daemon programmieren...

- Chemnitzer Linux User Group
- Freitag 2011-08-12
- Zum Frohen Zecher

Was ist ein „echter“ Unix daemon

- ~ ist ein Programm, dass mit dem Systemstart seine Arbeit aufnimmt.
- ~ ist in das `init`-System eingebunden.
- ~ ist speziell programmiert, damit er sich problemlos in das System integriert. Siehe:

  `http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize`
- ~ kümmert sich selbst um seine Log-Dateien.

Dough Potter's spell

- To properly daemonize, the following steps must be followed.
  - The fork() call is used to create a separate process.
  - The setsid() call is used to detach the process from the parent (normally a shell).
  - The file mask should be reset.
  - The current directory should be changed to something benign.
  - The standard files (stdin,stdout and stderr) need to be reopened.

Siehe Levent Karakas www.enderunix.org

- Daemonizing == using `fork()`
- Process independency == using `setsid()/setgrp()`
- Handling open inherited or Standard I/O descriptors, i.e. stdin/stdout/stderr
- Set File Creation Mask
- Setting a directory
- Mutual Exclusion / single copy
- Catching signals
- seting up Logging

Dan J. Bernstein alias cr.yp.to

cr.yp.to/djb

…

cr.yp.to/daemontools

Mathematiker und „computer scientist“
Verschüsselungsexperte
etliche Professuren rund um den Erdball

Installation (Debian squeeze)

- Installation des Paketes „daemontools“

  # apt-get install daemontools

  # apt-get install daemontools-run svtools

- Zwei Zusatzpakete
  - daemontools-run
    - aktiviert mit Installation svscanboot via inittab:
      `SV:123456:respawn:/usr/bin/svscanboot`
  - svtools
    - liefert Werkzeuge zur Ansicht und Verarbeitung von Log-Dateien

Bau den Dienst zusammen

- der Ordnung halber: Erzeuge ein neues Verzeichnis als und unter root

```
# mkdir service-leds
# cd service-leds
```

- Baue das Programm

```
# vi leds.c
# gcc leds.c -o leds
```

- Teste und Dokumentiere

Lass den Dienst rennen

- Erzeuge den `run` script in `/root/service-leds`

```
#!/bin/csh

exec ./leds # exec directs signals
```

```
# chmod 0700 run
```

- Verlinke den Dienst in das -verzeichnis

```
# ln -s /root/service-leds /etc/service/leds
```

- Was passiert?

```
# ps ax
```

Töpfchen steh!

- svstat <Dienstverzeichnis>
  - liefert den Status des Dienstes

```
# svstat /etc/service/leds
/etc/service/leds: up (pid 4570) 3 seconds
```

- svc -<Aktion> <Dienstverzeichnis>
  - steuert den Dienst

```
# svc -d /etc/service/leds # stopt den Dienst (down)
# svc -u /etc/service/leds # startet den Dienst (up)
# svc -t /etc/service/leds # startet neu
```

Hochverfügbarkeit!

- svscan verfolgt die laufenden Dienste
- aller 5 s wird nachgeschaut, ggf. der Dienst neu gestartet
- Bei Aktivierung der Logging Funktion wird das Logfile historisch gekürzt; ein Dateisystemüberlauf wird verhindert
- Problem: Ein Dienst der sich selbst „aufhängt“ (threads werden nicht sauber beendet usw.); der wird als „laufend“ weiter baumeln lassen

Mehr, mehr...

- Vortrag „Anwendungen hochverfügbar machen mit den Daemontools“
  - Dr. Erwin Hoffmann
  - www.fehcom.de
  - http://www.fehcom.de/qmail/docu/05.pdf

Anwendung im embedded Bereich

- Epiphan VGA2Ethernet – Linux basiertes VGA-Grabber System
- gcc basiertes SDK für die Prozessorarchitektur PowerPC verfügbar
- Epiphan Bibliotheken und Progammierbeispiele
- laufendes Grundsystem mit sshd über zwei konfigurierbare Ethernet Schnittstellen

embedded Linux System

- booted via uBoot vom USB stick
  (cuImage, uRamdisk)
- lädt ein Squash-Filesystem als „root“
- daemontools bereits installiert und via inittab
  mit supervise Einträgen für sshd, ntpd usw.
- Konfigurationsmöglichkeit via /boot/default.cf

Vorgehen

- Programm entwickeln und auf einem Linux-PC für die AMCC Architektur bauen („cross compiling“)
- per scp oder USB-stick kopieren und testen
- SquashFS auf dem PC entpacken und in der inittab einen zusätzlichen Dienst anmelden
- run script anlegen (hier /service/qtwallscreen)
- dabei binaries und Konfiguration von /boot einbinden
- SquashFS wieder zusammenpacken und neben cuImage und uRamdisk auf den boot stick packen

Dienst integrieren

- Binary und Konfiguration auf dem stick ergänzen
- Testen des automatischen Programmstarts
- Neue Versionen einspielen mit
  - mount -orw,remount /boot
  - scp ./cooleSoftware 192.168.137.100:/boot
  - ssh root@192.168.137.100
  - svc -t /service/

Beispiele

- Programme für die parallele Schnittstelle
- fsniper - „hotfolder manager“
- Projekte mit dem IO-Warrior
- media streaming
- ...

hope it works

- Ich wünsche
  - viele interessante Projekte
  - gutes Gelingen
  - viele interessante Fragen

vielen Dank!