

GNU-Make

Daniel Otte
(daniel.otte@rub.de)



Labor Projekttag
13.-16. November 2008

Beispiel Programm

GNU-Make

Ein einfaches Programm aus 3 Modulen

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

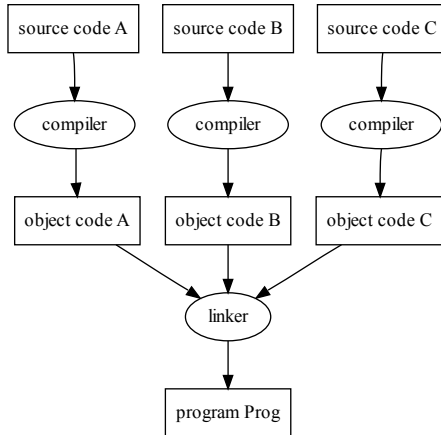
Variablen

Funktionen

EOP

Beispiel Programm

Ein einfaches Programm aus 3 Modulen



GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Beispiel Programm2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Ein anderes einfaches Programm aus 3 Modulen, mit mehr
Abhängigkeiten

Abhängigkeiten

Make

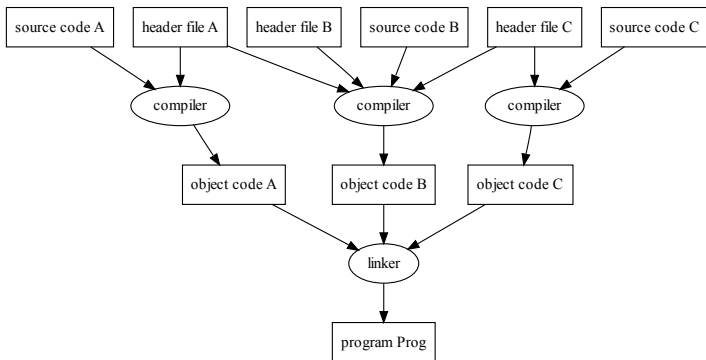
Variablen

Funktionen

EOP

Beispiel Programm2

Ein anderes einfaches Programm aus 3 Modulen, mit mehr Abhängigkeiten



GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Abhängigkeiten Programm2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Abhängigkeiten in Programm2

Abhängigkeiten Programm2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

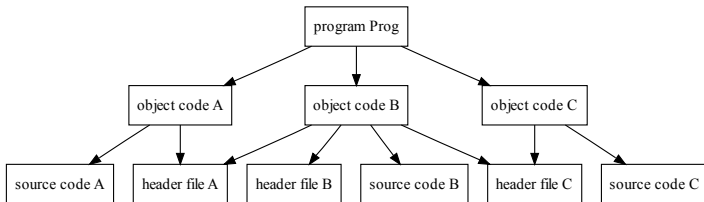
Make

Variablen

Funktionen

EOP

Abhängigkeiten in Programm2



Abhängigkeiten Programm2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Folgen einer Änderung in *header file A*

Abhängigkeiten Programm2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

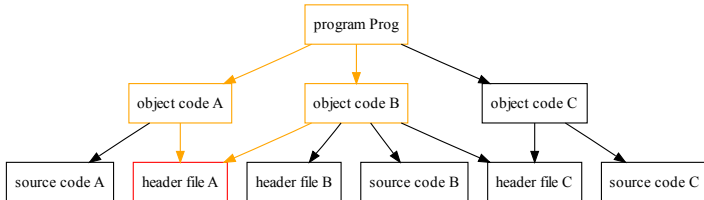
Make

Variablen

Funktionen

EOP

Folgen einer Änderung in *header file A*



Was Make für dich tut

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Was Make für dich tut

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

- Prüfung des Dateialters (mit Hilfe des Timestamps des Dateisystems)

Was Make für dich tut

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

- Prüfung des Dateialters (mit Hilfe des Timestamps des Dateisystems)
- (neu) Erzeugen der Dateien, deren Abhängigkeiten sich geändert haben

Was du tun must

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Was du tun must

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

- Make die Abhängigkeitsverhältnisse mitteilen
(welche Datei hängt von welcher ab)

Was du tun must

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

- Make die Abhängigkeitsverhältnisse mitteilen
(welche Datei hängt von welcher ab)
- Make mitteilen wie die Dateien zu bauen sind

Aufruf

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Normalerweise wird dem Make-Aufruf die Liste der Ziele (targets) übergeben, dabei kann es sich handeln um:

Wenn kein Target angegeben wird ist der Default *all*. Die Dokumentation der Optionen findest du in der Manpage.

Aufruf

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

```
make [...opts...] [target] {target}
```

Normalerweise wird dem Make-Aufruf die Liste der Ziele (targets) übergeben, dabei kann es sich handeln um:

Wenn kein Target angegeben wird ist der Default *all*. Die Dokumentation der Optionen findest du in der Manpage.

Aufruf

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

```
make [...opts...] [target] {target}
```

Normalerweise wird dem Make-Aufruf die Liste der Ziele (targets) übergeben, dabei kann es sich handeln um:

- Dateien die zu erstellen sind, oder

Wenn kein Target angegeben wird ist der Default *all*. Die Dokumentation der Optionen findest du in der Manpage.

Aufruf

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

```
make [...opts...] [target] {target}
```

Normalerweise wird dem Make-Aufruf die Liste der Ziele (targets) übergeben, dabei kann es sich handeln um:

- Dateien die zu erstellen sind, oder
- spezielle Targets die Instruktionen enthalten (Bsp. *clean*)

Wenn kein Target angegeben wird ist der Default *all*. Die Dokumentation der Optionen findest du in der Manpage.

Beispiel 1

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Ein einfaches Beispiel:

```
1 all: programm
3
3 programm: code_a.o code_b.o code_c.o
   _____gcc -o programm code_a.o code_b.o code_c.o
5
5 code_a.o: code_a.c code_a.h
7   _____gcc -c -o code_a.o code_a.c
7 code_b.o: code_b.c code_b.h code_a.h code_c.h
9   _____gcc -c -o code_b.o code_b.c
9 code_c.o: code_c.c code_c.h
11  _____gcc -c -o code_c.o code_c.c
```

Beispiel 2

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Ein einfaches Beispiel mit automatischen Variablen:

```
all: programm

programm: code_a.o code_b.o code_c.o
    _____gcc -o $@ $^

code_a.o: code_a.c code_a.h
    _____gcc -c -o $@ $<

code_b.o: code_b.c code_b.h code_a.h code_c.h
    _____gcc -c -o $@ $<

code_c.o: code_c.c code_c.h
    _____gcc -c -o $@ $<
```

Variable	Wert
\$@	target
\$^	Abhängigkeiten
\$<	erste Abhängigkeit

Beispiel 3

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Ein einfaches Beispiel mit generischen Regeln:

```
all: programm

programm: code_a.o code_b.o code_c.o
    _____gcc -o $@ $^

code_a.o: code_a.c code_a.h
code_b.o: code_b.c code_b.h code_a.h code_c.h
code_c.o: code_c.c code_c.h

%.o: %.c
    _____gcc -c -o $@ $<
```

Beispiel 4

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Ein einfaches Beispiel mit Variablen:

```
CC = gcc
```

```
all: programm
```

```
programm: code_a.o code_b.o code_c.o
```

```
_____$(CC) -o $@ $^
```

```
code_a.o: code_a.c code_a.h
```

```
code_b.o: code_b.c code_b.h code_a.h code_c.h
```

```
code_c.o: code_c.c code_c.h
```

```
%.o: %.c
```

```
_____$(CC) -c -o $@ $<
```

variable = wert

Zuweisung

\$(variable)

Referenzierung

It's all about ...

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

It's all about ...

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

■ Strings

It's all about ...

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

- Strings
- Lists (of Strings)

Listen sind ...

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Listen sind:

Listen sind ...

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

Listen sind:

- Strings, die Elemente sind durch
Whitespace getrennt

subst

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(subst from,to,text)`

Führt Stringersetzung durch.

Beispiel:

`$(subst, ee,EE,speed tree)` → `'spEEed trEE'`

`$(patsubst pattern,replacement,text)`

Führt Musterersetzung durch.

Beispiel:

`$(patsubst, %.c,%.d,code.c code.h edit.c) → 'code.d
code.h edit.d'`

strip

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(strip string)`

Entfernt Whitespace am Anfang und am Ende

Beispiel:

`$(strip, code.c code.h edit.c)` \longrightarrow `'code.d code.h edit.d'`

findstring

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

$\$(findstring\ find,in\)$

Sucht nach *find* in *in* und evaluiert zu *find* wenn gefunden
sonst zu ' ' (empty).

Beispiel:

$\$(findstring,\ a,\ a\ b\ c) \longrightarrow 'a'$

`$(filter pattern ...,text)`

Evaluiert zu einer Liste von Strings aus *text* die alle auf eins (oder mehrere) der *pattern* passen.

Beispiel:

`$(filter, %.c %.s, a.c b.h c.s) → 'a.c c.s'`

`$(filter-out pattern ...,text)`

Evaluiert zu einer Liste von Strings aus *text* die alle auf keins der *pattern* passen.

Beispiel:

`$(filter-out, %.c %.s, a.c b.h c.s) → 'b.h'`

`$(sort list)`

Sortiert die Liste und entfernt doppelte Einträge.

Beispiel:

`$(sort, foo bar foo lose)` → `'bar foo lose'`

`$(word n, text)`

Evaluiert zu dem n-ten String aus Text (Indizierung beginnend bei 1).

Beispiel:

`$(word 2, foo bar foo lose) → 'bar'`

words

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(words text)`

Evaluiert zu der Anzahl von Strings in *text*.

Beispiel:

`$(words foo bar foo lose) → '4'`

firstword

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(firstwordword names ...)`

Evaluiert zu dem ersten String in *names*.

Beispiel:

`$(firstword foo bar foo lose) → 'foo'`

lastword

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(lastwordword names ...)`

Evaluiert zu dem letzten String in *names*.

Beispiel:

`$(lastword foo bar foo lose) → 'lose'`

foreach

GNU-Make

Daniel Otte
(daniel.otte@rub.de)

Abhängigkeiten

Make

Variablen

Funktionen

EOP

`$(foreach name, list, expr)`

Evaluiert für jeden String in *list* *expr* wobei der String über die automatisch generierte Variable *name* zugänglich ist.
names.

Beispiel:

`$(foreach var, bar foo lose, $(var)_x) → 'bar_x foo_x lose_x'`

`$(call function,parameters)`

Ruft Funktion *function* auf. Besonders sinnvoll um nicht-standard Funktionen aufzurufen. (z.B. Funktionen aus der GMSL)

Beispiel:

`$(call uc,test) → 'TEST'`

`$(eval variable)`

Evaluiert *variable* im Makefile Kontext, d.h. *variable* kann Makefile Konstrukte (z.B. Rule, Dependancys) beinhalten.

Beispiel:

`$(call uc,test) → 'TEST'`

End Of Presentation