# Windows NT File System

„Ausgewählte Betriebssysteme"
Institut Betriebssysteme
Fakultät Informatik

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

Hardware Basics

- Sector:
  - addressable block on storage medium
  - usually 512 bytes (x86 disks)
- Cluster:
  - addressable block of file system format
  - multiple of sector size
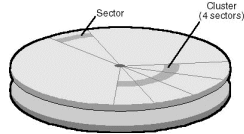
Win2K File System Formats

- CDFS: ISO 9660 (old CD-ROM FS)
- UDF (Universal Disk Format):
  - ISO 13346 compliant (for optical disk/DVD)
  - Replaces CDFS
  - Filenames can be 255 character long
  - Max path length is 1023 character
  - Filenames can be upper and lower case
- FAT12, FAT16, FAT32
  - FAT12 for anything smaller 16MB
  - FAT16 if explicitly specified (format command)
  - FAT32 anything bigger than 4GB
- NTFS

NTFS

- For volumes larger than 2GB default cluster size of 4KB is used
- Can (theoretically) address up to 16 exabytes using 64-bit cluster indices
- Limited to address using 32-bit indices → up to 128 TB (using 64KB clusters)

NTFS Cluster Sizes

| Volume Size	Default Cluster Size
512 MB or less	512 bytes
513 MB-1024 MB (1 GB)	1 KB
1025 MB-2048 MB (2 GB)	2 KB
Greater than 2048 MB	4 KB

- Default value can be overridden

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

FS Driver Architecture

- Local FSDs:
  - Manage volumes directly connected to computer
  - Responsible for registering with I/O manager
  - First sector on volume identifies volume, its format and location of metadata
- Remote FSDs:
  - Allow access to volumes connected to remote computers
 - Consists of two components (client & sever)

Local FSD

© Mark Russinovich & David Solomon (used with permission of authors)   9
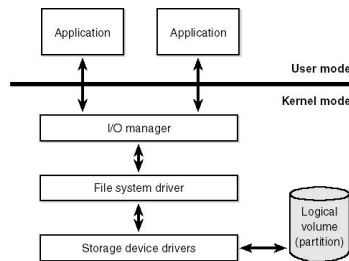
Local FSD (2)

- Device object is created for volume by FSD representing FS format
- I/O Manager connects FSD's device object to volume's device object
- Use cache manager to cache FS data (including metadata)
- Cooperate with memory manager:
  - Mapped file cannot be truncated or deleted
- Volume can be dismounted (for raw access)
  - First "normal" access remounts volume

Remote FSD

© Mark Russinovich & David Solomon (used with permission of authors)

11
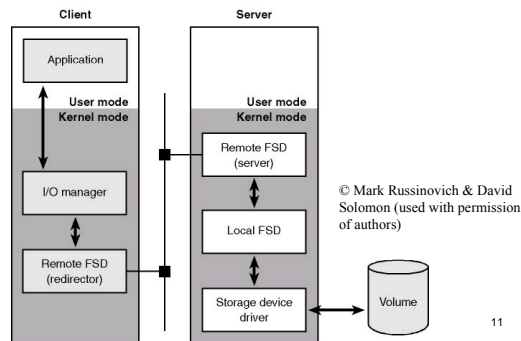
Remote FSD (2)

- Win2K uses Common Internet File System (CIFS) protocol (enhanced version of SMB)
- Client side FSD caches data (to synchronize *oplock protocol* is used)
- File and printer sharing built on it

Oplock Protocol

- „Opportunistic lock"
- Level I oplock granted for exclusive access (cached read and write)
- Level II oplock granted for shared access (cached read)
- Batch lock is Level I for multiple accesses with close operation in between (no additional oplock when reopening file)

Oplock Example
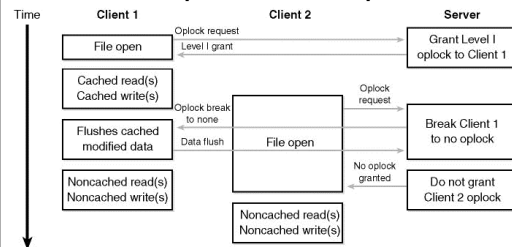
- If Client 1 only reads → both get Level II oplock

File System Operation
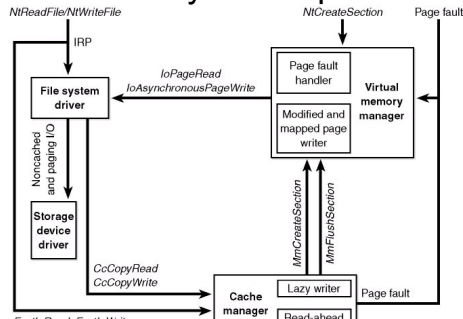
15

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

Advanced Features

- Multiple data streams
- Unicode-based names
- General indexing facility
- Dynamic bad-cluster remapping
- Hard links and junctions (soft-links)
- Link tracking
- Per-user volume quotas
- De-fragmentation
- Compression and sparse files (see later section)
- Change logging (see later section)
- Encryption (see later section)

Multiple Data Streams

- A file consists of streams
- One unnamed, default stream
- Stream name added to file name with colon (`stream.txt:longer`)
- Each stream has separate allocation size
- Each stream has separate file lock

Unicode Filenames

Examples

```
"TrailingDots..."
"SameNameDifferentCase"
"samenamedifferentcase"
"TrailingSpaces    "

"LongFileName"
"UnicodeName.ΦΔΠΛ"
"File.Name.With.Dots"
"File.Name2.With.Dots"
"Name With Embedded Spaces"
".BeginningDot"

"EIGHTCHR.123"
"CASEBLND.TYP"
```

POSIX subsystem

Win32 subsystem

MS-DOS Windows clients

© Mark Russinovich & David Solomon (used with permission of authors)   19

Hard Links and Junctions

- Hard links can be created with *CreateHardLink* and *ln*
  - Different names link to same file on disk
  - One file contains multiple $FILE_NAME attributes
- Junctions are soft links, based on reparse points
  - Reparse point has reparse tag, which allows to identify owner, and reparse data
  - Owner can alter pathname and reissue I/O or
  - Owner can remove reparse point, alter file and reissue I/O (archive and restore file automatically)

Link Tracking

- Links (e.g. shell shortcuts or OLE links) are another mechanism to "soft-link" files
- Link points to unique Object ID, which is stored in $OBJECT_ID attribute of file
- Target file can be allocated by querying for the Object ID
- Link tracking service implements the „link following"
- Mapping of Object IDs to filenames stored in file „$Extend:$O" (see slide 29/30)

Quotas

- Files are tagged with security ID (SID) of user
- Logical size of files counts against quota (not compressed size)
- Attempted violations and reached warning thresholds are logged in event log (and administrator can be notified)

Defragmentation

- NTFS does not automatically de-fragment disks
- NTFS provides de-fragmentation API
- Can be used to move file data, and obtain cluster information of file
- Win2K includes de-fragmentation tool

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
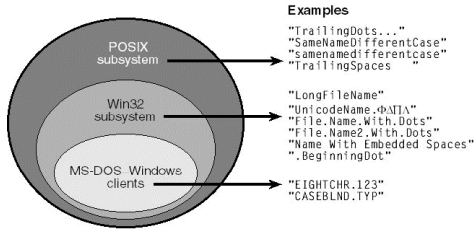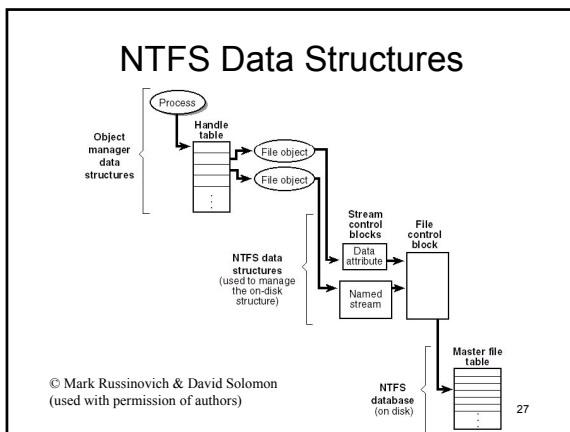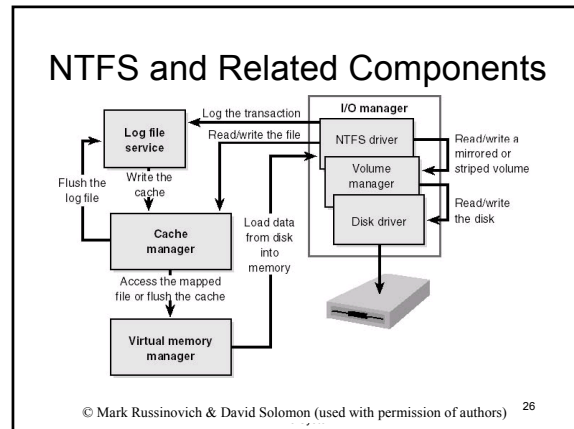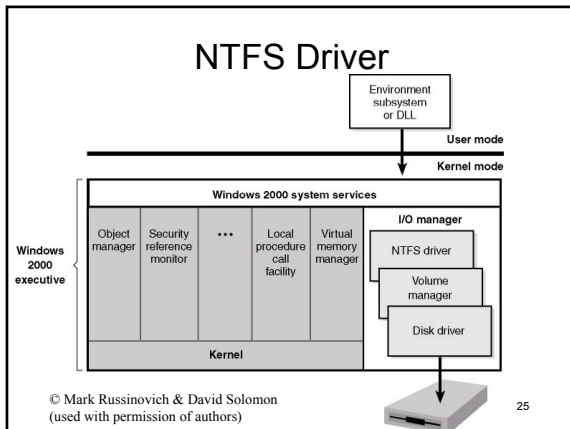 - Recovery Support
  - Encryption Support

NTFS Driver

Windows 2000 executive diagram showing Object manager, Security reference monitor, Local procedure call facility, Virtual memory manager, I/O manager (NTFS driver, Volume manager, Disk driver), Kernel, Windows 2000 system services, User mode / Kernel mode boundary, Environment subsystem or DLL.

25

NTFS and Related Components

Diagram with Log file service, Cache manager, Virtual memory manager, I/O manager (NTFS driver, Volume manager, Disk driver), labels: Log the transaction, Read/write the file, Flush the log file, Write the cache, Load data from disk into memory, Access the mapped file or flush the cache, Read/write a mirrored or striped volume, Read/write the disk.

26

NTFS Data Structures

Diagram: Object manager data structures (Process, Handle table, File object), NTFS data structures (used to manage the on-disk structure) with Stream control blocks (Data attribute, Named stream), File control block, Master file table, NTFS database (on disk).

27

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
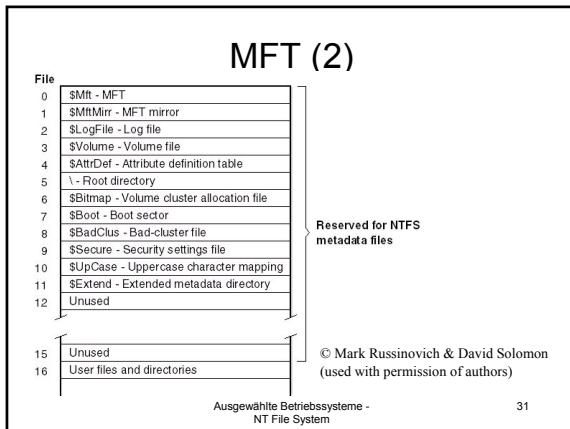 - Recovery Support
  - Encryption Support
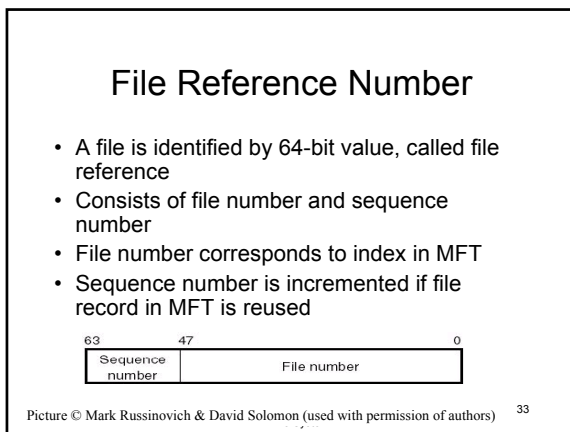
Ausgewählte Betriebssysteme - NT File System

28

NTFS On-Disk Structure

- Volumes: logical partitions (can span multiple partitions)
- Cluster: multiple of sector (always power of 2, e.g. 1,2,4,8 sectors)
- NTFS refers to physical locations on disk by logical cluster numbers (LCNs)
- NTFS refers to the data within a file by virtual cluster numbers (VCNs)

Ausgewählte Betriebssysteme - NT File System

29

Master File Table

- All data stored on volume is contained in files:
  - MFT, bootstrap data, allocation bitmap
  - Can relocate metadata
- MFT is array of file records
- File record has fixed size of 1KB
- MFT contains one record for each file on volume
- Metadata files have name starting with $
- On boot, volume is mounted by reading MFT and constructing internal data structures

Ausgewählte Betriebssysteme - NT File System

30

MFT (2)

File		
0	$Mft - MFT	
1	$MftMirr - MFT mirror	
2	$LogFile - Log file	
3	$Volume - Volume file	
4	$AttrDef - Attribute definition table	
5	\ - Root directory	
6	$Bitmap - Volume cluster allocation file	
7	$Boot - Boot sector	Reserved for NTFS
8	$BadClus - Bad-cluster file	metadata files
9	$Secure - Security settings file	
10	$UpCase - Uppercase character mapping	
11	$Extend - Extended metadata directory	
12	Unused	
15	Unused	
16	User files and directories	

© Mark Russinovich & David Solomon
(used with permission of authors)

Ausgewählte Betriebssysteme -
NT File System
31

MFT (3)

- $Mft and $MftMirr contain information about MFT (which blocks it occupies, ...)
- $LogFile contains recovery information
- NTFS starts searching for a file in Root directory
- $Bitmap shows free clusters
- $Secure volume wide security descriptor database
- $Boot – bootstrap code must be allocated at specific position on volume, but a file table entry is created, so inform can be read like file
- $Volume contains volume name, NTFS version, disk-corruption bit
- $Extend contains metadata, like quota, object ID file, ...

Ausgewählte Betriebssysteme -
NT File System
32

File Reference Number

- A file is identified by 64-bit value, called file reference
- Consists of file number and sequence number
- File number corresponds to index in MFT
- Sequence number is incremented if file record in MFT is reused

63	47	0
Sequence number	File number	

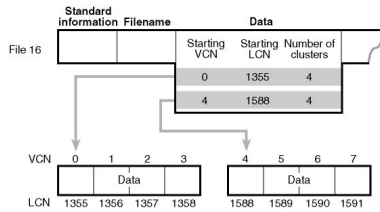Picture © Mark Russinovich & David Solomon (used with permission of authors)   33

File Record

- Strictly speaking: consists of attribute streams
- Each attribute:
  - Is identified by its attribute code
  - Has a value
  - Has an optional name (used to distinguish attributes of same type)
- E.g.:
  - $FILE_NAME attribute stores file name
  - $DATA attribute stores content of file

Ausgewählte Betriebssysteme -
NT File System
34

File Record (2)

- Small files fit into record
- Attributes with values stored in record are called resident attribute (standard information is always resident)
- Attribute header contains information if it is resident
- For big attributes clusters are allocated (so-called runs) and referenced from record
- These attributes are called non-resident

Ausgewählte Betriebssysteme -
NT File System
35

Resident/Non-Resident Attributes

© Mark Russinovich & David Solomon (used with permission of authors)   36
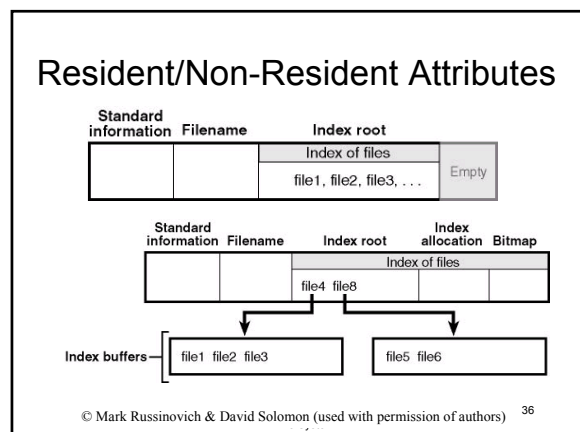
Non-Resident Attributes

- If multiple runs are needed to store an attribute, a mapping table of VCN is needed
- VCN (location in file), LCN (location on disk and size

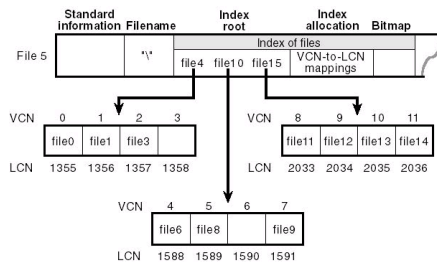© Mark Russinovich & David Solomon (used with permission of authors)   37

Directory Lookup

- For fast directory lookup an index tree is maintained
- Tree is B+ tree
- Each entry in tree contains information on file name, size, time stamp → directory information can be displayed without touching the file
- Requires this information to be updated in two places
- Each 4KB index buffer can contain 20-30 filenames

Directory Lookup (2)

© Mark Russinovich & David Solomon (used with permission of authors)   39
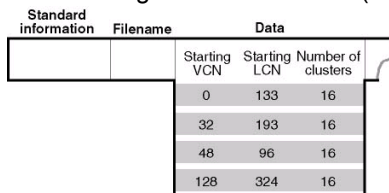
Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

Compression

- Compress sparse files (with holes), by not allocating runs for the holes (zeros)

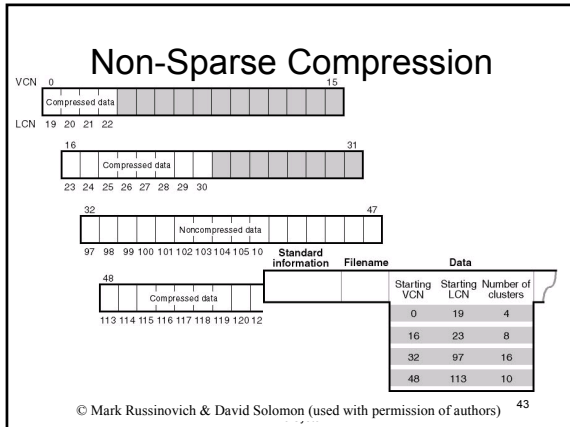| Starting VCN	Starting LCN	Number of clusters
0	133	16
32	193	16
48	96	16
128	324	16

© Mark Russinovich & David Solomon (used with permission of authors)   41

Compression (2)

- Non-Sparse Data compressed by combining 16 consecutive clusters to *compression units*
- Compress unit and if at least one cluster is saved, store compressed data.
- Distinguish compressed from uncompressed by length of run (number of clusters < 16)
- (runs with less than 16 clusters can be compressed too, but mapping becomes complicated and when stored again, stored in consecutive 16 cluster run)

Non-Sparse Compression

VCN 0   15
Compressed data

LCN 19 20 21 22

16   31
Compressed data

23 24 25 26 27 28 29 30

32   47
Noncompressed data

97 98 99 100 101 102 103 104 105 10

48
Compressed data

113 114 115 116 117 118 119 120 12

Standard information	Filename	Data		
		Starting VCN	Starting LCN	Number of clusters
		0	19	4
		16	23	8
		32	97	16
		48	113	10

© Mark Russinovich & David Solomon (used with permission of authors)   43

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

Recovery Support

- NTFS uses transaction-processing scheme to implement recoverability
- Recovery Procedures limited to file system data – user data never guaranteed to be fully updated after crash
- Sub-operation of transactions that alter file system data are logged before being carried through on disk
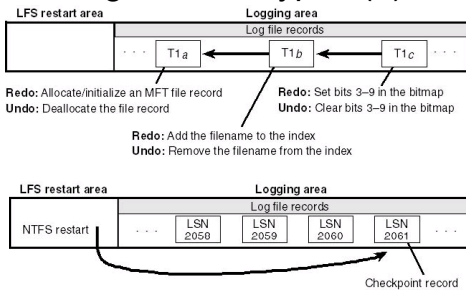- Logging done by Log File Service (LFS)

Log File Service (LFS)

- Log file divided into *restart area*:
  - 2 copies
  - Contains context information, such as location of start of recovery
- And *logging area*:
  - Treated as infinite (logs are written looping through area)
- Logical Sequence Numbers (LSN) used to identify records (64bit)
- Provides services to NTFS to open/close log file, read/write records

---

Transaction sequence

- Steps to ensure recoverability:
  1. NTFS calls LFS to record modification
  2. NTFS modifies volume
  3. Cache manager prompts LFS to flush Log to disk
  4. Cache manager flushes the volume changes
- Log file is also cached

Log Record Types

- Update Records
  - Redo information: how to reapply one sub-operation
  - Undo information: how to reverse one sub-operation
  - Contain physical state of data
- Checkpoint record
  - Written, when data has been updated in file as well

Log Record Types (2)

LFS restart area   Logging area

Log file records

· · · | T1$_a$ | ← | T1$_b$ | ← | T1$_c$ | · · ·

Redo: Allocate/initialize an MFT file record
Undo: Deallocate the file record

Redo: Set bits 3–9 in the bitmap
Undo: Clear bits 3–9 in the bitmap

Redo: Add the filename to the index
Undo: Remove the filename from the index

LFS restart area   Logging area

Log file records

NTFS restart · · · | LSN 2058 | LSN 2059 | LSN 2060 | LSN 2061 | · · ·

Checkpoint record

© Mark Russinovich & David Solomon (used with permission of authors)   49


Recovery

- Depends on two tables
  - Transaction table: keeps track of unfinished transactions
  - Dirty page table: contains modified pages, which contain file system structures
- Once every 5 seconds:
  - Transaction table and dirty page table written to log
  - Checkpoint written to log
- On recovery log-file is scanned three times
  - Analysis
  - Redo transactions
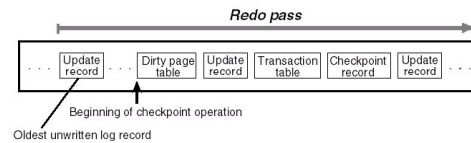 - Undo transactions

---

Analysis Pass

- Scan forward in log-file from beginning of last checkpoint operation to find update records to restore transaction and dirty page table
- Oldest update record, which's operation hasn't been carried out on disk, is determined (compared with dirty page table)
- If last checkpoint is older, Redo starts there

Analysis pass

· · · | Dirty page table | Update record | Transaction table | Checkpoint record | Update record | Update record | · · ·

Beginning of checkpoint operation

End of checkpoint operation
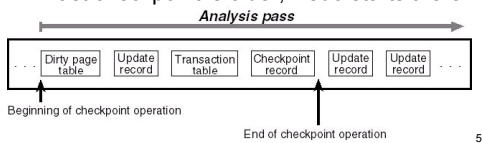
© Mark Russinovich & David Solomon (used with permission of authors)   51
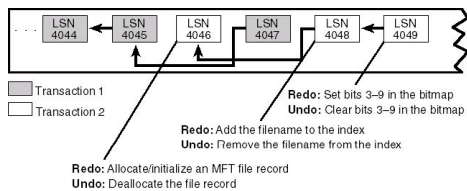

Redo Pass

- Looks for „page update" records
  - Which contain volume modification written before crash
  - But have not been flushed to disk
  - These updates are redone
  - After pass completed cache updates finished

Redo pass

· · · | Update record | Dirty page table | Update record | Transaction table | Checkpoint record | Update record | · · ·

Oldest unwritten log record

Beginning of checkpoint operation

© Mark Russinovich & David Solomon (used with permission of authors)   52

Undo Pass

- Roll back any transactions that weren't committed
- Undo operations are logged (in case of another power down)
- After Undo Pass is complete an „empty" LFS restart area is written

| LSN 4044 | LSN 4045 | LSN 4046 | LSN 4047 | LSN 4048 | LSN 4049 |

☐ Transaction 1
☐ Transaction 2

Redo: Set bits 3–9 in the bitmap
Undo: Clear bits 3–9 in the bitmap

Redo: Add the filename to the index
Undo: Remove the filename from the index

Redo: Allocate/initialize an MFT file record
Undo: Deallocate the file record

© Mark Russinovich & David Solomon (used with permission of authors)   53

Outline

- NTFS
  - File System Formats
  - File System Driver Architecture
  - Advanced Features
  - NTFS Driver
  - On-Disk Structure (MFT, ...)
  - Compression
  - Recovery Support
  - Encryption Support

Encryption

- User is assigned a private/public key pair (when he/she first encrypts a file)
- When file is encrypted:
  - content is encrypted using random number called file encryption key (FEK) and DESX (stronger version of DES)
  - FEK is encrypted using public key of user and RSA and attached to file
  - For each user FEK is encrypted respectively

Encryption (2)

- Private Key is stored in Registry (on disk "in a safe place") or smart card
- EFK is also encrypted for each Recovery Agent if recovery policy is defined
- Usage of private key:
  - CryptGetUserKey to tell crypto provider to subsequently use this user's key
  - CryptDecrypt is called to decrypt EFK