

## Auswahl der Shell<sup>1</sup>

In den meisten Linux-Systemen ist die Standardshell die Bash-Shell. Um herauszufinden, welche Shell bei Ihnen als Standard-Login-Shell festgelegt ist, geben Sie die folgenden Befehle ein:

```
$ who am i
costis
$ grep costis /etc/passwd
costis:x:1000:1000:costis:/home/costis:/bin/bash
```

Beachten Sie, dass in diesen und allen weiteren Befehlszeilenbeispielen der Befehl gefolgt von der Ausgabe dieses Befehls angezeigt wird. Nach Abschluss des Befehls wird Ihnen wieder die Eingabeaufforderung angezeigt.

Der Befehl `who am i` zeigt Ihren Benutzernamen an, und der `grep`-Befehl (ersetzen Sie „costis“ durch Ihren Benutzernamen) zeigt die Definition Ihres Benutzerkontos in der Datei `/etc/passwd` an. Das letzte Feld in diesem Eintrag zeigt, dass die Bash-Shell (`/bin/bash`) Ihre Standardshell ist (die Shell, die gestartet wird, wenn Sie sich anmelden oder ein Terminalfenster öffnen).

Es ist möglich, wenn auch unwahrscheinlich, dass Sie eine andere Standardshell eingerichtet haben. Um eine andere Shell auszuprobieren, geben Sie einfach den Namen der gewünschten Shell ein (Beispiele sind `ksh`, `tcsh`, `csch`, `sh`, `dash` und andere, vorausgesetzt, sie sind installiert). Sie können in dieser Shell ein paar Befehle ausprobieren und `exit` eingeben, wenn Sie fertig sind, um zur Bash-Shell zurückzukehren.

## Gründe für die Auswahl unterschiedlicher Shells

Sie könnten sich aus folgenden Gründen für die Nutzung unterschiedlicher Shells entscheiden:

- Sie sind an die Nutzung von UNIX System V Systemen gewöhnt (oft standardmäßig `ksh`) oder an Sun Microsystems und andere Berkeley UNIX-basierte Distributionen (häufig standardmäßig `csch`) und fühlen sich mit den dortigen Standardshells wohler.
- Sie möchten Shell-Skripte ausführen, die für eine bestimmte Shell-Umgebung erstellt wurden. Dafür müssen Sie die Shell verwenden, für die diese Skripte geschrieben wurden, damit Sie sie aus Ihrer aktuellen Shell heraus testen oder verwenden können.
- Sie bevorzugen einfach bestimmte Funktionen einer Shell gegenüber einer anderen. Ein Mitglied meiner Linux User Group bevorzugt zum Beispiel `ksh` anstelle von `bash`, weil ihm die Art und Weise, wie Aliase in `bash` verwendet werden, nicht gefällt.

Obwohl die meisten Linux-Benutzer eine Vorliebe für eine bestimmte Shell haben, können Sie schnell lernen, eine andere Shell zu verwenden, wenn Sie bereits mit einer vertraut sind, indem Sie gelegentlich die `man`-Seite der Shell zurate ziehen (zum Beispiel `man bash`). Die `man`-Seiten (später im Abschnitt „Informationen zu Befehlen erhalten“ beschrieben) bieten Dokumentation zu Befehlen, Dateiformaten und anderen Komponenten in Linux. Die meisten Benutzer verwenden `bash`, einfach weil sie keinen besonderen Grund haben, eine andere Shell zu nutzen. Der Rest dieses Kapitels behandelt die `bash`-Shell.

---

<sup>1</sup> Linux® Bible, Tenth Edition. Christopher Negus. © 2020 John Wiley & Sons, Inc. Published 2020 by John Wiley & Sons, Inc.

Bash enthält Funktionen, die ursprünglich für die sh- und ksh-Shells in frühen UNIX-Systemen entwickelt wurden, sowie einige csh-Funktionen. Bash wird in den meisten Linux-Systemen, die Sie verwenden, als Standardshell eingerichtet sein, mit Ausnahme einiger spezialisierter Linux-Systeme (wie einige, die auf eingebetteten Geräten laufen), die eine kleinere Shell erfordern, die weniger Speicher benötigt und weniger Funktionen erfordert. Die meisten Beispiele in diesem Kapitel basieren auf der bash-Shell.

Der einfachste Weg, einen Befehl auszuführen, besteht darin, einfach den Namen des Befehls in einer Shell einzugeben. Öffnen Sie auf Ihrem Desktop ein Terminalfenster und geben Sie den folgenden Befehl ein:

```
$date  
Mon Nov 11 08:27:13 PM EET 2024
```

Durch Eingabe des `date`-Befehls ohne Optionen oder Argumente werden der aktuelle Tag, Monat, das Datum, die Uhrzeit, die Zeitzone und das Jahr angezeigt, wie oben gezeigt.

Hier sind einige weitere Befehle, die Sie ausprobieren können:

```
$ pwd  
/home/costis  
$ hostname costis-VirtualBox  
$ ls  
Desktop Downloads Pictures Templates Documents Music Public Videos
```

Der Befehl `pwd` zeigt Ihr aktuelles Arbeitsverzeichnis an. Die Eingabe von `hostname` zeigt den Hostnamen Ihres Computers an. Der Befehl `ls` listet die Dateien und Verzeichnisse in Ihrem aktuellen Verzeichnis auf. Obwohl viele Befehle nur durch Eingabe des Befehlsnamens ausgeführt werden können, ist es üblicher, nach dem Befehl weitere Zeichen einzugeben, um dessen Verhalten zu ändern. Diese Zeichen und Wörter, die Sie nach einem Befehl eingeben können, werden als Optionen und Argumente bezeichnet.

## **Befehlssyntax verstehen**

Die meisten Befehle haben eine oder mehrere Optionen, die hinzugefügt werden können, um das Verhalten des Befehls zu ändern. Optionen bestehen in der Regel aus einem einzelnen Buchstaben, der von einem Bindestrich (-) gefolgt wird. Sie können jedoch mehrere Ein-Buchstaben-Optionen zusammenfassen oder jedem eine eigene Option mit einem Bindestrich voranstellen, um mehrere Optionen gleichzeitig zu verwenden. Zum Beispiel sind die folgenden beiden Verwendungen von Optionen für den `ls`-Befehl gleichwertig:

```
$ ls -l -a -t  
$ ls -lat
```

In beiden Fällen wird der `ls`-Befehl mit den Optionen `-l` (lange Liste), `-a` (versteckte Punktdateien anzeigen) und `-t` (nach Zeit sortieren) ausgeführt.

Einige Befehle enthalten Optionen, die durch ein ganzes Wort dargestellt werden. Um einem Befehl mitzuteilen, dass ein ganzes Wort als Option verwendet werden soll, setzen Sie normalerweise zwei Bindestriche (- -) davor. Zum Beispiel geben Sie `--help` in die Befehlszeile ein, um die Hilfeoption bei vielen Befehlen zu verwenden. Ohne die doppelten Bindestriche würden die

Buchstaben **h**, **e**, **l** und **p** als separate Optionen interpretiert. Einige Befehle verwenden kein doppeltes Bindestrich-Schema, sondern nur einen einzelnen Bindestrich vor einem Wort. Die meisten Befehle verwenden jedoch doppelte Bindestriche für Wortoptionen.

### **Viele Befehle akzeptieren auch Argumente**

Viele Befehle akzeptieren auch Argumente, die entweder nach bestimmten Optionen oder am Ende der gesamten Befehlszeile eingegeben werden können. Ein Argument ist eine zusätzliche Information, wie zum Beispiel ein Dateiname, Verzeichnis, Benutzername, Gerät oder ein anderer Eintrag, der dem Befehl mitteilt, worauf er angewendet werden soll. Zum Beispiel zeigt `cat /etc/passwd` den Inhalt der Datei `/etc/passwd` auf Ihrem Bildschirm an. In diesem Fall ist `/etc/passwd` das Argument. Normalerweise können Sie so viele Argumente auf der Befehlszeile haben, wie Sie möchten, beschränkt nur durch die maximale Anzahl von Zeichen, die in einer Befehlszeile erlaubt sind.

Manchmal ist ein Argument mit einer Option verbunden. In diesem Fall muss das Argument direkt nach der Option stehen. Bei einstelligen Optionen folgt das Argument typischerweise nach einem Leerzeichen. Bei Optionen, die aus einem ganzen Wort bestehen, folgt das Argument häufig nach einem Gleichheitszeichen (=). Hier sind einige Beispiele:

```
$ ls -hide=Desktop
Documents Music Public Videos Downloads Pictures Templates
```

Hier ist ein Beispiel für eine einstellige Option, der ein Argument folgt:

```
$ tar -cvf backup.tar /home/costis
```

In diesem `tar`-Beispiel sagen die Optionen, dass eine Datei (**f**) namens `backup.tar` erstellt (**c**) werden soll, die den gesamten Inhalt des Verzeichnisses `/home/chris` und dessen Unterverzeichnisse einschließt und dabei ausführliche (**v**) Meldungen anzeigt, während die Sicherung erstellt wird. Da `backup.tar` ein Argument zur Option **f** ist, muss `backup.tar` unmittelbar nach der Option stehen.

Hier sind einige Befehle, die Sie ausprobieren können. Sehen Sie, wie sie sich mit unterschiedlichen Optionen unterschiedlich verhalten:

```
$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
$ ls -a
. Desktop .gnome2_private .lessht Public
.. Documents .gnote .local Templates
.bash_history Downloads .gnupg .mozilla Videos
.bash_logout .emacs .gstreamer-0.10 Music .xsession-errors
.bash_profile .esd_auth .gtk-bookmarks Pictures .zshrc
.bashrc .fsync .log .gvfs Pictures
```

- Der Befehl `ls` alleine zeigt alle regulären Dateien und Verzeichnisse im aktuellen Verzeichnis an.
- Mit `-a` sehen Sie auch die versteckten Dateien im Verzeichnis (die mit einem Punkt beginnen).

```
$ uname
```

```
Linux
```

```
$ uname -a
```

```
Linux costis-VirtualBox 6.8.0-48-generic #48-Ubuntu SMP PREEMPT_DYNAMIC Fri Sep 27  
14:04:52 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

- Der Befehl `uname` zeigt den Typ des Systems an, das Sie verwenden (z. B. `Linux`).
- Mit `-a` sehen Sie zusätzlich den Hostnamen, die Kernel-Version und die Kernel-Release-Nummer.

```
$ date
```

```
Mon Nov 11 08:27:13 PM EET 2024
```

```
$ date +%d/%m/%y'
```

```
11/11/24
```

```
$ date +%d.%m.%y'
```

```
11.11.24
```

```
$ date +%A, %B %d, %Y'
```

```
Monday, November 11, 2024
```

- Der Befehl `date` zeigt das aktuelle Datum und die Uhrzeit an.
- Mit den Optionen wie  `+%d/%m/%y '` oder  `+%A, %B %d, %Y '` können Sie das Datum im gewünschten Format anzeigen lassen.

```
$ id
```

```
uid=1000(costis) gid=1000(costis)
```

```
groups=1000(costis),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),122(lpadmin),135(lxd),136(sambashare)
```

## Befehle finden

Nachdem Sie nun einige Befehle eingegeben haben, fragen Sie sich vielleicht, wo diese Befehle gespeichert sind und wie die Shell die von Ihnen eingegebenen Befehle findet. Um Befehle zu finden, die Sie eingeben, durchsucht die Shell das sogenannte „Pfad“ (PATH). Für Befehle, die nicht in Ihrem Pfad enthalten sind, können Sie den vollständigen Speicherort des Befehls eingeben.

Wenn Sie das Verzeichnis kennen, das den Befehl enthält, den Sie ausführen möchten, können Sie den Befehl durch Eingabe des vollständigen oder absoluten Pfads ausführen. Beispielsweise können Sie den Befehl `date` aus dem `/bin`-Verzeichnis wie folgt ausführen:

```
$ /bin/date
```

Das kann jedoch umständlich sein, besonders wenn der Befehl in einem Verzeichnis mit einem langen Pfad gespeichert ist. Eine bessere Methode ist, Befehle in bekannten Verzeichnissen zu

speichern und diese Verzeichnisse zur Umgebungsvariablen `PATH` der Shell hinzuzufügen. Der `PATH` besteht aus einer Liste von Verzeichnissen, die nacheinander nach den eingegebenen Befehlen durchsucht werden. Um Ihren aktuellen `PATH` anzuzeigen, geben Sie Folgendes ein:

```
$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Die Ausgabe zeigt einen typischen Standardpfad für einen regulären Linux-Benutzer. Die Verzeichnisse in der Pfadliste sind durch Doppelpunkte getrennt. Die meisten Benutzerbefehle, die mit Linux geliefert werden, sind im Verzeichnis `/bin`, `/usr/bin` oder `/usr/local/bin` gespeichert. Die Verzeichnisse `/sbin` und `/usr/sbin` enthalten administrative Befehle (einige Linux-Systeme fügen diese Verzeichnisse nicht in die Pfade regulärer Benutzer ein).

```
$ history 8
```

```
382 date
```

```
383 ls /usr/bin | sort -a | more
```

```
384 man sort
```

```
385 cd /usr/local/bin
```

```
386 man more
```

```
387 useradd -m /home/chris -u 101 chris
```

```
388 passwd chris
```

```
389 history 8
```

```
!n      # Run command number.
```

```
$ cat /etc/passwd | sort | less
```

```
$ gunzip < /usr/share/man/man1/grep.1.gz | nroff -c -man | less
```

```
$ alias p='pwd ; ls -CF'
```

```
$ alias rm='rm -i'
```