Einführung Shell

Axel Pemmann

14. November 2022

Wichtigste Schnittstelle Mensch - Maschine

Die Power der Kommandozeile

- Verfügbarkeit vielzähliger spezieller Kommandos
- Automatische Abläufe mittels Skripte
- Geringer Ressourcenbedarf
- Konfiguration von Maschinen ohne grafische Oberfläche
- Fernadministration über langsame Leitungen

Wichtigste Schnittstelle Mensch - Maschine

Die Power der Kommandozeile

- Verfügbarkeit vielzähliger spezieller Kommandos
- Automatische Abläufe mittels Skripte
- Geringer Ressourcenbedarf
- Konfiguration von Maschinen ohne grafische Oberfläche
- Fernadministration über langsame Leitungen

Wichtigste Schnittstelle Mensch - Maschine

Die Power der Kommandozeile

- Verfügbarkeit vielzähliger spezieller Kommandos
- Automatische Abläufe mittels Skripte
- Geringer Ressourcenbedarf
- Konfiguration von Maschinen ohne grafische Oberfläche
- Fernadministration über langsame Leitungen

Wichtigste Schnittstelle Mensch - Maschine

Die Power der Kommandozeile

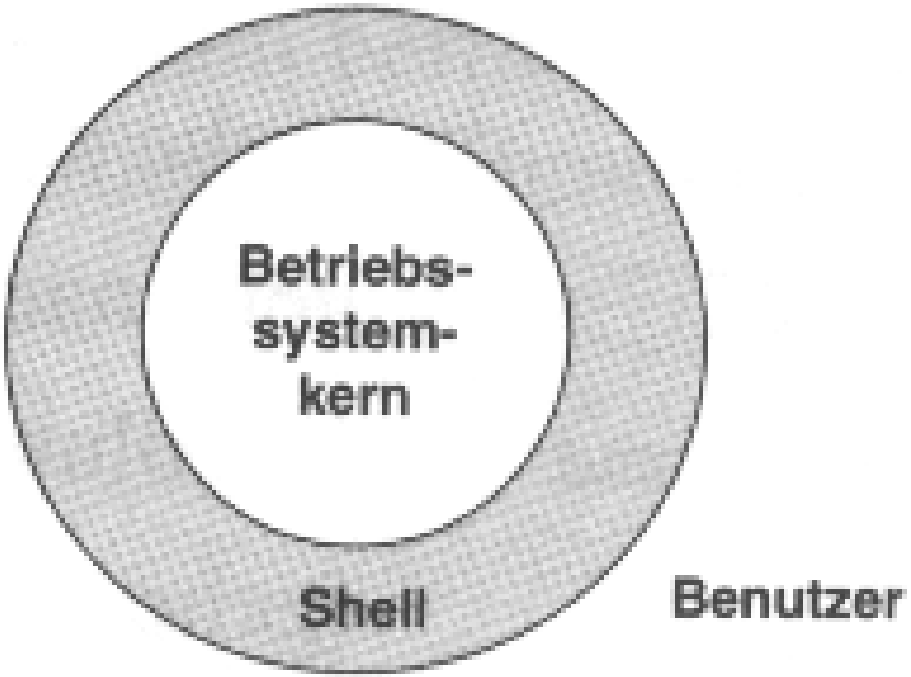
- Verfügbarkeit vielzähliger spezieller Kommandos
- Automatische Abläufe mittels Skripte
- Geringer Ressourcenbedarf
- Konfiguration von Maschinen ohne grafische Oberfläche
- Fernadministration über langsame Leitungen

Wichtigste Schnittstelle Mensch - Maschine

Die Power der Kommandozeile

- Verfügbarkeit vielzähliger spezieller Kommandos
- Automatische Abläufe mittels Skripte
- Geringer Ressourcenbedarf
- Konfiguration von Maschinen ohne grafische Oberfläche
- Fernadministration über langsame Leitungen

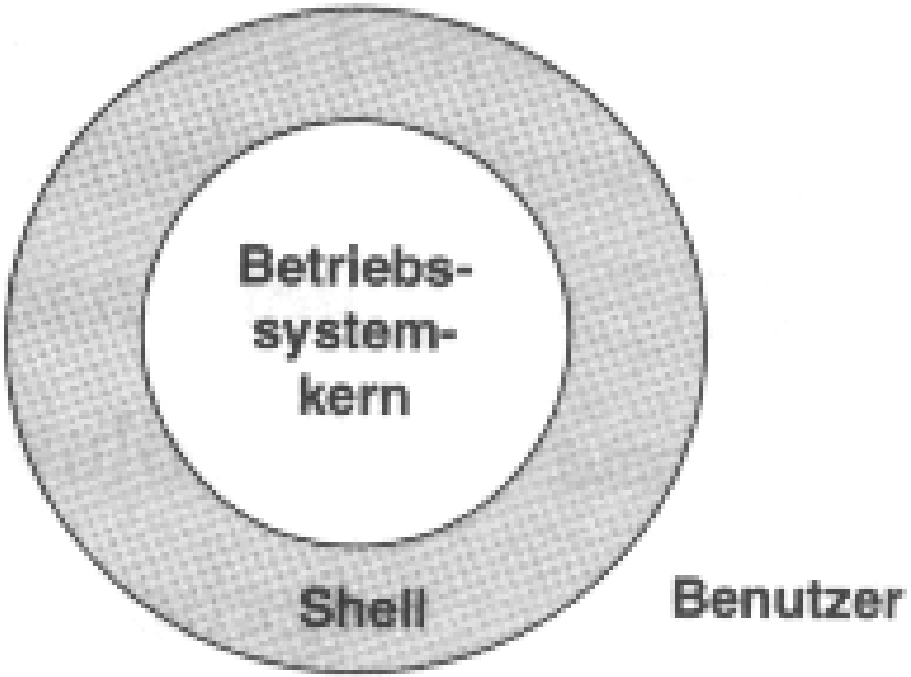
Viele Shells - eine Aufgabe.



Häufig verwendete Shells

- ash - Almquist Shell
- sh - Bourne Shell
- bash - Bourne Again Shell

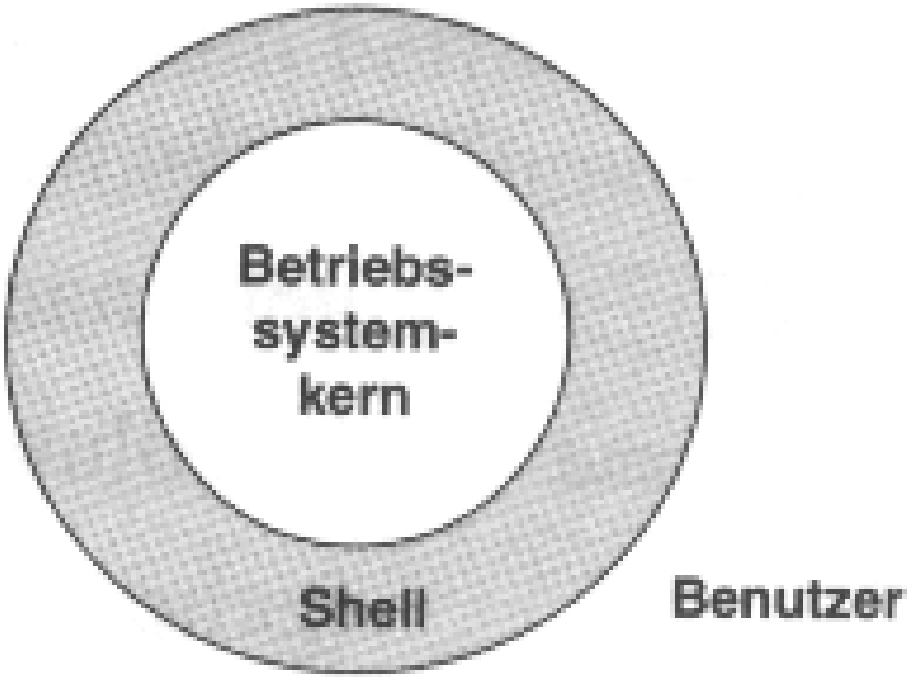
Viele Shells - eine Aufgabe.



Häufig verwendete Shells

- ash - Almquist Shell
- sh - Bourne Shell
- bash - Bourne Again Shell

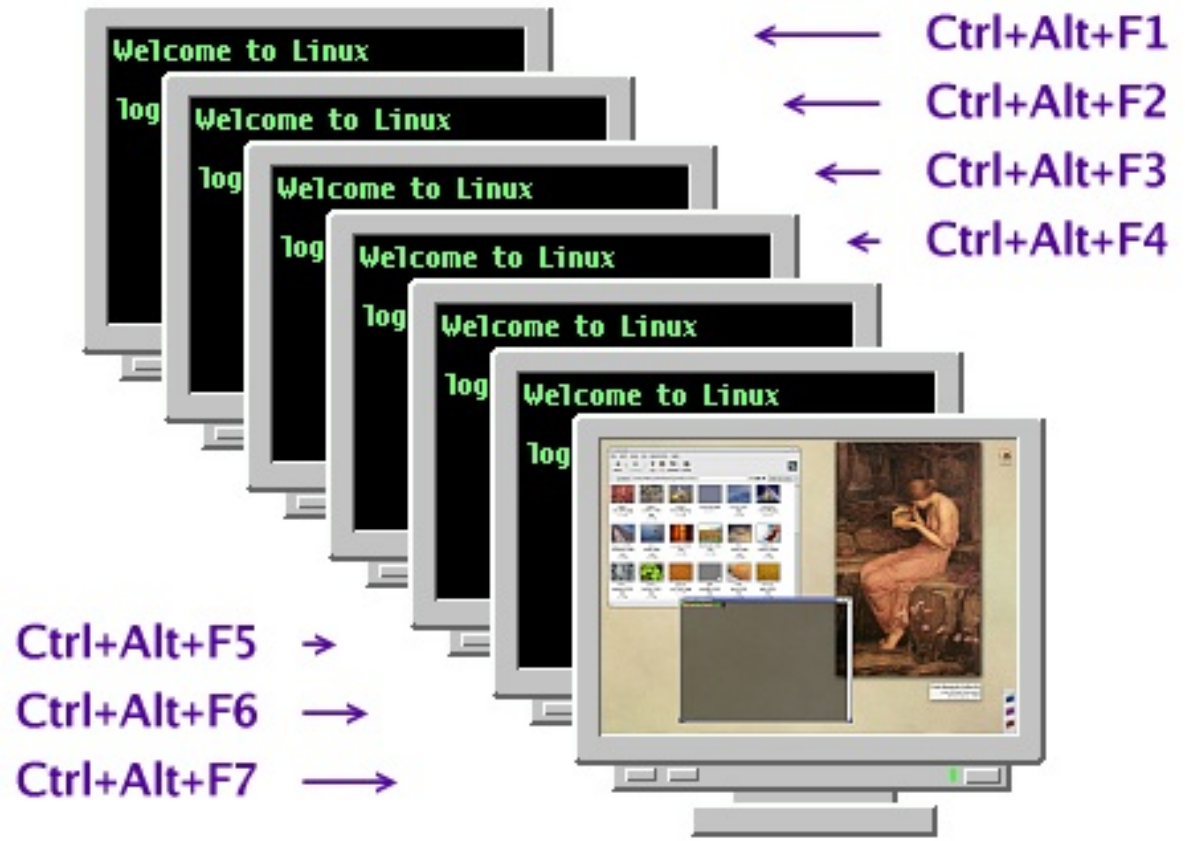
Viele Shells - eine Aufgabe.



Häufig verwendete Shells

- ash - Almquist Shell
- sh - Bourne Shell
- bash - Bourne Again Shell

Nur eine Shell zum Einloggen?



Wie ist eine Kommandozeile aufgebaut?

Ähnlich wie ein geschriebener Satz.

- IN BEZUG AUF DIE SHELL:

  Alle Bestandteile sind `Argumente`.
- IN BEZUG AUF EIN KOMMANDO:

  `Kommando Option Parameter Parameter`

  Zum Beispiel:

  ```
  cp -r /etc /srv/backup
  ```

Wo befinden sich die Kommandos?

Entweder in der Shell oder außerhalb.

- Ausgabe aller internen Kommandos:
  Kommando «enable»
- Prüfung, ob in- oder extern:
  Kommando «type»

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Kommandos

Die Shell interpretiert eine Zeile in einer bestimmten Reihenfolge, das eigentliche Kommando ist zuletzt dran:

1. Aliase (Kommando 'alias', Kürzel für andere Kommandos oder ganze Zeilen)
2. reserviertes Wort (z.B. if, case, for, while, do)
3. Shell-Funktionen (Kommando 'function')
4. interne, eingebaute Kommandos der Shell (z.B. cd, pwd, type, umask, source)
5. externe Kommandos (verstreut im Dateisystem liegende Anwendungen)

Abarbeitung von Expansionen

Die Shell führt weiterhin drei verschiedene Ersetzungen durch, bevor das eigentliche Kommando gesucht und ausgeführt wird:

1. Dateinamen (Joker, Wildcards): `*`, `?`, `[...]`, `[!...]`
2. Variablen, z.B.: `echo $PATH`
3. eingebettete Kommandozeilen:
   - Zwischen Backticks: `` `cmdline` ``
   - Mit Dollar und runden Klammern: `$(cmdline)`

Abarbeitung von Expansionen

Die Shell führt weiterhin drei verschiedene Ersetzungen durch, bevor das eigentliche Kommando gesucht und ausgeführt wird:

1. Dateinamen (Joker, Wildcards): `*`, `?`, `[...]`, `[!...]`
2. Variablen, z.B.: `echo $PATH`
3. eingebettete Kommandozeilen:
   - Zwischen Backticks: `` `cmdline` ``
   - Mit Dollar und runden Klammern: `$(cmdline)`

Abarbeitung von Expansionen

Die Shell führt weiterhin drei verschiedene Ersetzungen durch, bevor das eigentliche Kommando gesucht und ausgeführt wird:

1. Dateinamen (Joker, Wildcards): `*`, `?`, `[...]`, `[!...]`
2. Variablen, z.B.: `echo $PATH`
3. eingebettete Kommandozeilen:
   - Zwischen Backticks: `` `cmdline` ``
   - Mit Dollar und runden Klammern: `$(cmdline)`

Als Vordergrundprozess

Vorteile

- Tastatur-Input für Prozess möglich
- Output ist sichtbar (Debugging)
- Prozesse lassen sich leicht abbrechen/suspendieren

Nachteile

- Shell ist blockiert
- Zusätzlicher Prozessbedarf

Als Hintergrundprozess

Vorteile

- Shell ist nicht blockiert
- Eine App im Hintergrund ausführen: einfach das &-Zeichen anhängen
- Aufgaben trotz Ausloggen nicht blockieren (mit «nohup»)

Nachteile

- Kein Tastatur-Input für Prozesse möglich
- Prozessverwaltung und -kontrolle mit separaten Kommandos
- Vermischung der Terminalausgaben möglich