Allgemeines zu Unix (Solaris, Linux, MAC OS X, FreeBSD,Open BSD usw.)

- Multiuser- Multitasking Betrieb
- offenes System
  - unabhängig von den verschiedensten Hardwarekomponenten
  - Benutzeroberflächen folgen einheitlichem Standard
  - Zugriffe auf Datenbanken erfolgen über einheitliche Schnittstellen und Sprachen
  - es gibt einheitliche Betriebsstandards (Kommandos, Systemaufrufe,...)
  - die Bildschirmausgabe ist unabhängig vom Terminaltyp
  - einheitliche Umgebung für Netzwerke, Kommunikation und Transaktion

  Erreicht wurde dies durch Schaffung von Software mit zum Teil einheitlichen Schnittstellen

Aufbau:

- modularer Aufbau
- Module sind austauschbar
- Module gruppieren sich um den **Kernel** (Betriebssystemkern)
- funktionale Trennung zwischen Kernel und Modulen
- Bereich in dem die Module sich um den Kernel gruppieren wird **Shell** genannt

Kernel

- eigentliches Betriebsystem
- befindet sich direkt über der Hardware
- Aufgabe, die Hardware zu betreiben und zu nutzen
- Multiuser/multitasking
- Bestandteile des Kernels müssen auf die Systemhardware zugeschnitten werden (portieren des Linux-Systems, Treiber laden)

Shell:

Shell ist ein Kommandointerpreter zur Kommunikation zwischen Nutzer und Rechner. Sie wird als Prozess aufgerufen der gestartet wird, sobald sich ein Benutzer beim System anmeldet oder sobald als „Batch-Datei ein sogenanntes „Shellscript“ gestartet wird.

- Eingabemöglichkeit des Benutzers
- eingegebener Befehl wird ans Betriebssystem weitergeleitet, Rückmeldung des Betriebssystems wieder auf Shell ausgegeben
- Standard Shell ist bash (Bourn-Again-Shell)
- sehr mächtig (Befehle können verknüpft werden, flexibel, umfangreich)
- Vorteil: Speicherplatz, da keine graphische Oberfläche
- Shell ist selbst letztendlich auch ein Programm, das aufgerufen wird

An-Abmelden:

- Anmeldung nach Systemstart durch **Login** Benutzername
  Zum Schutz vor Zugriff; Identifizierung von Benutzer; Systemressourcen zur Verfügung stellen
- Benutzerpasswort, dann geht es zum zugeteilten „home“Verzeichnis

Benutzername, Passwort, Home-Verzeichnis des Nutzers, stehen in der **Systemdatei/etc/passwd**

Dateizugriffssteuerung:
Geöffnete und geänderte Datei wird nicht direkt auf die Festplatte zurückgeschrieben (Zeitersparnis) Deshalb **nie Rechner direkt ausschalten**, sondern mit *shutdown -h now* oder äquivalenten Befehlen herunterfahren. System nur mit **Logout** oder aus Shell mit **exit** verlassen.

Datei/verzeichnissystem

- Verzeichnissystem ist hierarchisch strukturiert (Baumstruktur)
- ausgehend von Wurzelverzeichnis (ROOT), mit Schrägstrich/ gekennzeichnet, zweigen Verzeichnisse und Unterverzeichnisse ab
- **logisches** Dateisystem (Unix kennt mehrere Festplatten und Partitionierungen dieser Platten, allerdings nicht für Anwender ersichtlich. Anwender arbeitet immer mit einer einheitlichen Verzeichnisstruktur.) **phys.** (man sieht physikalisch wo man ist)
  Das Filesystem kann über mehrere Platten verteilt sein: Transparenz für den Benutzer (Er sieht nur eine Partition, Einbindung können beim booten erfolgen)

- Unix unterscheidet nicht zwischen Dateien und Geräten, ein Gerät wird wie eine Datei betrachtet

Verzeichnisse

/bin:
- ausführbare Dateien (binarys)

/etc:
- Konfigurationsdatei für das Basissystem

/dev:
- devices
- hier stehen die Gerätetreiber

/lib:
- befinden sich C-Bibliotheken

/lost+found:
- für Pflege des Dateisystems zuständig

/usr:
- Anwendungen für den einzelnen user
- z.B. Kommandos und Manpages (Hilfedatei, Beschreibungen)

swap-Bereich:
- Auslagerung des Hauptspeichers, wenn dieser nicht mehr ausreicht
- auf swap können nur das Betriebssystem zugreifen (i.d.R. doppelter Arbeitsspeicher)

Dateien

Dateien =Objekte, die Daten in Form von Bytes aufnehmen, speichern und ausgeben

regular files:
- normale Dateien

directorys:
- Verzeichnisse

devices:
- Gerätedateien, die Dateien unter /dev repräsentieren normalerweise Geräte, Programme, öffnet die Geräte wie normale Dateien; Zugriff wird vom Kernel gesteuert

sockets:
- special files (setzen sich aus der IP-Adesse und einer Portnummer zusammen)
- Spezialdateien aus dem Bereich TCP/IP, mit denen der Datenaustausch zwischen zwei lokal verlaufenden Prozessen über das Dateisystem realisiert werden kann

FIFOs:

- named pipes, stellen eine zweite einfache Methode des Datentransports zwischen zwei Prozessen dar
- first in first out

Links:

- sind zusätzliche Namen (Verzeichniseinträge) für existierende Dateien
- symbolische Links: Spezialdatei, deren Inhalt Zeiger auf eine andere Datei ist
- Hardlinks: gleichwertiger Inhalt einer existierenden Datei

Rechte

Unix bietet wie andere Netzwerkbetriebsysteme eine abgestufte Rechtestruktur an. (Read, Write, Execute)

Es gibt abgestufte Rechte für:

- user
- group
- others

r = read, Verzeichniseinträge sind lesbar
w = write, Verzeichniseinträge können hinzugefügt oder gelöscht werden
x = execute, Name des Verzeichnisses kann in einem Pfad erscheinen, man kann hineinwechseln

Syntax

$ **chmod** u+x, g-w, o-r testdatei

User darf ausführen, Gruppe darf nicht schreiben, andere dürfen nicht lesen

Syntax

$ **chmod** 721 testdatei

User darf alles, Gruppe darf schreiben, Andere dürfen ausführen

Drei Zahlen jeweils für user, group, others
Jede der drei Zahlen eine Summe aus:

- 4=read
- 2=write
- 1=execute

Rechte kann immer nur der Besitzer oder **root** ändern.

Systemverwaltung unter Unix:
Systemverwalter ist der Benutzer **root** der in der Gruppe root Mitglied ist.
Benutzer anlegen:
Benutzer werden mit de Befehl **adduser** angelegt. Nach der Eingabe werden folgende Punkte abgefragt:

1. Benutzername (Loginname)
2. Voller Benutzername
3. Benutzergruppe
4. Benutzername (UID)
5. Stammverzeichnis des Benutzers
6. Benutzer zugeordnete Shell
7. Passwort

Alle Daten werden im Verzeichnis **/etc** in der Datei **passwd** gespeichert. Diese Datei kann nur von **root** verändert werden.

Benutzer entfernen: deluser oder rmuser

Benutzergruppen anlegen:
Benutzergruppen werden im Verzeichnis **/etc** in der Datei **group** verwaltet.
Benutzergruppe anlegen:
groupname:passwd:GDI:user-list

Gruppe: weisen alle ein bestimmtes Merkmal auf.
-> bevor man eine Benutzergruppe anlegt sollte man sich unbedingt Gedanken über die Hierarchie, Rechte usw. machen.

Linux Process-Essentials

Prozesse:
Alle Aktivitäten des Betriebsystems sind Prozesse. Dies sind unabhängige Programme, die durch das Betriebssystem aufgerufen und verwaltet werden können. Das Abbild des Laufenden Systems ist im Verzeichnis
/proc
gespeichert (alle Prozesse, CPU,MEM, IO, INTERRUPTS usw.).

Ein **Elternprozess** (Parent Process) kann wiederum andere Prozesse (Kinderprozesse) aufrufen und seine Prozessumgebung vererben.

Zombie: Prozess ohne Eltern

-> Zur Prozesskontrolle und Prozesssteuerung gibt es folgende Kommandos:
ps, kill, top, exit, batch, nice

Prozesszustände:
Aufgrund des Multitaskingbetriebs von Linux können Prozesse „gleichzeitig" ablaufen.

- Laufend (wird zur Zeit gerade ausgeführt)
- laufbereit (runnable; wurde noch nicht ausgeführt, steht jederzeit zur Ausführung bereit)
- wartend (sleeping; wartet auf ein Ereignis, z.B. ein Druckerinterrupt)
- beendet (terminated; wird aus der Auftragsverwaltung herausgenommen)
- Prozesse werden mit ps oder top (zeigt zusätzlich Prozessorleistung an) aufgerufen.

Folgende Informationen werden mit **ps -alwx** ausgegeben:

UID:	Benutzernummer
PID:	Prozessnummer des laufenden Prozesses
PPID:	Elternprozess
PRI:	Priorität des Prozesses
NI:	Nice Wert, Zweit Priorität
VSIZE:	virtuelle Größe des Prozesses
S:	Status des Prozesses
TTY:	Terminal aus dem gestartet wurde
COMMAND:	Befehl der Prozess ausgelöst hat
WCHAN:	Wait Chanel; Ereignis auf das Prozess wartet
TIME:	Die wirklich verbrauchte Rechenzeit in min und sek.

Prozesskommunikation:
Der Nutzer kann an die von ihm gestarteten Prozesse Signale schicken (der Superuser „root“ allen). Gesendet werden diese Signale durch:

kill –Signalnummer PID

Prozessabrufsignale: -3, -15, -9

Prozesstypen und Jobs:

Es gibt drei verschieden Arten von Arbeitsaufträgen:

1. Hintergrundprozesse:
   Programme oder Kommandos die sehr rechenintensiv sind. Keine Eingabe durch den Benutzer, werden nicht am Bildschirm gesehen.
2. Dämonprozesse:
   Prozesse die bestimmte Dienste umfassen, die nicht im Kernel implementiert sind. Prozesse laufen unabhängig von einem Terminal im Hintergrund. Sind permanent aktiv, werden nicht durch den Benutzer gestartet.
3. Jobs:
   Jedes Kommando das über die Shell eingegeben wird, werden durch die Shell verwaltet

Beispiel zur Prozessgenerierung:

Ein Elternprozess erzeugt mit Hilfe von **fork()** ein Kindprozess, fork() erzeugt einen neuen Prozessbaum,

Ein zweiter Kindprozess wird mit **exec()** erzeugt, excec() überschreibt den eigenen Prozessbaum mit einem neuen.

Nach Beendigung des zweiten Kindprozesses wird dieser zu einem Zombie.

Im **Zombiezustand** wartet der Prozess auf den von fork() automatisch erzeugten wait()-Punkt.

Wirkt dieser wait-Punkt nicht, muss der Prozess per Hand beendet werden.

Der Linux-Kernel:

Linux unterscheidet die Usermode (Benutzerebene) und den Kernelmode (Kernelebene).