Betriebssysteme - Das ultimative Cheat Sheet

Basierend auf Kurs 01670 - FernUniversität in Hagen

1 KE 1: Einführung & Grundlagen

1.1 Was ist ein Betriebssystem (BS)?

Abstrakte Maschine: Verbirgt Komplexität der Hardware, bietet einfache Schnittstelle (virtuelle Maschine).

Ressourcen-Manager: Verwaltet & verteilt Ressourcen (CPU, Speicher, Geräte) fair und effizient.

1.2 Klassische Aufgaben eines BS

- **Prozessverwaltung:** Erstellen, Beenden, Steuern von Prozessen
- **Speicherverwaltung:** Zuweisung und Verwaltung des Hauptspeichers
- **Geräteverwaltung:** Ansteuerung von Hardware über Treiber
- **Dateisystem-Management:** Organisation in Dateien & Verzeichnissen
- **Schutz & Sicherheit:** Prozesse voneinander isolieren
- **Benutzerschnittstelle:** Shell (CLI) oder GUI

1.3 Architektur & Kernkonzepte

- **Schichtenmodell:** Hardware → Kernel → Systemprogramme → Anwendungen
- **System- vs. Benutzermodus (Dual Mode):**
  - *User Mode:* Eingeschränkte Rechte für Anwendungen
  - *Kernel Mode:* Volle Rechte für den BS-Kern. Kritische Befehle (z.B. I/O) nur hier möglich
- **Systemaufruf (Trap):** Der einzige, kontrollierte Weg vom User- in den Kernel-Modus
- **Interrupts:** Signal von Hardware (Geräte, Timer) an die CPU, das die Kontrolle an den Kernel übergibt

1.4 Prozesszustände (5-Zustands-Modell)

1. **erzeugt (new):** Prozess wird erstellt
2. **bereit (ready):** Wartet auf CPU
3. **rechnend (running):** Wird von CPU ausgeführt
4. **blockiert (waiting):** Wartet auf Ereignis (z.B. I/O)
5. **beendet (terminated):** Abgeschlossen

1.5 Prozesskontrollblock (PCB)

Die "Visitenkarte"des Prozesses für das BS. Enthält: PID, Zustand, Programmzähler, Register, Speicherinfos, offene Dateien etc.

2 KE 2: Prozesse & Scheduling

2.1 Programm vs. Prozess

- **Programm:** Passive Datei mit Anweisungen
- **Prozess:** Programm in Ausführung. Aktive Entität mit Zustand (Code, Daten, Programmzähler, Register, Stack)

2.2 Threads (Leichtgewichtige Prozesse)

- Ein Prozess kann mehrere Threads haben
- **Teilen sich:** Code, Daten, geöffnete Dateien
- **Haben EIGENEN:** Programmzähler, Register, Stack
- **Vorteil:** Schnelleres Erstellen/Umschalten, einfache Kommunikation

2.3 Scheduling (CPU-Zuteilung)

Der **Scheduler** wählt den nächsten Prozess aus der 'bereit'-Liste.

Algorithmus	Vorteil	Nachteil
FCFS (nicht-präemptiv)	Einfach, fair	Kurze Prozesse warten lange
SJF (nicht-präemptiv)	Optimal bzgl. avg. Wartezeit	Lange Prozesse können verhungern
Priority (präemptiv/nicht-p.)	Wichtige Prozesse bevorzugt	Niedrige Prioritäten können verhungern
Round Robin (präemptiv)	Sehr fair, gut für interaktive Systeme	Overhead durch Kontextwechsel

3 KE 3: Hauptspeicherverwaltung

3.1 Grundproblem & MMU

Logische Adressen eines Prozesses auf physische Adressen im RAM abbilden. Erledigt von der **Memory Management Unit (MMU)**.

3.2 Techniken

Zusammenhängende Zuweisung:

- MFT (Feste Partitionen): → **interne** Fragmentierung
- MVT (Variable Partitionen): → **externe** Fragmentierung

Paging (Seitenorientiert):

- Physischer Speicher → **Seitenrahmen (Frames)**
- Logischer Speicher → **Seiten (Pages)**
- **Seitentabelle** pro Prozess bildet Pages auf Frames ab
- Löst externe Fragmentierung

Virtueller Speicher:

- **Konzept:** Log. Adressraum ¿ phys. Speicher. Teile des Prozesses liegen auf Festplatte (Swap-Bereich)
- **Mechanismus: Demand Paging.** Seite wird erst bei Bedarf geladen
- **Seitenfehler (Page Fault):** Zugriff auf eine nicht geladene Seite
- **Seitenauslagerung:** Wenn kein Rahmen frei ist, wird eine Seite ersetzt (z.B. via **LRU**-Algorithmus)
- **Thrashing:** System ist nur noch am Seiten-Swappen, weil Prozesse zu wenige Frames haben

4 KE 4: Synchronisation & Deadlocks

4.1 Problem: Race Conditions

In **kritischen Abschnitten**, wo mehrere Prozesse auf gemeinsame Ressourcen zugreifen. Benötigt **wechselseitigen Ausschluss**.

4.2 Mechanismen

Semaphore (Dijkstra): Zählervariable mit atomaren Operationen:

- `down()` (P, wait): Blockiert, wenn Zähler ≤ 0
- `up()` (V, signal): Weckt ggf. wartenden Prozess

Monitore: Hochsprachenkonstrukt, das Daten und Prozeduren kapselt und Mutex automatisch sicherstellt.

4.3 Deadlocks

Definition: Zyklisches Warten von Prozessen auf Ressourcen.

4 notwendige Bedingungen:

1. Wechselseitiger Ausschluss
2. Hold and Wait
3. Keine Unterbrechung
4. Zyklisches Warten

Umgang: Verhinderung, Vermeidung, Erkennung & Behebung, oder Ignorieren ("Vogel-Strauß-Algorithmus").

5 KE 5: Geräte- & Dateiverwaltung

5.1 Geräteverwaltung

- **Controller:** Hardware, die Geräte steuert
- **Gerätetreiber:** Software, die mit dem Controller kommuniziert
- **DMA (Direct Memory Access):** Ermöglicht Datentransfer zwischen Gerät und Speicher ohne CPU-Beteiligung

5.2 Dateisystem

- **Datei:** Abstraktion für permanenten Speicher
- **Verzeichnis:** Hierarchische Struktur zur Organisation
- **Dateizuordnung** (Wie werden Blöcke gespeichert?):
  - **FAT (File Allocation Table):** Verkettete Liste der Blöcke in zentraler Tabelle
  - **i-node (Index-Node):** Datenstruktur pro Datei mit Metadaten und Zeigern (direkt/indirekt) auf Datenblöcke (Standard in UNIX)

6 KE 6: Sicherheit

6.1 Ziele & Begriffe

- **Ziele:** Vertraulichkeit, Integrität, Verfügbarkeit
- **Subjekt:** Aktive Entität (Prozess, Benutzer)
- **Objekt:** Passive Entität (Datei, Gerät)

6.2 Kernmechanismen

Authentisierung: Wer bist du? (Passwort)

Autorisierung (Zugriffskontrolle): Was darfst du?

- **ACL (Access Control List):** Pro Objekt eine Liste mit Rechten für Subjekte
- **Capability:** Pro Subjekt eine Liste mit Tickets für Objekte

DAC (Discretionary): Besitzer legt Rechte fest

MAC (Mandatory): System erzwingt globale Regeln

- **Bell-LaPadula (Vertraulichkeit):** No Read Up, No Write Down"
- **Biba (Integrität):** No Read Down, No Write Up"

7 KE 7: Kommandosprachen

7.1 Kommandointerpreter (Shell)

- Schnittstelle zwischen Benutzer und BS
- Startet Prozesse (typisch: `fork()` & `exec()`)
- Verwaltet die Prozessumgebung (Variablen, offene Dateien)
- **I/O-Umlenkung:**
  - `>` leitet Ausgabe in Datei um
  - `<` liest Eingabe aus Datei
  - `|` (Pipe) leitet Ausgabe eines Prozesses als Eingabe an den nächsten weiter
- **Shell-Skripte:** Automatisierung von Kommandoabfolgen mit Variablen und Kontrollstrukturen