Einführung in das Arbeiten an Unix-Rechnern

23. 03.2007

Franz Schenk

[Unix] Einführung

- Ursprünge, Hintergründe
- Grundlagen, Konzepte
- Arbeiten mit Unix

[Unix] Was ist Unix

- Ein MehrbenutzerBetriebssystem

  gemeint ist entweder

  - das Original aus den Bell Laboratories
  - oder Implementierungen der Unix-Konzepte (BSD, GNU/Linux, GNU/Hurd, MacOS, Solaris ...)

- Eine Sammlung nützlicher Tools

- Die Umgebung, in der sie sich im CIP-Pool zurechtfinden müssen

[Unix] Für wen ist Unix

- Ursprünglich für Großrechner
- Durch weite Verbreitung von Linux auch für 'einfache' Benutzer
- Nicht unbedingt für jedermann

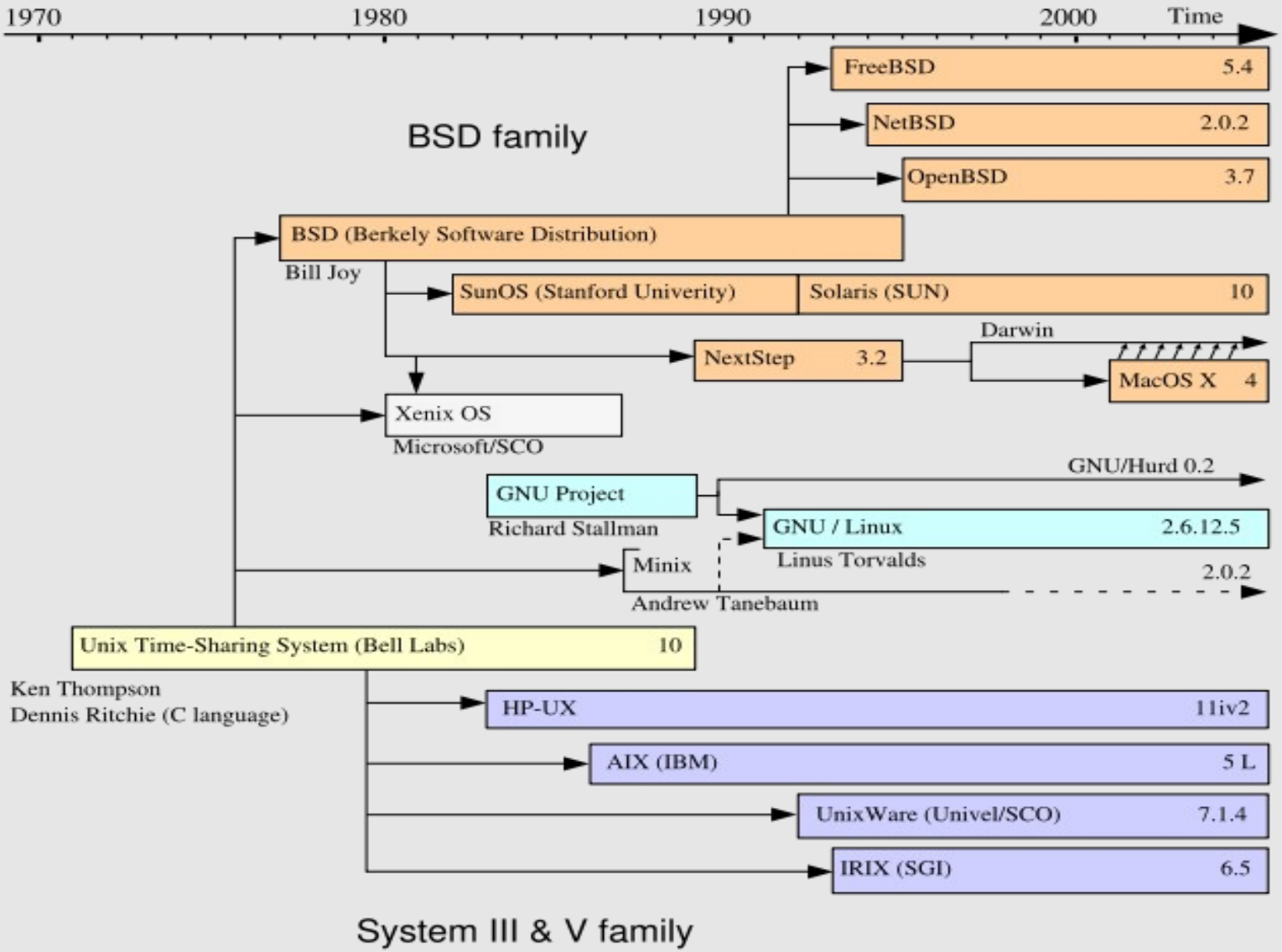
[Unix] Geschichte

- erste Variante von Ken Thompson 1969
  (in Assembler geschrieben)
- neue Version von Thomson und Dennis Ritchie 1972-74
  (komplettt neu geschrieben, dieses mal in C)
  - diese Version wurde frei zur Verfügung gestellt,
    vor allem an Universitäten eingesetzt
  - BSD entstand aus dem freien Unix an der Universität
    Berkley als eigene Modifikation des Unix Codes
- AT&T vermarktet kommerzielles UNIX (frühe 80er)
  -> Lizenzgebühren für Unix-Systeme und auch Derivate
  wie BSD, welche AT&T Code enthielten

[Unix] Geschichte

- Gründung von GNU 1983 durch Richard Stallman (Gnu is not Unix)
- bis ca 1990 sind die notwendigen Werkzeuge vorhanden (z.B. C compiler)
- FreeBSD erscheint 1992, von AT&T 'befreite' BSD-Unix-Version
- Linux als Unix-Kernel beginnt 1991, 1992 in GNU integriert und als erstes freies Unix-Betriebssystem vertrieben
- MacOS und Solaris sind eigene, teils auf BSD basierende, kommerzielle Unix-Derivate (mittlerweile freies OpenSolaris)

[Unix] Geschichte



Quelle: Wikipedia

[Unix] Vorteile von Unix

- Stabilität, durchdachtes Konzept
- Multi-User, Multi-Tasking System
- sicher, flexibel
- auf sehr vielen Rechnerarchitekturen lauffähig
- z.T. zugänglicher Quellcode und kostenlos

[Unix] Komponenten

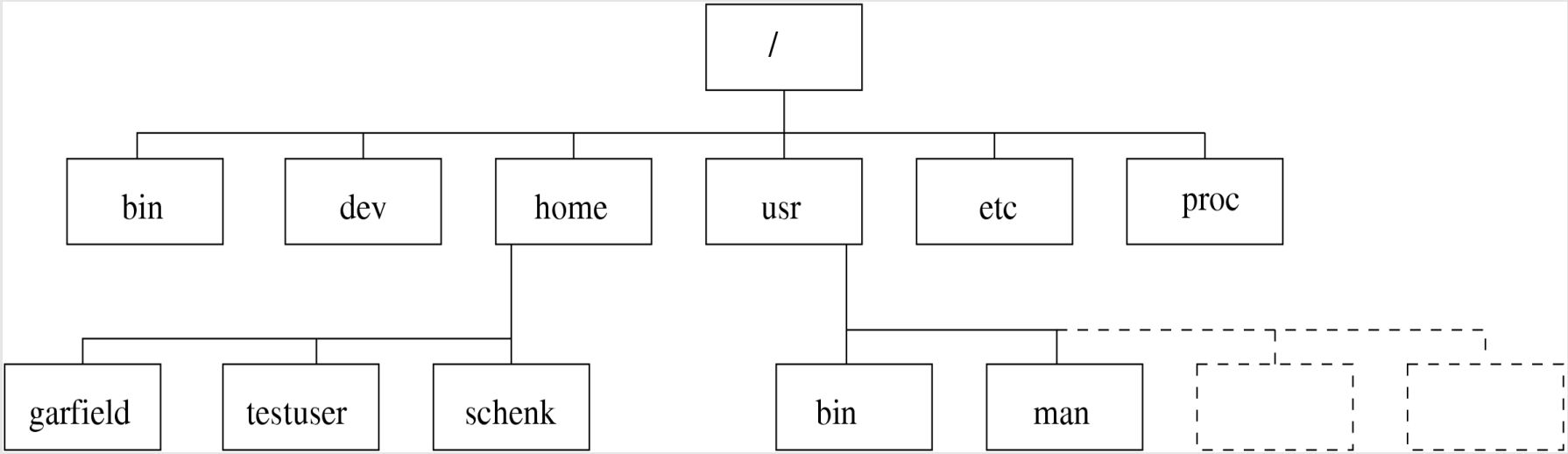
- Kernel: Betriebssystemkern
  - alleiniger Zugriff auf Geräte
  - stellt Filesystem zur Verfügung
  - verwaltet Prozesse
- Shell: Schnittstelle zwischen Benutzer und Kernel, CLI (Commandline Interface)
  - bash, zsh, ksh...
- Utilities
- Nicht zwingend: eine graphische Oberfläche

[Unix] Konzepte

- Alles ist ein Prozess
- Alles ist ein File -> Abbildung von Prozessen und Devices im hierarchischen Filesystem, vereinheitlichter Zugriff auf Resourcen
- Viele (auch kleine) Utilities, welche kombiniert werden können.

[Unix] Filesystem

- Ordner (directories) und Files hierarchisch geordnet
  - Ordner enthalten die Namen und Verweise auf Files
  - Files sind eine sequentielle Folge von Bytes
- Geräte (CD,Festplatte ...) werden als Teilbäume eingebunden
- case-sensitiv



[Unix] Pfade

Pfade sind die Lokalisatoren der Files im Filesystem.
Pfade können absolut (zur Baumwurzel)angegeben werden:

/home/schenk/data/beispiel.txt

oder relativ z.B. data/beispiel.txt

. .. ~ / sind spezielle Verzeichnisse

* läßt sich in Pfadausdrücken verwenden:

```
ls ~/*.txt
```

[Unix] Erste Befehle

Verzeichnisse anlegen: `mkdir tmp`

Bewegen im Verzeichnisbaum:

```
cd /tmp , cd , cd tmp , cd ~/tmp , cd ..
```

Wo bin ich? `pwd` (print current working directory)

Einfache Ausgabe:

```
echo 'Hallo Welt'
```

Ausgabeumleitung: > >>

```
echo 'Hallo Welt'  > tmp/test.txt
echo 'Hallo Welt'  >> tmp/test.txt
```

Anzeige von (Text-) Dateien

```
cat tmp/test.txt
more tmp/test.txt
less tmp/text.txt
```

[Unix] Noch mehr Befehle

'pipe' : Augabeweiterleitung

```
cat ~/tmp/test.txt | lc
```

#lc: linecount

suchen und finden: grep, find

```
find . -iname “test“
find ~/tmp -size -100c
grep Hallo `find ~/tmp -size -100c`
```

löschen:

```
rm filename
rmdir directory-name
```

[Unix] Editoren

Es kann nur einen geben?

- emacs / xemacs (C-h t)
- vi / vim / und Derivate (:help <enter>)
- nano / pico / und Derivate (C-g)
- nedit
- gedit
- kate, kwrite

...

Features, welche man haben will:
- Syntax-Highlighting
- suchen/ersetzen mit regulären Ausdrücken
- automatisches Einrücken
- ...

[Unix] Die Kommandozeile

Benötigt wird eine Konsole (z.B. xterm, konsole)
Diese stellt eine shell zur Verfügung (im Pool: *bash)*

Die Shell protokolliert mit: die *history [in vielen shells mit up/down]*

Muster eines Befehls:

```
befehlsname -option parameter
```

```
ls -al ~/tmp/test.txt
```

Hilfe zu einem Befehl: die *man-pages*

```
man ls
man man
```

alias als Abkürzung:

```
alias lstxt='ls -al *txt'
```

[Unix] Shell-Variablen

Setzen und lesen von Variablen

```
printenv EINEZAHL
echo $EINEZAHL
printenv EINEZAHL

printenv
```

Umgebungsvariablen

```
export PAGER=less
echo $PAGER

echo $SHELL
```

Pfadvariable

```
echo $PATH
export PATH=.
export PATH=$PATH:.
```

[Unix] Noch mehr Befehle

Befehle zur Benutzerverwaltung:
whoami , id
passwd

Befehle zur Prozessverwaltung:
top, ps
kill / killall
bg / fg

Befehle für Befehle
locate
which
apropos

Befehle für File-/Filesystemverwaltung:
tar, gunzip
mount
rm, rmdir, mv

[Unix] Zugangsrechte

Benutzer benötigen am System einen Benutzeraccount

Zugriffsrechte auf Dateien, Prozesse, Geräte ect. sind im Dateisystem geregelt (alles ist ein File)

Es gibt 'Besitzer' 'Gruppe' und 'Welt' (der Rest).
Jedes File gehört einem Benutzer und ist einer Gruppe zugehörig.

Für alle 3 Personengruppen gibt es drei unterschiedliche Rechte: read, write, execute

```
ls -al ~/tmp
-rw-r--r--  1 schenk schenk   11 2007-02-16 13:17 test.txt
```

[Unix] Zugriffsrechte Forts.

Zweite Variante:

```
schenk> chmod g+rwx,o-wx ~/tmp/test.txt
schenk> chmod g=u,o= ~/tmp.test.txt
```

[Unix] Zugriffsrechte

Modifikation der Zugriffsrechte:
Niemand ausser mir (dem Besitzer) soll die Datei test.txt lesen können:

```
schenk> chmod 444 test.txt
schenk> ls -al
-r--r--r--  1 schenk schenk 11 2007-02-16 13:17 test.txt

schenk> chmod g-rwx test.txt
schenk> ls -al
-r--r----- 1 schenk schenk 11 2007-02-16 13:17 test.txt
```

Erste Variante:
4: Leserecht 2: Schreibrecht 1: Ausführrecht
=> Lese-und Schreibrecht für Besitzer,Gruppe, sonst nichts:

```
schenk> chmod 660 test.txt
-rw-rw---- 1 schenk schenk 11 2007-02-16 13:17 test.txt
```

Weiter im Kurs:
- Benutzen des Editors zur Programmerstellung
- Benutzen des wichtigsten Werkzeug:
  der Compiler (z.B. gcc)