Zusammenfassung: Jahr 1

Inhaltsverzeichnis

1 Lernfeld 4 - Einfache IT-Systeme (Oening und Wächter)

1.1 Interrupts

Beispielsweise einmal pro Sekunde wird nach einem Interrupt geschaut. Es gibt aber auch Algorithmen, die nach jedem Befehl nach einem Interrupt.

Wenn ein Interrupt durchkommt, wird der Interrupt mit einer Bibliothek abgeglichen,
Interrupts: hardware-, softwarebedingte und direct memory access (DMA).
Ursachen für Interrupts

- Fehlersituation: Fehler bei Rechenoperationen (bspw. Div. durch Null)
- Software-Interrupt: bspw. *kill*
- Hardware-Interrupt: Periperieeinheiten meldet über ein Signal ein Ereignis an die Software.

Interruptvektor
Systemstapel Unterbrechungsroutine Benutzerprogramm
Programmzähler Register Stapelzeiger
Maskierung: Bit-Maske, bspw. Netzwerkmasken (255.255.255.0 oder Präfixe /24)
Unterbrechungskontext

1.2 Prozessmanagment (Scheduling)

Central Processing Units (CPUs) können trotz Hyperthreading und mehreren Kernen trotzdem nur einen Befehl gleichzeitig ausführen. Damit gleichzeitig mehrere Programme laufen können, muss festgelegt werden, welches Programm wann wie viel Prozessor-Zeit beanspruchen darf. Der Teil des Betriebssystem, der diese Zuteilung kontrolliert, wird Schedular genannt. Wie genau der Schedular die Zeit verteilt, legt der sogenannte Scheduling Algorithmus fest.

Ein Prozess ist ein Programm in Ausführung und besteht aus Programmdaten und dem Prozesskontext. Prozesse müssen in verschiedenen Situationen auf bestimmte Ereignisse warten, bspw. das Laden von Daten. In dieser Wartezeit werden dann andere Prozesse berechnet.

Im RAM liegen verschiedene Daten der Prozesse. Im sogenannten Programmsegment liegen der ausführbare Code des Programms. Im Datensegment liegen die Daten, die ein Prozess bearbeitet.

Alle Daten, die das Betriebssystem über einen Prozess verwalten muss,bezeichnet man als Prozesskontrollblock (PCB), und alle PCBs zusammen werden in einer Prozesstabelle organisiert. Im Prozesskontrollblock stehen insbesondere folgende Informationen:

- Prozessidentifikation: Jeder Prozess erhält eine ID zur eindeutigen Identifizierung
- Prozessorstatus: Enthält die Informationen über den Zustand des Programms, in dem es fortgesetzt werden soll
- Prozeskontrollinformationen: Priorität des Prozesses, Informationen über den Speicherbereich, geöffnete Dateien des Prozesses ...

1.2.1 Scheduling-Algorithmen: Varianten

Prinzipiell gibt es zwei Arten: (1) **Nicht preemptives Scheduling** und (2) **preemtives Scheduling**. Ersteres ist nicht interaktiv und wird bei Stapelverarbeitungssystemen benutzt. Befindet sich ein Prozess im Zustand „running“, so wird seine Ausführung solange fortgesetzt, bis er terminiert oder durch Warten auf Ressourcen blockiert. Letzter lassen die Unterbrechung von Prozessen zu, sodass eine Interaktion möglich wird. Dadurch geht der Prozess in den Zustand „ready“ über.

Die Anforderungen an einen Scheduling-Alorithmus hängen stark davon ab, welche Aufgaben ein System zu erfüllen hat. Je nach Aufgabe lässt sich so der Algorithmus optimieren.

Zu den Anforderungen an alle Algorithmen gehören die Faktoren **Fairness**, **Policy Enforcement**, **Balance**, **Datensicherheit**, **Skalierbarkeit** und **Effizienz**. Das bedeutet beispielsweise, dass jeder Prozess einen gerechten Anteil an der Prozessor-Zeit erhält und auch, dass jeder Prozess eine endliche Zeit warten muss. Weitere Anforderungen werden in nutzer- und systemorientierte Kriterien geteilt. Nutzerorientierte Kriterien spielen vor allem in interaktiven Systemen eine Rolle, wohingegen systemorientierte Kriterien in Echtzeitsystemem[1] im Vordergrund stehen.

- Stapelverarbeitungssysteme: Durchsatz soll maximiert werden, Minimierung der Verweildauer, Konstante/maximale CPU-Auslastung
- Interaktive Systeme: Antwortzeit sollte minimal sein, Verhältnismäßigkeit der Antwortzeiten, Anzahl der Interaktionen sollte maximal sein können
- Echtzeitsysteme: Sollzeitpunkt, Vorhersagbarkeit

1.2.2 Nicht-preemptive Scheduling-Algorithmen: Stapelverarbeitungssysteme

Stapelverarbeitungssysteme sind etwa alte Lochkarten-Maschinen. Dabei ist im Gegensatz zu interaktiven Systemen nach dem Starten des Prozess kein Eingreifen durch den Nutzer vorgesehen.

First-Come-First-Serve (FCFS) ist der einfachste aller Scheduling-Algorithmen, jedoch sind keine Unterbrechungen vorgesehen. D.h., ein Prozess läuft solange, bis er fertig ist. Dazwischen kann kein anderer Prozess auf die CPU zurückgreifen.

Der Algorithmus **Shortest-Job-First** (SJF) geht davon aus, dass die Laufzeit der Prozesse vorher bekannt ist. SJF wählt immer den Prozess zuerst, der am kürzesten ist.

Shortest Remaining Time Next (STRN) wählt, wie der Name bereits zeigt, den Prozess, der die niedrigste verbleibende Laufzeit besitzt.

1.2.3 Preemptive Scheduling-Algorithmen: interaktive Systeme

Round-Robin Scheduling (RRS) ist der älteste und einfachste Algorithmus für interaktive Systeme. Dabei erhält jeder Prozess abwechselnd einen gleich großen Zeitabschnitt (sog. Quantum). Das Verfahren wird auch Zeitscheibenverfahren oder Time-Sharing genannt.

Prioritätsbasiertes Scheduling. Wie wir bereits wissen, können Prozessen eine Priorität zugeordnet werden. Unter Linux wird die Priorität von dem sogenannten Nice-Wert repräsentiert. Beim RRS wird hingegen angenommen, dass alle Prozesse gleich wichtig sind.

Damit ein Prozess mit niedriger Priorität nicht immer wieder aufgeschoben wird, kann der Algorithmus beispielsweise bei jedem Takt den Wert die Priorität erhöhen.

Multilevel Feedback Queueing (MLFQ): Das besondere dieses Algorithmus ist - im Gegensatz zu SPN und SRPT - dass man keine Kenntnisse über die voraussichtliche Abarbeitungszeit braucht und trotzdem kurze Aufträge bevorzugen kann. Das Scheduling wird nun unterbrechend ausgeführt; Prioritäten werden dabei dynamisch vergeben.

1.3 Lizenzen

1.3.1 Open Source

Open Source nennt man Software, deren Lizenzbestimmungen in Bezug auf die Weitergabe der Software besagen, dass der Quelltext *öffentlich* zugänglich sein muss. Abhängig von der jeweiligen Lizenz kann diese unter Umständen auch beinhalten, dass die Software frei kopiert, modifiziert und verändert wie unverändert weiterverbreitet werden darf.

[1] In Echtzeitsystemen muss gewährleistet werden, dass ein bestimmter Prozess zu einer bestimmten Zeit berechnet wurde. Bei Nicht-Echtzeitsystemen ist nicht vorhersehbar, wann ein bestimmter Prozess abgeschlossen sein wird. Daher werden Echtzeitsysteme vor allem in kritischen Umgebungen verwendet, bspw. in embedded-systems

Open-Source-Software (OSS) steht unter einer der von der Open Source Initiative (OSI) anerkannten Lizenzen. Die OSI verwendet dabei den Begriff Open Source auf all die Software an, deren Lizenzverträge den folgenden drei Merkmalen entsprechen und die zehn Punkte der Open Source Definition erfüllen:

1. Die Software (d. h. der Quelltext) liegt in einer für den Menschen lesbaren und verständlichen Form vor.
2. Die Software darf beliebig kopiert, verbreitet und genutzt werden.
3. Die Software darf verändert und in der veränderten Form weitergegeben werden.

Ein frühes und bekanntes Beispiel für OSS ist Mozillas Firefox. Dieser entstand 2002 aus der Freigabe des Quelltextes des Netscape Navigators im Jahre 1998.

Zu beachten ist, dass Open Source Software im frei im Sinne von Freiheit (*free speech, not free beer*) ist und nicht im Sinne von kostenlos. Um Missverständnissen vorzubeugen wird sie daher *open* statt *free* genannt. Aber auch diese Regelung ist nicht unproblematisch. Beispielsweise kritisiert die Free Software Foundation (FSF) vor allem die Tatsache, dass der Begriff Open Source die Einsicht in den Quellcode einer Software hervorhebt, nicht aber die Freiheit, diesen Quellcode auch beliebig weiterzugeben oder zu verändern. Die PGP Corporation bezeichnet zwar PGP als Open Source, weil der Quellcode gegen eine Gebühr einsehbar ist, jedoch darf er weder verändert oder weitergegeben werden.

1.3.2 Freeware

Freeware bezeichnet im allgemeinen Sprachgebrauch Software, die vom Urheber zur kostenlosen Nutzung zur Verfügung gestellt wird. Freeware ist meistens proprietär und steht damit laut der Free Software Foundation im Gegensatz zu Freier Software, die weitläufigere Freiheiten, wie Veränderungen an der Software, gewährt.

1.3.3 Commercial Software

Commercial Software, oder auch Payware, ist Software, die für den Verkauf bestimmt ist. Kommerzielle Software kann sowohl properitär als auch unter free/open source Software sein.

1.4 Boot-Prozess

Beim Bootstrapping (Bootstrap-Prozess) wird der Bootloader (bspw. GRUB) aus dem Master Boot Record (MBR) gelesen und in den RAM geschrieben. Der MBR liegt im ersten Sektor der Festplatte/SSD. Der Bootloader lädt dann den Kernel in den RAM (s. Memory Mangament).

Der Ablauf ist wie folgt: BIOS -> POST -> INT -> BOOT -> HW Check -> Kernel

1.4.1 BIOS vs UEFI

Als Nachfolger des BIOS kommt langsam das UEFI durch. Dieses bietet viele Möglichkeiten, die auf System mit klassischem BIOS erst nach dem Laden des Betriebssystems möglich waren. Eine wichtige Neuerung ist der Secure-Boot Modus, mit dem nur vorher signierte Bootloader verwendet werden können. So wird das Risiko eine Infektion mit Schadsoftware über manipulierte Bootloader minimiert. Zusätzlich bietet UEFI die Möglichkeit einer Netzwerkverbindung ohne geladenem Betriebssystem. So kann zum Beipiel Fernwartung schon im Bootprozess ansetzen. Das UEFI ist durch die Nutzung der Grafikfähigkeit aktueller Hardware viel einfacher zu Bedienen als das klassische BIOS: Das UEFI ist außerdem dazu in der Lage, dass Treiber bereits darin eingearbeitet werden und so systemunabhängig verwendet werden. In Zukunft lassen sich auch speziell für das UEFI geschriebene Anwendungen schon

vor dem Laden des Betriebssystems nutzen. So könnten viele einfache Anwendungen schon bald ohne zeitaufwändigen Bootvorgang verwendet werden.

Die Kritik an UEFI gründet in erster Linie auf der Komplexität, da UEFI fast schon ein eigenes Betriebssystem darstellt.

1.4.2 Power On Self-Test (POST)

Der Power-On-Self-Test (POST) ist der erste Schritt des Bootvorgangs. In diesem Schritt führt der Prozessor die ersten Programmanweisungen des BIOS aus. Diese beinhalten einen Test, der überprüft, ob die notwendigen Geräte angeschlossen und auch funktionsfähig sind. Teilweise werden auch schon Informationen ausgewertet, welche im CMOS-Speicher auf der Hauptplatine abgelegt sind. In dieser Phase können nur hardwarebedingte Fehler auftreten, da das Betriebssystem erst später geladen wird. Diese wären z.B. ein defekter Arbeitsspeicher oder falsch montierte Verbindungskabel. Ansonsten können aus fehlerhafte Einstellungen im BIOS-Setup zu Fehlern führen. Dies kann z.B. durch einen Datenverlust im CMOS-Speicher vorkommen.

1.4.3 Initiale Startphase

Die Initiale Startphase ist die zweite Phase des Bootvorgangs und startet, nachdem der POST erfolgreich abgeschlossen wurde. Nun werden die Laufwerke nach der im BIOS eingestellten Reihenfolge nach einem Betriebssystem gefunden. Ist dieses gefunden, so wird der Bootsektor des Betriebssystems in den RAM geladen.

Ein wichtiger Risikofaktor ist dabei jedoch, dass der in diesem Bootsektor enthaltene Programmcode unabhängig vom Inhalt geladen und ausgeführt wird. So können an dieser Stelle Bootsektorviren ansetzten, die den restlichen Virencode dann danach nachladen können. Wenn der Virus vom Bootmedium in den Arbeitsspeicher geladen wurde, kann er sich auch auf der Festplatte einnisten, um im weiteren Verlauf bei jedem Systemstart präsent zu sein.

Ist als Bootlaufwerk die primäre Festplatte des Computers gefunden, wir zuerst der Masterbootsektor (MBR) geladen. Der MBR enthält die Partitionstabelle, welche die Informationen über die Aufteilung der Festplatte in einzelne Partitionen enthält, und den MBR-Code, welcher umgehend ausgeführt wird. Aus der Partitionstabelle ermittelt der MBR-Code die aktive primäre Partition. Aus dieser wird der betriebssystemabhängige Bootsektorcode geladen und ausgeführt.

1.4.4 Bootloader-Phase

Zunächst wird die CPU aus dem Real-Modus in den Protected-Modus umgeschaltet, damit der gesamte Arbeitsspeicher genutzt werden kann. Der Bootloader enthält bereits die notwendigen Routinen, um Datenträger, die in NTFS, FAT 16 oder FT 32 formatiert sind, lesen bzw. zu beschreiben. Nach diesem Schritt wird die Datei BOOT.INI aus dem Startlaufwerk analysiert. Wenn nur ein Betriebssystem in dieser Datei verzeichnet ist, so wird umgehend die Hardware-Erkennungs-Phase eingeleitet. Sind jedoch mehrere Systeme verzeichnet, so erscheint ein Auswahlmenü, in dem der Benutzer die gewünschte Installation auswählen kann. Wird in einer vordefinierten Zeit keine Auswahl getroffen, so wird die in der BOOT.INI eingetragene Standardinstallation gestartet. Diese Einstellungen lassen sich auch in der BOOT.INI Datei verändern.

1.4.5 Hardware-Erkennungs- und Konfigurations-Phase

In der Hardware Erkennungs- und Konfigurations-Phase kommt die NTDETECT.COM zum Einsatz. Diese sammelt Informationen zu den folgenden Hardwaretypen und Geräten:

- System Firmware Informationen, u. a. auch Datum und Uhrzeit
- verfügbare Bus und Adaptertypen

- Grafikadapter
- Tastatur
- Kommunikationsschnittstellen (z.B. serielle Ports)
- Festplattenlaufwerke, CD-ROM-Laufwerke, etc.
- Diskettenlaufwerke
- weitere Eingabegeräte (wie z. B. Maus)
- Parallele Schnittstellen
- Geräte die über den ISA-Bus angeschlossen sind

Weiterhin wird von NTDETECT die Realisierung und Verwendung von Hardware-Profilen umgesetzt. Die von NTDETECT gesammelten Daten werden an den Bootloader zurückgegeben und werden an die danach aufgerufene Kernel-Lade-phase übergeben.

1.4.6 Kernel-Lade-Phase

In der Kernel-Lade-Phase wird zunächst der Windows Kernel geladen. Danach wird auch ein „hardware abstraction layerr (HAL) in den Speicher geladen. Welcher der verfügbaren HAL geladen wird, ist von der verwendeten Hardware abhängig. Der HAL stellt eine einheitliche Zugriffsschnittstelle zur Hardware zur Verfügung. Kernel und HAL initialisieren dann die Windows-Ausführungsschicht. Diese ist eine Reihe von Software-Komponenten, welche unter anderem Dienste und Gerätetreiber startet. Im nächsten Schritt startet der Kernel den Session-Manager. Dieser legt unter anderem die System-Umgebungsvariablen und startet den Kernel-Modus-Anteil der Windows-Benutzeroberfläche, welche für das Umschalten vom Text- in den Grafik-Modus sorgt. Danach wir der Logon-Manager gestartet. Außerdem werden noch ausstehende Installationen aus der vorangegangenen Windows-Sitzung zu Ende geführt.

1.4.7 Benutzer-Logon-Phase

Vom Windows-Logon-Manager wird zuerst das lokale Zugriffsschutzsystem (LSA) gestartet. Danach wird der Benutzeranmelde-Dialog durchgestellt. Die vom Benutzer eingegebenen Daten werden dann an das LSA durchgegeben und von diesem Überprüft. Sind diese Daten richtig, war die Anmeldung erfolgreich und sämtliche Autostart-Vorgänge und benutzerspezifischen Einstellungen werden durchgeführt. Die automatisch auszuführenden Prozesse können hierbei an mehreren Stellen innerhalb der Registry sowie innerhalb der Verzeichnisstruktur stehen.

1.4.8 Plug & Play - Geräteerkennung

Die Geräteerkennung wird direkt nach der Benutzeranmeldung gestartet. Dann werden asynchron zu den Autostart-Vorgängen die neuen Plug & Play - Geräte erkannt und eingerichtet. Hierbei kann es vorkommen, dass ein weiterer Neustart erforderlich ist.

1.4.9 Beschleunidung des Bootprozesses

Ein erster Schritt bei der Beschleunigung des Bootvorgangs ist die Einstellung der richtigen Reihenfolge der Bootlaufwerke. Wird das System ohnehin in der Regel von dem gleichen Medium gebootet, kann dieses auch als Standardmedium eingestellt werden. Da dann auf diesem schon ein Betriebssystem gefunden wird, spart sich das System das durchsuchen der anderen Laufwerke und der Bootvorgang beschleunigt sich hierdurch. Ein weiterer Ansatzpunkt zu Beschleunigung des Bootvorgangs ist das Verwenden einer schnelleren Festplatte oder SSD. Hierdurch wird die Ladephase

des Betriebssystems verkürzt, da in einer kürzeren Zeit mehr Daten transferiert werden können. Der letzte Ansatzpunkt zur Beschleunigung des Bootvorgangs ist das Deaktivieren von Autostartprozessen. Das eigentliche Betriebssystem steht so schneller voll zur Verfügung und muss nicht erst noch Programme starten.

1.4.10 Änderungen in Windows 8

In Windows 8 setzt Microsoft ein neues Bootverfahren namens Hybrid Boot ein, welches ca. $\frac{1}{3}$ Zeitersparnis im Bootvorgang mitbringt. Dieses Verfahren mischt das normale Booten, das normale Herunterfahren sowie die Ruhestandsfunktionen der bisherigen Windows Versionen. Ist diese Funktion aktiviert und das System wird heruntergefahren, wird anderes als beim Ruhezustand der Benutzer abgelmeldet und alle offenen Programme geschlossen. Der Windows Kernel und alle Dienste werden jedoch genauso wie beim Ruhezustand nur „schlafen gelegt“ und die Daten werden in die Hiberfile.sys geschrieben. Beim Starten müssen der Kernel und die Dienste nur wieder aufgeweckt werden.

1.5 Memory Managment

CPUs process data. This data is stored on Hard Disc Drives (HDD) or Solid State Discs (SSD). When this data is needed, it will be placed into the Random Access Memory (RAM). This process is called **memory allocation**.

There are different types of memory, primary and secondary. **Secondary memory**, for example HDDs or SSDs, is used to store data permanently or in case the **primary memory** runs out of space. Today, RAM is typically used as primary memory.

Primary and secondary memory together form the **virtual memory**. Virtual addresses are used to form a coherent address space. Todays CPUs have Memory Managment Units (MMU) that translate virtual addresses into physical addresses. Therefore there is no difference for the program between data that is stored into primary memory and data that is stored into secondary memory. Of course there is a difference. The RAMs **access time** is 1000-times faster than HDD.

When data is stored into primary memory, there will be gaps between blocks of data. These blocks are called **segments**. **Pages** are segments of fixed size. Because the size of segments is not predictable, pages are used instead.

1.5.1 Fetch Strategies - When?

When should data be stored into primary memory? There are two types of answers. The **demand fetching strategy** stores data into primary memory when it is needed. **Prefetching strategies** try to anticipate which data will be needed and thus try to store data into primary memory before it is needed.

1.5.2 Placement Strategies - Where?

Where should data be stored in the primary memory? There are at least three strategies trying to answer this question. (1) **First-Fit** will place data into the first free segment or page, (2) **Best-Fit** will place data into the smallest possible segment, and (3) **Worst-Fit** will place data into the largest free segment.

1.5.3 Replacement Strategies - Which?

Which data should be stored into secondary memory? When there is not enough space left in the primary memory, there have to be strategies to decide, which data should be stored into secondary memory. These strategies are called replacement strategies. The process of moving data from primary into secondary memory is called **swapping** or **paging**. Typically Windows uses a swap file and Linux raw swap partitions without any filesystem.

The Principle of Optimality
To get the most out of your memory you want it to be managed efficiently. If one knew how many instructions needed to be executed before a page was referenced, it would be really efficient to replace the page that has to wait the longest. But to collect and process the data in question to get an estimate, would be less efficient than other methods.

Random Page Replacement
If there are no more free pages available, a random page will be chosen to be swapped.

FIFO - First in First Out
The FIFO strategy always replaces the first page that was stored in memory. Therefore it is by default always the oldest page on the system that will be swapped.

LRU - Least Recently Used
Unlike the Random Page Replacement and FIFO this method passively takes important files into account when replacing a page. Necessary system-pages and pages used by currently running programs are more likely to be used more often and therefore have newer time stamps. This method's problem is, that each page has to be assigned a time stamp of some kind and those times need to be compared first. This creates a large memory and processing overhead.

LFU - Least Frequently Used
This method of choosing pages for replacement is even better in choosing less important pages like the "LRU"method. Since only uses need to be incremented each time a page is modified or read, the overhead should be much smaller.

Second Chance
The second chance strategy keeps data in a FIFO-layout. It runs through pages and resets the reference bits until it finds a page that has none. Then this page will be swapped. Considering, that more active pages will not be as likely to have no reference bit set once the algorithm reaches it again, the outcome will be very similar to the LRU strategy.

2 Credits

Im Folgenden sind alle[2] Beitragenden zur Zusammenfassung aufgelistet:

1. LF01 Gottwald
2. LF02 Trenkmann
3. LF04 Wiegand
4. LF04 Oenings & Wächter
   (a) Interrupts:
   (b) Prozessmanagment:
   (c) Lizenzen:
   (d) Boot-Prozess:
   Sebastian Heinke, Tobias Krenz, Jonathan Reuter
   (e) Memory Managment:
   Mirko Großmann
   (f) OS: Windows
5. LF04 Wissmann
6. LF04 Digitaltechnik
7. LF05 Wächter
8. LF06 Abu Shebika
9. LF06 Dresen
10. DKO Fischer
11. PK Trenkmann

[2]Wer sich trotz eines Beitrages hier nicht wiederfindet, spricht mich am besten in der Schule darauf an.