Betriebssysteme

Übungen 02

Prof. Dr. Rainer Werthebach

Studiengang Informatik

Hochschule Aalen - Technik und Wirtschaft

Benutzer und Gruppen

- Benutzer:
  - `/etc/passwd`
  - `Login:Pass:UID:GID:Kommentar:Home-Verzeichnis:Shell`
  - `kneuma:x:500:100:Karl Neumann:/home/neumann:/bin/bash`
  - Passwort heute nur noch selten in /etc/passwd aus Sicherheitsgründen
- Gruppen:
  - `/etc/group`
  - `Group-Name:Pass:GID:Userliste`
  - `tesauros:x:513:kneuma,werthe,dsebel,jpfeifer`
- Passwörter:
  - `/etc/shadow`
  - Datei kann nur Root lesen und bearbeiten (Ausnahme: eigenes Passwort ändern)
  - jpfeifer:Bet6INiizihHc:14341:0:99999:7:::

Rechte (Protection Bits)

- Anzeigen durch `ls -l`
- Beispiel: `user@rechner ~ $ ls -la`

```
drwxr-xr-x  3 ikarus users    4096 2011-03-24 09:04 uebung_01/
-rwxrwxr-x  1 ikarus users     824 2010-12-21 17:17 upsync
drwxr-xr-x  7 ikarus users    4096 2011-03-16 19:31 Videos/
drwxr-xr-x  2 ikarus users    4096 2011-02-15 13:11 .vim/
-rw-------  1 ikarus users   16096 2011-03-26 17:07 .viminfo
-rw-rw-r--  1 ikarus users    2361 2010-11-30 14:33 .vimrc
```

Rechte (Protection Bits)

```
-rwxrwxr-x  1 ikarus users     824 2010-12-21 17:17 upsync
```

- 1. Spalte: „Datei“-Typ und Zugriffsrechte
- 2. Spalte: Hardlink count
- 3. Spalte: Besitzer
- 4. Spalte: Gruppe
- 5. Spalte: Größe
- 6. Spalte: Datum und Uhrzeit (der letzten Veränderung)
- 7. Spalte: Name

Rechte (Protection Bits) - Bedeutung

```
drwxr-x--- [...]
```

- `rwx`:
  - `r` → Read
  - `w` → Write
  - `x` → Execute
- `tuuugggooo`:
  - `t` → Type: Directory (d), Link (l) ,Pipe (p), Socket (s), Device (c,b)
  - `u` → User: Benutzerrechte
  - `g` → Group: Gruppenrechte
  - `o` → Others: Alle anderen; Rest der Welt

Rechte (Protection Bits) – SUID, SGID, Sticky-Bit

- SUID:
  - Führt das Programm unter der UID (und mit den Rechten) des gesetzten Nutzers aus
  - Beispiel: `passwd`
    ```
    -rwsr-xr-x  1 root root 37100 2011-02-14 23:12 passwd
    ```
- SGID:
  - Analog zu SUID nur das dabei die Gruppe berücksichtigt wird
- Sticky-Bit:
  - Bei Programmen: Programm bleibt permanent im Hauptspeicher (nicht mehr gängig)
  - Bei Verzeichnissen: Jeder Nutzer darf Dateien anlegen, aber nur der Besitzer oder root dürfen Dateien ändern

Rechte (Protection Bits) ändern

- `chmod` *`Rechte Datei/Verz.`* → Rechte ändern
  - Rechte in der Form:
    - Möglichkeit 1: Beispiele: a-wx, u+x, g+s, o-rwx
    - Möglichkeit 2: Beispiele: 750, 644, 775
  - Zu Möglichkeit 1:
    - u → User, g → Group, o → Others, a → All
  - Zu Möglichkeit 2:
    - Oktal Notation über binären Wert
    - Immer 3 Bit sind eine Stelle
    - `uuugggooo` → `UGO`
    - Beispiel: `rwxr-x---` → 111 101 000 → 750
  - Allgemeine Beispiele:
    - `$ chmod a+x Desktop/script1`
    - `$ chmod 777 Videos/clip1.avi`

Rechte (Protection Bits), Benutzer, Gruppen

- `umask` → umask setzen / auslesen
  - Standardrechte für neu angelegte Dateien / Verzeichnisse
  - Wird von 666 für Dateien und 777 für Verzeichnisse abgezogen
  - Beispiel: umask = 026, eine neue Datei wird angelegt
    666 → Standardwert
    -026 → umask
    640 → 110 100 000 → `rw- r-- ---`

- `chown` *`Benutzer Datei/Verz.`* → Benutzer / Besitzer ändern
  - 3. Spalte von `ls -l` → Besitzer
- `chgrp` *`Gruppe Datei/Verz.`* → Gruppe ändern
  - 4. Spalte von `ls -l` → Gruppe

Links

- Hardlinks:
  - `ln` *`Ziel`* `[Verknüpfungsname]` → Erstellen eines Hardlinks
  - Hardlinks verweisen auf die Inode
  - Inode: Verzeichniseintrag im Dateisystem mit Metadaten (z.B. Zugriffsrechte Dateityp, Größe, Hardlink count, …)
  - 2. Spalte von `ls -l` → Hardlink count
  - Wenn Hardlink count = 0 dann wird die Datei gelöscht bzw. zum überschreiben freigegeben.
- Symlinks (Symbolic Links):
  - `ln -s` *`Ziel`* `[Verknüpfungsname]` → Ein symbolischen Link erstellen
  - Typische Verknüpfung (Vergleichbar mit Windows-Verknüpfung)
  - In der `ls` Ausgabe an `Verknüfungsname -> Ziel` erkennbar

Ein- /Ausgabeumlenkung

- Standard-Eingabe (`stdin`) → Tastatur
- Standard-Ausgabe (`stdout`) → Bildschirm
- Standard-Fehlerausgabe (`stderr`) → Bildschirm

- Ausgabeumlenkung:
  - `Befehl > Ausgabedatei`
  - `Befehl >> Ausgabedatei` → An Ausgabedatei anhängen
- Eingabeumlenkung:
  - `Befehl < Eingabedatei`
- Fehlerumlenkung:
  - `Befehl 2> Fehlerdatei`
  - Beispiel:

```
$ cp * /media/usbstick 2> Fehlerprotokoll
```

Pipes

- Pipe-Symbol: `|`
- Leitet Ausgabe von Programm1 als Eingabe an Programm2 weiter
- Abstraktes Beispiel: `prog1 | prog2`
- Allgemeine Beispiele:
  - `$ head -n 3 1.txt | wc -w`
  - `$ cat punkte.txt | sort -r | head -n 10 >top_ten.txt`

Weitere Programme

- `cut [Datei]` → Teile einer Zeile ausgeben
  - Wichtige Parameter:
    `-d` → Delemiter (Trenner) mit dem z.B. Spalten getrennt werden sollen
    `-f` → Felder (Spalten) die ausgegeben werden sollen
  - Beispiel: `$ cut -d ':' -f 7 /etc/passwd`
- `tr "Menge1" "Menge2"` → Zeichen Übersetzen
  - Beispiel: `$ cat datei.txt | tr "ax" "by"`
- `diff Datei1 Datei2` → Unterschiede zwischen zwei Dateien

Weitere Programme

- `find [Verzeichniss] [Muster]`
  → Verzeichnis rekursiv nach Dateien durchsuchen
  - Sehr mächtig → sehr viele mögl. Parameter → `man find` (1100+ Zeilen)
  - Beispiele:

```
$ find Desktop -name "test.txt"
$ find . -cmin 10 -iname "*.doc"
$ find /usr/bin -user root -perm -4000
$ find .. -name "a*" -exec cp {} ~/Desktop/ \;
```

- `grep` *`Muster`* `[Datei]` → Dateien nach Mustern durchsuchen
  - Sehr mächtig → Parameter, reguläre Ausdrücke → `man grep` (400+ Zeilen)
  - Beispiele:

```
$ grep "abc" test.txt
$ ls -la /usr/bin | grep passwd
$ grep -v "[xyz]" datei.txt
```

Editoren

- `nano [Datei]` → Einfacher Editor
  - Gut für den einfach Einstieg
  - `<Strg>+<o>` → Speichern
  - `<Strg>+<x>` → Beenden
  - `<Strg>+<w>` → Suchen

- `vi, vim [Datei]` → Umfangreicher Editor
  - vi für „visual“
  - vim → vi iMproved
  - 3 Modi:
    Command Mode → Befehle über Tastenkürzel
    Insert Mode → Normale Textbearbeitung
    Execution Mode → Ausführen von Befehlen / Programmen

vi, vim

- Wechseln zum Insert Mode → `i`
- Zurück zum Command Mode → `<ESC>`
- Speichern → `:w`
- Beenden → `:q`
- Beenden ohne speichern → `:q!`
- Zeichen löschen → `x`
- 5 Zeilen löschen → `5dd`
- Einfügen → `p`
- Vorwärts suchen → `/muster`
- Rückwärts suchen → `?muster`
- Suchen und ersetzen → `%s/alt/neu/g`
- ...

vi, vim

- Navigation

Den Cursor mit Cursortasten „rumschubsen“ machen nur *Vi*-Novizen.
Vi-Experten „springen“ in großen Einheiten bzw. durch Textsuche.