**Файлова система. Логическа организация и физическо представяне.**   
Логическа организация на файлова система (ФС). Имена на файлове. Типове файлове - обикновен файл, специален файл, каталог, символна връзка, програмен канал.   
Вътрешна структура на файл. Атрибути на файл. Йерархична организация на ФС - абсолютно и относително пълно име на файл, текущ каталог. Физическа организация на   
ФС. Стратегии за управление на дисковото пространство. Системни структури, съдържащи информация за разпределението на дисковата памет и съхранявани   
постоянно на диска: за свободните блокове; за блоковете, разпределени за всеки един   
файл; за общи параметри на ФС. Примери за физическа организация на ФС: UNIX   
System V; LINUX; MS DOS; NTFS

На някои места съм писал ‘‘\_‘‘ вместо ‘‘ ‘‘, щото word се ебава

Някои неща се повтарят

Файлова система. Пространство на имената (namespace)

Повечето операционни системи (ОС) реализират йерархична (дървовидна) организация на файловете чрез специален тип файл, наречен каталог (directory). Всичко започва от коренния каталог, чието име е символът „/”. Логически всеки каталог съдържа записи за файлове и други каталози. Освен обикновените\_файлове\_и каталозите, много ОС реализират и други типове файлове: специални файлове(character special device files, block special device files),програмни\_канали\_(FIFO\_files), символни връзки(symbolic links) и др.  
Всеки файл има собствено име, което се съдържа в записа от родителския му  
каталог. В някои Unix системи собственото име е ограничено до 14 символа, в\_по-съвременните весии на Unix, в Linux и в POSIX се въвеждат дълги собствени\_имена\_до\_255\_символа.  
Всеки файл има едно абсолютно пълно име (absolute path name), което  
съответства на единствения път в дървото от коренния каталог до файла. То  
представлява списък от имената на каталозите в пътя, разделени със символа „/”,  
започвайки\_с\_името\_на\_коренния\_каталог. - /.  
Всеки процес има свой текущ каталог (current directory, работен каталог,  
working directory). Относно този каталог започва относителното пълно име (relative path name) на файл. То представлява списък от имената на каталозите в пътя, тръгвайки от текущия в момента каталог (отново разделени със символа„/”).  
Всеки потребител, регистриран в системата има свой начален каталог (home  
directory), чието име се помни във файла /etc/passwd. Веднага след вход в системата, текущ каталог е началния каталог на съответния потребител.  
В съвременната операционна система има 2 слоя на абстракция:  
1. Пространство на имената – всички дълготраини обекти, които съществуват в системата. Това е абстрактната файлова система;  
2. Реализацията – как са представени тези обекти и как са съпоставени на хардуера (всеки информационен обект, как и къде ще бъде в хардуера).  
В UNIX абстрактната файлова система е едно кореново дърво.  
"-" – обикновени файлове; "d" – папка или директория; "l" – символен линк (друго име на файл – съществуващ или не); "c" – символно устройство (character\_device); "b" – масив от баитове, който не е RAM паметта на компютъра,нито\_процесора\_и\_неговите регистри, а е някакво външно устройство, което съхранява някакъв масив от байтове и може да се ползват или записват байтове където решим (може да се индексира) "p" – именувана тръба (FIFO); "s" – крайна точка за комуникация (socket).Процес\_сървър,  
който предоставя връзки за съответни услуги (служи за да може клиента да намира сървъра).

В елемента st\_mode са кодирани два атрибута на файла – тип на файла и кода  
му на защита. Типът на файла се съхранява в старшите 4 бита на st\_mode. Да си припомним основните типове файлове, реализирани в съвременните Unix и\_Linux\_системи.

Тук\_има\_повторение   
1. Обикновен файл (regular file) е файл, съдържащ данни в някакъв формат.  
2. Каталог (directory). Това е файл, съдържащ записи за други файлове, всеки от които съдържа собственото име на файл и номера на i-node му. Чрез този тип\_се релизира йерархичната организация на файловата система.  
3. Символен специален файл (character special device file). Всеки файл от този тип съответства на символно устройство, напр., терминал, печатащо устройство\_и\_др.  
4. Блоков специален файл (block special device file). Всеки файл от този тип  
съответства\_на\_блоково\_устройство,\_например\_диск.  
5. Символна връзка (symbolic link или soft link). Символната връзка е файл, който\_сочи\_към\_друг\_файл.  
6. Програмен канал (pipe, FIFO file). Това е тип файл, реализиращ механизъм за\_междупроцесни\_комуникации.  
7. Сокет (socket) е тип файл, реализиращ механизъм за междупроцесни  
комуникации в мрежова среда.

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.
* Tova may ne go iska osven flagovi atributi  
  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).

Код на защита на файл и права на достъп  
С всеки файл е свързан потребител – собственик на файла и потребителска  
група, към която принадлежи собственика. Тези два атрибута са съответно в  
елементите st\_uid и st\_gid. (Там са числовите идентификатори на потребител и потребителска група, а не имената им.) Младшите 12 бита от st\_mode представляват кода на защита на файла и определят правата на потребителите за достъп до този файл.  
Различават се три типа достъп до файл – четене, писане и изпълнение (означавани със символите r, w, x в изхода на командата ls - l), които се прилагат спрямо всеки един файл независимо от типа му.

За обикновен файл:  
- Право r означава правото да отворим файла за четене, т.е. с флаг O\_RDONLY или  
O\_RDWR в open.  
- За да отворим файл в режим O\_WRONLY или O\_RDWR трябва да имаме право w. (за режима O\_RDWR е необходимо да имаме права r и w)  
- Право x е необходимо за да извикаме файл за изпълнение с примитив exec.  
За файл от тип каталог:  
- Право r означава правото да четем съдържанието на каталога. Напр., за  
изпълнение на командата ls -l dir се изисква право r за каталога dir.  
- Право w за каталог означава правото да създаваме или унищожаваме файлове от произволен тип в каталога. Напр., за изпълнение на командата rm dir/text се изисква право w за каталога dir.  
- Право x означава търсене на файлове в каталога и позициониране в каталог.  
Напр., за командите cat dir/text и cd dir се изисква право x за dir.  
Правата r и x при каталози действат независимо, x не изисква r и обратно,  
следователно при комбинирането им могат да се получат интересни резултати. Например, каталог с право x и без r за дадена категория потребители е така наречения "тъмен каталог". Потребителите може да имат право да четат файловете в каталога, само ако им знаят имената. Този метод се използва в FTP сървери, когато някои раздели от архива трябва да са достъпни само за “посветени” потребители.  
Битът S\_ISVTX в кода на защита, наречен Sticky bit (показва се като t при ls -  
l) , има интересна история. В ранните версии на Unix този бит е бил използван при файлове, съдържащи изпълним код на често използвани програми. Когато такава програма се извика за изпълнение за първи път, копие от образа на процеса остава в свопинг областта. Това позволява по-бързо зареждане в паметта при следващи извиквания на програмата. В съвременните Unix и Linux системи, реализиращи  
виртуална памет, тази техника вече не е нужна. Сега този бит се използва при каталози.  
Ако този бит не е вдигнат за каталог, е достатъчно процесът да има право w за каталога, за да може да унищожи всеки файл в него. Но ако битът е вдигнат за каталог, то процес може да унищожи файл от каталога ако има право w за каталога и е едно от трите:  
- собственик на файла  
- собственик на каталога  
- принадлежи на администратора(root)  
Така чрез Sticky bit може да се осигури допълнителна защита на файловете в каталога. Пример за каталог с вдигнат Sticky bit е /tmp, в който всички могат да създават файлове, но всеки може да изтрива само собствените си файлове.

Реализация на файлова система:  
Твърд диск – това е диск разделен на много „пътечки“:  
➡ Над повърхнината има електрическо устройство, което засича дали бита е 1 или 0;  
➡ Придвижването на главата на това устройство от една пътечка на друга е механично и става бавно;  
➡ Всяка пътечка е разделена на няколко сектора от по 512 или 1024 байта;  
➡ Главата не се допира до диска, а лети много малко над повърхността;  
➡ Софтуера знае колко е голям сектора и колко е времето за преминаване от един до друг сектор;  
➡ Файла се представя като много сектори;  
➡ Не са последователно разпределени байтовете на файловете, защото така не могат лесно да нарастват;  
➡ Файлът се разполага там където може, процесът на разхвърляне се нарича фрагментация.

Памет\_и\_управление\_на\_паметта  
Когато говорим за процесори и време се интересуваме само от текущите процеси,които чакат и са готови да смятат и има какво да смятат. Спящите процеси\_не\_ползват  
процесор, но когато говорим за памет – всички процеси използват памет, дори\_и\_спящите (те също са работещи програми, които се изпълняват). Физическите\_свойства\_на\_паметта  
са: енергозависимост (RAM са енергозависими, а HDD и flash паметите са енерго независими), време за достъп и многократно използване. Локалните данни,\_с\_които\_работи процеса, трябва да са си лично негови и да са предпазени(друг\_процес\_да\_не\_може\_да\_гидостъпва).Процесора\_има\_два\_режима:  
1. Real mode - програмата има достъп до цялата памет (до всички ресурси);  
2. Protected mode - програмата няма достъп до цялата памет, а само до тази, която\_и\_е\_предоставена,както\_и\_до\_някои\_инструкции.  
Техники\_за\_управление:  
Проста техника (сегментация): за паметта, която е предоставена на всеки процес имаме два регистъра: начало (f) и дължина (s) и проверяваме адреса на\_всеки\_байт,  
който използва този процес дали е в този интервал, т.е. ;  
IBM 360 техника (Paging): имаме n страници и всяка страница има атрибути, в които е записано как и кои процеси могат да използват тази страница.  
В съвременната операционна система аналог на Paging, но с доста свойства и  
хардуер, който да го поддържа, е известната с името „виртуална памет“ (VM). Виртуалната памет е абстрактно понятие. Всеки процес си има VM. Няма съответствие между VM и RAM. На някои страници от VM се съпоставя страница от RAM, на други може нищо да не е  
съпоставено, а на трети може да е съпоставена страница от HDD (hard disk drive).  
На всяка наша страница с данни се съпоставя страница от RAM и я използваме  
монополно. Може да има страници от RAM, които се използват от няколко процеса,но тези страници трябва да са само за четене. Затова трябва пак да има информация за всяка страница как ще се ползва, т.е. трябва да има тагове. Към всяка страница от VM има тагове, които казват къде е записана тя в RAM или HDD и казват за какво ще се ползва тази страница. За всяка страница от VM трябва да знаем къде се намира тя реално и тага  
ѝ. Тази информация се съдържа в таблица, която се нарича пряка адресна таблица,което означава, че за всеки процес имаме масив, който описва всяка негова страница. Има и друг вариант да се поддържа обща структура, която да показва тези данни за страниците от VM на всеки процес и тогава ключът в тази таблица ще е: <номер настраница, номер на процеса>, тази таблица се нарича обратна таблица. Memory management unit – в съвременния процесор го има и се грижи за адресните таблици, които са масиви някъде в паметта и има регистър, който сочи към тях. Видове кеш (три):

1. Кеш за инструкции – малко на брой дълги интервали, само се чете;

2. Кеш за данни – четене и писане на данни;

3. Кеш за таблиците за управление на виртуална памет.

TLB – Translation Lookahead Buffer: Кешове за няколко отделни таблици за управление на виртуална памет – съдържа 32-битови думи; Няма голям размер;Има две нива. При процесите делим паметта между тях и даваме на всеки толкова, колкото му трябва, но те са анонимни, те са парчета информация. При файлове, парчетата от памет трябва да са именувани и централния въпрос е за имената. Когато файла пази информация трябва да е в устойчивата памет. Когато променяме информация във файла, той трябва да е свързан с процес. Паметта може да я представим като множество от файлове – дълготрайно живеещи в информационната среда – информационни обекти (виж т. 13). Тъй като файловете ще  
съществуват дълго (те са споделяни между различни потребители и процеси) са важни имената им.

Алгоритъм на асансьора:  
Разместват се така заявките към четене и писане, че главата да изминава минимално разстояние;  
Има две приоритетни опашки: едната се състои от файове, които се намират по посока на движението на главата, а другата от файове, които се намират в обратната посока;  
Ако е празна главата на опашката по посока на главата на устройството, то се сменя посоката.  
Двете стратегии – кеширане и алгоритъмът на асансьора пораждат проблем за надеждността (внезапно спиране на тока). За да се справим с този проблем съществуват така наречените журнали (лог файлове). Записваме промените във файл, който се нарича журнал, в него се запазва пълната информация за операциите над файлове и когато се  
позапълни/препълни журнала, спираме и отразяваме операциите от журнала над файловете, ако спре тока журнала ще пази последните промени, а файловете ще са сиконсистентни. Четат се първо старите файлове и се проверява дали в журнала са  
променени, ако спре тока, докато пишем операцията/транзакцията, тя няма да се е изтрила преди да е завършила и затова ще я пазим все още в журнала. Във файловите системи транзакцията е елементарна файлова операция (read, write, open). Журналът (лог файла) се намира или в друг диск или в друг дисков дял, за да може двете паралелно да работят, но в персоналните устройства журналът е файл в самата файлова система или в същия дял.  
Две фази:  
1. Файловите операции в истинския ред по време се записват в журнала;  
2. Когато се натрупат се подават заявките на алгоритъма на асансьора.

Drugata strategiq e da vkarame nay-polzvanite chastti na pametta v RAM-ta