Einführung Betriebssysteme

Daniel KNITTL-FRANK
daniel.knittl-frank@fh-hagenberg.at
October 16, 2023

GNU+Linux

Wo?

- Smartphones: Android setzt auf einen Linux-Kernel auf
- Server: Webserver, Datenbankserver, …
  - Google, Facebook, …
- Science: Medical, Biology, Space
  - https://www.theverge.com/2021/2/19/22291324/linux-perseverance-mars-curiosity-ingenuity
- Animation: Render farms
- Autos: Tesla
- High-Performance Computing (HPC): Top 500 fastest computers
  - https://www.top500.org/statistics/details/osfam/1/
  - https://en.wikipedia.org/wiki/TOP500

Was?

- Freie, portable Mehrbenutzer-Betriebssysteme
- Basierend auf Linux-Kernel
- Open Source
  - Einzelpersonen
  - Non-Profit
  - Unternehmen

Lizenzen

- Proprietäre Software
  - Quelltext üblicherweise nicht zugänglich
  - Darf nicht verändert werden
- Freeware/Shareware
  - (Im Testzeitraum) gratis
  - Quelltext nicht zugänglich
- Free Software
  - Quelltext frei zugänglich
  - Weitergabe geregelt

Free Software

- Richard M. Stallman, “Free Software Movement”
- 1983: “4 Freedoms” (“4 Freiheiten”)
  - Run
  - Study
  - Redistribute
  - Improve
  - (verwenden, verstehen, verteilen, verbessern)

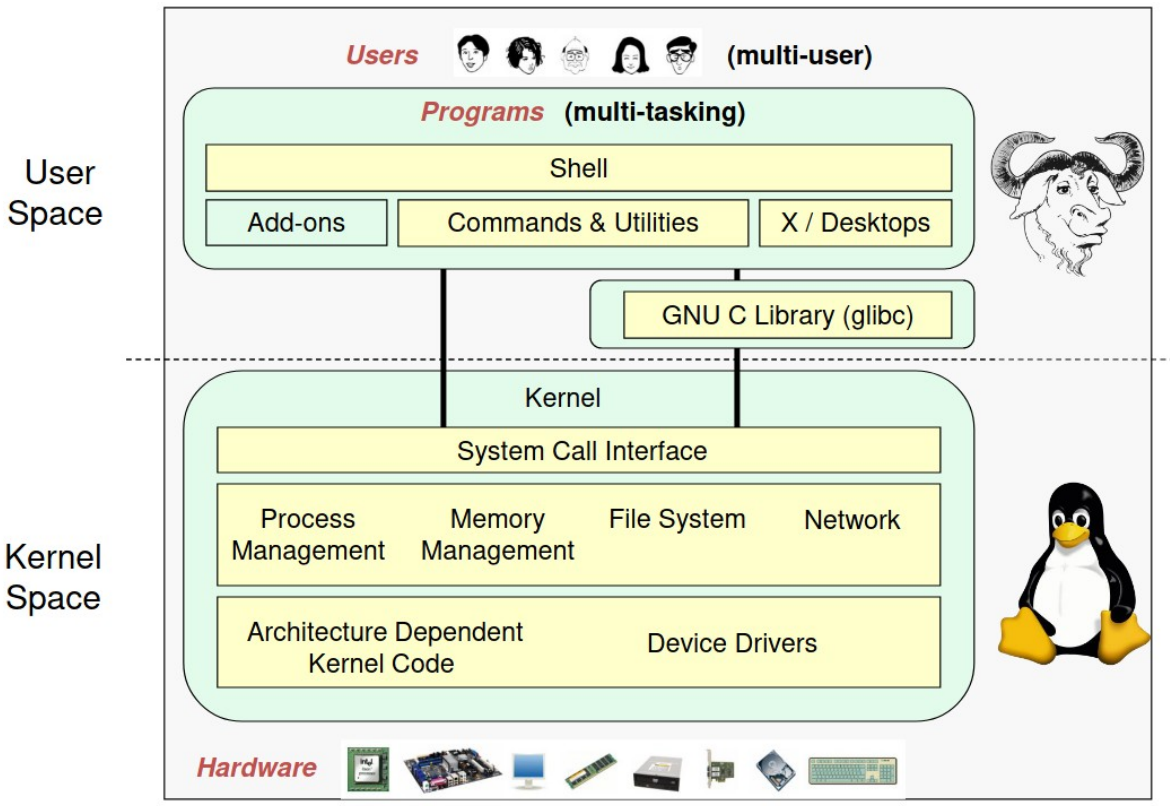
GNU, GPL

- ***G***NU's ***N***ot ***U***nix
- Ziel: Unix-ähnliches OS mit Sicherstellung der 4 Freiheiten
  - Kernel wurde nie fertiggestellt
- Copyleft
- GNU ***G***eneral ***P***ublic ***L***icense (1989)

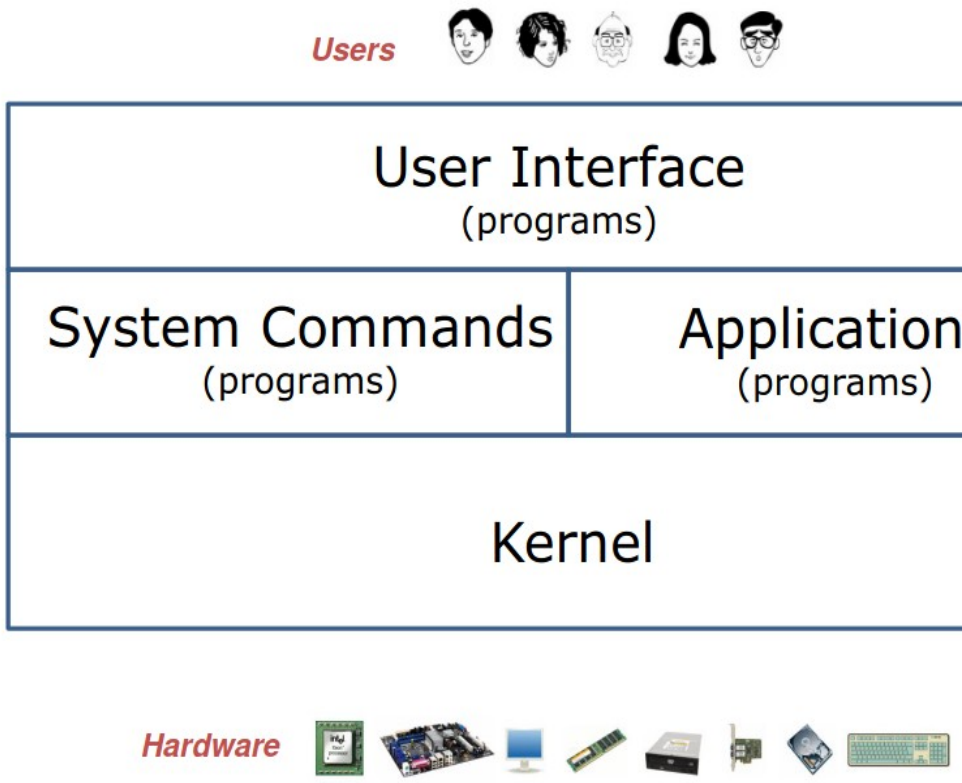
Linux

- 1991 begann Linus Torvalds mit der Entwicklung des OS-Kernels “Linux”
- Version 0.99 unter GPL-Lizenz veröffentlicht
- 1993 bereits mehr als 100 Entwickler weltweit
- Aktuelle Version: 6.5.7

GNU+Linux-Systemarchitektur



GNU+Linux-Systemarchitektur (vereinfacht)



CLI, GUI

- ***C***ommand ***L***ine ***I***nterface
  - Terminal
  - Shell
  - Textausgabe
  - Bedienung mittels (Text-)Kommandos
- ***G***raphical ***U****ser* ***I***nterface
  - Graphische Elemente
  - Bedienung mit Maus

Shells

- Kommunikation mittels “command line”
- Shell analysiert Kommando-Eingaben und führt Programme aus
- Viele unterschiedliche Shell-Implementierungen
  - Sh (Bourne shell)
  - Bash (Bourne again shell)
  - Csh (C shell)
  - Ksh (Korn shell)
  - Zsh (Z shell)
- Interaktive Nutzung sowie Skriptausführung möglich

System Commands

- Viele Commands und Utilities
- ls, cat, rm, …
- Utilities: vi, sort, find, tar, …
- Administration: useradd, groupadd, passwd, …

Anwendungen

- Zusätzlich installierbar
- Beispiele:
  - Apache2 oder nginx Web-Server
  - GIMP Bildbearbeitung
  - Blender 3D-Editor und -Animation
  - LibreOffice-Suite
  - Firefox-Browser
  - VLC-Videoplayer

Kernel

- Hardwareabstraktionsschicht
- Aufgaben:
  - Prozessverwaltung
  - Speicherverwaltung
  - Dateisystemzugriff
  - Eingabe und Ausgabe (Dateien, Netzwerk, Grafik, USB, …)

Eigenschaften eines GNU+Linux OS

- Multitasking
- Multibenutzer
- Austauschbare UIs (GUIs, CLIs)
- Kernelmodule zum dynamischen Laden von Treibern
- Berechtigungssystem
- Läuft auf vielen verschiedenen Plattformen
- Robust
- Mächtig (nach initialer Einarbeitung)

Distributionen

- (GNU+)Linux-Distributionen
- Zusammenstellung eines Kernels mit Anwendungsprogrammen
- Bietet ein vollständiges, arbeitsfähiges OS
- Distributionen unterscheiden sich in:
  - Installierte Standardprogramme
  - Verfügbare Programme
  - (Bezahlter) Support
  - Konfiguration

Bekannte Distributionen

- Arch Linux
- CentOS
- Debian
- Fedora
- Gentoo
- Linux Mint
- Mandriva
- Manjara
- OpenSUSE
- RHEL (Red Hat Enterprise Linux)
- Slackware
- Ubuntu

First Steps

Daniel KNITTL-FRANK
daniel.knittl-frank@fh-hagenberg.at
October 16, 2023

Dateisystem

- Hierarchisch aufgebaut
- Verzeichnisse, Unterverzeichnisse und Dateien durch “/” getrennt
- 1 Wurzelverzeichnis “/” (“root”)
- Jeder Benutzer hat ein Benutzerverzeichnis (“Home”)
- Ein aktives Verzeichnis (“Working Directory”)
- Partitionen und Laufwerke nicht direkt erkennbar
- Groß- und Kleinschreibung wichtig! “Datei.txt” und “datei.TXT” sind zwei unterschiedliche Dateien

Filesystem Hierarchy Standard (FHS)

- Definiert Verzeichnisstruktur
- Richtlinie, Distributionen können abweichen

Filesystem Hierarchy Standard (FHS)

- **/** root directory
  - **/bin** binary files forming the commands and shells used by the system administrator and users
  - **/boot** files used during the initial boot-up process including the kernel
  - **/dev** device files for connected hardware
  - **/etc** system configuration files
  - **/home** individual directories owned by each user
  - **/lib** shared libraries needed to boot the system and run the commands in the root filesystem (i.e. commands in /bin and /sbin)
  - **/lost+found** recovered files that were corrupted by power failures or system crashes
  - **/mnt** mount points for other file systems (CDs, USB drives, …)
  - **/opt** add-on software packages and/or commercial applications
  - **/proc** kernel level process information
  - **/root** home directory for the root user
  - **/sbin** system administration commands reserved for the superuser (root)
  - **/tmp** temporary files that are deleted when the system is rebooted or started
  - **/usr** program files and related files for use by all users
  - **/var** log files, print spool files, and mail queues

Prompt

- user@machine:/path/to/working/dir$
  - **user**: aktueller Benutzername
  - **machine**: aktueller Computername
  - **/path/to/working/dir**: aktuelles Verzeichnis
  - **$**: normaler Benutzeraccount
    - #: root-user (Administrator)

Kommandoeingabe

- Kommando und Parameter durch Leerzeichen getrennt
- Beliebig viele Parameter
- Reihenfolge teilweise irrelevant
- Dateipfade üblicherweise am Ende angegeben

Kommandoeingabe: Beispielaufrufe

- ps
- ps -e
- ps -F
- ps -e -F
- ps -eF
- ls
- ls -l
- ls /home/daniel
- ls -l /home/daniel
- echo 'Guten Tag'

Einfache Kommandos

- **man** Zeigt Handbücher zu Kommando an (“manual page”)
- **help** Zeigt Hilfe zu manchen Kommandos an
- **cal** Zeigt Kalender an
- **clear** Leert die aktuelle Anzeige
- **date** Gibt das Datum aus
- **exit** Beendet die aktuelle Shell-Session
- **history** Zeigt ausgeführte Kommandos an
- **hostname** Gibt den Computernamen aus
- **id** Gibt User- und Gruppen-Ids aus
- **ps** Gibt Prozessinformationen aus
- **uname** Gibt OS-Informationen aus
- **tty** Gibt das Terminal-Device der Session aus
- **who** Gibt alle eingeloggten Benutzer aus
- **whoami** Gibt Namen des eingeloggten Benutzers aus

Weitere Kommandos

- **cat filename** Gibt Dateiinhalt aus
- **cd directory** Wechselt das aktuelle Verzeichnis
- **echo string** Gibt Text aus
- **less filename** Zeigt Dateiinhalt seitenweise an
- **ls path** Listet Dateien eines Verzeichnisses
- **which command** Zeigt den Speicherort eines Kommandos an
- **bc** Taschenrechner (binary calculator)

Hilfe!

- “man kommando” öffnet Handbuch für Kommando
  - Scrollen mit Cursortasten oder PgUp/PgDn
- “apropos wort” oder “man -k wort” durchsucht Handbücher
- “help builtin” zeigt Hilfe für “builtin” an

Navigation

- cd “change directory”
- 2 Arten von Pfaden/Pfadangaben
  - Absolut
    - Immer vom Root-Verzeichnis (“/”) ausgehend
    - Aktuelles Arbeitsverzeichnis egal
    - Beginnen immer mit “/” (oder “~”)
  - Relativ
    - Abhängig vom aktuellen Arbeitsverzeichnis
- Wichtige Pfade:
  - “.” Aktuelles Arbeitsverzeichnis
  - “..” Elternverzeichnis (“eines drüber”)
  - “~” Home-Verzeichnis des Benutzers

Navigation – Beispiele

- Beispiel: Aktuelles Verzeichnis: /home/bob
  - “pwd” Gibt aktuelles Verzeichnis aus
- Wechseln in Verzeichnis “/home/bob/Dokumente”: “cd Dokumente”
- Wechseln in das übergeordnete Verzeichnis: “cd ..”
- Wechseln in Benutzerverzeichnis des Benutzers “alice”:
  - Absolut: cd /home/alice
  - Relativ:
    - cd ../alice
    - cd ./../alice
    - cd .././alice
    - cd ../alice/.
- Wechseln in eigenes Homeverzeichnis: “cd” oder “cd ~”
- Wechseln in das Root-Verzeichnis: “cd /”
- Wechseln in das Home-Verzeichnis des Benutzers “root”: “cd /root”; oder “cd /” gefolgt von “cd root”

Verzeichnisinhalt anzeigen

- **ls** Zeigt Inhalt an (einfach)
- **ls -l** Zeigt Inhalt mit Zusatzinfos (Änderungsdatum, Berechtigungen, Größe) an
- **ls -a** Zeigt alle (=auch versteckte) Dateien an
- **ls /path/to/dir** Zeigt Inhalt von “/path/to/dir” an (unabhängig von aktuellem Verzeichnis)
- **ls -l datei1 datei2** Zeigt Zusatzinfos zu 2 Dateien an
- **ls -al, ls -la, ls -all** Zeigt Zusatzinfos zu allen Dateien an
- **ls --all** Zeigt alle Datein an (einfach)

Erweiterte Dateiinfos

drwxr-xr-x 2 daniel eib 4096 Mar 14 13:37 verzeichnis
-rw-r—r-- 1 daniel eib 42 Mar 14 13:37 datei

- Dateityp und Berechtigungen
- Anzahl an Links
- Besitzer (owner)
- Gruppe (group)
- Größe in Bytes
- Änderungsdatum
- Name

Shell Wildcards/Globs

- Können als Platzhalter in Dateinamen verwendet werden:
  - “*” 0 oder mehrere Zeichen (Ausnahmen: “.” am Anfang, “/”)
  - “?” ein beliebiges Zeichen (Ausnahmen: “.” am Anfang, “/”)
  - “[abc]” a, b oder c
  - “[a-f]” a, b, c, d, e oder f
- Werden von der Shell vor der Ausführung des Kommandos ersetzt
- “\” (Backslash) oder Anführungszeichen “”” und Apostrophe “’” unterbinden Ersetzung durch Shell
- Beispiele:
  - Namen aller Textdateien im aktuellen Verzeichnis ausgeben: echo *.txt
  - Details zu allen Textdateien im aktuellen Verzeichnis anzeigen: ls -l *.txt
  - Namen der Home-Verzeichnisse ausgeben: echo /home/*
  - Dateinamen mit genau 4 Zeichen: “????”

Arbeiten mit Dateien und Verzeichnissen

- **touch**: Datei anlegen
- **mkdir**: Leeres Verzeichnis anlegen
- **rm**: Datei löschen (Achtung: keine Rückfrage!)
- **rmdir**: Leeres Verzeichnis löschen
- **cp**: Datei(en) kopieren
  - **cp a b** Kopiert Datei a nach b
  - **cp a b c directory** Kopiert a, b und c ins Verzeichnis “directory”
  - **cp -r dir1 dir2** Kopiert Verzeichnis “dir1” rekursiv nach “dir2”
- **mv**: Datei(en) verschieben oder umbenennen
  - **mv a b** Verschiebt Datei a nach b
  - **mv a b c directory** Verschiebt Dateien a, b und c ins Verzeichnis “directory”
  - **mv dir1 dir2** Verschiebt dir1 ins Verzeichnis dir2 ODER Benennt dir1 auf dir2 um
- **ACHTUNG: keine Rückfrage beim Löschen oder Überschreiben von Dateien!**

Textbearbeitung

Daniel KNITTL-FRANK
daniel.knittl-frank@fh-hagenberg.at
October 16, 2023

Textbearbeitung

- Einfache Kommandos zur Textbearbeitung und -filterung
- Jedes Kommando hat *eine* Aufgabe

Kommandos (Auszug)

- “wc” Word count
- “head”, “tail” Anfang und Ende von Text anzeigen
- “cut” Spalten extrahieren
- “sort” Text sortieren
- “uniq” Duplikate filtern
- (grep) Text suchen
- (sed) Stream editor

wc

- “word count”
- “wc” Zählt Zeilen, Wörter und Zeichen
- “wc -l” Zählt Zeilen
- “wc -w” Zählt Wörter
- “wc -c” Zählt Bytes (Characters)

wc (Beispiel)

$ cat datei

Dolor accusantium temporibus lorem repellendus quasi veritatis? Molestias consequuntur a quos perspiciatis doloribus? Officiis a natus dicta dolores fugiat, est earum. Ad similique necessitatibus nam placeat ab quo quae consequuntur

$ wc datei

4 30 233 datei

$ wc -l datei

4 datei

$ wc -w datei

30 datei

$ wc -c datei

233 datei

head / tail

- Zeigt Anfang bzw. Ende eines Textes an
- Standardmäßig 10 Zeilen

head / tail (Beispiel)

$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin

$ head -n1 /etc/passwd

root:x:0:0:root:/root:/bin/bash

$ tail -n1 /etc/passwd

sys:x:3:3:sys:/dev:/usr/sbin/nologin

cut

- Extrahiert Spalten aus Text
- Standardtrennzeichen ist Tabulator (“\t”)

cut (Beispiele)

$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin

$ cut -d: -f1 /etc/passwd

root
daemon
bin
sys

$ cut -d: -f1,3-4,6 /etc/passwd

root:0:0:/root
daemon:1:1:/usr/sbin
bin:2:2:/bin
sys:3:3:/dev

$ cut -c1,3,5 /etc/passwd

ro:
deo
bnx
ssx

sort

- Sortiert Text zeilenweise
- Viele Optionen (man sort)
  - -r Absteigend sortieren
  - -n Numerisch sortieren
  - -t Trennzeichen festlegen
  - -k Sortierschlüssel (key) festlegen
- Hinweis: Sortierung hängt teilweise von aktueller Systemsprache (“locale”) ab

sort (Beispiele)

$ cat datei

```
05
A
ab
a
12
2
1
10
z
```

$ sort datei

```
05
1
10
12
2
A
a
ab
z
```

$ sort -r datei

```
z
ab
a
A
2
12
10
1
05
```

$ sort -n datei

```
A
a
ab
z
1
2
05
10
12
```

uniq

- Filtert doppelte Zeilen aus Text
- Nur aufeinanderfolgende Zeilen werden gefiltert
- “uniq -c” zählt wie oft eine Zeile wiederholt ist

uniq (Beispiele)

$ cat datei

a
b
b
a

$ uniq datei

a
b
a

$ uniq -c datei

1 a
2 b
1 a

Eingabe-/Ausgabe-Umleitung

Daniel KNITTL-FRANK
daniel.knittl-frank@fh-hagenberg.at
10/25/22

Textbearbeitung

- Linux-Kommandos arbeiten meist textbasiert
- Text kann …
  - Aus Dateien gelesen werden
  - In Dateien geschrieben werden
  - An andere Kommandos übergeben werden

Standard-Streams

- Stdin (0) Standardeingabe
- Stdout (1) Standardausgabe
- Stderr (2) Standardfehlerausgabe
- In Programmiersprachen verfügbar:
  - C: FILE *stdin, *stdout, *stderr in <stdio.h>
  - C++: std::cin, std::cout, std::cerr in <iostream>
  - Java: System.in, System.out, System.err
  - Python: sys.stdin, sys.stdout, sys.stderr

Standardeingabe

- Selten notwendig, die meisten Kommandos können Dateien mittels Pfad öffnen
- Umleitung kann nur eine Datei auf einmal öffnen
- Ausgabe kann sich minimal unterscheiden:

```
$ wc < datei.txt
  4  30 233
$ wc datei.txt
  4  30 233 datei.txt
```

Standardausgabe

- Kommandoausgabe kann in Datei umgeleitet werden
- “>” überschreibt Dateiinhalt
  - ACHTUNG: Keine Rückfrage!
- “>>” hängt ans Ende einer Datei an

```
$ echo hallo > nachricht.txt
$ ls > verzeichnislisting.txt
$ date > datum.txt
$ date >> datum.txt
```

Standardfehlerausgabe

- Fehlerausgabe kann in Datei umgeleitet werden
- “2>” überschreibt Dateiinhalt
  - ACHTUNG: Keine Rückfrage!
- “2>>” hängt ans Ende einer Datei an

```
$ cat * > inhalte.txt 2> fehler.txt
$ grep ‘^’ > inhalte.txt 2>> fehler.tat
```

Besondere Dateien

- **/dev/null** ”Blackhole”, verwirft jegliche Eingabe (kann zum Ignorieren von Ausgabe verwendet werden)
- **/dev/zero** Kann genutzt werden, um eine unendliche Anzahl an Null-Bytes (0x00) zu lesen/erzeugen
- **/dev/random** Liefert zufällige Bytes (auf alten Kernels langsam!)
- **/dev/urandom** Liefert zufällige Bytes

Häufige Fehler

- Datei gleichzeitig Eingabe und Ausgabe => leere Datei
  - Falsch: sort <datei >datei
  - Falsch: sort datei >datei
  - Richtig: sort <datei >datei.sortiert; mv datei.sortiert datei
- Standardausgabe und Fehlerausgabe in gleiche Datei umleiten => korrupter Text
  - Falsch: cat * >ausgabe 2>ausgabe
  - Richtig: cat * >ausgabe 2>&1

Pipes

- Ausgabe und Eingabe von Kommandos können miteinander verknüpft werden
- Mittels Pipe “|”
  - FIFO-Buffer (First-In, First-Out)
- Baukastenprinzip: Kombinieren von einfachen Kommandos, um komplexe Aufgaben zu lösen

Pipes (Beispiel)

$ sort datei.txt | uniq -c | sort -n # Häufigkeit bestimmen
$ fold -b1 datei.txt | sort | uniq -c | sort -rn | head -1 # Häufigstes Zeichen/Byte
$ tail -2 datei.txt | head -1 # Vorletzte Zeile
$ cat * | wc -l # Gesamtanzahl an Zeilen bestimmen

Berechtigungen

Daniel KNITTL-FRANK
daniel.knittl-frank@fh-hagenberg.at
Jan 13, 2024

Berechtigungen

- Benutzer und Benutzergruppen
- Für Dateien und Verzeichnisse

Standardrechte im Dateisystem (1)

- Jede Datei ist einem Eigentümer/Besitzer (user, u) und einer Gruppe (group, g) zugewiesen
- Fällt der eigene Usere nicht in die zwei genannten Kategorien, gelten die Berechtigungen für “alle anderen” (others, o)
- Für jede Kategorie können drei Rechte festgelegt werden:
  - Lesen (read, r)
  - Schreiben (write, w)
  - Ausführen (execute, x)
- R, w und x können nur vom Besitzer oder von “root” geändert werden
- Berechtigungen können mit “ls -l” oder “stat” angezeigt werden

Standardrechte im Dateisystem (2)

```
drwxr-xr-x 2 daniel    nbl   4096 Nov  6 13:37 ordner/
-rw-rw-r-- 1 christian se_bb   42 May 19  2019 datei
-rwxr----- 1 root      root   420 Oct 31 12:48 setup.sh
^^^^^^^^^^   ^^^^^^^^^ ^^^^^ ^^^^ ^^^^^^^^^^^^ ^^^^^^^^
permissions  user      group size modification name
^|||||||||| type (d=directory, -=regular file)
 ^^^|||||| user
    ^^^||| group
       ^^^ others
```

Dateirechte

- **r** Dateiinhalt kann gelesen werden
- **w** Dateiinhalt kann geschrieben werden
- **x** Datei kann ausgeführt werden (Programm oder Skript)

Verzeichnisrechte

- **r** Verzeichnisinhalt kann angezeigt werden (ls)
- **w** Verzeichniseinträge können erstellt/geändert werden
  (Unterordner/Dateien erstellen, kopieren, verschieben, löschen)
- **x** In Verzeichnis kann gewechselt werden (cd)

Rechte (numerisch) (1)

- 4 **r**ead (binär 100)
- 2 **w**rite (binär 010)
- 1 e**x**ecute (binär 001)

Warum?

→ Kodierung als “Oktalzahl” (oktal = achtwertig; acht Ziffern: 0…7).
Oft “0” vorangestellt, um von Dezimalzahlen zu unterscheiden

Rechte (numerisch) (2)

- Summe ergibt Gesamtberechtigung:
- 7 => rwx (4+2+1)
- 6 => rw- (4+2+0)
- 5 => r-x (4+0+1)
- 4 => r-- (4+0+0)
- 3 => -wx (0+2+1)
- 2 => -w- (0+2+0)
- 1 => --x (0+0+1)
- 0 => --- (0+0+0)

Rechte (numerisch) (3)

- Pro Kategorie (user, group, others) eine Ziffer
- Beispiele:
  - 0764 => rwxrw-r--
  - 0640 => rw-r-----
  - 0777 => rwxrwxrwx (sollte vermieden werden!)

SUID und SGID

- “Set User ID” und “Set Group ID”
- Werden durch “s” statt “x” repräsentiert (z.B. bei “ls -l”)
- Ausführbare Dateien werden so mit den Berechtigungen des Dateibesitzers bzw. der Dateigruppe ausgeführt und nicht mit den Berechtigungen des aktuellen Benutzers
- /bin/passwd erlaubt so das Ändern des eigenen Passworts – denn nur der root User kann /etc/shadow schreiben
- SGID auf Verzeichnissen gibt neu erstellen Dateien automatisch die Gruppe des Verzeichnisses
- Oktal als zusätzliche Ziffer:
  - 4 SUID
  - 2 SGID
  - z.B.: 0**6**754 = rw**s**r-**s**-r--

Sticky Bit

- Auch bekannt als “Restricted deletion flag”
- Heutzutage meist nur auf Verzeichnisse gesetzt
- Nur “root” oder Eigentümer können Dateien löschen oder umbenennen
- Darstellung als “t” in Ausgabe von ls
- Beispiel: /tmp
- Oktal: 1
  - z.B.: 01777 = rwxrwxrwt

Zugriffsrechte ändern (1)

- **chmod** – change file mode bits
- Nur Eigentümer und root können Zugriffsrechte ändern
- “-R” setzt rekursiv (=inklusive aller Unterordner)
- Rechte können oktal oder symbolisch angegeben werden

Zugriffsrechte ändern (2)

- “chmod MODE PATHS…”, mit MODE =
  - Ziel (**u**…ser, **g**…roup, **o**…thers, **a**…ll)
  - Operation
    - **-** Berechtigung von aktuellem Wert entfernen
    - **+** Berechtigung zu aktuellem Wert hinzufügen
    - **=** Berechtigungen auf exakten Wert setzen
  - Wert (**r**…ead, **w**…rite, e**x**…ecute, **s**…UID/GID, (s)**t**…icky bit)
- Oder Berechtigung als Oktalzahl angeben

Zugriffsrechte ändern (3)

- Beispiele

```
# user bekommt execute:
chmod u+x paths
# gruppe write entziehen, others alles verbieten:
chmod g-w,o= paths
# allen (user, group, others) nur read+write erlauben,
# anschließend gruppe und others dann write entziehen:
chmod a=rw,go-w paths
# user=read&write, group=read, others=-:
chmod 0640 paths
# user=rwx, group=rx, others=r:
chmod 0754 paths
# setze SUID bit, user=rwx, group=r, others=r:
chmod 04744 paths
```

umask (1)

- “umask” definiert welche Rechte bei Erstellung von Dateien/Verzeichnissen *entzogen* werden
- Verzeichnisse starten bei 0777
- Dateien starten bei 0666

umask (2)

- Beispiele Verzeichnisse:
  - umask 0022: 0777 & ~0022 = 0755
  - umask 0137: 0777 & ~0137 = 0640
- Erklärung:
  - “&” is bitwise AND
    - 1 & 1 = 1
    - 1 & 0 = 0
    - 0 & 0 = 0
  - “~” is bitwise NOT
    - ~1 = 0
    - ~0 = 1

umask (2)

- Beispiele Dateien:
  - umask 0022: 0666 & ~0022 = 0644
  - umask 0137: 0666 & ~0137 = 0640
- Erklärung:
  - “&” is bitwise AND
    - 1 & 1 = 1
    - 1 & 0 = 0
    - 0 & 0 = 0
  - “~” is bitwise NOT
    - ~1 = 0
    - ~0 = 1

Eigentümer/Gruppe ändern

- chown owner:group PATH…
- Kann nur durch root geändert werden!
  - Andernfalls könnte man Zugriff auf eigene Dateien verlieren
  - “Verschenken” nicht möglich
- -R ändert Besitzer/Gruppe rekursiv (= inklusive aller Unterverzeichnisse)
- Varianten:
  - chown owner setzt neuen Besitzer
  - chown :group setzt neue Gruppe
  - chown owner:group setzt neuen Besitzer und neue Gruppe

Root User

- Direkter Login mit User “root” auf den meisten Systemen deaktiviert
  - Account hat kein Passwort
- Abhilfe: “sudo” (“superuser do”, “substitute user, do”)
- Mit “sudo” kann ein Programm als root ausgeführt werden, z.B.:
  - “sudo ls /root”

Konfiguration von sudo

- Per Konfiguration kann “sudo” für Benutzer freigeschaltet werden
- Konfiguration in Textdatei /etc/sudoers
  - Sollte nur mit Befehl “visudo” bearbeitet werden
    - Syntaxüberprüfung
    - Gleichzeitiges Bearbeiten unterbunden
  - Tippfehler können zum Aussperren aus dem System führen
  - Es ist möglich, sudo auf bestimmte Programmaufrufe einzuschränken

Sudo Passwort

- Normalerweise verlangt “sudo” das Passwort des aktuellen Benutzers
  - Je nach Konfiguration
  - Passwortabfrage kann auch deaktiviert werden (in sudoers-Datei)
- Sudo merkt sich das Passwort für eine gewisse Zeit (üblicherweise 5-15 Minuten)
- Mit “sudo -v” kann die Frist verlängert werden

Root vs sudo

- “root” …
  - Darf alles
  - Ist nicht für die tägliche Verwendung gedacht
- Standardinstallationen vieler Distributionen …
  - Deaktivieren den “root”-Account
  - Erlauben keine direkte Anmeldung mit “root”

Benutzerkonfiguration

- Verwaltung von lokalen Benutzerkonten in Textdateien
  - /etc/passwd Benutzerinfo
  - /etc/shadow Passwortinfo
  - /etc/group Gruppenzuordnung der Benutzer
- Modifikationen mittels CLI-Programmen

/etc/passwd

$ man 5 passwd

```
name:password:UID:GID:comment:homedir:shell
```

- Passwort ist immer “x” (tatsächliches Passwort in /etc/shadow)
- UID ist die eindeutige Benutzernummer
- GID ist die primäre Gruppe des Benutzers
- Systembenutzer (für Hintergrunddienste) und “menschliche” Benutzer
- Systembenutzer haben keine normale Shell definiert
- Systembenutzer: UID zwischen 100 und 999
- Menschliche Benutzer: UID größer gleich 1000

Benutzer anlegen

```
$ sudo adduser benutzername
```

- Interaktives Programm
- Fragt alle notwendigen Benutzerinfos ab
- Mit --system kann ein Systemuser angelegt werden

/etc/shadow

```
$ man 5 shadow
```

```
name:$alg$salt$digest:lastchange:minpwdage:maxpwdage:pwdwarn:pwdinactivity:accountexpiration:reserved
```

- Benutzername
- Verschlüsseltes Passwort (Deaktivierte Accounts: “!” oder “*” → kein direkter Login möglich, z.B. “root”)
- Datum der letzten Passwortänderung (in Tagen seit dem 1.1.1970)
- Mindestanzahl an Tagen zwischen Passwortänderungen
- Tage bis das Passwort ausläuft
- Wie viel Tage wird vor dem Auslaufen gewarnt
- Wie viel Tage nach Auslaufen des Passworts das Konto gesperrt
- Datum an dem der Account gesperrt wird (in Tagen seit dem 1.1.1970)
- Reserviert für zukünftige Verwendung

Leere Felder = kein Ablauf-/Sperrdatum

/etc/group

```
$ man 5 group
```

```
group:password:GID:users
```

- 1. Gruppenname
- 2. “Passwort”, meistens “x” – siehe “man 5 gshadow”
- 3. Eindeutige Gruppennummer
- 4. Benutzernamen durch Komma getrennt

Passwörter

- Passwörter werden gehasht gespeichert
- Auch für “root” nicht einsehbar (“One-way hash”)
- Beim Login wird das eingegebene Passwort erneut gehasht und mit dem gespeicherten Hash verglichen
- So gespeicherte Passwörter können nur durch “Brute force” (=Durchprobieren) aller möglichen Passwörter geknackt werden
- “Salt” (zu deutsch: Salz) verhindert das gleichzeitige Knacken von unterschiedlichen Accounts mittels Rainbow-Tables
- Oft mehrere “Runden”, um zu verlangsamen

Brute Force

- https://en.wikipedia.org/wiki/Password_strength
- https://www.keepsolid.com/passwarden/help/use-cases/how-long-to-crack-a-password
- “Wörterbuch-Attacken” (dictionary attacks) testen gängige Passwörter und Varianten
  - “Password123” dauert theoretisch lange, in der Praxis ist es sofort geknackt
- Viele Tools, z.B. “John the Ripper”, “Hashcat” https://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-user-passwords-linux-system-0147164/
- Beispiele bei 62 Zeichen (groß, klein, Ziffern) und 100 Millionen Passwörter pro Sekunde:
  - 4 Zeichen ~15Mio 0.1 s
  - 5 Zeichen ~900Mio 9,1 s
  - 6 Zeichen ~57Mrd 9,5 m
  - 7 Zeichen ~3.5Bio 9,8 h
  - 8 Zeichen ~218Bio 25,3 d
  - 9 Zeichen ~13.5Brd 4,3 y
  - 10 Zeichen~840Brd 266 y
  - 11 Zeichen ~52Trio 16500 y
  - 12 Zeichen ~3.5Trd 1023042 y

Regular Expressions

Daniel KNITTL-FRANK
p23687@fh-hagenberg.at
Jan 12, 2024

A Bit of Background

- Originated in 1951
- “Regular Events” by Stephen C. Kleene
- Equivalent to “finite automata” (Kleene’s theorem)
- Can be used to describe “regular languages”
- Type-3 grammar in Chomsky hierarchy

General

- Regular Expressions, RegExp, RegEx, RE
- Literals often marked with slashes: /regex/
- Slightly different syntax and features (BRE, ERE, PCRE)

BRE

Basic Regular Expressions

Simple Regex (1)

- A simple regex is /R/, which matches a single upper case R
  - RegEx
  - rrRRrr

Simple Regex (2)

- The regex /RE/ matches an R *immediately* followed by an E
  - RegEx [no match]
  - REgex
- /RE/ is composed of two regex /R/ and /E/

Matching Any Character: .

- The regex /./ matches *any* single character (except line breaks)
  - RegEx.
- /e./
  - regex

Escaping Meta Characters: \

- A backslash escapes the next character
  - Meta Characters become literals (\.)
  - Some literals become meta characters (\t)
- To match a dot, use /\./:
  - RegEx.

Repeating Regex: *

- Repeats the regex *immediately* preceding the quantifier
- 0..∞ repetitions
- *Greedy*, matches as much as possible
- /Ha**l***o/
  - Hao
  - Halo
  - Hallo
  - Halllo
- /a*****b/
  - aa*bb

Character Classes: []

- Match a single character from a set
- Sets can contain ranges
- /gr**[ae]**y/
  - gr**a**yhound
  - gr**e**yhound
- /0x**[0-9a-f_]***/
  - 0x**1e_e7**

POSIX Character Classes: [[::]]

[:upper:]	uppercase letters
[:lower:]	lowercase letters
[:alpha:]	upper- and lowercase letters
[:digit:]	digits
[:xdigit:]	hexadecimal digits
[:alnum:]	digits, upper- and lowercase letters
[:punct:]	punctuation (all graphic characters except letters and digits)
[:blank:]	space and TAB characters only
[:space:]	blank (whitespace) characters
[:cntrl:]	control characters
[:graph:]	graphic characters (all characters which have graphic representation)
[:print:]	graphic characters and space

Negative Character Classes: [^]

- Match a single character not in a set
- Complementary sets can contain ranges
- /[^0-9]/
  - I am 42 years old
- /[^[:lower:]]/
  - I am 42 years old

Anchors: ^ $

- Anchor pattern at beginning or end of text/line
- /**^**a/
  - aaaaa
- /a**$**/
  - aaaaa

Examples

- Matching identifiers in a programming language:
  - [A-Za-z_][A-Za-z0-9_]*

ERE

Extended Regular Expressions

Repetitions: {m,n} (1)

- Repeat preceding regex
- {n}   exactly n repetitions
- {0,n} max n repetitions
- {n,} min n repetitions
- {m,n} min m, max n repetitions

Repetitions: {m,n} (2)

- /w**{3}**/
  - **www**.example.com
- /[0-9]**{0,3}**/
  - 0,42,1337
- /[0-9]**{3,}**/
  - 0,42,1337
- /[0-9]**{2,3}**/
  - 0,42,1337

Repetitions: +

- 1..∞ repetitions (equivalent to {1,})
- /Ha**l+**o/
  - Ha**l**o
  - Ha**ll**o
  - Ha**lll**o

Optional: ?

- Preceding expression is optional (equivalent to {0,1})
- /colo**u?**r/
  - color**u**ful
  - colo**u**r**u**ful
- /http**s?**:/
  - http**s**://example.com
  - http**s**://fhlug.at

Groups: ()

- Use parentheses to group regexs
- Modifiers apply to full group
- /(ma)+/
  - madame
  - mama

Alternatives: |

- /a|b/ matches either “a” or “b”
- /Mr|Mrs|Ms/
  - Mr Smith
  - Mrs Coulter
  - Ms Monique
- Must be grouped if a subexpression
- /SE(vz|bb)/
  - SEvz
  - SEbb
  - SExy [no match]

grep

Print lines that match a pattern

man grep

- grep 'PATTERNS' [files…] - BRE
- grep -E 'PATTERNS' - ERE
- grep -F 'PATTERNS' - fixed strings

grep

- `grep '^root:' /etc/passwd`
- `seq 100 | grep '^42$'`

Examples

Putting it all together

Time

- 12 hour format
  - /[0-9]{2}:[0-9]{2} [ap]m/
  - /(0[0-9]|1[012]):[0-5][0-9] [ap]m/
- 24 hour format
  - /([01][0-9]|2[0-3]):[0-5][0-9]/

Date

- /[0-9]{2}.[0-9]{2}.[0-9]{4}/
- /(0[1-9]|[12][0-9]|3[01]).(0[1-9]|1[012]).[0-9]{4}/

E-Mail validation

- /^[^]+@[^]+\.[^]+$/
- Minimal variant: /@/
- RFC822: http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html

Numbers

- Integers: /[+-]?[0-9]+/
- Decimals: /[+-]?[0-9]+(\.[0-9]+)?/
- Scientific notation: /[+-]?[0-9]+(\.[0-9]+)?([eE][+-]?[0-9]+(\.[0-9]+)?)?/

Hex numbers

- /0x[0-9a-fA-F]+/

Hyperlinks

- /https?:\/\/[^]+/

Identifiers

- Identifiers in most programming languages:
- /[A-Za-z_][A-Za-z0-9_]*/

IPv4 Addresses

- `/[0-9]{1,3}(\.[0-9]{1,3}){3}/`
- `/(25[0-5]|2[0-4][0-9]|[01]?[0-9]{1,2})(\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]{1,2})){3}/`

Further Reading

- https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html
  IEEE Std 1003.1, 2018 Edition. Chapter 9, Regular Expressions
- https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM704.pdf
  Representation of Events in Nerve Nets and Finite Automata

- https://regexone.com/ Learn Regular Expressions with simple, interactive exercises
- https://regexr.com/ Learn, Build, & Test RegEx
- https://regex101.com/ build, test, and debug regex
- https://www.debuggex.com/ Online visual regex tester
- http://www.regviz.org/ Visual Debugging of Regular Expressions
- https://regex-vis.com/ Regex Vis
- https://regexper.com/ Regexper