Lernzettel

Systemprogrammierung

Universität: Technische Universität Berlin
Kurs/Modul: Systemprogrammierung
Erstellungsdatum: September 6, 2025

Lernzettel: Systemprogrammierung

(1) Einführung.
Systemprogrammierung behandelt maschinennahes Programmieren, Aufbau und Funktionsweise von Betriebssystemen sowie deren Ressourcenverwaltung. Ziel ist, theoretische Grundlagen zu verstehen, zu rekonstruieren und prototypisch in C umzusetzen. Typische Aspekte sind Nebenläufigkeit, Synchronisation, Kommunikation, Betriebsmittelverwaltung, Speicherhierarchie, Scheduling, Interrupt-Handling und Fehlerbehandlung.

(2) Nebenläufigkeit – Prozesse, Threads und Scheduling.
Nachrichten- oder zeitgesteuerte Ausführung von Abläufen, um Ressourcen effizient zu nutzen.
Prozesse vs. Threads. Ein Prozess besitzt eigenständigen Adressraum; Threads teilen sich denselben Adressraum desselben Prozesses und ermöglichen leichtere Kontextwechsel.
Scheduling. Ziele: faire Zuteilung, kurze Reaktionszeiten, hoher Durchsatz. Typische Strategien:
FCFS (First-Cit-First-Served), Round-Robin (Quantenvorgaben), Prioritätenplanung, sowie Mischformen.
Wichtige Größen: Quantenlänge q (CPU-Zeitquantum), Wartezeit W_i, Gesamtdurchlauf.
Kontextwechsel. Kostenfaktor durch das Speichern/ Wiederherstellen von Prozess- bzw. Thread-Kontexten; beeinflusst Gesamtsystemleistung.

(3) Synchronisation und Kommunikation.
Vermeidung von Race Conditions, No-Deadlock-Design und effiziente Interprozesskommunikation.
Synchronisationsmechanismen. Mutex, Semaphoren, Monitore, Barrieren.
Interprozesskommunikation (IPC). Pipes, Shared Memory, Message Queues, Sockets.
Deadlocks und Vermeidung. Ressourcenzyklus, Vermeidung durch Ressourcenordering, Deadlock-Prävention/ -Vermeidung.

(4) Betriebsmittelverwaltung – Geräteunabhängigkeit der Ein-/Ausgabe und Treiber.
Abstraktion der Hardware über Treiber-Stacks; Geräteunabhängigkeit durch I/O-Subsysteme.
Treiberarchitektur. Kernel-Module, Hooks, Interrupt-Handling-Interfaces.
I/O-Subsystem. Puffern, DMA, Interrupt-gestützte Verarbeitung, Polling als Alternative.

(5) Speicherhierarchie und Virtualisierung.
Speicherhierarchie. Cache, Hauptspeicher (DRAM), TLB, Paging, Paging- und Cache-Kohärenz.
Virtualisierung. Hypervisoren (Typ-1, Typ-2), Gastsysteme, Speicher- und Geräteeinabstraktion.

(6) Programmiertechnik – Exception Handling und Interrupt Handling.
Exception Handling. Fehlerbehandlung auf System-/Anwendungsebene, Rückgabewerte, errno, robuste Fehlerkette.
Interrupt Handling. Interrupt-Vector, ISR (Interrupt Service Routine), Maskierung, Prioritätskanäle, latency.

(7) Sicherheit, Gesellschaftliche Verantwortung und Nachhaltigkeit.
Sichere Betriebssysteme zur Vermeidung von Daten- und Identitätsdiebstahl, Absicherung kritischer Infrastrukturen.
Ressourceneffizienz, Energieverbrauch senken, verantwortungsvolle Entwicklung und Wartung.

(8) Prototypische Umsetzung in C – kleine Programme.
Ziel ist es, zentrale Konzepte in C zu implementieren, z. B. Threads via POSIX Threads (`pthread_create`, `pthread_join`), Synchronisation mit Mutex/Semaphore, einfache IPC-Szenarien, und einfache Treiber-/Konsolen-Interaktionen.