**8.Vorlesung**

# Betriebssysteme (BTS)

Christian Baun
cray@unix-ag.uni-kl.de

Hochschule Mannheim – Fakultät für Informatik
Institut für Betriebssysteme

27.4.2007

# Heute

- 1.Testklausur mit 41 Fragen aus den Vorlesungen 1 bis 7.

# Aufgabe 1

Beschreiben Sie, was ein Betriebssystem ist, wo seine Position ist und was seine Aufgaben sind.

- Verwaltung und Verteilung der Betriebsmittel auf die Benutzer. Betriebsmittel sind u.a. Prozessoren, Hauptspeicher, Ein-/Ausgabegeräte, Dateien und Netzwerkdienste.
- Bereitstellung einer Benutzerschnittstelle (GUI) zur Steuerung des Systems.
- Bereitstellung von Schutzmechanismen gegen unbefugte Zugriffe, Angriffe und fehlerhafte Bedienung.
- Bereitstellung von Werkzeugen zur Benutzer- und Dateiverwaltung.
- Bereitstellung von Bibliotheken und Entwicklungswerkzeugen.

# Aufgabe 2

Beschreiben Sie die Merkmale von **Stapelbetrieb**, **Dialogbetrieb** und **Echtzeitbetrieb**.

- Stapelverarbeitung ist üblicherweise eine interaktionslose Ausführung einer Folge von Jobs. Nach dem Start eines Jobs wird er bis zum Ende oder Auftreten eines Fehlers ohne Interaktion mit dem Benutzer abgearbeitet. Stapelbetrieb eignet sich gut zur Ausführung von Routineauftragen.
- Bei Dialogbetrieb können mehrere Benutzer an einem Computer gleichzeitig, konkurrierend, arbeiten. Jeder Benutzer glaubt, dass er die gesamten Rechenleistung der CPU für sich alleine hat. Die Verteilung der Rechenzeit geschieht mit Zeitscheiben und kann nach unterschiedlichen Strategien erfolgen.
- Echtzeitbetriebssysteme sind Multitasking-Betriebssysteme mit zusätzlichen Echtzeit-Funktionen für die Einhaltung von Zeitbedingungen und die Vorhersagbarkeit des Prozessverhaltens. Alle Programme sind ständig betriebsbereit und Ergebnisse stehen in einer vorgegebenen Zeitspanne zur Verfügung. Wesentliches Kriteriem von Echtzeitbetriebssystemen ist die Reaktionszeit.

# Aufgabe 3

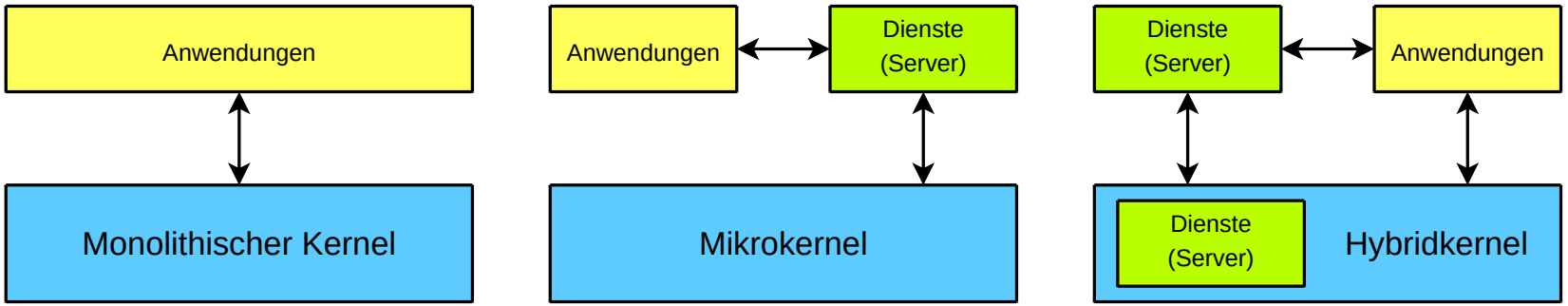
Nennen Sie fünf typische Einsatzgebiete von Echtzeitbetriebssystemen.

- Schweißrobotter
- Reaktorsteuerung
- ABS
- Überwachungssysteme auf der Intensivstation
- Flugzeugsteuerung
- Telefonanlage
- Parkschein- oder Fahrkartenautomat

# Aufgabe 4

Beim Aufbau von Betriebssystemen unterscheidet man die Kernelarchitekturen **Monolithischer Kernel**, **Minimaler Kern** (Mikrokernel) und **Hybridkernel** (Makrokernel). Worin unterscheiden sich diese Kernelarchitekturen und was sind die Vor- und Nachteile der unterschiedlichen Kernelarchitekturen?

- Die Kernelarchitekturen unterscheiden sich darin, ob die Funktionen, die sie dem Benutzer und seinen Applikationen anbieten, im Kernel enthalten sind, oder sich außerhalb des Kernels als Dienste (Server) befinden.



# Aufgabe 4 (Fortsetzung)

- **Monolithischer Kernel** haben im einfachsten Fall keine geordnete Struktur. Sie bestehen aus Funktionen, die sich beliebig gegenseitig aufrufen und beliebig auf interne Daten zugreifen können. Höhere Geschwindigkeit und hoher Entwicklungsaufwand für Erweiterungen.
- In **Mikrokerneln** sind nur die notwendigsten Funktionen zur Speicher- und Prozessverwaltung, Synchronisation und Prozesskommunikation. Die Gerätetreiber und Dienste (Server), befinden sich außerhalb des Kernels auf Benutzererebene. Schlechtere Performance. Geringer Entwicklungsaufwand für Erweiterungen.
- **Hybridkernel** sind ein Kompromiss zwischen monolithischen Kerneln und Mikrokerneln. Sie basieren auf dem Konzept der Mikrokernel, enthalten aber aus Geschwindigkeitsgründen einige Komponenten, die bei Mikrokerneln außerhalb des Kernels liegen. Höhere Geschwindigkeit als Mikrokernel. Höhere Stabilität als monolithischen Kernel.

# Aufgabe 5

Was versteht man unter einem Betriebssystemaufruf (System-Call)?

- Ein Systemaufruf ist ein Funktionsaufruf im Betriebssystem, der einen Sprung vom **User Mode** (Benutzermodus) in den priviligierten **Kernel Mode** (Kernel-Modus) auslöst ($\Longrightarrow$ Moduswechsel).
- Systemaufrufe stellen für die Benutzer-Prozesse die einzige Schicht zum Zugriff auf die Betriebssystemfunktionalität, also die Benutzung der Hardware eines Computer-Systems dar.
- Die Systemaufrufe erlauben es den Benutzerprogrammen, Prozesse, Dateien und andere Betriebsmittel zu erzeugen und zu verwalten.

# Aufgabe 6

Warum unterscheiden moderne Betriebssysteme zwischen **Benutzermodus** (User Mode) und **Kernel-Modus** (Kernel Mode)? Wäre es nicht besser, nur einen Modus zu haben?

- Die Trennung von Benutzermodus und priviligiertem Kernel-Modus hat Sicherheitsgründe.
- Komplexe und tendentiell anfällige Software, die mit den Rechten des priviligierten Kernel-Modus läuft, wäre ein großes Sicherheitsrisiko.

# Aufgabe 7

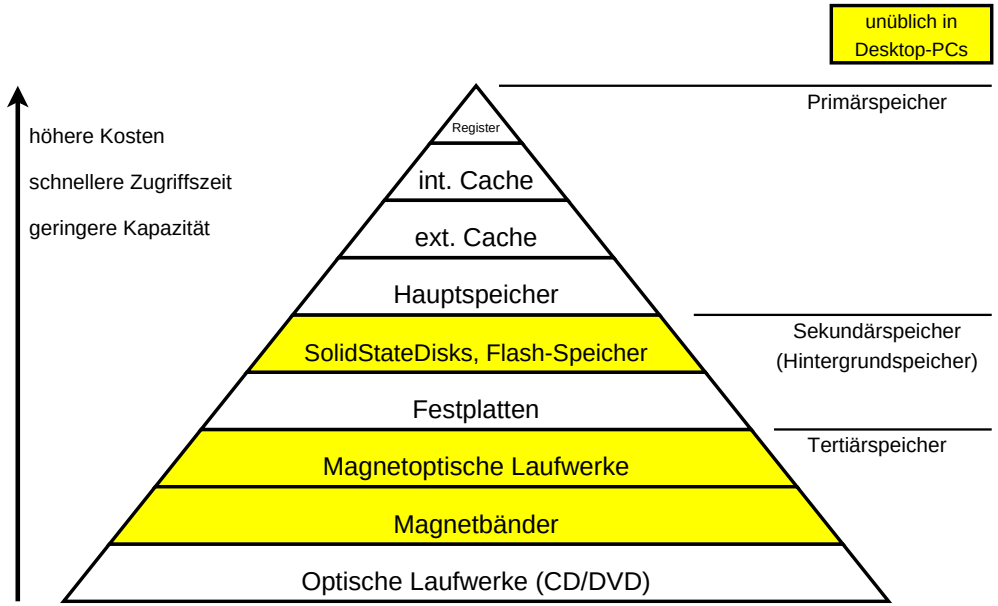
Beschreiben Sie die Funktionsweise eines Universalrechners mit einer Von-Neumann-Architektur und was ist die Aufgabe des Speichers in der Von-Neumann-Architektur?

- Ein von-Neumann-Rechner enthält die folgenden Komponenten:
  - Rechenwerk (Arithmetic Logic Unit): Führt Rechenoperationen und logische Verknüpfungen durch.
  - Steuerwerk: Interpretiert die Programmanweisungen und steuert die Befehlsabfolge.
  - Speicher: Daten und Programme liegen binär codiert im selben Speicher und sind für das Rechenwerk zugänglich.
  - Ein-/Ausgabesteuerung
  - Bus: Über diesen sind die Systemkomponenten miteinander verbunden. Prozessor und Speicher kommunizieren hier direkt miteinander. Befehle und Daten müssen über diesen Bus transportiert werden.

# Aufgabe 8

Warum macht es Sinn, den Speicher in einer Speicherpyramide abzubilden?
Was ist der Grund für die Speicher-Hierarchie?

- Der Grund für die Speicher-Hierarchie liegt im Preis/Leistungsverhältnis.
- Je schneller ein Speicher ist, desto teurer und knapper ist er.



# Aufgabe 9

Welche beiden Ergebnisse sind bei einer Daten-Anfrage an den Cache möglich? Nennen Sie diese beiden möglichen Ergebnisse und erklären Sie diese mit jeweils einem Satz.

- **Cache-Hit**: Die am Cache angefragten Daten sind vorhanden (Treffer).
- **Cache-Miss**: Die am Cache angefragten Daten sind nicht vorhanden (verfehlt).

# Aufgabe 10

Mit welchen beiden Kennzahlen kann die Effizienz eines Caches bewertet werden?

- **Hitrate**: Anzahl der Anfragen an den Cache mit Ergebnis Cache-Hit, geteilt durch die Gesamtanzahl der Anfragen. Das Ergebnis liegt zwischen Null und Eins. Je höher der Wert, desto höher ist die Effizienz des Caches.
- **Missrate**: Anzahl der Anfragen an den Cache mit Ergebnis Cache-Miss, geteilt durch die Gesamtanzahl der Anfragen.

# Aufgabe 11

Nennen Sie fünf Ersetzungsstrategien für die Cache-Datenverwaltung.

- **OPT** (Optimale Strategie)
- **LRU** (Least Recently Used)
- **LFU** (Least Frequently Used)
- **FIFO** (First In First Out)
- **TTL** (Time To Live)
- **Random**
- **WS** (Working Set)
- **Climb**

# Aufgabe 12

Erklären Sie die Unterschiede von **Least Recently Used** (LRU) und **Least Frequently Used** (LFU).

- **LRU**: Es wird immer der Datenblock aus dem Cache verdrängt, auf den am längsten nicht mehr zugegriffen wurde. Einfach zu implementieren, liefert sehr gute Resultate und verursacht wenig Overhead. Es wird nicht berücksichtigt, wie oft auf einen Datenblock zugegriffen wurde.
- **LFU**: Es wird der Datenblock wird aus dem Cache verdrängt, auf den am wenigsten zugegriffen wurde. Relativ einfach zu implementieren und berücksichtigt die Zugriffshäufigkeit eines Datenblocks. Datenblöcke, auf die in der Vergangenheit häufig zugegriffen wurde, können den Cache blockieren.

# Aufgabe 13

Was ist die Kernaussage der Anomalie von Laszlo Belady?

- Laszlo Belady zeigte in seiner Anomalie, dass unter sehr ungünstigen Umständen FIFO bei einem größeren Speicher sogar zu mehr Zugriffsfehlern (Miss) führt, als bei einem kleinen Speicher.
- Ursprünglich ging man davon aus, dass eine Vergößerung des Speichers immer zu weniger oder schlechtestenfalls gleich vielen Zugriffsfehlern führt.
- Bis zur Entdeckung von Belady's Anomalie hielt man FIFO für eine gute Ersetzungsstrategie.

# Aufgabe 14

Was sind **Write-Back** und **Write-Through**? Was sind die Unterschiede, Vor- und Nachteile?

- **Write-Back**: Schreibzugriffe werden nicht direkt an die nächsthöhere Speicherebene weitergegeben. Inkonsistenzen zwischen den Daten im Cache und auf dem zu cachenden Speicher entstehen. Die Daten werden erst zurückgeschrieben, wenn der betreffende Datenblock aus dem Cache verdrängt wird.
  $\Longrightarrow$ Höhere System-Geschwindigkeit, Daten gehen beim Systemausfall verloren.

- **Write-Through**: Schreibzugriffe werden sofort an die nächsthöhere Speicherebene weitergegeben.
  $\Longrightarrow$ Datenkonsistenz ist gesichert. Geringere System-Geschwindigkeit.

# Aufgabe 15

Nennen Sie die drei Möglichkeiten, die es gibt, damit eine Anwendung Daten von Ein- und Ausgabegeräten lesen kann. Was sind die Unterschiede, Vor- und Nachteile?

- **Busy Waiting**: Der Prozess sendet die Anfrage an das Gerät und wartet in einer Endlosschleife, bis die Daten bereit stehen. Leicht zu implementieren. Belastet den Prozessor und behindert die gleichzeitige Abarbeitung mehrer Programme.
- **Interrupt-gesteuert**: Der Prozess initialisiert die Aufgabe und wartet auf einen Interrupt (Unterbrechung) durch den notwendigen Interrupt-Controller. Die CPU ist während des Wartens nicht blockiert. Zusätzliche Hardware notwendig.
- **Direct Memory Access**: Ein zusätzlicher DMA-Baustein überträgt die Daten direkt zwischen Speicher und Controller ohne Mithilfe der CPU. Vollständige Entlastung der CPU. Hoher Hardware-Aufwand

# Aufgabe 16

Welche zwei Gruppen von Ein- und Ausgabegeräten gibt es bezüglich der kleinsten Übertragungseinheit. Was charakterisiert jede der beiden Gruppen? Nennen Sie für jede Gruppe zwei Geräte-Beispiele.

- **Zeichenorientierte Geräte**: Bei der Ankunft/Anforderung jedes einzelnes Zeichens wird immer mit dem Prozessor kommuniziert.
  $\Longrightarrow$ Maus, Tastatur, Drucker, Terminals, Magnetbänder, ...
- **Blockorientierte Geräte**: Die Datenübertragung wird erst bei Vorliegen eines kompletten Blocks (z.B. 1-4 kB) angestoßen.
  $\Longrightarrow$ Festplatten, CD-/DVD-Laufwerke, Disketten-Laufwerke, ...

# Aufgabe 17

Was halten Sie davon, dass Programme direkt auf Speicherstellen zugreifen? Ist das eine gute Idee? Begründen Sie ihre Antwort.

- Prozesse verwenden keine reale Hauptspeicheradressen und können daher nicht unmittelbar auf die Speicherstellen (Bytes) ihres Adressraums im Speichers zugreifen.
- Würden Prozesse direkt auf die Speicherstellen zugreifen, würde dies in einem Multitasking-System zu großen Problemen wegen der fehlenden Datensicherheit führen.
- Besser als der direkte Zugriff auf die Speicherstellen ist eine Abstraktion der Prozesse von den verwendeten Speichertechnologien und ihren gegebenen Ausbaumöglichkeiten.

# Aufgabe 18

Was ist der Adressraum eines Prozesses?

- Der **Adressraum** ist eine Abstraktion des physischen Speichers. Es ist der von der verwendeten Speichertechnologie und den gegebenen Ausbaumöglichkeiten unabängige **virtuelle Speicher**.
- Jeder Adressraum besteht aus Speicherstellen (Bytes), die von der Adresse 0 an, aufwärts durchnummeriert sind.
- Adressräume können nach Bedarf erzeugt oder gelöscht werden und sind voneinander abgeschottet und damit geschützt.
- Ein Prozess, der in seinem Adressraum arbeitet, kann nicht ohne vorherige Vereinbarung auf den Adressraum eines anderen Prozesses zugreifen.

# Aufgabe 19

Was ist virtueller Speicher? Was sind die Gründe für seine Existenz?

- Der virtuelle Speicher ist der vom tatsächlich vorhandenen Arbeitsspeicher unabhängige Adressraum, den ein Prozess für seinen Programmcode und Daten vom Betriebssystem zur Verfügung gestellt bekommt.
- Dank virtuellem Speicher wird der Hauptspeicher besser ausgenutzt, denn die Prozesse müssen nicht am Stück im Hauptspeicher liegen. Aus diesem Grund ist die Fragmentierung des Hauptspeichers kein Problem.
- Durch den virtuellen Speicher kann auch viel mehr Speicher angesprochen und verwendet werden als physisch im System vorhanden ist.

# Aufgabe 20

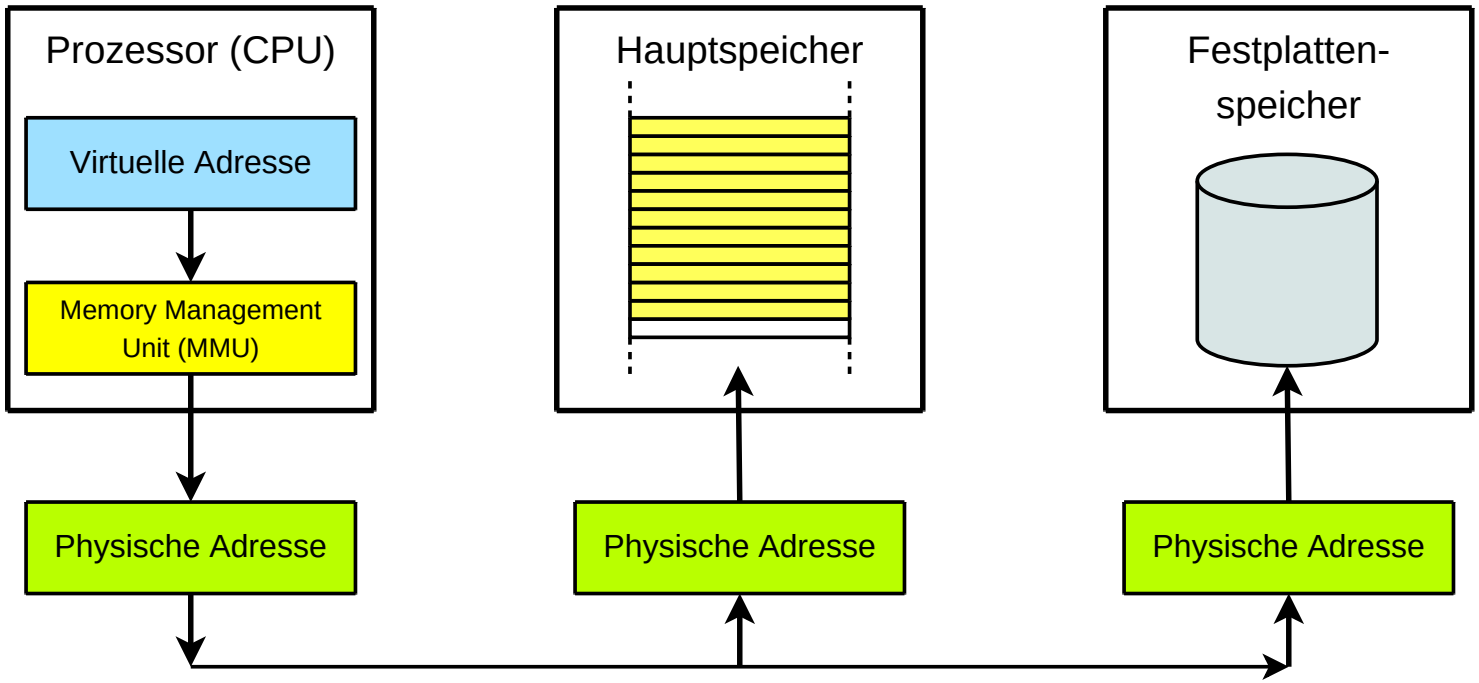
Was versteht man beim virtuellen Speicher unter Seiten (Pages), Rahmen (Frames) und dem Vorgang des Mappings?

- Der virtuelle Speicher ist genau wie der physische, reale Speicher in Blöcke gleicher Größe aufgeteilt. Diese Blöcke bezeichnet man beim virtuellen Speicher als **Seiten** und beim Hauptspeicher als **Rahmen**.
- Die Seiten des virtuellen Speichers werden auf die Rahmen im Hauptspeicher abgebildet. Dieser Vorgang wird als **Mapping** bezeichnet.

# Aufgabe 21

Was ist die Memory Management Unit (MMU) und was ist ihre Aufgabe?

- Memory Management Unit (MMU) in der CPU rechnet die virtuellen Adressen in reale Speicheradressen um.



# Aufgabe 22

Nennen Sie die beiden unterschiedlichen Konzepte von virtuellem Speicher und erklären Sie in wenigen Sätzen die Unterschiede, Vor- und Nachteile.

- Beim **Segmentorientierten Speicher** besteht der virtuelle Speicher eines Prozesses aus vielen Einheiten unterschiedlicher Länge. Die Aufteilung des virtuellen Speichers erfolgt nach logischen Gesichtspunkten. Einzelne Daten-, Code- oder Stackelemente verbleiben in einer virtuellen Speichereinheit. Durch die Segmente unterschiedlicher Länge kommt es zu einem internen und externen Verschnitt und damit zu Speicherverschwendung.
- Beim **Seitenorientierten Speicher** haben alle virtuellen Speichereinheiten die gleiche Größe, was die Speicherverwaltung vereinfacht. Durch die Segmente gleiche Länge kommt es nur zu einem internen Verschnitt.

# Aufgabe 23

Definieren Sie den Begriff des Prozesses.

- Ein **Prozess** ist ein Programm, dass sich in Ausführung bzw. Bearbeitung befindet.
- Prozesse sind dynamische Objekte und repräsentieren sequentielle Aktivitäten in einem Computersystem.
- Ein Prozess umfasst außer dem Programmcode noch den **Prozesskontext** und einen geschützten **Prozessadressraum**.

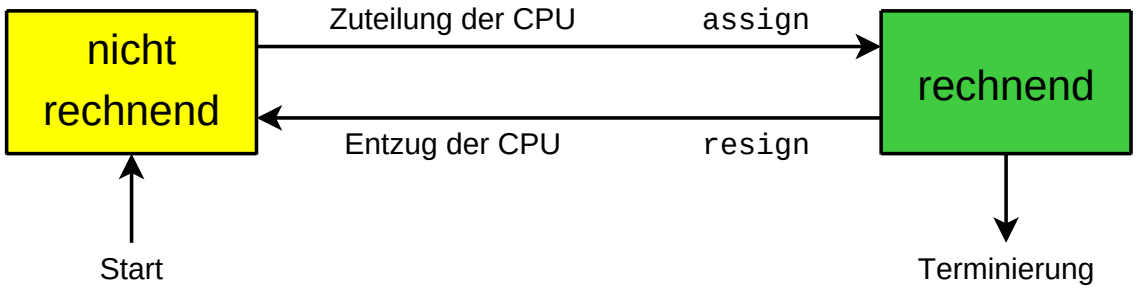
# Aufgabe 24

Nennen Sie die drei Arten von Kontextinformation, die das Betriebssystem speichert, und beschreiben Sie in wenigen Sätzen, welche Informationen darin enthalten sind.

- **Benutzer-Kontext**: Daten des Prozesses im zugewiesenen Adressraum.
- **Hardware-Kontext**: Inhalte der Register in der CPU zum Zeitpunkt der Prozess-Ausführung und die Seitentabelle. Beispiele sind Befehlszähler, Stack-Pointer, Integer-Register und Floating-Point-Register.
- **System-Kontext**: Informationen, die das Betriebssystem über einen Prozess speichert. Beispiele sind Prozessnummer (PID), Prozesszustand, PPID, Prioritäten, Laufzeit und geöffnete Dateien.

# Aufgabe 25

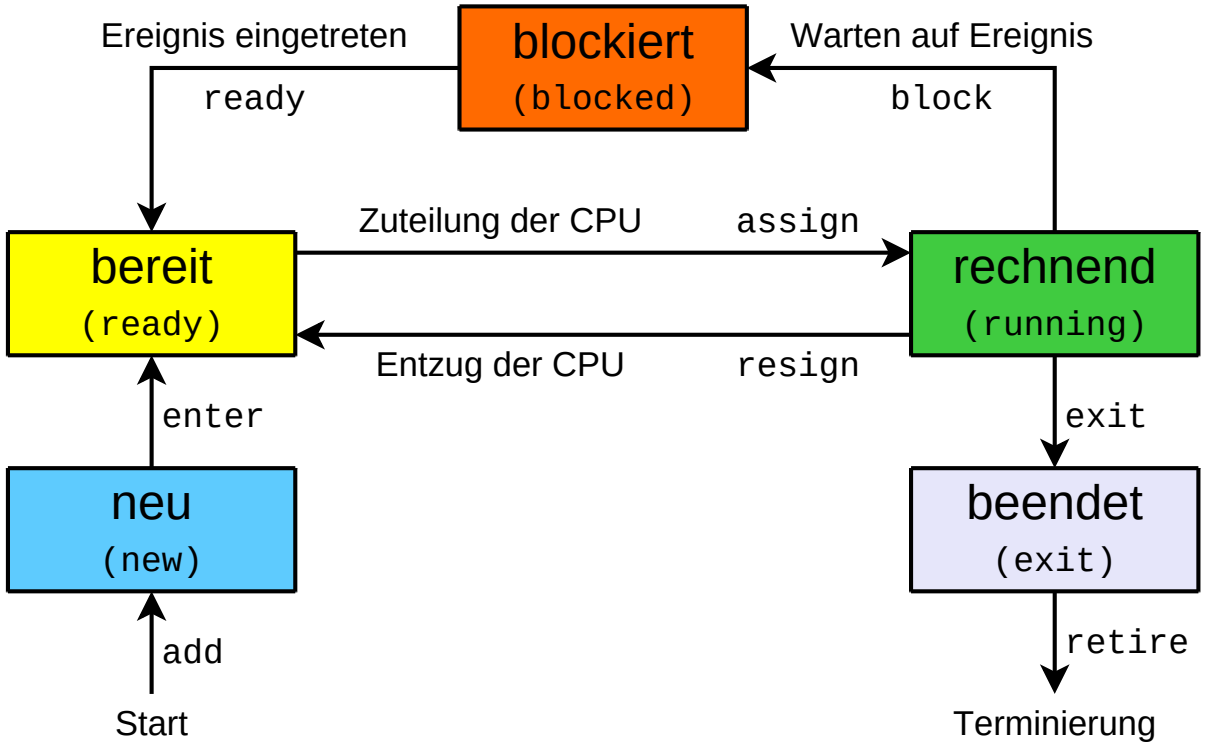
Das kleinste, denkbare Prozessmodell ist das 2-Zustands-Prozessmodell. Welche Zustände und Prozessübergänge enthält dieses Prozessmodell? Zeichnen Sie das 2-Zustands-Prozessmodell. Ist dieses Prozessmodell sinnvoll?




- Das 2-Zustands-Prozessmodell geht davon aus, dass alle Prozesse immer zur Ausführung bereit sind. Das ist aber unrealistisch.
- Es gibt fast immer Prozesse, die blockiert sind und z.B. auf das Ergebnis eines E/A-Geräts oder eines anderen Prozesses warten.

# Aufgabe 26

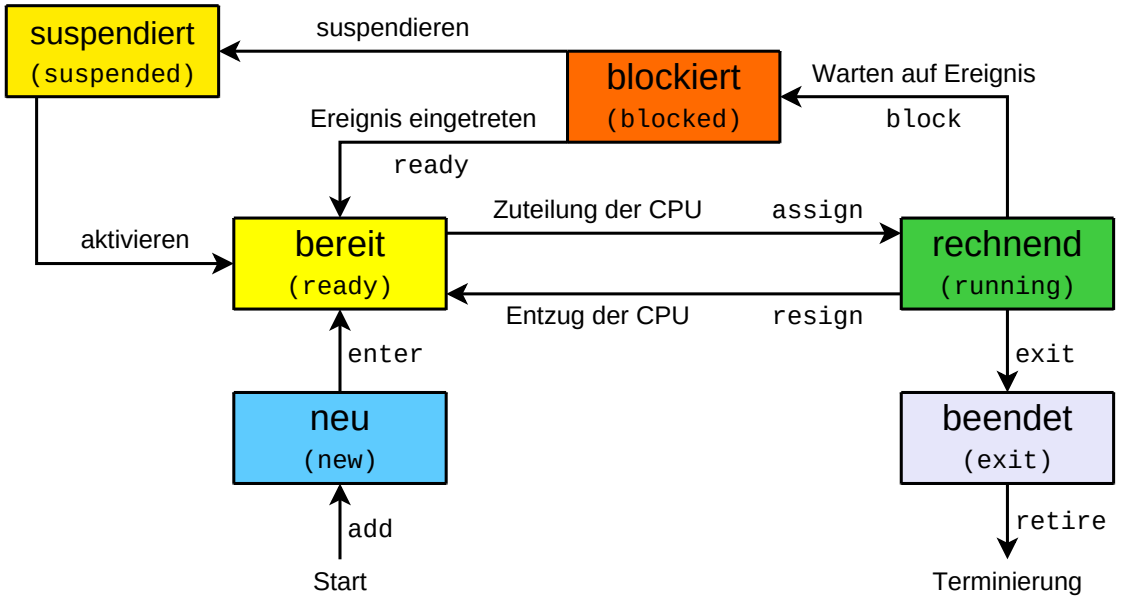
Zeichnen Sie das 5-Zustands-Prozessmodell mit den Zuständen **neu**, **bereit**, **blockiert**, **rechnend** und **beendet** mit seinen Prozessübergängen.



# Aufgabe 27

Um welchen Zustand kann das 5-Zustands-Prozessmodell sinnvoll erweitert werden?

- Ist nicht genügend (realer) Hauptspeicher für alle laufenden Prozesse verfügbar, werden Prozesse ausgelagert bzw. **suspendiert**.



# Aufgabe 28

Was sind Unterbrechungen und warum sind diese notwendig?

- Unterbrechungen sind Ereignisse, deren Behandlung keinen Aufschub zulässt.
- Beispiele sind Fehler in einer Rechenoperation und Rückmeldungen von Ein-/Ausgabe-Geräten.
- Unterbrechungen sind notwendig, weil häufig unvorhersehbare (deterministische) Ereignisse eintreten, auf die ein Computer-System reagieren muss.

# Aufgabe 29

Nennen Sie drei häufige Gründe für Unterbrechungen.

- **Fehlersituation**: Ein Fehler bei einer Rechenoperation, z.B. Division durch Null, Gleitkommafahler, Adressfehler, usw.
- **Software-Interrupt**: wird durch einen Prozess ausgelöst. Beispiele sind die `TRAP`-Funktion, um vom normalen Benutzermodus in den priviligierten Kernel-Modus zu wechseln und der Einzelschrittbetrieb beim Programmtest (Debugging, Trace).
- **Hardware-Interrupt**: Ein-/Ausgabe-Geräte liefern Rückmeldungen an einen Prozess oder das Auftreten eines Stromausfalls.

# Aufgabe 30

Was sind die Unterschiede zwischen **Interrupts** und **Exceptions**?

- Interrupts sind externe Unterberechungen. Sie werden durch Ereignisse außerhalb des zu unterbrechenden Prozesses ausgelöst. Ein Beispile ist, dass ein Ein-/Ausgabe-Gerät das Ende eines E/A-Prozesses meldet.
- Exceptions sind interne Unterbrechungen oder Ausnahmen/Alarme und werden vom Prozess selbst ausgelöst.

# Aufgabe 31

Was ist ein Thread und was sind die Unterschiede zwischen **Prozessen** und **Threads**?

- Ein **Thread** ist ein leichtgewichtiger Prozess und eine Aktivität (Programmausführung) innerhalb eines Prozesses.
- Es können mehrere nebenläufige Programmausführungen im gleichen Kontext aktiv sein und ihre Daten gemeinsam nutzen.
- Durch Threads kann ein Programm mehrfach an unterschiedlichen Stellen ausgeführt werden.
- Alle Threads eines Programms arbeiten in dem gleichen Adressraum und besitzen die gleichen Betriebsmittel. Aus diesem Grund können sie direkt miteinander kommunizieren und zusammenarbeiten. Prozesse können dies nicht.

# Aufgabe 32

Was sind die Unterschiede, Vor- und Nachteile zwischen **Kernel-Level-Threads** und **User-Level-Threads**?

- Bei Kernel-Level-Threads wird das Scheduling des Betriebssystems verwendet. Die Threads eines Prozesses können auf mehreren Prozessoren verteilt laufen. Bei User-Level-Threads geht das nicht.
- Ein Kernel-Level-Thread der blockiert, blockiert nur sich selbst. Ein User-Level-Thread der blockiert, blockiert den gesamten Prozess.
- Bei Kernel-Level-Threads ist jede Threadoperation ein Systemaufruf. User-Level-Threads sind effizienter.
- Mit User-Level-Threads können Threads auch auf Betriebssystemen ohne Thread-Unterstützung verwendet werden.
- Es kann immer nur ein User-Level-Thread eines Prozesses rechnen, da der Kernel nur den Prozess, aber nicht die User-Level-Threads kennt.

# Aufgabe 33:

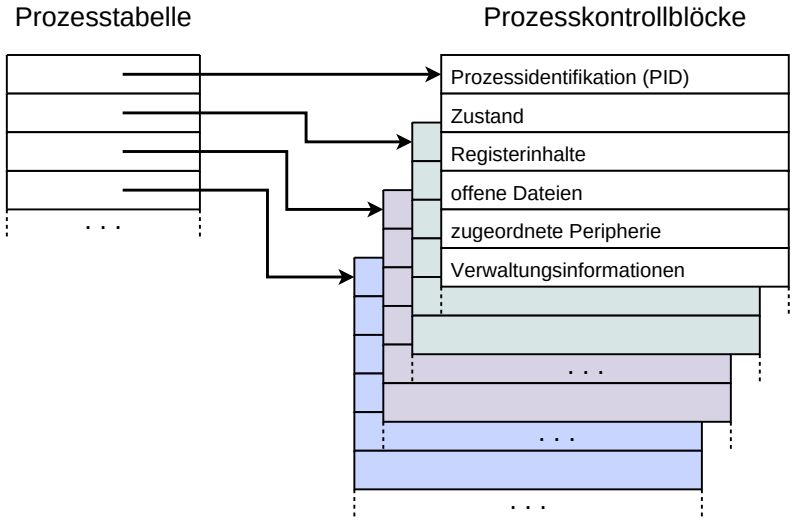
Nennen Sie ein Beispiel für den sinnvollen Einsatz von Threads.

- Ein **E-Mail-Programm** besteht aus vier Threads:
  - **Thread 1**: Interaktion mit dem Benutzer.
  - **Thread 2**: Formatierung der Programmoberfläche.
  - **Thread 3**: Automatisches Holen neuer E-Mails alle 5 Minuten.
  - **Thread 4**: Spam-Filter auf neue E-Mails anwenden.

# Aufgabe 34

Was ist die Prozesstabelle?

- In der **Prozesstabelle** werden die Informationen zu allen Prozessen des Systems gesammelt.
- In der Prozesstabelle, die als Array oder verkettete Liste realisiert ist, sind die Prozesskontrollblöcke zusammengefasst.



# Aufgabe 35

Was ist ein Prozesskontrollblock und wie viele Prozesskontrollblöcke gibt es?

- Für jeden Prozess existiert ein **Prozesskontrollblock**, der außer dem Inhalt des Adressraums alle Informationen bezüglich des Prozesses enthält.
- In dem Prozesskontrollblock eines jeden Prozesses befindet sich unter anderem die PID, der Prozesszustand, der Inhalt der Prozessorregister, eine Liste mit zugeordneten Bereichen im Hauptspeicher, geöffnete Daten und Verwaltungs-/Schedulinginformationen.
- Im Prozesskontrollblock werden die Informationen gespeichert, die gebraucht werden, wenn der Prozess von einem Zustand in einen anderen übergeht.

# Aufgabe 36

Warum führt das Betriebssystem Zustandslisten und welche Zustandslisten gibt es?

- Das Betriebssystem führt verkettete Listen für die Prozesse mit den Zuständen `bereit` und `blockiert`.
- Die Liste der Prozesse mit dem Zustand `bereit` enthält alle Prozesse, die unmittelbar ausgeführt werden können, aber auf die Zuteilung des Prozessors warten.
- Die Prozesse in der `bereit`-Liste können nach unterschiedlichen Kriterien, u.a. nach den Prozessprioritäten oder der Wartezeit, sortiert werden.
- Für einen Zustandsübergang eines Prozesses wird der Prozesskontrollblock des betreffenden Prozesses aus der alten Zustandsliste entfernt und in die neue Zustandsliste eingefügt.

# Aufgabe 37

Gibt es auch eine Zustandsliste für Prozesse mit dem Zustand `rechnend`?

- Für die Prozesse mit dem Zustand `rechnend` gibt es keine eigene Zustandsliste. Es gibt aber eine Variable des Betriebssystems, die auf den Prozesskontrollblock des aktuell rechnenden Prozesses zeigt.
- Beim Übergang eines Prozesses in den Zustand `rechnend` werden die Registerinhalte des Prozessors aus dem Prozesskontrollblock geladen.
- Beim Übergang aus dem Zustand `rechnend` in einen anderen Zustand werden die Registerinhalte des Prozessors in den Prozesskontrollblock gesichert (gerettet).

# Aufgabe 38

Welche Schritte werden bei der Erzeugung eines Prozesses vom Betriebssystem unternommen?

- Einen neuen Prozess **erzeugen**:
  1. Freie Prozessidentifikation (PID) belegen.
  2. Neuen Prozesskontrollblock anlegen.
  3. Felder des Prozesskontrollblocks mit Anfangswerten füllen.
  4. Die notwendigen Speicherbereiche reservieren und initialisieren.
  5. Den Prozess in die Prozesstabelle und den neuen Prozesskontrollblock in die Liste der Prozesse im Zustand `blockiert` einfügen.

# Aufgabe 39

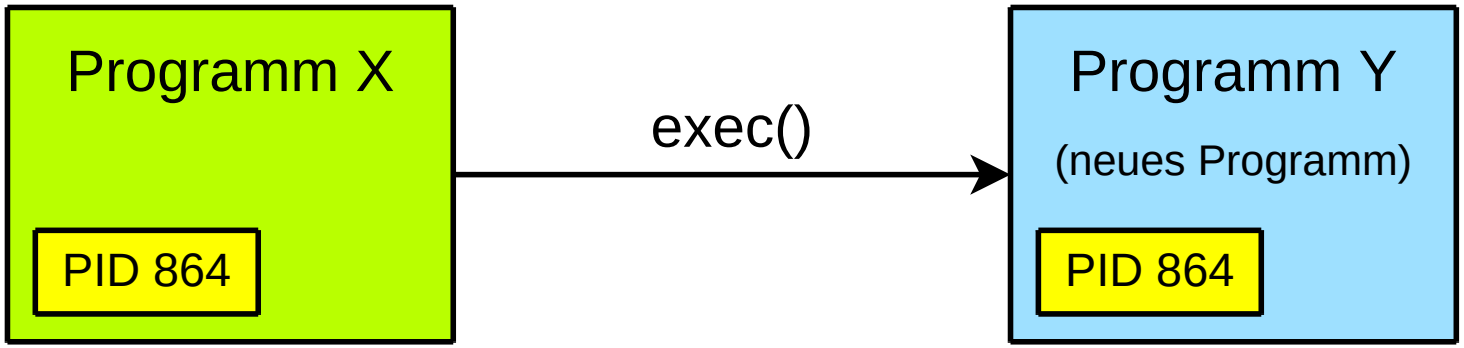
Mit welchem Systemaufruf kann unter Linux/UNIX Betriebssystemen ein neuer Prozess erzeugt werden. Was macht dieser Systemaufruf im Detail?

- Unter Linux/UNIX Betriebssystemen ist der Systemaufruf `fork()` die einzige Möglichkeit, einen neuen Prozess zu erzeugen.
- Ruft ein Prozess, der Elternprozess, `fork()` auf, wird eine identische Kopie als neuer Prozess (Kindprozess) gestartet.
- Nach der Erzeugung eines neuen Prozesses mit `fork()` hat der Kindprozess den gleichen Programmcode und die Befehlszähler haben den gleichen Wert, verweisen also auf die gleiche Stelle im Programmcode.
- Geöffnete Dateien und Speicherbereiche des Elternprozesses werden für den Kindprozess kopiert und stehen ihm nun getrennt zur Verfügung.
- Beide besitzen ihren eigenen Prozesskontext und eine eigene PID.

# Aufgabe 40

Was macht der Systemaufruf `exec()`?

- Mit dem Systemaufruf `exec()` wird ein Prozess durch einen anderen ersetzt. Es findet eine Verkettung statt.
- Bei `exec()` wird der aufrufende Prozess beendet und ein neuer gestartet. Dieser neue Prozess erbt sogar die Prozess-ID (PID) des aufrufenden Prozesses.



# Aufgabe 41

Was sind die Unterschiede zwischen den Systemaufrufen `fork()` und `exec()`?

- Im Gegensatz zu `fork()`, wo vom aufrufenden Prozess eine identische Kopie als neuer Prozess erzeugt wird, wird bei `exec()` der aufrufende Prozess beendet und ein neuer gestartet. Dieser neue Prozess erbt sogar die Prozess-ID (PID) des aufrufenden Prozesses.
- Wird vor einem Aufruf von `exec()` kein neuer Prozess mit `fork()` erzeugt, geht der Elternprozess verloren.

Nächste Vorlesung:

# 3.5.2007