Aufbau eines Betriebssystems

Zielgruppe: Lernende Informatik EFZ in den Fachrichtungen Plattformentwicklung und Applikationsentwicklung.

Ausgangslage

Ein Betriebssystem ist die zentrale Software zwischen Hardware, Anwendungen und Benutzenden. Es verwaltet Prozessorzeit, Arbeitsspeicher, Dateien, Geräte, Netzwerkverbindungen, Benutzerrechte, Dienste und laufende Programme.

Für Informatikerinnen und Informatiker ist dieses Verständnis wichtig, weil viele technische Probleme nur erklärbar sind, wenn man die Rolle des Betriebssystems versteht:

- Warum ist ein Computer langsam, obwohl nur wenige Fenster offen sind?
- Warum kann ein Prozess nicht auf eine Datei zugreifen?
- Warum belegt eine Anwendung viel Arbeitsspeicher?
- Warum startet ein Dienst nach einem Neustart automatisch?
- Warum funktioniert ein Netzwerkdienst lokal, aber nicht von aussen?
- Warum verhalten sich Windows und Linux bei Pfaden, Rechten und Services unterschiedlich?

Diese Unterrichtseinheit vermittelt nicht nur Begriffe, sondern zeigt, wie die wichtigsten Bestandteile eines Betriebssystems zusammenarbeiten.

Prüfe, dass du WSL in deiner Umgebung installiert hast oder Zugriff auf ein Linux System hast.

Lernziele

Die Ziele können aufgeteilt werden nach folgenden Punkte:

- erklären, welche Aufgaben ein Betriebssystem hat.
- den Unterschied zwischen Kernel, Shell, Services und Anwendungen beschreiben.
- Prozesse, Threads und Zeitscheiben grundlegend erklären.
- beschreibe, wie ein Betriebssystem CPU-Zeit auf mehrere Prozesse verteilt.
- Arbeitsspeicher, virtuellen Speicher und Swap/Pagefile unterscheiden.
- Dateisysteme, Laufwerke, Mount Points und Berechtigungen vergleichen.
- grundlegende Unterschiede zwischen Windows und Linux benennen.
- erklären, wie Netzwerkzugriffe über IP-Adressen, Ports und Dienste funktionieren.
- Services unter Windows und Linux einordnen.
- typische Diagnosewerkzeuge für Prozesse, Speicher, Disk und Netzwerk nennen.
- einschätzen, welche Betriebssystemkonzepte für Docker, Serverbetrieb und Entwicklung wichtig sind.

Grundidee eines Betriebssystems

Ein Betriebssystem hat drei zentrale Aufgaben:

1. Es verwaltet Hardware.
2. Es stellt Anwendungen eine einheitliche Umgebung bereit.
3. Es schützt Benutzer, Prozesse und Daten voreinander.

Ohne Betriebssystem müsste jede Anwendung selbst wissen, wie man Tastatur, Bildschirm, Festplatte, Netzwerk, Speicher und CPU verwendet. Das Betriebssystem stellt dafür

Schnittstellen bereit.

Typische Bestandteile:

- Kernel: Kern des Betriebssystems, verwaltet Hardware und Systemressourcen.
- Treiber: Software zur Ansteuerung bestimmter Hardware.
- Dateisystem: Organisation von Dateien und Verzeichnissen.
- Prozessverwaltung: Starten, Planen, Überwachen und Beenden von Programmen.
- Speicherverwaltung: Zuteilung und Schutz von Arbeitsspeicher.
- Netzwerkstack: Umsetzung von Netzwerkprotokollen wie TCP/IP.
- Benutzer- und Rechteverwaltung: Kontrolle, wer was tun darf.
- Systemdienste: Hintergrundprogramme für dauerhafte Aufgaben.
- Shell oder grafische Oberfläche: Bedienung durch Menschen.
- Anwendungen: Programme wie Browser, Editor, Datenbank oder Entwicklungsumgebung.

Schichtenmodell

Vereinfacht kann man ein Betriebssystem in Schichten betrachten:

```
Benutzerin / Benutzer
        |
Anwendungen
        |
Shell, GUI, Systembibliotheken
        |
Systemaufrufe
        |
Kernel
        |
Treiber
        |
Hardware
```

Beispiel: Eine Anwendung möchte eine Datei speichern.

1. Die Anwendung ruft eine Betriebssystemfunktion auf.
2. Der Kernel prüft Rechte und Pfade.
3. Das Dateisystem entscheidet, wo die Daten gespeichert werden.
4. Der passende Treiber spricht das Speichergerät an.
5. Die Hardware schreibt die Daten.

Kernel

Der Kernel ist der wichtigste Teil eines Betriebssystems. Er läuft mit sehr hohen Rechten und verwaltet zentrale Ressourcen.

Aufgaben des Kernels:

- Prozesse und Threads planen.
- Arbeitsspeicher zuteilen und schützen.
- Dateizugriffe koordinieren.
- Netzwerkpakete verarbeiten.
- Hardware über Treiber ansprechen.
- Systemaufrufe von Anwendungen entgegennehmen.

- Rechte und Isolation durchsetzen.

Anwendungen laufen normalerweise nicht direkt mit Kernelrechten. Sie verwenden Systemaufrufe, um kontrolliert auf Ressourcen zuzugreifen.

Windows und Linux beim Kernel

Frage doch mal bei AI nach “Gibt es nicht zwei unterschiedliche Kernelkategorien”?

- Monolithischer Kernel
- Mikrokernel

Windows:

- verwendet den Windows NT Kernel
- ist stark mit Windows-spezifischen Systemdiensten, Registry, Treibermodell und grafischer Oberfläche integriert
- bietet Win32, .NET und weitere APIs für Anwendungen

Linux:

- verwendet den Linux-Kernel.
- ist nur der Kernel; eine vollständige Linux-Distribution besteht zusätzlich aus Shell, Paketmanager, Systemdiensten, Bibliotheken und Werkzeugen.
- wird in vielen Varianten eingesetzt, zum Beispiel Ubuntu, Debian, Fedora, Alpine oder Red Hat Enterprise Linux.

Wichtig: “Linux” meint im Alltag oft die ganze Distribution. Technisch ist Linux aber der Kernel.

Prozesse, Threads und Zeitscheiben

Ein Prozess ist ein laufendes Programm mit eigenem Speicherbereich und eigenen Ressourcen. Beispiele:

- Browser
- Terminal
- Webserver
- Datenbank
- Editor

Ein Thread ist ein Ausführungsstrang innerhalb eines Prozesses. Ein Prozess kann einen oder mehrere Threads besitzen.

Beispiel:

- Ein Browserprozess kann einen Thread für die Benutzeroberfläche verwenden.
- Weitere Threads laden Webseiten, führen JavaScript aus oder verarbeiten Netzwerkdaten.

Zeitscheiben

Ein Computer führt scheinbar viele Programme gleichzeitig aus. In Wirklichkeit verteilt das Betriebssystem die CPU-Zeit rasant zwischen Prozessen und Threads.

Eine Zeitscheibe ist ein kurzer Zeitraum, in dem ein Prozess oder Thread auf der CPU laufen darf. Danach kann der Scheduler entscheiden, ob ein anderer Prozess CPU-Zeit erhält.

Vereinfacht:

```
Zeit 1: Browser läuft
Zeit 2: Editor läuft
Zeit 3: Webserver läuft
Zeit 4: Browser läuft wieder
```

Dieses schnelle Umschalten heisst Kontextwechsel. Dabei speichert das Betriebssystem den Zustand des aktuellen Prozesses und lädt den Zustand des nächsten Prozesses.

Scheduling

Der Scheduler ist der Teil des Betriebssystems, der entscheidet, welcher Prozess oder Thread als Nächstes CPU-Zeit bekommt.

Kriterien können sein:

- Priorität
- Wartezeit
- CPU-Auslastung
- Interaktive Reaktionszeit
- Hintergrund- oder Vordergrundprozess
- I/O-Wartezeiten, zum Beispiel Warten auf Disk oder Netzwerk

Windows und Linux verwenden unterschiedliche Scheduler-Implementierungen, verfolgen aber dasselbe Ziel: Die verfügbare CPU-Zeit soll fair, performant und passend zur Nutzung verteilt werden.

Arbeitsspeicher und virtueller Speicher

Der Arbeitsspeicher, RAM, ist ein schneller Speicher für aktuell benötigte Daten und Programme.

Das Betriebssystem verwaltet:

- welcher Prozess welchen Speicherbereich bekommt
- welche Speicherbereiche geschützt sind
- welche Daten gerade im RAM liegen
- welche Daten auf Disk ausgelagert werden

Virtueller Speicher

Moderne Betriebssysteme geben Prozessen einen virtuellen Adressraum. Ein Prozess sieht dadurch scheinbar seinen eigenen Speicherbereich. Der Kernel ordnet diese virtuellen Adressen dem echten physischen RAM zu.

Vorteile:

- Prozesse sind voneinander isoliert.
- Ein fehlerhafter Prozess kann nicht einfach fremden Speicher überschreiben.
- Speicher kann effizienter genutzt werden.
- Daten können bei Bedarf ausgelagert werden.

Swap und Pagefile

Wenn der RAM knapp wird, können Betriebssysteme Teile des Speichers auf Disk auslagern.

Linux:

- verwendet Swap, entweder als Swap-Partition oder Swap-Datei.
- Werkzeuge: `free`, `top`, `htop`, `swapon`.

Windows:

- verwendet ein Pagefile, zum Beispiel `pagefile.sys`.
- Werkzeuge: Task-Manager, Ressourcenmonitor, Performance Monitor.

Auslagerung ist langsamer als RAM. Wenn ein System stark swappt oder viel ins Pagefile schreibt, fühlt es sich oft sehr langsam an.

Disk, Dateisysteme und Pfade

Das Betriebssystem organisiert dauerhafte Daten auf Speichergeräten wie SSDs, HDDs oder virtuellen Disks.

Wichtige Begriffe:

- Blockgerät: Speichergerät, das blockweise gelesen und geschrieben wird.
- Partition: Bereich auf einem Datenträger.
- Dateisystem: Struktur, wie Dateien und Verzeichnisse gespeichert werden.
- Mount: Einbinden eines Dateisystems in die Verzeichnisstruktur.
- Berechtigungen: Regeln, wer Dateien lesen, schreiben oder ausführen darf.

Windows

Typisch für Windows:

- Laufwerksbuchstaben wie `C:\`, `D:\`.
- Pfadtrenner ist meistens `\`.
- häufiges Dateisystem: NTFS.
- Berechtigungen über NTFS Access Control Lists.
- zentrale Konfiguration teilweise in der Registry.
- Programme oft unter `C:\Program Files`.

Beispiel:

```
C:\Users\lernende\Documents\projekt\app.exe
```

Linux

Typisch für Linux:

- ein gemeinsamer Verzeichnisbaum mit `/` als Wurzel.
- keine Laufwerksbuchstaben.
- Dateisysteme werden in Verzeichnisse eingehängt.
- Pfadtrenner ist `/`.
- häufige Dateisysteme: ext4, xfs, btrfs.
- Berechtigungen über Besitzer, Gruppe und Rechtebits.
- Konfiguration oft als Textdateien unter `/etc`.

Beispiel:

```
/home/lernende/projekt/app
```

Wichtige Linux-Verzeichnisse:

- `/home`: Benutzerverzeichnisse.
- `/etc`: Systemkonfiguration.

- `/var`: veränderliche Daten, Logs, Spools, Datenbanken.
- `/usr`: installierte Programme und Bibliotheken.
- `/bin` und `/sbin`: wichtige Systemprogramme.
- `/tmp`: temporäre Dateien.
- `/dev`: Gerätedateien.
- `/proc`: laufende Kernel- und Prozessinformationen.

Netzwerk

Das Betriebssystem stellt den Netzwerkstack bereit. Anwendungen müssen TCP/IP nicht selbst komplett implementieren, sondern Betriebssystemfunktionen verwenden.

Wichtige Begriffe:

- IP-Adresse: Adresse eines Geräts oder Interfaces im Netzwerk.
- Port: Nummer für einen Dienst auf einem System.
- TCP: verbindungsorientiertes Transportprotokoll.
- UDP: verbindungsloses Transportprotokoll.
- DNS: Namensauflösung von Namen zu IP-Adressen.
- Socket: Kommunikationsendpunkt für Netzwerkverbindungen.
- Firewall: Regelwerk, das Netzwerkverkehr erlaubt oder blockiert.

Beispiel:

```
Browser -> TCP-Verbindung -> 93.184.216.34:443 -> Webserver
```

Der Port entscheidet, welcher Dienst angesprochen wird:

- `22`: SSH
- `53`: DNS
- `80`: HTTP
- `443`: HTTPS
- `5432`: PostgreSQL
- `3306`: MySQL/MariaDB

Netzwerk unter Windows und Linux

Windows:

- Konfiguration oft über grafische Einstellungen, PowerShell oder `netsh`.
- Firewall über Windows Defender Firewall.
- Diagnose mit `ipconfig`, `ping`, `tracert`, `netstat`, PowerShell-Cmdlets.

Linux:

- Konfiguration je nach Distribution über NetworkManager, systemd-networkd oder Konfigurationsdateien.
- Firewall zum Beispiel über `nftables`, `iptables` oder `ufw`.
- Diagnose mit `ip`, `ping`, `traceroute`, `ss`, `dig`, `curl`.

Services und Daemons

Services sind Programme, die im Hintergrund laufen und Aufgaben bereitstellen. Unter Linux spricht man häufig von Daemons.

Beispiele:

- Webserver

- Datenbankserver
- SSH-Server
- Druckdienst
- Update-Dienst
- Monitoring-Agent
- DNS-Server

Windows Services

Windows verwaltet Hintergrunddienste über den Service Control Manager.

Werkzeuge:

- Dienste-App
- Task-Manager
- PowerShell: `Get-Service`, `Start-Service`, `Stop-Service`
- `sc.exe`

Beispiel:

```
Get-Service
Get-Service -Name Spooler
Restart-Service -Name Spooler
```

Linux systemd Services

Viele moderne Linux-Distributionen verwenden `systemd` zur Verwaltung von Diensten.

Werkzeuge:

- `systemctl`
- `journalctl`

Beispiel:

```
systemctl status ssh
sudo systemctl restart ssh
journalctl -u ssh
```

Ein Service kann automatisch beim Systemstart gestartet werden. Das ist wichtig für Serverdienste wie Webserver, Datenbanken oder Container-Runtimes.

Anwendungen

Anwendungen sind Programme, die eine Aufgabe für Benutzer oder andere Systeme erledigen.

Beispiele:

- Browser
- Office-Programme
- Entwicklungsumgebung
- Datenbank-Client
- Webanwendung
- Serverprozess
- Kommandozeilenwerkzeug

Anwendungen nutzen Betriebssystemfunktionen für:

- Dateien
- Netzwerk
- Speicher
- Prozesse
- Benutzerrechte
- Eingabe und Ausgabe
- grafische Darstellung

Eine Anwendung kann direkt von einem Menschen gestartet werden oder als Service dauerhaft im Hintergrund laufen.

Unterschiede zwischen Windows und Linux

Bereich	Windows	Linux
Kernel	Windows NT Kernel	Linux-Kernel
Distribution	Ein Produkt mit Editionen	Viele Distributionen
Bedienung	stark GUI-orientiert, PowerShell wichtig	Shell sehr zentral, GUI optional
Pfade	`C:\Users\...`	`/home/...`
Laufwerke	Laufwerksbuchstaben	Ein Verzeichnisbaum mit Mount Points
Konfiguration	GUI, Registry, Dateien, Gruppenrichtlinien	meist Textdateien unter `/etc`, Tools
Softwareinstallation	Installer, Microsoft Store, winget	Paketmanager wie `apt`, `dnf`, `pacman`, `apk`
Services	Windows Services	Daemons, oft mit `systemd`
Rechte	NTFS ACLs, Benutzer, Gruppen, UAC	Besitzer, Gruppe, Rechtebits, ACLs, sudo
Logs	Ereignisanzeige, Event Viewer	Logdateien, `journalctl`
Netzwerktools	`ipconfig`, `netstat`, PowerShell	`ip`, `ss`, `dig`, `curl`
Gross-/Kleinschreibung	Dateinamen meist nicht case-sensitive	Dateinamen normalerweise case-sensitive
Typischer Einsatz	Desktop, Unternehmensumgebungen, Server	Server, Cloud, Container, Embedded, Entwicklung

Zusammenspiel an einem Beispiel

Beispiel: Ein Webserver läuft auf einem Linux-System.

1. Beim Systemstart startet `systemd` den Webserver-Service.
2. Der Webserver-Prozess wird vom Kernel erzeugt.

3. Der Scheduler gibt dem Prozess CPU-Zeit.
4. Der Prozess reserviert RAM für Konfiguration, Verbindungen und Cache.
5. Konfigurationsdateien werden aus `/etc` gelesen.
6. Webseiten oder Anwendungsdaten werden von Disk geladen.
7. Der Webserver öffnet einen Netzwerkport, zum Beispiel 443.
8. Die Firewall erlaubt oder blockiert eingehende Verbindungen.
9. Clients verbinden sich über IP-Adresse und Port.
10. Logs werden geschrieben und können mit `journalctl` oder Logdateien ausgewertet werden.

Dieses Beispiel zeigt, dass Kernel, Prozessverwaltung, Speicher, Disk, Netzwerk, Rechte und Services immer zusammenwirken.

Diagnosewerkzeuge

Windows

Aufgabe	Werkzeug
Prozesse anzeigen	Task-Manager, `Get-Process`
Services anzeigen	Dienste-App, `Get-Service`
Speicher prüfen	Task-Manager, Ressourcenmonitor
Disk prüfen	Datenträgerverwaltung, Ressourcenmonitor
Netzwerk anzeigen	`ipconfig`, `netstat`, PowerShell
Logs anzeigen	Ereignisanzeige

Linux

Aufgabe	Werkzeug
Prozesse anzeigen	`ps`, `top`, `htop`
Services anzeigen	`systemctl`
Speicher prüfen	`free`, `top`, `vmstat`
Disk prüfen	`lsblk`, `df`, `du`, `mount`
Netzwerk anzeigen	`ip`, `ss`, `ping`, `curl`, `dig`
Logs anzeigen	`journalctl`, Dateien unter `/var/log`

Wichtige Begriffe

- Betriebssystem: Software, die Hardware verwaltet und Anwendungen ausführt.
- Kernel: Kern des Betriebssystems mit Zugriff auf zentrale Ressourcen.
- Treiber: Software zur Ansteuerung von Hardware.
- Prozess: laufendes Programm.
- Thread: Ausführungsstrang innerhalb eines Prozesses.

- Scheduler: entscheidet über CPU-Zeit.
- Zeitscheibe: kurzer Zeitraum, in dem ein Prozess oder Thread CPU-Zeit erhält.
- Kontextwechsel: Umschalten zwischen Prozessen oder Threads.
- RAM: schneller Arbeitsspeicher.
- Virtueller Speicher: Speicherabstraktion für Prozesse.
- Swap/Pagefile: Auslagerung von Speicher auf Disk.
- Dateisystem: Struktur zur Organisation von Dateien.
- Mount Point: Einhängepunkt eines Dateisystems.
- Service/Dämon: Hintergrundprozess für dauerhafte Aufgaben.
- Port: Nummer eines Netzwerkdienstes.
- Socket: Netzwerkendpunkt einer Anwendung.
- Shell: die textbasierte Schnittstelle zum Betriebssystem.
- GUI: grafische Benutzeroberfläche.

Kontrollfragen

1. Welche Aufgaben hat ein Betriebssystem?
2. Warum dürfen Anwendungen normalerweise nicht direkt auf Hardware zugreifen?
3. Was ist der Unterschied zwischen Kernel und Anwendung?
4. Was ist ein Prozess?
5. Was ist ein Thread?
6. Was bedeutet Zeitscheibe?
7. Warum kann ein Computer mehrere Programme scheinbar gleichzeitig ausführen?
8. Was passiert bei einem Kontextwechsel?
9. Warum verwendet ein Betriebssystem virtuellen Speicher?
10. Was ist der Unterschied zwischen RAM und Swap/Pagefile?
11. Wie unterscheiden sich Windows- und Linux-Pfade?
12. Was ist ein Mount Point?
13. Was ist der Unterschied zwischen einer Anwendung und einem Service?
14. Warum braucht ein Webserver einen Port?
15. Welche Werkzeuge würdest du unter Linux verwenden, um Prozesse, Speicher, Disk und Netzwerk zu prüfen?

Praktische Aufgaben

Aufgabe 1: Prozesse vergleichen

Windows:

```
Get-Process
```

Linux:

```
ps aux
top
```

Auftrag:

- Suche drei laufende Prozesse.
- Notiere den Namen, ungefähre Speicherverwendung und Zweck.
- Unterscheide Benutzerprozess und Systemprozess.

Aufgabe 2: Services untersuchen

Windows:

```
Get-Service
```

Linux:

```
systemctl --type=service
```

Auftrag:

- Suche drei Services.
- Kläre, ob sie laufen oder gestoppt sind.
- Beschreibe, wofür sie vermutlich gebraucht werden.

Aufgabe 3: Speicher beobachten

Windows:

- Task-Manager öffnen.
- Reiter “Leistung” betrachten.

Linux:

```
free -h
top
```

Auftrag:

- Notiere RAM-Grösse und aktuelle Auslastung.
- Prüfe, ob Swap oder Pagefile verwendet wird.
- Starte eine grössere Anwendung und beobachte die Veränderung.

Aufgabe 4: Disk und Pfade vergleichen

Windows:

```
Get-PSDrive
```

Linux:

```
df -h
lsblk
mount
```

Auftrag:

- Vergleiche Laufwerksbuchstaben und Mount Points.
- Finde dein Benutzerverzeichnis.
- Erkläre den Unterschied zwischen `C:\Users\...` und `/home/...`.

Aufgabe 5: Netzwerkdienste prüfen

Windows:

```
ipconfig
netstat -ano
```

Linux:

```
ip addr
ss -tulpen
```

df = Disk Free, zeigt die Belegung von Dateisystemen an. -h = human-readable, zeigt die Belegung in lesbaren Einheiten (z.B. MB, GB) an.

Auftrag:

- Finde die lokale IP-Adresse.
- Suche offene Ports.
- Kläre, welcher Prozess oder Dienst einen Port verwendet.

Merksatz

Ein Betriebssystem ist nicht nur eine Oberfläche. Es ist die Schicht, die Hardware, Prozesse, Speicher, Dateien, Netzwerk, Dienste, Sicherheit und Anwendungen kontrolliert und miteinander verbindet.

Zeitbudget

Zeit: ca. 240min