Betriebssysteme

A brave story about an operating system lecture

Michael Weiß

17. Mai 2023

Ich widme dieses Dokument der Autokorrektur

Schlafen ist auch eine Form von Kritik,
vor allem im Theater
-George Bernard Shaw-.

Inhaltsverzeichnis

ANOTATION: Version 1.3, Interne Zusammenfassung, Nicht Fertig, Nicht für die Weitergabe bestimmt.

1 Vorbemerkung

Das hier ist ein Skript das ich erstelle und das ein wenig ausgeartet ist. Vorab Das ganze erhebt weder einen Anspruch auf Korrektheit noch auf Vollständigkeit. Es ist ein lose zusammengewürfeltes Dokument aus dezenten Quellen im Internet, Vorlesungsfolien und dem was mir noch so dazu einfällt. Das Ganze ist eigentlich nur für den Privatgebrauch Folglich, wenn Sie nicht ich sind und ich Ihnen das hier nicht Selbst gegeben habe, sind sie hier eigentlich falsch! Aber wenn sie schon mal da sind bleiben oder gehen Sie das ist Ihre Sache.

Was das hier ist?

Ein Dokument in dem Kapitel weise alles zusammengefasst wird was so in den Folien erwähnt und beim nachlesen gefunden oder aus dem Hirn ergänzt wurde. Am ende Jedes Kapitels gibt es eine Zusammenfassung.

Korrekturen sind herzlich willkommen und erwünscht, ich bin nicht der Weisheit letzter Schluss und erhebe auch keinen Anspruch darauf es zu sein. Also gerne Ergänzungsvorschläge oder Korrekturanmerkungen.

-Michael Weiß-

2 Komponenten einer Rechenanlage

2.1 Komponenten

Die CPU (Central Processing Unit) oder Prozessor ist das zentrale Rechenelement in einem Computer oder einem anderen elektronischen Gerät. Sie ist das Gehirn des Computers und verarbeitet alle Daten und Anweisungen, die vom Betriebssystem und Anwendungen gesendet werden. Die CPU ist ein kleiner Chip, der auf dem Motherboard montiert ist und aus Millionen von Transistoren besteht.

Die CPU besteht aus drei wichtigen Komponenten: dem Control Unit (CU), dem Arithmetic Logic Unit (ALU) und dem Cache.

2.1.1 CPU

Control Unit (CU): Die Control Unit ist für die Steuerung des Ablaufs von Programmen verantwortlich. Sie empfängt Anweisungen aus dem Arbeitsspeicher und entscheidet, welche Operationen ausgeführt werden müssen. Die CU ruft auch Daten aus dem Arbeitsspeicher ab und sendet sie an die ALU.

Arithmetic Logic Unit (ALU): Die ALU führt mathematische und logische Operationen aus, wie z.B. Addition, Subtraktion, Multiplikation, Division, Vergleiche und logische Operationen wie AND, OR und NOT.

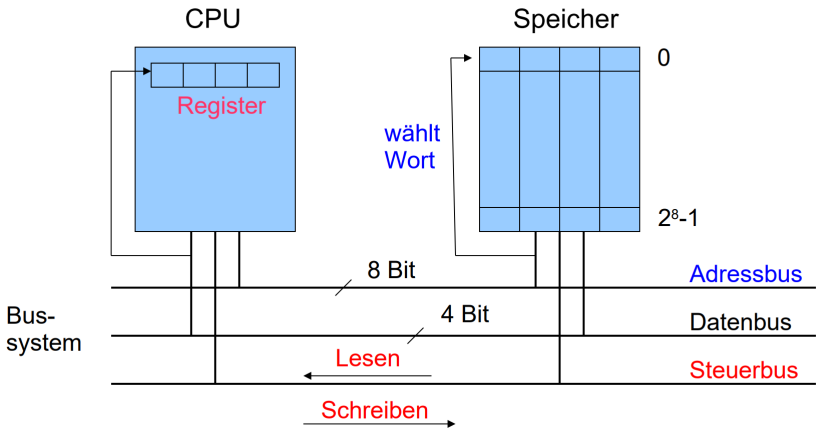
Cache: Der Cache ist ein schneller Speicher, der häufig verwendete Daten und Anweisungen enthält. Er kann die Verarbeitungsgeschwindigkeit der CPU verbessern, indem er schnellen Zugriff auf häufig verwendete Daten ermöglicht.

2.1.2 RAM

Der Hauptspeicher, auch als RAM (Random Access Memory) bezeichnet, dient als temporärer Speicherplatz für Daten und Programme, auf die die CPU zugreifen kann. Der Hauptspeicher ist ein flüchtiger Speicher, d.h. seine Inhalte gehen verloren, wenn der Computer ausgeschaltet wird.

Der Hauptspeicher besteht aus Speicherzellen, die in Zeilen und Spalten organisiert sind. Jede Speicherzelle kann eine binäre Zahl speichern, die entweder eine 0 oder eine 1 sein kann. Die Größe des Hauptspeichers wird in Bytes oder Gigabytes gemessen und bestimmt, wie viel Daten und Programme gleichzeitig im Speicher gehalten werden können.

Der Zugriff auf den Hauptspeicher erfolgt über eine eindeutige Adresse für jede Speicherzelle. Die CPU kann auf jede Speicherzelle im Hauptspeicher zugreifen, indem sie die Adresse der Zelle angibt.



2.2 Arcitekturen

Architekturen? Neuman, Havard und was Macht Turing da?!
Die von Neumann-Architektur und die Harvard-Architektur sind zwei grundlegende

Computerarchitekturen, während die Turing-Maschine ein theoretisches Konzept ist, das Alan Turing in den 1930er Jahren entwickelt hat.

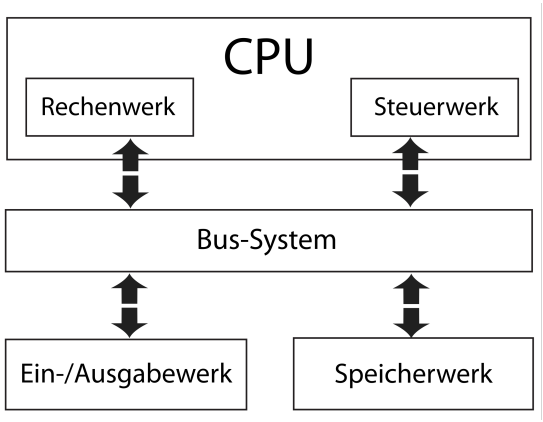


Abbildung 1: Von-Neumann-Architektur

von Neumann-Architektur Die von Neumann-Architektur ist eine Architektur, die in den meisten modernen Computern verwendet wird. Sie basiert auf der Idee, dass sowohl Daten als auch Anweisungen in einem gemeinsamen Speicher gespeichert werden. Das bedeutet, dass die CPU (zentrale Verarbeitungseinheit) die Daten und Anweisungen aus demselben Speicher liest und verarbeitet. Die von Neumann-Architektur ist sehr flexibel und ermöglicht es, dass das Betriebssystem und Anwendungsprogramme im selben Speicher ausgeführt werden können.

Harvard-Architektur Die Harvard-Architektur hingegen verwendet getrennte Speicher für Daten und Anweisungen. Das bedeutet, dass die CPU auf zwei getrennte Speicher zugreift, um Daten und Anweisungen zu lesen. Harvard-Architekturen werden oft in eingebetteten Systemen verwendet, bei denen Leistung und Energieeffizienz eine wichtige Rolle spielen.

Eine Turing-Maschine ist ein theoretisches Konzept, das Alan Turing entwickelt hat, um die Grenzen der Berechenbarkeit zu untersuchen. Eine Turing-Maschine besteht aus einem Band, auf dem Daten geschrieben werden können, einem Lesekopf, der die Daten liest und schreibt, und einem Zustandsdiagramm, das die Aktionen der Maschine steuert. Turing-Maschinen können verwendet werden, um zu zeigen, welche Probleme berechenbar sind und welche nicht.

In Bezug auf den Aufbau unterscheiden sich die von Neumann-Architektur und die Harvard-Architektur hauptsächlich darin, wie die Daten und Anweisungen gespeichert und abgerufen werden. Die von Neumann-Architektur verwendet einen gemeinsamen Speicher, während die Harvard-Architektur getrennte Speicher verwendet.

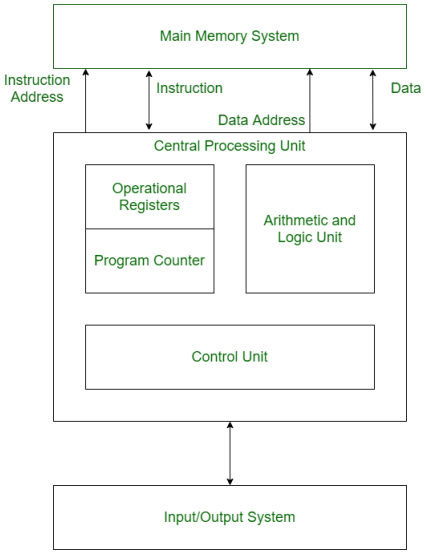


Abbildung 2: Harvard-Architektur

Zusammenfasung Die von-Neumann-Architektur und die Harvard-Architektur sind zwei grundlegende Architekturen für Computersysteme.

Die von-Neumann-Architektur besteht aus einem gemeinsamen Speicher für Daten und Programme, sowie einer CPU, die diese Daten verarbeitet. Die Harvard-Architektur hingegen verwendet separate Speicher für Daten und Programme und verfügt über separate Busse für den Daten- und Programmspeicher.

2.3 Schichten und schalen

Das Schichten- und Schalenmodell ist ein Konzept welches die Architektur von Betriebssystemen beschreibt. Es ist ein hierarchisches Modell, das verschiedene Schichten oder Ebenen von Funktionen definiert, die jeweils spezifische Aufgaben ausführen.

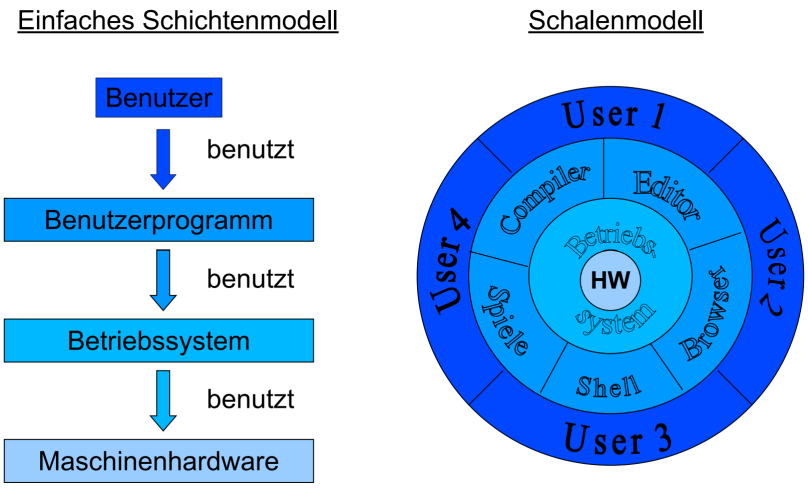


Abbildung 3: Modellansicht

Schichtenmodell beschreibt die Aufteilung des Betriebssystems in verschiedene Schichten, die jeweils eine bestimmte Funktion ausführen. Typischerweise gibt es drei Hauptebenen: die Anwendungsschicht, die Systemsoftware-Schicht und die Hardware-Schicht. Die Anwendungsschicht ist die oberste Schicht und umfasst alle Anwendungsprogramme, die auf dem Betriebssystem ausgeführt werden. Die Systemsoftware-Schicht ist die mittlere Schicht und umfasst das Betriebssystem selbst, sowie die Treiber, die es benötigt, um mit der Hardware zu kommunizieren. Die Hardware-Schicht ist die unterste Schicht und umfasst alle physischen Komponenten des Computers.

Schalenmodell bezieht sich auf die Möglichkeit, das Betriebssystem auf verschiedenen Ebenen zu nutzen. Jede Schicht kann eine oder mehrere Schalen oder Schnittstellen bereitstellen, die es dem Benutzer oder dem Programm ermöglichen, mit den Funktionen der jeweiligen Schicht zu interagieren. Auf diese Weise kann ein Benutzer oder Programm auf die verschiedenen Funktionen des Betriebssystems zugreifen, ohne direkt mit der darunterliegenden Schicht zu interagieren.

Insgesamt bietet das Schichten- und Schalenmodell eine klare Strukturierung und Aufteilung der Funktionen eines Betriebssystems, die es einfacher machen, die verschiedenen Teile des Systems zu verstehen und zu warten.

3 Generationen

Eine Generation wird hier so definiert: Betriebsystem + Rechnerarchitektur ⟵ Generation.

Generation 0 (1940 - 1950)

- Kein Betriebssystem
- Programmierung in Maschinenbefehlen durch umstecken von kabeln
- keine Programmiersprachen

Generation 1 (1950 - 1960)

- Lochkarten ersetzen umstecken von kabeln (genormt bis zu 80 Zeichen)
- Transistoren ersetzen Röhren.
- Programmiersprachen Assembler, Fortran Cobol

Generation 2 und 3 (1960 - 1975)

- IC(=itegrated circuits) werden verwendet
- LSI (=low scale Implementation) ermöglicht 10-100 mal mehr Transistoren
- Betriebssysteme sind Millionen Zeilen Assembler code
- Multiprogramming, Spooling, Timesharing kommen.

Generation 4 (1975 - 2006)

- VLSI = Abk. für Very-Large-Scale Integrated Circuit
- 100.000 Gatter
- Betriebssysteme wie MS-DOS, MS-Windows, UNIX/LINUX
- Computernetze kommen auf
- Rechenleistung steigt

Generation 5 (2006 – heute)

- PDAs smatphones etc kommen auf. Aktueller stand.

3.1 Timesharing

Neue Variante des Multiprogramming. Jeder Benutzer hat sein eigenes Terminal (Bildschirm, Tastatur direkt zum Computer). Die Rechenzeit wird in Stücke (Zeitscheiben, eng. Slices) aufgeteilt. Jeder Benutzer erhält der Reihe nach eine Zeitscheibe, in der er die CPU nutzen darf.

3.2 Spooling

spooling= (Simultaneous peripheral operations online) Jobs (Lochkartenstapel) werden direkt, wenn sie abgegeben werden auf Festplatte geladen. Betriebssystem kann nun neuen Job schneller beginnen.

3.3 Multiprogramming

Mehrprogrammverarbeitung, Multiprogramming (Mode), Multitasking; Betriebsart eines Computers, bei der sich mehrere Programme gleichzeitig ganz oder teilweise im Arbeitsspeicher befinden und abwechselnd von dem Zentralprozessor bearbeitet werden

4 Prozesse

Prozess ist der Ablauf eines Programmes in der Rechenanlage.

Ein Programm ist statisch es ändert seinen Programm code nicht. ein Prozess ist Das **Programm**+Zeitliche Dimension(zustand)=**Prozess**. Prozess ist aufgrund dessen das sich der zustand ändern kann dynamisch.

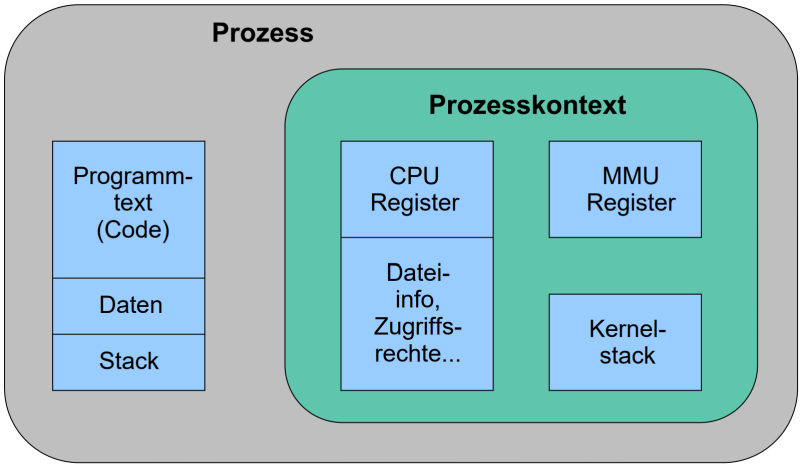
Für das Betriebsystem ist der Prozess ein objekt das dem Prozessor zugeteilt wurde, für den Anwender ist es der Handlungsträger. Alle moderne Betriebsysteme können mehrere Prozesse Verwalten.
Im Mehrprozessbetrieb unterscheidet man zwei Varianten:

- multiprozessorsysteme hier handelt es sich um echtes Multitasking
- Einprozessorsysteme Prozese müssen sequenziert werden und nebenläufig laufen

Die Ursprüngliche idee des Multitasking war es die auslastunf de r Cpu zu Optimieren.

4.1 Prozesskontext

Der Prozesskontext stellt einen Virtuellen Prozessor da. Prozesskontext und Prozess sind eng miteinander verbunden. Im Prozesskontext werden alle Informationen gespeichert die Notwendig sind um einen Unterbrochenen Prozess Fortzusetzen. Wenn ein Prozess gestartet wird, erstellt das Betriebssystem einen neuen Prozesskontrollblock(PCB) für **diesen** Prozess. Immer wenn der Prozess unterbrochen wird aktualisiert das Betriebssystem den Prozesskontext, im PCB. Damit wird verhindert, das Daten die zur Vortsezung nötig währen verloren gehen. Diese Daten währen zum Beispiel: CPU Register.



Prozesscontroll block Ein Prozesskontrollblock (PCB) ist eine Datenstruktur, die vom Betriebssystem verwendet wird, um alle wichtigen Informationen über einen laufenden Prozess zu speichern. Der PCB wird in einem Speicherbereich abgelegt, der dem Prozess zugewiesen ist und wird daher als ßpeicherresident"bezeichnet.

Der PCB enthält Informationen wie den Prozesszustand (bereit, blockiert, ausgeführt), Priorität, Speicherplatzbelegung, Programmzähler, Registerinhalte, offene Dateien und Ressourcen, auf die der Prozess zugreift.

Der PCB ist wichtig für die effektive Verwaltung von Prozessen auf einem Betriebssystem.

Zusammenfassend Der Prozesskontrollblock auch PCB abgekürzt, ist eine Datenstruktur, in der der Prozesskontext gespeichert wird. Sie wird verwendet um Prozesse Effizient zu verwalten. Bei der Erstellung eines Prozesses wird automatisch durch das Betriebssystem ein PCB erstellt in dem Der Prozesskontext gespeichert wird. Bei jeder Unterbrechung des Prozesses Wird der Prozesskontext im PCB aktualisiert. Dies verhindert, das Daten, wie Daten verloren gehen: Mögliche Daten PID Speicherreferenzen, Scheduling-Parameter.

PID ist die Process ID. Eine ID die eindeutig ist und mit der man jeden Prozess identifizieren und ansprechen kann.

PPID ist die Prarent Proces ID. Also Die ID des Parrent Prozesses.

4.2 Aus-lagerbarer Benutzerkontext

Der Benutzer Kontext umfasst Daten wie Stack, und Heap eines Prozesses. Daten die nicht aktiv benötigt werden, ausgelagert oder in den RAM verschoben. Deswegen aus lagerbar. Dies erlaubt eine bessere Nutzung des Speicherplatzes des Betriebssystems, was die Gesamtleistung des Systems verbessert. Prozesse beanspruchen nur die Ressourcen, die sie wirklich benötigen. Beim zugriff auf den ausgelagerten Benutzerkontext müssen die Daten aus dem Hauptspeicher(RAM) geladen werden, was aber immer noch effizienter ist, als ein vollständiger austausche zwischen Hauptspeicher und Festplatte, der Stadtfinden Müsste wenn man ganze Prozesse und nicht nur den Kontext auslagert.

5 Prozesse, Systeme und Zustände

Nun wissen wir schon einiges über Prozesse. Aber Wie läuft das ganze den jetzt im Rechner ab? Wo kommen Prozesse her? was machen sie und Wo gehen sie hin? Das schauen wir uns im Folgenden an. Wir verwenden hier Unix als Beispiel.

5.1 Parents and Childs

In Unix-ähnlichen Betriebssystemen sind Prozesse oft als Hierarchien organisiert, wobei ein Prozess als Ëlternprozessünd ein anderer als "Kindprozess"bezeichnet wird. Wenn ein Prozess einen Kindprozess erstellt, wird dieser Kindprozess zu einem direkten Kind des ursprünglichen Elternprozesses.

Erstellen eines Prozesses Der Elternprozess erstellt einen Kindprozess mit Hilfe der fork()-Systemaufrufsfunktion. Nachdem der Kindprozess erstellt wurde, führt er normalerweise eine andere Operation aus als der Elternprozess. Ein Kindprozess hat eine Kopie des Speichers des Elternprozesses und kann darauf zugreifen. Der Kindprozess kann dann den exec()-Systemaufruf aufrufen, um ein neues Programm in dem Speicherbereich des Kindprozesses zu starten und damit eine andere Operation auszuführen.

Der Elternprozess kann auf den Status des Kindprozesses mit Hilfe der wait()- oder waitpid()-Systemaufrufsfunktionen warten. Auf diese Weise kann der Elternprozess den Kindprozess überwachen und sicherstellen, dass er ordnungsgemäß ausgeführt wird.

Es ist wichtig zu beachten, dass die Kommunikation zwischen Eltern- und Kindprozessen normalerweise über Pipes oder andere Interprozesskommunikationsmechanismen erfolgt. Dadurch können sie Informationen austauschen und ihre Arbeit koordinieren.

Beenden Ein Prozess beendet sich selbst durch die Systemfunktion exit( ). Er teilt dem System seinen Exit-Status mit, den der Prozesserzeuger (sein Vaterprozess) lesen muss, damit der Prozess endgültig aus dem System entfernt wird.

5.2 Interprozesskommunikation IPC

Dies ist eine Exkursion für interessierte! Uns interessiert nur der Punkt Pipes! Geht es nur um den Stoff, dann lies hier die Zusammenfassung!

Interprozesskommunikation (IPC) ist ein Konzept in der Informatik, das beschreibt, wie Prozesse auf einem Betriebssystem miteinander kommunizieren können. Das IPC-Konzept ist wichtig, um die Zusammenarbeit und Koordination zwischen verschiedenen Prozessen auf einem Computer zu ermöglichen.

Es gibt verschiedene Methoden zur IPC, die in Betriebssystemen verwendet werden. Einige der gängigsten Methoden sind:

Pipes: Pipes kurz erklärt ermöglichen es, die Ausgabe eines Prozesses an die Eingabe eines anderen Prozesses weiterzuleiten.

Shared Memory: Shared Memory ist eine Methode zur IPC, bei der zwei oder mehr Prozesse auf denselben Bereich des Arbeitsspeichers zugreifen können. Dadurch können die Prozesse Daten austauschen und gemeinsam verwenden.

Message Queues: Message Queues sind eine Methode zur IPC, bei der Prozesse Nachrichten senden und empfangen können. Eine Message Queue ist eine spezielle Warteschlange, in der Nachrichten in der Reihenfolge empfangen werden, in der sie gesendet wurden.

Semaphore: Semaphore sind eine Methode zur IPC, die dazu dient, den Zugriff auf gemeinsam genutzte Ressourcen zu koordinieren. Semaphore können verwendet werden, um den Zugriff auf kritische Abschnitte im Code zu kontrollieren und Deadlocks zu verhindern.

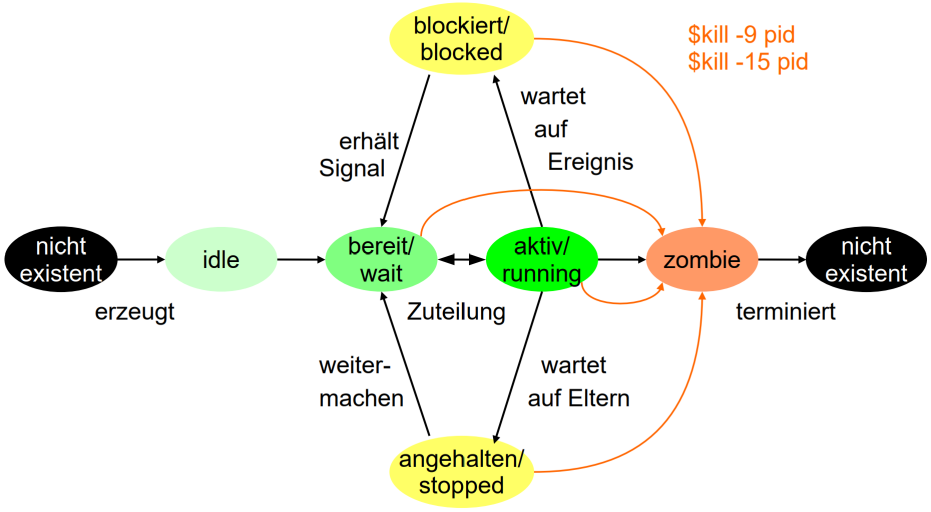
Sockets: Sockets sind eine Methode zur IPC, die es Prozessen ermöglicht, Daten über Netzwerkverbindungen auszutauschen. Sockets werden häufig in Client-Server-Anwendungen verwendet.

Die Auswahl der IPC-Methode hängt von den Anforderungen der spezifischen Anwendung ab, wie beispielsweise der Art der Daten, die ausgetauscht werden müssen, der Größe der Daten, der Anzahl der beteiligten Prozesse und der benötigten Geschwindigkeit.

Zusammenfassend: Wie schon erwähnt interessieren uns gerade für den Moment nur Pieps. Pieps kann man sich wie Rohre vorstellen. Sie geben den Output des einen Programms an das Andere Programm weiter, das diesen als Input verwendet. Pieps können in Linux mit "realisiert werden.

5.2.1 Piping

5.3 Prozesse in unix



In Oben stehenden Diagramm sehen wir viele was erstmal verwirrend wirkt, klären wir also erst einmal was passiert.

Also die schwarzen Ovale sind logischerweise, nicht existent, so ein bisschen aus dem nichts kommst du zum nichts gehst du mäßig.

IDLE Zustand In Unix bezeichnet der Begriff ïdleëinen Prozess, der nicht aktiv ist und auf eine Aufgabe wartet. Ein Prozess kann aus verschiedenen Gründen als ïdleëingestuft werden, zum Beispiel wenn er auf eine Eingabeaufforderung oder ein Ereignis wie das Eintreffen von Daten wartet.

In der Regel sind ïdle"Prozesse in Unix nicht problematisch und dienen dazu, Systemressourcen zu sparen, wenn ein Prozess nicht aktiv ausgeführt wird.

Bereit/wait Hier ist der Prozess Bereit, er wartet darauf aktiv zu werden. Solange er dies nicht ist ist er in der Warteschlange.

aktiv/running Dies ist der Zustand in dem der Prozess wirklich läuft, er also auf der CPU ausgeführt wird.

blockiert/blocked Ein Prozess im Blocked Zustand wen er auf eine bestimmte Bedingung wartet, um seine Arbeit fortzusetzen. Im Blocked Zustand ist der Prozess inaktiv und blockiert andere Prozesse, die darauf warten, auf die gleiche Ressource zuzugreifen.

Angehalten/stopped In diesem Zustand ist ein Prozess wenn der Eltern Prozess den Prozäss anhält, dies Geschieht z.b bei einem Debugger.

Zombie Ein Zombie-Prozess (auch als "defunctöder "dead"Prozess bezeichnet) ist ein Prozess in Unix-ähnlichen Betriebssystemen, der beendet wurde, aber dessen Exit-Status noch nicht vom übergeordneten Prozess abgeholt wurde. Der Zombie-Prozess nimmt noch Systemressourcen in Anspruch, aber er führt keine Aktivitäten mehr aus, da er bereits beendet wurde.

Zombie-Prozesse können entstehen, wenn der übergeordnete Prozess, der den Kinnprozess erstellt hat, nicht ordnungsgemäß auf die Beendigung des Kinnprozesses reagiert hat.

Das Vorhandensein von Zombie-Prozessen im System verursacht in der Regel keine größeren Probleme, **kann aber** aufgrund der Beschränktheit der möglicherweise irgend wan Problematisch werden.

Prozess Erzeugung Wenn ein Prozess erzeugt wird, wandert dieser in den ÏDLE-zustand. Hier Wartet er bis er alle Daten hat und in wartend kommt. Kommt er dann an die reihe so wechselt er in den Aktiven zustand. Wenn alles Regulär abläuft, wird er am ende Korrekt terminiert.

5.4 Ein bischen linux

Wir können dies Prozesse auch als Anwender einsehen. Dies geht unter Linux mit dem *ps* Befehl, hier eine Variante davon mit einigen Informationen:

```
ps -ax
PID     TTY             STATE   TIME    COMMAND
1       ?               S       0:04    init [2]
2       ?               SW      0:00    [keventd]
3       ?               SWN     0:00    [ksoftirqd_CPU0]
```

Was sehen wir hier?

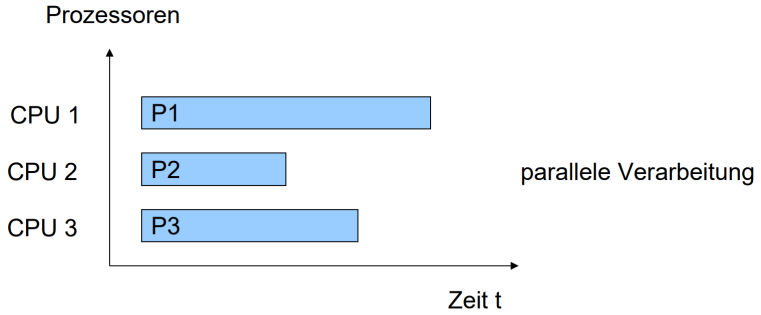
- PID (Prozess ID) ist eine Nummer, die jeden Prozess eindeutig identifiziert.
- Unter TTY wird der Name des Terminals angegeben, das den Prozess kontrolliert.
- TIME zeigt die von jedem dieser Prozesse bisher genutzte Rechenzeit an.
- CMD (oder COMMAND) zeigt den Befehl an, mit dem der Prozess gestartet worden ist.

Man kann sich mit *ps* noch wesentlich mehr anzeigen lassen, dies Sprengt hier aber den Rahmen. Wen dies interessiert, den Verweise ich hier auf die Man page.

Mehrprozessor system: Ein Mehrprozessorsystem ist ein Computersystem, das mehr als einen Prozessor (auch als CPU bezeichnet) hat. Es kann entweder aus mehreren physischen Prozessoren auf einer einzigen Hauptplatine oder aus mehreren Computern bestehen, die miteinander verbunden sind und als ein einziges System arbeiten.

Mehrprozessorsysteme bieten eine höhere Verarbeitungsgeschwindigkeit und eine höhere Skalierbarkeit im Vergleich zu Einprozessorsystemen. Durch die Verwendung mehrerer Prozessoren können mehrere Prozesse gleichzeitig ausgeführt werden, was die Ausführungszeit von Aufgaben reduziert und die Effizienz des Systems erhöht. Die Leistung eines Mehrprozessorsystems hängt von der Anzahl der Prozessoren, ihrer Geschwindigkeit, der Art der Verbindung und dem Betriebssystem ab, das die Prozessoren verwaltet.

Mehrprozessorsysteme werden in der Regel in rechenintensiven Anwendungen wie Datenbanken, Wissenschaft und Ingenieurwesen, Finanzen und Wettervorhersage eingesetzt. Es gibt auch spezialisierte Mehrprozessorsysteme wie Supercomputer, die für komplexe Aufgaben wie Simulationen und Berechnungen eingesetzt werden.



Einprozessor system Ein Einprozessorsystem ist ein Computersystem, das nur einen einzigen Prozessor (auch als CPU bezeichnet) hat. Das bedeutet, dass es nur in der Lage ist, einen Befehlssatz zu einer bestimmten Zeit auszuführen. In einem Einprozessorsystem werden die Prozesse sequentiell ausgeführt, was bedeutet, dass jeder Prozess nacheinander ausgeführt wird und erst dann, wenn der vorherige Prozess abgeschlossen ist.

Ein Einprozessorsystem ist in der Regel kostengünstiger und einfacher in der Verwaltung als ein Mehrprozessorsystem. Es ist geeignet für Anwendungen, die nicht unbedingt eine sehr hohe Verarbeitungsgeschwindigkeit erfordern und auch nicht viele gleichzeitige Prozesse ausführen müssen.

Die meisten Desktop-Computer und Laptops sind Einprozessorsysteme, die in der Lage sind, mehrere Aufgaben auszuführen, indem sie Prozesse in schneller Abfolge wechseln

Zusammenfassend Mehrprozessorsysteme Können echtes Multitasking, Einprozessorsysteme nicht. Echtes Multitasking heißt, das mehrere Prozesse Parallel ausgeführt werden können. Bei Einprozessorsystemen erreicht man dies indem man die Prozesse aufteilt(in Zeitscheiben (egl.slicing)) und diese dann nacheinander abarbeitet. Dadurch wirkt es so als Währen mehrere Prozesse am laufen, ähnlcih wie beim Film wo bei einer schnellen Abfolge an einzelnen Bildern eine Bewegung entsteht, wird aus vielen Zeitscheiben die schnell abgearbeitet werden ein "Multitaskingöder "Multiprogramming"realisiert.

5.5 Sequenzielle Verarbeitung

Die sequentielle Verarbeitung (auch serielle Verarbeitung genannt) bedeutet, dass die Verarbeitung von Daten oder Prozessen in einer bestimmten Reihenfolge erfolgt, bei der jeder Prozess oder Datenblock nacheinander ausgeführt wird, bevor der nächste gestartet wird. Die sequentielle Verarbeitung wird typischerweise in Einprozessorsystemen eingesetzt, da nur ein Prozessor zur Verfügung steht, um alle Aufgaben sequenziell auszuführen. In der sequentiellen Verarbeitung müssen Prozesse warten, bis der vorherige Prozess abgeschlossen ist, bevor sie gestartet werden können.

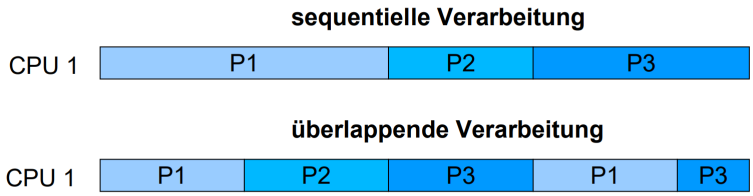
- Bestimmte Reihenfolge
- Prozesse werden nacheinander ausgeführt
- Prozesse müssen warten bis vorheriger Prozess abgeschlossen

5.6 überlappende Verarbeitung

Die überlappende Verarbeitung (auch parallele Verarbeitung genannt) bezieht sich auf die Verarbeitung von Daten oder Prozessen, die gleichzeitig stattfinden können, ohne dass sie aufeinander warten müssen. Mehrprozessorsysteme oder Multi-Core-Prozessoren

können die überlappende Verarbeitung ermöglichen, indem sie mehrere Prozesse gleichzeitig ausführen. In der überlappenden Verarbeitung können mehrere Aufgaben gleichzeitig ausgeführt werden, was die Ausführungszeit reduziert und die Effizienz des Systems erhöht.

- Prozesse Müssen nicht aufeinander Warten
- mehrere aufgaben gleichzeitig ausführbar
- Reduzierte Ausführungszeit gesteigerte Effizienz



5.7 Unterschiede zwischen sequenzieller und Überlappender Verarbeitung

Ein wichtiger Unterschied zwischen sequentieller und überlappender Verarbeitung besteht darin, dass in der sequentiellen Verarbeitung jeder Prozess oder Datenblock erst abgeschlossen werden muss, bevor der nächste gestartet werden kann. In der überlappenden Verarbeitung können jedoch mehrere Prozesse gleichzeitig ausgeführt werden, wodurch sich die Verarbeitungszeit verringert und die Effizienz erhöht wird.

6 Scheduling und Dispatching

Wenn wir uns moderne Rechner anschauen und dort einmal ein Programm wie den Task Manager öffnen sehen wir da viele Prozesse. Wäre das alles unkoordiniert einfach so, dann würde bald Chaos ausbrechen. Zum Glück, haben wir ja schon gelernt, das ein Betriebssystem die Ressourcen eines Rechners verwaltet und zuteilt. Aber wie macht es das und nach Welchen Kriterien wird das entschieden?

Das ist das um das es in diesem Kapitel geht. Aber Vielleicht klären wir erst einmal was eigentlich Scheduling und Dispatching ist.

Scheduling beschäftigt sich mit der Entscheidung, welcher Prozess als nächstes auf der CPU ausgeführt werden soll. Das Betriebssystem muss eine geeignete Strategie entwickeln, um die verfügbaren Ressourcen optimal zu nutzen. Dies kann beispielsweise durch Priorisierung von Prozessen oder durch die Verwendung von Zeitplänen erfolgen.

Dispatching ist die tatsächliche Ausführung des ausgewählten Prozesses auf der CPU. Das Betriebssystem muss sicherstellen, dass alle erforderlichen Ressourcen für den Prozess verfügbar sind und dass er nicht durch andere Prozesse beeinträchtigt wird. Das Dispatching wird in der Regel von einem Scheduler gesteuert.

6.0.1 Ziele des Prozessschedulings

- Maximale Auslastung der CPU: Ziel 100
- Minimale Wartezeit (Waiting time

Problem: Diese Ziele sind weder vollständig noch Konsistent **Beispiel** Autovermietung: Sind alle Wagen gut ausgelastet, müssen neue Kunden warten. Wartet kein Kunde, so existieren Autos ohne Benutzung.

=> es gibt keinen idealen Schedulingsalgorithmus!

Begriffe

- Ausführungszeit = eit die ein Prozess läuft (Ausführungszeit = Bedienzeit + warten auf I/O-Ende + Zeit in Warteschlange bereit)
- Wartezeit legt der Scheduler fest soll minimal sein
- Antwortzeit(respinse time) Zeit zwischen einer Eingabe und der Antwort

Das Scheduling kann in verschiedene Kategorien unterteilt werden.

- präemptives Scheduling
- nicht-präemptives Scheduling

prämptiven Scheduling Beim präemptiven Scheduling kann der Scheduler die Ausführung eines laufenden Prozesses unterbrechen, um einem höher priorisierten Prozess CPU-Zeit zuzuweisen.

nicht-prämetiven-Schedulin Beim nicht-präemptiven Scheduling wird hingegen die Ausführung eines Prozesses nicht unterbrochen, solange dieser noch läuft und CPU-Zeit benötigt.

Anmerkung:

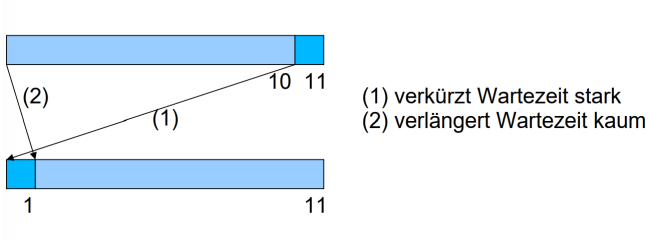
- non-preemptive (nicht unterbrechende)
- preemptive=unterbrechende

Es gibt verschiedene Scheduling-Algorithmen die zur Implementierung verwenden werden können. Wir beginnen mit den Algorithmen für das nicht-primitiven-Schedulings:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Highest Response Ratio Next (HRN)
- Priority Scheduling (PS)

First Come First Serve (FCFS) Implementiert durch FIFO Warteschlange (First in First Out) Einfach zu implementieren aber die Mittlere Ausführungszeit kann groß werden.

Shortest Job First (SJF) Der Prozess mit der gschätzt kürzersten Bedienzeit wird Vorgezogen. Dies begünstigt stark interaktive Prozesse. Das Minimiert die mittlere Wartezeit. Das Problem hier ist, das Prozesse die Rechenintensiv sind sehr lange Brauchen und eventuell auch einfach nie drankommen.



Highest Response Ratio Next (HRN)

$$H = \frac{Antwortzeit}{Bedienzeit} \; Sollmglichst Grosein.$$

H ist dann Groß wenn:

- 1) B Klein d.h. ziehe Jobs mit kurzer Bedienzeit vor
- 2) A groß d.h begrenzt Wartezeit von jobs mit langer Bedienzeit

HRN begünstigt Prozesse mit einer kurzen geschätzten Ausführungszeit und einer hohen Wartezeit, da diese eine höhere Antwortverhältniszahl aufweisen und somit priorisiert werden. Dadurch wird eine schnellere Antwortzeit und ein besserer Durchsatz erreicht. Ein Nachteil des HRN-Algorithmus ist, dass er einen hohen Overhead hat, da er die Wartezeit und Ausführungszeit jedes Prozesses kontinuierlich berechnen muss. Zudem kann er zur Verzögerung längerer Prozesse führen, da diese möglicherweise nicht prioritär ausgeführt werden.

Priority Scheduling (PS) Beim Priority Scheduling (PS) werden die Prozesse nach ihrer Priorität geordnet, wobei höher priorisierte Prozesse vor niedriger priorisierten Prozessen ausgeführt werden. Der **Vorteil** von PS besteht darin, dass wichtige Prozesse bevorzugt ausgeführt werden und somit eine höhere Systemleistung erreicht werden kann.
Ein **Nachteil** kann jedoch sein, dass niedrig priorisierte Prozesse möglicherweise lange auf ihre Ausführung warten müssen, was zu einer Verzögerung der Gesamtausführungszeit führen kann.

Es gibt auch algorithmen für Preemtives Scheduling:

Zeitscheiben Das Wichtigste Konzept hier ist das **Zeitscheibenverfahren.** Verfügbare zeit für das Betriebsmittel, also die CPU wird in einzelne, gleich große Zeitscheiben aufgeteilt. Diese werden im englischen Auch sliches genannt und die Technik dahinter Sliching.

Sobald eine neue Zeitscheibe beginnt unterbricht der Dispatcher den aktuellen Prozess und wählt den nächsten am Warteschlangen Anfang aus. Der unterbrochene Prozess kommt in die Bereit Warteschlange. Strategien:

- Round Robin (RR)
- Dynamic Priority Round Robin (DPRR)
- Shortest Remaining Time First (SRTF)
- Priority Scheduling (PS)
- Kombination FCFS-Strategie der RR-WS durch Prioritätsstrategie ersetzen.

Round Robin (RR) Jeder Prozess erhält dabei eine bestimmte Zeitdauer zur Ausführung, bevor er in die Warteschlange zurückkehrt und der nächste Prozess ausgeführt wird. Der Vorteil von RR besteht darin, dass Prozesse fair behandelt werden und kein Prozess die Ausführung zu lange monopolisieren kann. Ein Nachteil ist, dass kurze Prozesse möglicherweise unnötig lange warten müssen, bis sie ausgeführt werden, da längere Prozesse möglicherweise bereits eine höhere Priorität haben.

- RR = FCFS-Strategie + Zeitscheibenverfahre
- Antwortzeit ist proportional zu der mittleren Bedienzeit

Dynamic Priority Round Robin (DPRR) Dynamic Priority Round Robin (DPRR) ist eine Erweiterung des Round Robin-Algorithmus, bei dem jeder Prozess eine dynamische Priorität zugewiesen bekommt, die sich basierend auf der Anzahl an bereits ausgeführten Quanten und der Dringlichkeit des Prozesses verändert. Dadurch wird eine

fairere Behandlung von Prozessen mit unterschiedlicher Priorität erreicht und gleichzeitig sichergestellt, dass wichtige Prozesse schneller ausgeführt werden.

- RR erhält als Vorstufe eine prioritätsgesteuerte WS
- Priorität steigt mit der Zeit
- wenn Schwellpriorität erreicht ist kommt der Prozess in die RR Warteschlange

Shortest Remaining Time First (SRTF) Prozess mit der kürzesten verbleibenden Ausführungszeit als nächstes ausgeführt wird. Wenn ein neuer Prozess eintrifft, wird geprüft, ob er eine kürzere verbleibende Ausführungszeit hat als der aktuell ausgeführte Prozess. Wenn ja, wird der aktuelle Prozess unterbrochen und der neue Prozess ausgeführt. Der Vorteil von SRTF ist, dass Prozesse mit kürzeren Ausführungszeiten bevorzugt werden, was eine schnellere Durchlaufzeit und eine höhere Effizienz ermöglicht. Der Nachteil ist jedoch, dass längere Prozesse möglicherweise verhungern und nicht ausgeführt werden, wenn ständig kurze Prozesse eintreffen.

- SJF wird zu SRTF
- Job mit kürzester Restbedienzeit wird vorgezogen.

Priority Scheduling (PS) Jeder Prozess bekommt eine Priorität zugewiesen und der Prozess mit der höchsten Priorität wird als nächstes ausgeführt. Wenn ein neuer Prozess eintrifft, wird ihm eine Priorität zugewiesen, die höher oder niedriger als die Prioritäten der bereits ausgeführten Prozesse sein kann. Ist sie Höher Verdrängt er den aktuellen Prozess. Vorteil von PS ist, dass wichtige Prozesse bevorzugt werden und eine schnelle Ausführung gewährleistet wird. Der Nachteil ist jedoch, dass Prozesse mit niedrigerer Priorität möglicherweise verhungern und nicht ausgeführt werden, wenn ständig Prozesse mit höherer Priorität eintreffen.

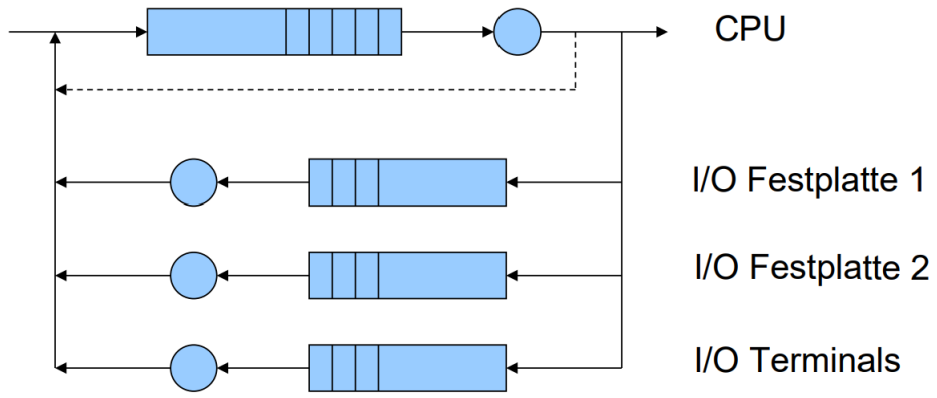
Zusammenfassend: Die Höchste Priorität wird genommen und ausgeführt

zum Schluss gibt es noch die Kombination: FCFS-Strategie der RR-WS durch Prioritätsstrategie ersetzen

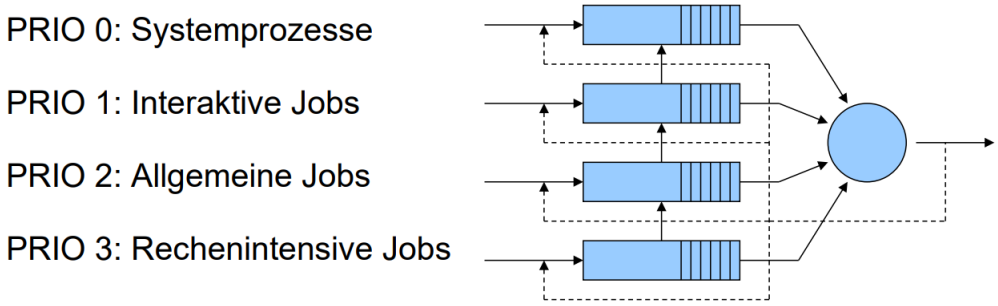
Multiple Warteschlangen Bei Multiple Warteschlangen handelt es sich um ein Scheduling-Verfahren, das in zwei Fällen angewendet werden kann. Im ersten Fall gibt es nur eine CPU, aber mehrere DMA-Kontroller, die jeweils eine eigene Warteschlange haben. Im zweiten Fall gibt es mehrere Jobs mit unterschiedlicher Priorität, für die jeweils eine eigene Warteschlange angelegt wird. Das Multi-level-Scheduling-Verfahren wird angewendet, um diese Warteschlangen zu verwalten und den Prozessen eine Priorität zuzuweisen, um eine effiziente Ausführung zu gewährleisten.

Einschub DMA Kontroller Anmerkung wenn bekannt überspringen. *Ein DMA (Direct Memory Access) Kontroller wird eingesetzt, um den Datentransfer zwischen Peripheriegeräten und dem Hauptspeicher eines Computers zu ermöglichen, ohne dass der Prozessor des Computers dafür verwendet werden muss. Der DMA-Kontroller übernimmt die Steuerung des Transfers und greift direkt auf den Hauptspeicher zu, um die Daten zu übertragen. Dadurch wird der Prozessor entlastet und kann andere Aufgaben ausführen, während der DMA-Kontroller den Datentransfer durchführt. Dies führt zu einer verbesserten Systemleistung und einer effizienteren Nutzung der CPU-Ressourcen. Wesentlich ist hierbei das die Daten dort in einem "Rutsch"gelesen werden können, nicht extra noch nach teilen von Datei gesucht werden muss.*

1 Fall Nur eine CPU aber mehrere DMA Kontroler (DMA= DIRECT MEMORY ACCSES)



2 Fall Hier wird eine extra Warteschlange Pro priorität aufgemacht.



Scheduling ist ein Wesentlicher Bestandteil von Betriebsystemen und trägt entscheidend zur Leistungsfähigkeit und Stabilität des Systems bei. Ein effektives Scheduling sorgt dafür, dass die Systemressourcen Optimal genutzt werden.

6.1 Multiple Scheduler

Wir vernachlässigen Bisher das nicht alle Prozesse im Hauptspeicher sind. Ok Aber wo ist das Problem?

Nja das Problem besteht darin, dasdie Festplatte viel langsamer angesprochen werden kann, als der Hauptspeicher.

Swaping Das Kopieren vom Hauptspeicher auf die Festplatte und von der Festplatte in den Hauptspeicher heißt Swaping.

Dafür verwendet man unter anderem die Swap Partition auf der Festplatte. Wir hatten es ja schon beim Prozesskontext das wir teile auslagern. Hier kommt das nun zum Tragen. Einziges Problem die Zugrifgeschwindigkeit der Festplatte.

Also wäre es doch Clever das jetzt mitzuberücksichtigen. Und genau das machen wir jetzt auch.

Man erhält eine Optimierung, wenn der neu ausgewählte Prozess im Hauptspeicher(RAM) ist.

Realisierung: Mittelzeitscheduler (zweiter/dritter Scheduler) regelt die Zuordnung der Prozesse zum Hauptspeicher.

6.1.1 Die Drei Arten im Überblick

Es gibt drei Arten von Scheduling in Betriebssystemen:

Langzeit-Scheduling (Job-Scheduling): Entscheidet, welche Prozesse in die Warteschlange aufgenommen werden sollen.

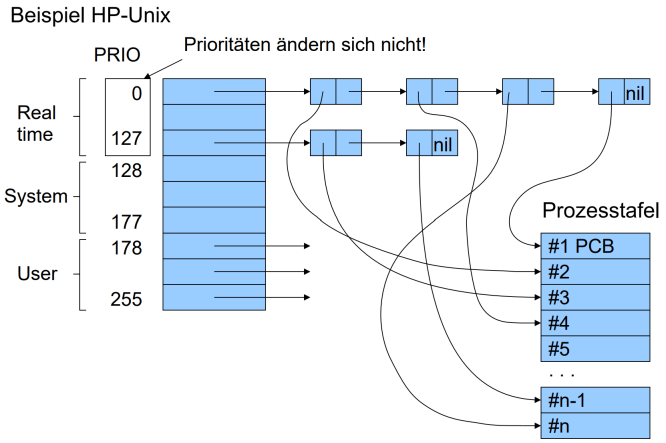
Kurzzeit-Scheduling (CPU-Scheduling): Entscheidet, welcher Prozess als nächstes die CPU erhält und für wie lange.

Mittelzeit-Scheduling (I/O-Scheduling): Entscheidet, wie E/A-Anforderungen von Prozessen in die Warteschlange aufgenommen und bearbeitet werden.

Das Langzeit-Scheduling ist ein grober Prozess, der selten durchgeführt wird. Das Kurzzeit-Scheduling ist ein feiner Prozess, der häufig durchgeführt wird. Das Mittelzeit-Scheduling liegt dazwischen und verwaltet E/A-Aufgaben. Zusammen stellen diese Scheduling-Arten sicher, dass die CPU und anderen Systemressourcen optimal genutzt werden, um die Leistung des Systems zu maximieren.

6.1.2 Realisierung in unix

RR mit multi-level feedback Scheduling Jeder Prozess hat eine initiale Priorität, die mit der Zeit erhöht wird. Für alle Prozesse mit der gleichen Priorität gibt es eine Warteschlange bei der das Prinzip First come first serve gilt, realisiert wird dies durch eine Verkettete Liste. Der Informationsteil der Listenelemente zeigt auf einen Proces kontroll Block (PCB) in der Prozesstafel



Da das jetzt erstmal viel input war, fassen wir das zusammen.

6.2 Zusammenfassend:

Das Scheduling hat das Ziel, die CPU-Auslastung zu maximieren und die Wartezeit zu minimieren.
Es gibt keinen idealen Scheduling-Algorithmus, da es immer Kompromisse zwischen diesen Zielen gibt.

Scheduling Plant wie die Zuteilung von Betriebsmitteln Dispatching Teilt diese dann auch wirklich zu

Man unterteilt in Prämitven Scheduling und nicht-prämetiven-Scheduling. Prämptives Scheduling heist prozesse können unterbrochen werden. Nicht-Prämetives-Scheduling heißt prozesse können nciht unterbrochen werden.

Algorithmen für das nicht-primitiven-Schedulings:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Highest Response Ratio Next (HRN)
- Priority Scheduling (PS)

Prämetives Scheduling:

- Round Robin (RR)
- Dynamic Priority Round Robin (DPRR)
- Shortest Remaining Time First (SRTF)
- Priority Scheduling (PS)
- Kombination FCFS-Strategie der RR-WS durch Prioritätsstrategie ersetzen.

Drei Typen von Scheudling:

- Langzeit sceduler
- kurzeitscheduler
- midelzeitsceduler

Auch existieren Multiple Warteschlangen. Hier gibt es entweder den Fall, das aufgrund der Vorhandenen Direct Memory Controler man für diese Eigene Warteschlangen erstellt oder Dass für jede Priorität eigene Warteschlangen erstellt werden.

Multiple Scheduler: Beachten das nciht alle Prozesse im Hauptspeicher sind. Verhindern, das Prozesse die Nicht im Hauptspeicher sind genommen werden, wofür man einen Zweit oder dritt scheduler einsetzt. Unix Systeme verwenden diesen Ansatz.

Scheduling ist Masgeblich für die Performence von Betriebsystemen.

7 Kritische Abschnitte

Der Schutz kritischer Abschnitte innerhalb des Betriebssystemkerns gehört zu den diffizilsten Aspekten der Kernelprogrammierung.

Entstehung Kritische Abschnitte entstehen durch Parallelität, z.B. wenn mehrere Prozesse auf das gleiche Betriebsmittel oder Datenstruktur zugreifen. Wenn kritische Abschnitte nicht geschützt werden, kann es zu Race Conditions kommen, bei denen das Ergebnis einer Operation vom Zufall abhängt.

Race-Condition Es handelt sich um eine beim Programmablauf auftretende, kritische Wettlaufsituation um den Zugriff auf eine Ressource oder Operation, die von zwei Threads unternommen wird. Eine der beiden Threads gewinnt und bekommt den Zugriff. Der unterlegene Thread führt den Zugriff später aus, was aber unerwünschte Folgen haben kann. Im Wikipedia-Artikel ist ein Beispiel für den Zugriff auf eine Speichervariable mit einem Zähler genannt

7.1 Lösungen

Es gibt mehrere Lösungsoptionen für kritische Abschnitte und Race Conditions in der Kernelprogrammierung. Hier sind einige Möglichkeiten:

Mutex: Ein Mutex (kurz für Mutual Exclusion) ist eine Sperre, die den Zugriff auf eine gemeinsam genutzte Ressource regelt. Ein Mutex wird vor dem Betreten eines kritischen Abschnitts erworben und nach dem Verlassen des Abschnitts freigegeben. So wird sichergestellt, dass immer nur ein Prozess gleichzeitig auf die gemeinsame Ressource zugreifen kann.

Semaphor: Ein Semaphor ist ein Zähler, der den Zugriff auf eine gemeinsame Ressource regelt. Wenn der Zähler auf Null ist, blockiert der Prozess, der auf die Ressource zugreifen möchte. Wenn der Zähler größer als Null ist, wird die Ressource freigegeben und der Zähler um eins verringert.

Spinlock: Ein Spinlock ist ähnlich wie ein Mutex, aber anstatt zu blockieren, wenn der Zugriff auf eine gemeinsame Ressource verweigert wird, dreht der Prozess einfach in einer Schleife und versucht es erneut, bis der Zugriff gewährt wird. Dies ist effizienter als ein Mutex, wenn die Verzögerung kurz ist, aber bei längeren Wartezeiten wird es ineffizient.

Atomic Operations: Atomare Operationen sind Operationen, die als Ganzes ausgeführt werden, ohne von anderen Prozessen unterbrochen zu werden. Diese Art von Operationen wird verwendet, um Konflikte beim Zugriff auf gemeinsam genutzte Ressourcen zu vermeiden.

Reader/Writer Lock: Ein Reader/Writer Lock ist eine Variante des Mutex, bei der mehrere Prozesse gleichzeitig lesend auf eine gemeinsame Ressource zugreifen können, aber nur ein Prozess schreibend. Dadurch können Lesezugriffe parallel ausgeführt werden, während Schreibzugriffe blockieren.

Diese Lösungsoptionen können je nach Anwendungsfall unterschiedlich effektiv sein. Ein guter Treiberprogrammierer muss die richtige Lösungsoption für seine spezifische Anwendung auswählen und implementieren, um kritische Abschnitte und Race Conditions zu vermeiden.