1. Zusammenfassung

Ziel der ersten Kurseinheit war, Ihnen eine erste Vorstellung von dem Aufgabenbereich und der Funktionsweise eines Betriebssystems zu vermitteln. Es gibt keine universell gültige Menge von Aufgaben eines Betriebssystems; die Aufgaben und Funktionen hängen ab von dem vorgesehenen Anwendungsbereich des Rechners, der zugrundeliegenden Hardware und den Kosten des Rechners (in Relation zu den finanziellen Möglichkeiten seiner Benutzer). Der große Einfluss der Kosten auf die von einem Betriebssystem anzubietenden Funktionen zeigte sich am deutlichsten am Stapelbetrieb, der in erster Linie der Kostenminimierung dient. Wir haben auch gesehen, dass Betriebssysteme neben Datenbankmanagementsystemen, Datenübertragungssystemen und Fenster- oder Graphik-Systemen nur ein Teil der Systemsoftware sind und dass die Grenzen zwischen diesen Bereichen oft nicht ganz scharf sind. Die wichtigsten und häufigsten Aufgabengebiete bzw. Dienstleistungen eines Betriebssystems sind: Gerätesteuerung, Schutz, Fehlerbehandlung, Realisierung von parallelen Prozessen, Prozess-Synchronisation bzw. -Kommunikation, Betriebsmittelverwaltung, Realisierung einer Kommandosprache und Administration. Applikationen können die Dienste des Betriebssystems über die (Applikations-) Programmierschnittstelle in Anspruch nehmen. Das gesamte Betriebssystem besteht aus Dienstprogrammen, Bibliotheken und dem Betriebssystemkern; der Kern selbst besteht aus einem hardwareabhängigen und einem hardwareunabhängigen Teil. Die Funktion des Betriebssystemkerns basiert ganz wesentlich auf Unterbrechungen, die z.B. von Geräten erzeugt werden können und einen asynchronen Aufruf einer Unterbrechungsroutine verursachen. Durch einen Zeitgeber können regelmäßige Unterbrechungen erzeugt werden, die das Betriebssystem dazu benutzen kann, die CPU abwechselnd verschiedenen Prozessen zuzuteilen und damit mehrere Programme scheinbar parallel auszuführen. Ob einzelne Funktionen in einem Betriebssystem realisiert werden können, hängt vielfach davon ab, ob die vorhandene Hardware dies unterstützt. Erste Beispiele, die wir in dieser Kurseinheit kennengelernt haben, waren: Speicherschutzmechanismen wie das Grenzregister, supervisor call, verschiedene CPU-Modi und der Zeitgeber.

2. Zusammenfassung

Eine der wichtigsten Leistungen eines Rechners besteht in der permanenten Speicherung von Daten. Hierzu werden Speichergeräte benutzt. Bei den heute verfügbaren Technologien dominieren Magnetplatten, Disketten und optische Platten als Direktzugriffspeicher und verschiedene Sorten von Bändern als Back-up- und Archivierungsmedium. Wir haben die Gemeinsamkeiten und Unterschiede zwischen verschiedenen Arten von Speichergeräten und die sich daraus für die Verwaltung der Geräte und Datenträger ergebenden Aufgaben kennengelernt. Aus Sicht der Benutzer sind Dateien die verfügbaren Speichermedien in einem Rechner. Die Dateien in einem Dateisystem sind üblicherweise in hierarchischen Verzeichnissen organisiert. Benutzerprogramme greifen auf eine Datei über die Operationen einer Zugriffsmethode zu. Zugriffsmethoden realisieren entweder Seiten, Zeichen oder Sätze als Speichereinheiten und realisieren eine Zugriffsstruktur wie die sequentielle oder index-sequentielle. Dateien sind virtuelle Geräte; generell werden oberhalb der Betriebssystemebene nur noch virtuelle Geräte über die Systemaufrufe zur Verfügung gestellt; das Betriebssystem selbst arbeitet mit realen Geräten. Wir haben diverse Unterschiede zwischen realen und virtuellen Geräten kennengelernt. Die Hauptaufgaben bei der Realisierung von Dateisystemen auf Platten sind die Verwaltung aller organisatorischen Angaben zu den Dateien (in Dateiattributen) sowie die Zuordnung von Sektoren zu Dateien und die Verwaltung der freien Sektoren einer Platte. Hierfür haben wir drei Techniken kennengelernt (FAT, Sektoradresstabellen, Sektorfolgen). Weiterhin haben wir diverse Maßnahmen kennengelernt, durch die der Zugriff auf die Sektoren einer Datei beschleunigt wird. Die E/A-Software ist in Schichten gegliedert. Man unterscheidet zwischen geräteabhängigen Teilen (Treiber) und geräteunabhängigen Teilen. Bei den Zugriffsmethoden unterscheidet man zusätzlich Schichten, in denen Seiten und Zeichen bzw. Sätze als Speichereinheiten realisiert werden. Beim Entwurf der E/A-Software müssen einige Prinzipien und Entwurfsziele beachtet werden. Wichigstes Ziel ist die Geräteunabhängigkeit von Benutzerprogrammen. Weitere Ziele sind einheitliche Namensräume für alle (virtuellen) Geräte, Kodierungsunabhängigkeit, Behandlung von parallelen Zugriffen und möglichst frühe Fehlerbehebung.

3. Zusammenfassung

Nach dem Durcharbeiten dieser Kurseinheit sollten Sie:

• den Unterschied zwischen Programm und Prozess erklären können.

• die Prozesszustände und die möglichen Zustandsübergänge mitsamt ihren Ursachen verstanden haben.

• den Zweck und Inhalt eines PCBs beschreiben können.

• die Aufgaben des Dispatchers und des Schedulers angeben können.

• die vorgestellten Scheduling Strategien verstanden haben und die Arbeitsweise an Beispielen demonstrieren können.

• den Unterschied zwischen Threads und Prozessen kennen und jeweilige Anwendungsgebiete dieser Modelle angeben können.

4. Zusammenfassung

Wir haben uns in dieser Kurseinheit mit der zentralen Frage befasst, wie der vorhandene physische Hauptspeicher eines Rechners zu verwalten ist. Wesentlichstes Ziel dabei war, für parallele Prozesse mehrere logische Hauptspeicher zu simulieren. Wir haben eine ganze Reihe von Speicherzuweisungsverfahren kennengelernt, die sich hinsichtlich ihrer Leistungsfähigkeit wesentlich unterscheiden; aus Performance- Gründen müssen allerdings bei den leistungsfähigeren Verfahren zentrale Funktionen in der Hardware realisiert sein. Die Aufteilung und Verwaltung des physischen Hauptspeichers muss gemeinsam mit dem Speicherschutz gesehen werden; die konkreten Mechanismen behandeln fast immer beide Problembereiche gemeinsam. Tendenziell bieten daher leistungsfähigere Verwaltungsverfahren auch besseren Schutz. Bei der Diskussion dieses Themas muss strikt getrennt werden zwischen den verschiedenen Formen, in denen ein Programm auftritt (Modul in einer höheren Programmiersprache bzw. Assembler, Bindemodul, Lademodul, geladenes Programm), den darin auftretenden Namen bzw. Adressen von Objekten und der unterschiedlichen Interpretation von Adressen im System- bzw. Benutzermodus (physische bzw. reale und logische Adressen). In diesem Zusammenhang spielt es außerdem eine Rolle, ob Programme statisch oder dynamisch gebunden werden, ob sie verschiebbar und ob sie gemeinsam benutzbar sind. Man kann die Speicherzuweisungsverfahren grob danach klassifizieren, ob sie einem Prozess einen einzigen zusammenhängenden Abschnitt des physischen Hauptspeichers zuweisen oder mehrere solche Abschnitte; bei der zweiten Gruppe von Verfahren muss, da der logische Hauptspeicher fast immer ein geschlossenes Intervall von Adressen hat, ein mehr oder weniger komplizierter Hardware-Mechanismus vorhanden sein, der logische in physische Adressen umsetzt. Bei der zusammenhängenden Speicherzuweisung wird i.A. der Rest des physischen Hauptspeichers, der nicht für den Betriebssystemkern benötigt wird, in ein oder mehrere Segmente aufgeteilt, die jeweils als Ganze einzelnen Prozessen zugewiesen werden. Man unterscheidet hier weiter Verfahren, die den physischen Hauptspeicher in feste bzw. variabel große Segmente aufteilen (MFT bzw. MVT). MVT ist wesentlich flexibler; als wichtigstes Detailproblem tritt hier die Verwaltung der variabel großen Segmente auf. Hierzu gibt es mehrere Verfahren (u. a. first fit, best fit, buddy). Die Verfahren, die den physischen Hauptspeicher nichtzusammenhängend zuweisen, können wiederum in zwei Gruppen klassifiziert werden: bei der ersten Gruppe werden nur wenige, variabel große Segmente zugewiesen, die der Applikation in Form von logischen Segmenten mehr oder weniger sichtbar sind. Bei der zweiten Gruppe wird der physische Hauptspeicher für Applikationen praktisch unsichtbar in viele Stücke gleicher Größe aufgeteilt, nämlich Seitenrahmen, der logische Hauptspeicher analog in Seiten. Die seitenorientierte Hauptspeicherverwaltung ist die Basis für virtuelle Hauptspeicher: bei diesen ist der logische Hauptspeicher eines Prozesses nur teilweise im physischen Hauptspeicher realisiert. Bei einem Zugriff zu einer nicht realisierten Seite tritt ein Seitenfehler ein und die fehlende Seite wird eingelagert (demand paging). Seitenfehler verursachen erheblichen Aufwand und dürfen nicht zu häufig werden. Aus diesem

Grund müssen einem Prozess zumindest so viele Seitenrahmen zugewiesen werden, dass alle Seiten der Lokalität, in der sich der Prozess befindet, realisiert sind. Diese Menge wird mit Hilfe der Arbeitsmenge approximiert, die wiederum mit Zugriffsbits annähernd bestimmt wird. Alternativ kann man die Seitenfehlerrate eines Prozesses überwachen und anhand dieser die Zahl der zugewiesenen Seitenrahmen bestimmen. Insgesamt dürfen nur so viele Prozesse parallel ausgeführt werden (also in den Zustand bereit überführt werden und Zeitscheiben zugeteilt bekommen), dass die Seitenfehlerrate ingesamt gering bleibt. Bei der Auslagerung von Seiten wählt man am besten die LRU-Strategie; diese lässt sich jedoch aus Aufwandsgründen nur approximativ z.B. mit Hilfe von Zugriffsbits realisieren.

5. Zusammenfassung

Nach dem Durcharbeiten dieser Kurseinheit sollten Sie in der Lage sein:

• die Notwendigkeit von Prozess-Kommunikation als zentrale Aufgabe von Betriebssystemen anhand praktischer Beispiele zu erläutern

• die bei konkurrenten Prozessen auftretenden Fragen differenziert aufzulisten

• disjunkte und überlappende Prozesse hinsichtlich ihrer Konsequenzen zu unterscheiden

• kritische Abschnitte in konkurrenten Prozessen von unkritischen Abschnitten abzugrenzen und das Problem des gegenseitigen Ausschlusses zu formulieren

• Aufgaben der Synchronisation mit Hilfe expliziter Synchronisationsvariablen zu lösen

• Semaphore von expliziten Synchronisationsvariablen zu unterscheiden

• das Erzeuger-Verbraucher-Problem algorithmisch zu beschreiben

• Kommunikations-Probleme durch Nachrichtenaustausch zu formulieren

• das Prinzip der Synchronisation mittels Monitore anschaulich zu erläutern

• die vier Deadlock-Bedingungen anhand von Beispielen zu demonstrieren

• Verfahren zur Deadlock-Erkennung von solchen zur Deadlock-Vermeidung und Deadlock-Verhinderung zu unterscheiden.

6. Zusammenfassung

Ziel dieser Kurseinheit ist, Sicherheitsfunktionen und -Mechanismen in Betriebssystemen vorzustellen. Um deren Motivation und Wirksamkeit diskutieren zu können, haben wir hier vorbereitend das Sicherheitsproblem ganz allgemein analysiert. Fassen wir noch einmal die wichtigsten Punkte dieser allgemeinen Analyse zusammen:

1. Das Sicherheitsproblem kann nahezu beliebig weit gefasst werden. In dieser Kurseinheit gehen wir nur auf solche Bedrohungen ein, die eine vom Betreiber oder Benutzer des Rechners unerlaubte oder ungewollte Benutzung eines Rechners darstellen (Sicherheit im engeren Sinn).
2. Das Sicherheitsproblem darf nicht auf den technischen Aspekt, also auf im Rechner realisierte Sicherheitsmaßnahmen, reduziert werden. Eine wirksame Sicherheitsstrategie muss neben technischen Maßnahmen auch Maßnahmen im organisatorischen oder psychologischen Bereich umfassen. Gegen viele Bedrohungen schützen Maßnahmen in den nichttechnischen Bereichen besser als technische Maßnahmen; in solchen Fällen ist es fraglich, ob in einem Rechner überhaupt Sicherheitsfunktionen gegen solche Bedrohungen realisiert werden sollten.

3. Sicherheitsfunktionen können nicht nur im Betriebssystem, sondern u.U. besser in darüberliegenden Schichten realisiert werden. Das Betriebssystem muss deshalb nicht alle denkbaren Sicherheitsfunktionen selbst enthalten, sondern die sichere Implementierung von spezielleren Sicherheitsfunktionen auf den höheren Ebenen ermöglichen.
4. Umgekehrt benötigt das Betriebssystem die Unterstützung der unter ihm liegenden Schicht, also der Hardware, um die von ihm angebotenen Sicherheitsfunktionen sicher implementieren zu können.

6. Zusammenfassung

Diese Kurseinheit behandelte den Problemkreis Sicherheit. Zunächst stellten wir fest, dass Sicherheit nicht isoliert auf den Rechner oder das Betriebssystem behandelt werden kann, sondern dass das gesamte organisatorische Umfeld mit einbezogen werden muss. Es ist daher zu trennen zwischen einer organisatorischen Sicherheitsstrategie und dem Anteil von dieser, der automatisiert (speziell durch das Betriebssystem) realisiert werden kann. Ferner sahen wir, dass die verschiedenen Schichten eines Rechnersystems (Hardware, Betriebssystem, ggf. Datenbanksystem, Applikation) bzgl. der Realisierung der automatisierten Sicherheitsstrategie zusammenwirken müssen. Vor allem müssen tiefere Schichten den darüberliegenden ausreichende Mittel, also Sicherheitsfunktionen, zur Verfügung stellen, damit diese speziellere Sicherheitsstrategien realisieren können. Die Sicherheitsfunktionen, die die höheren Schichten von einem Betriebssystem erwarten, variieren sehr stark abhängig von den Sicherheitszielen der konkreten Anwendung, der Art der benutzten Rechner und vielen anderen Faktoren. Eine gewisse Orientierung liefern seit neuem entwickelte Sicherheitsklassifikationen. Typischerweise erwartet man von einem Betriebssystem mehr oder weniger leistungsstarke Sicherheitsfunktionen in folgenden Bereichen: Identifikation, Authentisierung, Rechteverwaltung, Rechteprüfung, Beweissicherung, Speicherschutz und Datenübertragungssicherung. Basis oben aufgezählten Funktionen sind Speicherschutzverfahren, Schutz von Schnittstellen durch Betriebsmodi der CPU usw., die durch die Hardware unterstützt sind und die zunächst den Betriebssystemkern und die Anwendungen untereinander schützen. Ein grobes Raster, durch das die Aktionen eines Benutzers generell eingeschränkt werden können, sind Benutzerprofile. Im allgemeinen benötigt man jedoch feinkörnigere Kontrollen, bei denen für einzelne Subjekte und Objekte festgelegt werden kann, welche Zugriffe erlaubt sind. üblich sind vor allem diskretionäre Zugriffskontrollen, bei denen die Entscheidung über die Rechte an einem Objekt bei dem sog. Besitzer des Objekts liegen. Die wichtigsten Implementierungsformen sind Zugriffskontrollisten (vor allem für den persistenten Rechtezustand) und Profile bzw. Capability-Listen (vor allem für den transienten Rechtezustand). Diskretionäre Zugriffskontrollen können keine unerwünschten Informationsflüsse verhindern. Dies kann durch Informationsflusskontrollen verhindert werden. Die wichtigste Form von Informationsflusskontrollen sind solche auf Basis von Vertraulichkeits- und Integritätslabels von Subjekten und Objekten, bei denen unerwünschte Informationsflüsse infolge der *-Eigenschaft verhindert werden.

7. Zusammenfassung

In dieser Kurseinheit haben wir uns mit der Frage befasst, wie die Benutzer eines Rechners diesem Arbeitsaufträge übergeben können. Hierzu benötigen sie eine Kommandosprache. Der Kommandointerpreter benötigt dann seinerseits Mittel zum Starten von Prozessen. Das Starten eines Prozesses umfasst das Erzeugen eines Prozesskontrollblocks und eines logischen Hauptspeichers für den Prozess, das Laden des auszuführenden Programms in den Hauptspeicher, die übergabe von Parametern, das Einrichten von Umgebungsvariablen und das Vererben offener Dateien. Der Betriebssystemkern stellt hierzu Systemaufrufe zur Verfügung, durch die Prozesse erzeugt, gestartet, kontrolliert und beendet werden können. Es erweist sich als sehr vorteilhaft, wenn die gleichen Systemaufrufe auf die Standard-E/A-Geräte eines Prozesses und auf Dateien anwendbar sind: in diesem Fall können die Ein- bzw. Ausgaben eines Prozesses von den Standardgeräten auf Dateien umgelenkt werden. Die Kommandosprachen verschiedener Rechner sind sehr unterschiedlich, weil sie sich mehr oder weniger an die Funktionalität des Betriebssystemkerns und Merkmale der Standard-E/A-Geräte anpassen müssen. Dennoch gibt es einige allgemeine Anforderungen: Kommandosprachen sollten unabhängig vom

Betriebsmodus, benutzungsfreundlich und adaptierbar sein. Der Kommandointerpreter sollte nicht Teil des Betriebssystemkerns sein, sondern ein ganz normales Anwendungsprogramm. Ein einzelnes Kommando wird in mehreren Phasen verarbeitet: zunächst wird es bei Kommandoprozeduren aus einer Datei eingelesen und bei interaktiver Eingabe durch einen Editiervorgang erstellt; alternativ hierzu werden bei graphischen Benutzungsschnittstellen graphische Operationen mit auf dem Bildschirm dargestellten Objekten vorgenommen. Danach wird zunächst das Kommandoverb vorverarbeitet, wobei Aliase und/oder Abkürzungen ersetzt werden. Anschließend werden die Parameter in ähnlicher Weise vorverarbeitet. Besonders interessant ist die Möglichkeit, mit Mengen von Dateien zu arbeiten. Nach Abschluss der Vorverarbeitung wird schließlich das eigentliche Kommando entweder direkt im Kommandointerpreter, durch Ausführen einer Kommandoprozedur oder durch Starten eines Prozesses ausgeführt. Kommandosprachen enthalten neben den elementaren Kommandos auch noch Möglichkeiten zur Ablaufsteuerung und zum Umgang mit Variablen, die im Prinzip ähnlich wie in konventionellen Programmiersprachen, jedoch in vielen Details auf den besonderen Anwendungsbereich abgestimmt sind. Die Leistungsfähigkeit verschiedener Kommandosprachen variiert in dieser Hinsicht allerdings ganz erheblich. Besonders leistungsfähig ist das Konzept der Kommandosubstitution, durch das beliebige Textprozessoren oder in konventionellen Sprachen geschriebene Programme in Kommandoprozeduren eingebunden und dadurch sehr komplexe Systeme geschaffen werden können.

Ausarbeitung

Kurseinheit 1 – Einleitung

- Betriebssystem – Menge von Programmen, die es ermöglichen, einen Rechner zu betrieben, je Anforderung (Zweck des Rechners) hat das Betriebssystem unterschiedliche Aufgaben
- Schließen der Lücke zwischen Hardware und Software, Benutzer → Anwendungsprogramm → Betriebssystem → Hardware
- Aufgeben eines Betriebssystems: Gerätesteuerung, Schutz, Mehrprogrammbetrieb, Fehlerbehandlung, Prozesssynchronisation, Prozesskommunikation, Betriebsmittel- und Ressourcenverwaltung, Kommandosprache, Administration
- Bestandteile eines BS: Dienstprogramme, Bibliotheken, Betriebssystemkern
- Applikationsausführung: Quellprogramme durch Compiler in Bindemodul übersetzt, aus einem oder mehreren Bindemodulen und Standardbibliotheken erzeugt der Binder ein Lademodul, dieses wird vom Lader in den Speicher geladen – das Programm wird ausgeführt
- Ebenenmodell:
  digitale Logikebene (A/D-Wandler, Modem, Drucker, Tastatur, ...)
  Mikroprogrammebene (Gatter für elementare Funktionen)
  Konventionelle Maschinenebene (unterste Programmierebene z.B. für CPU)
  Betriebssystemebene (Systemaufrufe)
  Assemblersprache
  höhere Programmiersprache dazu kommen losgelöst Kommandosprachen
- Hauptspeicherverwaltung:
  BIOS (Basic Input Output System) enthält die hardwareabhängigen Teile des BS
  BDOS (Basic Disk Operating System) realisiert Diskzugriffe
  CCP (Console Command Prozessor)
  TPA (Transient Program Area)
  Interruptvektor
- Hardware löst Unterbrechungen aus → erhält CPU-Zeit
  Software löst Traps für Systemaufrufe aus
- Trap – Unterbrechung durch Programmfehler oder Supervisorcall (Maschinenbefehl)
- Systemmodus (privilegierte Befehle), Benutzermodus (eingeschränkte Befehle)
- parallele Prozesse durch virtuelle Prozessoren
- Laden des BS durch den Urlader (Betriebssystemkern) aus ROM → Ururlader
- Betriebsarten:

interaktiver Betrieb (Dialogbetrieb)

Stapelbetrieb (Batchjobs)

Hintergrundbetrieb (Batch + interaktiv)

Realzeitbetrieb

Teilhaberbetrieb (Dialogbetrieb, mehrere Benutzer von Terminals auf einen Prozess)

Kurseinheit 2 – Geräteverwaltung

- Primärspeicher – Sekundärspeicher, Kosten – Geschwindigkeit, Kapazität
  Register, Cache, Hauptspeicher, HD, CD, DVD, Magnetband
- Kommunikationsgeräte für Ein- und Ausgabe, Übertragungseinheit zwischen Hauptspeicher und Gerät ist ein Block = (512, 1024, 2048) Zeichen
- Controller – Gerätesteuereinheit, Kommunikation zwischen CPU, Hauptspeicher und Controller ermöglicht das Bussystem
- geräteunabhängige E/A-Software + geräteabhängige Treiber + Unterbrechungsverarbeitung
- Treiber im Hauptspeicher:

Benutzeradressraum	Benutzerprogramme
Kernadressraum	Betriebssystemkern + Plattentreiber, Druckertreiber, ...
Hardware	Plattencontroller, Druckercontroller
Geräte	Platten, Drucker, Maus

- asynchrone Systemaufrufe bei E/A-Operationen auf Maschinenebene (Operation wird nicht beendet)
- E/A-Auftragsliste für jedes Gerät + Gerätezustandstabelle
- Virtuelle Gräte: Dateien im Dateisystem, Spooling bei Druckern
- Unterschiede bei virtuellen Geräten:
  - wesentlich andere Funktionen, Spezifikationen
  - Operationen arbeiten synchron
  - es kann viel mehr virtuelle als reale Geräte geben
- Entwurfsziele für E/A-Software
  - Geräteunabhängigkeit von Applikationen
  - einheitliche Namen ermöglichen
  - Kodierungsunabhängigkeit
  - Fehlerbehebung (hardwarenah, tiefe Schichten)
  - wechselseitiger Ausschluss – exklusive E/A-Prozesse nacheinander – Deadlock

- Zugriff auf Festplatten geht über Scheiben, Zylinder, Spur, Sektor
  Nutzinhalt pro Sektor – Block (Präambel, Nutzdaten, Fehlerkorrekturfeld)
- Partitionen – Masterbootsektor, Partitionstabelle, Partition1, Pratition2, ... , PratitionX
- Pufferung im Hauptspeicher, Konflikte → Pufferung im Kernadressraum auch Double Buffering
- Pufferung im Controller → problemlose Datenübertragung per DMA
- Datentransport → Zugriffszeit besteht aus:
  - Suchzeit (Positionierungszeit) finden der Spur
  - Latenzzeit – Sektoren finden, abhängig von der Rotationsgeschwindigkeit
  - Übertragungszeit – Lesen und Schreiben der Daten, abhängig von der Schreibdichte der Daten und der Rotationsgeschwindigkeit
- Beschleunigung der Zugriffszeit:
  - Abarbeitung der Warteschlange durch SSTF (Shortest Seek time First) oder SCAN – wird vom Controller realisiert – Ohne das Betriebssystem
  - Interleavin – Interleavefaktor (Auslassen von Sektoren beim Schreiben gibt beim Lesen Zeit zur Datenübertragung und damit flüssiges Arbeiten – Kopf ist sonst über nächsten Sektor schon weg)
  - Pufferung – Sektoren im Voraus lesen - Lokalität
  - Zusammenhängende Vergabe von Sektoren beim Schreiben der Datei
- Dateisysteme:
- FAT- verkettete Liste in den Hauptspeicher geladen, Eintraf in Verzeichnis: Dateiname, Startsektor, in jedem Sektor steht der Folgesektor, im letzten Nil – es muss immer die gesamte Liste verarbeitet werden, um zu einem Sektor zu kommen
- Sektordressentabelle (i-nodes) – geht von vielen kleinen Dateien aus, bestimmte Sektoren können direkt angesprungen werden
- Beschleunigung des Zugriffs
  - E/A-Pufferung, asynchrones Schreiben (verwenden eines Puffers bei hohen Übertragungsgeschwindigkeit)
  - Optimierung von Sektorfolgen – freie Bereiche hinter Dateien, Reorganisation der Platten

Kurseinheit 3 – Prozessorverwaltung

- Programm – statisch, mehrere laufen parallel oder sequentiell
- Prozess – dynamischer Vorgang – ablaufendes Programm mit alles Registerinhalten, Variablen und Werten des Befehlszählers
- Bereiche im Hauptspeicher
  - Programmsegment (ausführbarer Code)

    - Stacksegment (lokale Variablen, ...)
    - Datensegment (zu bearbeitende Daten)
- Prozesszustände: erzeugt, bereit, rechnend, blockiert, beendet
- Erzeugung durch:
    - Dienst durch das BS
    - Systemaufruf durch einen existierenden Prozess
    - eine Benutzeranfrage
    - neuer Batchjob zum Starten
- Prozessbeendigung durch:
    - normale Beendigung
    - Fehler durch den Prozess selbst, Fehler den der Prozess festgestellt hat (Dateizugriff, ...)
    - Beendigung durch einen anderen Prozess
- Listen für jeden Prozesszustand
- non-preemptive Prozesse – Arbeiten bis sie blockieren oder beendet sind
- preemptive Prozesse – Betriebssystem kann dem laufenden Prozess den Prozessor entziehen
- Prozessorumschaltung durch den Dispatcher
    - Anhalten des rechnenden Prozesses
    - Sichern der Informationen in dessen Prozesskontrollblock
    - aus Kontrollblock den nächsten Prozess herstellen
    - Prozessor an den neuen Prozess übergeben
- Prozessorumschaltung bei
    - erwarteter E/A-Operation (rechnend - blockiert)
    - freiwilliges Abgeben oder die Zeitscheibe ist abgelaufen
    - Prozess ist fertig oder wird beendet
    - neuer Prozess wird erzeugt
    - Ereignis, auf das ein blockierter Prozess gewartet hat tritt ein (blockiert - bereit)
- Scheduling:
- Long-Term-Scheduler – wählt Prozesse aus, die in den Hauptspeicher geladen werden sollen
- Short-Term-Scheduler – entscheidet, welcher bereite Prozess bearbeitet wird
- Qualitätsmaßstäbe: Prozessorauslastung, Antwortzeit, Durchlaufzeit (wie lange braucht ein Prozess, wenn es konkurrierende gibt), Durchsatz (reine Rechenzeit), Fairness
    → alle Größen bis auf Fairness ändern ihre Werte
- Durchlaufzeit = Wartezeit (Zeit bis zur CPU-Zuteilung) + Bedienzeit (CPU-Burst – reine Rechenzeit)
- non-preemtives Scheduling (für Dialogbetrieb nicht geeignet):

- First Come First Serve (FCFS) + einfache Implementierung, fair, geringer Aufwand, - schlecht für Dialogbetrieb
- Shortest Job First (SJF) + minimale Antwortzeiten, gut für Dialogbetrieb, - schlecht für Batchbetrieb, schwierig Zeit zu ermitteln, die der Prozess braucht
- Priority Scheduling (Unterschiedliche Gewichtung von Prozessen, SJF – je kleiner die Bedienzeit desto höher ist die Priorität) + wichtige Aufgeben schnell erledigt, - statisch, nicht fair; offen, wie die Priorität vergeben wird

- preemtives Scheduling:
  - Round Robin (FCFS mit Zeitscheibe - Quantum) + fair, einfache Implementierung, geringer Verwaltungsaufwand, - lange Prozesse sind benachteiligt
  - Shortest Remainig Time First (SJF mit Quantum – nach Ablauf neue Aufteilung nach Restzeit)
  - Prioraty Scheduling (dynamische Vergabe der Zeitscheibe)
- kombinierte Strategien
  - Feedback Scheduling (erst hohe Prio und kurze Zeitscheibe, wird der Prozess nicht fertig, dann geringere Prio und längere Zeit) + Prozesse werden mehrmals Kategorisiert, größere Fairness, - hoher Verwaltungsaufwand
  - Multiple Queues (Klasseneinteilung mit fester Zeit pro Klasse)
  1. Systemprozesse (intern FCFS) 60%
  2. Dialogprozesse (intern Round Robin) 30%
  3. Hintergrundprozesse (Batch) (rechenzeitabhängiges Feedbackscheduling) 10%

  + gute Differenzierung zwischen den Prozessen, vielseitig, - hoher Verwaltungsaufwand; aufwendig, gute Einstellungen zu finden
- Auswahl der Strategie – mathematischer Ansatz (nach CPU-Burst, Durchsatz, Prozessorauslastung) – Interpretationsproblem – welche Zeit ist für den Dialogbetrieb tolerierbar → realer Test
- Prozessmodell:
  + allgemeine Verfügbarkeit
  - hoher Verwaltungsaufwand (Programmsegment, Codesegment, Datensegment)
  - aufwendige Kommunikation zwischen den Prozessen
  - Parallelität innerhalb von Prozessen ist nur aufwendig zu beschreiben
- leichtgewichtige Prozesse – Threads – haben eigenen Stack, Programmzähler und Registersatz aber gemeinsames Datensegment und Programmsegment
- Vorteile:
  - Erreichbarkeit (ein Thread blockiert, die anderen laufen weiter)
  - Teilen der gesamten Ressourcen (HD, Prozessor, ...)
  - mehr Effizienz (bei einem Wechsel der Threads nur Register tauschen)

- Anwendungsgebiete: Mehrprozessormaschinen, Gerätetreiber für langsame Geräte, verteilte Systeme
- Benutzer Threads – Thread-Library-Funktion → Erzeugen und Beenden unabhängig vom Kern
- Kernel-Threads – Kernel verwaltet Threads wie Prozesse

Kurseinheit 4 – Hauptspeicherverwaltung

- Systemmodus: physische Adresse entspricht dem physischen Hauptspeicher
- Benutzermodus: logische Adresse wird von der MMU in physische Adresse umgeformt (Basisregister)
- einfach zusammenhängende Speicherzuweisung
    - es befindet sich nur ein Programm im Arbeitsspeicher, Schutz des BS durch Grenzregister
    - im Mehrprogrammbetrieb muss das zweite Programm komplett ausgelagert sein (swapping)
- mehrfach zusammenhängende Speicherzuweisung
    - MFT – feste Segmente unterschiedlicher Größe, innere Fragmentierung, best-avaible-fit, best-fit-only (zugewiesenes Segment ist z.B. nur 50% größer wie benötigt)
    - MVT – variable Segmente nach Anforderung, externe Fragmentierung, First-fit, Next-fit, Best-fit, Worst-fit, Buddy – Verdichten, wenn nicht genügend große Segmente vorhanden sind
- nichtzusammenhängende Speicherzuweisung
    - Trennung von Programmen und Daten im Speicher (Minimierung der Fragmentierung)
    - keine Adressierung sondern logische Segmente analog zu physischen Segmenten
- Paging – Seitenorientierte Speicherzuweisung
    - logische Adresse besteht aus Seitennummer und Offset
    - logischer Hauptspeicher → Seitentabelle (Seitennummer) → physischer Hauptspeicher (Seitenrahmennummer)
    - swapping – nur so viele Seiten auslagern, wie benötigt
    - Schutz durch Protection-Bit
    - Beschleunigung durch Liste der im Speicher vorhandenen Seiten
- Virtueller Hauptspeicher – demand Paging
- Ein- und Auslagern von Seiten → logische Adressen müssen größer sein als physische (swap-Bereich)
- dirty Bits – nur geänderte Seiten müssen zurück geschrieben werden
- Lokalität des Zugriffsverhalten
- Seitenauslagerungsstrategien → Minimierung der Seitenfehlerrate
- LRU (Last Recently Used) – die am längsten nicht benutzen Seiten werden ausgelagert (Zugriffs-(R)-Bit)

  Seitentabelleneinträge: R-Bit (Referenzbit), Dirty-Bit, Protected-Bit (Zugriffsregeln), Present-Bit
- FIFO (First In First Out) – spezial: Second-Chance mit benutztem R-Bit

- Clock-Algorithmus (Secon-Chance als Kreis) – wird ein Seite zugegriffen, wird das R-Bit auf 1 gesetzt, wird ein auszulagernde Seite gesucht, werden alle R-Bits von 1 auf 0 gesetzt, bis eine Seite mit einer 0 gefunden wird, diese wird ausgelagert

Kurseinheit 5 – Prozesskommunikation

- physische Betriebsmittel – Hardware, logische Betriebsmittel – simuliert durch Software
- konkurrierende Betriebsmittel benutzen gemeinsam die gleichen Betriebsmittel aus Kostengründen, wegen gemeinsamer Programmteile, gemeinsamer Daten oder Redundanz
- determiniert – gleiche Programme – gleiche Ergebnisse
- nicht determiniert – unabhängige Reaktion auf Ereignisse
- disjunkte Prozesse – voneinander unabhängige Prozesse
- überlappende Prozesse – zeitkritisch → unkritische Abschnitte – kritische Abschnitte
- shareable (gemeinsam nutzbare) Betriebsmittel (CPU, Dateien lesend)
- nonshareable Betriebsmittel → periphere Geräte, veränderliche Daten ← gegenseitiger Ausschluss

-Synchronisationsvariablen

- Prozesse mit verzahnten Schleifen und gemeinsamen Variablen
- Deadlock, wenn ein Prozess beendet ist und der andere auf eine Reaktion wartet
- besser jeder Prozess hat eigene Variablen, der andere Prozess prüft nur
- aktives Warten auf Ereignis

-Semaphore

- kein aktives Waten (nicht ständig Fragen, ob kritischer Abschnitt frei ist)
- Prozess deaktivieren und wieder aufwecken
- Semaphore sind Integer-Variablen mit einer Initialisierung größer Null (normalerweise eins), wird eine Ressource benutzt, wird eine down-Operation durchgeführt, der Wert wird um ein reduziert, prüft ein Prozess die Variable und sie hat den Wert Null blockiert der Prozess, gibt ein Prozess eine Ressource wieder frei, wird durch eine up-Operation die Ressource wieder frei gegeben

- Nachrichtenaustausch
  - Synchronisation bei Synchronisationsvariablen und Semaphore durch gemeinsame Daten
  - z.B. Sender → Nachricht → Ringpuffer → Empfänger
  - normalerweise Einwegkommunikation, auch Zweiwegkommunikation mit Betätigung
- Monitore
  - Betriebsmittel (Prozeduren, Variablen, ...) für mehrere Prozesse zugänglich
  - nur ein Prozess kann im Monitor sein → hat die Rechte
  - Warteschlangen für den Eintritt in den Monitor

- Deadlock – Systemverklemmungen
    - Prozesse verklemmen sich gegenseitig
    - ein Prozess besitzt ein Betriebsmittel und fordert ein anderes an, das ein weiterer Prozess bereits besitzt, der wiederum auf das Betriebsmittel des ersten Prozesses wartet
- Bedingungen für Deadlocks:
    - Wechselseitiger Ausschluss (exklusiver Zugriff)
    - Nichtunterbrechbarkeit (no preemtion)
    - Hold-And-Wait exkusives Belegen, während Prozess auf ein Betriebsmittel weiteres wartet
    - zyklische Wartebedingung – geschlossene Kette von belegten bzw. angeforderten Mitteln
- Beseitigung von Deadlocks:
    - Abbruch aller verklemmten Prozesse
    - Schrittweiser Abbruch, bis Deadlock sich löst
    - Prozesse schreiben Checkpoints – zurücksetzen bis zum letzten Checkpoint
    - Schrittweisen zurücksetzten bis zum letzten Checkpoint, bis Deadlock sich löst
- Vermeidung von Deadlocks:
    - Banker Algorithmus – es gibt eine Matrix von verfügbaren Ressourcen, ein Prozess muss anmelden, wie viel Ressourcen er maximal benötigt, Beispiel: es gibt 10 Einheiten einer Ressource, Prozess 1 benötigt max 6 und besitzt schon 4, Prozess 2 benötigt max 8 und besitzt 3 – er kann jetzt nur eine weitere Ressource erhalten, damit für Prozess 1 zwei Einheiten offen bleiben und er nicht blockiert

Kurseinheit 6 – Sicherheit

- Schäden:
    - Verlust der Vertraulichkeit
    - Verlust der Datenintegrität
    - Verlust der Systemverfügbarkeit
    - Finanzieller Verlust durch Ausfall
    - Missbrauch der Hardware/Software
- Bedrohungen:
    - unzulässige Aktionen von zugelassenen Benutzern
    - Fehlverhalten von Programmen
    - Materialfehler (z.B. bei Festplatten)
    - Fehler bei der Datenübertragung
    - fehlerhafte Eingabe von Daten

- Eindringlinge – Hacker, Würmer, ...

- organisatorische Sicherheitsstrategie
- automatisierte Sicherheitsstrategie (besteht aus Schichten – Schutz auf den Ebenen (Hardware, Betriebssystem, Applikation (DBMS)))
- Subjekte – aktive Instanzen, die Zugriffe haben
- Objekte – passive Instanzen, - zu schützende Einheit
- Sicherheitsanforderungen an BS:
  - Aufwand für Implementierung – Preis
  - Laufzeitaufwand
  - Administrationsaufwand
- Funktionsbereiche:
  - Identifikation und Authentisierung
  - Zugriffskontrollen
  - Beweissicherung, Audit
  - Gewährleistung der Funktionsfähigkeit
- Benutzerverwaltung – Liste von erlaubten Benutzern
- Identifikation – Benutzer muss sich als registrierter Benutzer ausweisen
- Authentisierung – muss nachweisen, dass er die angegebene Person ist
- Verwendung von Benutzergruppen (Zusammenfassen von Benutzern mit gleichen Rechten)
- Programme als Subjekte - SETUID-Bit zur Verwaltung, der Prozess erhält die Rechte des Besitzers des Programms (die höher sind, als die Rechte des Benutzers der das Programm ausführt – Passwort ändern)
- Benutzerprofile:
  - Verbot von Ausübung sicherheitsrelevanter Funktionen
  - Ort, Zeit der Benutzung lässt sich beschränken
  - Verbrauch von Rechenzeit, HD-Platz, Papier kann dokumentiert und limitiert werden
- Zugriffskontrollen – Kein Verlust der Vertraulichkeit, keine unzulässige Benutzung von Ressourcen
- persistenter Rechtezustand – Rechte unabhängig von Prozessen, den Rechnerzustand überdauernde Rechtezuweisung
- Benutzerverwaltung → persistente Objekte
- transisenter Rechtezustand – nach Rechteprüfung erteilter Zugriff für eine Prozess, Verwaltung im Dateikontrollblock
- transientes Subjekt – Berechtigung wird mit dem Beenden der Sitzung gelöscht (nicht persistent)
- Zugriffskontrollmodelle: diskretionär, Informationsflusskontrollen

- diskretionär – Zugriff auf Objekte auf Basis der Identität von Subjekten – Problem: ein Benutzer kann durch kopieren den Inhalt einer Datei einer Person zur Verfügung stellen, die keine Berechtigungen für diese Datei hat
- man unterscheidet offene Systeme mit explizitem Verbieten und geschlossene Systeme mit explizitem Erlauben
- Berechtigungen werden durch UID, GID-Paar gesteuert
- Modi: read, write, append, delete, execute, navigate, owner
- Zugriff nach S x O x M →W
    - S – aktuell vorhandene Menge der Subjekte
    - O – aktuell vorhandene Menge der Objekte
    - M – Menge der Modi (statisch)
    - W – Werte von Rechtefestlegungen (erlaubt, verboten)
- Zugriffskontrollmatrix →
- Objekte in Spalten – granulatorientiert (Zugriffskontrollisten, Schutzbits) – eignen sich zur Implementierung des persistenten Rechtezustands
- Subjete in Zeilen – subjektorientiert (Profile, Capabilities)) – eignen sich zur Implementierung des transienten Rechtezustands
- Zugriffskontrollisten (ACL) – einfach ersichtlich, wer welche Rechte für ein Objekt hat, schwierig zu ermitteln welche Rechte ein Subjekt hat
- Schutzbits – ACL mit drei Einträgen (Besitzer, Gruppe, übrige Benutzer) mit je drei Bit (rwx)
- Profile – Gegenstück zu ACL
- Capabilities – äquivalent zu Profileintrag, Capabiliti-Ticket ermöglicht Zugriff auf ein Objekt, Tickets verteilen die Rechte, sie sind im System verteilt und müssen vor Manipulation geschützt werden – z.B. im Dateikontrollblock – wird beim Öffnen einer Datei angelegt, ist nicht persistent
- Informationsflusskontrollen – disktetionäre Zugriffskontrollen erlauben keine Kontrolle des Informationsfluss → disktetionäre Zugriffskontrollen durch Informationsflusskontrollen ergänzen
- Modelle auf Basis von Sicherheitsklassen
- Vertraulichkeitsklassen: offen, vertraulich, geheim, streng geheim
- Zuordnung zu den Klassen durch Labels
- ein Prozess darf Objekte gleicher und niedriger Klassen lesen und Objekte gleicher und höherer Klassen schreiben, ein Prozess darf die Vertraulichkeitsklasse eines Objekts nicht ändern

Kurseinheit 7 – Kommandosprachen

- Teil des Betriebssystems
- zu startender Prozess besteht aus Prozess im Hauptspeicher, Parametersatz wird übergeben, Standardausgabe- und Eingabegeräte werden festgelegt, Prozess läuft in einer Umgebung ab
- Elternprozesse erzeugen Kindprozesse mit eigener ID
- ein Prozess wird von außen durch einen Kill abgebrochen oder nach seiner Beendigung durch ein exit
- Umgebung für einen Kindprozess: beim Start übergebene Parameterwerte, Umgebungsvariablen prozessspezifisch ←→ global , offene Dateien – Dateikontrollblock (Rechte: lesen, schreiben, ...), E/A-Geräte – Umlenkung
- Kommandosprachen: <Kommandoverb> <Parameterliste>
- Umsetzung: Aliasauflösung, Wildcardexpansion, Ein/Ausgabeumlenkung
- Wildcarts: * (beliebiges Ende/Anfang), ? (beliebiges Zeichen), [xX] Liste von Möglichkeiten für Zeichen