7. Файлова система. Логическа организация и физическо представяне. Основни примитиви Логическа организация на файлова система

Начална страница
Автореферат
Анализ
Бизнес-план
Биография
Глава
Диплом
Доклад
Задача
Закон
Занятие
Заседание

Навигация на страницата:

7. Файлова система. Логическа организация и физическо представяне. Основни примитиви.

.1Логическа организация на файлова система

Файлът е основна единица, чрез която операционните системи осигуряват съхранението на постоянните обекти данни.

- **Имена на файлове.**

Имената на файловете представляват символни низове с определена дължина. В някои операционни системи има конвенции името да се разделя на две части: име и разширение, като разширението обикновено служи за обозначаване на типа на файла.

В UNIX/Linux тази конвенция не е вграден в ядрото и затова се допуска да има повече от едно разширение. В повечето операционни системи за разделител между име и разширение се използва точка.

- **Типове файлове – обикновен файл, специален файл, каталог, символна връзка, програмен канал и др.**

- Обикновен файл – файл служещ за съхраняване на информация върху външен носител; информацията може да бъде програма, данни, текст и пр.
- Специален файл – служи за абстракция на външно устройство; всеки такъв файл съответства на едно периферно устройство; може да има два типа специални файлове: символни и блокови. Това позволява да се пишат програми, които не зависят от типа на устройството.
- Каталог – файл с определен формат, съдържащ информация за част или всички файлове в системата, позволяващ да се осъществи връзка между името на файла и данните в него, както и да се организират и групират файловете в системата.
- Символна връзка – файл, чиито данни представляват името на друг файл и служат като указател към него.

- Програмни канали – услуга предоставяна от операционната система, позволяваща комуникация между конкурентни процеси.

Вътрешна структура на файл.

Всеки файл се разглежда от операционната система като последователност от обекти. Тези обекти могат да бъдат:

- Записи с определена дължина и структура – Най-малката единица за четене и писане е един запис. Това поражда проблеми, тъй като при различните видове файлове записите трябва да са с различна дължина и формат, което изисква различен подход при операциите четене и писане.
- Байтове – По този начин се позволява унифициран подход при операциите за четене и писане, както и прости системни примитиви.

Атрибути на файл.

Файловите атрибути за разделени на четири групи:

- Атрибути, свързани със защита на файла – собственик, създател, права за достъп, парола за достъп.
- Атрибути показващи важни моменти от съществуването на файла – момент на създаване, на последна промяна, на последен достъп, на унищожаване.
- Атрибут показващ размер на файла – показва брой байтове или брой записи плюс размер на един запис.
- Флагови атрибути – readonly (само за четене), system (системен файл), secure deletion (сигурно изтриване), undelete (при изтриване се съхранява информация, помагаща евентуално възстановяване), immutable (не може да бъде променян), hidden (файлът е скрит).

- **Йерархична организация на файлова система – абсолютно и относително пълно име на файл, текущ каталог.**

В старите операционни системи всички файлове са се намирали логически на едно и също място. В съвременните операционни системи е въведено понятието каталог (директория, папка). Всяка директория може да съдържа както файлове, така и други директории. Така се реализира дървовидна структура, която позволява добра организация на разположението на файловете. Корен на дървото е главната директория. В някои операционни системи има само едно дърво, независимо от броя на дисковете и външните устройства. В други, за всеки диск има отделно дърво. В следствие на въвеждането на дървовидна структура, всеки файл, освен собственото си име, притежава още

- пълно име – то се определя от това в кой каталог е файла и служи за идентификация на файла в рамките на цялата йерархична структура.
- абсолютно пълно име – пълният път от корена на файловата система до файла
- относително пълно име – пътят спрямо някоя директория; наличието на относително пълно име се налага от факта, че повечето операционни системи поддържат понятието текущ каталог на даден процес и често се налага да се определи местоположението на файл спрямо текущия каталог.

.2Физическо представяне на файловата система

Основни цели на физическата организация са ефективност, надеждност, сигурност и разширяемост.

- **Стратегии за управление на дисковото пространство.**

Стратегиите за управление на дисковото

пространство се обуславят в зависимост от следните въпроси:

1. Кога се разпределя дискова памет за файл
   - Еднократно при създаване на файла (статично)
   - Заделя се нова порция при нарастване на файла (динамично)
2. Колко непрекъснати области да заема един файл
   - Една област, която да побере всички данни
   - Много, физически не съседни области
3. Каква да е единицата за разпределение на дискова памет
   - Дали да е с променлив или фиксиран размер
   - Какъв трябва да е размерът й.

Най-често използваните стратегии са:

1. Статично и непрекъснато разпределение – следните сериозни проблеми съществуват:
   - колко да е голям блокът за един файл;
   - трябва да се знае предварително евентуалният размер на файла;
   - какво да се прави, когато този размер се надвиши.
2. Динамично и поблоково разпределение с фиксиран размер на блока – важен въпросът за размера на блока, които се разпределя: ако блокът е малък, се пести дискова памет, но файловете се фрагментират и се намалява бързодействието на файловата система; ако блокът е голям, се губи дискова памет.

- **Системни структури, съдържащи информация за разпределението на дисковата памет и съхранявани постоянно на диска**

При работата на операционната система са необходими структури, които съхраняват информация за текущото разпределение на дисковата памет. Те играят основна роля в алгоритмите, които операционната система използва за заделяне и освобождаване на дискова памет.

- Структури за свободните блокове

Когато операционната система се опитва да задели памет, тя се нуждае от информация за свободните блокове. Има три популярни реализации на структури за свободни блокове:

1. Свързан списък на свободните блокове

Всеки свободен блок съдържа адреса на следващия свободен блок, а адресът на първия се пази в специална структура. Когато трябва да се задели памет се разпределя първия свободен блок, а адресът на втория се запазва в специалната структура. Тази реализация е ненадеждна при сривове, защото е пръсната по всички свободни блокове и при повреда на някой блок, всички негови наследници се губят. Предимството е, че е проста за реализация.

2. Свързан списък от блокове с номера на свободни блокове

Свободните блокове се разделят на групи. Първият блок от всяка група съдържа номерата на блоковете от групата и указател към първият блок от следващата група. Тази структура е по-компактна, позволява по-малко грешки и е по-ефективна.

3. Карта (таблица) на диска (bitmap)

Поддържа се масив от елементи, всеки от които отговаря на един блок от дисковото пространство. Съседни елементи отговарят на физически съседни блокове. Ако стойността на даден елемент е 1, то съответният му блок е свободен; ако е 0, то блокът е зает. Предимствата на тази структура са, че е компактна и може да отчита физическото съседство на блоковете. Недостатък е че има фиксиран размер на файловата система.

- Структури за блоковете, разпределени за всеки един файл

Тези структури съдържат информация за това кои блокове принадлежат на даден файл и в каква последователност.

1. Свързан списък на блоковете на файла – Всеки блок от даден файл съдържа адреса на следващия, а последният съдържа маркер за край на файл (EOF). Друга структура съдържа адреса на първия блок на всеки файл. Тази структура е ненадеждна при сривове, защото е пръсната по всички блокове на файла. Освен това при произволен достъп до даден блок от файла, трябва да се обходят всички предходни блокове.
2. Карта (таблица) на блоковете на файла – масив, който съдържа цялата основна информация за разпределението на дисковото пространство. Всеки елемент съдържа номера на следващия блок за даден файл. Тази структура се нарича File Allocation Table (FAT) и се използва в MS DOS. Има отделна структура, където се съхраняват имената на файловете и адреса на първия им блок.
3. Индекс – Всеки файл си има собствена структура индекс, в която се съдържа информация за разпределените за файла блокове в съответната логическа последователност. В Unix и Linux се използва дърво – всеки файл си има индексен описател, който при големи файлове става корен на дърво. В HPFS в OS/2 се използва B+ дърво.

- Структури за общи параметри на файловата система

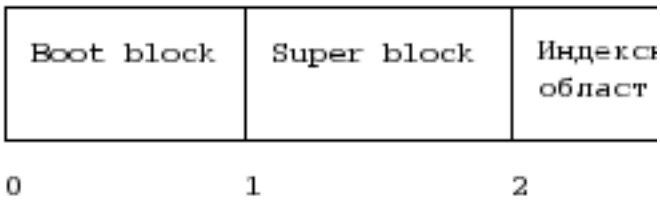
Тези структури служат за съхраняване на важна информация за файловата система на глобално ниво: големина на блока, адреси на началата на различните области, общ брой свободни индексни описатели, общ брой свободни блокове и пр.

Примери за физическа организация на файлова система

- UNIX System V

Дисковото пространство се разглежда като последователност от блокове с фиксиран размер, днес най-честно с размер 1 KB. Дискът е разделен на четири области:

Boot block заема един блок и съдържа програмата,



която стартира зареждането на операционната система.

Super block също заема един блок и съдържа най-общи параметри на файловата система: дължина на индексната област (s_isize), максимален номер на блок във файловата система (s_fsize), брой свободни блокове (s_nfree), номера на свободни блокове (s_free), брой свободни индексни описатели (s_ninode), номера на свободни индексни описатели (s_inode), флагове за заключване на списъка със свободните блокове (s_flock) и на списъка със свободните индексни описатели (s_ilock) и други параметри.

Индексната област се състои от индексни описатели, които имат еднаква структура. Размерът на областта зависи от броя на индексните описатели, които тя съдържа. Този брой се определя при създаване на файловата система и не може да се променя. Индексните описатели

съдържат информация за файловете. Всеки индексен описател може да описва най-много един файл.

В индексните описатели се пази следната информация за файл:

1. Атрибути – 24 байта

а) di_mode – тип на файла (старши 4 бита) и код на защита (12 бита)

б) di_nlink – брой твърди връзки или имена на файл, т.е. колко пъти в записи на каталози е цитиран този номер на индексен описател

в) di_uid и di_gid – определят собственика и групата на файла

г) di_size – размера на файла в байтове

д) di_atime, di_mtime, di_ctime – време на последен достъп, последна промяна и създаване

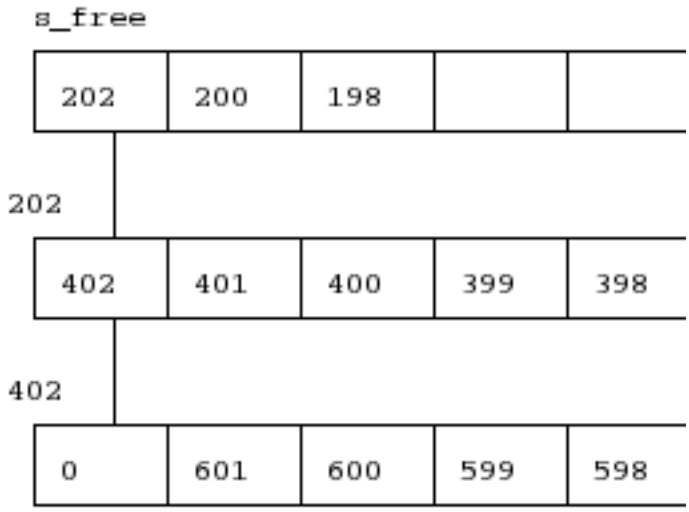
2. Адресна информация – 40 байта (di_addr)

Ако файлът е специален в di_addr се съхранява номера на устройството, на което съответства специалния файл.

Ако файлът е обикновен, то в полето di_addr се съхраняват 13 дискови адреса. Първите 10 адреса са директни адреси на първите 10 блока с данни на файла. Ако файлът е по-голям от 10 блока, тогава се използват косвени блокове. Косвените блокове се намират в областта за данни, но съдържат номера на блокове, а не данни на файла. Единадесетия адрес съдържа номер на косвен блок, който съдържа номерата на следващите блокове с данни на файла. Това се нарича *единична косвена адресация*. Дванадесетия блок съдържа номер на косвен блок, който съдържа номера на косвени блокове, които вече съдържат номера на блокове с данни. Това се нарича *двойна косвена адресация*. Чрез тринадесетия блок се реализира *тройна косвена адресация*.

Списък на свободните блокове е реализиран чрез свързан списък. Главата на този списък се намира в първият елемент на масива s_free от супер-блока. Следващите елементи от s_free са номера на свободни блокове. Първият елемент на s_free съдържа номер на блок от областта за данни, който блок има за първи елемент номер на друг блок с адреси на свободни блокове, а всички останали елементи са адреси на свободни блокове.

Списък на свободни индексни описатели е



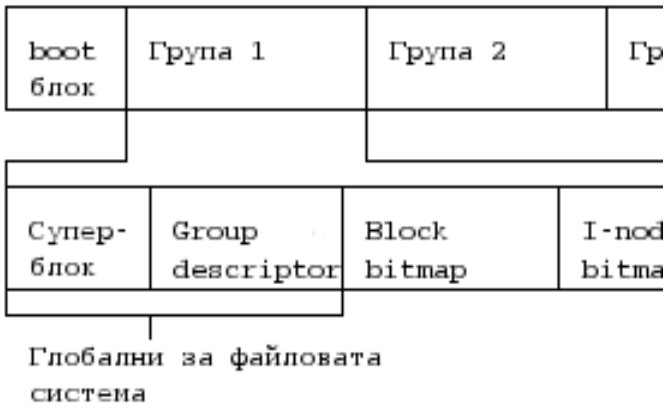
реализиран само чрез масива s_inode от супер-блока. Той не продължава в свързан списък, защото реално файловата система може да функционира и без него, тъй като ядрото на операционната система може да разпознае свободен индексен описател, но се използва s_inode за повишаване на бързодействието.

Каталози са представени като файлове, в които за всеки файл, съдържащ се в каталога, се записва структура с две полета: индексен описател и име на файл. Всеки каталог (дори празният) има два стандартни записа: 1) име: „.“; индексен описател: индексен описател на текущия каталог; 2) име „..“; индексен описател: индексен описател на каталога, в който се намира текущият каталог.

- LINUX

Дисковото пространство се разделя на еднакви блокове с размер, който се задава при създаване на файловата система. Блокът е най-малката единица за работа.

Boot block заема един блок и съдържа програмата,



която стартира зареждането на операционната система.

Останалите блокове се обединяват в еднакви по размер групи. Всяка група съдържа част от файловата система и копие на глобалните системни структури, критични за цялостността на системата – суперблока и описателите на групите.

Group descriptor – съдържа:

а) адрес на блок с битова карта на блоковете за групата

б) адрес на блок с битова карта на i-node за групата

в) адрес на първи блок на индексната област в групата

г) брой свободни блокове в групата

д) брой свободни i-node в групата

е) брой каталози в групата

ж) резервирано поле от 14 байта

Group descriptor-ите се записват във всяка група по ред на номерата един след друг.

Block bitmap – описва свободните блокове

I-node bitmap – описва свободните индексни описатели. Всеки индексен описател съдържа информация за това кои блокове принадлежат на даден файл. Това се постига чрез указатели, които сочат към блокове от областта за данни на групата.

Размерът на i-node-а е 128 байта и съдържа следните полета:

а) 12 полета с директни адреси на блокове за данни

б) 1 поле за единична косвена адресация

в) 1 поле за двойна косвена адресация

г) 1 поле за тройна косвена адресация

д) атрибут за тип на файл и код на защита

е) време на създаване

ж) време на последна промяна

з) време на последен достъп

и) брой блокове, заемани от файла

й) време на унищожаване

к) флагове – undelete, secure deletion, immutable, append, compress, archive

л) размер на файла в байтове

м) брой твърди връзки

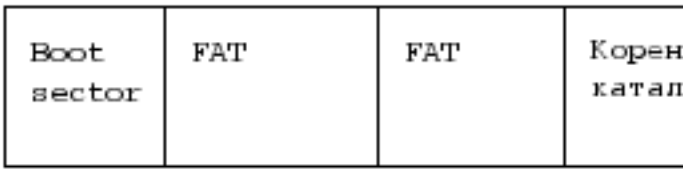
н) собственик и група на файла

Каталози са представени като файлове. За всеки файл, който съдържа един каталог, съществува запис в този каталог със следната структура: номер на i-node за съответния файл; дължина на записа; дължина на името на файла; име на файла. Всеки каталог (дори празният) има два стандартни записа: 1) име: „.“; индексен описател: индексен описател на текущия каталог; 2) име „..“; индексен описател: индексен описател на каталога, в който се намира текущият каталог.

- MS DOS

Всеки дял или диск има своя собствена файлова система, наричана том (volume). Дисковото пространство е последователност от сектори и е разделено на пет области:

Boot sector – съдържа програмата, която стартира зареждането на операционната система, както и параметри на тома, като размер на клъстер, размер на тома, размер и брой копия на FAT, размер на коренния каталог.



FAT – карта на файловете и съдържа цялата информация за паметта, разпределена за файловете, за свободното дисково пространство, за дефектните сектори и формата на диска.

Памет за файлове се разпределя динамично и единицата за разпределение се нарича *клъстер*. Клъстерът е последователност от един или повече сектора и всички клъстери от един том са с еднакъв размер. Първите два записа във FAT са резервирани за код, идентифициращ тома. Останалите елементи във FAT съответстват на клъстери със съответни номера. Ако даден клъстер е свободен, то съответният му запис във FAT съдържа 0. Ако не е свободен, то съответният му запис във FAT съдържа или номер на следващ клъстер, или максимален номер на клъстер, което означава EOF.

Коренен каталог – това е област, в която се съдържат данни за коренния каталог на тома. Коренният каталог не е файл.

Каталози са файлове, които съдържат записи за други файлове или каталози. Записите са с фиксиран размер и съдържат:

а) име на файл

б) разширение

в) флагове

г) час на създаване или последно изменение

д) дата на създаване или последно изменение

е) номер на първи клъстер

ж) размер на файла.

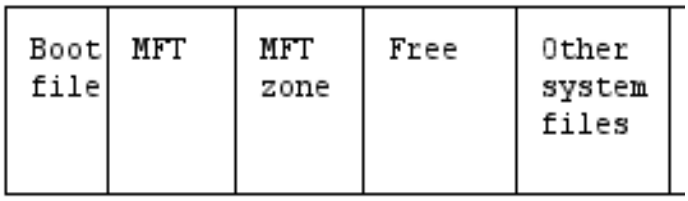
Всички каталози без коренния съдържат двата стандартни записа с име „.“ и с име „..“. Коренният каталог пък съдържа един специален запис за етикета на тома в първите две полета на записа.

- NTFS

Единицата за разпределяне на дисково пространство в NTFS е клъстер. Адресът на клъстер от началото на том се нарича LCN (Logical Cluster Number), а адресът в рамките на определен файл е VCN (Virtual Cluster Number).

В NTFS всичко се третира като файл: и данните, и метаданните, т.е. на всеки том има обикновени файлове, каталози и системни файлове. Главният системен файл е MFT (Master File Table).

Файлът MFT представлява индекс на всички



файлове на тома. Съдържа записи по 1 KB и всеки файл от тома е описан чрез един запис, включително и MFT. Освен MFT има и други файлове с метаданни, чиито имена започват с $ и са описани в първите 24 записа на MFT. За да се намали фрагментирането на MFT, се поддържа буфер от свободно дисково пространство – MFT зона, докато не се запълни останалото дисково пространство. Адресът на началото на MFT файла се намира в Boot файла. MFT файлът съдържа:

а) $Mft – MFT файл

б) $MftMirr – файл с копие на първите записи от MFT

в) $LogFile – Журнален файл при поддържане на транзакции

г) $Volume – Файл с описание на тома

д) $AttrDef – Файл с дефиниции на атрибутите

е) \ – Коренен каталог на тома

ж) $Bitmap – Файл, съдържащ битова карта на клъстерите на тома

з) $BadClus – Файл с лошите клъстери на тома.

Всеки файл се съхранява като съвкупност от двойки атрибут-значение. Един от атрибутите на

файл са данните; нарича се unamed data attribute. NTFS поддържа и named data attributes, които се задават от потребителя; по този начин, един файл може да се състои от няколко именовани потока от данни. Други атрибути са име на файл, стандартна информация и др. Всеки атрибут се съхранява като отделен поток от байтове.

Всеки MFT запис съдържа заглавие на записа и атрибути на файла. Всеки атрибут се съхранява като заглавие на атрибута и стойност. Заглавието на атрибута съдържа тип, име, флагове на атрибута и информация за разположението на данните му. Един атрибут е резидентен, ако данните му се побират в един запис заедно със заглавията на всички атрибути. Ако един атрибут не е резидентен, заглавието му (което е винаги резидентно) съдържа информация за клъстерите, разпределени за стойността му (данните на атрибута). Адресната информация е описание на екстенти. Всеки екстент е непрекъсната последователност от клъстери, разпределени за данните на съответния атрибут и се описва от адрес на началния клъстер и дължина (VCN, LCN и брой клъстери). Ако един файл има много атрибути и имената не могат се съберат в един MFT запис, се разпределят допълнителни записи.

Каталогът съдържа записи с променлива дължина, всеки от които съответства на файл или подкаталог. Всеки запис съдържа името и индексът на файла в MFT, както и копие на стандартната информация за файла. Записите в каталога се съхраняват под формата на B+ дърво. Ако каталогът е малък, той се съхранява изцяло в съответния му MFT запис. Когато стане голям, се заделят екстенти, които заедно с атрибута $INDEX_ROOT от MFT записа са организирани в B+ дърво и се наричат индексни буфери.

Забележка: на изпита ще бъде избрана част от изброените примери за файлова организация.

.3Основни системни примитиви на файлова система

- **Работа с обикновен файл – създаване, отваряне, четене, писане, позициониране и др.**

int creat (const char* filename, mode_t mode)

Създава нов файл (обикновен);

filename – името на файла;

mode – кода на защита или правата за достъп до файла;

връща – при успех число по-голям или равно на нула, представляващо файлов дескриптор от системните таблици.

int open (const char* filename, int flag[, mode_t mode])

Отваря файл; ако файлът не съществува, първо го създава, при което се използва третият аргумент;

flag – конструира се от различни флагове, които определят режима на отваряне на файла.

int close (int fd)

Затваря файл, при което се освобождава файловият дескриптор, съдържащ се във fd.

fd – файлов дескриптор на затварящия файл.

ssize_t read (int fd, void* buffer, size_t nbytes)

Примитив за четене от файл; опитва се да прочете nbytes наброй байта от файл с файлов дескриптор fd, започвайки от текущата позиция, като се записват прочетените байтове в buffer;

връща – брой прочетени байтове; при грешка -1.

ssize_t write (int fd, void* buffer, size_t nbytes)

Примитив за писане във файл; опитва се да запише nbytes наброй байта от buffer във файл с файлов дескриптор fd;

връща – брой записани байтове; при грешка -1.

off_t lseek (int fd, off_t offset, int flag)

Примитив за позициониране във файл;

offset – задава отместване на текущата позиция в брой байтове;

flag – указва от къде бъде отчитано отместването – относно началото (SEEK_SET), относно текущата

позиция (SEEK_CUR) или относно края (SEEK_END).

int dup (int fd)

Примитив за копиране на файлов дескриптор – създава нов файлов дескриптор към файла, сочен от fd. При успех връща новия файлов дескриптор.

- **Изграждане на йерархична организация на файлова система – създаване и унищожаване на каталог, създаване и унищожаване на връзки, смяна на текущ каталог**

int mkdir (const char* dirname, mode_t mode)

Създава празен каталог; за него се разпределя един блок за данни, в който има записи „.“ и „..“.

int rmdir (const char* dirname)

Унищожава каталог; каталогът не трябва да съдържа файлове.

int chdir (const char* dirname)

Сменя се текущият каталог на процеса; старият текущ каталог на процеса се освобождава и се зарежда индексният описател на новия каталог. Всички каталози по пътя трябва да съществуват и процесът да има execute права за тях.

int link (const char* oldname, const char* newname)

oldname – име на съществуващ обикновен файл

newname – име на твърдата връзка

int symlink (const char* toname, const char* fromname)

toname – име на съществуващ файл, към който ще сочи символичната връзка

fromname – име на символичната връзка

При твърдата връзка се гарантира съществуването на файла и след като оригиналното име е унищожено, докато при символната връзка това не е така. Всъщност дори не се проверява съществуването на оригиналният файл при

създаване на символна връзка. Символната връзка се интерпретира при опит за достъп до файла чрез нея. Освен това символната връзка може да се създава през границите на файловата система към обикновен файл, специален файл и към каталог.

int unlink (const char* name)

Указаното име на файл се изключва от файловата система; полето за броя на твърдите връзки в индексния описател на файла се намалява с 1; ако този брой стане 0, то файлът се унищожава.

- **Защита на файловата система.**

При създаване на файл има аргумент mode, в който се задават кода на защита на файла и той се запомня в i-node-а на файла. При всеки достъп до файла се проверяват правата на собственика на процеса за достъп до файла.

int chmod (const char* name, mode_t mode)

Сменя правата за достъп до файла; процесът, изпълняващ този примитив, трябва да е на администратора или на собственика на файла.

int chown (const char* name, uid_t owner, gid_t group)

Сменя собственика на файла name, като owner е идентификатор на новия собственик, а group новата на новата група. chown може да бъде изпълнен от администратора или от собственика на файл.

mode_t umask (mode_t cmask)

Определя маската за създаване на файл; тази маска влияе на примитивите, изпълнявани от процеса за създаване на файлове – creat, open, mkdir, като модифицира кода на защита на създадените файлове; действителният код на защита при създаване е mode&~cmask; при успех се връща старата маска.

Каталог: DIP
DIP -> Съдържание
DIP -> Решение за въвеждане на културно-национална автономия в Пиринска Македония, като се подготви присъединяването на този край към нрм

DIP -> Златен кестен
DIP -> У в о д 2 историография и изворов материал 4
DIP -> 6. Вестникът в интернет: може ли да привлича читатели
DIP -> Компютърна система с процесор Intel Pentium II компютърните системи с процесор Intel Pentium II използват архитектура „северен-южен мост”
DIP -> 3 II. Teoretická ČÁST
DIP -> Дипломатически корпус (справочник) София Юни, 2016 г. Съдържание
DIP -> 3. 15 Дънна платка с Chipset Intel 965 за процесор Intel Pentium Чипсети от серията 96x