ADMIN:DOCS

LPIC-1: Die Linux-Kommandozeile verstehen: Shell, Terminal und erste Befehle

Von der Shell bis zum ersten Login: Entdecke die Macht der Linux-Kommandozeile und lege das Fundament für deine LPIC-1-Zertifizierung und Linux-Administration.

Kategorie: LPIC-1 Serie **Veröffentlicht:** 13.05.2025 **Zuletzt geprüft:** 02.11.2026

Willkommen zum ersten Teil unserer umfassenden LPIC-1-Serie. In diesem Artikel widmen wir uns einem der fundamentalen Themenblöcke aus dem LPIC-1 Prüfungsmodul 101: den GNU- und Unix-Befehlen unter Linux. Wenn du dich auf dem Weg zur Zertifizierung als Linux-Administrator befindest oder einfach deine Linux-Kenntnisse vertiefen möchtest, bist du hier genau richtig.

Über LPIC-1

Die LPIC-1-Zertifizierung (Linux Professional Institute Certification Level 1) ist der Einstieg in die Welt der professionellen Linux-Administration. Sie gliedert sich in die beiden Prüfungsmodule 101 und 102. Unsere Artikelserie deckt schrittweise alle relevanten Themenbereiche ab, die du für beide Module benötigst – immer mit dem Fokus auf praktische Anwendbarkeit und verständliche Erklärungen.

> **Wichtiger Hinweis:** Diese Artikelserie ersetzt keinen offiziellen Prüfungsvorbereitungskurs für die LPIC-1-Zertifizierung. Sie dient vielmehr als praxisorientierte, didaktisch aufbereitete Ergänzung für dein Selbststudium und soll dir helfen, die teils komplexen Inhalte besser zu verstehen und anzuwenden.

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Zielgruppe der Artikelserie

Dieser Artikel richtet sich an:

- Linux-Einsteiger mit technischem Interesse, die ihre Kommandozeilenkenntnisse systematisch aufbauen möchten
- Angehende IT-Profis, die gezielt auf die LPIC-1-Zertifizierung hinarbeiten
- Umsteiger aus anderen IT-Bereichen, die ihre Linux-Kenntnisse auf ein solides Fundament stellen wollen
- Praktiker, die Wert auf nachvollziehbare Beispiele und Erklärungen legen, statt nur theoretisches Wissen anzuhäufen

Didaktischer Aufbau & Praxisfokus

Der Artikel ist didaktisch so aufgebaut, dass du Schritt für Schritt durch die Befehlswelt von Linux geführt wirst – vom einfachen Navigieren im Dateisystem bis hin zu komplexeren Aufgaben wie Prozess- und Rechteverwaltung.

Dabei setzen wir auf ein durchdachtes Konzept:

- Jeder Befehl wird nicht nur erklärt, sondern auch mit praktischen Beispielen veranschaulicht
- Wir beleuchten, **warum** bestimmte Befehle wichtig sind und **wofür** du sie in der Praxis einsetzen wirst
- Spezielle Hinweise zeigen dir, worauf bei der LPIC-1-Prüfung besonders geachtet wird

> **Hinweis zu den visuellen Orientierungshilfen:** In diesem Kurs unterstützen dich farblich hervorgehobene Infoboxen mit Praxistipps, Warnungen vor Stolperfallen und praktischen Übungen bei der optimalen Vorbereitung auf die LPIC-1-Prüfung.

Wie du am besten mit diesem Artikel arbeitest

Für den größten Lernerfolg empfehle ich dir, parallel zum Lesen eine Linux-Umgebung zu nutzen – sei es eine virtuelle Maschine, ein Docker-Container oder eine native Installation. Linux lernt man am besten durch praktisches Ausprobieren! Teste die Befehle, experimentiere mit Parametern und beobachte genau, was passiert.

So, genug der Vorrede – lass uns eintauchen in die Welt der Linux-Kommandozeile, die nicht nur das Herzstück jeder LPIC-1-Prüfung bildet, sondern auch dein tägliches Handwerkszeug als angehender Linux-Administrator sein wird.

Linux-Kommandozeile verstehen

Was ist die Shell?

Die Shell ist das Herzstück der Interaktion mit deinem Linux-System – eine textbasierte Schnittstelle zwischen dir als Benutzer und dem Betriebssystemkern (Kernel). Anders als bei grafischen Benutzeroberflächen, bei denen du mit Mausklicks arbeitest, gibst du in der Shell textbasierte Befehle ein, die dann vom System ausgeführt werden.

□□ Praktisches Beispiel:

Wenn du einen Befehl wie `ls` eingibst und Enter drückst, passiert folgendes:

1. Du gibst ein: `ls`
2. Die Shell interpretiert: „Der Benutzer möchte den Inhalt des aktuellen Verzeichnisses sehen“
3. Die Shell sucht nach dem Programm `ls`
4. Das Programm wird ausgeführt
5. Das Ergebnis wird auf dem Bildschirm angezeigt.

Der Name "Shell" (englisch für "Hülle" oder "Schale") ist übrigens kein Zufall. Sie umhüllt den Kernel – den eigentlichen Kern des Betriebssystems – wie eine Schale und schützt ihn vor direkten Benutzereingriffen.

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Die Shell als Übersetzer und Vermittler

Die Shell erfüllt mehrere entscheidende Funktionen:

- **Befehlsinterpretation**: Sie versteht und interpretiert deine eingegebenen Befehle
- **Programmausführung**: Sie startet Programme und verwaltet deren Ausführung
- **Ein- und Ausgabesteuerung**: Sie leitet Daten zwischen Programmen und Standardgeräten
- **Skriptausführung**: Sie kann Folgen von Befehlen (Skripte) automatisiert ausführen
- **Fehlerbehandlung**: Sie meldet Erfolg oder Misserfolg von Befehlen zurück

> □□ **Tipp:** Die Shell hat auch ein "Gedächtnis" – mit der Pfeiltaste nach oben kannst du durch deine zuletzt eingegebenen Befehle navigieren. Das spart viel Tipparbeit, besonders bei langen oder komplexen Befehlen!

Warum ist die Shell für Linux so wichtig?

In der Windows-Welt wird die Kommandozeile oft als letztes Mittel gesehen, wenn die grafische Oberfläche nicht weiterhilft. In Linux ist das anders:

- Die Shell bietet oft den **schnellsten und effizientesten Weg**, Aufgaben zu erledigen
- Viele **Administrationsaufgaben** sind über grafische Oberflächen nur eingeschränkt möglich
- **Automatisierung** komplexer Aufgaben ist durch Shell-Skripte einfach umzusetzen
- Die Shell bietet **konsistente Funktionalität** über verschiedene Linux-Distributionen hinweg
- Bei **Serverinstallationen** ist oft gar keine grafische Oberfläche vorhanden

> ⚠ **Achtung:** Als Linux-Administrator wirst du einen erheblichen Teil deiner täglichen Arbeit in der Shell verbringen. Daher ist ein sicherer Umgang damit absolute Grundvoraussetzung für die LPIC-1-Zertifizierung!

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Die Shell in Aktion

Die meisten Einsteiger sind anfangs überrascht, wie mächtig und effizient die Shell sein kann.

Praxis-Szenario: Filterung und Batch-Kopieren

Du möchtest alle Textdateien finden, die das Wort "`wichtig`" enthalten und diese in ein neues Verzeichnis kopieren. Mit grafischen Werkzeugen wäre das eine langwierige Aufgabe mit vielen Klicks.

In der Shell:

```
mkdir wichtige_dateien
find . -type f -name "*.txt" -exec grep -l "wichtig" {} \; -exec cp {} wichtige_dateien/
\;
```

Diese zwei Zeilen erledigen die komplette Aufgabe – vom Erstellen des Zielverzeichnisses bis zum Auffinden und Kopieren aller relevanten Dateien.

Interaktion mit der Shell

Die grundlegende Interaktion mit der Shell folgt einem einfachen Muster:

Die Shell zeigt einen **Prompt** (Eingabeaufforderung), der typischerweise so aussieht:

```
username@hostname:~/verzeichnis$
```

Du gibst einen `Befehl` ein (eventuell mit Optionen und Argumenten):

```
ls -la /home
```

Du drückst `Enter` zur Ausführung, und die Shell zeigt das `Ergebnis` an und präsentiert dann wieder den Prompt für den nächsten Befehl.

> **Typischer Anfängerfehler:** Linux-Befehle sind case-sensitive (Groß-/Kleinschreibung wird unterschieden). Der Befehl `LS` ist nicht dasselbe wie `ls` und wird zu einer Fehlermeldung führen.

Der Stellenwert der Shell in der Praxis

Die Beherrschung der Shell ist für jede Form der Linux-Administration unverzichtbar. Für die LPIC-1-Prüfung brauchst du ein solides Verständnis der Shell-Grundlagen, da diese die Basis für fast alle weiteren Themengebiete bilden – vom Dateisystem über Benutzerverwaltung bis hin zu Netzwerkkonfiguration.

> **Tipp für die Prüfungsvorbereitung:** Die LPIC-1-Prüfung enthält zahlreiche Fragen, die direkt oder indirekt mit der Shell zusammenhängen. Achte besonders auf die Befehlssyntax, Parametereingabe und die Bedeutung verschiedener Shell-Funktionen.

Das Verständnis der Shell als zentrale Schnittstelle zwischen Benutzer und System ist der Schlüssel zu deinem Erfolg mit Linux. In den nächsten Abschnitten werden wir uns verschiedene Shell-Arten anschauen und deine ersten praktischen Schritte in der Kommandozeile gehen.

Bash vs. andere Shells

Die Bash (Bourne Again Shell) ist die häufigste Shell in den meisten Linux-Distributionen – aber sie ist längst nicht die einzige. Verschiedene Shells haben unterschiedliche Stärken und Schwächen, die du kennen solltest, besonders im Hinblick auf die LPIC-1-Prüfung.

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Die Bash – Der Linux-Standard

Die **Bash** (Bourne Again SHell) wurde 1989 von Brian Fox als freie Erweiterung der ursprünglichen Bourne Shell (sh) entwickelt. Sie hat sich aus gutem Grund als Standard in den meisten Linux-Distributionen durchgesetzt:

- Umfangreiche **Befehlszeilenbearbeitung** (editieren mit Pfeiltasten, Historyfunktionen)
- Leistungsstarke **Tab-Vervollständigung** für Befehle und Dateinamen
- Flexible **Aliase** und **Funktionen** zur Arbeitsoptimierung
- **Skriptfunktionen** mit Variablen, Schleifen, Bedingungen und mehr
- Anpassbare **Prompt**-Darstellung

□□ Praktisches Beispiel Bash-Features:

```
# Tab-Vervollständigung
cd /ho[TAB]  # vervollständigt zu "/home/"

# Befehlshistorie
history      # zeigt letzte Befehle
!42          # führt Befehl Nr. 42 aus der History aus

# Alias für häufige Befehle
alias ll='ls -la'
ll           # führt 'ls -la' aus
```

Wichtige alternative Shells

1. Die Bourne Shell (sh)

Die ursprüngliche Unix-Shell von Stephen Bourne ist die Mutter aller modernen Shells und heute meist als symbolischer Link auf eine andere Shell implementiert:

- Maximale **Kompatibilität** für Skripte
- **Minimalistische** Funktionalität

- Kaum interaktive Features

> **Tipp:** Wenn du Shell-Skripte schreibst, die auf vielen Systemen laufen sollen, verwende /bin/sh-Syntax für maximale Portabilität.

2. Die Z Shell (zsh)

Die Z-Shell hat in den letzten Jahren stark an Popularität gewonnen und ist inzwischen die Standardshell in macOS:

- Erweiterte **Tab-Vervollständigung** und Rechtschreibkorrektur
- **Themenanpassung** mit Frameworks wie Oh-My-Zsh
- Verbesserte **Globbing**-Fähigkeiten (Dateinamensmuster)
- Bash-**Kompatibilität** bei erweiterten Features

3. Die Korn Shell (ksh)

Entwickelt von David Korn bei AT&T, kombiniert sie POSIX-Konformität mit erweiterten Funktionen:

- Starker Fokus auf **Skriptsprachen**-Fähigkeiten
- Gute **Performance** auch bei komplexen Skripten
- Wird häufig in **Enterprise**-Umgebungen eingesetzt

4. Die C Shell (csh) und TENEX C Shell (tcsh)

Diese Shells verwenden eine C-ähnliche Syntax und bieten spezielle Funktionen:

- **Syntax** angelehnt an die Programmiersprache C
- Erweiterte **Aliasing**-Funktionen
- Nützliche **Job-Control**-Features

> ⚠ **Achtung:** Die Skriptsyntax der C Shell unterscheidet sich erheblich von anderen Shells! Skripte sind nicht ohne Weiteres portierbar.

5. Die Friendly Interactive Shell (fish)

Eine moderne Shell mit Fokus auf Benutzerfreundlichkeit:

- Automatische **Syntaxhervorhebung**
- Intelligente **Vorschläge** basierend auf der Befehlshistorie
- Intuitives **Webinterface** zur Konfiguration
- Geringere **Kompatibilität** zu traditionellen Shells

Aktuelle Shell ermitteln und wechseln

Es gibt mehrere Wege, herauszufinden, welche Shell du gerade benutzt:

```
# Anzeige der aktuellen Shell
echo $SHELL
/bin/bash

# Oder alternative Methode
ps -p $$
PID  TTY          TIME CMD
1234 pts/0        00:00:00 bash
```

Zwischen Shells wechseln

Das Wechseln zwischen verschiedenen Shells ist einfach – gib einfach den Namen der gewünschten Shell ein:

```
bash    # Wechselt zur Bash
zsh     # Wechselt zur Z-Shell
sh      # Wechselt zur Bourne-Shell
```

Um deine Standard-Shell dauerhaft zu ändern, verwendest du den Befehl `chsh`:

```
chsh -s /bin/zsh
```

> **Typischer Fehler:** Vergiss nicht, dass nach dem Wechsel der Standard-Shell ein erneutes Einloggen nötig ist, damit die Änderung wirksam wird!

Shell-Skript-Kompatibilität & Shebang

Ein wichtiger Aspekt, besonders für Systemadministratoren, ist die unterschiedliche Skript-Syntax der verschiedenen Shells:

```
# Bash/sh-Stil (POSIX-konform)
for i in 1 2 3; do
    echo $i
done

# C-Shell-Stil
foreach i (1 2 3)
    echo $i
```

```
end
```

> **Prüfungstipp:** Für die LPIC-1 solltest du dich hauptsächlich auf die Bash und ihre Features konzentrieren, aber auch die grundlegenden Unterschiede zu anderen Shells kennen. Die Prüfung enthält oft Fragen zur Unterscheidung zwischen POSIX-konformen Shells (sh, bash, ksh) und C-ähnlichen Shells (csh, tcsh).

Die Shebang-Zeile am Skriptanfang

Die sogenannte Shebang-Zeile am Anfang eines Skripts bestimmt, mit welcher Shell es ausgeführt wird:

```
#!/bin/bash    # Wird mit Bash ausgeführt
#!/bin/sh      # Wird mit sh ausgeführt
#!/bin/zsh     # Wird mit Z-Shell ausgeführt
```

> ⚠ **Wichtig für die Praxis:** Wenn ein Skript mit anderen Shells kompatibel sein soll, verwende #!/bin/sh und halte dich an POSIX-konforme Konstrukte. Für fortgeschrittene Funktionen, die nur in bestimmten Shells verfügbar sind, wähle die entsprechende Shell explizit.

Dieses Wissen über verschiedene Shells und ihre Eigenschaften ist nicht nur theoretisch wichtig – in der Linux-Administration wirst du je nach Einsatzgebiet mit unterschiedlichen Shell-Umgebungen konfrontiert. Die Bash ist zwar am weitesten verbreitet, aber gerade in etablierten Unternehmensumgebungen oder spezialisierten Einsatzgebieten können auch andere Shells zum Einsatz kommen.

Terminal-Emulator vs. virtuelle Konsole

Wenn du mit der Linux-Kommandozeile arbeitest, kannst du dies auf zwei grundsätzlich verschiedene Arten tun: über einen Terminal-Emulator innerhalb einer grafischen Benutzeroberfläche oder über eine virtuelle Konsole. Obwohl beide den Zugang zur Shell ermöglichen, gibt es wichtige Unterschiede, die jeder Linux-Administrator kennen sollte.

Die virtuelle Konsole (TTY) – Der direkte Weg

Virtuelle Konsolen (auch TTYs genannt) sind direkte Zugangspunkte zur Shell ohne grafische Oberfläche dazwischen. Sie sind besonders wichtig für die Systemadministration und Fehlerbehebung.

Hardware → Kernel → Virtuelle Konsole → Shell

So greifst du auf virtuelle Konsolen zu:

```
□□**Praktisches Beispiel:**
Drücke Strg+Alt+F1 bis Strg+Alt+F6, um zwischen verschiedenen virtuellen Konsolen zu
wechseln. Mit Strg+Alt+F7 (oder in manchen Distributionen F1 oder F2) kehrst du zur
grafischen Oberfläche zurück.
```

Eigenschaften der virtuellen Konsole:

- Funktioniert auch, wenn die grafische Oberfläche abstürzt oder nicht verfügbar ist
- Benötigt minimale Systemressourcen
- Begrenzte Darstellungsmöglichkeiten (meist nur Text, eingeschränkte Farbunterstützung)
- Standardmäßig 6 virtuelle Konsolen verfügbar (tty1 bis tty6)

> ⚠ **Achtung:** Als Linux-Administrator musst du mit virtuellen Konsolen vertraut sein! Sie sind oft deine letzte Rettung, wenn die grafische Oberfläche nicht mehr funktioniert.

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Der Terminal-Emulator – Grafisch und flexibel

Terminal-Emulatoren sind Programme, die innerhalb einer grafischen Benutzeroberfläche laufen und das Verhalten eines klassischen Terminals nachahmen (oder "emulieren").

Hardware → Kernel → Grafische Oberfläche → Terminal-Emulator → Shell

Gängige Terminal-Emulatoren:

- **GNOME Terminal** (Standard in GNOME-Umgebungen)
- **Konsole** (KDE-Umgebungen)
- **xterm** (klassischer X11-Terminal)
- **rxvt** und **urxvt** (leichtgewichtige Alternativen)
- **Terminator** (mit Splitting-Funktion)
- **XFCE Terminal** (in XFCE-Umgebungen)
- **Alacritty** (GPU-beschleunigter Terminal)

Eigenschaften von Terminal-Emulatoren:

- Bieten moderne Funktionen wie Tabs, Splits, transparente Hintergründe
- Unterstützen umfangreiche Farbschemen und Anpassungen
- Erlauben Copy & Paste mit der Maus
- Lassen sich mit der grafischen Oberfläche integrieren (Drag & Drop)
- Benötigen eine laufende grafische Umgebung

> □□ **Tipp:** Die meisten Terminal-Emulatoren lassen sich mit Tastenkombinationen wie Strg+Shift+T um neue Tabs erweitern – viel praktischer als mehrere Fenster zu öffnen!

Technische Architektur im Vergleich

Um die Unterschiede wirklich zu begreifen, hilft ein Blick auf die technische Architektur:

Virtuelle Konsole (TTY):

1. **Hardware-Eingabe & Ausgabe**Tastatur- und Monitorsignale werden direkt über BIOS/UEFI an den Kernel übergeben
2. **Kernel-TTY-Treiber**Der Linux-Kernel stellt virtuelle Textterminals über die Gerätedateien /dev/tty1 bis /dev/tty6 bereit
3. **Login-Initialisierung**Der Systemdienst getty (agetty) initialisiert das Terminal und startet den Login-Prompt
4. **Interaktive Shell**Nach erfolgreicher Authentifizierung startet die Login-Shell (z. B. /bin/bash) als direkter Kindprozess

Terminal-Emulator (PTY):

1. **Eingabe & Display-Server**Tastatur- und Mausevents laufen über den grafischen Display-Server (Wayland oder X11)
2. **Terminal-Emulator-Fenster**Die grafische Anwendung (z. B. GNOME Terminal, Konsole, Alacritty) rendert Schrift und Fenster
3. **Pseudo-Terminal-Paar (PTY)**Kommunikation erfolgt über Multiplexer (/dev/ptmx) und Slave-Endpunkt (/dev/pts/X)
4. **Interaktive Shell**Die Shell läuft als Kindprozess des Pseudo-Terminals und verarbeitet Befehle textbasiert

Die Unterschiede haben auch praktische Auswirkungen:

- **Mausunterstützung:** In virtuellen Konsolen ist sie begrenzt oder nicht vorhanden
- **Grafikfähigkeit:** Terminal-Emulatoren können Bilder anzeigen, virtuelle Konsolen nicht
- **Tastaturbelegung:** Kann in beiden Umgebungen unterschiedlich sein
- **Zugriffsrechte:** Virtuelle Konsolen haben oft direkteren Zugriff auf Hardware

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

> **Typischer Anfängerfehler:** Viele Linux-Neulinge geraten in Panik, wenn sie versehentlich in eine virtuelle Konsole wechseln. Merke dir: `Strg+Alt+F7` (oder F1/F2) bringt dich zurück zur grafischen Oberfläche!

Wann nutzt man was?

Die Wahl zwischen Terminal-Emulator und virtueller Konsole hängt vom Anwendungsfall ab:

Virtuelle Konsole verwenden für:

- Systemreparatur, wenn die grafische Oberfläche nicht startet
- Niedrige Ressourcennutzung auf älteren Systemen
- Serveradministration ohne grafische Umgebung
- Direktere Hardwarezugriffe

Terminal-Emulator verwenden für:

- Alltägliche Administration parallel zu anderen grafischen Anwendungen
- Arbeiten mit mehreren Terminal-Sitzungen (Tabs)
- Kopieren/Einfügen zwischen Terminal und anderen Anwendungen
- Angepasste Darstellung mit Themes und Farbschemata

Besonderheiten in der Praxis

In beiden Umgebungen arbeitest du mit der gleichen Shell, aber es gibt einige Unterschiede im Verhalten:

- **Umgebungsvariablen:** In Terminal-Emulatoren sind oft mehr Umgebungsvariablen gesetzt
- **Farbunterstützung:** Terminal-Emulatoren unterstützen meist 256 Farben oder mehr, virtuelle Konsolen oft nur 8 oder 16
- **Zeichensatzunterstützung:** Terminal-Emulatoren bieten bessere Unicode- und Sonderzeichenunterstützung

Praktisches Beispiel:

Quelle: https://admindocs.de/de/lpic-1-serie/lpic-1-serie-die-linux-kommandozeile-verstehen-shell-terminal-und-erste-befehle

Um zu sehen, in welcher Umgebung du dich befindest:

```
tty
/dev/pts/0    # Terminal-Emulator (pseudoterminal)
```

oder

```
tty
/dev/tty1     # Virtuelle Konsole
```

Relevanz für die LPIC-1-Prüfung

Für die LPIC-1-Zertifizierung solltest du:

- Den Unterschied zwischen beiden Zugangsmethoden kennen
- Wissen, wie man zwischen virtuellen Konsolen wechselt
- Die wichtigsten Terminal-Emulatoren und ihre Grundfunktionen kennen
- Verstehen, wann welche Methode angemessen ist

> □□ **Prüfungstipp:** In der Prüfung kann es Fragen dazu geben, wie du in einer problematischen Situation auf das System zugreifst, wenn die grafische Oberfläche nicht funktioniert. Die Antwort ist fast immer: über die virtuelle Konsole!

Eine praktische Fähigkeit, die du entwickeln solltest, ist die Flexibilität, in beiden Umgebungen effektiv zu arbeiten. Als Linux-Administrator wirst du in manchen Situationen keine Wahl haben

und musst mit den verfügbaren Werkzeugen zurechtkommen – sei es ein grafischer Terminal oder die pure Konsole.

Erste Schritte: Einloggen und Navigation

Jede Linux-Reise beginnt mit dem Login und den ersten Navigationsschritten im System. Dieser Abschnitt führt dich durch diese fundamentalen Prozesse – vom Anmelden bis zum sicheren Erkunden deines Systems.

Der Login-Prozess (TTY vs. Display-Manager)

Je nach Konfiguration deines Systems präsentiert sich der Login-Prozess unterschiedlich:

Grafische Anmeldung: Bei den meisten Desktop-Distributionen erscheint ein grafischer Login-Bildschirm, der Benutzername und Passwort abfragt. Nach erfolgreicher Anmeldung kannst du einen Terminal-Emulator öffnen, um zur Kommandozeile zu gelangen.

Anmeldung in der Konsole: Serverinstallationen oder minimale Systeme zeigen oft direkt eine textbasierte Login-Aufforderung:

```
Ubuntu 24.04 LTS tty1
login: username
Password: **********
```

□□ Praktisches Beispiel:

Bei der Eingabe des Passworts erscheinen keine Zeichen oder Sternchen – das ist normal und ein Sicherheitsfeature! Deine Eingabe wird dennoch registriert.

> ⚠ **Achtung:** Nach mehreren fehlgeschlagenen Loginversuchen sperrt Linux den Zugang temporär, um Brute-Force-Angriffe abzuwehren. Bei Problemen warte einige Minuten,

> bevor du es erneut versuchst.

Aufbau des Shell-Prompts

Nach erfolgreicher Anmeldung begrüßt dich der Shell-Prompt, der in etwa so aussieht:

```
username@hostname:~/verzeichnis$
```

Dieser Prompt enthält wertvolle Informationen:

- `username`: Dein aktueller Benutzername
- `@hostname`: Der Name deines Systems
- `~/verzeichnis`: Dein aktuelles Arbeitsverzeichnis (~ steht für dein Home-Verzeichnis)
- `$`: Zeigt an, dass du als normaler Benutzer angemeldet bist (bei root-Rechten steht hier #)

> □□ **Tipp:** Der Prompt ist anpassbar! Später in der LPIC-Serie lernst du, wie du ihn nach deinen Vorstellungen gestalten kannst.

Erste Orientierung im System (whoami, pwd, ls)

Nach dem Login ist es wichtig, sich zu orientieren. Hier sind die essentiellen ersten Befehle:

```
whoami
username
```

□□Zeigt dir, als welcher Benutzer du eingeloggt bist – besonders nützlich, wenn du zwischen Benutzerkonten wechselst.

```
pwd
/home/username
```

□□**p**rint **w**orking **d**irectory – zeigt dir, in welchem Verzeichnis du dich gerade befindest.

```
ls
Dokumente  Downloads  Bilder  Videos
```

□□**l**i**s**t – listet Dateien und Verzeichnisse im aktuellen Ordner auf.

```
ls -l

total 16
drwxr-xr-x 2 username users 4096 May 10 15:20 Dokumente
drwxr-xr-x 3 username users 4096 May 12 09:45 Downloads
drwxr-xr-x 2 username users 4096 Apr 30 18:10 Bilder
drwxr-xr-x 2 username users 4096 Apr 30 18:10 Videos
```

Mit der Option -l erhältst du detaillierte Informationen zu jeder Datei und jedem Verzeichnis.

Typischer Fehler: Neulinge vergessen oft, dass Linux zwischen Groß- und Kleinschreibung unterscheidet. LS ist nicht dasselbe wie ls. Wenn du eine Fehlermeldung wie command not found erhältst, überprüfe die Schreibweise!

Navigation im Dateisystem mit cd

Um dich im Dateisystem zu bewegen, verwendest du den Befehl `cd` (**c**hange **d**irectory):

```
cd Dokumente
pwd
/home/username/Dokumente
```

Mit `cd` ohne Argumente kehrst du immer zu deinem Home-Verzeichnis zurück:

```
cd
pwd
/home/username
```

In der Verzeichnisstruktur navigieren:

```
cd ..          # Eine Ebene nach oben
cd ../..       # Zwei Ebenen nach oben
cd /           # Zum Root-Verzeichnis
cd ~           # Zum Home-Verzeichnis (gleich wie "cd" ohne Argumente)
cd -           # Zum vorherigen Verzeichnis zurück
```

Praktisches Beispiel:

```
pwd
/home/username
cd /etc
pwd
/etc
cd -
/home/username
```

Hilfe auf der Kommandozeile finden (man & --help)

Eine der wichtigsten Fähigkeiten ist es, zu wissen, wie man Hilfe bekommt:

Der man-Befehl (für Manual/Handbuch):

```
man ls
```

> Öffnet die ausführliche Manualseite für den Befehl ls.

Die --help Option:

```
ls --help
```

> Zeigt eine kurze Hilfeinformation direkt in der Shell an.

Der info-Befehl (detaillierter als man):

```
info ls
```

> Bietet oft strukturiertere und umfassendere Dokumentation.

Die whatis-Abfrage (Kurzbeschreibung):

```
whatis ls
ls (1)               - list directory contents
```

> **Tipp:** In man-Seiten navigierst du mit den Pfeiltasten. Drücke q zum Beenden und / gefolgt von einem Suchbegriff zum Suchen.

> ⚠ **Wichtig:** Lerne, die Manpages zu lesen! Sie sehen anfangs komplex aus, enthalten aber die zuverlässigste Information direkt vom System.

Typische Fehlersituationen und Lösungsansätze

Neulingen passiert häufig folgendes:

```
cd Dokumente
bash: cd: Dokumente: No such file or directory
```

Diese Fehlermeldung bedeutet, dass das Verzeichnis "Dokumente" im aktuellen Arbeitsverzeichnis nicht existiert.

Um das Problem zu lösen:

- Überprüfe mit pwd, wo du dich befindest
- Liste mit ls die verfügbaren Verzeichnisse
- Achte auf die exakte Schreibweise (inkl. Groß-/Kleinschreibung)
- Versuche es erneut oder verwende absolute Pfade: cd ~/Dokumente

Nützliche Basisbefehle für den Einstieg

Zum Abschluss hier noch einige grundlegende Befehle für deine ersten Schritte:

```
date          # Aktuelles Datum und Uhrzeit anzeigen
cal           # Kalender des aktuellen Monats anzeigen
clear         # Bildschirm leeren (oder Strg+L)
```

```
exit          # Shell beenden und ausloggen (oder Strg+D)
history       # Liste der zuletzt eingegebenen Befehle
```

□□ Praktisches Beispiel:

```
history | grep cd
123  cd Dokumente
145  cd ..
156  cd /etc
```

> □□Zeigt alle cd-Befehle aus deiner History an – so findest du schnell zuvor besuchte Verzeichnisse wieder.

Warum ist das für die LPIC-1 wichtig?

Die Navigation und grundlegende Orientierung im System sind fundamentale Fähigkeiten für jeden Linux-Administrator. In der LPIC-1-Prüfung wirst du damit konfrontiert:

- Befehle zum Erkunden des Dateisystems (pwd, ls, cd)
- Verschiedene Methoden, um Hilfe zu bekommen (man, --help, info)
- Unterschiede zwischen relativen und absoluten Pfaden
- Grundlegende Shell-Funktionen und Tastenkombinationen

> □□ **Prüfungstipp:** Die LPIC-1-Prüfung enthält oft praktische Szenarien, bei denen du bestimmte Dateien im System finden oder in spezifische Verzeichnisse navigieren musst. Übe diese Grundlagen, bis sie in Fleisch und Blut übergehen!

Das brauchst du später, um komplexere Aufgaben wie Systemkonfiguration, Softwareinstallation oder Troubleshooting effizient durchführen zu können. Jede anspruchsvollere Linux-Aufgabe baut auf diesen grundlegenden Navigationsfähigkeiten auf.

LPIC-1 Prüfungsinformationen & Vorbereitung

Zum Abschluss unseres ersten Artikels erhältst du hier die wichtigsten Eckdaten zur Durchführung und Vorbereitung auf die offizielle LPIC-1-Zertifizierung.

Prüfungsaufbau (Exam 101 & Exam 102)

Die LPIC-1-Zertifizierung besteht aus zwei separaten Prüfungen:

- **LPIC-1 Exam 101**: Deckt Systemarchitektur, Linux-Installation und Paketverwaltung, GNU- und Unix-Befehle sowie Geräte und Dateisysteme ab
- **LPIC-1 Exam 102**: Behandelt Shells und Shell-Skripte, Benutzerschnittstellen, administrative Aufgaben, Systemdienste, Netzwerkgrundlagen und Sicherheit

Jede Prüfung umfasst 60 Fragen (Multiple-Choice und Lückentextaufgaben) und muss innerhalb von 90 Minuten absolviert werden. Zum Bestehen benötigst du mindestens 500 von 800 möglichen Punkten

Prüfungsabnahme, Kosten und Anmeldung

Du hast zwei Hauptoptionen, um die LPIC-1-Prüfungen abzulegen:

- **Online-Prüfung mit Fernaufsicht (OnVUE):** Die Prüfung wird über das Pearson VUE OnVUE-System absolviert und per Webcam überwacht.
- **Prüfung in einem autorisierten Testzentrum:** Vor Ort in Pearson VUE-Testzentren weltweit (in vielen Städten wie Berlin, Hamburg, Frankfurt, München, Köln etc.).

Zusätzlich werden LPIC-Prüfungen oft zu vergünstigten Community-Konditionen auf Open-Source-Events (wie den Chemnitzer Linux-Tagen, FrOSCon oder Tübix) angeboten.

Um dich zur Prüfung anzumelden, benötigst du:

- Eine offizielle **LPI-ID**, die du kostenfrei im LPI-Portal registrierst.
- Ein gültiges **Ausweisdokument mit Lichtbild** für die Identitätsprüfung am Testtag.

Die Prüfungsgebühr für die LPIC-1 Exam 101 beträgt in der Regel ca. 200 USD (bzw. der regionale Euro-Äquivalentpreis). Die Prüfungen sind auf Deutsch und Englisch sowie weiteren Sprachen verfügbar.

> **Tipp:** Achte bei der Vorbereitung darauf, die aktuellen Lernziele für die Version 5.0 (Prüfungscodes 101-500 und 102-500) zu verwenden, da diese die verbindliche Grundlage der Prüfungsfragen bilden.

Befehlsreferenz (Cheatsheet)

whoami | Zeigt den aktuellen Benutzernamen der aktiven Shell-Sitzung anpwd | Gibt das aktuelle Arbeitsverzeichnis aus (print working directory)

`$ ls -la | Listet alle Dateien detailliert inklusive versteckter Dotfiles auf cd /pfad | Wechselt in das angegebene Verzeichnis (cd allein springt ins Home-Verzeichnis) echo $SHELL | Gibt den Pfad zur Standard-Login-Shell des Benutzers aus chsh -s /bin/zsh | Ändert die Standard-Login-Shell des aktuellen Benutzers dauerhaft tty | Gibt den Dateinamen des verbundenen Terminals aus (/dev/ttyX vs. /dev/pts/X) man befehl | Öffnet das ausführliche Manual des gewünschten Befehls which befehl | Zeigt den vollständigen Pfad der ausführbaren Programmdatei an`

Weiterführende Ressourcen

Fazit

Die Beherrschung der Kommandozeile ist nicht nur der wichtigste Prüfungsbaustein des Moduls 101, sondern das tägliche Handwerkszeug eines jeden Administrators. Du hast in diesem Einstiegsmodul verstanden, wie die Shell als Schnittstelle zum Kernel agiert, worin sich TTYs und grafische Terminal-Emulatoren unterscheiden und wie du dich souverän auf der

Kommandozeile orientierst.

Im nächsten Teil unserer LPIC-1-Serie widmen wir uns der praktischen Dateiverwaltung: LPIC-1: Grundlegende Navigation und Dateisystembefehle – dort lernst du das Anlegen, Verschieben, Finden und Verarbeiten von Dateien nach exakten LPI-Prüfungsstandards.

Zur Kursübersicht: Alle LPIC-1-Artikel & Module