access fonksiyonu belirtilen testleri çalıştıran işlemin gerçek kullanıcı numarası ve gerçek grup numarasına göre yapar.

```
#include <unistd.h>

int access(const char *pathname, int mode);
```

Dönüş: OK ise 0, hata ise -1.

mode	Tanım
R_OK	Okuma testi
W_OK	Yazma testi
X_OK	Çalıştırma testi
F_OK	Dosya var mı?

access.c

umask fonksiyonu

- umask(2) fonksiyonu kullanılarak dosya oluşturma maskesi ayarlanır.
- Dosya oluşturma maskesinde ayarlı olan bitler dosyada kapalı olarak ayarlanır.

```
#include <sys/stat.h>

mode_t umask(mode_t mask);
```

Dönüş: önceki dosya oluşturma maskesini döner.

- umask.c

umask fonksiyonu

- Kullanıcılar umask değerlerini değiştirerek oluşturulan dosyaların varsayılan izinlerini kontrol edebilirler.
- Bu değer octal olarak gösterilir ve her bir maske ile kapatılacak olan bir izni temsil eder.

Mask bit	Meaning
0400	user-read
0200	user-write
0100	user-execute
0040	group-read
0020	group-write
0010	group-execute
0004	other-read
0002	other-write
0001	other-execute

```
$ umask
0022
$ umask -S
u=rwx,g=rx,o=rx
$ umask 027
$ umask -S
u=rwx,g=rx,o=
```

chmod ve fchmod fonksiyonları

- Bu fonksiyonlar var olan bir dosyanın izinlerini değiştirmemize olanak verir.

```
#include <sys/stat.h>

int chmod(const char *pathname, mode_t mode);

int fchmod(int filedes, mode_t mode);
```

Dönüş: OK ise 0, hata ise -1.

- Bir dosyanın izinlerini değiştirebilmemiz için değiştirmek isteyen işlemin efektif kullanıcı numarasının dosyanın sahibi ile aynı olması veya işlemin super kullanıcı haklarına sahip olması gerekmektedir.

chmod ve fchmod fonksiyonları

- Mode değişkenini ayarlamak için aşağıdaki sabitleri bit olarak OR işlemine sokmalıyız.

mode	Description
S_ISUID	set-user-ID on execution
S_ISGID	set-group-ID on execution
S_ISVTX	saved-text (sticky bit)
S_IRWXU	read, write, and execute by user (owner)
S_IRUSR	read by user (owner)
S_IWUSR	write by user (owner)
S_IXUSR	execute by user (owner)
S_IRWXG	read, write, and execute by group
S_IRGRP	read by group
S_IWGRP	write by group
S_IXGRP	execute by group
S_IRWXO	read, write, and execute by other (world)
S_IROTH	read by other (world)
S_IWOTH	write by other (world)
S_IXOTH	execute by other (world)

changemod.c

Örnekte görüleceği üzere dosya izinlerini önceki izinlere bağlı olarak veya direk olarak ayarlayabiliriz.