4.18 Funktionen

Sind alle **Gnu-Erweiterung**.

4.18.1 Funktionen, die Text transformieren

Funktionen, die Text transformieren

`$(subst` *from,to,text*`)`	Textersetzung: `$(subst ee,EE,feet on the street)` ergibt `'fEEt on the strEEt'`
`$(patsubst` *pattern,replacement,text*`)`	Textersetzung mit Muster: `$(patsubst %.c,../bin/%.o,foo.c.c bar.c)` ergibt `'../bin/foo.c.o ../bin/bar.o'`
`$(strip` *string*`)`	Führenden, anhängenden und verdoppelten Whitespace entfernen: `$(strip a b      c    )` ergibt `'a b c'`
`$(findstring` *find,in*`)`	Sucht Teilstring in String.
`$(filter` *pattern ...,text*`)`	Nur die Worte im Text, auf die die Muster passen.
`$(filter-out` *pattern ...,text*`)`	Nur die Worte im Text, auf die die Muster *nicht* passen.
`$(sort` *list*`)`	Sortiert Liste von Worten, entfernt Doppel.

Kurs Unix-Tools, WS 01/02, Jan Bredereke 36

Statt $(patsubst ...) kann man auch **einfacher** die **Ersetzungen bei Variablenreferenzen** nehmen (s.o.)

$(strip ...) ist **gut für Stringvergleiche** in bedingten Anweisungen.

4.18.2 Funktionen für Dateinamen

Funktionen für Dateinamen

`$(dir` *names ...*`)`	Directory-Anteile der Namen in der Liste
`$(notdir` *names ...*`)`	Namen in der Liste *ohne* ihre Directory-Anteile
`$(suffix` *names ...*`)`	Liste der Suffixe der Namen
`$(basename` *names ...*`)`	Namen in der Liste *ohne* ihre Suffixe
`$(addsuffix` *suffix,names ...*`)`	Hängt den Suffix an jeden Namen in der Liste an
`$(addprefix` *prefix,names ...*`)`	Hängt den Prefix vor jeden Namen in der Liste
`$(join` *list1,list2 ...*`)`	Konkateniert die Worte der beiden Liste paarweise
`$(word` *n,text ...*`)`	Liefert das n-te Wort
`$(wordlist` *s,e,text ...*`)`	Liefert die Worte von Nummer s bis e
`$(words` *text ...*`)`	Liefert die Anzahl der Worte
`$(firstword` *names ...*`)`	Liefert das erste Wort der Liste
`$(wildcard` *pattern*`)`	Expandiert das Dateinamen-Wildcard-Muster in eine Liste von existierenden Dateinamen

Kurs Unix-Tools, WS 01/02, Jan Bredereke 37

4.18.3 Funktionen – Ganz hartes Zeugs

Funktionen – Ganz hartes Zeugs

`$(foreach` *var,list,text*`)`	Iteration
`$(if` *condition,then-part[,else-part]*`)`	Bedingte Expansion
`$(call` *variable,param,param,ldots*`)`	Aufruf einer selbstdefinierten Funktion
`$(origin` *variable*`)`	Test, ob Variable: definiert / aus Makefile / von Kommandozeile / . . .
`$(shell` *shell-anweisung*`)`	Shell-Anweisungs-Expansion (analog Backquotes in Skripten)
`$(error` *text*. . .`)`	Generiere Abbruch durch Fehler mit Meldung *text*
`$(warning` *text*. . .`)`	Druckt Warnmeldung, bricht aber nicht ab. Liefert leeren String.

Kurs Unix-Tools, WS 01/02, Jan Bredereke 38

4.19 Gnu-`make` und andere `make`s

Gnu-`make`-Spezifika, die nicht mit allen `make`s gehen

- include mehrerer Dateien auf einmal
- Ersetzungen bei Variablenreferenzen
- Muster-Regeln mit `%`
- parallele Ausführung
- Anhängen an Variablendefinitionen mit `+=`
- einfach-expandierte Variablen
- unechte Ziele mit `.PHONY`
- Funktionen
- bedingte Ausführung
- Statische Muster-Regeln
- generierte Makefiles

. . . also alles, was Spaß macht. `:-(`

Kurs Unix-Tools, WS 01/02, Jan Bredereke 39

4.20 Beispiel: LaTeX-Kompilierung der Skriptnotizen

`Makefile-genFamMem`:

```
#
# makefile fuer die Skriptnotizen und die Folien zum
# Kurs Unix-Tools im WS 01/02
#
```

```
# Jan Bredereke, Universitaet Bremen
#

###############################################################################

# generierte include-Dateien

# Die folgende, generierte Datei enthaelt zum einen die Abhaengigkeiten
# von einzelfolien/folie-%.ps zu folien.dvi, und sie definiert zum
# anderen die Variable einzelfolien mit allen diesen Postscript-Dateien.
include folien-label.mk

# Die folgenden, generierten Dateien enthalten die Abhaengigkeiten der
# Dateien folien.dvi und skriptnotizen.dvi von mit \verbatimtabinput
# eingeschlossenen Quelldateien.
include folien.d
include skriptnotizen.d

###############################################################################

# Variablen

###############################################################################

# Ziele fuer den manuellen Aufruf

# das Default-Ziel:
all: skriptnotizen.dvi

# Loeschen aller generierten Dateien:
clean:
	-rm skriptnotizen.dvi
	-rm skriptnotizen.aux
	-rm skriptnotizen.log
	-rm skriptnotizen.d
	-rm $(einzelfolien)
	-rmdir einzelfolien
	-rm folien.dvi
	-rm folien.aux
	-rm folien.log
	-rm folien.d
	-rm folien-label.aux
	-rm folien-label.mk
	-rm folien-label.d

# Regel: alle diese Ziele sind keine echten Dateien:
.PHONY: all clean

###############################################################################

# allgemeine Regeln
```

```
# Uebersetzen von LaTeX-Dateien in DVI-Dateien:
%.dvi %.aux: %.tex
        latex $<

# Erzeugen einer Postscript-Datei einer einzelnen Overhead-Folie aus
# einer DVI-Datei mit vielen Folien:
einzelfolien/folie-%.ps: folien.dvi einzelfolien
        dvips -pp $* -o $@ folien

# Erzeugen der make-include-Dateien fuer Abhaengigkeiten durch
# \verbatimtabinput in *.tex-Dateien:
%.d: %.tex
        sed -n -e 's/^[^%]*\\verbatimtabinput{\([^}]\+\)}.*$$/$*.dvi: \1/p' \
            $< > $@

###########################################################################

# konkrete Abhaengigkeiten

skriptnotizen.dvi: skriptnotizen.tex skriptnotizen.sty \
        $(einzelfolien) folien-label.aux

folien.dvi: folien.tex

einzelfolien:
        mkdir einzelfolien

folien-label.aux: folien.aux
        sed -n -e \
's/^\(\\newlabel{folie:\([a-zA-Z]\+\)}{{[^}]*}{\([0-9]\+\)}}\)$$/\1\\newcommand{\\folie\2}{\3}/p'\
 $< > $@

folien-label.mk: folien-label.aux
        sed -e 's!.*{\([0-9]\+\)}$$!skriptnotizen.dvi: einzelfolien/folie-\1.ps!' \
               $< > $@ && \
            sed -e 's!.*{\([0-9]\+\)}$$!einzelfolien+= einzelfolien/folie-\1.ps!' \
               $< >> $@

folien.d: folien.tex
skriptnotizen.d: skriptnotizen.tex
```

4.21 autoconf/automake – ein Überblick

Autoconf/Automake – ein Überblick

Autoconf

- Erzeugt ein Shell-Skript, das ein SW-Paket automatisch konfiguriert.
- Nur der Paket-Autor braucht Autoconf, nicht der Benutzer.
- Keine Benutzer-Interaktion notwendig.
- Alle benötigten Features werden einzelnd getestet: Hybride Systeme möglich.
- Die Skripten zur Feature-Erkennung werden von allen Paketen geteilt.

Automake

- Erzeugt Makefile-Template für Autoconf.
- Alle Gnu-Standard-Make-Ziele werden automatisch generiert.