Lokalne Systemy Plików

Eryk Kopczyński
Maciej Osiński
Katarzyna Sokołowska

Systemy Operacyjne '2002

W programie

W programie

- Wprowadzenie:
  - ⋆ Do czego służy księgowanie ?

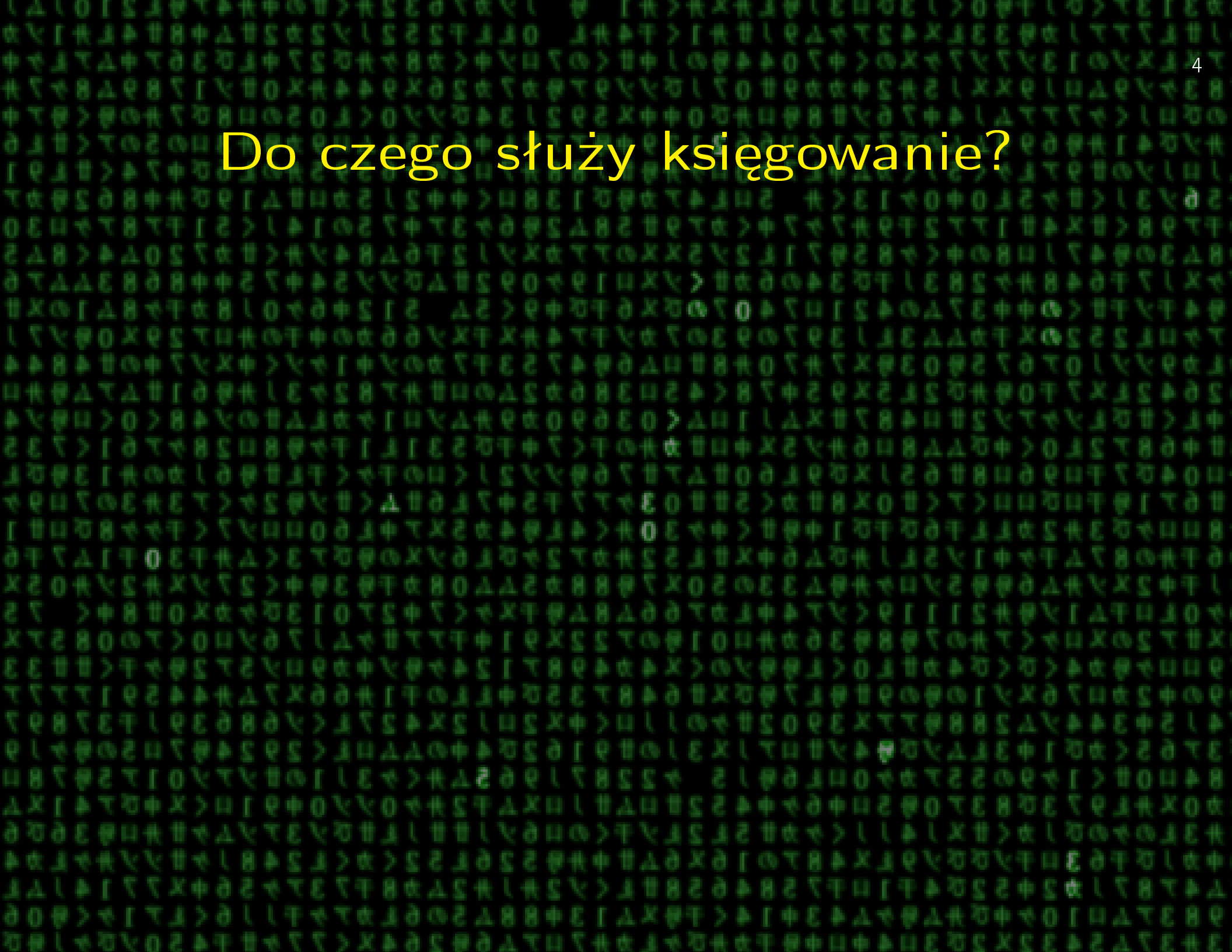
W programie

- Wprowadzenie:
  - ⋆ Do czego służy księgowanie ?
- Wypalmy sobie płytkę
  - ⋆ Iso9660
  - ⋆ Rocky Ridge
  - ⋆ Joliet
  - ⋆ UDF

- Powered by Bill
  - ★ FAT, FAT32
  - ★ NTFS, NTFS 5

- Powered by Bill
  - ⋆ FAT, FAT32
  - ⋆ NTFS, NTFS 5
- Powered by Penguin
  - ⋆ Minix, Ext, Ext2, Ext3
  - ⋆ EFS, XFS
  - ⋆ JFS
  - ⋆ ReiserFS 3.6, ReiserFS 4
  - ⋆ Wydajność

Do czego służy księgowanie?



Do czego służy księgowanie?

Księgowanie umożliwia łączenie operacji na systemie plików w niepodzielne transakcje. Niepodzielne, czyli cała transakcja będzie wprowadzona lub odrzucona.

Księgowanie polega na zapisywaniu do pewnego miejsca (*log*) dokonywanych zmian przed zapisaniem ich na docelowe miejsce.

Jeśli zapis do logu się powiedzie, co sprawdzamy przez *punkty kontrolne*, to transakcja jest zatwierdzana (*commit*) i zmiany nie mogą przepaść.

Dopiero teraz zmiany są nanoszone na docelowe miejsce.

W razie awarii będzie można nanieść te zmiany (jeśli zostały zatwierdzone) lub odrzucić (jeśli transakcja niezatwierdzona) bez ryzyka utraty spójności danych na dysku.

Księgowanie zapewnia spójność systemu oraz znacznie skraca czas wstawania systemu po awarii (nie trzeba sprawdzać całego dysku). Zysk może oznaczać naprawianie systemu przez kilka sekund zamiast kilku godzin (duże serwery).

Rodzaje księgowania

Rodzaje księgowania

tylko meta-dane księgujemy wyłącznie struktury systemu plików; zaletą jest duża wydajność

Rodzaje księgowania

tylko meta-dane księgujemy wyłącznie struktury systemu plików; zaletą jest duża wydajność

pełne księgujemy meta-dane oraz same dane; zaletą jest pełna spójność, wadą wydajność

Rodzaje księgowania

tylko meta-dane księgujemy wyłącznie struktury systemu plików; zaletą jest duża wydajność

pełne księgujemy meta-dane oraz same dane; zaletą jest pełna spójność, wadą wydajność

W przypadku księgowania meta-danych mamy pewność, że system plików jest w spójnym stanie, ale możemy stracić modyfikowane dane !

Księgowanie – uwaga na notebooki

Księgowanie – uwaga na notebooki

Mechanizm księgowania zakłada, że operacja *flush* rzeczywiście zapisuje na dysk.

W przypadku niektórych dysków przeznaczonych dla notebooków to nie skutkuje (oszczędzanie baterii).

System plików może uznać, że zmiany zostały zapisane fizycznie na dysku, podczas gdy będą w pamięci podręcznej dysku. Awaria (nawet rozładowanie baterii) w takim momencie może doprowadzić do utraty spójności struktur FS.

Jest to problem związany z dyskiem i żaden FS nie potrafi sobie z nim poradzić. Jeśli nie jesteśmy pewni, że nasz dysk nie posiada tej cechy, to nie należy używać księgowania !

Wypalmy sobie płytkę

Wypalmy sobie płytkę

ISO 9660 podstawowy sposób nagrywania płytek CD-ROM

Joliet rozszerzenie ISO 9660 dla Windows

Rocky Ridge rozszerzenie ISO 9660 dla systemów Unikso-wych

UDF standard dla DVD-ROM i CD-RW

ISO 9660

ISO 9660

- Jednokrotny zapis – od razu ostateczna wersja
- stosowana alokacja ciągła
- ograniczenia (nie zawsze przestrzegane):
  - ⋆ 8 poziomów zagnieżdżeń katalogów
  - ⋆ nazwy plików do 31 znaków [A-Z0-9_.]

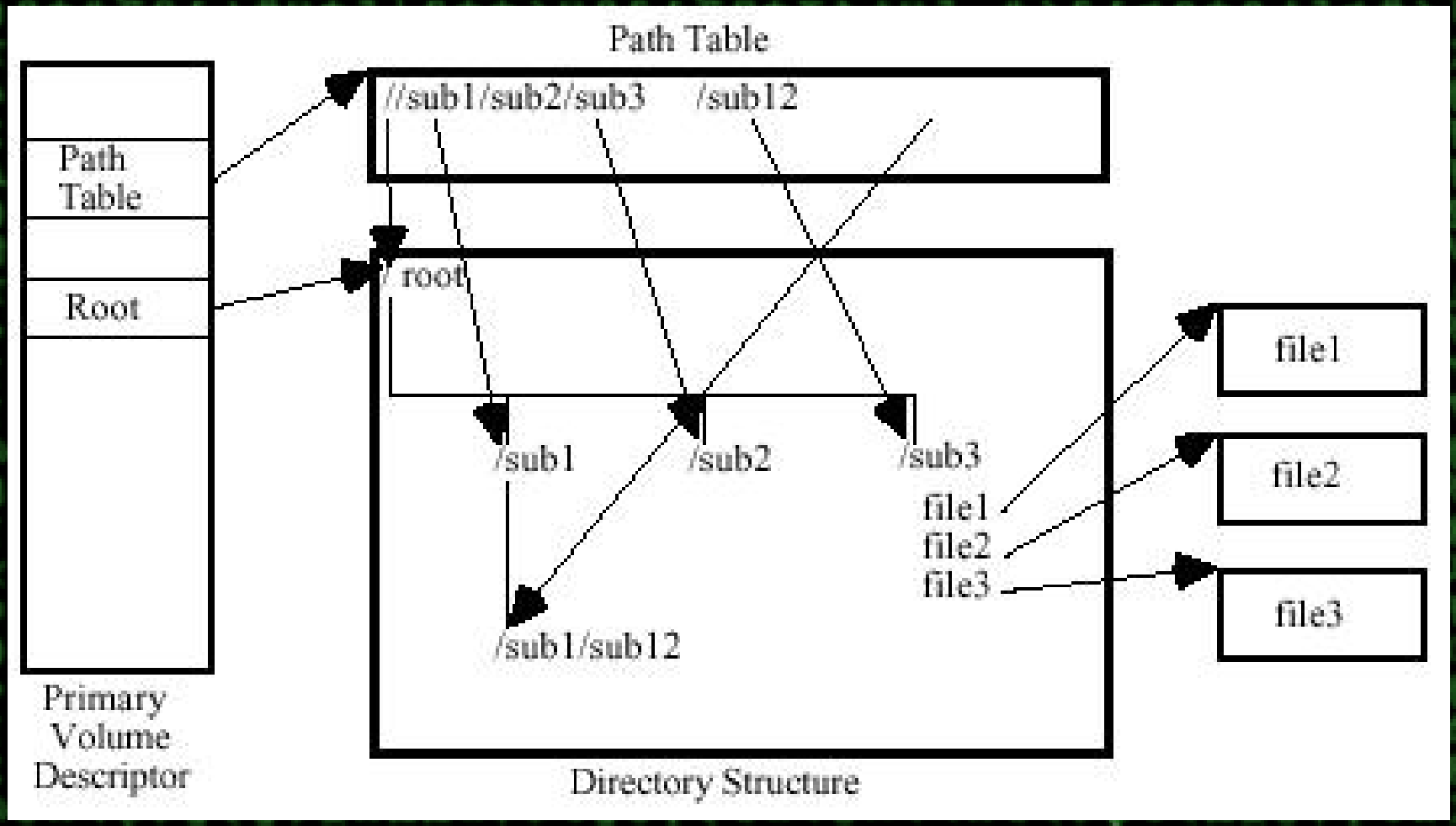
ISO 9660 – struktura

- sektor wielkości 2048 bajtów
- deskryptor *Volume Descriptors* (jeden lub więcej)

  system operacyjny (może być kilka deskryptorów)
  nazwa dysku
  path_table długość i początek
  katalog główny długość i początek

ISO 9660 – *path_table*

- lista wszystkich katalogów
- najpierw katalog główny, potem alfabetycznie podkatalogi
- wpisy zmiennej długości (maks. 31 znaków)
- różne deskryptory mogą mieć różne tablice


Path Table
//sub1/sub2/sub3
/sub12
Path Table
Root
/ root
/sub1
/sub2
/sub3
/sub1/sub12
file1
file2
file3
file1
file2
file3
Primary Volume Descriptor
Directory Structure

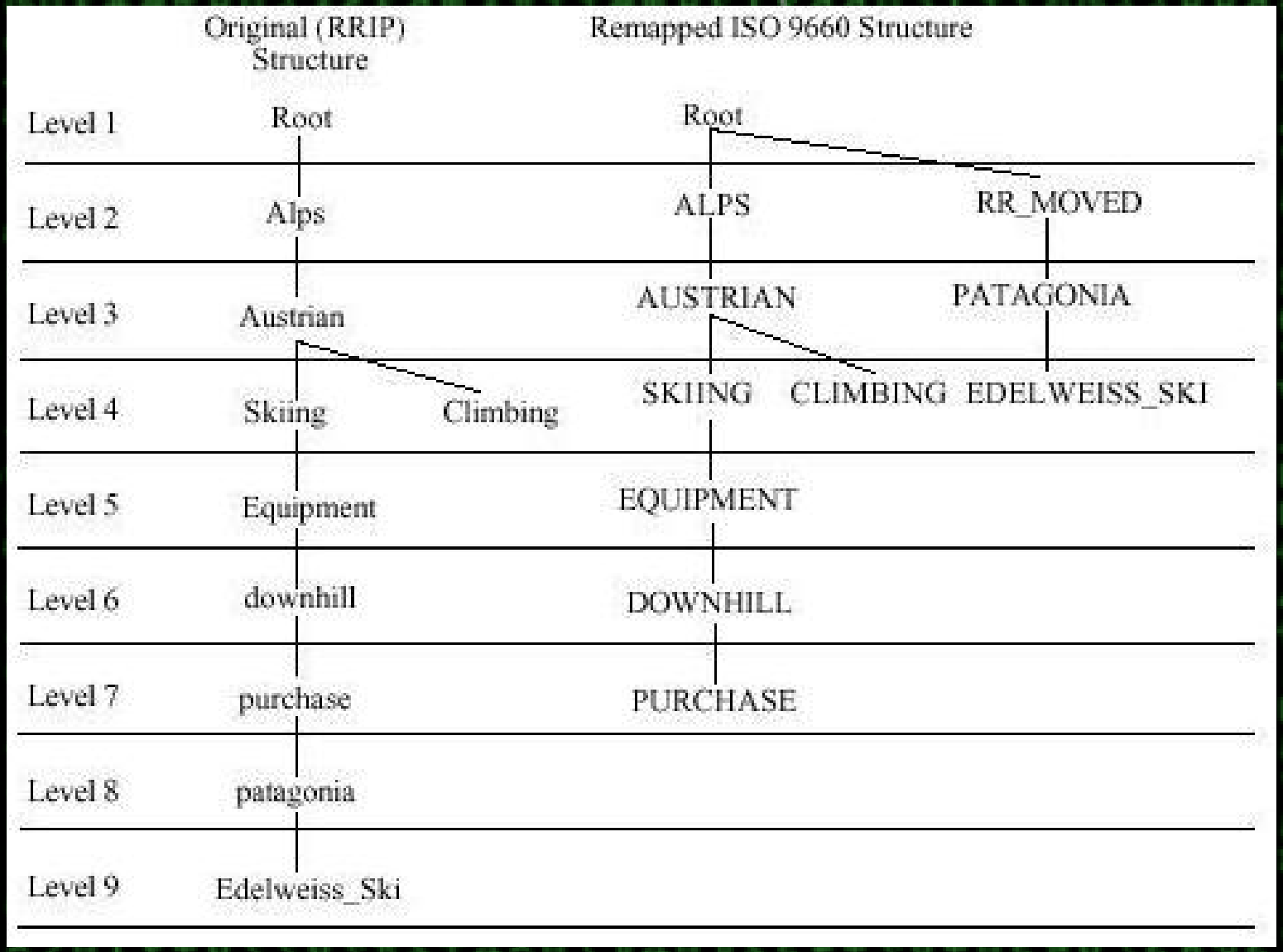
ISO 9660 – pliki & katalogi

- katalog jak plik (+specjalny atrybut)
- wpisy w katalogach:
  - ⋆ każdy dla jednego pliku lub katalogu
  - ⋆ pierwszy sektor, długość, atrybuty, nazwa, czas
  - ⋆ wpisy uporządkowane alfabetycznie

ISO 9660 – rozszerzenia

ISO 9660 – Rocky Ridge

- rozszerzenie standardu ISO 9660
- prechowywanie atrybutów uniksowych (właściciel, uprawnienia, dowiązania symboliczne)
- obejście oganiczenia zagnieżdżenia katalogów


Original (RRIP) Structure
Remapped ISO 9660 Structure
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Level 8
Level 9
Root
Alps
Austrian
Skiing
Climbing
Equipment
downhill
purchase
patagonia
Edelweiss_Ski
Root
ALPS
RR_MOVED
AUSTRIAN
PATAGONIA
SKIING
CLIMBING
EDELWEISS_SKI
EQUIPMENT
DOWNHILL
PURCHASE

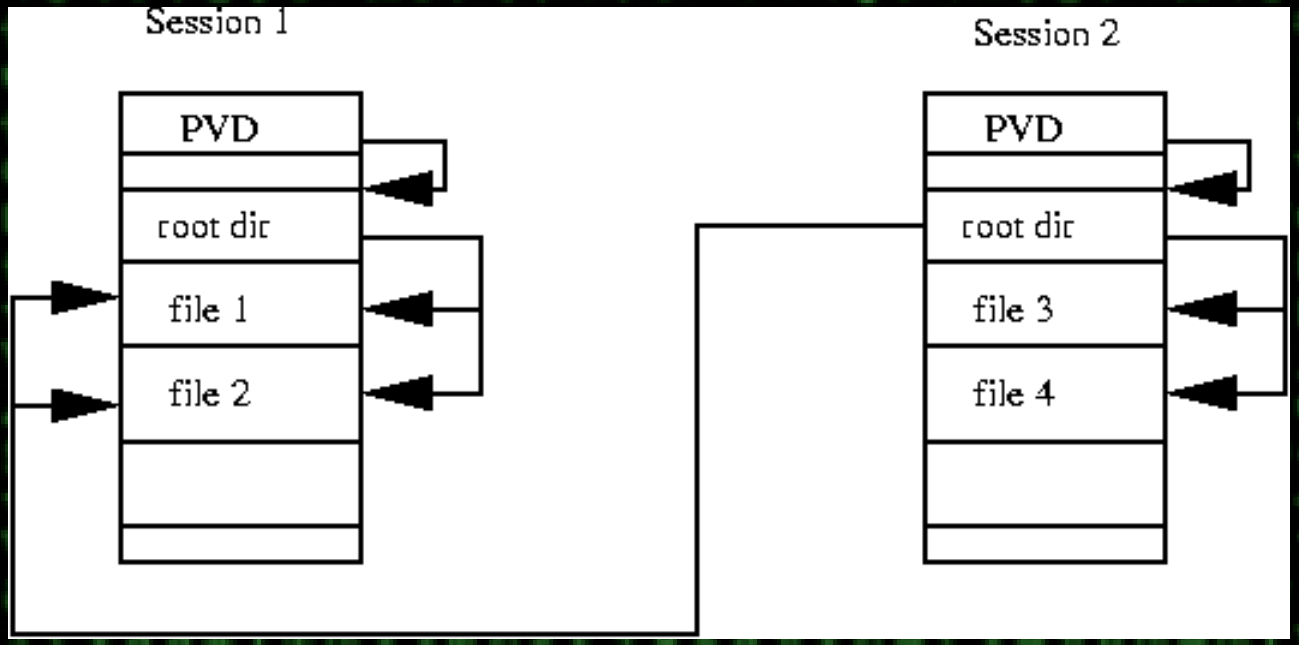
Joliet

Joliet

- stworzone dla Windows 95
- nazwy w Unicodzie, długość do 64 znaków
- zniesione ograniczenie do 8 poziomów katalogów
- informacje trzymane niezależnie od danych ISO 9660

Dyski CD-R, sesje

Dyski CD-R, sesje



Nowe *meta-dane* (deskryptor dysku, tablica ścieżek, dane o katalogach).

UDF

UDF

- Universal Disk Format
- mniej ograniczeń
- dla płyt DVD-ROM oraz CD-RW
- można zmieniać dane

Płytki – podsumowanie

ISO 9660 alokacja ciągła, krótkie nazwy plików (31 znaków), różne deskryptory dla różnych SO

Joliet dłuższe nazwy plików (64 znaki), dane niezależne od danych ISO 9660

Rocky Ridge atrybuty Uniksowe

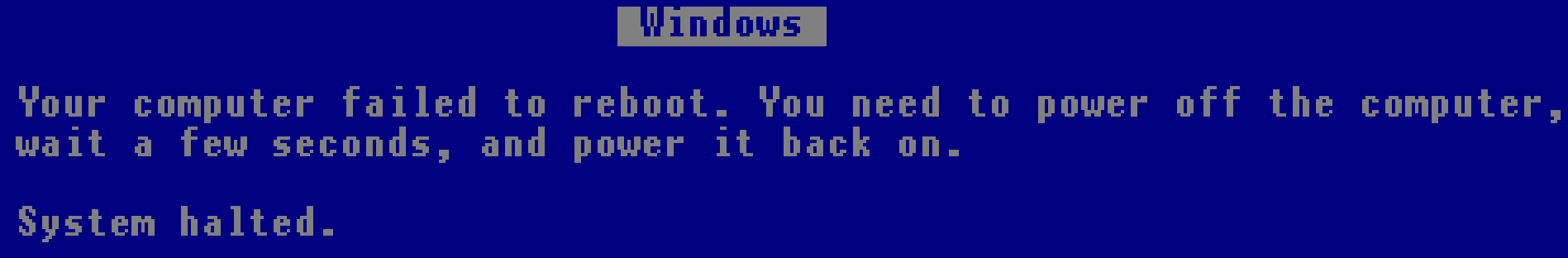
UDF nowy system dla DVD-ROM

Windows

Your computer failed to reboot. You need to power off the computer, wait a few seconds, and power it back on.

System halted.

Powered by Bill G.



Powered by Bill G.

Powered by Bill G.

FAT system z MS DOS

FAT 32 najpopularniejszy system plików ?

NTFS 4 nowa technologia

NTFS 5 i jeszcze nowsza

FAT

FAT

- 16-bitowy (12 dla dyskietek)
- ograniczenie do 2 GB (4 GB w NT)
- zastosowanie:
  - ⋆ małe dyski (do 200 MB); oszczędność miejsca
  - ⋆ dyskietki (FAT 12)

FAT – struktura

Boot sector

Tablica FAT (*File Allocation Table*)

Kopia Tablicy FAT operacje na obu

Katalog główny rozmiar ustalony przy formatowaniu

Obszar danych podzielony na *clustery* (kilka sektorów)

FAT – Tablica FAT

- 16 bitów dla każdego *clustera*:
  - ⋆ wolny / uszkodzony / należy do pliku
  - ⋆ następny *cluster*
- ograniczenie do 2^{16} clusterów
- odczyt z oryginału, kopia do naprawiania

FAT – pliki & katalogi

- wpisy 32-bitowe:

  nazwa 8+3 (obchodzone w Windows)
  rozmiar, czas modyfikacji
  pierwszy kluster
  atrybuty

- atrybuty plików:

  archive
  read only
  hidden, system
  directory (tylko główny ma ustalony rozmiar)
  volumeID fikcyjny plik – etykieta dysku

FAT

- 16-bitowy
- system podzielony na clustry
- stosować dla małych dysków i dyskietek
- dla dużych dysków duże clustry, czyli fragmentacja

FAT 32

FAT 32

- Windows 95 (OEM 2)
- podstawowy system w Windows 9x
- 32-bitowe wpisy
- duże dyski

FAT/FAT 32 – długie nazwy

- obchodzenie ograniczenia 8+3
- unikatowy krótki wpis (MOJDOK~3.TEX)
- zamaskowany długi (w kawałkach)
- obsługa wielkości liter

FAT 32 a FAT

- FAT na małych dyskach (do 200 MB)
- FAT 32 na dużych dyskach (2 GB) – cluster 4kB dla FAT 32 i 32 kB dla FAT
- zmienny rozmiar katalogu głównego
- FAT 32 działa po uszkodzeniu oryginału FAT


Microsoft®
Copyright © 1985-1996
Microsoft Corporation
Microsoft®
Windows NT®
Workstation 4.0
oraz Microsoft Internet Explorer
Ten program jest chroniony prawem autorskim zgodnie z opisem w oknie dialogowym Windows NT - informacje.

NTFS 4

NTFS 4

- New Technology File System
- Windows NT 4
- wersja 1.1, czyli 4.0
- zgodny z POSIX.1

NTFS – struktura

- cluster z reguły 4 kB
- wszystko poza *bootsectorem* to pliki
- *meta-dane* to też pliki
- a pliki mogą być umieszczane wszędzie

NTFS – meta-dane

nazwy zaczynają się od $

$MFT Master File Table

$Logfile ostatnie modyfikacje

$Bitmap mapa wolnych sektorów

$Volume o dysku (m.in. nazwa, wersja NTFS)

NTFS – MFT

- Master File Table
- każdy plik ma wpis (ustalonej długości)
- plik to kolekcja atrybutów
- atrybuty plików we wpisie lub na zewnątrz
- małe pliki przechowywane we wpisie

NTFS – MFT – atrybuty plików

Data zawartość pliku

Name nazwa (Unicode, do 256 znaków), opcjonalnie też 8+3

Standard Information m.in. czas

programy mogą dodawać własne

NTFS – inne

kompresja wybranych plików, przezroczysta dla użytkownika

księgowanie metadanych, przez *$Logfile*

b-drzewa dla plików w katalogu

NTFS 5

NTFS 5

- Windows 2000
- wersja 3.0, czyli 5.0

NTFS 5 – szyfrowane pliki

- Encrypting File System
- szyfrowanie plików, uzupełnienie uprawnień
- ochrona przed montowaniem z innego SO
- przezroczyste dla użytkownika

NTFS 5 – sparse files

- obsługa rzadkich plików
- szybsze od kompresjii
- fragmenty puste (0) nie zajmują dysku

NTFS 5 – reparse points

- specjalny znacznik dla pliku/katalogu
- program zastępuje operacje na pliku
- montowanie
- symulowanie dowiązań symbolicznych

NTFS 5 – inne

quota

księgowanie dodane *change Journal*

- zmiany w systemie plików
- jaki plik, jak zmieniony
- bez treści zmian

Powered by Bill G.

FAT MS DOS

- system 16-bitowy (dyski do 2 GB)
- nazwy 8+3

FAT 32 Windows 95

- system 32-bitowy
- również dyski powyżej 2 GB

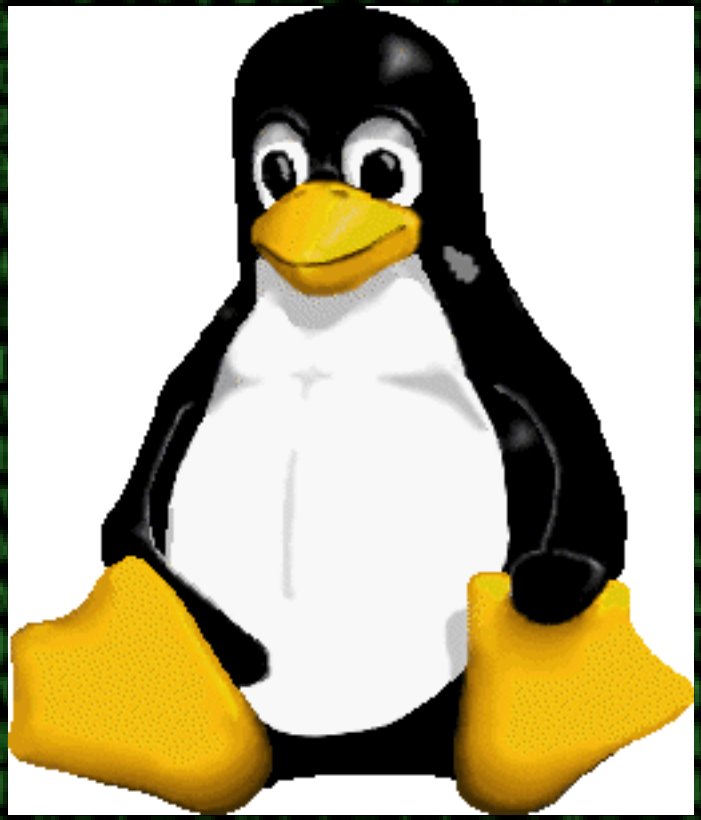
NTFS Windows NT 4

- efektywny (b-drzewa)
- księgowanie meta-danych
- uprawnienia
- nazwy 256 (Unicode)

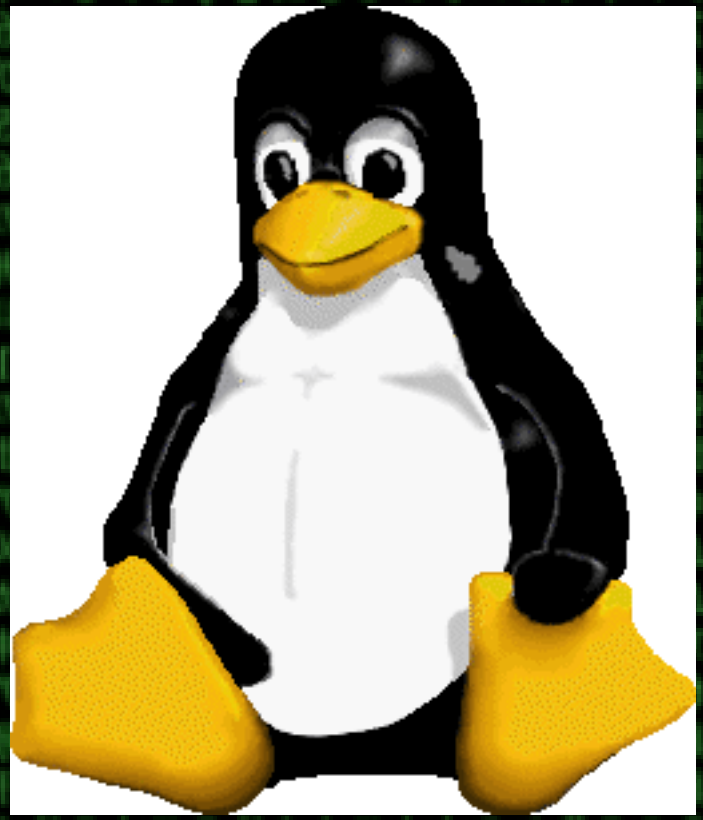
NTFS 5 Windows 2000

- Encrypted File System, sparse files, quota

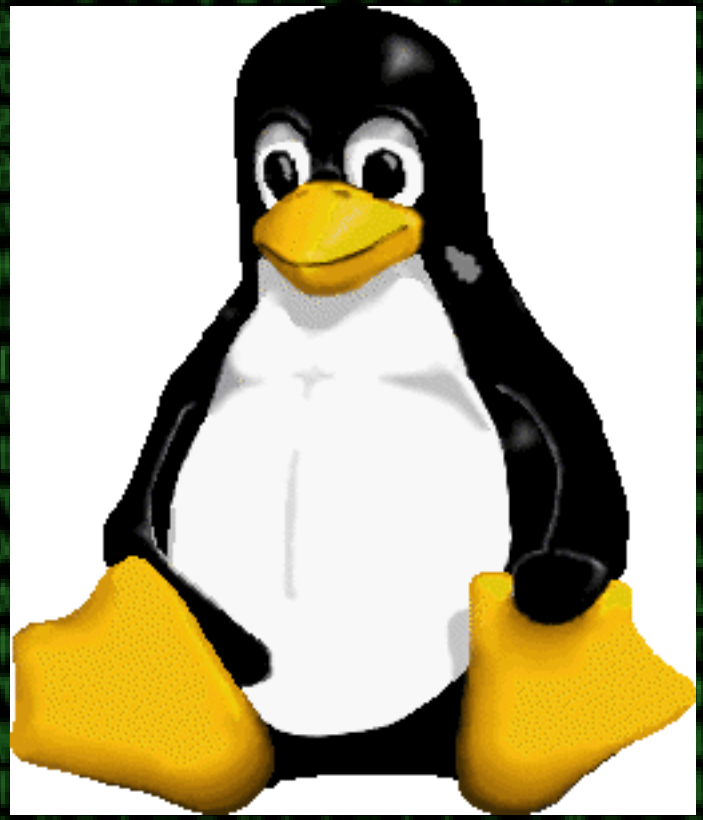
Można teraz bezpiecznie wyrzucić komputer.



Powered by Penguin



Powered by Penguin



- Minix, Ext
- Ext2, Ext3
- EFS, XFS
- JFS
- ReiserFS 3.6 i 4

Minix

Minix

- pierwszy system plików w Linuksie
- wydajny i stabilny (w 1991)
- plik i system plików – 64 MB
- nazwa do 16 znaków (30 w nowszej wersji)
- używany w nauczaniu SO :)-, czasem dla dyskietek

Ext

- od 1992, po dodaniu VFS
- wolne i-węzły i bloki w listach (nieposortowane) – duża fragmentacja i mała wydajność
- plik i system plików – 2 GB
- nazwy do 255 znaków

Ext 2

Ext 2

- 1993 w celu zastąpienia Ext
- oparty na Ext i FFS
- najpopularniejszy FS w Linuksie
- silne narzędzia – fsck
- powstało wiele dodatków i rozszerzeń

Ext 2 – struktura

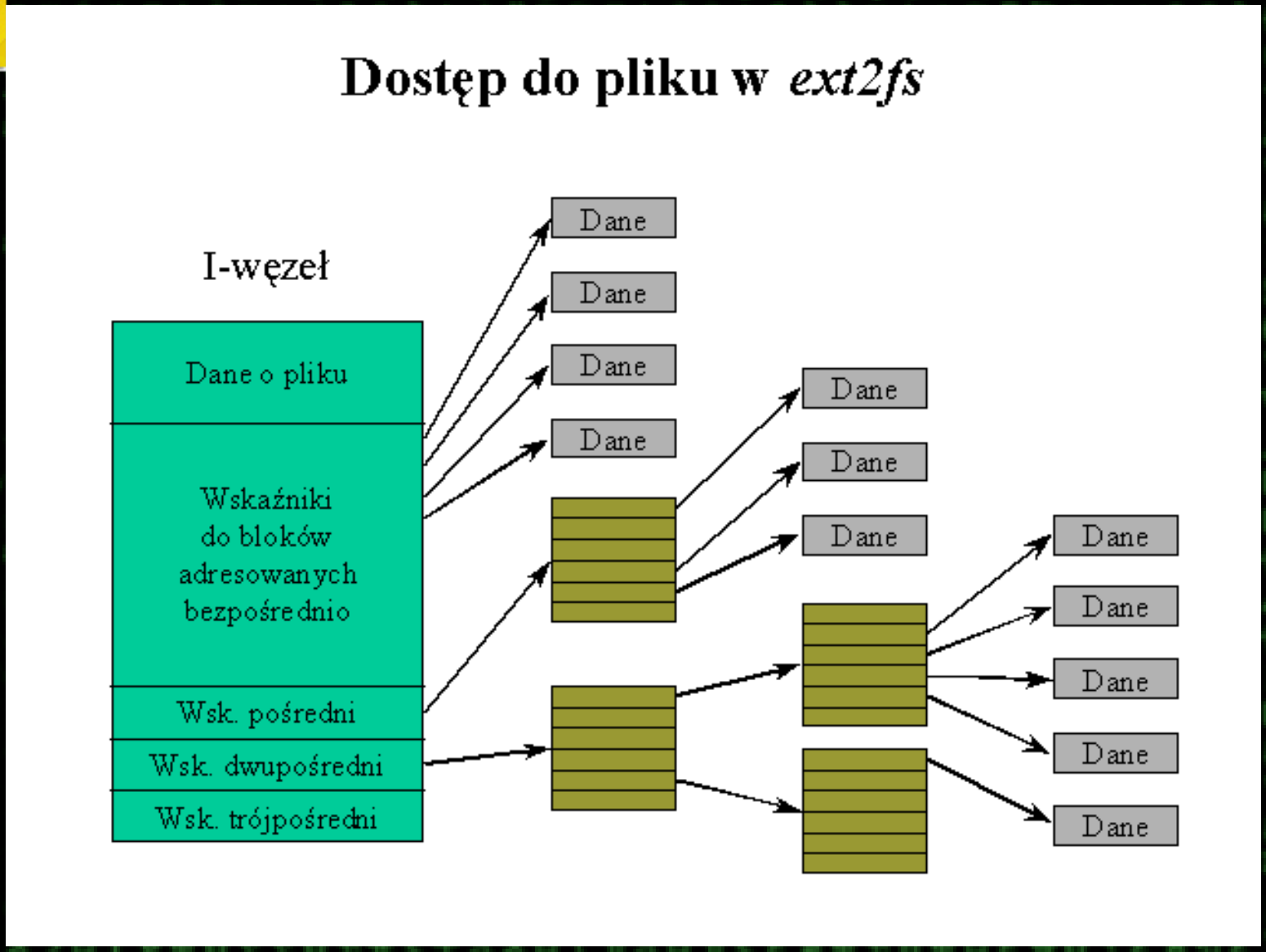
- superblok
- grupy bloków :
  - **kopia superbloku** redundancja
  - **GDT** tablica deskryptorów grup (redundancja)
  - **bitmapa bloków i i-węzłów**
  - **tablica i-węzłów**
  - **bloki danych**
- **bloki** podstawa budowy danych

i-węzły podstawa budowy meta-danych

adresowanie bloków

- **bezpośrednie** (małe pliki)
- **pośrednie**
- **2-pośrednie**
- **3-pośrednie**

katalogi listowe struktury



Adresowanie bezpośrednie i pośrednie w Ext2

Ext 2 – indeksowane katalogi

- jedno z rozszerzeń Ext2
- flaga *EXT2_INDEX_FL* i-węzła
- pierwsze 512 bloków katalogu używane jako bloki indeksujące (drzewo, korzeń w bloku 0.)
- dane od bloku 512. (liście)

Ext 2 – blok indeksujący

- zawiera pary <**klucz**, **adres**>:

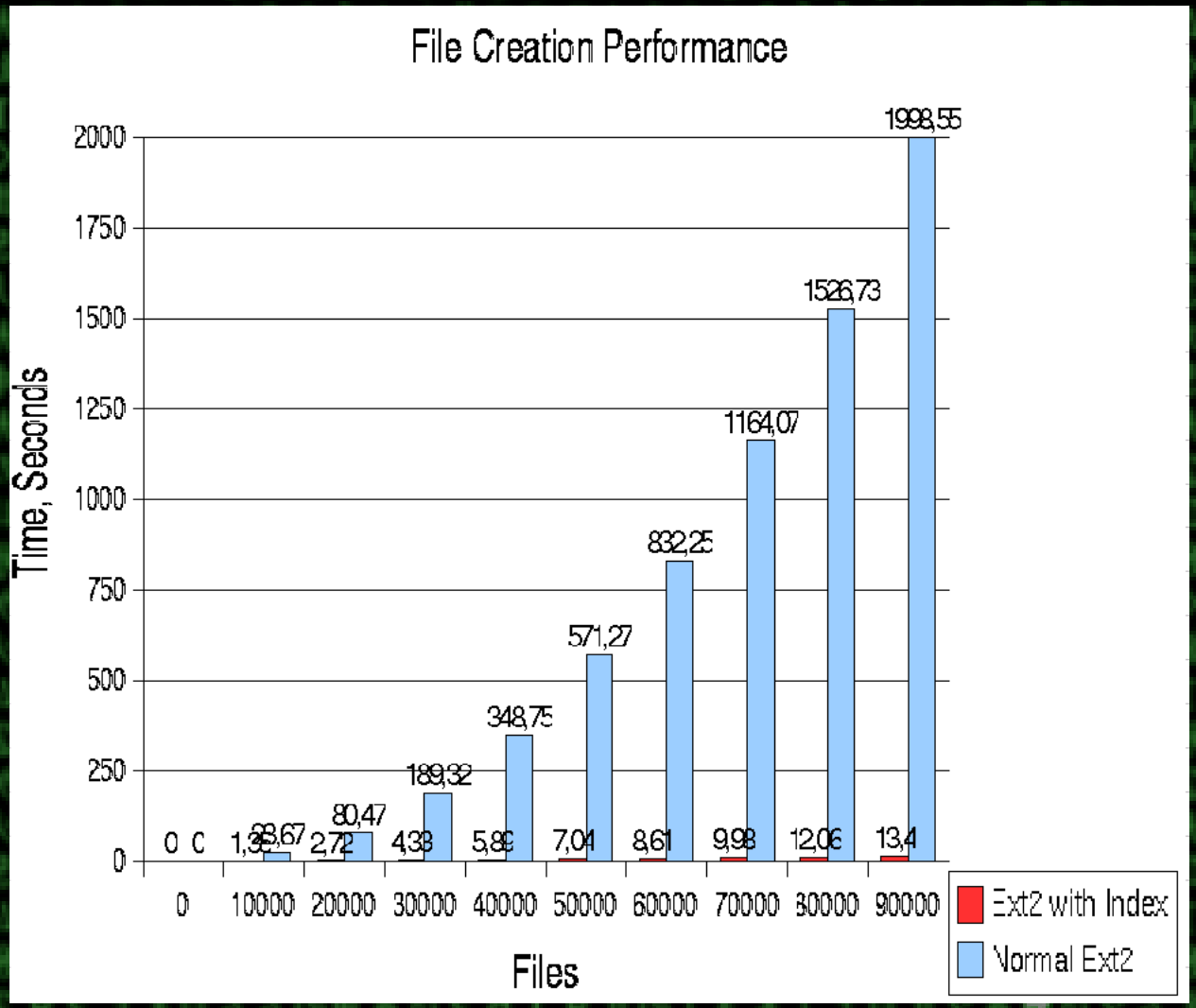
  klucz haszujący, najmniej znaczący bit to *znacznik kolizji*
  (czy kolidujące klucze rozbite pomiedzy bloki indeksujące)
  adres bloku lub bloku indeksującego (jesli wielopoziomowo)

- 512 bloków obsługuje około 90 000 plików

- kolejny poziom – około 50 000 000 plików

Ext 2 – indeksowanie – kolizje

- staramy się umieszczać te same klucze w jednym bloku
- jeśli nie, rozbijamy na dwa i ustawiamy w pierwszym bloku **znacznik kolizji**
- w 0. bloku struktury indeksującej (korzeń) id funkcji haszującej, więc można ją podmieniać, aby zwiększyć wydajność



Indeksowane katalogi a wydajność Ext2

Ext 2

- sprawdzony, stabilny system
- brak księgowania
- proste struktury danych (niska wydajność)
- słaba wydajność dla dużych plików (adresowanie pośrednie)

Ext2 – wada

Podstawową wadą Ext2 jest brak journalingu – duży serwer może wstawać przez kilka godzin.

Stabilność i zaufanie użytkowników oraz problemy z odtwarzaniem po awarii były motywacją powstania Ext3.

Ext3

Ext3

- dr Stephen Tweedie, 1999r.
- wsparcie RedHat
- oparty na Ext2, odziedziczył dużą część kodu
- dodano mechanizm księgowania
- kilka usprawnień (m.in. szybki zapis synchroniczny)

Ext3 – kompatybilność z Ext2

- poprawnie odmontowany Ext3 można montować jako Ext2
- w drugą stronę ma działać, ale jeszcze nie działa
- Ext3 to usprawnienie Ext2, a nie tworzenie nowego FS; zalety:
  - ⋆ odziedziczenie stabilności, przewidywalności
  - ⋆ zaufanie użytkowników
  - ⋆ silne narzędzia (*fsck*)
  - ⋆ łatwa migracja (*tune2fs*), nawet bez odmontowywania

Ext3 – struktura

brak istotnych zmian w stosunku do Ext2

superblok 3 dodatkowe pola

- **s_journal_inum** i-węzeł, od którego zaczyna się journal
- **s_journal_dev** urządzenie, na którym jest journal
- **s_last_orphan** lista i-węzłów do usunięcia

indeksowane katalogi implementowane przez drzewa r-b

- **hash** klucz, część podstawowa
- **minor** niższa część klucza (klucz 64-bitowy), obecnie nie-używana (problemy z VFS)
- **file_type** trzymamy typ pliku, żeby nie zaglądać do i-węzła
- **next** lista, przy kolizji kluczy

Ext3 – księgowanie

- niezależny od Ext3
- interfejs JBD (Journaling Block Devices)
- przechowywanie całych zmienionych bloków
- ściskanie (*squish*) bloków
- 3 tryby księgowania

Ext3 – tryby księgowania

data=writeback księgowanie tylko meta-danych

data=journal pełne księgowanie (meta-dane i dane); duży journal zwiększa wydajność

data=ordered meta-dane, ale wyniki zbliżone do pełnego:

- atomowe transakcje – modyfikacje danych i metadanych
- zapis danych (na dysk)
- zapis meta-danych do logu

Ext3 – księgowanie od środka

- modyfikowane bloki w buforach (*journal_block_tag_t*)
- bufory łączone w transakcje (*transaction_t*)
- journal jeden dla FS (*journal_t*)
- superblok journala (*journal_superblock_t*)
- punkty kontrolne i zatwierdzanie (*commit*)
- flush, w razie awarii odtwarzanie z logu

Ext3 – journal_block_tag_t

t_blocknr numer bloku na dysku

t_flags flagi

Ext3 – transaction_t

t_journal journal dla transakcji

t_state stan transakcji; flagi:

- T_RUNNING akceptuje zmiany
- T_LOCKED zmiany w trakcie, nowych nie przyjmujemy
- T_FLUSH zmiany zakończone, zapisujemy na dysk
- T_COMMIT dane zapisane, zapisujemy „commit record”
- T_FINISHED zakończone, transakcję przechowujemy dla checkpoint'ów

t_log_start gdzie w logu zaczyna się commit dla tej transakcji

t_buffers lista cykliczna (dwukierunkowa) buforów dla metadanych transakcji

t_nr_buffers liczba buforów na liście *t_buffers*

t_reserved_list zarezerwowane dla transakcji bufory, które nie były jeszcze modyfikowane

t_sync_datalist lista buforów, które należy „flushować” przed operacją commit (ochraniane przez *journal_datalist_lock*)

t_checkpoint_list jw, ale przed osiągnięciem checkpoint

t_cpnext, t_cpprev następna i poprzednia transakcja, spośród oczekujących na checkpoint

t_forget lista buforów do usunięcia, po operacji commit

Ext3 – journal_t

j_flags flagi:

- **JFS_UNMOUNT** wątek księgowania jest niszczony
- **JFS_ABORT** księgowanie anulowane z powodu błędów
- **JFS_FLUSHED** superblok journala został zapisany
- **JFS_LOADED** superblok journala został załadowany

j_superblock superblok dla journala

j_running_transaction bieżąca transakcja

j_commiting_transaction transakcja wysyłana do zapisu na dysk

j_wait_transaction_locked kolejka transakcji oczekujących na commit

j_wait_done_commit jw, na zakończenie potwierdzania

j_wait_logspace jw, na zakończenie punktów kontrolnych

j_wait_updates jw, na zakończenie modyfikowania

j_checkpoint_sem semafor do ochrony punktów kontrolnych

j_sem główny semafor

j_head pierwszy nieużywany blok w journalu

j_tail najstarszy używany blok w journalu

j_free ile wolnych bloków

j_first, j_last numer pierwszego bloku, który można użyć oraz pierwszego za ostatnim

j_dev, j_blocksize, j_blk_offset urządzenie oraz rozmiar bloku dla lokalizacji journala

j_fs_dev urządzenie, n aktórym trzymany jest księgowany system plików; jeśli lokalnie, to równe *j_dev*

j_maxlen maksymalny rozmiar dla journala

j_tail_sequence najstarsza transakcja w logu (numer)

j_commit_sequence transakcja ostatnio zatwierdzona

j_max_transaction_buffers ograniczenie na liczbę bloków z meta-danymi dla transakcji

j_commit_interval ograniczenie na czas trwania transakcji przed zatwierdzeniem

j_all_journals lista wszystkich journali

Ext3 – journal_superblock_t

s_blocksize rozmiar bloku

s_maxlen ilość bloków w pliku journala

s_first pierszy blok dla logu

s_max_transaction limit bloków dla transakcji

s_max_trans_data limit bloków danych dla transakcji

Ext3

- Ext2 + journaling
- zbliżona wydajność (mały narzut księgowania)
- stabilny system, wspierany przez RedHat.com

EFS

EFS

- SGI, dla systemu IRIX
- partycje dyskowe (do 8 GB) i CD-ROM (nie-ISO)
- zastąpiony przez XFS
- niska wydajność i skalowalność
- rozmiar pliku do 2 GB

- długie podnoszenie po awarii (fsck)

XFS

XFS

XFS

- Silicon Graphics
- powstał w 1994 (IRIX)
- zastąpił EFS w 1996
- od 1999 jako open-source dla Linuksa

XFS – cele projektowe

- szybkie wstawanie po awarii
- duże systemy plików
- duże pliki
- duża liczba plików
- duże katalogi
- obsługa rzadkich plików

XFS – ograniczenia?

ograniczenia dla systemu to 2^{64} B (IRIX) i 2 TB (Linux)

blok 512 B do 64 kB

plik 2^{63} B (IRIX) i 64 TB (Linux, strona 16 kB)

XFS – architektura modułowa

- menadżer przestrzeni
- menadżer we/wy
- menadżer katalogów
- menadżer transakcji
- menadżer wolumenów
- bufor, sterowniki dyskowe

XFS – menadżer przestrzeni

- alokacja przestrzeni dyskowej dla plików
- odwzorowywanie plik -> bloki dyskowe
- zarządza grupami alokacji, i-węzłami, wolną przestrzenią

XFS – grupy alokacji

od 16 MB do 4 GB

wielowątkowość i wieloprocesorowość

wolne ekstenty 2 b+drzewa (położenie, rozmiar)

i-węzły przydzielane dynamicznie po 64

numer nr AG, nr i-węzła w AG

b+drzewo i-węzłów w AG (numer, lokalizacja)

XFS – menadżer we/wy

- pomiędzy VFS IRIX'a i menadżerem przestrzeni
- obsługa żądań we/wy
- wysokopoziomowa obsługa plików (przez menadżera przestrzeni)

XFS – menadżer katalogów

- obsługa katalogów
- wewnątrz i-węzła lub w liściu (duży katalog)
- bez sortowania nazw – szukamy liniowo

XFS – menadżer transakcji

- aktualizacja danych i księgowanie
- księgowanie meta-danych

XFS – menadżer wolumenów

- warstwa pomiędzy XFS a urządzeniami dyskowymi

XFS – Linux (a IRIX)

- blok ograniczony rozmiarem strony
- system plików do 2 TB
- interfejs do quotactl() i narzędzi quoty
- maksymalny offset w pliku to 16 TB (strona 4kb) lub 64 TB (strona 16kb)

XFS – podsumowanie

- dla dużych systemów
- AG wspierają wielowątkowość
- skalowalność
- architektura modułowa
- przenoszony z IRIX na inne platformy (Linux)

Journaled File System

Journaled File System

JFS

- Journaled File System
- system 64-bitowy
- ograniczenie 4 PB na system (blok 4kB)
- system zorientowany na transakcje
- cele: skalowalność, niezawodność, wydajność

JFS – struktura

blok 512, 1024, 2048 lub 4096 B

ekstent zestaw ciągłych bloków agregatu

AG grupy alokacji

- grupowanie powiązanych danych
- maks. 128 grup
- każda min. 8192 bloki agregatu

JFS – struktura

agregat

JFS – struktura

agregat

część lodówki

JFS – struktura

agregat

- część lodówki

OUPS ..

JFS – struktura

agregat obszar alokacji przestrzeni dyskowej

- jeden na partycji
- tablica bloków dyskowych
- superblok
- mapa alokacji (stan każdego bloku)

JFS – struktura

zestaw plików

- grupa plików i katalogów
- niezależne, montowalne poddrzewo
- cały w jednym agregacie

JFS – ekstent

- wewnątrz jednego agregatu, duży w kilku AG
- podstawa struktur adresowych JFS
- odwzorowanie logicznych offsetów na adresy dyskowe
- alokowany dla pliku
- <offset_logiczny, długość, adres_fizyczny>
- b+drzewo trójek (offset jako klucz) w i-węźle

JFS – ekstent – xad

- deskryptor alokacji ekstentu (**xad**)
- opisuje 2 przestrzenie:

  fizyczna, bloków dyskowych *length* bloków od bloku *address*

  logiczna, bajtów w pliku *length* * *AGBS* bajtów od bajtu *offset* * *AGBS*

- *AGBS* Aggregate Block Size

- pola:

  flag m.in. copy-on-write

  address adres ekstentu (w AGBS); (40-bit) z 2 pól addr1 i addr2

  offset pierwszy blok ekstentu (w AGBS); (40-bit) z 2 pól off1 i off2

  len długość ekstentu (w AGBS); (24-bit)

JFS – mapa alokacji bloków

- śledzenie zaalokowanych i wolnych bloków w agregacie
- rośnie i zmniejsza się dynamicznie ze wzrostem agregatu
- strony długości 4 k; 3 rodzaje:

  dmap mapy zajętych bloków (jedna strona dla 8 k bloków)

  kontrolne dmap i bmap pomagają szybko znaleźć duży ekstent lub wolne bloki

JFS – i-węzeł

- alokowane dynamicznie (po 32)
- i-węzeł tworzony, gdy używany (a nie, gdy alokowany)
- 1 licznik generacyjny i-węzłów
- części: POSIX, dla VFS, deskryptory alokacji ekstentów, dodatkowe

JFS – plik

- i-węzeł z korzeniem b+drzewa danych użytkownika, indeksowanym po offsecie ekstentów
- obsługa plików rzadkich
- dowiązania symboliczne próbują przechowywać ścieżkę w i-węźle

JFS – katalog

- księgowany plik meta-danych
- wpis – nazwa oraz i-węzeł
- małe (do 8 wpisów) w i-węźle katalogu
- dla dużych w b+drzewie (po nazwie pliku)
- kompresja sufiksowa nazw

JFS – podsumowanie

- agregat – obszar alokacji przestrzeni dyskowej (jeden na partycji)
- ekstent – zestaw ciągłych bloków agregatu
- zastosowanie – internetowe serwery plików


namesys

ReiserFS 3.6

ReiserFS 3.6

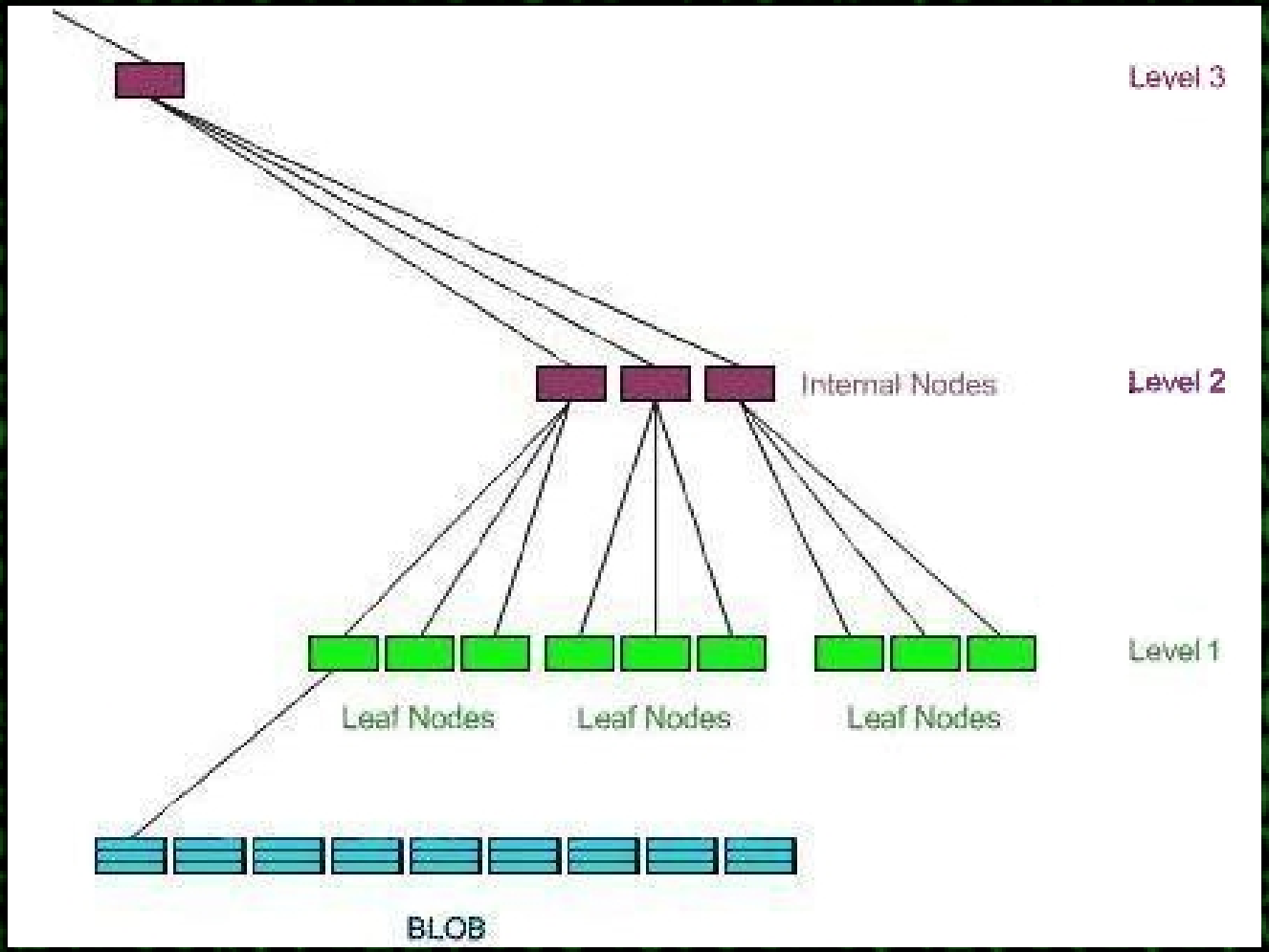
- Hans Reiser, Namesys
- napisany od zera
- dużo błędów
- stabilny od wiosny'2002

Reiser 3.6 – cechy

- b+drzewa
- małe pliki (do 3984 B dla bloku 4kB) w węzłach drzewa
- wydajność i oszczędność miejsca dla małych plików
- księgowanie meta-danych

Reiser 3.6 – b+drzewa

- w węzłach tylko klucze i wskaźniki (bez danych) – stopień rozgałęzienia
- dane w liściach lub poniżej liści (BLOB)
- do szybkiego znajdowania obiektów
- wysokość maksymalna 5



BLOB w ReiserFS 3.

Reiser 3.6 – balansowanie b+drzewa

Zasady – minimalizacja liczby:

- używanych węzłów
- zmienianych przez balansowanie
- jw, spoza pamięci podręcznej
- jeśli przenosimy, to jak najwięcej (dla liści)

Reiser 3.6 – węzły

wewnętrzny klucze i wskaźniki do poddrzew

niesformatowany BLOB

liście *pozycje* i ich nagłówki

Reiser 3.6 – pozycje (item)

pośrednie wskaźniki do węzłów niesformatowanych

bezpośrednie małe pliki lub ogony dużych

katalogowe wpisy katalogowe

Stat jak *i-węzły*

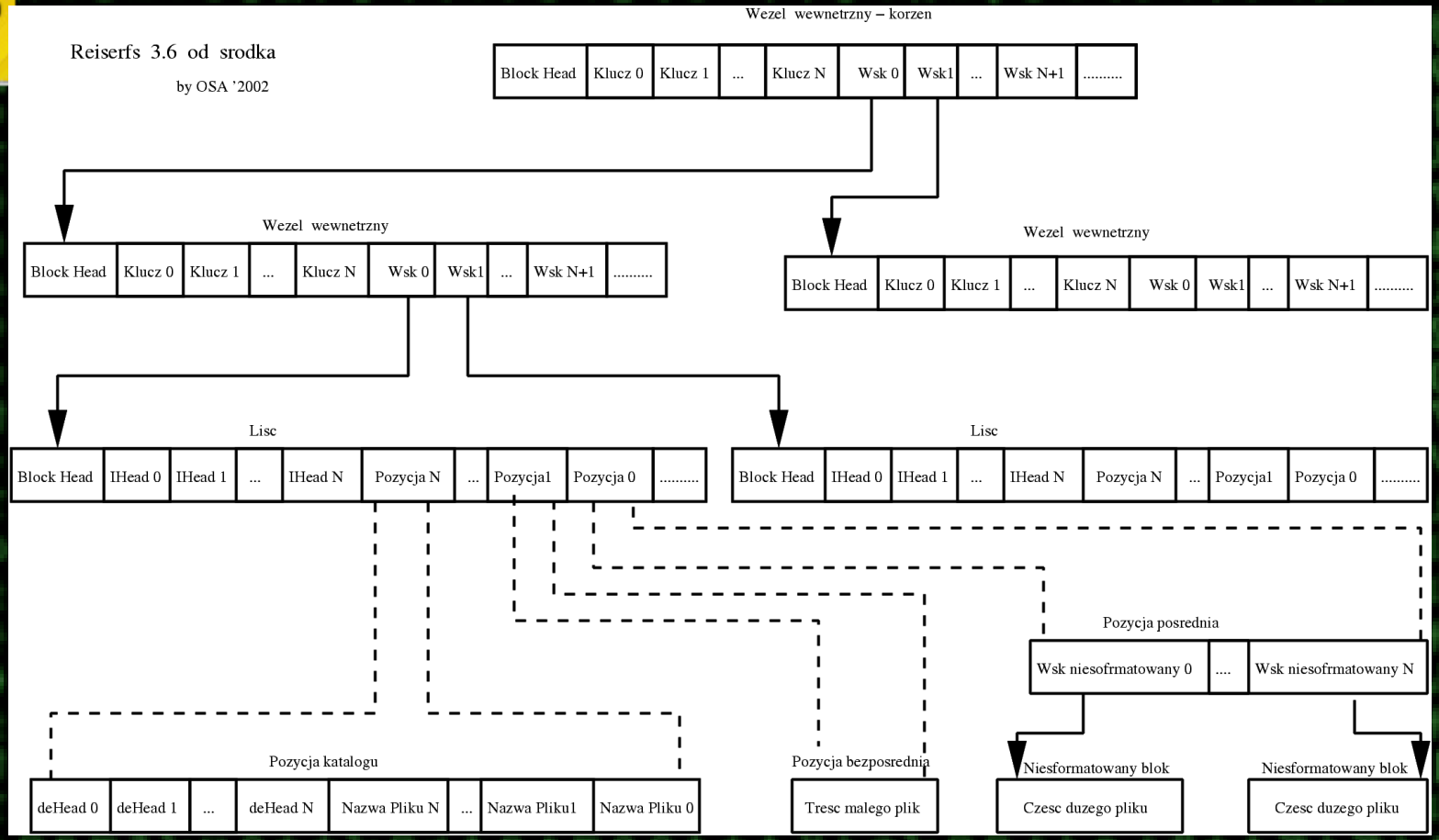
Reiser 3.6 – nagłówek pozycji

ih_key klucz

ih_free_space unia: przesunięcie lub liczba plików w katalogu

ih_item_len długość pozycji

ih_item_location przesunięcie do pozycji


Reiserfs 3.6 od srodka
by OSA '2002
Wezel wewnetrzny – korzen
Block Head
Klucz 0
Klucz 1
...
Klucz N
Wsk 0
Wsk1
...
Wsk N+1
.....
Wezel wewnetrzny
Wezel wewnetrzny
Lisc
Lisc
IHead 0
IHead 1
IHead N
Pozycja N
Pozycja1
Pozycja 0
Pozycja posrednia
Wsk niesofrmatowany 0
....
Wsk niesofrmatowany N
Pozycja katalogu
deHead 0
deHead 1
deHead N
Nazwa Pliku N
Nazwa Pliku1
Nazwa Pliku 0
Pozycja bezposrednia
Tresc malego plik
Niesformatowany blok
Czesc duzego pliku
Niesformatowany blok
Czesc duzego pliku

Reiser 3.6 – obiekt

- plik lub katalog
- identyfikator obiektu
- do identyfikacji *klucz*

Reiser 3.6 – klucz

indeksuje b+drzewo

k_dir_id id katalogu nadrzędnego

k_object_id id obiektu

k_offset przesunięcie od początku obiektu

k_uniqueness rodzaj pozycji

Reiser 3.6 – plik

- zbiór pozycji pośrednich
- mały plik lub ogon (reszta mniejsza od bloku) w pozycji bezpośredniej
- problemy z rosnącymi ogonami

Reiser 3.6 – katalog

pozycje katalogowe, te z wpisów

wpisy (nagłówek, nazwa) listowo

nagłówek (deHead):

- id obiektu i rodzica
- lokalizacja w pozycji
- przesunięcie do wpisu

Reiser 3.6 – Stat (i-węzeł)

POSIX (typ, uprawnienia, uid, gid, rozmiar, urządzenie, twarde linki, czasy)

sd_first_direct_byte przesunięcie do danych w pozycji bezpośredniej (o ile są)

Reiser 3.6

- b+drzewo
- dobra wydajność dla małych plików (pozycje bezpośrednie)
- gorsza dla dużych (BLOB) – poziom poniżej liści
- księgowanie meta-danych


REISER
4

Reiser 4

Reiser 4

- planowane wydanie – 31.12.2002
- wsparcie the Defence Advanced Research Projects Agency (DARPA)
- cel – stworzenie bezpiecznego odpornego na ataki systemu plików dla wojska i zastosowań cywilnych

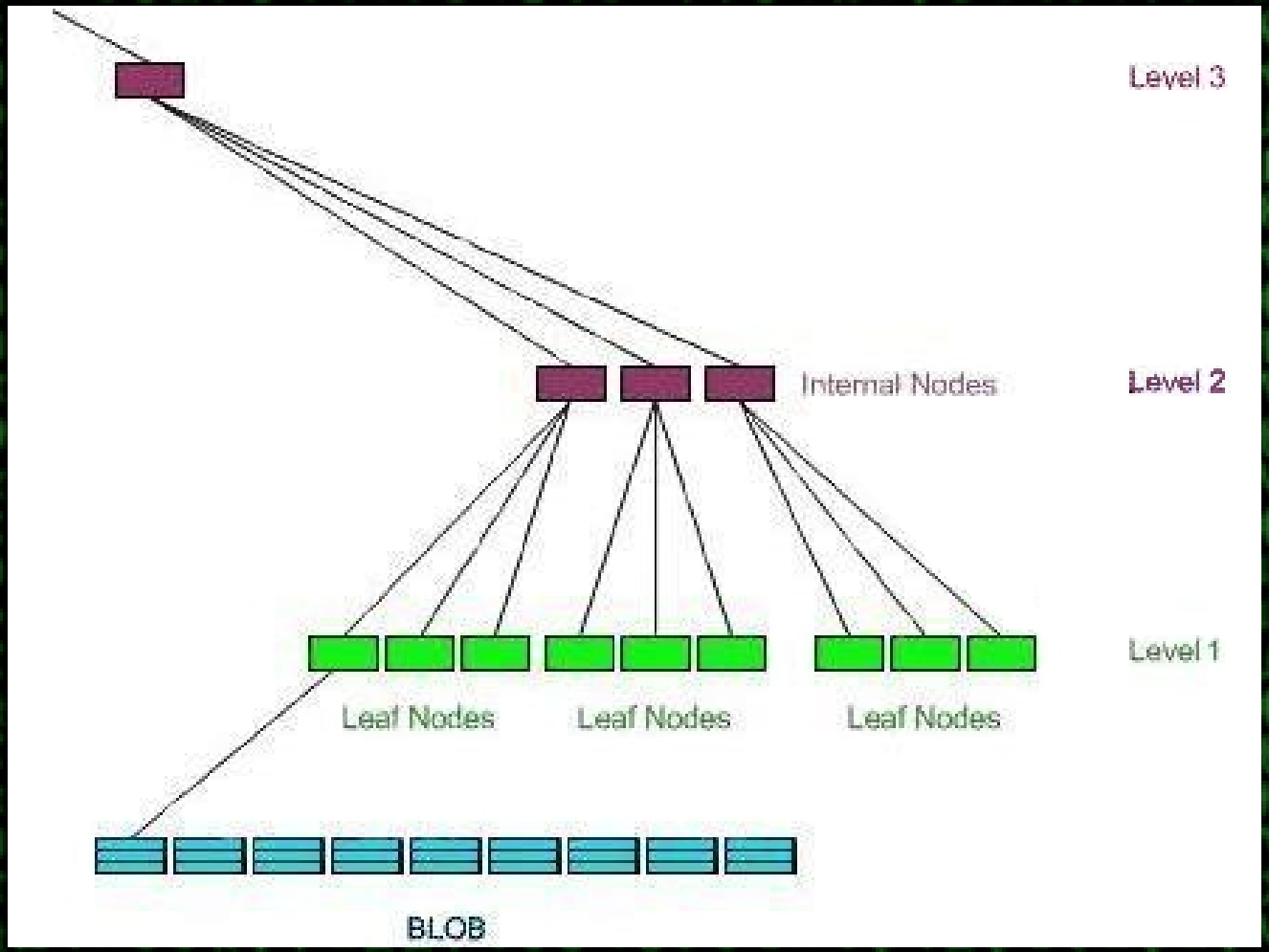
Reiser 4 – semantyka

- czystsza „semantyka” systemów plików
- czytelny podział na warstwy: semantyczna, przechowywania
- 8 standardowych wtyczek: plików, katalogów, haszowania, bezpieczeństwa, pozycji, przypisywania kluczy, szukania węzłów i pozycji

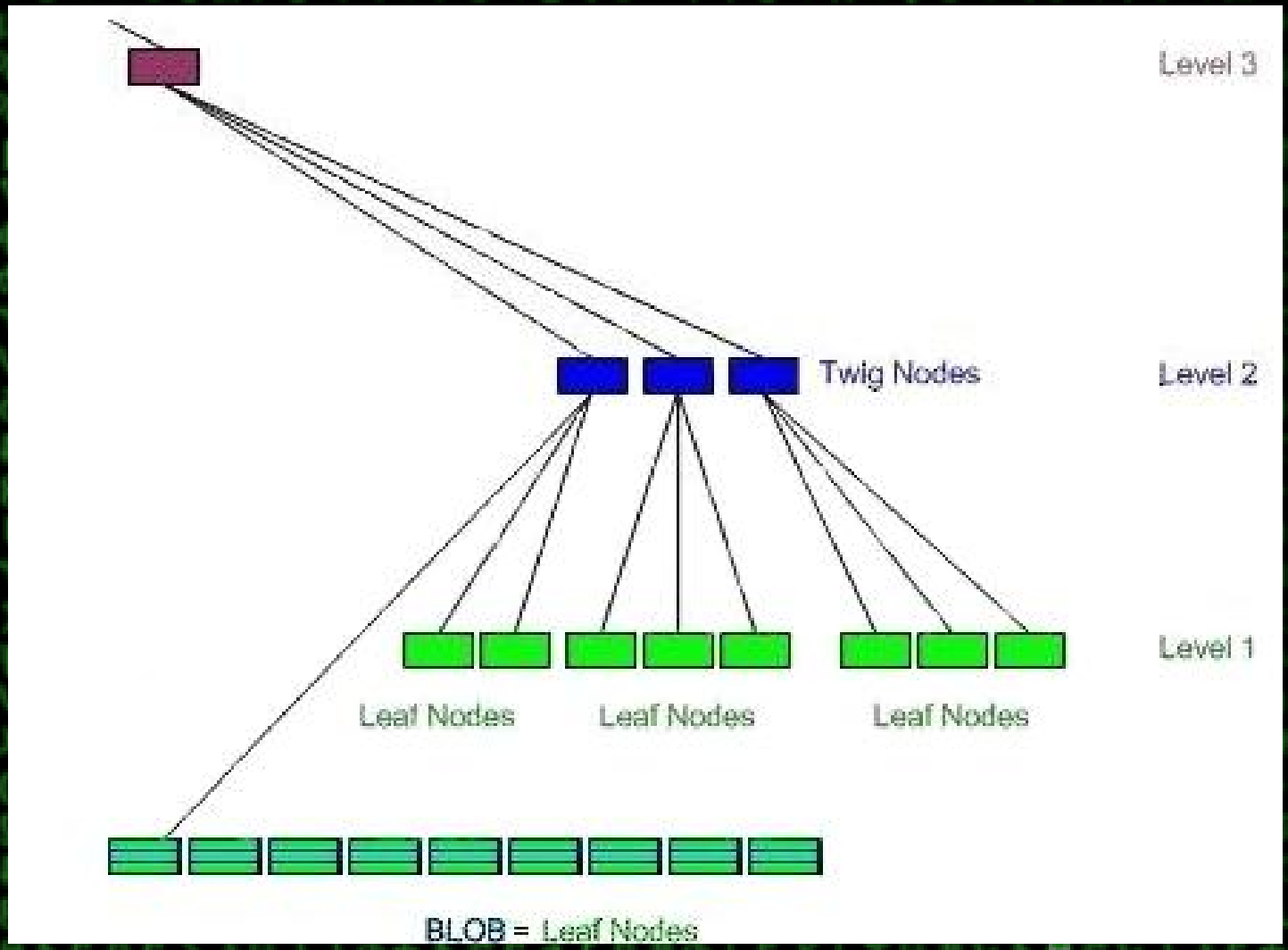
Reiser 4 – węzły

Dodano nowy typ węzła – *twig node*:

- na poziomie 2 (nad liśćmi)
- wskaźniki do liści z pozycjami (item)
- wskaźniki do BLOBów (niesformatowane dane)
- krótsza ścieżka do BLOBu – szybsza obsługa dużych plików



Wersja z ReiserFS 3.



Zmiana w ReiserFS 4.

Reiser 4 – pozycje

- w nagłówku (*itemHead*) nowe pole – *Item_Plugin_id*, określające *wtyczkę* obsługującą ten rodzaj pozycji
- zmiana kolejności w liściu – najpierw pozycje (item) potem ich nagłówki (itemHead)

BlHead	Item 0	..	Item N	Puste	IHead N	..	IHead 0

Reiser 4 – szyfrowanie

- Encryption on Commit
- szyfrowanie plików po zatwierdzeniu
- nie szyfrujemy od razu po *write*
- wzrost wydajności

Reiser 4 – wandering logs

- tradycyjnie podwójny zapis – do logu i na miejsce docelowe
- tutaj tylko raz
- zmieniana przynależność bloków z logu
- bloki nie są na stałe przypisane dla logu

Reiser 4 – klucze

Klucze mogą byc duplikowane – kompromis pomiędzy wielkością klucza (nie muszą być unikatowe) a szybkością (wyszukiwanie liniowe dla tego samego klucza).

Obiekty są wyszukiwane w B+drzewie, indeksowanym po wartości klucza (haszowanego, bo klucze nie muszą być unikalne).

Reiser 4 – podsumowanie

- szybkie księgowanie (wandering logs)
- skalowalność (wsparcie dla SMP)
- lepsza obsługa dużych plików

Wydajność

Wydajność

Testowanie wydajności operacji I/O. Odczyt i zapis sekwencyjny. Dla zapisu i odczytu losowego dane są zbliżone dla wszystkich systemów.

Maszyna testowa: 4 x 500 MHz, 2.4 GB RAM, kernel 2.5.40

Wydajność [MB/s]

wątków	JFS (w)	JFS (r)	Ext3 (w)	Ext3 (r)	Reiser (w)	Reiser (r)
1	11	202.5	10.5	199.6	9.5	201
2	12	278.5	10	283	5	283.6
4	12.2	310	7.5	310	5.4	308
6	12	310	1.2	310	5.05	308

Mówiłem o:

- ISO 9660, Joliet, Rocky Ridge, UDF
- FAT, FAT 32, NTFS, NTFS 5
- Minix, Ext, Ext2, Ext3
- EFS, xFS, JFS
- ReiserFS 3.6, ReiserFS 4

Można teraz bezpiecznie wyrzucić komputer.

Za naruszenie praw autorskich do grafiki i znaków handlowych autorzy przepraszają.

Można teraz bezpiecznie wyrzucić komputer.

Autorzy zapewniają, że podczas tworzenia tej prezentacji nie ucierpiał żaden blok dyskowy.

Można teraz bezpiecznie wyrzucić komputer.