Notizen zum Kurs
Unix-Tools

WS 01/02
Universität Bremen

Stand: 22. Oktober 2001

(Zur Notation: Text in schattierten Kästen wird an die Tafel geschrieben, und Text in ovalen Kästen ist ein Dateiinhalt, der am Rechner vorgeführt wird.)

1 Überblick über die Veranstaltung Unix-Tools

1.1 Vorstellung des Veranstalters

- Jan Bredereke
- Assistent in **AG Betriebssysteme**/Verteilte Systeme

1.2 Ziele

Unter Unix stehen **leistungsfähige Tools** zur Verfügung,

- um Daten zu **analysieren** und zu **manipulieren,**
- insbesondere im **Zusammenspiel** mit weiteren Programmen.

Ergänzung zur Vorlesung **Betriebssysteme** von Jan Peleska.

Ziel dieses Kurses ist es,

- sich mit einigen oft gebrauchten Werkzeugen **vertraut zu machen** und
- zu wissen, wie und **wann man sie nutzbringend anwenden** kann.

1.3 Geplante $\boxed{\text{Inhalte}}$

Auswahl von Werkzeugen:

- sed
- perl
- make
- lex
- yacc

sed

- nützlich insbesondere in **Shell-Skripten**
- **selektieren** und **manipulieren** von Informationen **aus Dateien** oder der **Standardausgabe** von Kommandos

perl

- **komplexere** solche **Aufgaben**
- **sowohl** für **kurze Skripte** gut geeignet als auch für **große**, ausgewachsene **Programme** wie etwa aktive Webseiten mit Datenbankanbindung
- im wesentlichen auf die **Grundlagen** von perl **beschränken**, die **für Alltagsaufgaben** ausreichen

make

- erlaubt die **Automatisierung** der vielen Schritte, die zur **Generierung eines Programms** aus vielen Quelldateien notwendig sind
- **Verwaltung der Abhängigkeiten** dieser Schritte, was bei **Programmänderungen** wichtig wird

lex & yacc

- **generieren Frontends für** selbstgeschriebene, spezialisierte **Analyseprogramme**
- zum **Beispiel** für Compiler, Interpreter und interaktive Benutzerschnittstellen

1.4 Verfügbarkeit der Werkzeuge

Sind **auf allen Unix-Plattformen** verfügbar. Sollten an der **Uni überall** installiert sein.

Statt lex und yacc werden wir im wesentlichen die **Gnu-Versionen flex und bison** nehmen.

Die Gnu-Versionen aller Tools sollten **auch auf Microsoft-Plattformen** verfügbar sein.

Auf Microsoft-Plattformen wird allerdings **oft nmake** statt make verwendet.

1.5 Vorgehensweise

Vortrag und **Online-Übungen** im Kurs.

Kleine Hausaufgaben mit wenig Zeitaufwand. **Freiwillig,** aber **empfehlenswert** zum richtigen Verstehen.

Geschwindigkeit hängt stark von den **Vorkenntnissen** ab. Bitte **Rückmeldungen!**

Gehe gerne auf **Wünsche** zum Inhalt ein.

Bücher zum Kurs stehen auf der **Web-Seite**. Dort auch einige **Manuals zum Runterladen**.

Literatur

- N. N.: **sed(1) Manual-Seite**
  in: LunetIX Linuxhandbuch (Juli 1993).
- Mike Haertel, James A. Woods und David Olson: **grep(1) Manual-Seite**
  in: LunetIX Linuxhandbuch (Juli 1993).
- Randal L. Schwartz and Tom Phoenix: **Learning Perl**
  O'Reilly, 3. Auflage (Juli 2001). ISBN 0-596-00132-0.
- Larry Wall, Tom Christiansen und Jon Orwant: **Programming Perl**
  O'Reilly, 3. Auflage (Juli 2000). ISBN 0-596-00027-8.
- Richard M. Stallman und Roland McGrath: **GNU Make - A Program for Directing Recompilation**
  Version 3.79. Free Software Foundation (April 2000). ISBN-1-882114-80-9. GNU General Public License.
- John R. Levine, Tony Mason und Doug Brown: **lex & yacc**
  O'Reilly, zweite korrigierte Auflage (1995). ISBN 1-56592-000-7.
- Vern Paxson: **Flex, version 2.5**
  University of California (1990).
- Charles Donelly and Richard Stallman: **Bison**
  Free Software Foundation (1995). ISBN-1-882114-45-0. GNU General Public License.

Web-Seite des Kurses: http://**www.tzi.de/~agbs**/lehre/ws0102/unix-tools/unix-tools.html

Kurs Unix-Tools, WS 01/02, Jan Bredereke 1

Die **Perl**-Bücher gibt es auch in **deutscher Ausgabe**, allerdings das erste noch **nicht** in der **dritten Auflage**.

U.a. im **Addison-Wesley-Verlag** gibt es **ähnliche** Bücher, insbesondere vom Autoren Helmut **Herold**.

Prüfungen: Da der Kurs nur 2 SWS hat, kann man sich **nicht darüber prüfen** lassen. Er ist als freiwillige **Ergänzung zur Betriebssysteme**-Vorlesung gedacht.

2 Der batchorientierte Zeilen-Editor `sed`

2.1 Einfache Textersetzungen

Einige **Beispiele** für **typische Problemstellungen** aus dem Unix-Alltag.

- kleine Textmodifikation
- Formatierung einer Liste etwas ändern

2.1.1 Problem: Betreff in Vacation-Text einsetzen

Teilaufgabe in einem `procmail`-Skript. (`procmail` wird in dieser Vorlesung nicht behandelt.)

```
Liebe Kollegen, liebe Freunde,

mein Email-Programm hat die Nachricht betreffs
"<SUBJECT>"
empfangen. Ich werde sie lesen, sobald ich zurueck im Buero bin,
d.h. am Dienstag, den 4. September 2001, und werde sobald wie
moeglich reagieren.

Mit freundlichen Gruessen,
Jan Bredereke
```

....

```
Liebe Kollegen, liebe Freunde,

mein Email-Programm hat die Nachricht betreffs
"Re: Forschungsantrag"
empfangen. Ich werde sie lesen, sobald ich zurueck im Buero bin,
d.h. am Dienstag, den 4. September 2001, und werde sobald wie
moeglich reagieren.

Mit freundlichen Gruessen,
Jan Bredereke
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 2

2.1.2 Problem: Festen Präfix in einer Dateiliste entfernen

```
sed/passwd
sed/passwd_names.sh
sed/passwd_names.txt
sed/vacation-msg-filled.txt
sed/vacation-msg-tmpl.txt
sed/vacation-msg.sh
```

....

```
passwd
passwd_names.sh
passwd_names.txt
vacation-msg-filled.txt
vacation-msg-tmpl.txt
vacation-msg.sh
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 3

2.1.3 Lösung: Betreff in Vacation-Text einsetzen

```
subject='Re: Forschungsantrag'
sed "s/<SUBJECT>/${subject}/g" \
    < vacation-msg-tmpl.txt > vacation-msg-filled.txt
```

Die Variable **„subject“** wird in Wirklichkeit **von** `procmail` bestimmt.

2.1.4 Lösung: Festen Präfix in einer Dateiliste entfernen

```
sed -e 's!sed/!!' \
    < prefixfiles.txt > noprefixfiles.txt
```

Anmerkung: Wir mußten **„!“ statt „/“** nehmen, weil „/“ schon im Suchmuster vorkommt.

2.2 Die Grundkommandos von `sed`

Die Grundkommandos von `sed`

Syntax
sed [**-e** *Editorkommando*] [**-f** *Scriptdatei*] [**-n**] [*Datei...*]

Editorkommando
Adresse Funktion

Funktion

s/*Ausdruck***/***Ersetzung***/**[**g**][**p**]	(substitute)	ersetzt auf Ausdruck passenden Text
d	(delete)	löscht die aktuelle Zeile, nichts wird gedruckt
!*Funktion*		führt *Funktion* aus, falls die Adresse <u>nicht</u> paßt

Adresse
(leer)
/*Ausdruck***/**

Ausdruck
ein regulärer Ausdruck wie bei `grep`

Modus
g: ersetze alle passenden Texte, nicht nur den ersten
p: falls Ersetzung stattgefunden hat, drucke Zeile (trotz Option **-n**)

Kurs Unix-Tools, WS 01/02, Jan Bredereke 4

Wenn die **s-Funktion Slashes in Ausdruck oder Ersetzung** enthält, kann man statt der drei Slashes **drei beliebige andere Zeichen** verwenden: s!foo!bar!

Wenn der Ausdruck der **Adresse Slashes enthält**, kann man **stattdessen folgendes** verwenden: *c Ausdruck c*

Weitere Funktionen, Adressen und Optionen finden sich **in der Manualseite** von `sed`. **Man braucht sie** fast **nie**, weil sich die gleichen Ergebnisse **leichter mit** `perl` erreichen lassen.

2.3 Warum Reguläre Ausdrücke

- variable Textanteile erkennen
- interessante Textanteile extrahieren

2.3.1 Problem: Einen variablen Präfix in einer Dateiliste entfernen

```
./article/implicit-concepts.tex
./article/fi-example-sli.tex
./article/circumvent.tex
./article/blocked.tex
./article/maint-tel-req.tex
./article/tina-concepts.tex
./dagstuhl-talk.tex
./dagstuhl.tex
```

....

```
implicit-concepts.tex
fi-example-sli.tex
circumvent.tex
blocked.tex
maint-tel-req.tex
tina-concepts.tex
dagstuhl-talk.tex
dagstuhl.tex
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 5

2.3.2 Lösung: Einen variablen Präfix in einer Dateiliste entfernen

```
sed -e 's!^.*/!!' \
    < varprefixfiles.txt > novarprefixfiles.txt
```

2.3.3 Problem: Liste aller Benutzer aus /etc/passwd extrahieren

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
wwwrun:x:30:65534:Daemon user for apache:/var/lib/wwwrun:/bin/bash
named:x:44:44:Nameserver Daemon:/var/named:/bin/bash
nobody:x:65534:65534:nobody:/var/lib/nobody:/bin/bash
brederek:x:500:100:Jan Bredereke:/home/brederek:/bin/bash
mueller:x:501:100:B. Mueller:/home/mueller:/bin/bash
meyer:x:502:100:L. Meyer:/home/meyer:/bin/bash
schulze:x:503:100:G. Schulze:/home/schulze:/bin/bash
```

....

```
root
bin
daemon
Daemon user for apache
Nameserver Daemon
nobody
Jan Bredereke
B. Mueller
L. Meyer
G. Schulze
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 6

2.3.4 Lösung: Liste aller Benutzer aus /etc/passwd extrahieren

Schritt für Schritt an der Tafel entwickeln:

```
sed \
    -e 's/^\([^:]*:\)\{4\}\([^:]*\):.*/\2/' \
    < passwd > passwd_names.txt
```

2.3.5 Problem: Liste aller „echten“ Benutzer aus /etc/passwd extrahieren

Erweiterung des vorigen Problems.

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
wwwrun:x:30:65534:Daemon user for apache:/var/lib/wwwrun:/bin/bash
named:x:44:44:Nameserver Daemon:/var/named:/bin/bash
nobody:x:65534:65534:nobody:/var/lib/nobody:/bin/bash
brederek:x:500:100:Jan Bredereke:/home/brederek:/bin/bash
mueller:x:501:100:B. Mueller:/home/mueller:/bin/bash
meyer:x:502:100:L. Meyer:/home/meyer:/bin/bash
schulze:x:503:100:G. Schulze:/home/schulze:/bin/bash
```

```
Jan Bredereke
B. Mueller
L. Meyer
G. Schulze
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 7

2.3.6 Lösung: Liste aller „echten“ Benutzer aus /etc/passwd extrahieren

Ein **weiteres** `sed`**-Kommando vorweg**, der Rest bleibt gleich:

```
sed -e '/^\([^:]*:\)\{2\}5[0-9][0-9]:.*/!d' \
    -e 's/^\([^:]*:\)\{4\}\([^:]*\):.*/\2/' \
    < passwd > passwd_realnames.txt
```

2.4 Aufbau der Reguläre Ausdrücke von `sed` (und `grep`)

Reguläre Ausdrücke von `sed` (und `grep`)

Ausdruck
eine Folge von *Muster*n, optional mit *Zwischenraummarkierung*en dazwischen

Muster
ein Muster ist ein einzelnes *Zeichen* oder eine *Gruppierung*

Zeichen

c	einzelner Buchstabe: paßt auf sich selbst
.	Punkt: paßt auf jedes Zeichen außer dem Zeilenende
c	Backslash: macht aus Sonderzeichen normale Zeichen
[*Buchstaben*]	Menge von Buchstaben: paßt auf eines der Zeichen aus dieser Menge
	Buchstaben können auch Bereiche *von-bis* sein
	die Buchstaben „[“, „]“ und „-“ müssen ggf. ganz vorne oder hinten stehen
[^*Buchstaben*]	Komplementmenge von Buchstaben: paßt genau auf keines der Zeichen

Reguläre Ausdrücke (Fortsetzung)

Gruppierung

*Muster**	eine Folge von 0 oder mehr Mustern
Muster?	0 oder einmal das Muster
Muster\\+	eine Folge von 1 oder mehr Mustern
Muster\\{*n*\\}	eine Folge von *n* Mustern (fehlt in man-Pages)
Muster\\{*n*,*m*\\}	eine Folge von *n* bis *m* Mustern (fehlt in man-Pages)
\\(*Ausdruck*\\)	faßt Ausdruck zusammen; markiert Text für *N*
N	das *N*-te mit Klammern eingeschlossene Muster
Muster\\\|*Muster*	entweder das erste oder das zweite Muster

Zwischenraummarkierung

^	Zeilenanfang
$	Zeilenende
\\<	Wortanfang
\\>	Wortende

2.5 Online-Aufgaben

Finden einer Lösung **interaktiv** mit den Teilnehmern, am besten mit **Laptop/Beamer**.

2.5.1 Aufgabe: Erkennen von Email-Adressen

Es soll ein regulärer Ausdruck geschrieben werden, der möglichst genau auf die folgenden Email-Adressen paßt:

```
brederek@tzi.de
brederek@informatik.uni-bremen.de
brederek@gemini.informatik.uni-bremen.de
brederek@saturn.informatik.uni-bremen.de
root@saturn.informatik.uni-bremen.de
jan.bredereke@web.de
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 10

Es sollen aus einer Datei genau diese Email-Adressen ausgegeben werden.

2.5.2 Aufgabe: Extraktion von include-Dateinamen aus einer LaTeX-Quelle

Eine LaTeX-Quelldatei enthält Zeilen der Art

```
\verbatimtabinput{sed/passwd_names.sh}
```

Es soll eine Liste der Dateinamen in diesen Kommandos ausgegeben werden.

Mit „%“ auskommentierte LaTeX-Befehle sollen dabei ignoriert werden.

2.5.3 Aufgabe: Extraktion aller derzeit aktiven Benutzer aus „w“-Ausgabe

Ausgabe des w-Kommandos:

```
 12:24pm  up 56 day(s),  2:42,  10 users,  load average: 0.00, 0.02, 0.03
User     tty           login@  idle   JCPU   PCPU  what
alone    smb/0        12:24pm                      /usr/local/lib/samba/smbd -D
nobody   smb/1        12:24pm                      /usr/local/lib/samba/smbd -D
roefer   smb/2        12:14pm                      /usr/local/lib/samba/smbd -D
root     pts/1         1Aug01 2days  20:31      2  /usr/local/bin/bash -login
roefer   smb/4        12:18pm                      /usr/local/lib/samba/smbd -D
mawe     pts/2         1Aug01    14   6:06     25  contool
root     pts/3         1Aug01    13   9:29      5  /usr/local/bin/bash -login
cxl      pts/4        Tue 5pm  1:23     35     35  xemacs
cxl      pts/5        Thu 9am 5days  10:22         -bash
brederek pts/6        11:19am                      w
```

Gewünscht:

- Liste von Benutzernamen
- keine Benutzer mit „idle“-Zeit
- keine Samba-Daemons (d.h. nur Benutzer an Pseudo-TTYs)
- keine Kopfzeile des w-Kommandos

Kurs Unix-Tools, WS 01/02, Jan Bredereke 11

2.6 Hausaufgaben

2.6.1 Schützen von Sonderzeichen in der Vacation-Text-Ersetzung

Problem: Wenn der Betreff selbst Schrägstriche enthält, dann wird das `sed`-Kommando falsch interpretiert. Lösungsidee: Füge vor alle Schrägstriche (und vor alle Backslashes) im Betreff Backslashes ein.

Aufgabe: Schreibe ein `sed`-Kommando, das seine Standard-Eingabe entsprechend transformiert. Nimm ggf. die man-Pages zu `sed` zu Hilfe.

2.7 Kombinieren von Tools: Bearbeiten von ganzen Dateibäumen mit `find`

Die Grundkommandos von `find`

Syntax
find *Verzeichnis,...* [-*Option...*] [-*Test...*] [-*Aktion...*]

Test
-name *Muster* Dateiname paßt zu Muster (mit „*", „?", „[...]" wie bei Shell)
(viele Tests zu Dateialter, Dateiattributen, Dateityp, ...)

Aktion
-print druck Dateinamen mit Pfad
-exec *Kommando* **;** führt Kommando aus
(etliche Varianten dieser Kommandos)

Option
(diverse zur Art der Baumsuche, ...)

Anmerkungen
- Im *Kommando* von **-exec** wird der Ausdruck „**{}**" durch den Dateinamen mit Pfad ersetzt.
- Bei Aufruf von der Shell müssen „**{}**" und „**;**" geeignet in Anführungszeichen gesetzt werden.
- Ist keine Aktion angegeben, ist **-print** Default.
- Tests und Aktionen können mit Klammern und boolschen Operatoren zu komplexen Suchausdrücken kombiniert werden.

Kurs Unix-Tools, WS 01/02, Jan Bredereke 12

2.7.1 Aufgabe: Liste aller Dateien unter CVS

Man kann Dateien mit dem **Versionsverwaltungssystem CVS** verwalten, sogar **ganze Bäume** von Dateien. Aber **nicht alle Dateien** werden dabei unter CVS **verwaltet**. Zum Beispiel Objektdateien werden ausgelassen.

CVS legt in jedem verwalteten Verzeichnis ein **Unterverzeichnis** `CVS` an, und darin u.a. eine **Datei** `CVS/Entries`, in der alle verwalteten Dateien zusammen mit Zusatzinformationen aufgeführt sind.

Beispiel einer Datei `CVS/Entries`

```
D/WWW////
/kursaufbau.txt/1.5/Mon Sep 17 16:26:56 2001//
D/sed////
/.cvsignore/1.7/Mon Sep 24 10:26:33 2001//
D/perl////
/skriptnotizen.mk/1.7/Thu Sep 27 08:02:23 2001/-kb/
/skriptnotizen.sty/1.7/Tue Oct  2 12:52:00 2001//
/folien.tex/1.15/Thu Oct  4 19:24:36 2001//
/skriptnotizen.tex/1.22/Thu Oct  4 19:24:40 2001//
/vorstellung-folien.dvi/1.1/Tue Oct  9 15:03:09 2001/-kb/
/vorstellung-folien.tex/1.1/Tue Oct  9 15:03:07 2001/-kb/
```

Kurs Unix-Tools, WS 01/02, Jan Bredereke 13

Es soll eine **Liste aller Dateinamen** ausgegeben werden, die mit CVS verwaltet werden. Dies soll **für alle Unterverzeichnisse** eines Verzeichnisses geschehen. Der Einfachheit halber soll nur der Dateiname **ohne den ganzen Pfad** ausgegeben werden.