UNIX Werkzeuge (Aufbaukurs)

Christoph Stockmayer

05.04.2001

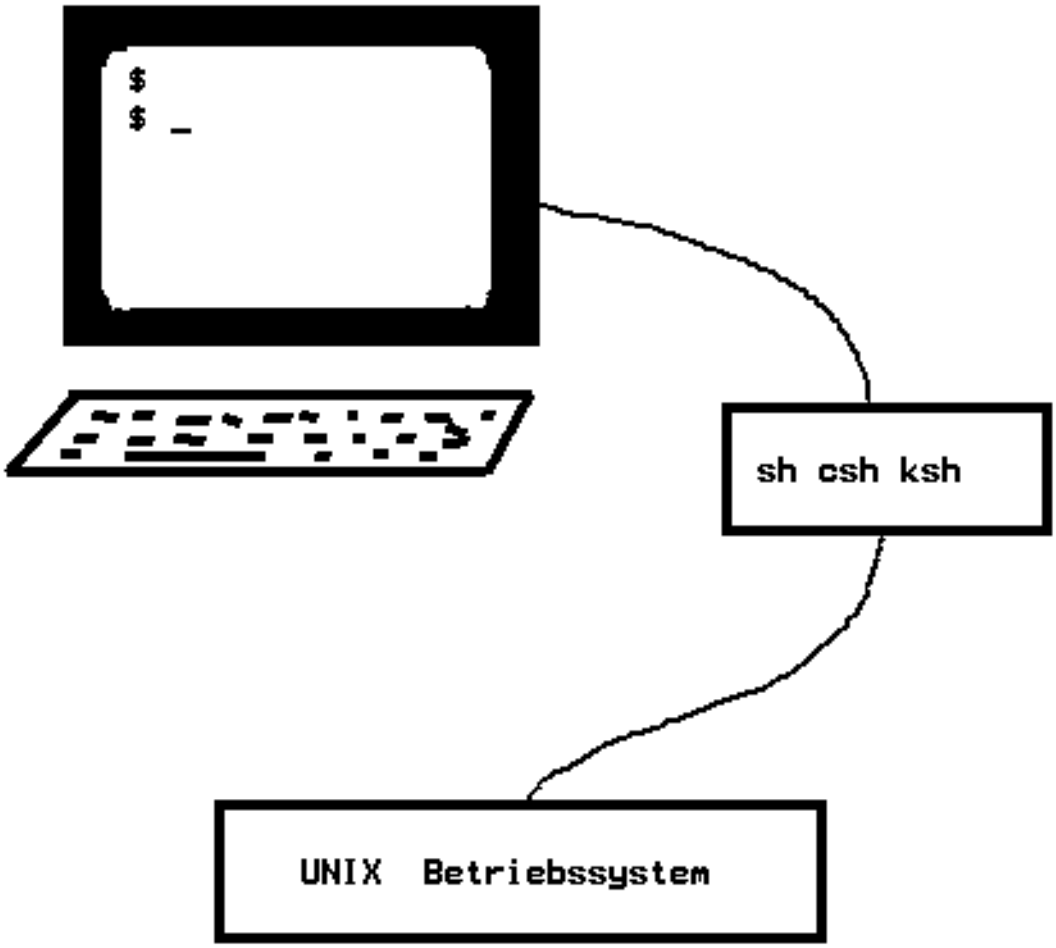
Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Inhalt

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Shell



- Lebensdauer der Shell:

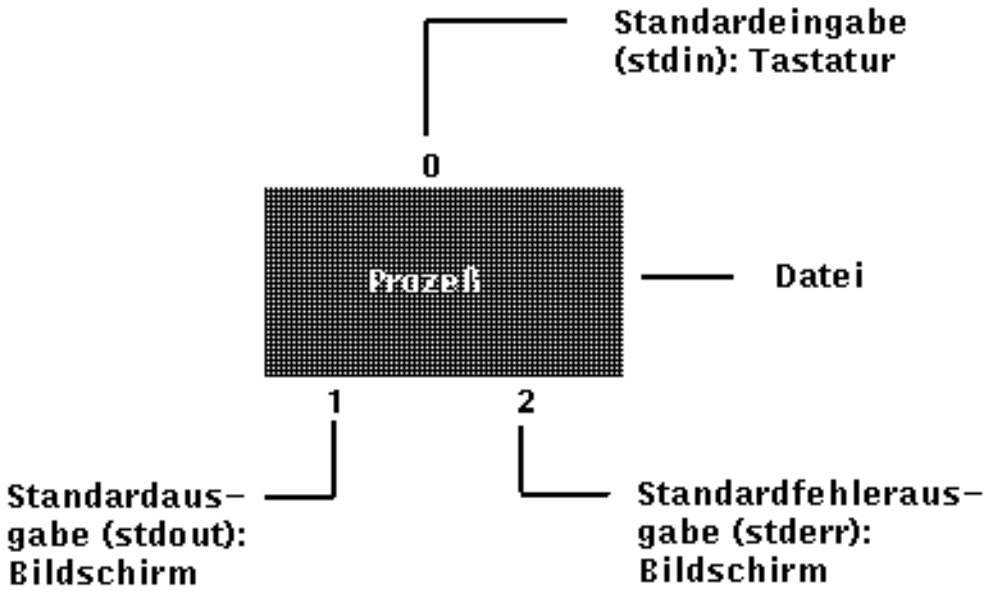
  vom login bis logout (EOF <CTRL-d>, exit)

- Interpretiert Kommandos und startet Prozesse

- Auch Subshells möglich: sh ksh csh

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Shell: Dateikonzept



Beispiel:

```
date

date > /dev/rmt/c0s0

date > file

date >> file

cat xxx 2> err
cat err

write usr1 < textdat

                        # Here-Dokument
write usr2 <<!
hallo
halli
!
```

Shell: Kommandos

- Shell-intern
- Programmaufrufe

Syntax: ls -l /etc/termcap > file &
| more

Kommando-name **Option** **Datei** **Umlenken oder Pipen**

Blank **Blank** **Hintergrund**

Beispiel:

```
sort -r file > filesort 2>> err &
sort file   2>> err     | lp &
mail usr3    < brief
exit
echo hallihallo
cd zieldir
pwd
```

Erzeugen von Dateinamen:

*	**beliebige Zeichenfolge (auch leer)**
?	**Einzelzeichen**
[]	**Auswahl von Zeichen (1 Zeichen)**
	[abcd]
	[a-d] Bereich
	[!a-d] Alle auβer ...

Substitution vor Ausführung

Beispiel:

```
prog prog.c prog.lst prog.o prog.out prog.s

1)  ls ???? p*[ct] *g*
2)  echo  prog*
3)  echo  prog.[co]
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

weitere Substitution

Kommandosubstitution ‘

Kommandos in rückwärtigen Anführungszeichen werden zuvor ausgeführt.

Beispiel:

```
more ‘cat listfile‘

PFAD=‘pwd‘
```

Funktionen

Definition von Kommandofolgen innerhalb der Shell:

```
f() {
    kom1
    kom2
    ...
}
```

```
Aufruf: f
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Shell: Variablen

* Stringvariablen

```
mark=/home/schul/usr1
```

* Variablensubstitution

```
mv  dat1  $mark
```

```
b=/home/usr1/test
ls > ${b}file1
```

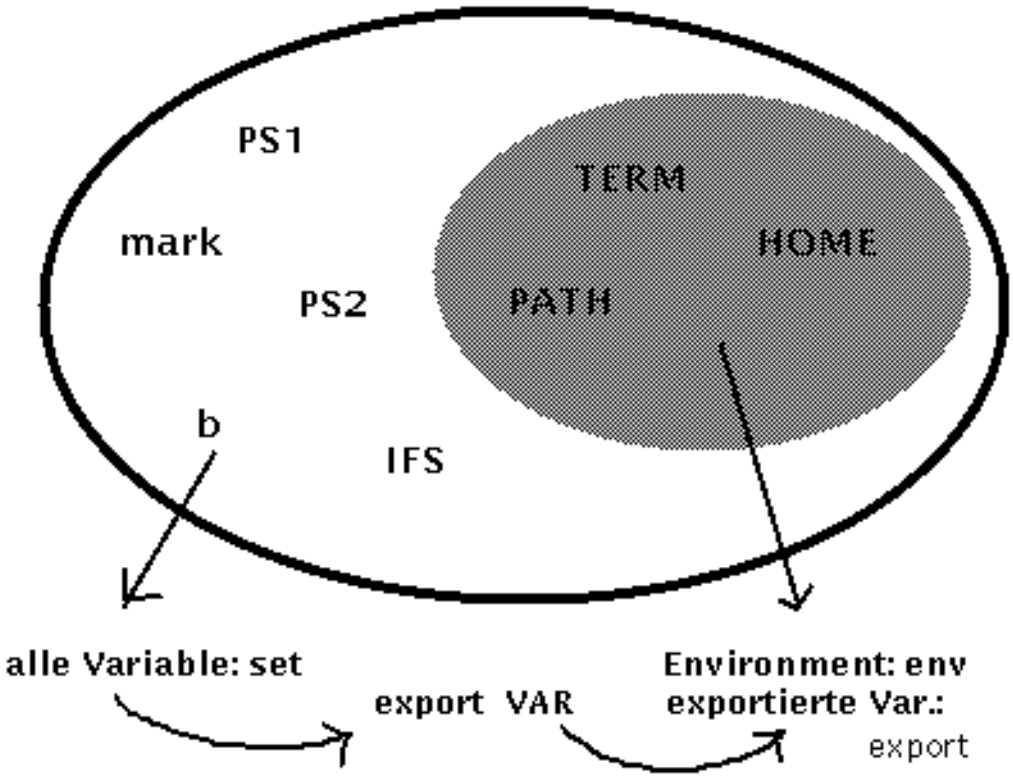
* Verhindern der Expandierung

"$b*?"	Verhindert Dateinamengenerierung
’$b*?’	Verhindert Dateinamengenerierung und Variablensubstitution
\$b	Entbindet nachfolgendes Zeichen von Sonderfunktion

* vordefinierte Variablen (set):

PATH:	Suchpfad	TERM:	Terminaltyp
PS1 :	1. Prompt	PS2 :	2. Prompt
HOME:	Heimatverzeichnis	IFS :	Trennzeichen

Shell: Umgebung



Das Variablen-set ist nur für die aktuelle Shell gültig.

Das Environment wird an neue Prozesse weitergereicht.

Löschen von Variablen mit **`unset VAR`**.

Shell: Prozeduren

* shell (sh) ist auch Kommando, Input von File

  Beispiel: Datei **com**:

```
#!/bin/sh
pwd
ls -l
cat $1
```

* Aufruf einer Datei, die Kommandos enthält:

sh file arg1 arg2 ...	**sh com file**
file arg1 arg2 ...	wenn x für execute gesetzt **com file**
. file	keine Subshell (Variablen!)
exec file	aktuelle Shell wird überlagert

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Argumente

* Übergabe von Argumenten an shell-Skripts:

Beim Aufruf eines Kommandos wird dem neuen Prozeß
das Environment zusammen mit den Argumenten aus
der Befehlszeile übergeben:

$0: Name der Kommandodatei
$1: 1. Argument arg1
$2: 2. Argument arg2
.
$9: 9. Argument arg9
$*: alle Argumente
$@: -"-

$#: Anzahl der Argumente
$$: PID des Prozesses
$!: PID letzter Hintergrundproz.
$?: Exitcode letztes Kommando
$-: Shell-Optionen

Shell: Traps

Ignorieren von Unterbrechungssignalen:

```
trap '''' SignalNr
```

Ausführen von Kommandos bei Erreichen von Signalen:

```
trap ''Kommando'' SignalNr
```

Zurückstellen der Signale:

```
trap SignalNr
```

Beispiel:

```
trap "rm tmp; exit 1" 2 3
```

Shell: Kontrollstrukturen FOR

```
for VAR  < in  L1  L2 .... >
do
    Kommandos
done
```

Solange Listenelemente L1, L2, ... da sind, werden die Kommandos ausgeführt. Jedes Listenelement wandert in die Variable VAR.
Falls <> fehlt: Argumente des Skript-Aufrufs.

Beispiel:

```
for i
do
      grep $i adressen
done
```

```
for i in *.c
do
      vi $i
done
```

```
#       Kommentar
```

Shell: Kontrollstrukturen CASE

```
case STRING in
    MUSTER1) Kommandos ;;
    MUSTER2) Kommandos ;;
    . . . . .
esac
```

Beispiel:

```
File2 an File1 anhängen (append):
"append file"
"append file1 file2"

case $# in
1) cat >> $1    ;;
2) cat $2 >> $1 ;;
*) echo "usage: append to [from]" ;;
esac
```

Im Muster sind die Metazeichen der Shell erlaubt, zusätzlich auch der OR-Operator |:

```
-*)                          ..... ;;
MUSTER1 | MUSTER2 )    ..... ;;
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Shell: test - Kommando

Strings

```
test s          ist true, falls   s nicht leer ist
test s1 = s2    ist true, falls   s1 gleich s2
test s1 != s2   ist true, falls   s1 ungleich s2
```

Zahlen abfragen:

```
test z1 -eq z2 ist true, falls   z1 gleich z2
        -ge   -gt   -le   -lt   -ne
```

Fileattribute:

```
test -f file    ist true, falls  file ist File
test -d file    ist true, falls  file ist Verzeichn.
test -r file    ist true, falls  file lesbar
     -w   -x   -s   -c    ....
```

true entspricht dem Exit-Code 0

false entspricht dem Exit-Code <>0

test kann ersetzt werden durch [.....]

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Shell: Kontrollstrukturen IF

```
   if Kommandos
   then
      Kommandos
< elif Kommandos
   then
      Kommandos          >
< else
      Kommandos          >
   fi
```

Wenn der Exit-Code des 1. Kommandos wahr ist, werden die then-Kommandos gestartet. Im anderen Fall wird der Exit-Code weiterer Kommandos abgefragt und falls wahr, diese then-Kommandos ausgeführt. Ansonsten werden die else-Kommandos gestartet, falls vorhanden.
< ... > Bereiche sind optional.

Beispiel:

```
if true
then
     echo "jawohl"
else
     echo "nie"
fi
```

Shell: Kontrollstrukturen WHILE

```
while Kommandos
do
    Kommandos
done
```

Solange der Exit-Code der while-Kommandos wahr ist, wird die Schleife weitergeführt. Ist er falsch, bricht die Schleife ab.

Beispiel:

```
while test $1
do
     ....
     shift      # Argumentenliste um 1 schieben
done


while  [ "$YN" != YES ]
do
     read YN    # Einlesen von stdin nach Variable YN
done


until test -f file
do sleep 300 ; done  # 300 sec schlafen
```

Shell: Kontrollstrukturen

break	**Abbrechen der Schleife**
continue	**Weit. Schleifendurchgang**
exit xxx	**Ende der Shell, xxx ist Exit - Code**

Beispiel:

```
for i
do
    if test -d $i
    then
       break
    elif test -f $i
       continue
    fi
    exit 1
done
exit 0
```

Shell: Kontrollstrukturen

Beispiel:

```
Rekursives Durchlaufen eines Directory-
Baumes und Listen des Inhalts aller Verzeichnisse.

Aufruf:   dr start-verzeichnis


cd $1

pwd

ls

for i in *

do

     if [ -d $i ]

     then

           $HOME/dr $i       # oder PATH-Variable

     fi

done
```

Shell: Programmtest

```
sh OPT name
```

-x Ausgabe des auszuführenden Kommandos (nach Parameterersetzung).

-v Ausgabe der gelesenen Kommandozeile

-n keine Ausführung

set -x set +x für die aktuelle Shell

Auswertung

1. Parameter- und Variablensubstitution ($)
2. Kommando-Substitution (‘)
3. Blank-Interpretation
4. Dateinamengenerierung (*,?,[...])"

Beispiel:

```
#!/bin/sh
# Lineprinter-Spooler
# Aufruf $0 -[npc] files ...

trap "rm -f /usr/spool/lprt/*.$$ /usr/spool/lpd/lock;
      exit" 2
TTY=/dev/lp
SPOOL=/usr/spool/lprt/lprt.
LOCK=/usr/spool/lpd/lock
export TTY SPOOL LOCK
DRMODE=$1

case $DRMODE in
-n)     shift; nroff $* | math > ${SPOOL}$$ ;;
-p)     shift; pr -o10 -l72 $* | math > ${SPOOL}$$ ;;
-c)     shift; cat $* | math > ${SPOOL}$$ ;;
*)      cat $* > ${SPOOL}$$
esac

(       while [ -f $LOCK ]
        do
                sleep 5
        done
        > $LOCK
        cat <  ${SPOOL}$$ > $TTY
        rm -f  ${SPOOL}$$
        rm -f  $LOCK
)&
!
```

Kommandointerpreter csh / ksh

Leistungsumfang

- Möglichkeiten der Bourne-Shell
- Kommandohistorie
- Aliasmechanismus
- erweiterte Namensexpandierung
- erweiterte Prozesskontrolle
- Variablen auch als logische und numerische Typen verwendbar
- Schutzmechanismus gegen Überschreiben
- mehr Kontrollstrukturen (bei csh: C-like)
- weitere automatisch aufgerufene Dateien

Aufruf und Beenden der csh / ksh

Start:

- csh <Optionen> <Argumente>
- ksh <Optionen> <Argumente>
- vorgegeben in /etc/passwd

Automatisches Ausführen der Dateien:

- $home/.login (bei login-csh)
- $home/.cshrc (bei jeder csh)
- $HOME/.kshrc (bei ksh mit gesetzter ENV-Variablen)

Prompt:

- % für normale Benutzer (csh)
- $ für normale Benutzer (ksh)
  PWD ist verfügbar
- # für Superuser (root)
  (gesetzt in Variable prompt/PS1)
- > 2. Prompt (wie sh)

Ende: - EOF (CTRL-d)

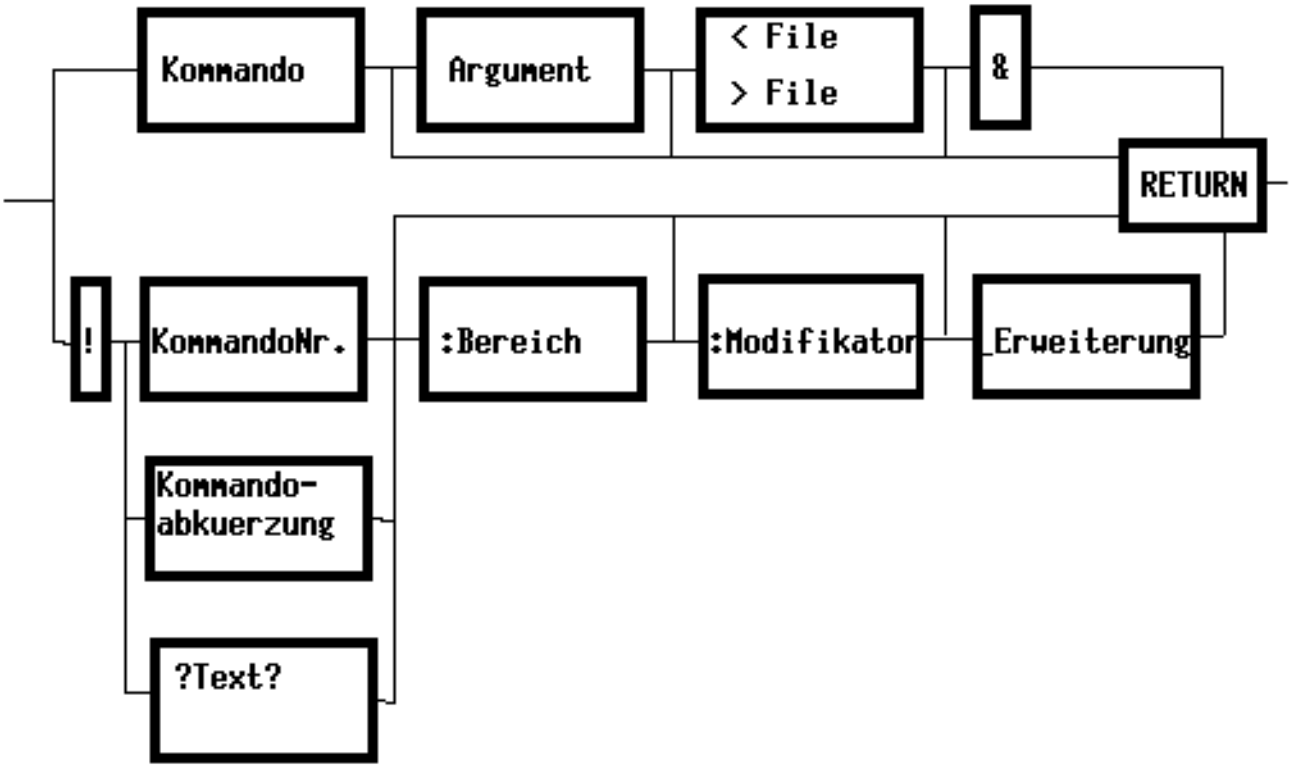
- exit
- login (nur bei login-csh)

Automatisches Ausführen der Datei:

- $home/.logout (csh)

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Kommandosyntax csh



- oberer Teil wie sh
- unterer Teil mit History-Mechanismus

Einschalten des History-Mechanismus:

set history=[Anzahl der Kommandos]

Einschalten der Kommandonummerierung:

set prompt="! <Prompt-Zeichen>"

Beispiel:

```
set history=20
set prompt="! --> "
```

Auflisten des History-Speichers:

history

Kommandowiederholung (s. Graphik):

!!	letztes Kommando wiederholen
!12	Kommando 12 wird wiederholt
!his	Kommando, welches mit "his" beginnt, wird wiederholt z.B. history
!?to?	Kommando, welches "to" enthält, wird wiederholt z.B. history

<> : Individuelle Eingabe

Bereich:

<n>	n-te Wort (bei 0 beginnend)
^	1. Wort
$	letztes Wort
<n1>-<n2>	n1-tes bis n2-tes Wort
-<n>	bis zum n-ten Wort
<n>-	ab dem n-ten Wort

Beispiel:

```
!29:0           Kommando 29, nur Kommandoname

!his:1-3        Kommando, das mit "his" beginnt,
                Worte 1 bis 3

!?ls?:-2        Kommando, das "ls" enthält,
                bis zum 2. Wort

!35:1-          Kommando 35, ab 1. Wort
```

Modifikator:

s/alt/neu/	Substituiere "alt" gegen "neu", statt / kann jedes Zeichen stehen auch: ^alt^neu
g	global (steht vor dem Modifikator) gs/../../
&	letzte Substitution noch einmal
p	neues Kommando nur anzeigen
h	extrahiert Pfadnamen

Beispiel:

```
!!:s/-l/-a/:p

!20:gs+a+b+:h

^-l^-a
```

Kommandosyntax ksh

Einschalten des History-Mechanismus:

```
EDITOR=vi   # oder emacs
set -o vi       # oder set -o emacs
```

Einschalten der Kommandonummerierung und Pfadangabe:

PS1="! \$PWD <Prompt-Zeichen>"

Beispiel:

```
PS1="! \${PWD#$HOME/} -->"
```

Auflisten des History-Speichers:

history

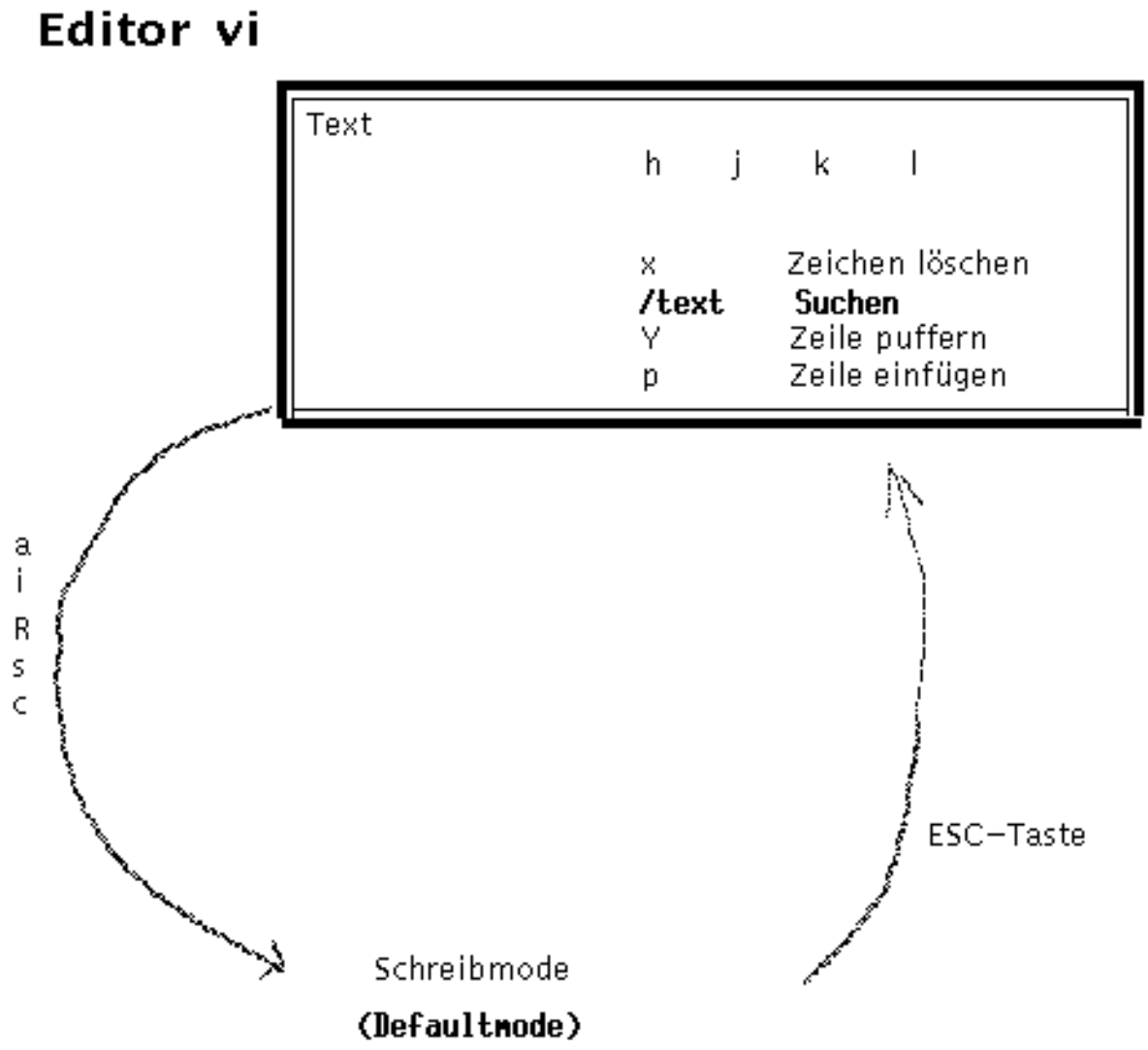
Kommandowiederholung wie vi bzw. emacs (s. Graphik):

r <Nr>

Editorbefehle

v

Editorbefehle ksh



csh/ksh: Prozesskontrolle

Anlegen eines Hintergrundprozesses:

```
... &
```

Rückmeldung: Auftrags(Job)nummer und PID

Information des Benutzers über Hintergrundprozesse:

- meldet Zustand des/der Prozesse
- meldet, wenn Prozess Eingabe von stdin benötigt, gleichzeitig wird Prozeβ gestoppt
- "jobs" gibt Liste der Hintergrundprozesse aus
- Info erst namch dem nächsten Prompt, es sei denn, Variable "notify" ist gesetzt

Terminieren von Prozessen:

- kill <-SignalNr> %Auftrags(Job)nummer
- kill <-SignalNr> PID

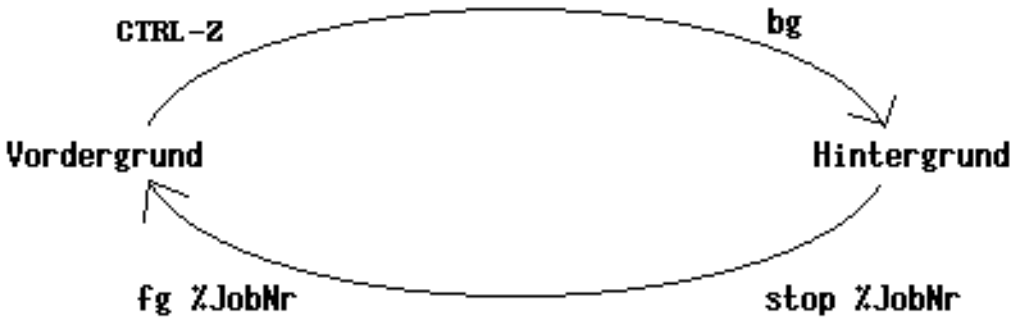
Stoppen von Prozessen:

- <CTRL-z> oder <CTRL-b> bei aktuellem Prozeβ (stty -a)
- stop %Auftragsnummer oder

  stop PID bei Hintergrundprozessen

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Wechsel Vordergrund <--> Hintergrund



- nur für gestoppte Prozesse möglich
- nachträgliches Inhintergrundstellen
- Tastatureingaben für Hintergrundprozesse

Kommandodateien, Optionen

```
csh  <Optionen>  Kommandodatei  <Arg>
ksh  <Optionen>  Kommandodatei  <Arg>
```

-n Test ohne Ausführung

-v Protokoll (oder "set verbose")

-f .cshrc wird ignoriert (csh)

oder nach chmod +x Kommandodatei:

```
Kommandodatei <Arg>
```

- die Kommandodatei wird von der csh ausgeführt,
  wenn die erste Zeile mit dem Kommentar #!/bin/csh beginnt.
- die Kommandodatei wird von der ksh ausgeführt,
  wenn die erste Zeile mit dem Kommentar #!/bin/ksh beginnt.

Beispiel:

```
#!/bin/csh
# Test fuer ein csh-Skript
find / -name csh -print &          File "cshcom"
jobs
```

Aliasmechanismus

Möglichkeit der Kurzform für eine Kommandofolge

Definieren von alias csh:

```
alias  Kuerzel  Kommando
```

Beispiel:

```
% alias such find / -name
% such csh -print
```

Definieren von alias ksh:

```
alias  Kuerzel=Kommando
```

Beispiel:

```
$ alias such="find / -name"
$ such csh -print
```

Liste der Ersetzungen: **alias <Kürzel>**

Undefiniert machen: **unalias Kürzel**

Namensexpandierung

Metazeichen stehen für Pfad- und Dateinamen:

*	steht fuer beliebige Zeichen auch keines
?	steht fuer **ein** einzelnes Zeichen
[ ]	steht fuer ein Zeichen aus der angegeben Auswahl: Aufzaehlung: abc Bereich: a-c Negation: ! (nur ksh)
~	steht fuer den Pfad der Homedirectory
~usr	mit Benutzernamen dahinter fuer die Homedirectory des jeweiligen Benutzers
{ , }	steht fuer eine Auswahl mehrerer Worte, Trennzeichen ist Komma (nur csh)

Beispiel:

```
cat *.c
cat [a-f]*.c
cat [gh]?.c
ls ~uucp
ls ~/test
ls {~,~root}/bin
```

csh: Variablen

- Alle Variablen sind vom Typ String
- Variablennamen maximal 20 Zeichen lang
- expr-Kommando bereits eingebaut

Setzen von lokalen Variablen:

```
set Var = Inhalt
```

Beispiel:

```
set zahl=1
set prompt="--> "
set verbose
```

csh: Variablen

Liste der lokalen Variablen:

```
set
```

Löschen von Variablen:

```
unset VAR
```

Inhalt der Variablen:

```
$VAR
```

Beispiel:

```
set

unset zahl

echo $prompt
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

csh: Variablen

Setzen der globalen Variablen (Environment):

```
setenv Var Inhalt
unsetenv Var
env
```

Beispiel:

```
setenv TERM vt100

unsetenv zahl
```

csh: Variablen

Rechnen mit (lokalen) Variablen:

```
@ Var++
@ Var--
@ Var Op Ausdruck
```

- In-/Dekrement
- Operatoren: += -= *= /= %=

Beispiel:

```
% @ a++
% @ a*=10
```

ksh: Variablen

Rechnen mit (lokalen) Variablen:

```
let i=i+1
(( i=i+1 ))
(( i=(i+48)/3 ))
```

- Grundrechenarten und Klammerung
- siehe auch **`expr`** und **`bc`**

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

csh: vordefinierte Variable

$argv	Feld mit Aufrufparameter (Auch $0 ... $n möglich)
$child	PID des letzten Hintergrundprozesses
$cwd	Pfadname des aktuellen Verzeichnisses
$history	Größe des Historyspeichers
$savehist	Größe der auf Platte zu merkenden Kommandos
$home	Homeverzeichnis
$path	Suchpfad
$prompt	1. Prompt (! im Prompt setzt Kommandonummer)
$status	Exit-Code des letzten Kommandos
$<	Zeile von stdin lesen
$$	PID der csh

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

ksh: vordefinierte Variable

$0 ... $xx	Argumente vom Aufruf
$PWD	Pfadname des aktuellen Verzeichnisses
$HOME	Homeverzeichnis
$PATH	Suchpfad
$PS1	1. Prompt (! im Prompt setzt Kommandonummer)
$?	Exit-Code des letzten Kommandos
$TMOUT	Time-Out in sec
$RANDOM	Zufallszahl
$EDITOR	Editor für Kommandozeile (ed, vi, emacs)
$ENV	Startup-Datei

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

csh/ksh: Steuervariablen

set echo	Kommando wird angezeigt (csh)
set ignoreeof	Terminieren nur durch "exit"
set noclobber	existierende Datei wird beim Umlenken nicht überschrieben
set notify	Zustandsmeldungen kommen sofort (csh)
set time	verbrauchte Zeit anzeigen (csh)
set verbose	erzeugtes Kommando anzeigen

Bei ksh ist "set -o VAR" zu verwenden

csh/ksh: Interne Kommandos

exit <Ausdruck>	Terminierung der Shell und Rücklieferung des Ausdrucks als Exitcode
exec Kommando	csh/ksh wird durch Kommando überlagert
shift	Positionsparameter werden um eine Position nach vorn geschoben
wait	Warten auf Beendigung eines Hintergrundprozesses

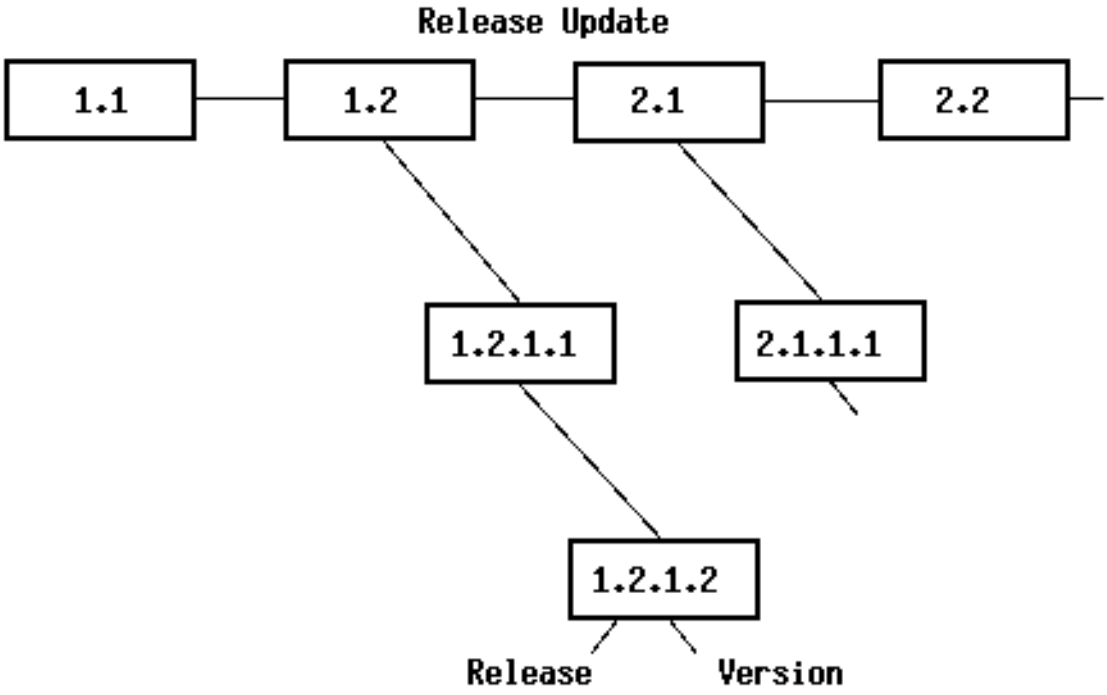
csh/ksh: Ein-/Ausgabeumlenkung

<	datei	"datei" wird statt "stdin" gelesen
<<	marke	nachfolgender Text wird bis "marke" statt "stdin" gelesen
>	datei	"stdout" geht an "datei"
>!	datei	csh: es findet keine Bestandsprüfung statt
>|	datei	ksh (bei gesetztem $noclobber)
>&	datei	"stdout" und "stderr" gehen an "datei" (csh)
> datei **2>&1**		"stdout" und "stderr" gehen an "datei" (ksh)
>>	datei	Anhängen von "stdout" an "datei"
|		"stdout" geht in eine Pipe
|&		"stdout" und "stderr" gehen in eine Pipe (csh)
2>&1 |		"stdout" und "stderr" gehen in eine Pipe (ksh)

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

SCCS: Quellcodeverwaltung

- Speichern von Quellcode
- Versions- und Release-Kontrolle
- Abruf bestimmter Versionen
- Überwachung der Zugriffsrechte und der Modifikation
- Änderungslogbuch



SCCS: Anlegen, Aendern SCCS - Datei

- legt SCCS - Datei an
- ändert Parameter
- bei einigen UNIX-Versionen vorstellen von "sccs"

admin Option SCCS-Datei

s.XXXX

-n Neuanlegen

-iFile Source-File für SCCS

-rRel Release-Nummer

-aLogin Namen der Mitarbeiter,die ändern dürfen

-eLogin Löschen der Namen

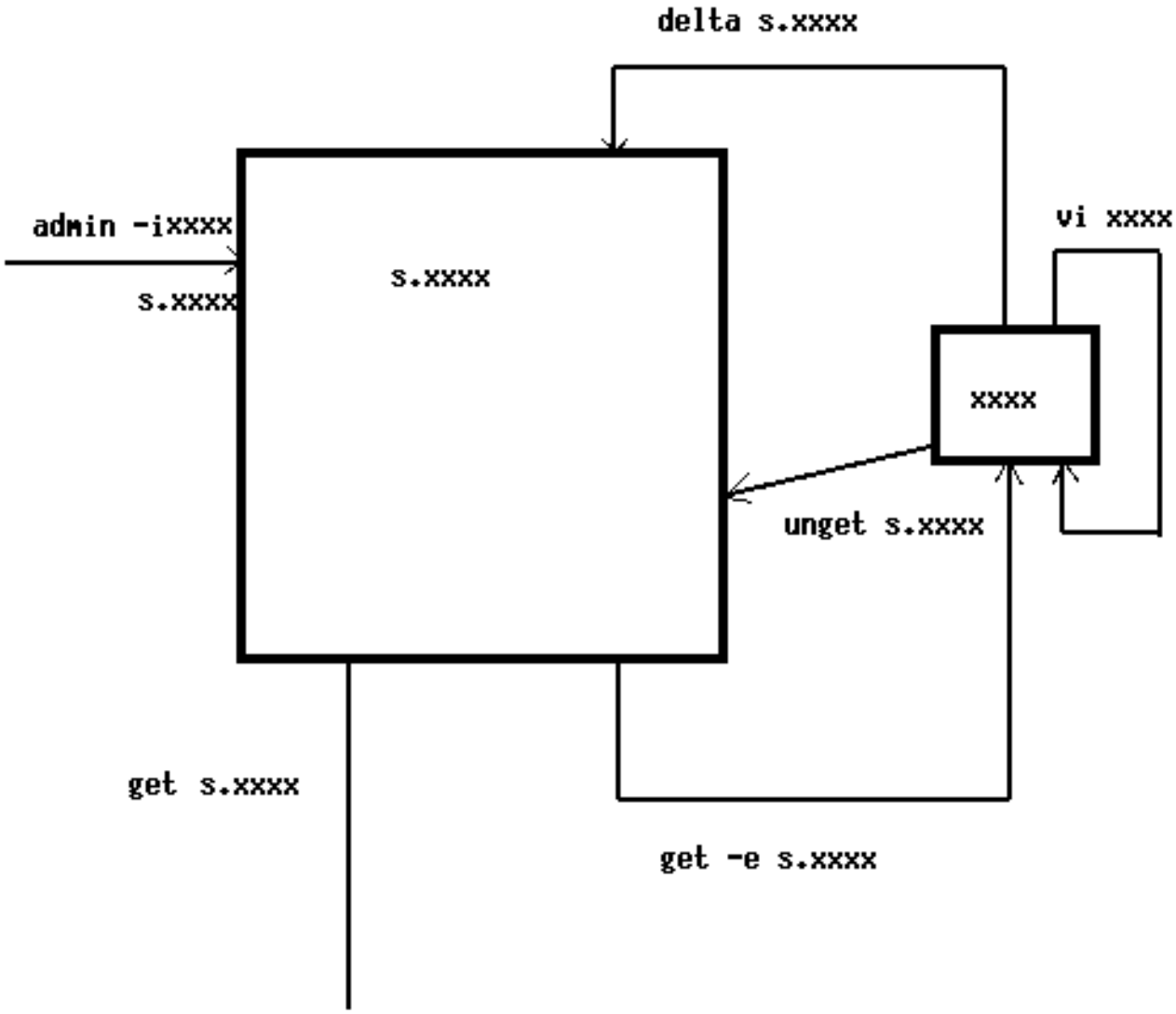
-yKom Kommentar

Beispiel:

```
admin   -icfile.c  s.cfile.c
rm cfile.c
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

SCCS: Anlegen, Aendern SCCS - Datei



SCCS: Entnahme von Releases

- bestimmte Version entnehmen (read only)
- neue Version / Release erzeugen

get	**Option**	**SCCS-Datei**

ohne	read-only-File der letzten Version wird entnommen
-e	neue Version / Release wird erzeugt
-rRel	bestimmtes Release
-lLogbuch -lp	
-m	Version und Zeilenangaben
-p	Version nach stdout

Beispiel:

```
get   -e   -r2.1  s.cfile.c
```

SCCS: Zurueckstellen

- Zurückstellen einer neu kreierten Version

```
unget   SCCS-Datei
```

Neue Version SCCS uebergeben

- analysieren der Änderungen
- Änderungen speichern

```
delta   Option   SCCS - Datei
```

-p Änderungen werden an stdout angezeigt

Beispiel:

```
unget  s.cfile.c
delta  -p  s.cfile.c
```

SCCS: weitere Kommandos

cdc Option SCCS-Datei

-rRel

-y Kommentar

Ändert Kommentar

comb Option SCCS-Datei

bis -pRel zusammenfassen mehrerer Versionen

Zusammenfassen

SCCS: weitere Kommandos

help arg	Fehlerkommentar zu Arg
sact SCCS-Datei	Information über aktuele Änderungen (nach get -e ...)
sccsdiff Option SCCS-Datei -rRel1 -rRel2	Vergleich zweier Versionen
val Option SCCS-Datei -rRel	Konsistenzprüfung
what File	Identifikation von Source-, Objekt- und Binärfiles (%Z%)
prs Option SCCS-Datei -rRel -d:UN:	Infos über SCCS-Datei zeige alle autorisierten Benutzer
rmdel Option SCCS-Datei -rRel	Löschen der letzten Version

SCCS: Schluesselworte

Schlüsselworte können in der Quelldatei integriert werden,
sie werden von der get-Funktion mit aktuellen Daten versehen.

%I% SCCS - ID
%R% Release
%L% Version
%D Datum
%T% Zeit
%F% SCCS-File-Name
%Z% zur Erkennung durch what (@(#))

Beispiel:

```
/* SCCS-ID : %I%  vom  %D% */
static char SCCS_ID[] = "%Z% SCCS-ID : %I% ";
```

Programmgenerator make

- Erzeugung eines ablauffähigen Programms aus vielen Teilen (Files, Bibliotheken, SCCS-Dateien)
- Bei Änderung eines Teils kann make mit Minimalaufwand neues Programm generieren
- Abhängigkeiten eingebaut oder durch Beschreibung

```
make   <Option>   <Ziel>
```

zu erzeugendes Programm
(Standard: erstes Programm)

-f File Beschreiberfile
(sonst: makefile
Makefile
s.makefile
s.Makefile)

Optionen:

-d fuer debugging-Zwecke
-p Makrodefinitionen
-n Pseudobearbeitungen
-r Rückgängigmachen der eingebauten Regeln
-s Keine Protokollierung
-i ignorieren von fehlerhaften Kommandos
-t Zeitaktualisierung der Zieldatei
-q Abfrage auf "up-to-date" (Exitcode 0 wahr)

make: Aufbau [mM]akefile

# Kommentar	Kommentar immer möglich
Var = Inhalt	Setzen von Variablen (Makros)
Ziel : Quellen	Abhängigkeiten
<TAB> Kommando zum Erzeugen des Ziels aus Quellen	
<TAB> weitere Kommandos	
....	weitere Ziele

Beispiel:

```
# Beispielmakefile
prog: file1.o  file2.o
      cc  -g  file1.o  file2.o  -o prog
file1.o: file1.c  includefile.h
      cc -c file1.c
```

make: Variablen (Makros)

- alle Environment-Variablen
- beim Aufruf mitgegebene Variablen
  (überdecken gesetzte Variablen im Skript)
- Neuformulierungen:

  Var = Inhalt

Beispiel:

```
OBJ = file1.o  file2.o  file3.o
F  = f77
CFLAGS = -g
```

- Interne Variablen:

  $@ Voller Name des Ziels

  $* Ziel ohne Endung

  $< Name der rufenden Datei

  $? Abhängigkeitsdateien, die neuer als das Ziel sind

Zugriff auf Variable:

$(name) bei Mehrzeichennamen

$x bei Einzeichennamen

make: Allgemeine Abhaengigkeiten

eingebaut:

.out .o .c .y .l .s

angebbar:

.SUFFIXES: .e1 .e2 .e3 .e4 ... .en

Kommandos zur allgemeinen Transformation:

.e2.e1:
<TAB> Kommando zur Erstellung von *.e1 aus *.e2

Beispiel:

```
.SUFFIXES: .x  .z
.z.x:
        cp $<  $*.x
```

```
$ make bsp.x
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

make: Anmerkungen

- Metazeichen der shell können verwendet werden:
  *, ?, []

- Abhängigkeiten z.T. Implementierungsabhängig
  prüfen mit "make -p"

- Eingebaute Makros anschauen:
  "make -p"

- bei Abbruch wird Ziel gelöscht

- "touch datei" erneuert Modifikationsdatum

make

Beispiel:

```
LIBS= lib.os lib.io lib.fs lib.munet lib.bmt lib.sock
NEWLIBS= lib.du
OBJS = l.o c.o linesw.o table.o name.o

all: unix

unix:   $(OBJS) $(LIBS)
        ed CURver.c < version.ed >/dev/null
        $(CC) -c $(CFLAGS) CURversion.c
        $(LD) -o /newunix -x param $(OBJS) CURver.o $(LIBS)
        @echo new unix in /newunix

aunix:
        ./get aunix
        -$(MAKE) allunix

allunix: $(OBJS) $(LIBS) lib.nosup lib.startup
        ed CURver.c < version.ed >/dev/null
        $(CC) -c $(CFLAGS) CURver.c
        $(LD) -o /aunix -x param $(OBJS) CURver.o \
                lib.nosup $(LIBS) lib.startup
        @echo new aunix in /aunix

libs:
        -cd ml; $(MAKE)
        -cd os; $(MAKE)
        -cd io; $(MAKE)
        -cd io/nosup; $(MAKE)

c.o: conf.h $(INC)/space.h $(INC)/opt.h $(INC)/init.h
l.o linesw.o table.o: conf.h

clean:
        rm -f *.o os/*.o io/*.o io/nosup/*.o ml/*.o nudnix/*.o
        rm -f fs/*/*.o startup/*.o munet/*.o bmt/*.o bmt/*.oo
        rm -f sock/*/*.o sock/*/libsock
```

Bibliotheken: ar

- mehrere (übersetzte) Dateien zusammenfassen
- Binder ld kann aus Bibliothek Module extrahieren
- dynamische Bibliotheken erzeugen mit "cc -G"

```
ar  Option  Bibliothek-Datei  Modul
```

Name der Bibl.:
xxx.a
libxxx.a

Für Binderoption
-lxxx :
libxxx.a
(in /lib, /usr/lib, /usr/ccs/lib, bzw. -L...)

Source- oder Objekt-File

-q	am Ende anhängen
-r	Ersetzen
-d	Löschen
-x	Herausholen (extrahieren)
-t	Inhaltsverzeichnis

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

ar: Ordnen und Inhaltsverzeichnis

- Beschleunigen des Zugriffs

ranlib Bibliothek-Datei

nur bis System V

Beispiel:

```
ar -q libbibl.a  *.o
ar -t libbibl.a
ar -r liblibbibl.a   file1.o  file2.o
ar -d libbibl.a   file3.o
ar -x libbibl.a   test.o
ranlib  libbibl.a
cc file.c -lbibl
```

Suchprogramm grep

- Suchen nach Textmuster und Ausgabe der gefundenen Zeile

```
grep <Option> Suchtext Datei
```

Optionen:

-c passende Zeilen zählen

-fdatei Suchtext steht in ’datei’

-i Gross- und Kleinbuchstaben gleich behandeln

-l Namen der Dateien ausgeben

-n mit Zeilennummer

-v alle Zeilen, auf die Suchtext nicht passt

Beispiel:

```
grep -n -v return *.c
vi ‘grep -l main *.c‘
```

grep: regular expressions

Suchtext kann sein:

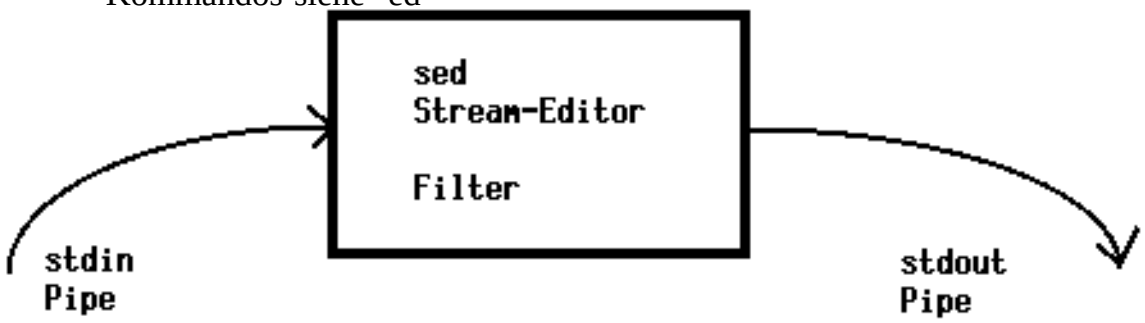
beliebiger Text	
Symbole des ’ed’:	
^	am Anfang der Zeile ’^....’
$	am Ende der Zeile ’....$’
.	beliebiges Zeichen
[ ]	Auswahl
[^]	negierte Auswahl
...*	0 bis n-mal
\{ \}	bestimmte Anzahl mal
\< \>	Wortanfang/Wortende
egrep: ...+	1 bis n-mal
... |...	ODER-Verknüpfung
fgrep: keine regulären Ausdrücke	

Beispiel:

```
grep ’Kapitel.[1-9]’ *
grep ’^[^ab].*9$’  *.c
```

Streameditor sed

- editieren des Datenstroms
- Kommandos siehe ’ed’




```
sed <Option> skript <Datei>
```

-f Datei Editieranweisungen in Datei

-n Keine Standardausgabe

skript:

d	Zeile löschen
a\	anfügen
i\	einfügen
c\	ändern
p	ausgeben
s/.../.../g	ersetzen (g: global)
r file	lesen der Datei ’file’
q	beenden

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

sed

Vor jedem Kommando muβ Zeile oder Zeilenbereich stehen!

Sonst gilt das Kommando für jede Zeile!

Beispiel:

```
$ sed 's/alt/neu/g' datei
$ sed '2a\
> xxxxx '
$ sed '2d
> 3r file'
$ sed  -f sed.txt datei
$ cat *.c | sed 's/\[100\]/[200]/g' | pr -l72 | lp
```

In den Mustern sind regular expressions und
auch Subausdrücke möglich bzw. Einsetzen des gefundenen Musters:

Beispiel:

```
sed 's/^.\(....\).*$/---\1---/'
sed 's/ [a-z][a-z]* /---&---/'
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Unterausdruecke expr

1) Rechnen:

expr 1 + 2

expr 3 * 5

auch - / und \(\)

Beispiel:

```
i=‘expr $i + 1‘
```

2) Textlängenbestimmung:

expr TEXT : ’.*’

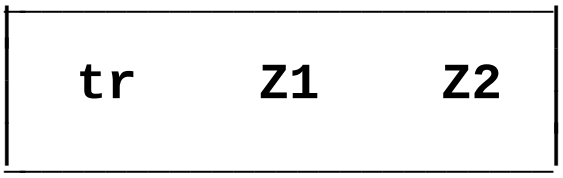
3) Textstücke extrahieren

expr TEXT : ’...\(...\)...’

Beispiel:

```
expr abcdefghijk : ’ab..\(.*f.\).*’
```

Zeichen umwandeln mit tr



Zeichen Z1 in der Standardeingabe umwandeln in Zeichen Z2

Beispiel:

```
tr a b
tr '[a-z]' '[A-Z]'
tr '\012' '\015'
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

weitere Kommandos

- cut -dDEL -f1,3,5 file	Spalten extrahieren
- basename /a/b/c.c .c	Basisname eextrahieren (c)
- dirname /a/b	Verzeichnisname extrahieren (/a)
- fold -w 80 fmt -w 80	Zeilen umbrechen
- tput	Bildschirmattribute
- diff	Filevergleich
- head -20	erste 20 Zeilen einer Datei
- tail -f	letzte Zeilen einer Datei (-f follow)

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

Listenverarbeitung mit awk

- Zeilenweise Abarbeitung
- wird bestimmtes Muster gefunden > Aktion
- zusätzlich Anfangs- und Endaktionen

```
awk   <Option>  <skript>   Datei
```

-Fc Feldtrenner = ’c’

-f file Kommandos aus ’file’

script kann Kommandos enthalten

Beispiel:

```
awk -F: ’{ print $1}’ /etc/group
nawk ’
     { print $2 $5 $7+$9 }
    ’ tabelle
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

awk: Aufbau Kommando

```
 BEGIN    { Initialisierungsanweisungen: i=0 }
 Muster   { Aktion }
/^abc/    { print $2 }
$1=="ab"{ i += 10 }
i == 10   { i = 0 }
      .....
 END      { Schluβ-Anweisungen: print i }
```

- fehlt Muster, wird Aktion für jede Zeile ausgeführt
- fehlt Aktion, wird Zeile ausgegeben
- Zeile in Felder aufgeteilt, Feldseparator steht in FS oder in Option
- Eingangsdatei in Records aufgeteilt (Zeilen)

Beispiel:

```
awk '
  BEGIN { FS = ":" }
        { print $2 }
    ' /etc/passwd

awk '
  /^a/  { print $1 }
    '
```

awk: Kommandos

Var = Ausdruck	Setzen von Variablen
print Liste	Ausgeben
printf("....", arg, ...)	Formattiertes Ausgeben (siehe cc)
if (Bedingung) Kommando <else Kommando>	if - Verzweigung (siehe cc)
exit	Ende

Beispiel:

```
BEGIN   { a = 0 }
        {
          a = a+1
          print $0
          if ( a == 3 ) exit
          else printf ("nicht zuende\n")
        }
END     { printf ("%s\n",a) }
```

awk: Kommandos

while (Bedingung) Kommando

for (Kommando; Bedingung; Kommando)
Kommando

break

continue

- Schleifen siehe cc

Beispiel:

```
BEGIN   { a = 0 }
        { while (a<10) printf("a: %d\n",a++) }
/^aaa/  { a = 0 }
/^[b-g]/{ for (b = 0; b<20; b++) print $0 }
```

Suchmuster

- in /..../ für regular expressions
- siehe ed oder grep
- in "...." für Texte

awk: Operatoren

= + - * / %	
++ --	De-/Inkrement
+= -= *= /= %=	Rechnen und Zuweisen
! || &&	logisch not, oder, und
< > <= >= == !=	Vergleich
- wie in C definiert	
~ !~	Vergleich mit regulärem Ausdruck

Funktionen

length (var)	Länge Zeichenkette
exp (var)	e var
log (var)	ln var
sqrt (var)	var
int (var)	Ganzzahl
substr(var,start,len)	Substring
rand()	Zufallszahl
srand()	Setzen Zufallsgenerator

Beispiel:

```
BEGIN   { a = 0 }
        {       a++
                print length(a) exp(a) int(10.2) }
```

Ingenieurbüro Stockmayer GmbH
36179 Bebra, Solzer Str. 4, 06622/919610
90571 Schwaig bei Nürnberg, Dreihöhenstr. 1, 0911/505241
EMail: sto@stockmayer.de, Fax: 0911/5009584

awk: Vordefinierte Variable

NF	Anzahl Felder pro Zeile
NR	Zeilennummer
FILENAME	Datei
OFS	Ausgabe - Feldtrenner
FS	Eingabe - Feldtrenner
$1 .... $n	Felder 1 bis n
$0	Zeile

Beispiel:

```
{
   printf ("%d %d\t %s\n", NR, NF, $0)
   print  $2
}
```

Literatur

UNIX System V.4 fuer Einsteiger und Fortgeschrittene (Einfuehrung/Uebersicht/Vergleich V.3-V.4)	Stapelberg	Addison
UNIX System V Rel. 4 (Uebersicht ueber V.4)	Rosen/Rosinski/Faber	Tewi
UNIX (Manualersatz)	Gulbins	Springer
Keine Angst vor UNIX (Einsteigerlektuere)	Wolfinger	VDI
Der UNIX-Werkzeugkasten (Fortgeschritten)	Kernighan/Pike	Hanser
UNIX-Shells (Shellprogrammierung)	Herold	Addison
The Kornshell (Shellprogrammierung)	Bolsky/Korn	Prentice
The C Shell (Shellprogrammierung)	Anderson/Anderson	Prentice
UNIX - Wie funktioniert das Betriebssystem (Internas)	Bach	Hanser
UNIX Systemverwaltung (Systemverwalter)	Fiedler/Hunter	Tewi
UNIX Systemsicherheit (Sicherheit)	Wood/Kochan	Tewi