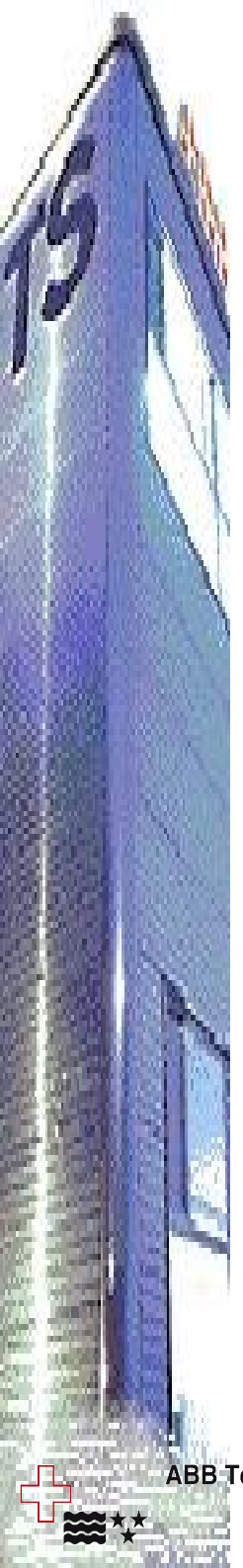
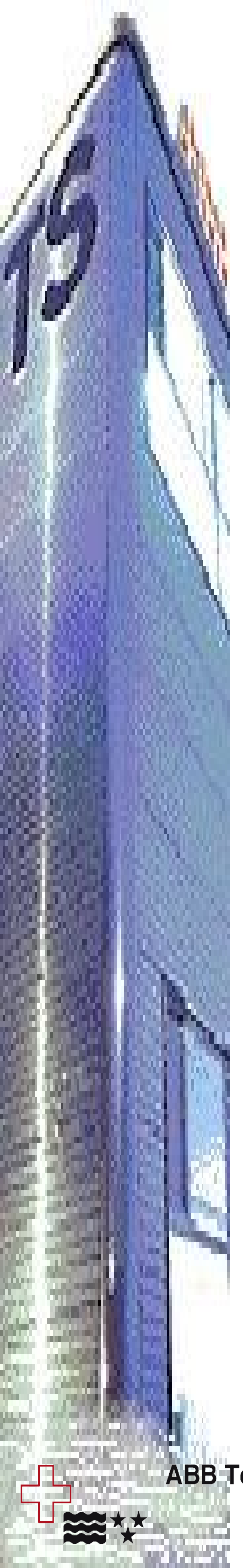


Zugriffsrechte

Ziele

- Das Linux-Rechtekonzept vertieft beherrschen
- Zugriffsrechte für Dateien und Verzeichnisse vergeben können
- Die Begriffe *umask*, SUID, SGID und *sticky bit* kennen
- Die Dateiattribute der ext-Dateisysteme kennen

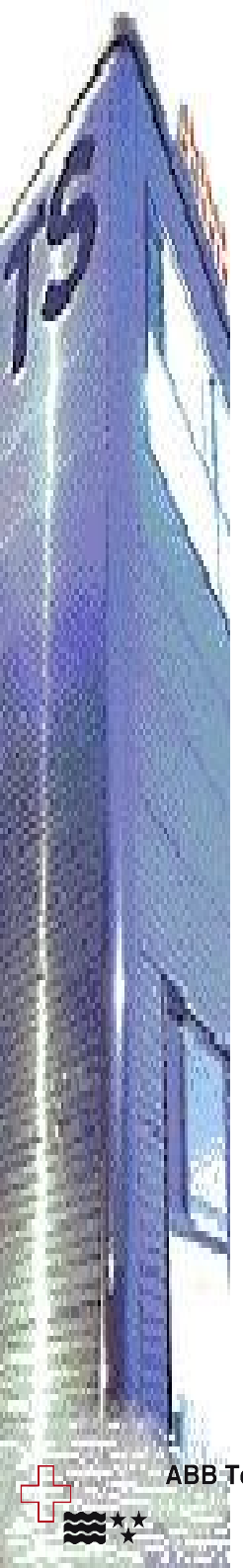


Das Linux-Rechtekonzept

- Standard UNIX Rechte Konzept
  - für Prozesse, Dateien und Verzeichnisse
- jede Datei und jedes Verzeichnis ist
  - genau einem Benutzer (Besitzer)

    und
  - genau einer Gruppe

    zugeordnet



Das Linux-Rechtekonzept

(Fortsetzung)

- jede Datei hat getrennte Rechte für
  - Besitzer
  - Gruppe
  - alle anderen Benutzer
- separate Rechte für jede dieser Mengen
  - Schreibrechte
  - Leserechte
  - Ausführungsrechte

Das Linux-Rechtekonzept

(Fortsetzung)

- die Gruppe und alle andern Benutzer haben nur dann Rechte:
  - wenn der Eigentümer diese Rechte einräumt
- Die Einstellung der Rechte heisst
  - Zugriffsmodus

Das Linux-Rechtekonzept

(Fortsetzung)

- Linux-Dateisysteme z.B.
  - ext2, ext3 und XFS

  erlauben Zugriffskontrolllisten
  - engl. *access control lists* »ACL«
- Diese erlauben für mehrere Benutzer und Gruppen weitere Rechte zu vergeben

Das Linux-Rechtekonzept

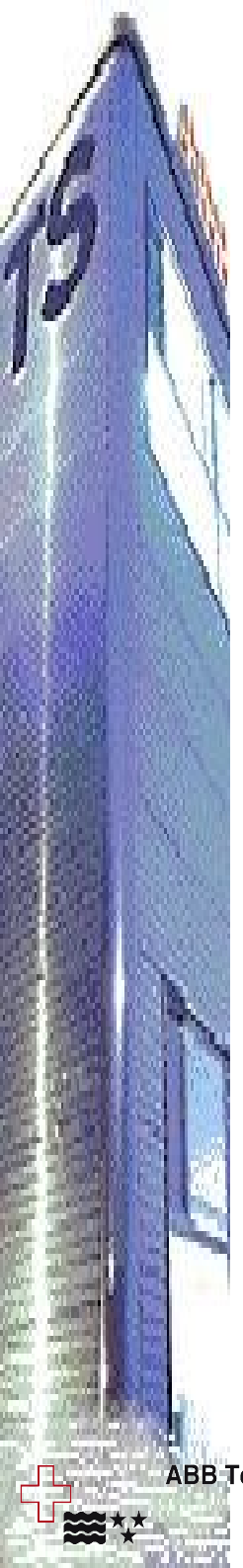
(Fortsetzung)

- beim Erzeugen einer Datei wird der Zugriffsmodus zuerst nach einem bestimmten Muster festgelegt
  - umask
- der Benutzer kann mit `umask` den Zugriffsmodus selber einstellen

Das Linux-Rechtekonzept

(Fortsetzung)

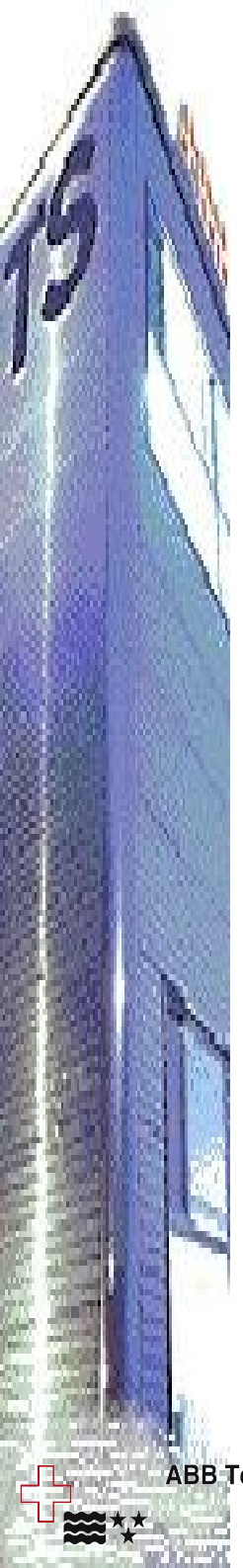
- Einige Programme vergeben Berechtigungen, die von den Voreinstellungen abweichen
  - z.B der Linker, wenn er ein ausführbares Programm erzeugt
- es gibt auch Fälle, in denen die Voreinstellungen nicht funktionieren
  - z.B. Shell-Skripte oder Perl-Programme



Das Linux-Rechtekonzept

(Fortsetzung)

- Eigentümer und Gruppenzuordnung einer Datei kann der Systemverwalter mit `chown` ändern
- normale Benutzer können die Gruppenzuordnung einer Datei in gewissen Grenzen mit `chgrp` modifizieren



Die *umask*

- Neue Dateien und Verzeichnisse werden, mit den standard Zugriffsmodi angelegt
  - 666 für Dateien
  - 777 für Verzeichnisse
- die *umask* ist eine Oktalzahl
- das Komplement wird mit dem Standard-zugriffsmodus bittweise UND-verknüpft

Die *umask*

(Fortsetzung)

- Ein Beispiel – die umask sei 027:

1	Wert der *umask*:	027	----w-rwx
2	Komplement davon:	750	rwxr-x---
3	Zugriffsmodus neue Datei:	666	rw-rw-rw-
4	Resultat (2 UND 3):	640	rw-r-----

Die *umask*

(Fortsetzung)

- Die *umask* wird mit dem Kommando `umask` gesetzt
- explizit oder in einer Startdatei der Shell
- typischerweise
  - ~/.profile
  - ~/.bash_profile
  - ~/.bashrc

Die *umask* (Syntax)

- Das Kommando `umask` bekommt einen Parameter
  - `umask [-S|<Umask>]`
  - Parameter sind oktal oder symbolisch
    - `umask 027`

      ist dasselbe wie ...
    - `umask u=rwx,g=rx,o=`

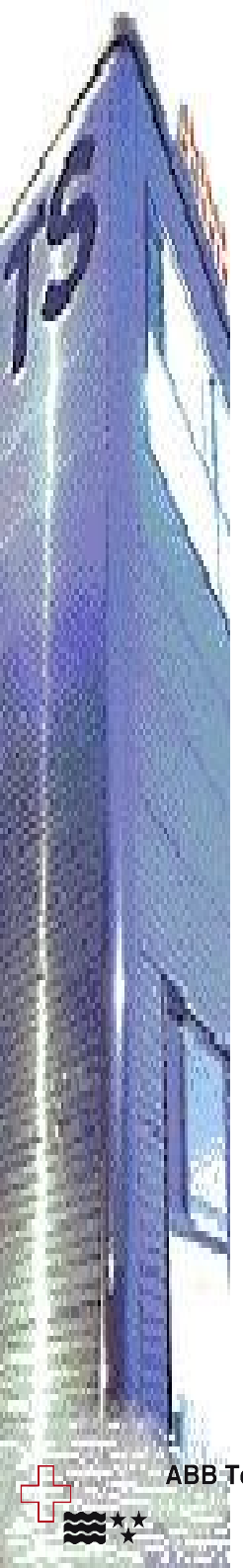
Die *umask* (Syntax)

(Fortsetzung)

- ohne Parameter, bzw. mit `-S` wird die aktuelle *umask* angezeigt
  - `umask`

    `0027`
  - `umask -S`

    `u=rwx,g=rx,o=`



Die *umask*

(Fortsetzung)

- mit *umask* können nur Zugriffsrechte entfernt werden
- mit *umask* werden keine Zugriffsrechte von bestehenden Dateien verändert
- Es gibt keine Möglichkeit, einer Datei automatisch Ausführungsrecht zu geben

Die *umask*

(Fortsetzung)

- die *umask* hat auch Einfluss auf den Befehl `chmod` im “+“-Modus aufgerufen
- äquivalent zu
  - `chmod a+ <Datei>`
- die in der *umask* gesetzten Rechte werden nicht modifiziert

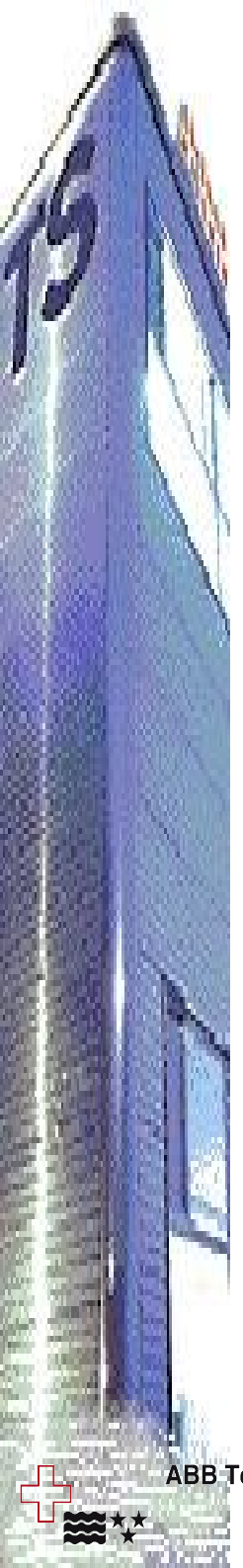
Die *umask*

- ein Beispiel:

```
$ umask 027
$ touch file
$ chmod +x file
$ ls -l file
-rwxr-x--- 1 tux users 0 May 25 14:30 file
```

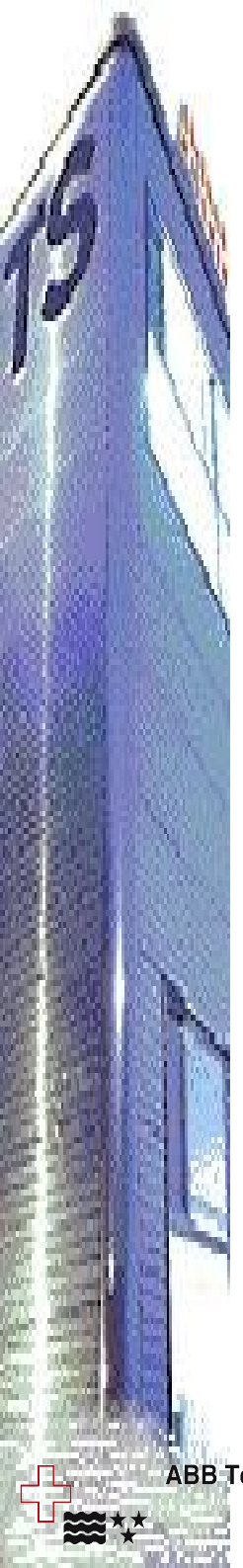
- Das `chmod +x` setzt das Ausführrecht für den Dateieigentümer und die Gruppe, aber nicht für den »Rest der Welt«, da die *umask* kein Ausführrecht für den »Rest der Welt« enthält

Fragen?



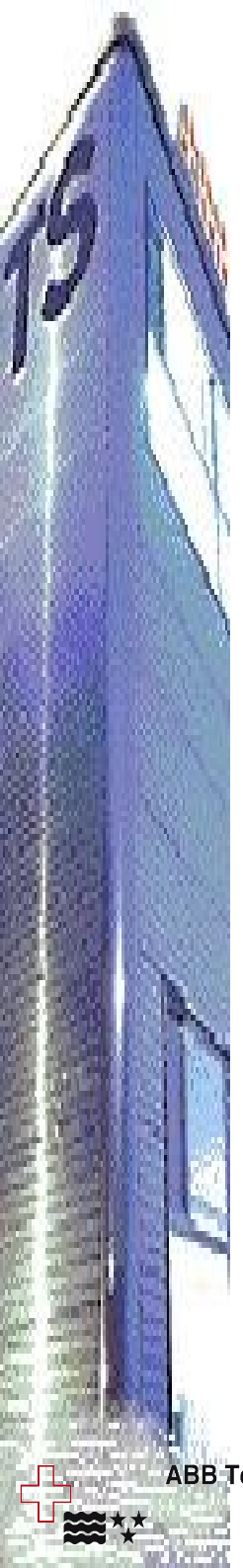
Übungen

- [3.1] Wie muss eine numerische umask aussehen, die dem Benutzer alle Rechte lässt, aber Gruppenmitgliedern und dem »Rest der Welt« alle Rechte entzieht?
- Was ist die entsprechende symbolische umask?



Übungen

- [3.2] Vergewissern Sie sich, dass der Unterschied zwischen den Kommandos `chmod +x` und `chmod a+x` wirklich so aussieht wie beschrieben.



Eigentum an Prozessen

- Der Eigentumsbegriff gilt auch für die Prozesse im System
- Die meisten Kommandos erzeugen einen Prozess im Arbeitsspeicher des Rechners
- es befinden sich i.d.R. immer mehrere Prozesse gleichzeitig im Speicher
- sie werden vom Betriebssystem streng voneinander abgegrenzt

Eigentum an Prozessen

(Fortsetzung)

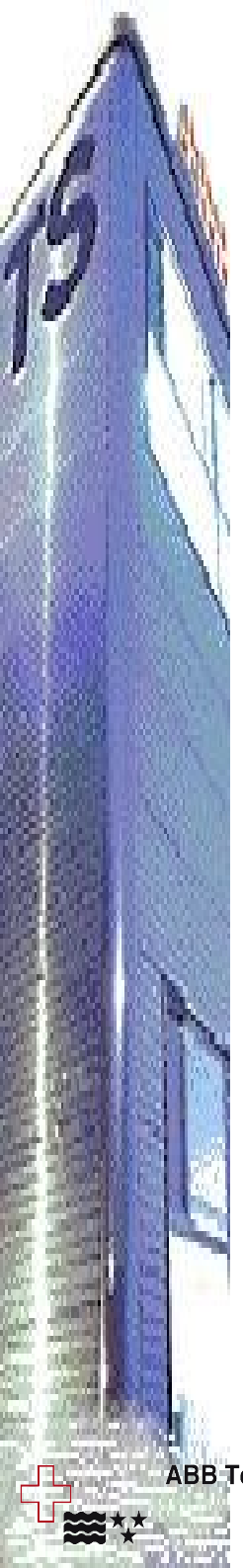
- Prozesse werden mit ihrem virtuellen Adressraum einem Benutzer zugeordnet
- i.d.R. dem Benutzer, der den Prozess gestartet hat
- Prozesse können ihre Identität wechseln
  - SUID-Mechanismus

Eigentum an Prozessen

(Fortsetzung)

- Eigentümer der Prozesse werden vom Programm `ps` mit der Option `-u` angezeigt
- `ps -u`

```
USER    PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
bin      89  0.0  1.0  788 328   ?   S  13:27 0:00 rpc.portmap
test1   190  0.0  2.0 1100  28   3   S  13:27 0:00 bash
test1   613  0.0  1.3  968  24   3   S  15:05 0:00 vi XF86.tex
nobody  167  0.0  1.4  932  44   ?   S  13:27 0:00 httpd
root      1  0.0  1.0  776  16   ?   S  13:27 0:03 init [3]
root      2  0.0  0.0    0   0   ?  SW  13:27 0:00 (kflushd)
```

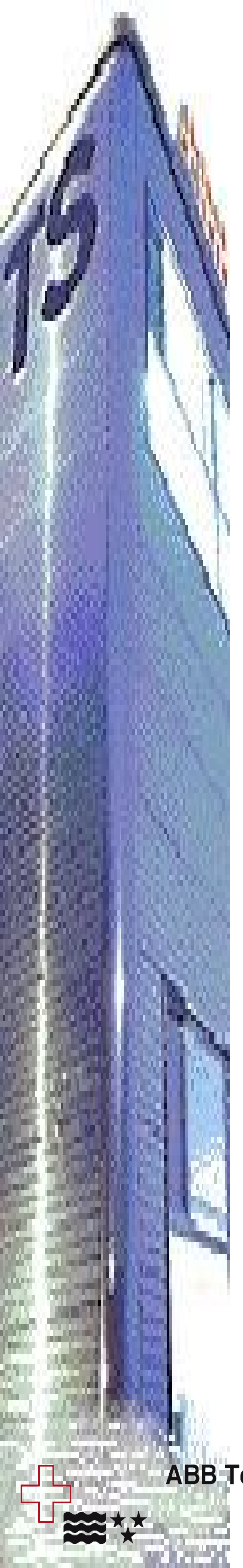


Besondere Zugriffsrechte für ausführbare Dateien

- Beim Auflisten der Dateien sehen Sie manchmal Anzeigen wie:

```
-rwsr-xr-x 1 root shadow 32916 Dec 11 20:47 /usr/bin/passwd
```

- hier ist an Stelle des Ausführungsrecht das **Set-UID-Bit** gesetzt
- mit Hilfe des **SUID-Bit** kann ein Programm mit den Rechten des Eigentümers ausgeführt werden



Besondere Zugriffsrechte für ausführbare Dateien

(Fortsetzung)

- Analog zum Set-UID-Bit gibt es auch ein **SGID-Bit**
- ein Prozess wird dann mit den Rechten der Gruppenzugehörigkeit ausgeführt

Besondere Zugriffsrechte für ausführbare Dateien

(Fortsetzung)

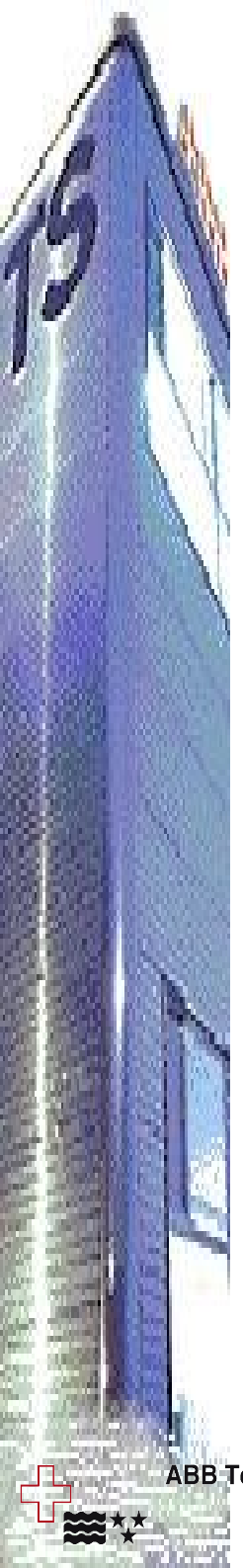
- die SUID- und SGID-Modi werden mit dem Kommando chmod verändert
- symbolische Schlüssel wie
  - u+s (setzt das SUID-Bit)

    oder
  - g-s (löscht das SGID-Bit)

Besondere Zugriffsrechte für ausführbare Dateien

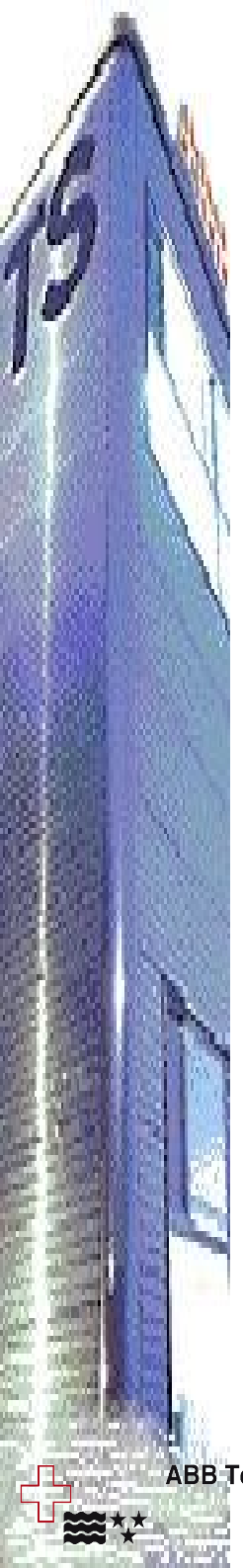
(Fortsetzung)

- Auch in oktalen Modi können Sie diese Bits setzen
- eine vierte Ziffer ganz links hinzufügen
  - das SUID-Bit hat den Wert 4
  - das SGID-Bit hat den Wert 2
- z.B. `chmod 4755 <Dateiname>`



Besondere Zugriffsrechte für Verzeichnisse

- das SGID-Bit eines Verzeichnisses wird gesetzt
- Zuordnung des Eigentums an Dateien nach dem Verursacherprinzip
- in einem Verzeichnis erzeugte Dateien gehören der gleichen Benutzergruppe wie das Verzeichnis selbst



Besondere Zugriffsrechte für Verzeichnisse

(Fortsetzung)

- beim Verschieben von Dateien wird deren Gruppe nicht angepasst
- beim Kopieren bekommen Dateien die Gruppe des Verzeichnisses
- Programme wie `chgrp` arbeiten in SGID-Verzeichnissen völlig normal
- das SUID-Bit hat auf Verzeichnisse keine Wirkung

Besondere Zugriffsrechte für Verzeichnisse

(Fortsetzung)

- ein weiterer Spezialmodus für Verzeichnisse ist das **sticky bit**
  - auch t-Modus genannt

```
drwxrwxrwt 7 root root 1024 Apr 7 10:07 /tmp
```

- gemeinsame Verwendung öffentlicher Verzeichnisse
- Datei in diesem Verzeichnis können nur von ihrem Eigentümer gelöscht werden

Besondere Zugriffsrechte für Verzeichnisse

(Fortsetzung)

- das sticky bit kann über die symbolische Angabe mit `chmod`
  - `+t` bzw. `-t`

  gesetzt oder gelöscht werden

- im oktalen Modus hat es in derselben Ziffer wie SUID und SGID den Wert 1

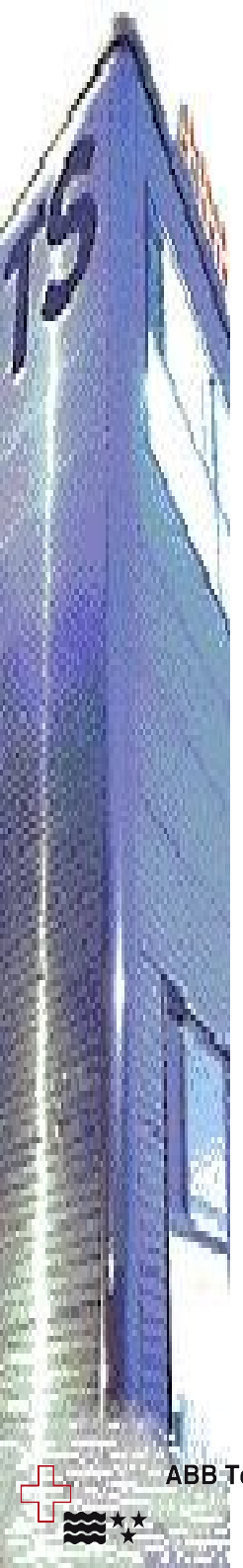
Fragen?

Übungen

- [3.3] Was bedeutet das spezielle Zugriffsrecht »s«?
- Wo finden Sie es?
- Können Sie dieses Recht auf eine selbst erstellte Datei setzen?

Übungen

- [3.4] Was bedeutet das spezielle Zugriffsrecht »t«?
- Wo finden Sie es?



Übungen

- [3.6] Wenn Sie Ihren PC für ein paar Minuten mit einer root-Shell alleine lassen, könnten findige Benutzer versuchen, irgendwo im System eine Shell zu hinterlegen, die sie SUID root gemacht haben, um auf diese Weise nach Bedarf Administratorrechte zu bekommen.
- Funktioniert das mit der Bash?
- Mit anderen Shells?

Dateiattribute

- Moderne Dateisysteme wie *ext2*, *ext3* unterstützen neben den Zugriffsrechten weitere Dateiattributte
  - auch ACLs genannt
- es ist vom aktuellen Kernel abhängig, welche Attribute unterstützt werden
- nur *root* darf diese setzen oder löschen
- sie sind auch für Prozesse verbindlich

Dateiattribute

(Fortsetzung)

Die wichtigsten Dateiattribute

Attribut	Bedeutung
A	atime wird nicht aktualisiert (interessant für mobile Rechner)
a	(append-only) An die Datei kann nur angehängt werden
d	Datei wird von dump nicht gesichert
i	(immutable) Datei kann überhaupt nicht verändert werden
j	Schreibzugriffe auf den Dateiinhalt werden im Journal gepuffert (nur ext3)
S	Schreibzugriffe auf die Datei werden »synchron«, also ohne interne Pufferung, ausgeführt

Dateiattribute

(Fortsetzung)

- A-Attribut
  - kann auf Notebook-Rechnern dafür sorgen, dass die Platte nicht ständig läuft
- append-only-Attribut
  - kann z.B. Protokolldateien vor Veränderungen schützen
- immutable-Attribut
  - kann z.B. Konfigurationsdateien schützen

Dateiattribute

(Fortsetzung)

- mit dem Kommando `chattr` kann man Attribute setzen und löschen
  - ein vorgesetztes »+« setzt ein Attribut
  - ein vorgesetztes »-« löscht ein Attribut
  - ein vorgesetztes »=« bestimmt ein Attribut

```
chattr +a /var/log/messages  Nur anhängen
chattr -R +j /data/wichtig   Daten ins Journal
chattr -j /data/wichtig/nichtso ... Ausnahme
```

Dateiattribute

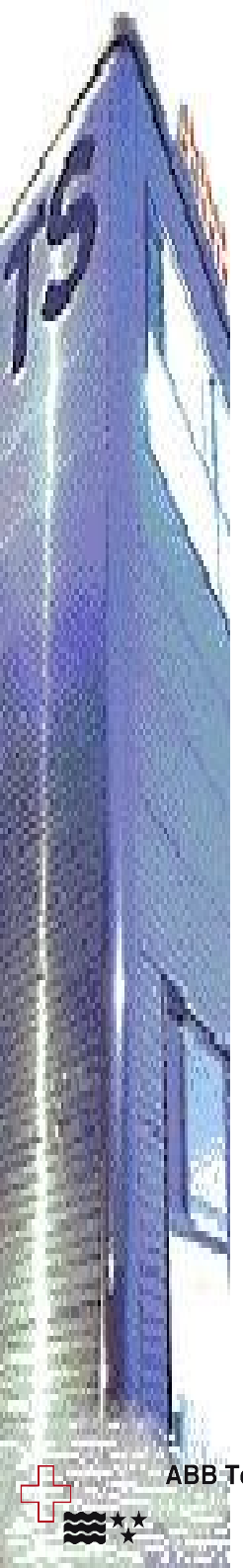
(Fortsetzung)

- mit dem Kommando `lsattr` kann man die Attribute anschauen

```
lsattr /var/log/messages
-----a----------- /var/log/messages
```

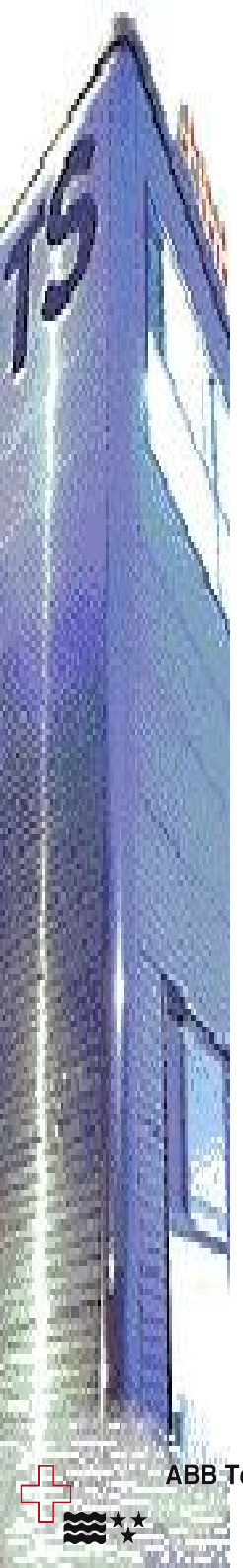
- Jeder Strich steht für ein mögliches Attribut
- *lsattr* unterstützt verschiedene Optionen wie -R, -a und -d (wie *ls*)

Fragen?



Übungen

- [3.7] Vergewissern Sie sich, dass die a- und i-Optionen funktionieren wie behauptet.
- Sie können auch die A-Option testen



Übungen

- [3.8] Können Sie für eine Datei alle Striche in der lsattr-Ausgabe zum Verschwinden bringen?