HOWTO zu UNIX-Links

(C) 2007-2013 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: unix-links-HOWTO.txt,v 1.9 2025/02/23 20:14:55 tsbirn Exp $

Dieses Dokument beschreibt Eigenschaften und Einsatzzweck von Links unter UNIX.

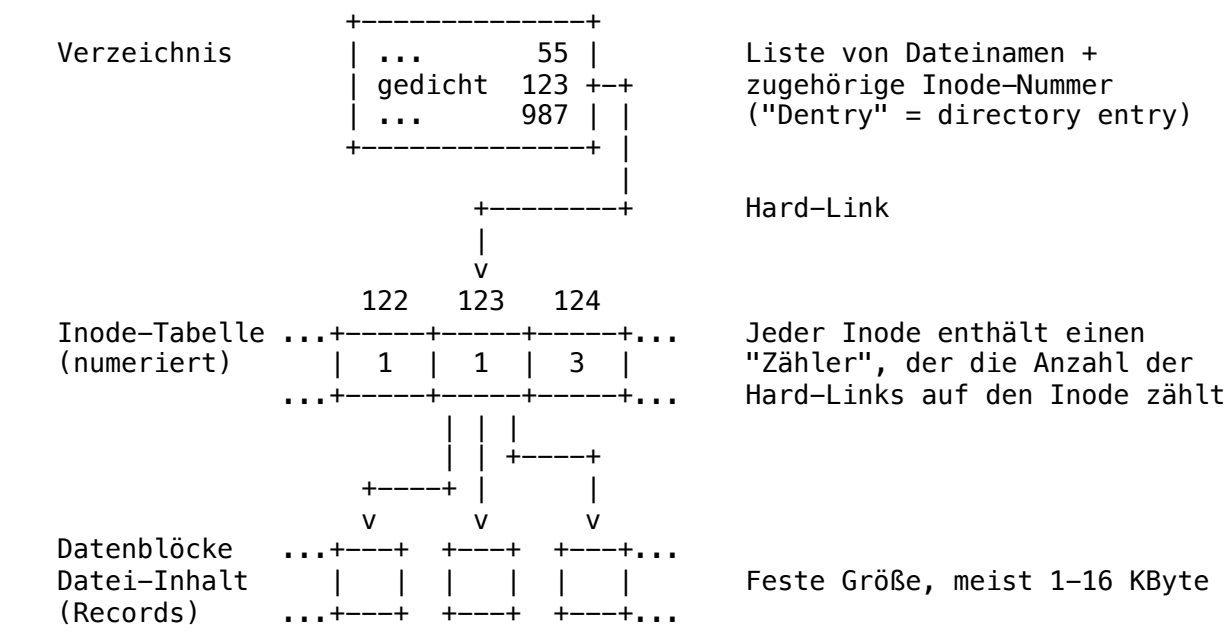
INHALTSVERZEICHNIS

1) Übersicht
2a) Hard-Link (Name + I-Node-Nummer)
2b) Inode ("Index-Knoten", mit Datei-Attributen)
2c) Datei-Inhalt
2d) Beispiel
3) Einsatzzweck von Links
4) Die beiden Link-Arten Hard-Link und Symbolischer Link
5) Vergleich mit Windows

---

1) Übersicht

Eine Datei setzt sich unter UNIX aus 3 Komponenten zusammen, die auch an 3 verschiedenen Stellen im UNIX-Dateisystem abgelegt werden:



2a) Hard-Link (Name + I-Node-Nummer)

In jedem Verzeichnis ist nur eine Liste sogenannter "Hard-Links" bestehend aus Dateiname + I-Node-Nummer ("Dentry" = directory entry) eingetragen. Diese verbinden den Dateinamen mit der zugehörigen Verwaltungsstruktur namens "I-Node" (Index Node). Es kann mehr als einen Hard-Link auf die gleiche Datei geben. In jedem Inode wird intern mitgezählt, wieviele Hard-Links aktuell auf ihn zeigen.

2b) Inode ("Index-Knoten", mit Datei-Attributen)

Der I-Node enthält (bis auf den Datei-Namen und den Datei-Inhalt) alle für eine Datei relevanten Informationen. Er zeigt weiterhin auf die Datenblöcke, in

denen sich entweder der Inhalt der Datei befindet oder die auf weitere
Datenblöcke zeigen (direkte und 1/2/3-fach indirekte Verweise).

```
  +-----------------------------------------------------+
  | Dateityp                                            |
  | Geräte-Nummer                                       |
  | Zugriffsrechte                                      |
  | Hard-Link Zähler                                    |
  | Besitzer                                            |
  | Besitzergruppe                                      |
  | Dateigröße                                          |
  | Datum:                                              |
  |   Access: Letzter lesender oder schreibender Zugriff |
  |   Modify: Letzte Änderung am Inhalt = Datenblöcke   |
  |   Change: Letzte Änderung am Inode  = Attribute     |
  |   Birth:  Erstellung der Datei                      |
  | Direkte Verweise auf Datenblöcke                    |
  | Indirekte Verweise auf Datenblöcke:                 |
  |   Einfach indirekt                                  |
  |   Zweifach indirekt                                 |
  |   Dreifach indirekt                                 |
  +-----------------------------------------------------+
```

2c) Datei-Inhalt

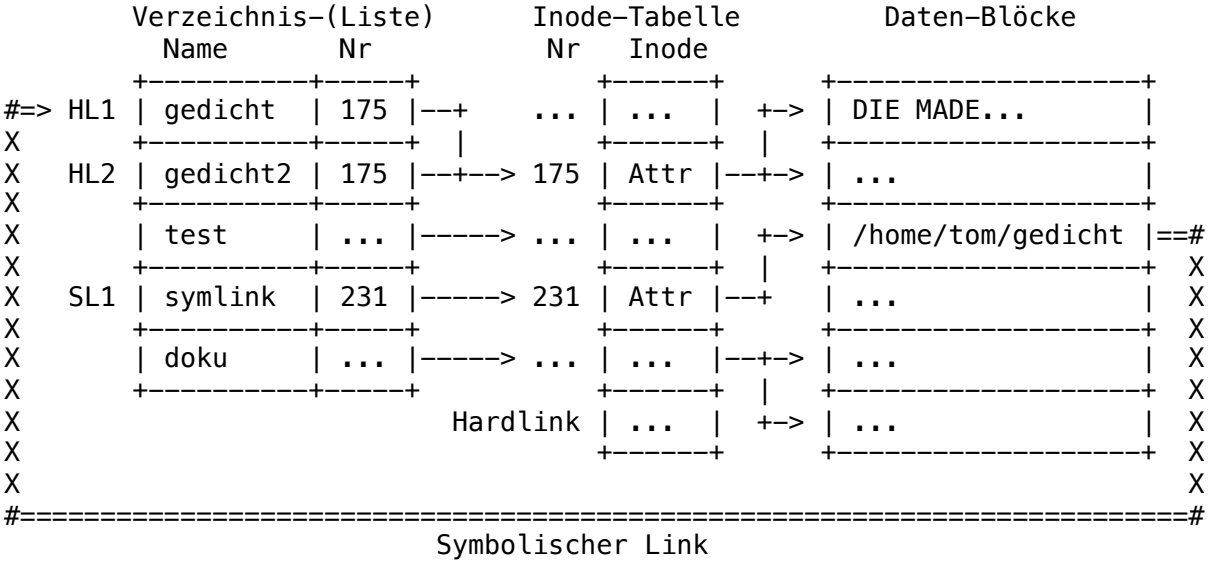
Der Datei-Inhalt ist in Datenblöcken abgespeichert, die im Dateisystem
verstreut liegen.

2d) Beispiel

* Aktuelles Verzeichnis ist "/home/tom/".

* Im Verzeichnis gibt es 2 Hard-Links namens "gedicht" und "gedicht2"
  auf die gleiche Datei.

* Weiterhin gibt es darin 1 Symbolischen Link namens "symlink",
  der auf "gedicht" zeigt.



3) Einsatzzweck von Links

* Immer den gleichen Namen für ein Gerät zur Verfügung stellen, egal welche
  konkrete Hardware dahinter steckt (Bsp: "/dev/mouse").

* Für den schnellen Zugriff in Benutzer-Verzeichnissen eine "Kopie" einer
  (tief) im Dateisystem gespeicherten Datei ablegen.

* Plattenplatz sparen, falls die gleiche Datei mehrfach im Dateisystem vorkommen
  soll (Verweise statt Kopien).

* Schreibfehler oder Namensänderungen wegen FHS (File Hierarchy Standard)
  abfangen (z.B. YaST = yast = zast, conf.modules = modules.conf).

* Konsistenz erhalten bei Versionierung, Updates, Konfigurationsdateien
  (Bsp: /usr/src/linux -> linux-2.2.5.SuSE).

* Eine von mehreren Versionen einer Datei oder eines Programms als Standard
  auswählen (z.B. /usr/bin/bash -> /usr/local/bin/bash-4.0)

* UNIX-Besonderheit: Das gleiche Programm/Skript kann unter verschiedenen Namen
  referenziert werden und verhält sich dann je nach Name, unter dem es
  aufgerufen wird, verschieden (Bsp: mv = cp = ln, mkfs.ext2 = mke2fs).

Beispiele für mehrere Namen für die gleiche Datei sind:

UNIX	Kommandos	Grund
SuSE-Linux	Yast2 yast2 zast yast1 yast	Schreibfehler abfangen
Solaris	cp mv ln	Programmierung erleichtern
Linux	gunzip gzip zcat	Programmierung erleichtern
Solaris	/usr/ucb/ps /usr/bin/ps	BSD + SYSV-Verhalten gleichz.

4) Link-Arten Hard-Link und Symbolischer Link

UNIX unterscheidet 2 Arten von Links, die aber beide für den gleichen Zweck eingesetzt werden, nämlich zusätzliche Namen für eine einzelne Datei zur Verfügung zu stellen:

  1) Hard-Link: Physikalischer Link (per I-Node)
  2) Soft-Link: Symbolischer Link (per Name)

1) Hard-Links gibt es --- wie oben beschrieben --- für jede Datei mindestens
   einen, beliebig viele weitere können folgendermaßen angelegt werden:

```
ln DATEI HARDLINK      # Analog "cp DATEI KOPIE"
```

  Ihre Verwendung ist in einigen Punkten eingeschränkt (z.B. nur innerhalb einer
  Partition einsetzbar).

2) Symbolische Links (auch Soft-Links) werden folgendermaßen angelegt:

```
ln -s DATEI SOFTLINK   # Analog "cp DATEI KOPIE"
```

   und verweisen per (Pfad)Name auf eine andere Datei. Der Pfadname kann relativ
   oder absolut sein.

   Man kann sich das so vorstellen, dass eine Datei als Wert den (Pfad)Namen
   einer anderen Datei enthält. Erst beim Zugriff auf einen Symbolischen Link wird
   festgestellt, ob seine Zieldatei wirklich existiert --- und falls ja --- diese
   benutzt. Sie sind mit HTML-Links oder Windows-Verknüpfungen zu vergleichen
   (Verweis per URL = Name auf eine andere HTML-Seite).

Die beiden Link-Typen können folgendermaßen charakterisiert werden:

	Hard Physical	Soft Symbol.

Referenz per	I-Node	Name
Anzahl Links auf Datei bekannt	Ja	Nein
Über Partitionen hinweg möglich	Nein	Ja
Ziel kann ein Verzeichnis sein	Nein	Ja
Rekursion möglich (Verz. enthält sich selbst)	Nein	Ja
Ziel muß existieren (kann nicht ins Leere zeigen)	Ja	Nein
Geschwindigkeit	schnell	langsam
Unter WINDOWS vorhanden (Verknüpfung)	Nein	Ja

5) Vergleich mit Windows

Symbolische Links gibt es unter Windows nur in Form der "Desktop-Icons" = Verweise auf Programme, Daten oder Verzeichnisse. Allerdings können diese wirklich nur auf dem Desktop liegen, innerhalb von Verzeichnissen sind sie nicht möglich. D.h. dieser Link-Typ ist unter Windows nur eingeschränkt verfügbar.

Hard-Links gibt es unter Windows erst mit dem Dateisystem NTFS. Zu ihrer Verwaltung gibt es aber keine grafische Möglichkeit im Windows Explorer, sondern sie sind nur per Kommandos nutzbar.