# 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:
    - Oktale Notation des binären Wertes
    - Immer 3 Bit sind eine Oktalzahl
    - `uuugggooo` → `UGO`
    - Beispiel: `rwxr-x---` → 111 101 000 → 750
  - allgemeine Beispiele:
    - `$ chmod a+x Desktop/script1`
    - `$ chmod 777 Videos/clip1.avi`

# man chmod

- chmod changes the file mode bits of each given file according to mode, which can be

  **either a symbolic representation** of changes to make, **or an octal number** representing the bit pattern for the new mode bits.

- The format of a symbolic mode is **[ugoa...][[-+=][perms...]...]**, where perms is either zero or more letters from the set **rwxXst**, or a single letter from the set ugo. Multiple symbolic modes can be given, separated by commas.

- A combination of the letters ugoa controls which users' access to the file will be changed: the user who owns it (**u**), other users in the file's group (**g**), other users not in the file's group (**o**), or all users (**a**). If none of these are given, the effect is as if (**a**) were given, but bits that are set in the umask are not affected.

# man chmod

- The operator **+** causes the selected file mode bits to be added to the existing file mode bits of each file; **-** causes them to be removed; and **=** causes them to be added and causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.
- The letters **rwxXst** select file mode bits for the affected users: read (**r**), write (**w**), execute (or search for directories) (**x**), execute/search only if the file is a directory or already has execute permission for some user (**X**), set user or group ID on execution (**s**), restricted deletion flag or sticky bit (**t**). Instead of one or more of these letters, you can specify exactly one of the letters **ugo**: the permissions granted to the user who owns the file (**u**), the permissions granted to other users who are members of the file's group (**g**), and the permissions granted to users that are in neither of the two preceding categories (**o**).

# man chmod

- A **numeric mode** is **from one to four octal digits** (0-7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. **The first digit selects the set user ID (4) and set group ID (2) and restricted deletion or sticky (1) attributes.** The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.
- chmod never changes the permissions of symbolic links; the chmod system call cannot change their permissions. This is not a problem since the permissions of symbolic links are never used. However, for each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

# man chmod

- SETUID AND SETGID BITS
- chmod clears the set-group-ID bit of a regular file if the file's group ID does not match the user's effective group ID or one of the user's supplementary group IDs, unless the user has appropriate privileges. Additional restrictions may cause the set-user-ID and set-group-ID bits of MODE or RFILE to be ignored. This behavior depends on the policy and functionality of the underlying chmod system call. When in doubt, check the underlying system behavior.
- For directories chmod preserves set-user-ID and set-group-ID bits unless you explicitly specify otherwise. You can set or clear the bits with symbolic modes like u+s and g-s. To clear these bits for directories with a numeric mode requires an additional leading zero, or leading = like 00755 , or =755

# man chmod

- RESTRICTED DELETION FLAG OR STICKY BIT
- The restricted deletion flag or sticky bit is a single bit, whose interpretation depends on the file type. For directories, it prevents unprivileged users from removing or renaming a file in the directory unless they own the file or the directory; this is called the restricted deletion flag for the directory, and is commonly found on world-writable directories like /tmp. For regular files on some older systems, the bit saves the program's text image on the swap device so it will load more quickly when run; this is called the sticky bit.

# 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 i-node
  - i-node: Eintrag im Dateisystem mit Metadaten für Dateien und Verzeichnisse (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, Softlinks):
  - `ln -s` *`Ziel`* `[Verknüpfungsname]`→ Einen 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 einfachen schnellen 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 in den Insert Mode → `i`
- zurück zum Command Mode → `<ESC>`
- speichern → `:w` `eventuell erst <ESC>`
- 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.