Übungen zur Vorlesung
Testautomatisierung
Sommersemester 2004

AG BS
Jan Peleska, Stefan Bisanz,
Aliki Tsiolakis

Blatt 1

Revision: 1.2

Testen des Unix-Kommandos `ln` (Erzeugung von Soft- und Hard Links)

Aufgabe 1: Strukturierte Anforderungsspezifikation

Setzen Sie den Inhalt der Manual Page ln(1) vollständig in eine tabellarische Anforderungsspezifikation (Requirements Specification) um:

Requirements Tag	Requirement
R001	<Beschreibung der Anforderung>
R002	...
...	

Nummerieren Sie die Requirements Tags mit R001, R002, ...
Verwenden Sie die Manual Page der Rechner {x01,...,x25}.informatik.uni-bremen.de:

```
LN(1) -- March 2004 -- ln (coreutils) 5.0 -- FSF

NAME
     ln - make links between files

SYNOPSIS
     ln [OPTION]... TARGET [LINK_NAME]
     ln [OPTION]... TARGET... DIRECTORY
     ln [OPTION]... --target-directory=DIRECTORY TARGET...

DESCRIPTION

     Create a link  to the specified TARGET  with optional LINK_NAME.  If LINK_NAME  is omitted, a
     link with the  same basename as the TARGET  is created in the current  directory.  When using
     the second  form with more than  one TARGET, the last  argument must be  a directory;  create
     links  in DIRECTORY  to  each TARGET.   Create hard  links  by default,  symbolic links  with
     --symbolic.  When creating hard links, each TARGET must exist.

     Mandatory arguments to long options are mandatory for short options too.

     --backup[=CONTROL]
          make a backup of each existing destination file

     -b   like --backup but does not accept an argument

     -d, -F, --directory
          hard link directories (super-user only)

     -f, --force
          remove existing destination files

     -n, --no-dereference
          treat destination that is a symlink to a directory as if it were a normal file
```

```
     -i, --interactive
          prompt whether to remove destinations

     -s, --symbolic
          make symbolic links instead of hard links

     -S, --suffix=SUFFIX
          override the usual backup suffix

     --target-directory=DIRECTORY
          specify the DIRECTORY in which to create the links

     -v, --verbose
          print name of each file before linking

     --help
          display this help and exit

     --version
          output version information and exit

     The backup suffix is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.  The version
     control method may be selected via the --backup option or through the VERSION_CONTROL
     environment variable.  Here are the values:

     none, off
          never make backups (even if --backup is given)

     numbered, t
          make numbered backups

     existing, nil
          numbered if numbered backups exist, simple otherwise

     simple, never
          always make simple backups

AUTHOR
     Written by Mike Parker and David MacKenzie.

REPORTING BUGS
     Report bugs to <bug-coreutils@gnu.org>.

COPYRIGHT
     Copyright (C) 2003 Free Software Foundation, Inc.
     This is free software; see the source for copying conditions.  There is NO warranty; not even
     for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO
     The full documentation for ln is maintained as a Texinfo manual.  If the info and ln programs
     are properly installed at your site, the command

          info ln

     should give you access to the complete manual.
```

Aufgabe 2: Entwurf von Testfällen

Entwerfen Sie eine Kollektion von Testfällen, welche zusammen alle Anforderungen aus Aufgabe 1 prüfen. Jeder Testfall wird als strukturierter Text beschrieben:

@tag <Testfall-Id>

@description <Anschauliche Erläuterung des Testziels>

@condition <Vorbedingung(en) zur Ausführung des Testfalls>

@event <Auslösen des Prüfschritts, d.h. Form des Aufrufs von `ln` mit zugehörigen Optionen und Eingabeparametern>

@expected <Erwartetes Resultat für diesen Aufruf>

Wählen Sie TC001, TC002, ... als Format für die Testfall-Ids.

Ein Testfall kann dazu dienen, mehrere Anforderungen zu prüfen; umgekehrt können mehrere Testfälle erforderlich sein, um eine Anforderung vollständig abzudecken. Somit gibt es eine n:m-Beziehung zwischen Anforderungen und Testfällen.

Aufgabe 3: Zuordnung von Anforderungen und Testfällen

Stellen Sie eine Anforderungsüberdeckungsmatrix (Requirements Coverage Matrix) in der folgenden Form auf:

	R001	R002	R003	...
TC001		•		
TC002		•	•	
TC003	•			
...				

Ein • in der Zelle (tc, r) bedeutet, daß Testfall tc dazu beiträgt, die Anforderung r zu prüfen.

Aufgabe 4: Entwurf von Testprozeduren

Entwerfen Sie eine möglichst kleine Kollektion von Testprozeduren, welche alle Testfälle automatisch ausführen. Damit sind alle Anforderungen abgedeckt, vgl. Aufgaben 2 und 3. Schreiben Sie die Prozeduren als bash-Skripte, welche im Dateisystem die Vorbedingungen (existierende Verzeichnisse, Links, Zugriffsrechte etc.) für jeden Testfall herstellen, den erforderlichen `ln`-Aufruf durchführen und hinterher die Wirkung des `ln`-Kommandos gegen die erwarteten Resultate prüfen.

Hinweis: Verzeichnisse, Dateien und Links können mit dem `if`-Konstrukt der bash auf Existenz abgefragt werden. Zugriffsrechte können mit `ls -l <file> | grep -E <filter>` geprüft werden.

Stellen Sie sicher, daß die bash-Skripte (1) ausführbar (syntaktisch korrekt) sind und daß sie (2) Ausgaben erzeugen, aus denen der jeweilige Testfall, dessen Wirkung sowie das Testergebnis hervorgeht. Verwenden Sie dafür einen der Rechner {x01,...,x25}.informatik.uni-bremen.de.

Binden Sie die Ausgaben der bash-Skripte in Ihre Aufgabenlösung ein!

Abgabe: Bis Montag, 10. Mai 2004, im Tutorium.

Geben Sie alle Aufgabenlösungen sowohl (1) *ausgedruckt* (in MZH8210, MZH8170, oder im Tutorium) als auch (2) *elektronisch* (mailto:bisanz@informatik.uni-bremen.de) ab.

In allen Dokumenten und Dateien die Namen aller Gruppenmitglieder nicht vergessen!