6. APRIL 2019

PRAKTIKUM BETRIEBSYSTEME

AUFGABE 1 – UNIX (LINUX)

CHAU WU, MICHAEL PRUSAKOWSKI
BAI3 BSP/3
TDM/SLZ

INHALTSVERZEICHNIS

1 Erste Erfahrungen mit der bash – Shell (Linux-Befehlsinterpreter)

1.1 a) Testprotokoll

1.1.1 Befehl: strg-C

Testszenario:

1. Schritt Programm „ping“ mit Parameter „www.google.de“ ausführen.
2. Schritt Während der Laufzeit von Programm „ping“ Tastenkombination „strg-C“ eingeben.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **ping www.google.de**

Ausgabe:

PING www.google.de (216.58.207.35) 56(84) bytes of data.
64 bytes from fra16s24-in-f3.1e100.net (216.58.207.35): icmp_seq=1 ttl=51 time=18.7 ms
64 bytes from fra16s24-in-f3.1e100.net (216.58.207.35): icmp_seq=2 ttl=51 time=20.3 ms

Beobachtung:

Programm „ping“ wurde gestartet und hat im Beobachtungszeitraum zwei „ping“-Ergebnisse ausgegeben.

Das Programm „ping“ wird weiterhin ausgeführt.

2. Schritt

Eingabe:

Tastenkombination: **strg-C**

Ausgabe:

^C
--- www.google.de ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 18.743/19.557/20.371/0.814 ms

Beobachtung:

Das Programm „ping“ wurde vorzeitig abgebrochen. (Ergebnisse des vorzeitig abgebrochenen „ping“-Programms werden noch ausgegeben.)

Testergebnis:

Die Tastenkombination strg-C bricht ein laufendes Programm ab.

1.1.2 Befehl: cat FILE

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt Programm „cat“ mit dem Parameter „foo.txt“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **cat foo.txt**

Ausgabe:

Hello, world.

Beobachtung:

Der Inhalt der im 2. Schritt erzeugten Textdatei „foo.txt“ wird im Terminal ausgegeben.

Testergebnis:

Das Programm „cat“ gibt den Inhalt der Textdatei FILE auf der Standartausgabe (stdout) aus.

1.1.3 Befehl: cd DIR

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

Testergebnis:

Das Programm „cd“ wechselt das aktuelle Verzeichnis zum Pfad DIR.

1.1.4 Befehl: chmod [ugoa][+/-][rwx] FILE

Testszenario:

1. Schritt	Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt	Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt	Überprüfe ob Andere nur Leserechte haben.
4. Schritt	Ändere mit Programm „chmod“ die Berechtigungen für Andere auf Lese- und Schreibrecht der Textdatei „foo.txt“.
5. Schritt	Überprüfe ob Andere Lese- und Schreibrechte haben.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls -la**

Ausgabe:

total 12

drwxrwxr-x 2 xxx xxx 4096 Mär 31 22:24 .

drwxrwxr-x 3 xxx xxx 4096 Mär 31 22:12 ..

-rw-rw-r-- 1 xxx xxx 14 Mär 31 22:24 foo.txt

Beobachtung:

Das Programm „ls“ mit dem Parameter „-la“ listet alle Dateien und Ordner im aktuellen Verzeichnis auf, mit unter Anderem Berechtigungen für Lese-/Schreib-/Ausführberechtigung.

Andere haben auf die Textdatei „foo.txt“ nur Leseberechtigung.

4. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **chmod a+rw foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine sichtbare Beobachtung.

5. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls -la**

Ausgabe:

total 12

drwxrwxr-x 2 xxx xxx 4096 Mär 31 22:24 .

drwxrwxr-x 3 xxx xxx 4096 Mär 31 22:12 ..

-rw-rw-rw- 1 xxx xxx 14 Mär 31 22:24 foo.txt

Beobachtung:

Siehe 3. Schritt.

Andere haben auf die Textdatei „foo.txt“ Lese- und Schreibberechtigung.

Testergebnis:

Das Programm „chmod“ ändert Zugriffsrechte bzgl. Der Datei FILE.

1.1.5 Befehl: cp [-i] FILE1 FILE2

Testszenario:

1. Schritt	Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt	Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt	Vorhandene Dateien im aktuellen Ordner auflisten.
4. Schritt	Programm „cp“ mit Parameter „foo.txt“ und „bar.txt“ ausführen.
5. Schritt	Vorhandene Dateien im aktuellen Ordner auflisten und überprüfen, ob die Textdatei „bar.txt“ erzeugt wurde.
6. Schritt	Programm „cp“ mit Parameter „-i“, „foo.txt“ und „bar.txt“ ausführen.
7. Schritt	Vorhandene Dateien im aktuellen Ordner auflisten und überprüfen, ob die Textdatei „bari.txt“ erzeugt wurde.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls**

Ausgabe:

foo.txt

Beobachtung:

Das Programm „ls“ listet alle Dateien und Ordner im aktuellen Verzeichnis auf.

Die Textdatei „foo.txt“ wurde erzeugt.

4. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **cp foo.txt bar.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Die Textdatei „foo.txt“ wurde kopiert und als „bar.txt“ gespeichert.

5. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls**

Ausgabe:

bar.txt foo.txt

Beobachtung:

Die Textdatei „bar.txt“ wurde erzeugt.

6. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **cp -i foo.txt bar.txt**

Ausgabe:

cp: overwrite 'bar.txt'?

Beobachtung:

Das Programm „cp“ mit dem Parameter „-i“ fordert den Benutzer zu einer Bestätigung auf, wenn eine Datei überschrieben werden soll.

Die Textdatei „foo.txt“ wurde kopiert und als „bar.txt“ gespeichert.

7. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls**

Ausgabe:

bar.txt foo.txt

Beobachtung:

Die Textdatei „bar.txt“ wurde überschrieben.

Testergebnis:

Das Programm „cp“ kopiert eine Datei FILE1 nach FILE2.

Der Parameter „-i“ fordert den Benutzer zu einer Bestätigung auf, wenn eine Datei überschrieben werden soll.

1.1.6 Befehl: date

Testszenario:

1. Schritt Programm „date“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **date**

Ausgabe:

Mo 1. Apr 00:32:32 CEST 2019

Beobachtung:

Das aktuelle Datum und die aktuelle Uhrzeit wurden ausgegeben.

Testergebnis:

Das Programm „date“ gibt das aktuelle Datum und die aktuelle Uhrzeit aus.

1.1.7 Befehl: df

Testszenario:

1. Schritt Programm „df“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **df**

Ausgabe:

Filesystem	**1K-blocks**	**Used**	**Available**	**Use%**	**Mounted on**
udev	1989476	0	1989476	0%	/dev
tmpfs	403972	6120	397852	2%	/run
/dev/mapper/ubuntu--vg-root	26768440	5114336	26768440	21%	/
tmpfs	2019852	132	2019720	1%	/dev/shm
tmpfs	5120	4	5116	1%	/run/lock
tmpfs	2019852	0	2019852	0%	/sys/fs/cgroup
/dev/sda1	482922	141235	316753	31%	/boot
tmpfs	403972	64	403908	1%	/run/user/1000

Beobachtung:

Informationen über die Dateisysteme werden ausgegeben.

Testergebnis:

Das Programm „df“ zeigt Informationen über die Dateisysteme an.

1.1.8 Befehl: echo STRING

Testszenario:

1. Schritt Programm „echo“ mit Parameter „Hello, world.“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **echo „Hello, world.“**

Ausgabe:

Hello, world.

Beobachtung:

Übergebene Zeichenkette wurde ausgegeben.

Testergebnis:

Das Programm „echo“ gibt die Zeichenkette STRING auf der Standartausgabe (stdout) aus.

1.1.9 Befehl: env

Testszenario:

1. Schritt Programm „env“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **env**

Ausgabe:

XDG_VTNR=7

LC_PAPER=de_DE.UTF-8

LC_ADDRESS=de_DE.UTF-8

XDG_SESSION_ID=c1

XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/xxx

LC_MONETARY=de_DE.UTF-8

CLUTTER_IM_MODULE=xim

SESSION=ubuntu

GPG_AGENT_INFO=/home/xxx/.gnupg/S.gpg-agent:0:1

TERM=xterm-256color

VTE_VERSION=4205

XDG_MENU_PREFIX=gnome-

SHELL=/bin/bash

QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1

WINDOWID=60817482

LC_NUMERIC=de_DE.UTF-8

UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1061

GNOME_KEYRING_CONTROL=

GTK_MODULES=gail:atk-bridge:unity-gtk-module

USER=xxx

LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;3

1:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:

QT_ACCESSIBILITY=1

LC_TELEPHONE=de_DE.UTF-8

XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0

XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0

SSH_AUTH_SOCK=/run/user/1000/keyring/ssh

SESSION_MANAGER=local/xxx-VirtualBox:@/tmp/.ICE-unix/1311,unix/xxx-VirtualBox:/tmp/.ICE-unix/1311

DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path

XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg

DESKTOP_SESSION=ubuntu

PATH=/home/xxx/bin:/home/xxx/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

QT_IM_MODULE=ibus

QT_QPA_PLATFORMTHEME=appmenu-qt5

LC_IDENTIFICATION=de_DE.UTF-8

XDG_SESSION_TYPE=x11

PWD=/home/xxx/Desktop/BSP/Aufgabe 1

JOB=dbus

XMODIFIERS=@im=ibus

GNOME_KEYRING_PID=

LANG=en_US.UTF-8

GDM_LANG=en_US

MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path

LC_MEASUREMENT=de_DE.UTF-8

COMPIZ_CONFIG_PROFILE=ubuntu

IM_CONFIG_PHASE=1

GDMSESSION=ubuntu

SESSIONTYPE=gnome-session

GTK2_MODULES=overlay-scrollbar

SHLVL=1

HOME=/home/xxx

XDG_SEAT=seat0

LANGUAGE=en_US

LIBGL_ALWAYS_SOFTWARE=1

GNOME_DESKTOP_SESSION_ID=this-is-deprecated

XDG_SESSION_DESKTOP=ubuntu

LOGNAME=xxx

DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-wGdS3B1pyr

XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share:/var/lib/snapd/desktop

QT4_IM_MODULE=xim

LESSOPEN=| /usr/bin/lesspipe %s

INSTANCE=

XDG_RUNTIME_DIR=/run/user/1000

DISPLAY=:0

XDG_CURRENT_DESKTOP=Unity

GTK_IM_MODULE=ibus

LESSCLOSE=/usr/bin/lesspipe %s %s

LC_TIME=de_DE.UTF-8

LC_NAME=de_DE.UTF-8

XAUTHORITY=/home/xxx/.Xauthority

_=/usr/bin/env

OLDPWD=/home/xxx

Beobachtung:

Die Umgebungsvariablen wurden angezeigt.

Testergebnis:

Das Programm „env“ zeigt die Umgebungsvariablen an.

1.1.10 Befehl: exit

Testszenario:

1. Schritt Program „exit“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **exit**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Die Shell wurde beendet.

Testergebnis:

Das Programm „exit“ beendet die Shell oder das Script.

1.1.11 Befehl: export VAR

Testszenario:

1. Schritt	Variable „foo“ deklarieren und Wert „Hello, world.“ zuweisen.
2. Schritt	Variable „bar“ deklarieren und Wert „Goodbye.“ zuweisen.
3. Schritt	Program „export“ mit Parameter „foo“ ausführen.
4. Schritt	Neue Shell als root starten.
5. Schritt	Variable foo ausgeben.
6. Schritt	Variable bar ausgeben.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **foo=“Hello, world.“**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine Sichtbare Beobachtung, Vaiable „foo“ wurde der Wert „Hello, world.“ zugewiesen.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **bar=“Goodbye.“**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine Sichtbare Beobachtung, Vaiable „bar“ wurde der Wert „Goodbye.“ zugewiesen.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **export foo**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine Sichtbare Beobachtung. (Variable „foo“ wurde an alle Kindprozesse vererbt.)

4. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **sudo bash**

Ausgabe:

[sudo] password for xxx:

Beobachtung:

Der Benutzer wird aufgefordert zur Passworteingabe für den Superuser-Modus.

Nach erfolgreicher Passworteingabe wird in der Shell eine neue Shell als „root“ ausgeführt.

5. Schritt

Eingabe:

root@xxx-VirtualBox:~$ **echo $foo**

Ausgabe:

Hello, world.

Beobachtung:

Die Variable „foo“ wurde ausgegeben, mit dem Inhalt aus dem 1. Schritt.

6. Schritt

Eingabe:

root@xxx-VirtualBox:~$ **echo $bar**

Ausgabe:

Beobachtung:

Die Variable „bar“ wurde ausgegeben, mit leerem Inhalt, obwohl der Elternprozess der Variable den Wert „Goobye“ zugewiesen hat.

Testergebnis:

Das Programm „export“ vererbt die Variable VAR an alle Kindprozesse.

1.1.12 Befehl: find DIR -name FILE -print

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt Das Programm „find“ mit den Parameter „./ -name foo.txt -print“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **find ./ -name foo.txt -print**

Ausgabe:

./Desktop/BSP/Aufgabe 1/foo.txt

Beobachtung:

Die Textdatei namens „foo.txt“ wurde im aktuellen Verzeichnis gefunden.

Der Dateipfad der Textdatei „foo.txt“ wurde ausgegeben

Testergebnis:

Das Program „find“ findet eine Datei namens FILE beginnend im Verzeichnis DIR.

Der Parameter -print gibt den gefundenen Dateipfad zurück.

1.1.13 Befehl: grep [-r] STRING FILE

Testszenario:

1. Schritt	Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt	Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt	Eine Textdatei „bar.txt“ erzeugen mit dem Inhalt „Hello, world.“.
4. Schritt	Das Programm „grep“ mit den Parameter „-r „world“ ./“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >bar.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „bar.txt“ erzeugt mit dem Inhalt „Hello, world.“.

4. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **grep -r “world” ./**

Ausgabe:

./bar.txt:Hello, world.

./foo.txt:Hello, world.

Beobachtung:

Es wurden alle Vorkommen der Zeichenkette „world“ in allen Dateien in dem aktuellen Verzeichnis gefunden und deren Pfade und Positionen der Zeichenketten angezeigt.

Testergebnis:

Das Programm „grep“ sucht in der Datei FILE nach der Zeichenkette STRING.

Der Parameter „-r“ führt dazu, dass in dem angegebenen Verzeichnis rekursiv in allen Dateien nach der Zeichenkette STRING gesucht wird.

1.1.14 Befehl: ls [-l] [FILE]

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt Das Programm „ls“ ausführen mit dem Parameter „-l“.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **ls -l**

Ausgabe:

total 4

-rw-rw-rw- 1 xxx xxx 14 Mär 31 22:24 foo.txt

Beobachtung:

Das Programm „ls“ mit dem Parameter „-l“ listet alle Dateien und Ordner im aktuellen Verzeichnis auf, mit unter Anderem Berechtigungen für Lese-/Schreib-/Ausführberechtigung und Erstellungsdatum.

Testergebnis:

Das Programm „ls“ zeigt den aktuellen Verzeichnis-Inhalt als Liste von Dateinamen an.

Der Parameter „-l“ wird verwendet um ein langes Listenformat zu nutzen.

1.1.15 Befehl: man PROG

Testszenario:

1. Schritt Program „man“ ausführen mit Parameter „ping“.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **man ping**

Ausgabe:

NAME

ping, ping6 - send ICMP ECHO_REQUEST to network hosts

SYNOPSIS

ping [-aAbBdDfhLnOqrRUvV] [-c count] [-F flowlabel] [-i interval] [-I interface] [-l preload] [-m mark] [-M pmtudisc_option] [-N nodeinfo_option] [-w deadline] [-W timeout] [-p pattern] [-Q tos] [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp option] [hop ...] destination

DESCRIPTION

ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (``pings'') have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of ``pad'' bytes used to fill out the packet.

ping6 is IPv6 version of ping, and can also send Node Information Queries (RFC4620). Intermediate hops may not be allowed, because IPv6 source routing was deprecated (RFC5095).

OPTIONS

-a Audible ping.

-A Adaptive ping. Interpacket interval adapts to round-trip time, so that effectively not more than one (or more, if preload is set) unanswered probe is present in the network. Minimal interval is 200msec for not super-user. On networks with low rtt this mode is essentially equivalent to flood mode.

-b Allow pinging a broadcast address.

-B Do not allow ping to change source address of probes. The address is bound to one selected when ping starts.

-c count

Stop after sending count ECHO_REQUEST packets. With deadline option, ping waits for count ECHO_REPLY packets, until the time-

Beobachtung:

Eine Beschreibung des Programms „ping“ wird angezeigt.

Testergebnis:

Das Programm „man“ gibt eine Beschreibung des Programms PROG aus.

1.1.16 Befehl: mkdir DIR

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Programm „mkdir“ mit Parameter „foo“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **mkdir foo**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das Verzeichnis „foo“ wurde in dem aktuellen Verzeichnis erstellt.

Testergebnis:

Das Programm „mkdir“ erstellt in dem Pfad DIR ein Verzeichnis.

1.1.17 Befehl: more FILE

Testszenario:

1. Schritt	Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt	Eine Textdatei „foo.txt“ erzeugen mit 100 Zeilen
3. Schritt	Programm „more“ ausführen mit dem Parameter „foo.txt“

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo "1" >foo.txt; for i in {2..100}; do echo $i >>foo.txt; done;**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Die Textdatei „foo.txt“ wurde erzeugt mit 100 Zeilen.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **more foo.txt**

Ausgabe:

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
--More—(20%)

Beobachtung:

Der Inhalt der Textdatei „foo.txt“ wurde seitenweise ausgegeben.

Testergebnis:

Das Programm „more“ zeigt die Textdatei FILE seitenweise an.

1.1.18 Befehl: mv QUELLE ZIEL

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.
3. Schritt Programm „mv“ mit Parameter „foo.txt“ und Parameter „bar.txt“.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

3. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **mv foo.txt bar.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Die Textdatei „foo.txt“ wurde umbenannt in „bar.txt“.

Testergebnis:

Das Programm „mv“ verschiebt die Datei QUELL bzw. verschiebt diese.

1.1.19 Befehl: PROG

Testszenario:

1. Schritt Programm „ping“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **ping**

Ausgabe:

Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]

[-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]

[-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]

[-w deadline] [-W timeout] [hop1 ...] destination

Beobachtung:

Das Programm „ping“ wurde ausgeführt (und gibt Benutzungshinweise aus, aufgrund von Fehlenden Parametern).

Testergebnis:

Der Befehl startet das ausführbare Programm PROG.

Das Programm wird in den in $PATH angegebenen Verzeichnissen gesucht.

1.1.20 Befehl: ps [-efa]

Testszenario:

1. Schritt Programm „ps“ mit dem Parameter „-efa“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **ps -efa**

Ausgabe:

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 04:59 ?        00:00:01 /sbin/init splash
root         2     0  0 04:59 ?        00:00:00 [kthreadd]
root         4     2  0 04:59 ?        00:00:00 [kworker/0:0H]
root         6     2  0 04:59 ?        00:00:00 [mm_percpu_wq]
root         7     2  0 04:59 ?        00:00:00 [ksoftirqd/0]
root         8     2  0 04:59 ?        00:00:00 [rcu_sched]
root         9     2  0 04:59 ?        00:00:00 [rcu_bh]
root        10     2  0 04:59 ?        00:00:00 [migration/0]
root        11     2  0 04:59 ?        00:00:00 [watchdog/0]
root        12     2  0 04:59 ?        00:00:00 [cpuhp/0]
root        13     2  0 04:59 ?        00:00:00 [kdevtmpfs]
root        14     2  0 04:59 ?        00:00:00 [netns]
root        15     2  0 04:59 ?        00:00:00 [rcu_tasks_kthre]
root        16     2  0 04:59 ?        00:00:00 [kauditd]
root        17     2  0 04:59 ?        00:00:00 [khungtaskd]
root        18     2  0 04:59 ?        00:00:00 [oom_reaper]
root        19     2  0 04:59 ?        00:00:00 [writeback]
root        20     2  0 04:59 ?        00:00:00 [kcompactd0]
…
```

Beobachtung:

Es wurden alle Prozesse aufgelistet.

Testergebnis:

Das Programm „ps“ zeigt Prozess-Informationen an.

Der Parameter „-e“ wird genutzt, um alle Prozesse auszuwählen.

Der Parameter „-f“ wird genutzt, um ein vollständiges Format anzuzeigen.

Der Parameter „-a“ wird genutzt, um Prozesse auszuschließen, die nicht mit dem Terminal assoziiert werden.

1.1.21 Befehl: pstree

Testszenario:

1. Schritt Programm „pstree“ ausführen.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **pstree**

Ausgabe:

```
systemd─┬─NetworkManager─┬─dhclient
        │                ├─dnsmasq
        │                ├─{gdbus}
        │                └─{gmain}
        ├─VBoxClient───VBoxClient───{SHCLIP}
        ├─VBoxClient───VBoxClient
        ├─VBoxClient───VBoxClient───{X11 events}
        ├─VBoxClient───VBoxClient─┬─{dndHGCM}
        │                         └─{dndX11}
        ├─VBoxService─┬─{automount}
        │             ├─{control}
        │             ├─{cpuhotplug}
        │             ├─{memballoon}
        │             ├─{timesync}
        │             ├─{vminfo}
        │             └─{vmstats}
        ├─accounts-daemon─┬─{gdbus}
        │                 └─{gmain}
        ├─acpid
        ├─agetty
        ├─avahi-daemon───avahi-daemon
        ├─colord─┬─{gdbus}
```

Beobachtung:

Es wurden alle Prozesse in einer Baumstruktur ausgegeben.

Testergebnis:

Das Programm „pstree“ zeigt Prozess-Informationen als Baumstruktur an.

1.1.22 Befehl: VAR=VALUE

Testszenario:

1. Schritt Variable „foo“ den Wert „bar“ zuweisen.
2. Schritt Die Variable „foo“ ausgeben.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **foo=”bar”**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine Sichtbare Beobachtung. (Der Wert „bar“ wurde der Variable „foo“ zugewiesen.)

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **echo $foo**

Ausgabe:

bar

Beobachtung:

Der im 1. Schritt zugewiesene Wert „bar“ der Variable „foo“ wurde ausgegeben.

Testergebnis:

Der Shell-Variable VAR wird der Wert VALUE zugewiesen.

1.1.23 Befehl: > FILE

Testszenario:

1. Schritt Zum Ordner ~/BSP/Aufgabe 1/ navigieren.
2. Schritt Eine Textdatei „foo.txt“ erzeugen mit dem Inhalt „Hello, world.“.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **cd ~/Desktop/BSP/Aufgabe\ 1/**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Das aktuell gewählte Verzeichnis ist vom Home-Verzeichnis zum Verzeichnis ~/Desktop/BSP/Aufgabe 1 gewechselt.

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~/Desktop/BSP/Aufgabe 1$ **echo 'Hello, world.' >foo.txt**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Es wurde eine Textdatei „foo.txt“ erzeugt mit dem Inhalt „Hello, world.“.

Testergebnis:

Die Standartausgabe (stdout) wird auf die Datei FILE umgelenkt.

Die Datei FILE wird ggf. neu erzeugt oder überschrieben.

1.1.24 Befehl: $VAR

Testszenario:

1. Schritt Der Variable „foo“ den Wert „bar“ zuweisen.
2. Schritt Die Variable „foo“ ausgeben.

1. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **foo=”bar”**

Ausgabe:

Keine Ausgabe.

Beobachtung:

Keine Sichtbare Beobachtung. (Der Wert „bar“ wurde der Variable „foo“ zugewiesen.)

2. Schritt

Eingabe:

xxx@xxx-VirtualBox:~$ **echo $foo**

Ausgabe:

bar

Beobachtung:

Der im 1. Schritt zugewiesene Wert „bar“ der Variable „foo“ wurde ausgegeben.

Testergebnis:

Der Befehl ersetzt die Zeichenkette **$**VAR durch den aktuellen Wert der Variable VAR.

1.2 B) ANTWORTEN

Was enthalten die folgenden Umgebungsvariablen (Environment Variables)?

- $HOME Enthält den Pfad des aktuell angemeldeten Benutzers.
- $PATH Gibt eine Reihe von Verzeichnissen an, in denen sich ausführbare Programme befinden.
- $UID Die Benutzer ID.
- $USER Benutzername.

Was bewirkt der Befehl "cd $HOME" ? Gibt es eine einfachere Alternative?

Wechselt das aktuell ausgewählte Verzeichnis zum Home (~) Verzeichnis.

cd ~ ist die kürzere Alternative.

Was für eine Funktion haben die folgenden Eingaben?

- ↑ In der Befehlshistorie ein Befehl zurück gehen.
- ↓ In der Befehlshistorie ein Befehl vorwärts gehen.
- strg-d (in leerer Zeile) Beendet die Shell.

Was ist die Funktion der .bashrc Datei im Verzeichnis $HOME?

Die bashrc ist eine Konfigurations-Datei für die Bash, welche bei jedem Öffnen eines Terminals mitgeladen wird. In ihr kann man Abkürzungen für Befehle (also ein alias) eintragen, man verändert hier das Aussehen des Prompts oder man kann zusätzliche Skripte einfügen, welche einem das Arbeiten mit der Shell erleichtern.

2 SHELL-SKRIPTE

2.1 A) FRENAME.SH <STRING>

```
#!/bin/bash
# printUsage gibt die Syntax für die Verwendung von frename aus.
printUsage() {
	echo "Usage: frename <string>"
}
# Prüft ob die Anzahl der übergebenen Argumente gleich 1 ist.
if [[ $# -eq 1 ]]; then
	# Wenn ja, dann...
	# Prüft ob das erste Argument leer ist.
	if [[ -z "$1" ]]; then
		# Wenn ja, dann...
		# Ausgabe der "Usage Message".
		printUsage
	else
		# Ansonsten...
		# Iteration durch alle Dateien im aktuellen Verzeichnis.
		for file in *; do
			# Extrahiert den Dateinamen ohne Dateiendung, der Datei aus dem aktuellen
			# Iterationsschritt.
			filename="${file%.*}"
			# Extrahiert die Dateiendung ohne Dateinamen, der Datei aus dem aktuellen
			# Iterationsschritt.
			extension="${file#*.}"
			# die Datei aus dem aktuellen Iterationsschritt wird umbenannt.
			# Der neue Dateiname setzt sich zusammen aus dem Dateinamen + dem
			# ersten Argument + der Dateiendung.
			mv $file $filename$1.$extension
		done
	fi
else
	# Ansonsten...
	# Ausgabe der "Usage Message"
	printUsage
fi
```

2.2 B) TRY_HOST.SH [-H|-S <SEC>] <HOSTNAME>|<IP-ADDRESS>

```
#!/bin/bash
# printUsage gibt die Syntax für die Verwendung von try_host aus.
printUsage() {
	echo "Usage: try_host [-h|-s <sec>] <hostname>|<IP-Address>"
}

# Prüft, ob die Anzahl der Argumente gleich 1 ist.
if [[ $# -eq 1 ]]; then
	# Wenn ja, dann...
	# Prüft ob das erste Argument leer ist.
	if [[ -z $1 ]]; then
		# Wenn ja, dann...
		# Ausgabe der "Usage Message"
		printUsage
	else
		# Ansonsten...
		# Prüft ob das erste Argument gleich "-h" ist.
		if [[ $1 == "-h" ]]; then
			# Wenn ja, dann...
			# Ausgabe der "Usage Message"
			printUsage
		else
			# Ansonsten...
			# Wiederhole endlos...
			while [[ true ]]; do
				# Sendet einen einzelnen Ping an den Host / die IP-Addresse aus dem ersten
				# Argument und verhindert die Ausgabe.
				ping -n 1 $1 >/dev/null
				# Prüft, ob der letzte ausgeführte Befehl erfolgreich war.
				if [[ $? -eq 0 ]]; then
					# Wenn ja, dann...
					# Ausgabe des Host's/IP-Addresse mit dem Vermerk "OK".
					echo $1 "OK"
				else
					# Ansonsten...
					# Ausgabe des Host's/IP-Addresse mit dem Vermerk "FAILED".
					echo $1 "FAILED"
				fi
				# Wartet 10 Sekunden.
				sleep 10
			done
		fi
	fi
elif [[ $# -eq 3 ]]; then
	# Ansonsten, wenn die Anzahl gleich 3 ist...
	# Prüft, ob das erste Argument leer ist.
	if [[ -z $1 ]]; then
		# Wenn ja, dann...
		# Ausgabe der "Usage Message"
		printUsage
	else
		# Ansonsten...
		# Prüft, ob das zweite Argument leer ist.
		if [[ -z $2 ]]; then
			# Wenn ja, dann...
			# Ausgabe der "Usage Message"
			printUsage
		else
			# Ansonsten...
			# Prüft ob das dritte Argument leer ist.
			if [[ -z $3 ]]; then
				# Wenn ja, dann...
				# Ausgabe der "Usage Message"
```

```
                                printUsage
                        else
                                # Ansonsten...
                                # Prüft, ob das erste Argument gleich "-s" ist.
                                if [[ $1 == "-s" ]]; then
                                        # Wenn ja, dann...
                                        # Regulärer Ausdruck für Zahlen von 0-9 mit mindestens einer Ziffer.
                                        re='^[0-9]+$'
                                        # Prüft, ob das zweite Argument dem regulären Ausdruck entspricht.
                                        if [[ $2 =~ $re ]]; then
                                                # Wenn ja, dann...
                                                # Wiederhole endlos...
                                                while [[ true ]]; do
                                                        # Sendet einen einzelnen Ping an den Host / die IP-
                                                        # Addresse aus dem dritten Argument und verhindert
                                                        # die Ausgabe.
                                                        ping -n 1 $3 >/dev/null
                                                        # Prüft, ob der letzte ausgeführte Befehl
                                                        # Erfolgreich war.
                                                        if [[ $? -eq 0 ]]; then
                                                                # Wenn ja, dann...
                                                                # Ausgabe des Host's/IP-Addresse mit dem
                                                                # Vermerk "OK".
                                                                echo $3 "OK"
                                                        else
                                                                # Ansonsten...
                                                                # Ausgabe des Host's/IP-Addresse mit dem
                                                                # Vermerk "FAILED".
                                                                echo $3 "FAILED"
                                                        fi
                                                        # Wartet die Anzahl an Sekunden die im zweiten
                                                        # Argument übergeben wurde.
                                                        sleep $2
                                                done
                                        else
                                                # Ansonsten...
                                                # Ausgabe der "Usage Message"
                                                printUsage
                                        fi
                                else
                                        # Ansonsten...
                                        # Ausgabe der "Usage Message"
                                        printUsage
                                fi
                        fi
                fi
        fi
else
        # Ansonsten...
        # Ausgabe der "Usage Message"
        printUsage
fi
```

3 C-PROGRAMM MIT SYSTEMAUFRUFEN

3.1 MKFILE

```
// Einbinden externer Bibliotheken.
#include <stdio.h>
#include <limits.h>
#include <fcntl.h>
#include <string.h>
// Einstiegspunkt für mkfile.
int main () {
	// Der Dateiname.
	char filename[30];
	// Die Länge des Dateinames.
	int filename_length;
	// Ausgabe der Zeichenkette "Name der neuen Datei: ".
	printf("Name der neuen Datei: ");
	// Liest die ersten 30 Zeichen, einschließlich des
	// Zeilenumbruchs die der Benutzer eingibt (aus der Standarteingabe stdin).
	fgets(filename, 31, stdin);
	// Ermittelt die Länge des eingegebenen Dateinames.
	filename_length = strlen(filename);
	// Prüft ob das letzte Zeichen des Dateinames ein
	// Zeilenumbruch ist.
	if( filename[filename_length-1] == '\n' ) {
		// Wenn ja, dann...
		// Das letzte Zeichen aus dem Dateinamen wird entfernt.
		filename[filename_length-1] = 0;
	}
	// Erstellt eine leere Datei mit Lese-/Schreib-/Ausführberechtigung
	// für den Benutzer.
	// Anschließend wird geprüft ob bei dem Erstellen der Datei
	// ein Fehler aufgetreten ist.
	if (creat(filename, 0700) == -1)
	{
		// Wenn ja, dann...
		// Gibt eine Fehlermeldung für den Fehlerfall aus.
		printf("Die Datei %s wurde nicht erfolgreich angelegt!\n", filename);
		// Gibt eine -1 für den Fehlerfall zurück.
		return -1;
	} else {
		// Gibt eine Meldung für den Erfolgsfall aus.
		printf("Die Datei %s wurde erfolgreich angelegt!\n", filename);
	}
	// Gibt eine 0 für den Erfolgsfall zurück.
	return 0;
}
```