Entwicklungsumgebung zur Systemprogrammierung

Editor
Zahlreiche Unix und Linux-Derivate stellt eine POSIX-kompatible Umgebung zur Systemprogrammierung zur Verfügung. Zur Editierung kann z.B.

- der in der Vorlesung vorgestellte Editor nano
- gedit (Ubuntu Standard-Texteditor mit graphischer Benutzungsschnittstelle)
- vi (Standard Unix Text-Editor)
- GNU emacs
- ggf. auch Eclipse (als Editor, nicht als IDE)

verwendet werden. Hier sollte sich jeder des bevorzugten Editors bedienen. Zur Compilation von C-Sourcen steht der GNU-C-Compiler zur Verfügung. Zum Test der Umgebung sollte man einmal das folgende prüfen:

- Erstellen eines C-Programms (z.B. HelloWorld.c) mit dem Editor der Wahl.
- Kompilation des Programms mit dem GNU-C-Compiler. Dieses kann z.B. in einem Terminalfenster erfolgen: gcc –o hello HelloWorld.c
- (Alternativ kann auch ein Makefile erstellt werden und über den Make-Mechanismus kompiliert und das ausführbare Programm erstellt werden)
- Starten des Programms in einem Terminalfenster mit ./hello, im Terminal erscheint dann die Ausgabe des Programms.

API zur Systemprogrammierung unter Posix
Prozessmanagement:

Aufruf	Beschreibung
`pid = fork()`	Erzeugt einen neuen Kindprozess vom Vater
`pid = waitpid(pid, &statloc, options)`	Wartet auf Beendigung eines Kindes
`s = execve(name, argv, environp)`	Speicherabbild eines Prozesses ersetzen
`exit(status)`	Prozess beenden und status zurückgeben

Dateimanagement:

Aufruf	Beschreibung
`fd = open(file, how, ...)`	Datei zum Lesen, Schreiben öffnen
`s = close(fd)`	Offene Datei schließen
`n = read(fd, buffer, nbytes)`	Daten aus Datei in Puffer lesen
`n = write(fd, buffer, nbytes)`	Daten aus Puffer in Datei schreiben
`position = lseek(fd, offset, whence)`	Dateilesezeiger bewegen
`s = stat(name, &buf)`	Status einer Datei ermitteln

Prof. Dr.-Ing. Martin Hoffmann
martin.hoffmann@fh-bielefeld.de

Verzeichnis- und Dateimanagement:

Aufruf	Beschreibung
`s = mkdir(name, mode)`	Erzeugen eines neuen Verzeichnisses
`s = rmdir(name)`	Löschen eines leeren Verzeichnisses
`s = link(name1, name2)`	Neuer Eintrag name2 zeigt auf name1
`s = unlink(name)`	Verzeichniseintrag löschen
`s = mount(spezial, name, ag)`	Dateisystem einhängen
`s = umount(special)`	Eingehängtes Dateisystem entfernen

Verschiedenes:

Aufruf	Beschreibung
`s = chdir(dirname)`	Wechsel des aktuellen Verzeichnisses
`s = chmod(name, mode)`	Änderung der Dateirechte
`s = kill(pid, signal)`	Signal an einen Prozess schicken
`seconds = time(&seconds)`	Zeit seit dem 1. Januar 1970 in Sekunden abfragen

Kommentar zu `time()`: Schaltsekunden werden nicht mitgezählt. Bei Speicherung der Zeit als 32 Bit Integer (signed) wird es am 19. Januar 2038 um 3:14:08 UTC zu einem Speicherüberlauf kommen. Daten vor dem 13. Dezember 1901 20:45:52 UTC sind mit diesem Ansatz nicht darstellbar. Bei Verwendung von unsigned-Darstellung kommt es in 2106 zu einem Problem (Zeiten vor dem 1. Januar 1970 sind nicht darstellbar).

Prof. Dr.-Ing. Martin Hoffmann
martin.hoffmann@fh-bielefeld.de

Beispiele:

Beispiel 1

`link("/home/user1/file", "/home/user2/linkToFile" )`:

Vorher		Nachher	
`/home/user1/`	`/home/user2/`	`/home/user1/`	`/home/user2/`
`file`		`file`	`linkToFile`

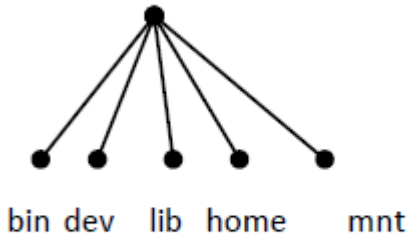
`file` und `linkToFile` verweisen auf den gleichen Bereich auf der Festplatte.

Beispiel 2

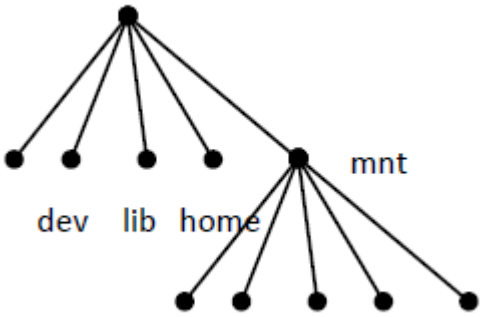
`mount("/dev/fd0", "mnt")`

bindet das Diskettenlaufwerk ein. Alle Geräte (Drucker, Scanner, ...), Dateisysteme, Laufwerke (CD, DVD, Floppy, USB-Stick,...) werden in das Filesystem eingebunden und wie eine Datei angesprochen!

Vorher



Nachher



Prof. Dr.-Ing. Martin Hoffmann
martin.hoffmann@fh-bielefeld.de