Die Linux Shell

Effektives Arbeiten ohne Maus

Agenda

- Kurzvorstellung **LW**systems GbR
- Grundlagen Unix/Linux
  - Verzeichnishierarchie
  - Prozesse
- Arbeiten mit der Shell
  - Autocompletion
  - Editieren der Kommandozeile
- Wichtige Kommandos, systematische Fehlersuche

Kurzvorstellung

- Gründung 2004
- Gesellschafter:
  - Dipl.-Ing. Ansgar H. Licher
  - Dipl.-Ing. Martin Werthmöller
- jeweils über 15 Jahre praktische Erfahrung
  - IT-Administration auf unterschiedlichen Plattformen
  - IT- und Organisationsberatung
  - IT-Management und Projektleitung, Personalführung

Kompetenz-Schwerpunkte

- Zukunftsweisende IT-Strategien und -Konzepte
- plattformübergreifende Systemintegration
- EDI / EDIFACT Schnittstellen- und Integrationsprojekte
- EDV-Revision
- IT-Sicherheitsanalysen und -konzepte
- IT-Betriebsführungs- und Notfallkonzepte
- Projektleitung in IT-Projekten
- Service und Administration Server, Netze, WAN, VPN

Die Entwicklung von Unix und Linux

- 1969 bei den Bell Laboratories entwickelt
- 1973 in der „Hochsprache“ C neu geschrieben
- 1978 Aufspaltung in System V und BSD
- 1984 X-Window System
- 1991 Linux Kernel
- 1992 386/BSD
- 2002 MacOS X

Die Philosophie von Unix / Linux

- alles ist eine Datei
- modularer Aufbau
- für jede Aufgabe ein spezialisiertes Werkzeug
- jedes Werkzeug ist als Filter gestaltet
- Portabilität von Programmen und Daten
  - POSIX Standard
  - Konfiguration in ASCII Dateien

Prozesse

- im Speicher ablaufendes Programm
- Standardkanäle für die Ein- und Ausgabe
  - Standardeingabe (STDIN)
  - Standardausgabe (STDOUT)
  - Standardfehlerausgabe (STDERR)
- jeder Prozess hat eine ID
- Signale zur Steuerung
- Daemons

Dateitypen

- normale Dateien – Daten und Programme
- Verzeichnisse – Container für Dateien
- Gerätedateien – Schnittstellen zur Hardware
- Sockets – Kommunikationsendpunkte
- FIFOs – First In First Out
- Hardlinks – Verschiedene Namen einer Datei
- Symlinks – Verweise auf eine Datei

Die Verzeichnishierarchie

- Baumstruktur der Verzeichnishierarchie
- Grobe Unterteilung in
  - verteilte (Netzwerk-) Daten / lokale Daten
  - variable Daten / statische Daten
- Dateisysteme werden gemounted
  - von lokalen Partitionen
  - vom Servern
  - von Datei-Images

Die einzelnen Verzeichnisse

- / - das Wurzelverzeichnis
- /bin, /sbin - ausführbare Programme
- /etc - lokale Konfigurationsdateien
- /home - Heimatverzeichnisse der Benutzer
- /usr - User System Resources
- /var - häufig geänderte Daten
- /tmp – Temporäre Dateien
- /proc und /dev

Berechtigungen

- Unterscheidung in Benutzer, Gruppe, Alle
- Verwaltung mit **chmod** und **chown**
- Read, Write und Execute
- Set User ID on Execution
- Dateiattribute mit **chattr** verwalten

```
-r-xr-xr-x  ausführbar.txt
-r--r--r--  lesbar.txt
--w--w--w-  schreibbar.txt
-rwsr-xr-x  suid.txt
```

Dateinamen

- Pfadtrenner ist **/**
- auf aktuellen Linux- Dateisystemen bis zu 255 Zeichen
- können fast alle ASCII Zeichen enthalten
- unerwünschte Effekte können auftreten bei
  - Leerzeichen im Dateinamen
  - **-** Zeichen als 1. Zeichen
  - **\n** (newline) im Dateinamen

Die shell

- interaktiver Befehlsinterpreter
- nicht mit dem "DOS Fenster" zu vergleichen
- extrem konfigurierbar
- Funktionen zur komfortablen Bedienung
- verschiedene Shells
  - Bourne shell: **`bash`**
  - C-Shell: **`tcsh`**
  - spezielle Shells: **`scponly, sash, ...`**

Shellscripte

- gespeicherte Abfolge von Shellkommandos
- sind schnell und einfach zu erstellen
  - Kommandos in Datei schreiben
  - Datei im Programmsuchpfad speichern
  - Datei mit dem execute Recht versehen
- Kommandointerpreter wird in 1. Zeile angeben

  `#!/bin/sh`
- Verzweigungen und Schleifen sind möglich

Autocompletion

- steigert den Komfort erheblich
- Vervollständigen von Kommandos mit **<TAB>**
- Suche nach Programmen im Pfad oder Datei- und Verzeichnisnamen
- programmierbare Autocompletion
  - Anzeige von Kommandoparametern
  - Anzeige von passenden Dateitypen (**`gunzip`**, **`xpdf`**)
  - Anzeige von Hostnamen (**`ping`**, **`ssh`**)

Editieren der Kommandozeile

- Steuerung über Tastenkombinationen
- Aufruf mit **<Alt>** oder **<Strg>** und Taste
- Tastenkürzel sind beliebig konfigurierbar
- Tastaturmakros aufzeichnen und abspielen
- Bedienung an Editoren angepasst
  - emacs – ... nur Kaffee kochen kann er noch nicht
  - vi - ... Small minds, small executables

Befehle wiederholen

- Blättern in der Befehlshistorie
  - **<Strg>-p**, **<Strg>-n**, Pfeiltasten
- Suchen in der History
  - **<Strg>-r**, **<Strg>-s**
- Editieren des Befehls
  - **<Strg>-k**, **<Strg>-y**
- Mehrfacheingabe eines Zeichens
  - **<Esc> <zahl>**

sonstige Editierfunktionen

- Einfügen des letzten Arguments
  - **<ESC> .**
- Kommando aus history editieren und ausführen
  - **fc** – letzes Kommando im Editor bearbeiten
  - **fc 20 25** – Kommandos 20 bis 25 bearbeiten
- Verzeichnisse zwischenspeichern
  - **pushd, popd**

Platzhalter und Substituierungen

- Wildcards: **`* ? [A-Z]`**
- Kommando-Substitution
  - **`rm `ls -1 [A-Z]*``**
- Prozess-Substitution
  - Befehl wird in eigener Subshell ausgeführt
  - Kein Einfluss auf die aktuelle Shell-Umgebung
  - **`(befehl;befehl)|grep error`**

Kommandoausführung

- Kommando abschicken mit **<RET>**
  - Shell „zerlegt“ Kommando in Token
  - Trennzeichen sind: **| & ; < > <RET>**
  - Maskieren mit **** oder in **"** bzw. **'** setzen
- Rückgabewert des Kommandos: **$?**
- Kommandos nacheinander ausführen
  - durch Trenner mit unterschiedlichen Funktionen
  - **& ; || && ( )**

Expansionen

- nach dem Zerlegen der Kommandozeile in Token
- Brace Expansion: **`ls /var/www/{html,php}`**
- Tilde Expansion: **`cd ~martin`**
- Pathname Expansion: **`ls /var/log/*/*.log`**
- History Expansion: **`!120`**
- Arithmetische Expansion: **`$(($i+1))`**

Ein- und Ausgabeumleitung

- Eingabeumleitung
  - **<** liest aus Datei
- Ausgabeumleitung
  - **>** und **>>** schreiben in Datei
- Befehlsverkettung (Pipes)
  - **|** Verbindung von STDOUT und STDIN
- STDOUT und STDERR umleiten
  - **`ls /usr /nofile > /tmp/out.txt 2>&1`**

Umgebungsvariablen

- Auf Umgebungsvariablen zugreifen
  - **`echo $HOME`**
- Umgebungsvariablen setzen
  - **`VARNAME="Der Wert"`**
- Umgebungsvariablen exportieren
  - **`export VARNAME`**
- Standard- Umgebungsvariablen
  - z.B.: **`$PS1 $HOME $PWD $PAGER $EDITOR`**

Konfiguration der Shell

- Konfigurationsdateien
- **`.bash_profile`** oder **`.profile`** für Login-Shells
- **`.bashrc`** für interaktive nicht-Login Shells
- Typische Einstellungen
  - aliase
  - Kommandoprompt mit **PS1**
  - "dircolors"
- **`.bash_logout`**

Jobkontrolle

- Prozess im Hintergrund ausführen
  - mit **&** aufrufen
  - mit **<Strg>-z** in den Hintergrund schalten
- Anzeigen der Hintergrund- Prozesse
  - **jobs** zeigt Jobnummer und Status aktueller Jobs
- Jobsteuerung
  - **fg** und **bg** schicken in den Vorder- bzw. Hintergrund
  - **wait <n>** wartet auf Ende von Job n

wichtige Kommandos I

- **`apropos -`** Welches Kommando für ...?
- **`man`** – Referenzmanuals
- **`find, locate`** – Dateien suchen
- **`grep`** – Zeichenketten in Dateien suchen
- **`tail`** – das Ende einer Datei ausgeben
- **`ps`** – Status von Prozessen anzeigen
- **`kill`** – Signale an Prozesse schicken

wichtige Kommandos II

- **`top`** – CPU und Speichernutzung anzeigen
- **`df`**, **`du`** – Speicherplatzbedarf anzeigen
- **`lsof`** – List open files
- **`fuser`** – welcher Prozess benutzt die Datei?
- **`netstat`** – Netzwerkdaten anzeigen
- **`strace`** – Systemaufrufe ausgeben
- **`screen`** – Screen Manager

systematische Fehlersuche

- Was besagt die Fehlermeldung genau?
- Schreibt das Programm Meldungen ins System Log?
- Kann das Programm im Verbose- oder Debug- Modus aufgerufen werden?
- Die Systemaufrufe mit strace(1) ausgeben
- Netzwerk mit tcpdump(1) bzw. ethereal(1) analysieren
- **LW**systems anrufen

Vielen Dank für Ihre Aufmerksamkeit!