

## Systemmanagement und Sicherheit

### 2. Übung

#### Aufgabe 1 (Links und tar)

In dieser Aufgabe benötigen Sie evtl. Informationen aus den Manualpages von `ln` und `tar`.

Erzeugen Sie mit dem Kommando `dd` eine Datei, die 1 MB groß ist.

```
dd if=/dev/random of=testfile bs=1M count=1
```

Erzeugen Sie einen Hardlink `testlink` auf `testfile`.

Archivieren Sie beide Dateien mit `tar`. Wie groß ist das Archiv?

Ändert sich die Antwort, wenn `testlink` ein Softlink (engl. *symbolic link*) ist?

Ergründen Sie hierzu auch das Verhalten bei Verwendung der `tar` Option `-L`.

#### Aufgabe 2 (Systemcalls)

Tragen Sie für jedes der folgenden Kommandos einige Angaben zusammen:

`rm`, `mv`, `chmod`, `chown`, `mkdir`, `rmdir`, `kill`, `ln`, `sleep`, `wget`

- eine Kurzbeschreibung (eine Zeile)
- den Installationspfad (mit Hilfe des Kommandos `which`)
- ein Beispiel
- den/die entscheidenden Systemcall(s) (höchstens 2)
  - die entscheidenden Systemcalls tauchen relativ am Ende der Ausgabe auf, weil vorher Libraries geladen und Nebenbedingungen überprüft werden
  - oft heißt der Systemcall genau so wie das Kommando

Beispiel: Kommando *rm*

- a) löscht Dateien, aber standardmäßig keine Directories
- b) `/bin/rm`
- c) `rm text.doc`
- d) `unlink()`

Bemerkungen:

- Lesen Sie die zum jeweiligen Systemcall gehörige Manualpage.
- Die von einem Prozeß aufgerufenen Systemcalls werden mittels `ktrace/kdump` angezeigt.
- Bei `wget` interessiert hauptsächlich, an welcher Stelle die Verbindung zur Webseite aufgebaut wird.

### Aufgabe 3 (Inode Informationen)

Sehen Sie sich die Manualpage `stat(2)` an.

Diese beschreibt drei Funktionen, beachten Sie deren Unterschiede. Benutzen Sie diejenige Funktion, die für einen symbolischen Link den Dateityp *symbolischer Link* ausgibt und nicht den Dateityp, auf den durch den Link verwiesen wird.

Schreiben Sie nun ein C-Programm, das für beliebig viele als Kommandozeilenparameter angegebene Dateien (falls diese existieren) die im folgenden genannten Angaben ausgibt.

- Filetyp
- User-ID und Gruppen-ID (Besitzer der Datei, Gruppeneigentümer) und den Namen des Benutzers (`getpwuid()`).
- Zugriffsbits im Oktalsystem
- Zeit des letzten Zugriffs
- Zeit der letzten Inode-Änderung
- Zeit der letzten Dateiänderung
- Zeit der Dateierstellung

Das Ausgabeformat der Uhrzeit sollte der Ausgabe des `date` Kommandos der Shell entsprechen. Hierzu verwenden Sie die Funktionen `time(3)` und `ctime(3)`.

Welche Filetypen gibt Ihr Programm für die folgenden Dateien aus (pipe=FIFO):

Teil	Datei	regulär (a)	dir (b)	pipe (c)	socket (d)	char (e)	link (f)
a)	/dev/ada0						
b)	/bin/sh						
c)	/usr/bin/tar						
d)	/var/spool						
e)	/etc/master.passwd						
f)	/tmp/.X11-unix/X0						

Hinweis: Sie können die Ausgabe Ihres Programms mit der des `file`-Kommandos vergleichen.

Falls die angegebene Datei nicht vorhanden ist, soll eine Fehlermeldung ausgegeben werden (`perror()` benutzen, um `errno` auszuwerten).

Erzeugen Sie ein `Makefile`, damit Ihr Programm mit dem Kommando `make` übersetzt werden kann.