Unit 8: File System

8.1. Background: File System Concepts

Roadmap for Section 8.1

- General Concepts - Files and File Attributes
- File Operations
- Directories
- Linking File Names and File Content
- File Protection
- Network File Systems

Background: UNIX File Systems

File:

- Logical storage unit
- Unit of abstraction:
  - Physical properties of storage devices are abstracted away
- Nonvolatile memory
- UNIX data files are unstructured text files
- Programs, commands: executable files
  - Files are organized in a directory structure

File Attributes

- Files are named
  - For the convenience of the human user
  - File names are case-sensitive
- File attributes typically consist of:
  - Name
  - Type (different file types may be used by OS)
  - Location (pointer to device, location on device)
  - Size (current size, possibly maximum size)
  - Protection (read/write/execute (rwx) rights for users/groups/others)
  - Time and date for creation/modification/last access
  - Owner of a file

File Operations

A file is an abstract data type

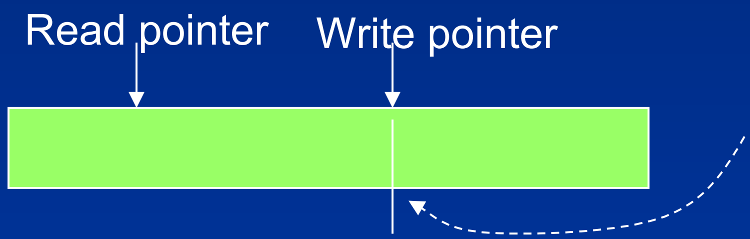
- Values + set of applicable operations
- A set of basic file operations has to be supported by the OS

- Creating a file
  - Allocate space in the file system, generate directory entry
- Writing a file
  - Name of the file and information to be written has to be specified
  - System has to maintain a write pointer for the file
- Reading a file
  - File name and a pointer to memory to receive data has to be specified
  - System maintains a read pointer for the file

File Operations (contd.)

- Repositioning within a file (file seek)
  - Directory is searched for the appropriate entry
  - Read/write pointers for the file are set to a given value
- Deleting a file
  - Release all file space; erase directory entry
  - Instead of deleting a file, UNIX allows to unlink() files – file is deleted by the OS if the last link to a file disappears
- Truncating a file
  - File attributes remain unchanged
  - File length is set to zero (or to some specified value)

Additional ops:
- Append
- Rename
- Set attributes
- ...



UNIX supports the „truncation-on-close“ flag to truncate files

Obtaining Access to a File

- Open() system call
  - Takes a file name, searches the directory, checks file protection
  - Copies directory entry into open-file table
  - Returns a pointer to the entry in open-file table for subsequent use
- Close() system call
  - Flushes cached file data back to the storage device
  - Deletes entry from open-file table
  - Frees system resources
- Operation in a multiuser environment
  - Per-process and global open-file tables
  - System maintains reference counts for opened files

Directories

- Record information about groups of files
- Management of files
  - Single-Level directory: most simple; all files in the same directory
  - Two-Level directory: separate directory for each user
  - Tree-Structured (hierarchical) directories: most common
- Operations on directories:
  - Search for a file
  - Create a file (directory entry)
  - Delete a file (directory entry)
  - List a directory
  - Rename a file
  - Traverse the file system (recursive)

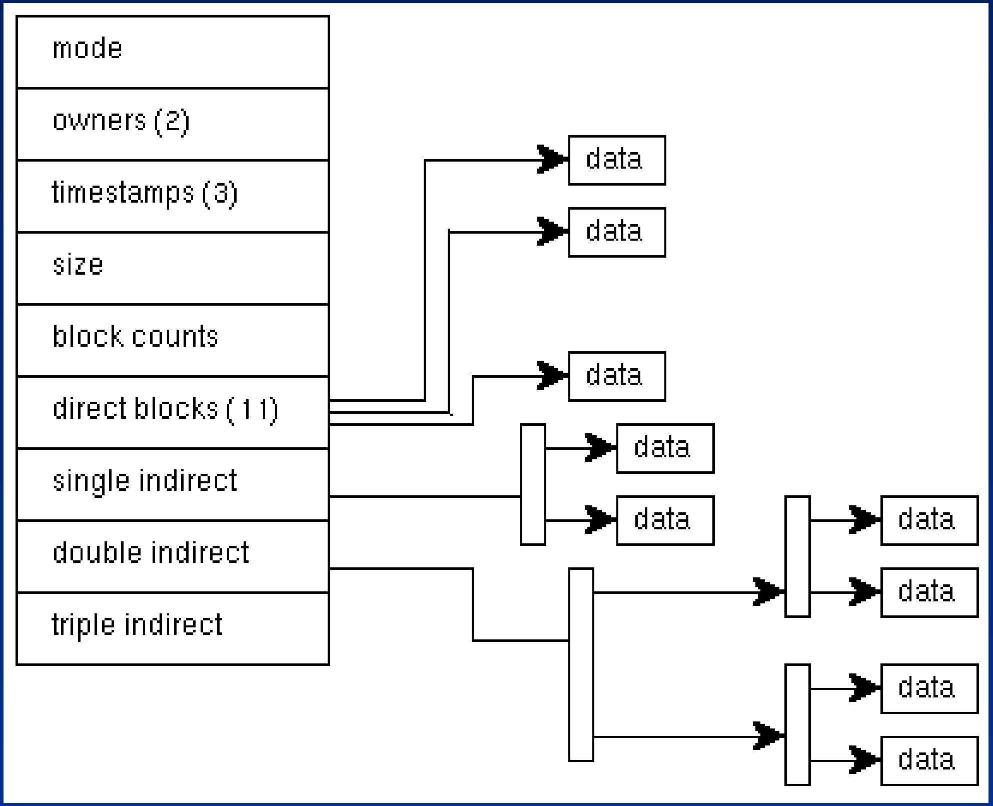
UNIX Directories

- Fully hierarchical, tree-structured
- Directories are represented as files
  - Problem: Truncation
- Processes have a current working directory
  - pwd command
- Each user has a home directory
  - cd; echo $HOME – commands to obtain info about the home dir.
- The file system has a single root directory
  - cd / - command changes working directory to root directory
- Special names identify neighbors in the directory tree
  - ./ - the current directory
  - ../ - the directory one level above the current directory

Linking Names and File Content

- UNIX separates file names and file content
  - file content may have multiple (different) names
  - ln command associates new name with existing file
- File content identified by:
  - (Device, File system on device, i-node)
  - i-node contains references to all blocks making up a file
  - a free-node list is maintained for each file system

Information contained in a UNIX i-node



File Protection

- Access rights can be independently defined for:
  - (u) user – Owner (creator) of a file
  - (g) group – Group
  - (o) other – all other users of the UNIX system
- Example:

```
luna test ( 48 )-% ls -lisa
total 2
421908   1 drwxr-xr-x    2   apolze    1024     Jan  7 15:06 .
116884   1 drwxr-xr-x   13   apolze    2048     Jan  7 15:06 ..
116992   0 -rw-------    1   apolze       0     Jan  7 15:05 Mail.txt
116991   0 -rw-rw-rw-    1   apolze       0     Jan  7 15:05 test.c
```

File Protection (contd.)

- Access rights for a file:
  - (r) Read access right; List right for directorisy
  - (w) Write access right; includes delete/append rights
  - (x) Execute access right; Traverse right for directories
- Binary representation:
  - (x): Bit 0 (+1)
  - (w): Bit 1 (+2)
  - (r): Bit 2 (+4)
- Rights can be combined
  - Read+Write access right: 6
  - Read+Execute access right: 3
  - Read-only: 2

Distribution – Network File Systems

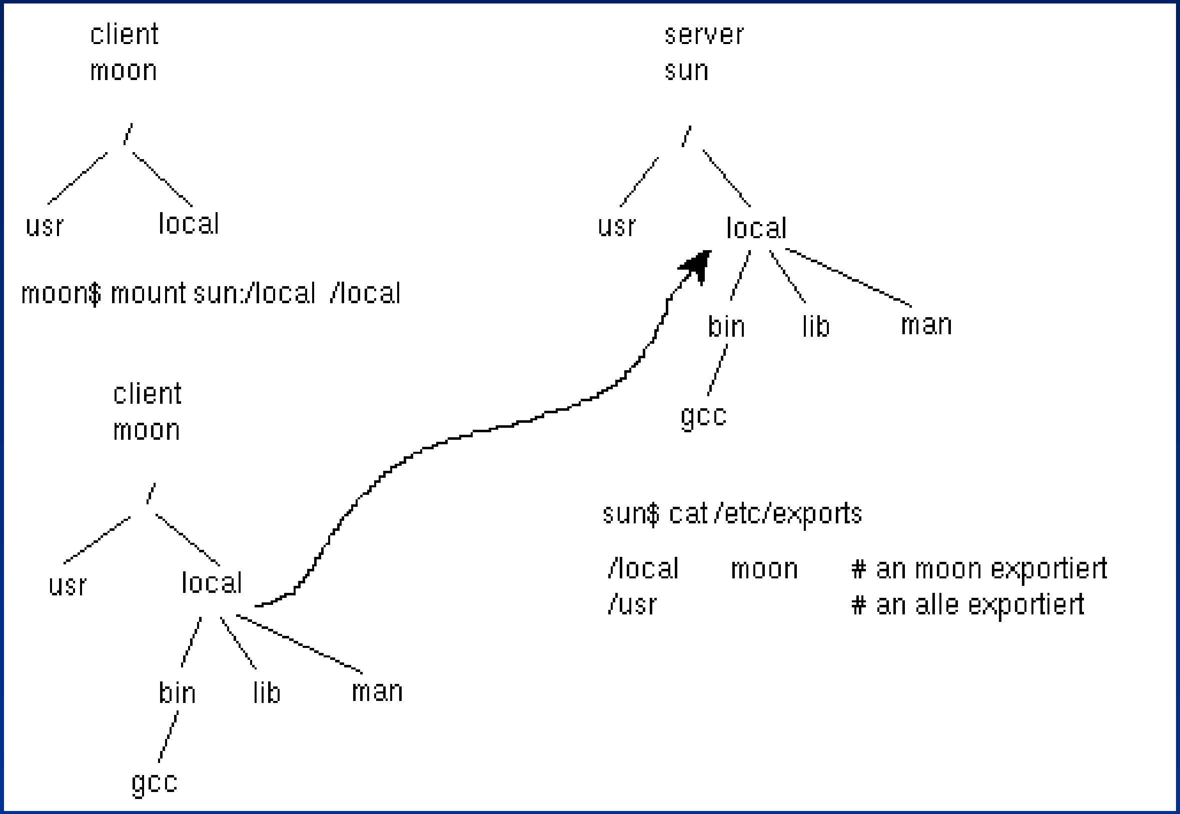
- Various approaches towards distributed file systems:
  - SUN Network File System (Standard)
  - UNIX United
  - Andrew File System
  - Sprite
  - Locus
- SUN NFS
  - Client/Server-System (based on remote procedure call (RPC))
  - File system operations are forwarded from client to server
  - Server executes actual file system operations, returns results
  - Client has access to remote resources
  - Stateless operation (Reliability !)

Operation of a Network File System

- A set of operations is implemented as RPC-callable functions:
  - Searching for a file in a directory
  - Reading a set of directory entries
  - Manipulating links and directories
  - Accessing file attributes
  - Reading and writing files
- Logical connection between client and server has to be established
  - mount protocol
- NFS works in heterogeneous environments
  - Machine-independent protocol for data representation (XDR)
- Stateless protocol
  - Network file system may tolerate client crashes (reboots)

Mounting a Remote File System

- Computer "sun" exports the "/local"-file system to computer "moon"



```
moon$ mount sun:/local  /local
```

```
sun$ cat /etc/exports

/local     moon     # an moon exportiert
/usr                # an alle exportiert
```

Exported to moon
Exported to all computers on the net

Windows – UNIX File System Interoperability

- Windows uses per default the Server Message Block (SMB) protocol to implement distributed file systems

SMB/CIFS (remote) resource sharing:

- Samba – www.samba.org

POSIX libraries / toolchains for Windows API:

- Cygwin – sources.redhat.com/cygwin/
- U/WIN – www. research.att.com/sw/tools/uwin/
- NuTCracker / MKS Toolkit – www.datafocus.com
- Interix, SFU – www.microsoft.com/windowsserversystem/sfu/

Windows Subsystem for Linux, in Windows 10 – see the WSL presentation

What's Samba all about?

- Free SMB and CIFS client/server for UNIX & other OSes, since 1992. Last version: 4.7.x (as of December 2017).
- Functionality:
  - a SMB server, to provide Windows NT and LAN Manager-style file and print services to SMB clients such as Windows, Warp Server, smbfs and others.
  - a NetBIOS (rfc1001/1002) nameserver, which amongst other things gives browsing support. Samba can be the master browser on your LAN if you wish.
  - a ftp-like SMB client so you can access PC resources (disks and printers) from UNIX, Netware and other operating systems

Further Reading

- Mark E. Russinovich, David A. Solomon, and Alex Ionescu, "*Windows Internals*", 6th Edition, Microsoft Press, 2012.
  - Chapter 12 – File Systems (from pp. 391)
    - Windows File System Formats (from pp. 392)

  Remark: this chapter will be in part 2 of 7th edition!

- Abraham Silberschatz, Peter B. Galvin, and Greg Gagne, "*Operating System Concepts*", 9th Edition, John Wiley & Sons, 2013.
  - Chapter 11 – File-System Interface
  - Chapter 12 – File-System Implementation
  - Chapter 17.9 – Distributed File Systems