

HOWTO zu UNIX-Links

(C) 2007 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>

OSTC GmbH, <http://www.ostc.de>

\$Id: unix-links-HOWTO.txt,v 1.4 2008-12-22 15:39:06 tsbirn Exp \$

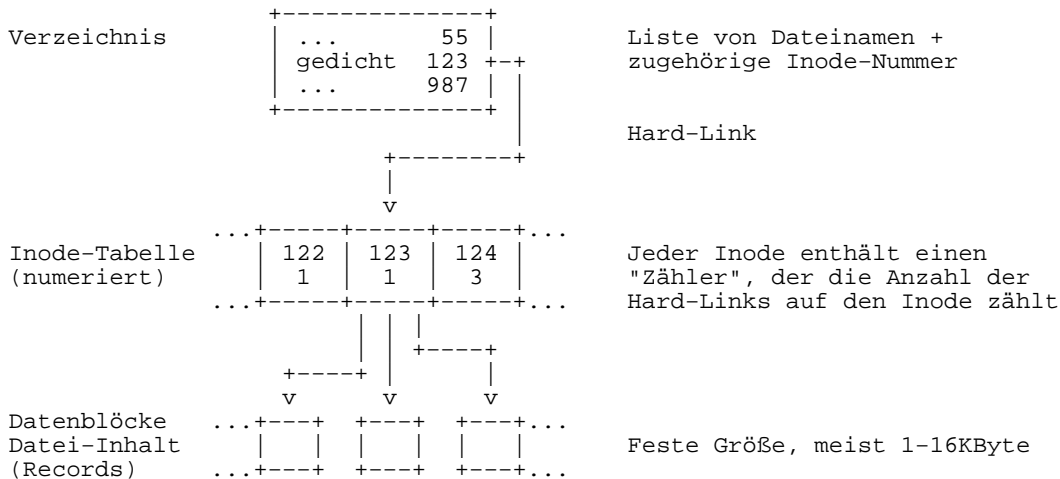
Dieses Dokument beschreibt Eigenschaften und Einsatzzweck von Links unter UNIX.

Inhaltsverzeichnis

- 1) Übersicht
- 2a) Hard-Link (Name + I-Node-Nummer)
- 2b) Inode ("Index-Knoten", mit Datei-Attributen)
- 2c) Datei-Inhalt
- 3) Zwecke von Links
- 4) Die beiden Link-Arten
- 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 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, jeder Inode zählt intern mit, wieviele Hard-Links auf ihn zeigen.

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

Der I-Node enthält (bis auf den Namen und den Datei-Inhalt) alle für eine Datei relevanten Informationen. Er zeigt weiterhin auf die Datenblöcke, in denen der Inhalt der Datei steht oder die auf weitere Datenblöcke zeigen (direkte und 1/2/3-fach indirekte Verweise).

- * Dateityp
- * Geräte-Nummer
- * Zugriffsrechte
- * Anzahl Hard-Links
- * Besitzer
- * Besitzergruppe
- * Dateigröße
- * Datum
- + Access: Letzter lesender/schreibender Zugriff
- + Modify: Letzte Änderung am Inhalt = Datenblöcke
- + Change: Letzte Änderung am Inode = Attribute
- * Direkte Verweise auf Datenblöcke
- * Indirekte Verweise auf Datenblöcke
 - + Einfach indirekt
 - + Zweifach indirekt
 - + Dreifach indirekt

2c) Datei-Inhalt

Datenblöcke mit Datei-Inhalt verstreut über das Dateisystem.

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.

Verzeichnis-(Liste)		Inode-Tabelle		Daten-Blöcke	
Name	Nr	Nr	Inode		
#=> HL1	gedicht	175	---+	+-> DIE MADE...
X					
X HL2	gedicht2	175	---+> 175	Attr	--+> ...
X					
X	test	510	---->	+-> /home/tom/gedicht ==#
X					
X SL1	symlink	231	----> 231	Attr	--+ ... X
X					
X	doku	012	---->	--+> ... X
X					
X			+-> ... X
X					
X					
X					
#=====					
Symbolischer Link					

3) Zwecke 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 die "Kopie" einer (tief) verschachtelten Datei dort ablegen.
- * Plattenplatz sparen, wenn die gleiche Datei mehrfach im Dateisystem vorkommen soll (Verweise statt Kopien).
- * Schreibfehler oder Namensänderungen wegen FHS (File Hierarchy Standard) abfangen (YaST = yast = zast, conf.modules = modules.conf).
- * Konsistenz erhalten bei Versionierung, Updates, Konfigurationsdateien (Bsp: /usr/src/linux -> linux-2.2.5.SuSE).
- * 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) Die beiden Link-Arten

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

- * Hard-Links (Physikalische Links)
- * Soft-Links (Symbolische Links)

Hard-Links gibt es wie --- oben beschrieben --- für jede Datei mindestens einen, sie sind aber in einigen Punkten eingeschränkt (z.B. nur innerhalb einer Partition einsetzbar). Soft-Links werden zusätzlich angelegt und sind mit HTML-Links oder Windows-Verknüpfungen zu vergleichen (Verweis per URL = Name auf eine andere HTML-Seite). Erst beim Zugriff über einen Soft-Link kann festgestellt werden, ob das Ziel existiert. Die beiden Link-Typen können folgendermaßen charakterisiert werden:

	Hard Physical	Soft Logical
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

Soft-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. auch dieser Link-Typ ist unter Windows nur sehr eingeschränkt verfügbar.

Hard-Links gibt es unter Windows erst ab dem Dateisystem NTFS, es gibt aber keine Schnittstelle (Kommandos oder grafische Oberflächen), um sie von Anwenderseite aus anzusprechen.