htw saar

Studiengang Kommunikationsinformatik
Prof. Dr.–Ing. Damian Weber
Dipl.-Inf. Marion Bohr
Sarah Theobald, M.Sc.

Systemmanagement und Sicherheit

2. Übung

Aufgabe 1 (Systemcalls)

Tragen Sie für jedes der folgenden Kommandos einige Angaben zusammen:
`rm, mv, chmod, chown, mkdir, rmdir, kill, ln, sleep, wget`

a) eine Kurzbeschreibung (eine Zeile)

b) den Installationspfad (mit Hilfe des Kommandos `which`)

c) ein Beispiel

d) 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 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 2 (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.

Übertragen Sie `Makefile` und Sourcecode auf den Solaris-Rechner `stl-s-stud`. Übersetzen und starten Sie es dort. Welche Anpassungen mussten Sie vornehmen?

Mit Hilfe von

```
#ifdef __FreeBSD__
...
#endif
```

kann Code gekapselt werden, der nur auf FreeBSD-Maschinen compiliert wird. Ebenso gibt es

```
#ifdef __sun__
...
#endif
```

für Solaris (SunOS).

Alle vordefinierten Compilermakros lassen sich mit

```
cpp -dM </dev/null
```

auflisten.