Systemprogrammierung

Grundlage von Betriebssystemen

Sachwortverzeichnis

Wolfgang Schröder-Preikschat

7. Juni 2016

Die in der Vorlesung (mündlich oder schriftlich) verwendeten Akronyme und Sachworte sind in der nachfolgenden Aufzählung zusammengefasst. Dabei werden die Akronyme nach den Sachworten, für die sie die Verkürzung bilden, aufgeschlüsselt. Für englischsprachige Sachwörter werden, soweit bekannt, die deutschsprachigen Entsprechungen angegeben. In der Beschreibung ↑*hervorgehobene* Sachwörter zeigen einen Kreuzverweis an. Alle Sachworte werden erklärt, wobei dies im Zusammenhang mit dem hier relevanten Kontext der Systemprogrammierung und in Bezug auf Betriebssysteme geschieht. Die Formulierungen erheben nicht den Anspruch auf Gültigkeit auch für andere Fachrichtungen in der Informatik. Ebenso erhebt die Aufzählung nicht den Anspruch auf Vollständigkeit für das in der Vorlesung behandelte Fachgebiet.

ABB *atomic basic block*, (dt.) ↑*unteilbarer Grundblock*.

Abfangung Konstruktion, die einen gegenwärtigen ↑*Prozess* gegen unkontrollierten Absturz sichert (in Anlehnung an das ↑*Bauwesen*). Die Fangstelle (↑*trap*) für eine ↑*Aktion*, bei der eine ↑*Ausnahmesituation* eingetreten ist. Der Prozess wird bei dieser Aktion, die er selbst verursacht, abgefangen: an der Fangstelle wird eine ↑*synchrone Ausnahme* erhoben.

Ablaufinvarianz Merkmal von einem ↑*Programm*, mehr als einen ↑*Handlungsstrang* gleichzeitig zuzulassen, ohne sich in den daraus resultierenden zeitlich überlappenden Abläufen gegenseitig beeinflussen zu können. Eine solche Überlappung von Abläufen zeigt sich beispielsweise bei der ↑*Ausnahmebehandlung*, bei der nämlich der normale Ablauf eines Programms überlappt wird von dem unnormalen Ablauf durch einen ↑*Ausnahmehandhaber*. Variante davon ist das Programm zur ↑*Unterbrechungsbehandlung*, um im Betriebssystem auf eine ↑*Unterbrechungsanforderung* der Hardware reagieren zu können. Ähnliche Abläufe resultieren aus der Möglichkeit zur ↑*Parallelverarbeitung* ein und desselben Programms. Damit diese frei von gegenseitiger Beeinflussung sind, muss für den betreffenden Abschnitt ↑*Eintrittsinvarianz* gelten: kein Zugriff auf statische oder globale Variablen, keine Veränderung seines Textanteils und kein Aufruf eines nicht eintrittsinvarianten ↑*Unterprogramms*.

Ablaufplan Ergebnis der ↑*Ablaufplanung*.

Ablaufplanung Verfahren, das die Zuteilung des Prozessors an einen ↑*Prozess* oder eine Gruppe von Prozessen regelt. Das Verfahren arbeitet rechnerunabhängig (*off-line*) oder mitlaufend (*on-line*) zu den Prozessen, es liefert dementsprechend einen statischen oder dynamischen ↑*Ablaufplan*, der zur ↑*Laufzeit* der Prozesse unverändert befolgt wird oder aktuellen Bedürf- und Geschehnissen angepasst werden kann. Grundlage bildet ein auf das jeweilige Verfahren zugeschnittener ↑*Einplanungsalgorithmus*.

Ablaufsteuerung Steuerung von einem ↑*Programmablauf* mithilfe aufeinanderfolgender Befehle oder Bedingungen (Duden).

Abruf- und Ausführungszyklus Hauptschleife von einem ↑*Interpreter*: (1) den nächsten Befehl aus dem ↑*Programm* lesen, (2) ihn dekodieren, gegebenenfalls in Einzeloperationen zerlegen, alle benötigten Operanden entsprechend der jeweils spezifizierten ↑*Adressierungsart* laden und (3) die Operation/en durchführen. Schließlich (4) die Abfrage, ob während dieser Befehlsausführung eine ↑*Ausnahme* erhoben wurde und gegebenenfalls den ↑*Unterbrechungszyklus* einleiten. Führt eine ↑*Aktion* herbei.

abstrakte Maschine Maschine, die nur gedanklich und nicht physisch existiert. Sie ist entweder ein als Modell geschaffenes Konstrukt, um komplexe Vorgänge innerhalb von Rechensystemen auf theoretischer Ebene zu untersuchen (Automat, Turing-Maschine) oder ein über eine wohldefinierte Schnittstelle zugängliches Softwaregebilde, das allein oder im Ensemble mit anderen Maschinen ihrer Art hilft, eine ↑*semantische Lücke* zu schließen (↑*Betriebssystem*). Letztere Variante meint ein ↑*Programm*, das zwar ein physisches ↑*Betriebsmittel* in Form von ↑*Speicher* belegt, als Software jedoch selbst ein „nicht technisch-physikalischer Funktionsbestandteil“ (Duden) darstellt.

Adressbereich Abschnitt von linearen Adressen in einem ↑*Adressraum*.

Adressbreite Bitanzahl, die zur Darstellung einer ↑*Adresse* im ↑*Dualsystem* zur Verfügung steht. Typisch waren oder sind 16, 18, 20, 24, 32, 48 und 64 Bits pro Adresse (Stand 2016).

Adresse Nummer einer ↑*Speicherstelle*.

Adressierungsart Lokalisierung der/des Operanden von einen ↑*Maschinenbefehl*. Gängig sind: Registeradressierung, der Befehl enthält den Namen des dem Operanden zugeordneten Prozessorregisters; Direktwertadressierung, der Befehl enthält den Operanden selbst; direkte Adressierung, der Befehl enthält die Speicheradresse des Operanden; indirekte Adressierung, der Befehl enthält den Namen des die Speicheradresse des Operanden führenden Prozessorregisters; relative Adressierung, der Befehl enthält den auf eine Basisadresse bezogenen Abstand zum Operanden; indizierte Adressierung, der Befehl enthält den Namen des den zu einer Basisadresse bezogenen Abstand zum Operanden führenden Prozessorregisters.

Adressraum Menge von nichtnegativen ganzen Zahlen, endlich, wobei jedes Element darin eine ↑*Adresse* repräsentiert. Die Kardinalität dieser Menge ist bestimmt durch die ↑*Adressbreite* der (realen/virtuellen) Maschine.

Adresszähler Zeiger in ein in ↑*Assembliersprache* formuliertes Programm, über den die einzelnen Programmanweisungen ihre jeweilige Speicheradresse erhalten. Jeder Programmabschnitt (Text, Daten, ↑*BSS*) besitzt einen eigenen, mit 0 initialisierten Zeiger. Bei der ↑*Assemblierung* einer Programmanweisung wird ihr der aktuelle Zeigerwert als Speicheradresse zugeordnet und der Zeiger wird um die Länge (in Bytes) dieser Anweisung erhöht. So erhält etwa der Name der Marke (d.h., ein ↑*Symbol*) einer Assembliersprachenanweisung einen Wert, der die ↑*Adresse* der Anweisung relativ zum Abschnittsbeginn repräsentiert. Der Zeiger kann als Operand in einer Assembliersprachenanweisung verwendet werden. Typisch ist das Dollarzeichen (`$`): einem Symbol vorangestellt liefert es dessen Adresswert als Operand.

Aktion Ausführung der Anweisung einer Maschine durch einen ↑*Prozessor*. Eine Anweisung, die eine Einzelaktion auf höherer Ebene beschreibt, kann als ↑*Aktionsfolge* auf tieferer Ebene stattfinden, beispielsweise: `ADD X,Y` ↦ `LOAD X; ADD Y; STORE X`.

Aktionsfolge Reihe von nacheinander oder gleichzeitig stattfindenden ↑*Aktionen*. Dabei können die Aktionen gänzlich oder in Teilen auch demselben ↑*Handlungsstrang* zugehören.

allgemeiner Semaphor ↑*Semaphor*, bei dem die ↑*Ablaufsteuerung* eine *Ganzzahlvariable* benutzt, der einen ganzzahligen elementaren Datentyp (`int`) besitzt. Die Operationen ↑P und ↑V sind zählende Operationen, daher auch *zählender Semaphor*. Zustandsänderungen in Bezug auf die enthaltene Ganzzahlvariable bilden jeweils eine ↑*Elementaroperation*, die für gewöhnlich aber nicht als ↑*atomare Operation* vorliegt: *Subtraktion* beziehungsweise *Addition*, um 1. Die Ablaufsteuerung lässt einen ↑*Prozess* in P blockieren, wenn der Variablenwert vor der Subtraktion 0 ist. Anderenfalls lässt P den Prozess voranschreiten. Die Subtraktion kann bedingt oder unbedingt erfolgen. Im ersten Fall entspricht der Semaphor einer nichtnegativen ganzen Zahl, da ein in die Blockierung geleiteter Prozess den Wert nicht weiter dekrementiert. Allein anhand des Zahlenwerts ist dann nicht feststellbar, ob ein Prozess in P blockiert und durch V gegebenenfalls zu deblockieren ist. Dies ist im zweiten Fall (ganze Zahl) möglich, da jeder blockierende Prozess den Zahlenwert weiter in den negativen Bereich bringt. Der Absolutwert liefert dann die Anzahl der in P blockierten Prozesse. Damit eröffnet sich ein größerer Spielraum für die technische Auslegung der ↑*Warteschlange*, also ob diese als Datenstruktur pro Semaphor ausgeprägt (↑*Interferenz*) oder durch den ↑*Planer* berechnet (↑*Latenzzeit*) werden soll. Im ersten Fall ist die Ausprägung als Datenstruktur üblich, da dann V darüber einfach prüfen kann, ob Prozesse zur Deblockierung anhängig sind. Typischer Anwendungsfall dieser Art von Semaphor ist die Verwaltung „zählbarer“ ↑*Betriebsmittel*, nämlich die Vergabeprozedur sowohl für ein ↑*konsumierbares* (Erzeuger-Verbraucher-Beziehung) als auch für ein ↑*wiederverwendbares Betriebsmittel* (begrenzter Puffer/Speicher; begrenzte Anzahl von Hardware-/Softwareeinheiten irgendwelcher Art).

Antwortzeit Zeitspanne zwischen Absetzen eines Auftrags durch einen ↑*Prozess* und der Entgegennahme der Antwort daraufhin im selben Prozess. Beispielsweise die zur ↑*Laufzeit* im ↑*Maschinenprogramm* anfallende ↑*Latenzzeit* einer per ↑*Systemaufruf* abgeforderten und im ↑*Betriebssystem* durchgeführten ↑*Systemfunktion*.

Anwendungsfaden ↑*Faden* im ↑*Maschinenprogramm*, der daselbst implementiert ist und nicht erst durch einen Faden im ↑*Betriebssystemkern* entsteht. Sämtliche für solch einen Faden erforderlichen Betriebsmittel sowie für seine Verwaltung nötigen Funktionen stellt das als ↑*nichtsequentielles Programm* ausgelegte Maschinenprogramm. Dies betrifft insbesondere den ↑*Laufzeitkontext* eines Fadens und die Funktionen zur ↑*Ablaufplanung*, ↑*Einlastung* und ↑*Synchronisation*. Ein ↑*Prozesswechsel*, um innerhalb des Maschinenprogramms vom aktuellen zu einem anderen Faden umzuschalten, verläuft im lokalen ↑*Adressraum* und auch unter ↑*Speicherschutz* ohne ↑*Systemaufruf*. Jeder dieser Fäden ist vom Bautyp her ein ↑*federgewichtiger Prozess*, für den zur Steuerung und Überwachung keine kostspielige ↑*Systemfunktion* erforderlich ist und er selbst dazu auch keine benutzt. Für das ↑*Betriebssystem* ist ein solcher Faden kein ↑*Objekt erster Klasse*, das heißt, der durch den Faden konkretisierte ↑*Prozess* ist dem Betriebssystem gänzlich unbekannt. Eine vom Maschinenprogramm aus beanspruchte Systemfunktion läuft damit nicht im Namen des effektiv beanspruchenden Fadens, sondern nur im Namen derjenigen Entität ab, die im Betriebssystem das jeweils in Ausführung befindliche Maschinenprogrammen repräsentiert. Ist dies bespielsweise ein ↑*schwer-* oder ↑*leichtgewichtiger Prozess* und wird dieser dann durch die Systemfunktion blockiert (↑*Prozesszustand*), blockieren implizit alle Fäden, die unbekannterweise eben auf diese eine ↑*Prozessinkarnation* durch eine ↑*Aktion* im Maschinenprogramm abgebildet wurden.

Arbeitsspeicher ↑*Hauptspeicher*, eventuell erweitert um einen peripheren ↑*Speicher* zur Ablage von zeitweilig für die Programmausführung nicht benötigte Text-/Datenbestände. Die Erweiterung ist funktional transparent oder intransparent, das heißt, Zugriffe auf den Erweiterungsspeicher sind in den Programmen selbst nicht formuliert (↑*virtueller Speicher*) oder explizit durch spezielle Anweisungen zum Ausdruck gebracht (↑*Überlagerungsspeicher*).

Assemblierer Softwareprozessor, der ein in ↑*Assembliersprache* formuliertes Programm in ein semantisch äquivalentes Programm in ↑*Maschinensprache* umwandelt.

Assembliersprache Programmiersprache, um alle Bestandteile, aus denen sich ein ↑*Maschinenprogramm* schließlich zusammensetzt, zu versammeln (*assemblieren*). Die Sprachelemente dienen der Platzierung und Anordnung von Text und -datenbereiche in einem gemeinsamen ↑*Adressraum*, der symbolischen Darstellung von einem ↑*Maschinenbefehl*, der Vergabe von Namen für eine ↑*Adresse*, der Zuordnung von Attributen zu den Namen (für den ↑*Binder*) und der Deklaration eines solchen Namens als global sichtbar außerhalb der ↑*Übersetzungseinheit*. Eine Anweisung in dieser Sprache ist unterteilt in vier Komponenten wie folgt: `[label] mnemonic [operands] [comment]`, wobei die Angaben in eckigen Klammern optional sind. Die Einzelkomponenten sind eine Marke (*label*), die dem an dieser Stelle geltenden Adresswert (↑*Adresszähler*) einen Namen gibt; ein als Gedächtnisstütze dienendes ↑*Mnemon* (*mnemonic*), das entweder den Operationsteil von einem ↑*Maschinenbefehl* oder einen ↑*Pseudobefehl* für den ↑*Assemblierer* benennt; der ebenfalls mnemonisch formulierte Operandenteil (*operands*) des Maschinenbefehls; sowie ein Kommentar (*comment*), um die Anweisung im semantischen Zusammenhang zu erklären.

Assemblierung Vorgang der Übersetzung eines Programms durch einen Assemblierer.

atomare Operation Operation, für die ↑*Unteilbarkeit* gilt. Eine solche Operation kann weder durch eine ↑*Unterbrechung* aufgespalten werden, noch durch ↑*Parallelverarbeitung* gleichzeitig mehrfach zur Ausführung gelangen.

Aufgabe Etwas, was einem ↑*Prozess* zu tun aufgegeben ist (in Anlehnung an den Duden). Ein Auftrag, eine bestimmte Funktion zu erfüllen. Dieses Etwas kann implementiert sein als ein ↑*Unterprogramm*, also keinen eigenständigen ↑*Handlungsstrang* definieren oder, umgekehrt, sehr wohl durch einen eigenständigen Handlungsstrang realisiert sein, der dann ein oder mehrere Unterprogramme autonom und in einer bestimmten Reihenfolge ausführt. Im letzteren Fall repräsentiert solch ein Unterprogramm eine ↑*Teilaufgabe* (↑*Job*), die dann auch als kleinste Ausführungseinheit verstanden wird. Aufgabe wie auch Teilaufgabe sind Einheiten einer ↑*Ablaufplanung*, in der dann ein Prozess eben durch solch eine Einheit bestimmt ist.

Aufrufkonvention Methode, nach der ein ↑*tatsächlicher Parameter* einem ↑*Unterprogramm* übergeben wird. Ist das Unterprogramm eine Funktion, legt die Methode zusätzlich die Art und Weise der Rückgabe des Funktionswerts fest. Platzhalter für die Über- beziehungsweise Rückgabewerte sind Prozessorregister oder liegen auf dem ↑*Laufzeitstapel*, mit fester Zuordnung der Parameterposition zu einem Prozessorregister oder Festlegung der Reihenfolge beim Stapeln der Parameter (↑*cdecl*: von rechts nach links). Darüber hinaus wird bestimmt, welche Prozessorregister innerhalb des Unterprogramms frei verwendbar sind. Hierzu erfolgt eine Differenzierung zwischen ↑*flüchtige* und ↑*nichtflüchtige Register*, wobei nur die Inhalte letzterer invariante Merkmale bei der Unterprorgammausführung darstellen.

Ausnahme Sonderfall, eine Abweichung vom normalen Programmablauf. Der ↑*Prozess* wird unterbrochen und gegebenenfalls nach erfolgter ↑*Ausnahmebehandlung* wieder aufgenommen.

Ausnahmebehandlung Vorgang zur Bearbeitung eines Sonderfalls bei der Programmausführung. Ursache ist eine ↑*Ausnahmesituation* in der (realen/virtuellen) Maschine, auf der ein ↑*Programm* abläuft: Die in diesem Programm direkt/indirekt kodierte ↑*Aktion* oder ↑*Aktionsfolge*, beispielsweise ein Befehl der ↑*CPU* oder eine Betriebssystemfunktion ausgelöst durch einen ↑*Systemaufruf*, kann nicht normal vollendet werden. Für die Bearbeitung eines solchen Sonderfalls ist ein Programm (↑*Ausnahmehandhaber*) vorzusehen, das auf der Maschine, deren Aktion/Aktionsfolge die Ausnahme erhebt, zur Ausführung kommt. Typisches Beispiel ist eine Routine zur ↑*Unterbrechungsbehandlung* in oder zur Signalbehandlung auf einem Betriebssystem, also ein Programm für eine reale beziehungsweise ↑*virtuelle Maschine*.

Ausnahmehandhaber ↑*Unterprogramm* zur ↑*Ausnahmebehandlung*.

Ausnahmesituation Umstand, der eine ↑*Aktion* dazu bringt, eine ↑*Ausnahme* zu erheben. Im Falle der in einem ↑*Maschinenprogramm* beschriebenen Aktion beispielsweise ein ungültiger

↑*Maschinenbefehl* beziehungsweise ↑*Systemaufruf* oder eine Schutz-/Zugriffsverletzung oder ein ↑*Seitenfehler* beim ↑*Abruf- und Ausführungszyklus*.

Bauwesen Gesamtheit dessen, was mit dem Errichten von Bauten zusammenhängt (Duden). Bei dem hier gemeinten Bau handelt es sich um das ↑*Betriebssystem*.

Bedingungssynchronisation Synonym für ↑*unilaterale Synchronisation*.

Bedingungsvariable Synchronisationsvariable, deren gespeicherter Wert unzugänglich ist für einen ↑*Prozess*; Programmierkonvention. Kommt in einem ↑*Monitor* zur Anwendung, um den gegenwärtigen Prozess auf ein ↑*Ereignis* zu synchronisieren und bei der Blockierung implizit den wechselseitigen Ausschluss aufzuheben. Macht die ↑*unilaterale Synchronisation* von Prozessen möglich, unter der Prämisse, dass zu einem Zeitpunkt höchstens ein Prozess im Monitor sein darf. Auf der Variablen sind nur zwei Operationen definiert: `wait` (a. `await`), um das durch die Variable repräsentierte Ereignis zu erwarten, den Prozess zu blockieren, und `signal` (a. `cause`), um eben dieses Ereignis anzuzeigen, einen Prozess zu deblockieren. Erwartet kein Prozess das Ereignis, ist `signal` wirkungslos. Dies bedeutet aber auch, dass eine Ereignisanzeige in Form eines Zustands nicht in der Variablen gespeichert wird. Im Gegensatz dazu speichert `wait` den Zustand, dass einer oder mehrere Prozesse in Erwartung auf ein Ereignis blockiert sind. In dem Fall ist mit jedem ↑*Exemplar* einer solchen Variablen eine ↑*Warteschlange* von Prozessen assoziiert
Untrennbar verbunden mit dem Monitor, das heißt, einem Konzept der *Typisierung* in höheren Programmiersprachen, indem nämlich die korrekte Verwendung von Operationen zur ↑*Synchronisation* durch einen ↑*Kompilierer* abprüfbar wird und so Programmierfehler vermieden werden können. Die zur ↑*Bedingungssynchronisation* erforderlichen Anweisungen erzeugt der Kompilierer, wie auch die Instruktionen, um den Monitor zeitweilig verlassen zu können, ohne diesen in der Zwischenzeit gesperrt zu halten. Mangels Sprachunterstützung ist dieses Konzept jedoch viel mehr zur Programmierkonvention entartet, das heißt, dem Menschen als ↑*Prozessor* wird es erleichtert, ein (vermeintlich) korrektes ↑*nichtsequentielles Programm* zu formulieren.

Befehlszähler Register der ↑*CPU*, das ihrer Befehlsfolgesteuerung (*instruction sequencer*) die gegenwärtige Position im ablaufenden Programm anzeigt; auch ↑*PC* genannt. Diese Positionangabe ist eine ↑*Adresse* im ↑*Arbeitsspeicher*, an der der ↑*Abruf- und Ausführungszyklus* der CPU einen ↑*Maschinenbefehl* erwartet und die pro Zyklus implizit um die Befehlslänge (heute (2016): der Anzahl von Bytes, die der komplette Befehl umfasst) inkrementiert wird. Bei ↑*CISC* hängt diese Länge vom jeweiligen Befehl ab, bei ↑*RISC* ist sie normalerweise für alle Befehle gleich. Je nach CPU geschieht die Weiterschaltung des PC vor (*pre-increment*) oder nach (*post-increment*) dem Abrufen eines Maschinenbefehls: das heisst, im Falle einer ↑*Ausnahmesituation* bei der Befehlsausführung zeigt der PC entweder auf den Befehl, der die ↑*Ausnahme* hervorgerufen hat (*pre*: ab Motorola 68020), oder auf seinen Nachfolger (*post*: Motorola 68000/10, IA-32) im Befehlsstrom. Letzterer Fall macht eine ↑*Ausnahmebehandlung*, die zur Wiederaufnahme der Befehlsausführung führt, gegebenenfalls unmöglich, da der dann im ↑*Unterbrechungszyklus* gesicherte Inhalt des PC den Nachfolger des gescheiterten Befehls adressiert und eine Rückrechnung der Adresse nur möglich wäre, wenn alle Befehle der CPU gleich lang sind — was für CISC (IA-32) im Allgemeinen nicht zutrifft und auch bei RISC nicht sein muss. Abhilfe schafft hier eine Präventivmaßnahme der CPU, in der sie nämlich den jeweiligen Inhalt des PC vor Weiterschaltung vermerkt und in Abhängigkeit von der Ausnahme den vermerkten oder den aktuellen Wert im Unterbrechungszyklus sichert, sie also nach der Art der Ausnahme unterscheidet (*fault-class exceptions*, Intel 64 und IA-32). Zusätzlich zur impliziten Veränderung durch ein Inkrement, kann der PC auch explizit durch spezielle Maschinenbefehle einen Wert erhalten. Bei einer CPU, die den PC zum ↑*Programmiermodell* zugehörig definiert (PDP-11 Register R7, ARM7 Register R15), kann die Veränderung sogar durch normale Schreibbefehle geschehen. In beiden Fällen erfolgen damit Sprünge im Programmablauf, womit Fallunterscheidungen, Schlei-

fen, Aufrufe von einem ↑*Unterprogramm* (ebenso ↑*Unterbrechungshandhaber*) und Rückkehr daraus erst möglich werden.

Benutzthierarchie siehe ↑*funktionale Hierarchie*.

Bereitliste Aufstellung von Exemplaren des Bautyps ↑*Prozess*, die bereit (↑*Prozesszustand*) zur ↑*Einlastung* von einem ↑*Rechenkern* sind. Eine vom ↑*Planer* verwaltete ↑*Warteschlange*, die im Falle mitlaufender (*on-line*) ↑*Ablaufplanung* von ihm auch fortgeschrieben wird. Ob diese Aufstellung als statische (Tabelle) oder dynamische (einfach/doppelt verkettete Liste) Datenstruktur oder in Kombination von beiden (Vorrang- oder Prioritätenliste) implementiert ist, hängt ganz vom ↑*Einplanungsalgorithmus* ab. Konkretes Listenelement in all diesen Varianten ist der ↑*Prozesskontrollblock*, der direkt (ein Verkettungsglied enthaltend) oder indirekt (referenziert durch ein Verkettungsglied) in die Liste aufgenommen wird.

Betriebsart Art und Weise der durch ein ↑*Betriebssystem* ermöglichten Betriebsführung in einem Rechensystem. Varianten der Betriebsführung können etwa sein, entweder nur eine oder auch mehrere Teilnehmende zu unterstützen, die dann womöglich ein oder mehrere Programme, gestapelt oder interaktiv, laufen lassen und jedes Program nur einen ↑*Handlungsstrang* oder mehrere davon hat. Dabei kann die Programmausführung pseudo- oder echt parallel erfolgen, sowohl für einen Mono- als auch ↑*Multiprozessor*, als Ein- oder ↑*Mehrkernprozessor*. Außerdem können die Prozesse entweder keinen oder weichen, festen oder harten Zeitanforderungen unterliegen. Jede dieser Optionen, einzeln für sich genommen oder in geeigneten Kombinationen, begründet eine eigene Rechnerbetriebsart, für die das Betriebssystem jeweils bestimmte Systemfunktionen bereitstellen muss.

Betriebslast Gemeinkosten durch überschüssigen oder indirekten Bedarf beispielsweise an Rechenzeit, Speicher, Bandbreite oder Energie, um eine bestimmte Aufgabe zu erledigen.

Betriebsmittel Mittel, das ein ↑*Prozess* zur Vollbringung der ihm gemäß ↑*Programm* auferlegten Aufgabe benötigt. Eine Ressource, die in Hardware (↑*Prozessor*, ↑*Speicher*, ↑*Peripherie*) oder Software (Programmtext- und -datenbestände) vorliegen kann, begrenzt verfügbar und wiederverwendbar ist oder unbegrenzt zur Verfügung stehen kann und dann aber (in einer Erzeuger-Verbraucher-Beziehung) als konsumierbar gilt.

Betriebsmodus Arbeitsmodus, in dem eine (reale/virtuelle) Maschine aktiv ist. Ein ↑*privilegierter Befehl*, der im falschen Arbeitsmodus zur Ausführung gelangt, wird von der Maschine abgefangen (↑*trap*) und im Betriebssystem als ↑*Ausnahme* behandelt. So lässt sich die ↑*Integrität* des diese Maschine verwaltenden Betriebssystems gewährleisten. Dazu läuft die Maschine im privilegierten Modus, wenn das Betriebssystem aktiv ist, und im unprivilegierten Modus, wenn ein ↑*Maschinenprogramm* aktiv ist und das Rechensystem im ↑*Mehrprogrammbetrieb* benutzt werden soll. Der Moduswechsel wird allgemein durch eine ↑*Abfangung* oder ↑*Unterbrechung*, speziell durch einen ↑*Systemaufruf* bewirkt.

Betriebssystem Menge von ↑*Programmen* und ↑*Unterprogrammen*, die Programme oder Anwendungen unterstützen und die Programmierung oder Bedienung eines Rechensystems erleichtern sollen, die Ausführung von Programmen überwachen und steuern, das Rechensystem nach einer bestimmten Art und Weise für einen Anwendungsfall betreiben, eine ↑*abstrakte* oder ↑*virtuelle Maschine* implementieren. Eine ↑*Softwareschicht*, die die ↑*Betriebsmittel* eines Rechensystems verwaltet, die Betriebsmittelvergabe steuert und die Betriebsmittel nach gewissen Kriterien an mitbenutzende Rechenprozesse verteilt.

Betriebssystemkern Herzstück von einem ↑*Betriebssystem*. Definiert als wesentliche Funktion das ↑*Operationsprinzip* für eine ↑*abstrakte Maschine*, die durch ein Betriebssystem implementiert wird. Das Operationsprinzip bestimmt letztlich die Architektur des Betriebssystems und damit auch den Funktionsumfang seines Kerns. Darüberhinaus stellt die für den jeweiligen Anwendungsfall vom Betriebssystem zu unterstützende ↑*Betriebsart* gegebenenfalls weitere Anforderungen, die sich auch in zusätzlicher und vom Kern zu erbringender Funktionalität

niederschlagen kann. So lässt sich ohne Angabe der Betriebsart der tatsächliche Funktionsumfang eines solchen Kerns ebenso wenig festlegen wie der eines Betriebssystems. Fordert die Betriebsart beispielsweise eine Form von ↑*Simultanverarbeitung*, wird der Kern wenigstens Funktionen zur ↑*Prozesseinplanung*, ↑*Unterbrechungsbehandlung* und ↑*Synchronisierung* umfassen. Läuft es zusätzlich noch auf ↑*Mehrprogrammbetrieb* hinaus, werden im Kern zumindest grundlegende Funktionen zum ↑*Speicherschutz* und zur ↑*Ausnahmebehandlung* sowie ein ↑*Systemaufrufzuteiler* hinzukommen.

Betriebssystemlatenz ↑*Latenzzeit* hervorgerufen durch einen ↑*Systemaufruf* samt Durchführung der damit verbundenen ↑*Systemfunktion*. Je nach ↑*Betriebsart* beziehungsweise ↑*Operationsprinzip* von einem ↑*Betriebssystem* ist damit die Verzögerung, die ein ↑*Handlungsstrang* im ↑*Maschinenprogramm* erfahren kann, gegebenenfalls unbestimmt und nicht berechenbar. Ist die Stelle, an der eine solche Verzögerung wirkt, im Fall eines Systemaufrufs im Maschinenprogramm noch bestimmbar, macht demgegenüber eine ↑*Unterbrechung* eine exakte Vorhersage unmöglich. Bestenfalls kann die ↑*Zwischenankunftszeit* von Unterbrechungen abgeschätzt werden.

Bindefehler ↑*Ausnahmesituation* beim Zugriff auf ein Programmtext oder -daten enthaltendes ↑*Segment* im ↑*Arbeitsspeicher*. Die von einem ↑*Prozess* beim Zugriff verwendete ↑*symbolische Adresse* des Segments ist gültig, jedoch im Moment des Zugriffs hat es noch keinen Platz im Arbeitsspeicher dieses Prozesses. Die Bindung des Segmentnamens (↑*Symbol*) an eine Segmentadresse geschieht erst zur ↑*Laufzeit* von dem betreffenden ↑*Maschinenprogramm*, sie ist dynamisch. Das Betriebssystem enthält dazu einen ↑*Binder*, der die Auflösung des Namens in eine ↑*Adresse* und damit verbunden die Platzierung des Segments im Rahmen einer ↑*Ausnahmebehandlung* durchführt und den beim Zugriff abgefangenen Prozess danach weiter fortsetzt: ↑*partielle Interpretation* des Zugriffs.

bindender Lader ↑*Lader*, der die in einem ↑*Lademodul* noch enthaltenen unaufgelösten (externen) Referenzen gegebenenfalls auflöst. Dabei bildet jede dieser Referenzen eine ↑*symbolische Adresse* von Programmtext oder -daten, typischerweise vorrätig in einer oder mehrerer Bibliotheken. Die Bibliothek mit dem ↑*Objektmodul*, das die gesuchte symbolische Adresse in der ↑*Symboltabelle* listet, wird zur Herstellung einer ↑*Bindung* herangezogen. Steht in keiner Bibliothek ein solches Objektmodul zur Verfügung, scheitert der Ladevorgang.

Binder Dienstprogramm, das mehrere Objektmodule zu einem ↑*Objektmodul* oder ↑*Lademodul* zusammenfügt.

Bindung Beziehung zwischen einem ↑*Symbol* und einer ↑*Adresse*, repräsentiert durch einen Eintrag in der ↑*Symboltabelle*.

binärer Semaphor ↑*Semaphor*, bei dem die ↑*Ablaufsteuerung* eine *binäre Variable* benutzt. Entweder realisiert als ↑*allgemeiner Semaphor* mit Wertebereich $[0, 1]$ oder speziell ausgelegt als boolescher elementarer Datentyp (`bool`). Bei letzterer Variante sind die in den Operationen ↑*P* und ↑*V* erfolgenden Zustandsänderungen in Bezug auf die enthaltene *boolesche Variable* implizit jeweils eine ↑*atomare Operation*: *Zuweisung* von `false` (P) beziehungsweise `true` (V). Die Ablaufsteuerung lässt einen ↑*Prozess* in P blockieren, wenn der Variablenwert `false` ist. Anderenfalls setzt P die (`true` enthaltene) Variable auf `false` und lässt den Prozess voranschreiten. Allein anhand des Wahrheitswerts ist dann nicht feststellbar, ob ein Prozess in P blockiert und durch V gegebenenfalls zu deblockieren ist. In dem Fall hätte V nur die Wahl, die Variable auf `true` zu setzen und den ↑*Planer* die ↑*Prozesstabelle* ablaufen zu lassen (↑*Latenzzeit*), um gegebenfalls zu deblockierende Prozesse fest- und wieder bereitzustellen. Verwendet P jedoch eine semaphoreigene Datenstruktur als ↑*Warteschlange*, kann V darüber einfach prüfen, ob Prozesse zur Deblockierung anhängig sind. Ist die Warteschlange leer, setzt V den Variablenwert auf `true`. Anderenfalls entnimmt V der Warteschlange den nächsten Prozess (↑*Interferenz*) und deblockiert ihn, ohne den Wahrheitswert zu verändern: dieser bleibt `false`, solange noch Prozesse am Semaphor warten. Typischer Anwendungsfall

dieser Art von Semaphor ist ein ↑*kritischer Abschnitt* oder ↑*Monitor* beziehungsweise überall dort, wo ↑*wechselseitiger Ausschluss* erforderlich ist.

Blattprozedur ↑*Unterprogramm*, das kein weiteres Unterprogramm im selben ↑*Adressraum* aufruft, auch nicht sich selbst.

blockierende Synchronisation ↑*Synchronisation*, bei der ein ↑*Prozess* darauf warten muss, bis er an der Reihe ist, eine bestimmte ↑*Aktion* oder ↑*Aktionsfolge* durchzuführen. Diese Aktion/Aktionsfolge ist als ↑*kritischer Abschnitt* beschrieben. Auch als ↑*pessimistische Nebenläufigkeitssteuerung* bezeichnet.

BSS *block started by symbol*, (dt.) Bereich von ausgespartem Raum in einem in ↑*Assembliersprache* formulierten Programm. Der jeweilige Raum wird durch eine Marke (↑*Symbol*) gekennzeichnet, er erhält einen Namen. Seine Größe erhöht den ↑*Adresszähler* bei der ↑*Assemblierung* entsprechend, ohne jedoch einen Inhalt an seiner Stelle zu hinterlassen: der Raum bleibt leer.

busy waiting (dt.) ↑*geschäftiges Warten*.

C prozedurale, imperative Programmiersprache (1969).

C++ objektorientierte, imperative Programmiersprache (1979), Erweiterung von ↑*C*.

cache (dt.) Zwischenspeicher.

cache line (dt.) Zwischenspeicherzeile.

cdecl ↑*Aufrufkonvention* von ↑*C*.

CISC *complex instruction set computer* (dt.) Rechner mit vielschichtigem (komplexen) Befehlssatz.

compiler (dt.) Kompilierer.

computer (dt.) ↑*Rechner*.

concurrency control (dt.) ↑*Nebenläufigkeitssteuerung*.

condition variable (dt.) ↑*Bedingungsvariable*.

condition-code register (dt.) Statusregister.

continuation (dt.) ↑*Fortsetzung*.

coroutine (dt.) ↑*Koroutine*.

covered channel (dt.) ↑*verdeckter Kanal*.

CPU *central processing unit*, (dt.) Zentraleineinheit.

critical section (dt.) ↑*kritischer Abschnitt*.

CSIM *complete software interpreter machine*, (dt.) vollständig in Software realisierter ↑*Interpreter*.

CTSS Mehrplatz-/Mehrbenutzerbetriebssystem. Abkürzung für „*Compatible Time-Sharing System*“. Führte das ↑*Zeitteilverfahren* ein, um einem ↑*Prozess* die lllusion vom eigenen ↑*Prozessor* zu verschaffen (↑*partielle Virtualisierung*) und so mehrere Prozesse zugleich stattfinden lassen zu können (↑*Simultanbetrieb*). Das System war kompatibel mit ↑*FMS*. Erste Installation im Jahr 1961 (modifizierte IBM 7094).

Dämon ↑*Prozess*, der im Hintergrund stattfindet und dabei eine bestimmte ↑*Systemfunktion* erfüllt. Der Prozess findet in einem dedizierten ↑*Maschinenprogramm*, also oberhalb eines Betriebssystems, oder in dem Betriebssystem selbst statt.

Darwin Mehrplatz-/Mehrbenutzerbetriebssystem, von ↑*UNIX* abstammend. Erste Installation März 1999 (PowerPC), Basis für Mac OS X. Programmiert in ↑*Objective-C*, ↑*C++* und ↑*C*.

data processing system (dt.) Datenverarbeitungssystem, ↑*Rechensystem*.

Datensegment ↑*Segment*, das Programmdaten speichert.

dauerhaftes Betriebsmittel ↑*wiederverwendbares Betriebsmittel*.

demon (dt.) Dämon.

Dienstprogramm ↑*Programm* für systemnahe Aufgaben, meist zum ↑*Betriebssystem* gehörend. Solche Aufgaben bestehen beispielsweise darin, ↑*Objektmodule* zu einem ↑*Lademodul* zusammenzubinden (`ld(1)`), Dateiverzeichnisse aufzulisten (`ls(1)`), Dateien zu kopieren (`cp(1)`), Druckaufträge „aufzuspulen“ (`lp(1)`), den Zustand von Prozessen darzustellen (`ps(1)`), Parameter der Netzwerkschnittstelle zu konfigurieren (`ifconfig (8)`) oder Statusdaten im Betriebssystem zu manipulieren (`sysctl (8)`).

Direktzugriffsspeicher ↑*Speicher* für wahlfreien/direkten Zugriff auf jedes ↑*Speicherwort* über eine jeweils eindeutig zugeordnete ↑*Adresse*.

dispatching (dt.) ↑*Einlastung*.

DMA *direct memory access*, (dt.) Speicherdirektzugriff.

DRAM dynamischer ↑*RAM*.

Dringlichkeit Grad, in dem die Bearbeitung eines Auftrags drängt, in dem ein ↑*Prozess* den ↑*Prozessor* zugeteilt bekommen oder abgeschlossen werden muss.

Dualsystem Zahlensystem, das nur zwei Ziffern (0, 1) zur Darstellung aller Zahlen als Potenzen von 2 verwendet.

Durchlaufzeit Zeitspanne zwischen der Ankunft von einem ↑*Prozess* und seiner vollständigen Beendigung. Bezugspunkt dabei ist der ↑*Planer*, der nämlich bestimmt, wann ein Prozess das erste Mal auf die ↑*Bereitliste* kommt, wann dieser Prozess zur ↑*Einlastung* von dem ↑*Prozessor* ansteht, wie oft der Prozess einer ↑*Verdrängung* unterzogen wird und wann der Prozess die ↑*Ablaufplanung* verlässt.

Durchsatz Anzahl von Aufträgen, die ein ↑*Rechen-*, ↑*Betriebssystem* oder ↑*Prozess* pro Zeiteinheit verarbeiten kann. Beispielsweise die Anzahl von Ein-/Ausgabeaufträgen, abgesetzt von einem einzelnen Prozess oder einer bestimmten Gruppe von Prozessen.

Echtzeit Betrieb von einem ↑*Rechensystem*, bei dem ein ↑*Programm* zur Verarbeitung anfallender Daten ständig betriebsbereit ist, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen (nach DIN ISO/IEC 2382). Vorgegebene Zeit, die ein ↑*Prozess* in der Realität verbrauchen darf (in Anlehnung an den Duden).

edge-triggered interrupt (dt.) flankengesteuerte ↑*Unterbrechung*, ↑*Flankensteuerung*.

EEPROM *electrically erasable PROM*, (dt.) elektrisch löschbarer ↑*PROM*.

Einlastung Vorgang der Belegung einer Verarbeitungseinheit mit einem Auftrag. Die Verarbeitungseinheit ist ein ↑*wiederverwendbares Betriebsmittel* bestimmter Art, das nach Gebrauch für einen weiteren Auftrag entsprechender Art weiterhin verwendet werden kann. Typisches Beispiel für einen Auftrag ist eine ↑*Aufgabe*, für deren Erledigung ein ↑*Prozess* stattfinden und dazu wiederum ein ↑*Prozessor* zugewiesen, das heißt, unter Last gesetzt werden muss. Nach

Gebrauch des Prozessors ist dieser frei für einen weiteren Prozess (↑*Prozesswechsel*). Anderes Beispiel ist Ein-/Ausgabe, bei der ein ↑*Peripheriegerät* durch einen Prozess den Auftrag zur Lieferung von Eingabe- oder Annahme von Ausgabedaten erhält. Nach Gebrauch des Geräts ist dieses frei für einen weiteren Ein-/Ausgabeauftrag.

Einplanungsalgorithmus Lösungs- und Bearbeitungsschema, Handlungsvorschrift zur Planung der ↑*Einlastung*. Ist charakterisiert durch die Reihenfolge von Aufträgen in einer ↑*Warteschlange* und die Bedingungen, unter denen die Aufträge dort eingereiht werden. Grob unterschieden wird dabei zwischen anwendungs- und systemorientierten Kriterien, das heißt, dem in einer Anwendung wahrnehmbaren Systemverhalten einerseits und der effektiven und effizienten Auslastung der ↑*Betriebsmittel* andererseits. Charakteristische Merkmale anwendungsorientierter Kriterien in Bezug auf das ↑*Betriebssystem* sind ↑*Antwortzeit*, ↑*Durchlaufzeit*, ↑*Termineinhaltung* und ↑*Vorhersagbarkeit*. Demgegenüber stehen wünschenswerte Merkmale systemorientierter Kriterien wie ↑*Durchsatz*, ↑*Prozessorauslastung*, ↑*Gerechtigkeit*, ↑*Dringlichkeit* und ↑*Lastausgleich*. Beiden Orientierungen mit ein und demselben Verfahren gerecht zu werden, ist in aller Regel nicht möglich. Ein mehrstufiger Ansatz kann Abhilfe schaffen, wobei den einzelnen Stufen dann bestimmten Kriterien zugeordnet sind. In solch einem Szenario erfolgt dann jedoch immer eine Schwerpunktsetzung, die sich durch die Reihenfolge ergibt, in der die einzelnen Ebenen durchlaufen werden und dadurch ein Kriterium ein anderes dominiert.

einseitige Synchronisation ↑*unilaterale Synchronisation*.

Eintrittsinvarianz Charakteristik eines Programmabschnitts zum ↑*Wiedereintritt*.

Elementaroperation Primitive einer (realen/virtuellen) Maschine; grundlegende, wesentliche Operation in Bezug auf eine bestimmte Ebene der Abstraktion. Eine ↑*Aktion*, die auf dieser Ebene in einem Schritt (ungeteilt, atomar) stattzufinden scheint.

Elop Abk. für ↑*Elementaroperation*.

EPROM *erasable PROM*, (dt.) löschbarer ↑*PROM*.

Ereignis Auftreten von einem Geschehnis hervorgerufen durch einen ↑*Prozess*, beobachtbar von dem Prozess selbst oder einem anderen Prozess. Das Geschehnis ist im ↑*Rechensystem* direkt beobachtbar, indem ein Prozess etwa die Veränderung des Inhalts von einem ↑*Speicherwort* durch Abfragen (*polling*) oder Abwarten (*waiting*) wahrnimmt. Letzteres wird durch ↑*unilaterale Synchronisation* erreicht, das heißt, der Prozess wartet (aktiv/geschäftig oder passiv/schlafend) solange, bis ihm die Veränderung durch einen anderen Prozess explizit angezeigt wird. Das Geschehnis kann aber auch indirekt beobachtbar sein, wenn es einem Prozess nämlich möglich ist, aus wahrnehmbarem Systemverhalten auf das Auftreten gewisser Vorgänge im Rechensystem zu schließen (↑*covert channel*).

exception (dt.) Ausnahme.

exception handler (dt.) Ausnahmehandhaber.

exception handling (dt.) Ausnahmebehandlung.

Exemplar Einzelstück aus einer Menge gleichartiger Stücke (Duden). Die Stücke sind gleichartig, weil ihnen dasselbe Modell, dieselbe Bauart zugrunde liegt: sie sind vom selben *Bautyp*. So ist eine Programmvariable ein solches Einzelstück, ebenso wie ein Objekt allgemein.

FAA *fetch and add*, atomare Veränderung von einem ↑*Speicherwort*, hier: alten Wert abrufen, dann addieren und die Summe speichern (Postinkrement).

Faden ↑*Handlungsstrang* mit eigenem ↑*Laufzeitkontext* und typischerweise mitlaufender (*on-line*) ↑*Ablaufplanung*. Grob differenziert in ↑*Anwendungsfaden* und ↑*Systemkernfaden*.

false sharing (dt.) ↑*irrige Mitbenutzung*.

Faser ↑*Faden*, der strikt kooperativer ↑*Ablaufplanung* unterliegt und damit nicht gleichzeitig mit anderen seiner Art abläuft. Auch *leichtgewichtiger Faden*, da bei dieser Art der Ablaufplanung die Kontrolle durch das ↑*Betriebssystem* entfällt, der Aufwand für die ↑*Zustandssicherung* auf ein Minimum reduziert werden kann und ↑*Synchronisation* implizit sichergestellt ist. Verschiedentlich auch einer ↑*Koroutine* gleichgestellt.

feather-weight process (dt.) ↑*federgewichtiger Prozess*.

federgewichtiger Prozess ↑*Prozess*, der als ↑*Anwendungsfaden* mit anderen Prozessen seiner Art zusammen im gemeinsamen und durch ↑*Speicherschutz* isolierten ↑*Adressraum* stattfindet.

fetch-execute-cycle (dt.) Abruf- und Ausführungszyklus.

fiber (dt.) ↑*Faser*.

fibril (dt.) ↑*Fäserchen*.

first-class object (dt.) ↑*Objekt erster Klasse*.

flag bit (dt.) Markierungsbit.

Flankensteuerung Auslöser für die ↑*Unterbrechung* ist ein Wechsel des Pegelstands auf der Störleitung (*interrupt line*) zur ↑*CPU*, hervorgerufen durch die ↑*Peripherie*. Der Impuls zu diesem Wechsel ist nur lang genug, um von der CPU als ↑*Unterbrechungsanforderung* erkannt zu werden. Gängig ist die Zwischenspeicherung (↑*latch*) des Signals, da die CPU normalerweise nur zwischen zwei Durchläufen des ↑*Abruf- und Ausführungszyklus*, also nach der Ausführung von einem ↑*Maschinenbefehl*, den Unterbrechungszyklus fährt. Jedoch erkennt die CPU nur höchstens einen Impuls pro Durchlauf, wiederholte Impulse (*reassertion*) gehen unerkannt verloren. Eine ↑*Unterbrechungssperre* verlängert dieses Intervall und erhöht die Wahrscheinlichkeit verpasster Unterbrechungsanforderungen. Dieser Aspekt ist besonders virulent bei Mitbenutzung der Störleitung (*interrupt sharing*), wenn nämlich mehr als ein ↑*Peripheriegerät* entweder direkt oder indirekt, im letzteren Fall durch einen ↑*PIC*, an die CPU angeschlossen werden soll. Wiederholte Impulse auf der Störleitung sind durch die verschiedenen und voneinander unabhängigen Peripheriegeräte in solch einer Konstellation Normalität. Damit ist der flankengesteuerte Ansatz recht anfällig für den Verlust von Unterbrechungsanforderungen, im Gegensatz zur ↑*Pegelsteuerung*.

flüchtiges Register Konzept der ↑*Aufrufkonvention*: der Inhalt eines solchen Prozessorregisters gilt als unbeständig. Vorkehrungen zur Sicherung und Wiederherstellung des Registerinhalts werden im ↑*Unterprogramm* nicht getroffen, sondern gegebenenfalls im aufrufenden Programm (*caller-saved*). So definiert beispielsweise ↑*cdecl* für ↑*x86* die Register `EAX`, `ECX` und `EDX` als flüchtig, wobei `EAX` den Funktionsrückgabewert speichert.

FMS Abkürzung für „*FORTRAN Monitor System*“. Gilt als erstes wirkliches ↑*Betriebssystem*, basierend auf Magnetbandgeräte, das automatisch mehr als ein ↑*Programm* nacheinander im Stapel (*batch*) verarbeiten konnte. Die Programme konnten zudem in ↑*FORTRAN* formuliert sein und wurden dann vor Ausführung automatisch der ↑*Kompilation* unterzogen. Erste Installation im Jahr 1955 (IBM 704).

formaler Parameter Bestandteil der formalen Schnittstelle (Signatur) von einem ↑*Unterprogramm*. Bezeichner eines Platzhalters zur Übernahme eines zuweisungskompatiblen Werts (↑*tatsächlicher Parameter*) als Argument eines Unterprogrammaufrufs.

FORTRAN Abk. für „*formula translation*“; prozedurale, imperative Programmiersprache (1957).

Fortsetzung Mechanismus zur Übergabe der Kontrolle über den ↑*Rechenkern* an einen anderen ↑*Handlungsstrang*. Letzterer ist in dem Fall nicht als ↑*Faden* implementiert (auch in keiner Variante davon) und besitzt daher keinen eigenen ↑*Laufzeitstapel*, um *implizit* darauf beim ↑*Prozesswechsel* den Zustand zur Wiederaufnahme des Programmablaufs sichern zu können. Stattdessen teilen sich alle Handlungsstränge ein und denselben Stapel und sichern diesen Zustand *explizit* in ein ↑*Objekt erster Klasse*. Dieses Objekt liegt in einer Form vor, die es erlaubt, es als Funktion aufrufen zu können. Beim Aufruf sichert der Handlungsstrang seinen aktuellen Zustand und gibt sich einen neuen Zustand, den er einem zweiten Objekt entnimmt. Soweit es den ↑*Prozessorstatus* betrifft, ist der Zustand nur definiert durch ↑*nichtflüchtige Register* und insbesondere den ↑*Befehlszähler*. Damit wird der Handlungsstrang nicht an die Stelle des Funktionsaufrufes zurückkehren, sondern immer an eine Stelle, die durch den Befehlszähler des wiederhergestellten Zustands bestimmt ist.

FPU *floating point unit*, (dt.) Gleitkommaprozessor.

funktionale Hierarchie Gesamtheit von in einer Rangfolge stehenden Ebenen, wobei jede Ebene durch eine bestimmte Menge von Funktionen definiert ist. Der Idee nach stellt eine Ebene in einer solchen Anordnung eine (reale/virtuelle) Maschine für die nächst höhere Ebene zur Verfügung. Im zugehörigen Entwurf wird eine Ebene oberhalb einer anderen Ebene platziert, um zum Ausdruck zu bringen, dass das korrekte Funktionieren ersterer von der Existenz einer korrekten Implementierung der Funktionen letzterer abhängt: dass die obere Ebene die untere „benutzt“.

Fäserchen ↑*Faden*, der ausschließlich im ↑*Betriebssystemkern* (↑*Linux*) residiert. Ein solcher Faden bildet die technische Grundlage, um eine ↑*Systemfunktion*, etwa ausgelöst durch einen asynchronen ↑*Systemaufruf*, unabhängig von anderen Vorgängen stattfinden zu lassen.

Gastbetriebssystem ↑*Betriebssystem*, das einen ↑*VMM* oder ein ↑*Wirtsbetriebssystem* benutzt (↑*Benutzthierarchie*). In reiner Ausführung nimmt das Gastsystem diese Benutzung nicht wahr: jede in diesem System verursachte ↑*Ausnahme* wird vom jeweiligen Wirt abgefangen (↑*trap*) und in funktionaler Hinsicht „transparent“ behandelt. Demgegenüber kann das Gastsystem in der Realisierung einer eigenen ↑*Systemfunktion* jedoch auch vom Wirtssystem profitieren, allerdings impliziert dies Änderungen im Gastsystem (unreine Ausführung).

Gerechtigkeit Prinzip des Verhaltens von einem ↑*Betriebssystem*, das jedem ↑*Prozess* gleichermaßen Fortschritt zusichert. Fairness in der ↑*Synchronisation* von Prozessen oder der Vergabe von einem oder mehrerer ↑*Betriebsmittel* an einen Prozess.

Gerätetreiber ↑*Unterprogramm* in einem ↑*Betriebssystem*, durch das ein bestimmtes ↑*Peripheriegerät* verwaltet und bedient wird. Nach außen stellt solch ein Unterprogramm typischerweise eine für eine Klasse von Peripheriegeräten einheitliche Schnittstelle zur Verfügung (*major device*, ↑*UNIX*) nach innen ist es speziell zugeschnitten auf das jeweilige ↑*Exemplar* eines Geräts dieser Klasse (*minor device*, ↑*UNIX*).

geschäftiges Warten Situation, in der ein ↑*Prozess* durch anhaltendes Abfragen (*polling*) von einem ↑*Speicherwort* ein bestimmtes ↑*Ereignis* erwartet. Gegebenenfalls unterbricht sich der Prozess nach jedem Abfrageschritt (re.)

```
void probe(event) {
    while(*event == 0);
}
```

```
void probe(event) {
    while (*event == 0)
        favor(event);
}
```

und signalisiert damit dem ↑*Planer*, den ↑*Prozessor* einem anderen Prozess zuteilen zu können. Der wartende Prozess wechselt jedoch nur dann von laufend nach bereit (↑*Prozesszustand*), wenn es in dem Moment einen bereitgestellten (anderen) Prozess gibt und dieser gemäß ↑*Prozesseinplanung* keine niedrigere Priorität besitzt. In logischer Hinsicht behält der wartende Prozess den Prozessor, obwohl er dabei selbst keine produktive Arbeit verrichten kann.

Behält der Prozess den Prozessor auch in physischer Hinsicht, trägt er selbst zur Verlängerung seiner Wartezeit bei, da kein anderer Prozess den Prozessor zugeteilt bekommt, um so das erwartete Ereignis herbeiführen zu können. Letzteres trifft auch dann zu, wenn die Prozesseinplanung nach einem ↑*Zeitteilverfahren* arbeitet: dann ist die Länge des dem Prozess zugebilligten Zeitintervalls maßgeblich dafür, wann ihn ein anderer Prozess verdrängt, der dann eventuell das erwartete Ereignis bewirkt.

Granularität Anzahl von Untergliederungen eines Elements (Duden). Beispielsweise die Byteanzahl pro ↑*Speicherwort*, Speicherwortanzahl pro ↑*Zwischenspeicherzeile*, Zwischenspeicherzeilenanzahl pro ↑*Seite*, Seitenanzahl pro ↑*Segment* oder Segmentanzahl pro ↑*Adressraum*, wenn insbesondere verschiedene Ebenen in der ↑*Speicherhierarchie* zusammenhängend betrachtet werden.

Handlungsstrang ↑*Aktionsfolge*, die der Bearbeitung einer komplexen und zusammenhängen Aufgabe entspricht. Mehrere Handlungsstränge können ein und dieselbe Aktionsfolge gemeinsam haben (Exemplare desselben Typs), nur in Teilen der Aktionsfolge überschneiden (Exemplare verschiedener Typen, aber mit wenigstens einem gemeinsamen ↑*Unterprogramm*) oder komplett verschiedene Aktionsfolgen ausmachen (Exemplare verschiedener Typen).

Hauptprogramm ↑*Programm*, das in logischer Hinsicht von keinem anderen Programm aufgerufen wird: es beginnt seine Ausführung, indem es vom ↑*Betriebssystem* gestartet wird. Technisch geschieht dies in bestimmten Fällen sehr wohl durch einen Aufruf, beispielsweise einen in der Form `main(argc, argv, envp)` bei ↑*C*. Hier erfolgt der Aufruf durch eine spezielle ↑*Aktionsfolge* zur Inbetriebnahme (*startup*) des Programms überhaupt, und zwar im Namen von der vom Betriebssystem vorher dazu eingerichteten ↑*Prozessinkarnation*. Dieser Aufruf bewirkt auch, dass `main()` sehr wohl zurückkehren kann. So wird `main()` letztlich von einer speziellen ↑*Mantelprozedur* aufgerufen, die zudem für die gegebenenfalls benötigte Parameterversorgung (`argc`, `argv`, `envp`) sorgt. Nach Rückkehr aus `main()` zu dieser Mantelprozedur verlässt der ↑*Prozess* per ↑*Systemaufruf* (`_exit(2)`) das ↑*Maschinenprogram*, betritt das Betriebssystem und terminiert dort.

Hauptspeicher Physisch verfügbarer ↑*Speicher* zur Programmausführung, in dem alle dazu erforderlichen Programmtext- und -datenbestände flüchtig (temporär: ↑*DRAM*, ↑*SRAM*) oder nichtflüchtig (permanent: ↑*ROM*, ↑*PROM*; semi-permanent: ↑*EPROM*, ↑*EEPROM*, ↑*NVRAM*) vorliegen. ↑*Direktzugriffsspeicher* (↑*RAM*), im Allgemeinen als flüchtig ausgelegt verstanden in Form von Schreib-lese-Speicher.

Hauptsteuerprogramm ↑*Programm* im ↑*Betriebssystemkern*, das alle wesentlichen Funktionen zur Mehrfachnutzung der zugrunde liegenden (realen/virtuellen) Maschine entsprechend der gewünschten ↑*Betriebsart* umfasst. Mindestens enthalten im Funktionsumfang sind ↑*Prozesseinplanung* und ↑*Ausnahmebehandlung*. Eine typische weitere Funktion ist der ↑*Speicherschutz*, sofern die bereitzustellende Betriebsart dies erfordert.

heavy-weight process (dt.) ↑*schwergewichtiger Prozess*.

hierarchische Struktur Anordnung der Teile eines Ganzen, die durch eine Methode der Aufteilung eines Systems in seine Einzelbestandteile und die Art der Relation zwischen Teilepaaren bestimmt ist. Strukturen sind hierarchisch, wenn eine solche Relation $R(\alpha, \beta)$ Ebenen entstehen lässt. Dabei ist Ebene_0 eine Menge von Teilen α, so dass es kein β gibt mit $R(\alpha, \beta)$. Ebene_i, für $i > 0$, ist Menge von Teilen α, so dass gilt: es existiert ein β auf Ebene_{i-1} mit $R(\alpha, \beta)$ und falls $R(\alpha, \gamma)$, dann liegt γ auf Ebene_{i-1}. Dabei sind folgende Arten von R geläufig: „benutzt“, wenn das korrekte Funktionieren von α die Existenz wenigstens einer korrekt funktionierenden Implementierung von β voraussetzt (↑*Benutzthierarchie*, ↑*funktionale Hierarchie*); „ruft“, wenn α einen Aufruf an ↑*Unterprogramm* β enthält (Aufrufhierarchie); „beauftragt“, wenn α einen Auftrag an β abgibt und beide Vorgänge dabei unabhängig von der relativen Geschwindigkeit des jeweils anderen Vorgangs operieren (Prozesshierarchie).

Hintergrundspeicher ↑*Speicher* zur Auslagerung von Programmen und Daten; auch Sekundärspeicher. Peripherer Speicher, der indirekt über Ein-/Ausgabeoperationen des Betriebssystems bedient wird.

Hypervisor ↑*Programm*, das eine ↑*virtuelle Maschine* steuert und überwacht; entspricht einem ↑*VMM*, Typ I oder II. Üblichweise stellt dieses Programm mehrere Exemplare desselben Bautyps einer virtuellen Machine zur Verfügung und verwaltet diese entsprechend (Mehrfachnutzung). Legendarisch als Steuerprogramm für das „Leitsystem“ (↑*supervisor*) von ↑*VM/370* bestimmt, woraus sich die Namensgebung ableitet.

idle (dt.) ↑*Leerlauf*.

Informationsstruktur Tripel von Typen von Informationsträgern, ihrer Repräsentation und der Menge der auf sie anwendbaren Operationen. Auch zu verstehen als Menge von abstrakten Datentypen einer (realen/virtuellen) Maschine.

Integrität Unverletzlichkeit von Programmtext und -daten, die korrekte Funktionsweise eines Systems gemäß Spezifikation.

Interferenz Überlagerung beim Zusammentreffen zweier oder mehrerer Vorgänge, die sich dadurch möglicherweise gegenseitig beeinflussen. Jeder Vorgang entspricht dabei einem ↑*Prozess*, der wenigstens ein ↑*Betriebsmittel* mit einen anderen Prozess gemeinsam hat. Teilen sich zwei Prozesse beispielsweise die ↑*CPU* im Zeitmultiplexverfahren (↑*partielle Virtualisierung*), kann der eine Prozess durch den anderen verdrängt werden. Der verdrängende Prozess überlagert den verdrängten Prozess und verzögert diesen dadurch. Darf das Betriebsmittel nur exklusiv genutzt werden, ist im Falle mehrerer Prozesse, die gleichzeitig darauf zugreifen, ↑*Synchronisation* explizit sicherzustellen. Ein Prozess, der ein solches Betriebsmittel hält, beeinflusst andere Prozesse, die dieses Betriebsmittel gleichzeitig beanspruchen und sich synchronisieren müssen, wodurch letztere sich gegenseitig selbst beeinflussen. Spezielle Ausprägung davon ist ein ↑*kritischer Abschnitt*. Grundsätzlich kann ↑*Synchronisierung* die ↑*Ablaufplanung* überlagern, was Störungen im ↑*Ablaufplan* zur Folge haben kann; genauer: ↑*blockierende Synchronisation*, die eine Prozessreihenfolge festlegt und dadurch der Entscheidung der Ablaufplanung möglicherweise entgegenwirkt.

Interpretation Vorgang der Deutung und bedingten Ausführung der Anweisungen in einem Programm durch einen Interpreter. Ausgeführt werden nur gültige Anweisungen. Ob und unter welchen Bedingungen eine Anweisung gültig ist und was eine ungültige Anweisung zur Folge hat, legt das ↑*Operationsprinzip* der (realen/virtuellen) Maschine fest, die der Interpreter implementiert. Der Befehlssatz dieser Maschine definiert die Menge der allgemein gültigen Anweisungen (d.h., Befehle), aus denen die Programme für diese Maschine formuliert werden. Ungültige Anweisungen werden von der Maschine ignoriert oder bewirken eine ↑*Ausnahme*, die von der Maschine erhoben wird. Eine solche Ausnahme kann die ↑*partielle Interpretation* der betroffenen Anweisung bedeuten.

Interpreter Soft-, Firm- oder Hardwareprozessor, der die Anweisungen eines Programms deutet und direkt ausführt. Gegebenenfalls erfolgt zuvor eine ↑*Vorübersetzung* durch einen Kompilierer, um die semantische Lücke zwischen der Quellsprache, in der das Programm formuliert ist, und der ↑*Interpretersprache*, in der das zu deutende Programm vorliegen muss, zu verringern und dadurch die Interpretation zu beschleunigen.

Interpretersystem Prinzip nach dem die ↑*Interpretation* von einem ↑*Programm* geschieht. Normalerweise erfolgt diese *total*, also vollständig durch einen ↑*Interpreter*, der die (reale/virtuelle) Maschine, für die das Programm bestimmt ist, implementiert. Im Falle einer ↑*Ausnahme* kann jedoch ein anderer Interpreter helfend eingreifen und teilweise die Programmausführung übernehmen (↑*partielle Interpretation*).

interrupt (dt.) ↑*Unterbrechung*, unterbrechen.

interrupt handler (dt.) ↑*Unterbrechungshandhaber*.

IPL *interrupt priority level*, (dt.) ↑*Unterbrechungsprioritätsebene*.

IRQ *interrupt request*, (dt.) Unterbrechungsanforderung.

irrige Mitbenutzung Zugriffsmuster auf einzelne Strukturelemente (↑*Speicherwort*) der kleinsten Verwaltungseinheit einer Speicherressource (↑*Zwischenspeicherzeile*, ↑*Seite*), wobei nur letztere der Mitbenutzung (*sharing*) unterliegt. Auch wenn simultane Schreib-lese-Vorgänge verschiedene Strukturelemente betreffen, also logisch nicht zueinander in Konflikt stehen, besteht in solch einem Fall dennoch ein physischer Schreib-lese-Konflikt, nämlich in Bezug auf die umfassende Verwaltungseinheit. Zur Konsistenzwahrung zwischengespeicherter globaler Daten bewirkt jeder Schreibvorgang entweder die Ausspülung (*write-invalidate*) oder die Aktualisierung (*write-update*) der betreffenden Speicherressource. Dies hat leistungsmindernde Blocktransfers zur Folge, obwohl in logischer Hinsicht keine Zugriffskonflikte bestehen.

Java objektorientierte, imperative Programmiersprache (1995), eigentlich ↑*Interpretersprache*.

job (dt.) Tätigkeit, hier: ↑*Teilaufgabe*.

JVM *Java Virtual Machine*, Teil der Laufzeitumgebung für ein in ↑*Java* formuliertes ↑*Programm*. Typischerweise als ↑*CSIM* realisierte ↑*virtuelle Maschine*, die ↑*Zwischenkode* (konkret: ↑*Java* Bytekode) direkt ausführt. Jedes Programm läuft auf seiner eigenen virtuellen Maschine.

kernel thread (dt.) ↑*Systemkernfaden*.

Kompilation Vorgang der Übersetzung eines Programms durch einen Kompilierer.

Kompilierer Softwareprozessor, der ein Programm, das in einer Quellsprache formuliert vorliegt, in ein semantisch äquivalentes Programm einer Zielsprache übersetzt.

konsumierbares Betriebsmittel ↑*Betriebsmittel*, das logisch in unbegrenzter Anzahl vorhanden und von dem jede Einheit verfügbar ist. Wird eine Einheit davon von einem als *Konsument* (Verbraucher) auftretender ↑*Prozess* erworben (*acquire*), hört sie in dem Moment auf zu bestehen, sie erlischt, verliert Gültigkeit, wird implizit zerstört. Nur ein als *Produzent* (Erzeuger) dieses Betriebsmittels agierender Prozess kann Einheiten davon in beliebiger Anzahl freisetzen (*release*). Dies kann zu beliebigen Zeitpunkten geschehen, sofern der Prozess seinerseits nicht den Erwerb einer Betriebsmitteleinheit erwartet und demzufolge blockiert ist. Jede freigesetzte Einheit des Betriebsmittels wird sofort verfügbar. Beispiele sind Signale der Hardware (↑*IRQ*, ↑*NMI*) oder Software (↑*Semaphor*, `signal(2)`), Nachrichten oder jede Form von Daten, die im weitesten Sinne Ein- oder Ausgabe eines Prozesses darstellen.

Kontrollstruktur Spezifikation der für die Interpretation benötigten Algorithmen und der Transformation der Informationsträger einer (realen/virtuellen) Maschine.

Koordination Aufeinanderabstimmen von Vorgängen, diese nebenordnen. Die dazu in einem ↑*Rechensystem* vorhandenen Maßnahmen greifen unterschiedlich. Zum einen die ↑*Ablaufplanung*, die einen ↑*Prozess* nebenordnet bevor dieser seine Tätigkeit (erneut) aufnimmt. Sind für die Prozesse alle Daten-, Kontrollfluss- und Zeitabhängigkeiten vorab bekannt, ist ein (statischer) ↑*Ablaufplan* möglich, durch den die Prozesse implizit koordiniert stattfinden werden. Fehlt jedoch dieses Wissen im Moment der ↑*Prozesseinplanung* oder verbietet eine zu hohe Berechnungskomplexität die mitlaufende (*on-line*) Erstellung und Fortschreibung eines solchen Plans, ist zum anderen explizite ↑*Nebenläufigkeitssteuerung* der dann stattfindenden Prozesse erforderlich. Letztere unternehmen die Prozesse selbst, indem sie Einvernehmen über den weiteren Ablauf in Abhängigkeit der von ihnen gerade beabsichtigen ↑*Aktion* oder ↑*Aktionsfolge* erzielen und sich so nebenordnen.

Koroutine ↑*Unterprogramm*, das mit anderen Unterprogrammen zusammen auf derselben Stufe steht. Jedes dieser Unterprogramme nimmt die Rolle als ↑*Hauptprogramm* ein, obgleich es kein solches Hauptprogramm wirklich gibt: bei Ausführung wird keins dieser Unterprogramme aufgerufen, jedoch kann, wechselseitig und in kooperativer Art und Weise, jedes das andere fortsetzen (*resume*). Dazu unterbricht ein solches Unterprogramm seine Ausführung, um diese auf Veranlassung eines anderen Unterprogramms dieser Art später wieder aufzunehmen, wobei der ↑*Prozessorstatus* des jeweils unterbrochenen Unterprogramms als invariant gilt. Da zudem keins dieser Unterprogramme aufgerufen wird, können sie nirgends hin zurückkehren: der Versuch dazu stellt eine ↑*Ausnahmesituation* dar. Beabsichtigt das Unterprogramm seine Beendigung, macht es dies durch einen entsprechenden globalen Zustand deutlich, unterbricht seine Ausführung und veranlasst dadurch die Möglichkeit seiner Entsorgung von anderer Stelle. Neben der ↑*Fortsetzung* ist ein solches Unterprogramm eine weitere Option, um einen autonomen und mit eigenem ↑*Laufzeitkontext* ausgestatteten ↑*Handlungsstrang* zu implementieren (↑*nichtsequentielles Programm*).

kritischer Abschnitt Bereich in einem ↑*Programm*, der zu jedem Augenblick nur von höchstens einen ↑*Prozess* besetzt sein darf. Für Prozesse, die gleichzeitig einen solchen Bereich betreten wollen, ist ↑*blockierende Synchronisation* sicherzustellen.

Lademodul Eingabedatei eines Laders, Ausgabedatei eines Binders.

Lader ↑*Systemfunktion*, die das durch ein ↑*Lademodul* beschriebene ↑*Maschinenprogramm* in den ↑*Arbeitsspeicher* platziert und abschließend einen ↑*Prozess* damit verknüpft. Dieser Prozess existiert bereits und hat den ↑*Systemaufruf* zum Laden selbst abgesetzt (UNIX `exec`) oder er wird beim Ladevorgang erzeugt (VMS `run`, Windows `spawn`).

Ladezeit Zeitpunkt zu dem ein ↑*Programm* zur Ausführung in den ↑*Arbeitsspeicher* geladen wird oder Zeitspanne eben dieses Ladevorgangs.

Lastausgleich Berechnungen beziehungsweise Aufträge auf mehr als einen ↑*Rechenkern* von einer ↑*CPU* oder einem ↑*Multiprozessor* verteilen. Setzt ↑*Parallelverarbeitung* voraus.

latch (dt.) Auffangregister; zustandsgesteuertes FlipFlop (1-Bit ↑*Speicher*).

Latenzzeit Zeitspanne einer technisch bedingten Verzögerung.

Laufzeit Zeitpunkt zu dem ein ↑*Prozess* stattfindet oder Zeitspanne einer ↑*Aktion*.

Laufzeitkontext Gesamtheit an Zustandsdaten der (realen/virtuellen) Maschine, die einen ↑*Prozess* definieren. Neben dem ↑*Laufzeitstapel* ist ein weiteres wesentliches Bestandteil dieser Daten der ↑*Prozessorstatus*. Kommt es zur Unterbrechung eines Prozesses, spiegelt der in dem Moment gültige Prozessorstatus den Kontext zur späteren Fortsetzung eben dieses Prozesses wider. Um den Prozess fortsetzen zu können, als wenn seine Unterbrechung nie geschehen wäre, ist der Prozessorstatus invariant zu halten. Dies wird erreicht durch eine zeitweilige ↑*Zustandssicherung* in eine dem Prozess eigene Datenstruktur im ↑*Hauptspeicher*: Der Prozessorstatus wird bei der Unterbrechung in diese Datenstruktur gesichert und zur Fortsetzung wieder daraus geladen.

Laufzeitstapel ↑*Stapelspeicher* von einem ↑*Handlungsstrang*; dient vornehmlich der zeitweiligen Lagerung lokaler Daten.

LCFS *last come, first served*, (dt.) wer zuletzt kommt, mahlt zuerst; Reihungsverfahren, gleichbedeutend mit ↑*LIFO*.

Leerlauf Zustand der Untätigkeit, nämlich wenn auf einem ↑*Prozessor* kein ↑*Prozess* stattfindet. Entweder wird der Prozessor gerade eben durch einen auf einem anderen Prozessor stattfindenden Prozess hochgefahren, durchläuft seine Initialisierungsprozedur, und hat vom

↑*Planer* noch keinen Prozess zugewiesen bekommen (↑*Multiprozessor*) oder im Moment der Blockierung des auf ihm stattfindenden Prozesses steht kein anderer Prozess zur ↑*Einlastung* mehr bereit. Der Prozessor kann nur noch durch einen externen Prozess aus diesen Zustand befreit werden, indem ihm „von außen“ ein Prozess zur Einlastung bereitgestellt wird. Dies kann durch eine ↑*Unterbrechungsanforderung* geschehen, die die Deblockierung eines blockierten Prozesses bewirkt oder, vorausgesetzt der Prozessor ist aktiv untätig, indem er auf der ↑*Bereitliste* plötzlich einen Prozess vorfindet, den der Planer von einem anderen Prozessor aus dort abgelegt hat. Im letzteren Fall ist ein ↑*Leerlaufprozess* aktiv wartend darauf, dass die Bereitliste wieder gefüllt wird (↑*busy waiting*). Normalerweise jedoch wird ein Prozessor in solch einer Situation in den ↑*Schlafzustand* versetzt, um nicht unnötig das ↑*Rechensystem* zu strapazieren, Energie zu verbrauchen, Abwärme zu produzieren — damit Unkosten zu verursachen und die Umwelt zu belasten.

Leerlaufprozess ↑*Prozess*, der den ↑*Leerlauf* von einem ↑*Prozessor* kontrolliert; der zwar logisch untätig ist, aber physisch für das System tätig sein kann. Es gibt zwei grundsätzlich verschiedene Modelle für einen solchen Prozess. In dem einen Fall übernimmt immer jener Prozess die Rolle der Leerlaufkontrolle, der jüngst zurückliegend gerade blockiert ist und in dem Moment keinen anderen für seinen Prozessor lauffähigen Prozess in der ↑*Bereitliste* vorfindet. Damit kann ein beliebiger Prozess in diese Rolle gebracht werden, die Identität des leerlaufenden Prozesses ist nicht eindeutig bestimmt. Jedoch eröffnet sich so die Option, dem physischen ↑*Prozesswechsel* vorzubeugen, wenn nämlich der Prozess in seiner Leerlaufphase deblockiert wird. Zu beachten ist, dass lediglich ein logischer Prozesswechsel vollzogen wird, nämlich vom ↑*Prozesszustand* laufend nach untätig: ein Wechsel der ↑*Prozessinkarnation* erfolgt nicht. In dem anderen Fall steht für den Leerlauf eine eigene Prozessinkarnation zur Verfügung. Der physische Prozesswechsel ist damit zwingend und erfolgt immer dann, wenn im Moment der Blockierung eines Prozesses die Bereitliste keinen anderen Prozess für den Prozessor enthält. In dem Modell zieht der Leerlauf immer zwei physische Prozesswechsel nach sich: erstens vom blockierenden zum leerlaufenden Prozess und zweitens vom leerlaufenden zum bereitgestellten Prozess. Dies auch dann, wenn der bereitgestellte Prozess sich als derjenige erweist, der vorher zum leerlaufenden Prozess gewechselt ist.

leichtgewichtiger Prozess ↑*Prozess*, der als ↑*Systemkernfaden* mit anderen Prozessen seiner Art zusammen im gemeinsamen und durch ↑*Speicherschutz* isolierten ↑*Adressraum* stattfindet.

level-triggered interrupt (dt.) pegelgesteuerte ↑*Unterbrechung*, ↑*Pegelsteuerung*.

LIFO *last in, first out*, (dt.) der umgekehrten Reihe nach; Lagerungsverfahren, gleichbedeutend mit ↑*LCFS*.

light-weight process (dt.) ↑*leichtgewichtiger Prozess*.

Linux Mehrplatz-/Mehrbenutzerbetriebssystem, von ↑*UNIX* abstammend. Erste Installation September 1991 (Intel 80386), programmiert in ↑*C*.

load module (dt.) Lademodul.

location counter (dt.) Adresszähler.

logiche Synchronisation Synonym für ↑*unilaterale Synchronisation*.

Mantelprozedur Umhüllung für ein ↑*Unterprogramm*, um es in andere (eigentümliche) Software zu integrieren. Beispiel ist etwa ein in ↑*Assembliersprache* zu formulierender Aufruf eines in ↑*C* vorliegenden Unterprogramms zur ↑*Unterbrechungsbehandlung*, der entsprechend ↑*Aufrufkonvention* die Inhalte der im Unterprogramm frei verwendbaren Register (↑*nichtflüchtiges Register*, *called-saved*) sichert und wiederherstellt. Allgemein jede Art von Software zur Adaptation der formalen Schnittstelle eines Unterprogramms.

Markierungsbit Boole'sche Binärziffer, wobei der Ziffernwert einen bestimmten Zustand widerspiegelt, den es festzuhalten gilt. Auch kurz als *Merker* bezeichnet.

Maschinenbefehl Elementare Anweisung in einem ↑*Maschinenprogramm*. Eine solche Anweisung besteht aus einen obligatorischen *Operationsteil*, der die ↑*Aktion* der Maschine bezeichnet, und einen optionalen *Operadenteil*, der diese Aktion mit den von der Maschine zu verarbeitenden Informationen verknüpft. Letzterer weist je nach Maschinenart eine unterschiedliche Anzahl und Form von Adressangaben auf: 0-Adressbefehl, ausschließlich implizite Adressierung der Operanden (Stapelmaschine); 1-Adressbefehl, implizite Adressierung des ersten und explizite Adressierung des zweiten Operanden (Akkumulatormaschine); 2-Adressbefehl, explizite Adressierung beider Operanden, wobei ein Operand gleichzeitig Quell- und Zieloperand ist (↑*CISC*); 3-Adressbefehl, explizite Adressierung der beiden Quell- und des einen Zieloperanden (↑*RISC*).

Maschinenkode Verschlüsselung eines ↑*Maschinenbefehls*. Üblich ist die Darstellung eines solchen Befehls als Hexadezimalzahl: so bedeutet $\texttt{05}_{16}$ bei einem x86-kompatiblen Prozessor die Addition mit dem Akkumulator. Früher war auch die Darstellung als Oktalzahl verbreitet (Rechner der PDP-Familie).

Maschinenprogramm ↑*Programm*, das zum Ablauf oberhalb eines Betriebssystems bestimmt ist. Ein solches Programm besteht aus Instruktionen (↑*Maschinenbefehle*) des Prozessors, die *explizit* auch Aktionen (↑*Systemaufrufe*) eines Betriebssystems auslösen können. Das Programm ist zudem *implizit* abhängig von einem Betriebssystem, wenn es von der Gültigkeit eines Systemzustands ausgeht, die es nicht selbst durch Systemaufrufe anfordert, sondern zu seiner ↑*Ladezeit* vom Betriebssystem zugesichert wird und zur ↑*Laufzeit* bestehen bleibt.

Maschinensprache Programmiersprache, deren Sprachelemente in Form von ↑*Maschinenkode* repräsentiert sind. Die eigentliche Programmiersprache einer ↑*CPU*.

Maschinenwort Adressierbare Informationseinheit eines Prozessors (↑*CPU*). Allgemein ausgelegt als Bitvektor, dessen Länge die ↑*Wortbreite* des Prozessors definiert.

Massenspeicher ↑*Speicher* zur dauerhaften Ablage sehr großer Mengen von Daten aller Art. Oft Synonym für Sekundärspeicher. Umfasst jedoch auch Tertiärspeicher, der nicht permanent im Rechensystem angeschlossen ist und sich in Form von Archiven zeigt.

MCU *microcontroller unit*, (dt.) Mikrokontrollereinheit.

Mehrfädigkeit Fähigkeit einer (realen/virtuellen) Maschine mehr als einen ↑*Faden* zugleich durch ↑*Parallelverarbeitung* ausführen zu können. Grundlage bildet ein ↑*nichtsequentielles Programm*, das die einem jeden Faden zukommende ↑*Aufgabe* beschreibt.

Mehrkernprozessor ↑*CPU*, die aus mehr als einen Rechenkern besteht.

Mehrprogrammbetrieb ↑*Betriebsart*, bei der eine (reale/virtuelle) Maschine räum- und zeitlich von mehr als einem ↑*Machinenprogramm* zugleich benutzt werden kann.

mehrseitige Synchronisation ↑*multilaterale Synchronisation*.

MMU *memory management unit*, (dt.) Speicherverwaltungseinheit.

Mnemon (gr.) erinnern.

Modul Softwareeinheit mit eigenem Datenmodell und eigenem Satz von Operationen darauf. Die Daten sind nur über die moduleigenen Operationen zugänglich (*information hiding*).

Monitor Datentyp, Klasse mit impliziten Eigenschaften zur ↑*Synchronisation*; Programmierkonvention. Für alle Operationen, die auf ein ↑*Exemplar* eines solchen Datentyps angewendet werden, gilt ↑*wechselseitiger Ausschluss*. Diese Operationen müssen in der Datentypschnittstelle spezifiziert sein. Per Definition ist jedes ↑*Unterprogramm*, das eine solche Operation implementiert, ein ↑*kritischer Abschnitt*. Innerhalb eines solchen Abschnitts kann ein ↑*Prozess* eine ↑*Bedingungsvariable* verwenden, um sich auf ein ↑*Ereignis* zu synchronisieren beziehungsweise ein solches anzuzeigen.
Ursprünglich ein Konzept der *Typisierung* in höheren Programmiersprachen, indem nämlich die korrekte Verwendung von Operationen zur Synchronisation durch einen ↑*Kompilierer* abprüfbar wird und so Programmierfehler vermieden werden können. Die zum wechselseitigen Ausschluss erforderlichen Anweisungen erzeugt der Kompilierer, wie auch die Instruktionen, um einen kritischen Abschnitt zeitweilig verlassen zu können, ohne diesen in der Zwischenzeit gesperrt zu halten. Mangels Sprachunterstützung ist dieses Konzept jedoch viel mehr zur Programmierkonvention entartet, das heißt, dem Menschen als ↑*Prozessor* wird es erleichtert, ein (vermeintlich) korrektes ↑*nichtsequentielles Programm* zu formulieren.

Multics Mehrplatz-/Mehrbenutzerbetriebssystem, Abkürzung für „*Multiplexed Information and Computing Service*“. Konzeptpapiere im Jahr 1965, erste echte Installation Dezember 1967 (GE 645), programmiert in EPL (*early* ↑*PL/1*), in Betrieb bis Oktober 2000.

multilaterale Synchronisation ↑*Synchronisation*, die sich auf jeden beteiligten ↑*Prozess* auswirken kann. Jeder der betroffenen Prozesse bestreitet dasselbe Protokoll, um Synchronisation in Bezug auf eine bestimmte ↑*Aktion* oder ↑*Aktionsfolge* zu erzielen. Kommt hierzu ↑*blockierende Synchronisation* zur Geltung, wird jeder dieser Prozesse durch das Protokoll blockiert werden können. Für ↑*nichtblockierende Synchronisation* ist davon auszugehen, dass dieses Protokoll jeden der beteiligten Prozesse die Aktion/Aktionsfolge wiederholen lässt. Diese Art der Synchronisation ist typisch, wenn Prozesse gleichzeitig ein ↑*wiederverwendbares Betriebsmittel* anfordern, das jedoch nur exklusiv belegt und genutzt werden darf. Eine besondere Variante davon ist ein ↑*kritischer Abschnitt*, für den ↑*wechselseitiger Ausschluss* von Prozessen sicherzustellen ist. Alle diese Verfahren wirken mehrseitig.

Multiplexverfahren Methode, deren Urspung in der Nachrichtentechnik liegt, nämlich um mehrere Signale (↑*konsumierbares Betriebsmittel*) simultan über ein Medium (↑*wiederverwendbares Betriebsmittel*) übertragen zu können. In einem ↑*Betriebssystemkern* findet diese Methode Anwendung, um mehr als einen ↑*Prozess* zugleich auf dem ↑*Prozessor* einer (realen/virtuellen) Maschine stattfinden lassen zu können. Grundlage dafür bilden entsprechende Verfahren zur ↑*Prozesseinplanung*, die ↑*Simultanverarbeitung* ermöglichen.

multiprocessing (dt.) Simultanverarbeitung.

multiprocessing (dt.) Simultanverarbeitung.

multiprogramming (dt.) ↑*Mehrprogrammbetrieb*.

Multiprozessor Prozessorsystem, das aus mehr als einer ↑*CPU* besteht.

multitasking (dt.) ↑*Mehrprogrammbetrieb*, wobei ↑*Programm* hier zu differenzieren ist. Im Vordergrund steht die ↑*Aufgabe*, die eben durch ein Programm beschrieben wird. Jedes Programm kann entweder nur eine oder mehrere Aufgaben beschreiben. Sind es mehrere Aufgaben, können diese von derselben oder verschiedenen Arten (Typen) sein. Zudem steht die wirkliche Ausprägung einer Aufgabe nicht im Vordergrund: eine Aufgabe kann einen eigenständigen ↑*Handlungsstrang* darstellen oder teilt sich einen solchen mit anderen Aufgaben.

multithreading (dt.) ↑*Mehrfädigkeit*.

Mutex Kunstwort, von ↑*mutual exclusion*; Sperrmechanismus, mit dem ↑*wechselseitiger Ausschluss* ausgeübt wird. Der Mechanismus sichert zu, dass nur der ↑*Prozess* die Sperre aufheben kann, der diese zuvor auch gesetzt hat. Typischerweise wird mit diesem Konstrukt ein

↑*kritischer Abschnitt* eingefasst. Einen damit eingefassten Abschnitt kann nur der Prozess entsperren, der diesen zuletzt sperrte, betrat und jetzt verlässt. Für jeden anderen Prozess scheitert die ↑*Aktion* — was in dem Fall auf einen Programmierfehler hindeutet und eigentlich eine ↑*Ausnahmesituation* darstellt, im Allgemeinen jedoch ohne Auswirkung für den offensichtlichen fehlgeleiteten Prozess bleibt. Grundlage für die Implementierung kann ein ↑*binärer Semaphor* sein: In der Sperroperation (`lock`) wird nach dem ↑*P* der gegenwärtige Prozess vermerkt, dessen Identität in der Entsperroperation (`unlock`) noch vor dem ↑*V* mit demjenigen Prozess überprüft wird, der den kritischen Abschnitt verlässt:

```
void lock(mutex) {
    P(mutex.sema);
    mutex.owner = self();
}
```

```
void unlock(mutex) {
    if (mutex.owner != self())
        raise(EPERM);
    mutex.owner = NULL;
    V(mutex.sema);
}
```

In dem Beispiel liefert `self()` entweder eine ↑*PID* oder den ↑*Prozesszeiger*, und mit `raise()` wird eine ↑*Ausnahme* erhoben, nämlich dass der gegenwärtige Prozess eine für ihn unerlaubte Operation durchführt. Dieses Beispiel geht davon aus, dass die ↑*Ausnahmebehandlung* richtigerweise die Termination des fehlgeleiteten Prozesses erzwingt und damit die Ausführung der gescheiterten Entsperroperation nicht wieder aufnimmt.

mutual exclusion (dt.) gegenseitiger oder ↑*wechselseitiger Ausschluss*.

Nebenläufigkeit Verhältnis von nicht kausal abhängigen Ereignissen, die sich also nicht beeinflussen. Ein Ereignis ist nebenläufig zu einem anderen, wenn es im Anderswo (d.h., weder in der Zukunft noch in der Vergangenheit) des anderen Ereignisses liegt, weder Ursache noch Wirkung ist: wenn zu dem anderen Ereignis keine Daten-, Kontrollfluss- oder Zeitabhängigkeit besteht.

Nebenläufigkeitssteuerung Verfahren, das trotz ↑*Nebenläufigkeit* einer ↑*Aktion* oder ↑*Aktionsfolge* die Entwicklung korrekter Berechnungen sicherstellt. Dabei kann eine ↑*optimistische* und ↑*pessimistische* Herangehensweise praktiziert werden.

nichtblockierende Synchronisation Gegenteil von ↑*blockierende Synchronisation*: jeder ↑*Prozess* versucht ungehindert, eine bestimmte ↑*Aktion* oder ↑*Aktionsfolge* durchzuführen, auch wenn dies gleichzeitig geschieht. Dabei darf die Berechnung dieser Aktion/Aktionsfolge dann jedoch nur lokale Signifikanz für jeden der beteiligten Prozesse haben. Um globale Signifikanz zu erreichen und damit eine Zustandsänderung für alle Prozesse sichtbar zu machen, ist jeder beteiligte Prozess verpflichtet, seine lokale Berechnung nach außen zu bestätigen. Die Bestätigung gelingt nur, wenn kein anderer Prozess diese zwischenzeitig bereits erreicht hat. Anderenfalls scheitert die Bestätigung und der betroffene Prozess wiederholt die Aktion/Aktionsfolge. Auch als ↑*optimistische Nebenläufigkeitssteuerung* bezeichnet.

nichtflüchtiges Register Konzept der ↑*Aufrufkonvention*: der Inhalt eines solchen Prozessorregisters gilt als beständig. Vorkehrungen zur Sicherung und Wiederherstellung des Registerinhalts werden im ↑*Unterprogramm* bei Bedarf getroffen (*callee-saved*).

nichtsequentieller Prozess ↑*Prozess*, der ein ↑*nichtsequentielles Programm* zur Grundlage hat. Bei ↑*Parallelverarbeitung* kann eine ↑*Aktion* oder ↑*Aktionsfolge* in solch einem Prozess zeitlich überlappt und dadurch möglicherweise beschleunigt vonstattengehen. Jedoch birgt dies auch die Gefahr einer ↑*Wettlaufsituation*, deren Vorbeugung wiederum eine Geschwindigkeitseinbuße mit sich bringt.

nichtsequentielles Programm ↑*Programm*, das Konstrukte zur Formulierung explizit paralleler Abläufe verwendet. Die Konstrukte können in der Programmiersprache, in der das Programm formuliert wurde, enthalten sein oder werden sprachunabhängig durch eine Programmbibliothek (z.B. *POSIX Threads*, *pthreads*; aber auch UNIX Signale, `signal(3)`) bereitgestellt.

NMI *non-maskable interrupt*, (dt.) nichtmaskierbare Unterbrechung.

nonvolatile register (dt.) ↑*nichtflüchtiges Register*.

NVRAM *non-volatile RAM*, (dt.) nichtflüchtiger ↑*RAM*.

object module (dt.) Objektmodul.

Objective-C objektorientierte, imperative Programmiersprache (1981), Obermenge von ↑*C*.

Objekt erster Klasse Exemplar eines bestimmten Bautyps, das im ↑*Arbeitsspeicher* abgelegt (d.h., einer Variablen zugewiesen) werden, ↑*tatsächlicher Parameter* beziehungsweise Funktionsergebnis sein oder zur ↑*Laufzeit* erstellt werden kann und eine eigene Identität besitzt. Für solch ein Objekt oder Konstrukt gibt es in seinem Bezugssystem (d.h., ↑*Programm*) keine Einschränkung in der Erzeugung oder Nutzung.

Objektmodul Eingabedatei eines Binders, Ausgabedatei eines Assemblierers. Neben Programmtext oder -daten, ist die ↑*Symboltabelle* wichtigstes Merkmal dieser Datei.

Operationskode Identifikation für einen Befehl einer (realen/virtuellen) Maschine. Beispielsweise der ↑*Maschinenkode*, der den ↑*Maschinenbefehl* einer CPU spezifiziert. Allgemein aber die Nummer des Befehls, der von einer Maschine durchgeführt werden soll.

Operationsprinzip Definition des funktionellen Verhaltens der Architektur einer (realen/virtuellen) Maschine durch Festlegung einer ↑*Informationsstruktur* für diese Maschine und einer ↑*Kontrollstruktur* des Programmablaufs.

optimistische Nebenläufigkeitssteuerung Gegenteil von ↑*pessimistische Nebenläufigkeitssteuerung*: jeder ↑*Prozess* lässt die kritische ↑*Aktion* oder ↑*Aktionsfolge* als eine ↑*Transaktion* stattfinden. Scheitert die Transaktion für einen Prozess, wird sie von ihm wiederholt bis sie gelingt oder der Prozess eine ↑*Ausnahme* erhebt. Typisches Beispiel für solch ein Ablaufmuster ist die ↑*nichtblockierende Synchronistation*. Dem Ansatz liegt die Annahme zugrunde, wonach ein Prozess mit anderen Prozessen nicht in Konflikt gerät, da eine gemeinsame und gleichzeitige Transaktion eher unwahrscheinlich ist.

Ortstransparenz Eigenschaft, durch die einem Subjekt (↑*Prozess*) der tatsächliche Ort eines Objekts verborgen bleibt. Der für den Zugriff verwendete Name enthält keinen Hinweis darüber, ob das Objekt sich mit dem Subjekt denselben ↑*Adressraum*, Prozessor(kern) oder Rechner teilt. Ein solcher Name repräsentiert eine ↑*symbolische Adresse*.

overhead (dt.) ↑*Betriebslast*.

overlay (dt.) ↑*Überlagerung*.

P Abk. für (hol., Kunstwort) *prolaag*, für (dt.) erniedrigen; synonym mit (en.) *down*, *wait*, *acquire*.

page (dt.) Seite.

page fault (dt.) Seitenfehler.

pager (dt.) Seitenabrufer.

paging (dt.) Seitenumlagerung.

panic (dt.) ↑*Panik*.

Panik Gefahr für das Rechensystem bedeutende und im ↑*Betriebssystem* aufgetretene ↑*Ausnahmesituation*, die nicht behandelbar ist und den sofortigen Stillstand bewirkt.

paralleler Prozess ↑*nichtsequentieller Prozess*.

paralleles Programm ↑*nichtsequentielles Programm*.

Parallelität ↑*Nebenläufigkeit*, wenn in einem ↑*Rechensystem* die Unabhängigkeit von Vorgängen gegeben ist. Dabei ist ein solcher Vorgang ein ↑*Prozess*, der durch Vervielfachung oder Mehrfachnutzung von einem ↑*Prozessor* echt- oder pseudoparallel zu anderen Prozessen stattfindet. Voraussetzung dabei ist die Fähigkeit zur ↑*Parallelverarbeitung* durch ein ↑*Betriebssystem*.

Parallelverarbeitung Betriebsart eines Rechensystems, durch die mehrere Abläufe desselben Programms oder verschiedener Programme parallel stattfinden können. Erreicht wird dies durch Vervielfachung oder Mehrfachnutzung des Prozessors, das heißt, räumliche oder zeitliche Partitionierung seiner Verarbeitungseinheiten. Ersteres begründet beispielsweise einen ↑*Multiprozessor* oder ↑*Mehrkernrozessor*, letzteres ist durch ↑*partielle Virtualisierung* eben nur der Verarbeitungseinheit erreichbar. Sind mehrere Abläufe in demselben Programm möglich, wird letzteres als *nichtsequentiell* bezeichnet.

partielle Interpretation Interpretation der Anweisung einer (realen/virtuellen) Maschine durch verschiedene Interpreter, die in einer ↑*funktionalen Hierarchie* zueinander angeordnet sind. In diesem arbeitsteiligen und strikt stapelorientierten Vorgang kooperieren Interpreter, die einerseits eine reale und andererseits wenigstens eine ↑*virtuelle Maschine* implementieren. Dabei definiert der Interpreter der realen Maschine die unterste Ebene in der Hierarchie, er startet den ↑*Abruf- und Ausführungszyklus* zur Interpretation. Eine ↑*Ausnahmesituation*, die zur ↑*Unterbrechung* der gegenwärtigen ↑*Aktion* auf tieferer Ebene führt, kann bewirken, dass ein Interpreter auf nächst höherer Ebene gestartet und somit eine virtuelle Maschine aktiviert wird. Dieser nachgeschaltete Interpreter läuft als Teil der ↑*Ausnahmebehandlung*, er sorgt letztlich dafür, dass die unterbrochene Aktion fortgesetzt wird. Entweder vollendet er diese Aktion, beendet damit auch die durch ihn bereitgestellte virtuelle Maschine, oder er leitet seinerseits eine Ausnahmebehandlung ein, aktiviert damit eine weitere virtuelle Maschine (Rekursion). Beendigung einer virtuellen Maschine bedeutet in solch einem Szenario, dass die (reale/virtuelle) Maschine, die zuvor aktiv war, die Kontrolle über die weitere Programmausführung zurück erhält. Typische Beispiele für diese Art der Interpretation sind der ↑*Systemaufruf* und ein ↑*Seitenfehler*, mit dem ↑*Betriebssystem* als Interpreter. Andere Beispiele liefern ↑*sensitive Befehle*, die durch einen ↑*Hypervisor* interpretiert werden. In jedem dieser Fälle fährt zeitweilig eine virtuelle Maschine hoch, um unterbrochene Aktionen zu vollenden und sich dann alsbald wieder abzuschalten.

partielle Virtualisierung Mehrfachnutzung einer ausgewählten Hardwareeinheit eines Rechensystems durch ein Zeitmultiplexverfahren. Ursprünglich nur bezogen auf die ↑*CPU*, um nämlich einem ↑*Prozess* die Illusion von einem eigenen Prozessor zu geben. Technische Grundlage dafür sind Systemfunktionen zur ↑*Simultanverarbeitung*. So können mehrere Prozesse zugleich stattfinden, indem jedem ein eigener virtueller Prozessor zur Verfügung steht. Das Konzept ist aber übertragbar auf andere Hardwareeinheiten, insbesondere dem ↑*Hauptspeicher*. Hier kann durch zeitweilige ↑*Umlagerung* ganzer Programme in den ↑*Hintergrundspeicher* derselbe Hauptspeicherbereich zur zeitweiligen Mehrfachnutzung durch mehrere Programme bereitgestellt werden.

PC *program counter* (dt.) ↑*Befehlszähler* oder *personal computer* (dt.) Arbeitsplatzrechner.

PCB *process control block*, (dt.) ↑*Prozesskontrollblock*.

Pegelsteuerung Auslöser für die ↑*Unterbrechung* ist eine Mindestzeit an Beständigkeit des Pegelstands auf der Störleitung (*interrupt line*) zur ↑*CPU*. Für eine ↑*Unterbrechungsanforderung* erniedrigt (erhöht) die ↑*Peripherie* den Pegelstand und hält diesen Zustand, bis der ↑*Unterbrechungshandhaber* der anfordernden Peripherie die Unterbrechung bestätigt. Im Moment dieser Bestätigung, und zwar an dem entsprechenden ↑*Peripheriegerät*, wird der dort eingestellte Pegelstand zurückgenommen. Stellt dasselbe Gerät sofort nach der Bestätigung und noch während die ↑*Unterbrechungsbehandlung* läuft eine weitere Anforderung, kommt diese

normalerweise implizit bei der Rückkehr aus der Unterbrechungsbehandlung zur Geltung — oder auch bereits vorher, wozu der Unterbrechungshandhaber jedoch die für die CPU in dem Moment gültige ↑*Unterbrechungsprioritätsebene* durch einen ↑*privilegierten Befehl* explizit herabsetzen muss. Damit werden auch wiederholte Anforderungen (*reassertion*) nicht verpasst, im Gegensatz zur ↑*Flankensteuerung*. Jedoch darf der Unterbrechungshandhaber die Bestätigung nicht auslassen, da er sonst bei Rückkehr aus der Unterbrechungsbehandlung sofort wieder aufgerufen wird, damit seinen erneuten Aufruf nicht etwa mit einer weiteren Unterbrechung in Verbindung bringen kann und, was viel gravierender ist, der unterbrochene ↑*Prozess* womöglich niemals fortgesetzt wird. Auslassen der Bestätigung einer pegelgesteuerten Unterbrechung kann ↑*Panik* auslösen.

Performanz Leistungsfähigkeit eines einzelnen Systembestandteils, eines Subsystems oder eines ganzen Systems, Hardware oder Software.

peripheral (dt.) Peripheriegerät.

Peripherie Gesamtheit der an einer ↑*Zentraleinheit* angeschlossenen Geräte, jedes davon auch als ↑*Peripheriegerät* bezeichnet.

Peripheriegerät Steuerung durch einen ↑*Prozessor* benötigendes Gerät eines Rechensystems. Jede Form von Ein-/Ausgabegerät (z.B. Tastatur, Bildschirm, Maus, Platte; Sensoren, Aktoren), aber auch Erweiterungskarten beziehungsweise Anschlüsse zur seriellen/parallelen Ein-/Ausgabe von Daten.

persistentes Betriebsmittel ↑*dauerhaftes Betriebsmittel*.

pessimistische Nebenläufigkeitssteuerung ↑*Nebenläufigkeitssteuerung* unter der Annahme, dass ein ↑*Prozess* mit wenigstens einem anderen Prozess wahrscheinlich in Konflikt gerät, wenn eine gemeinsame ↑*Aktion* oder ↑*Aktionsfolge* gleichzeitig stattfindet. Dem möglichen Konflikt wird vorgebeugt, indem die Prozesse die betreffende Aktion/Aktionsfolge nur als Ganzes nacheinander durchführen können. Dies lässt sich durch ↑*blockierende Synchronisation* erreichen oder auf Basis einer ↑*Ablaufplanung*, die die Prozesse (in der kritischen Phase) strikt nacheinander stattfinden lässt.

PIC *programmable interrupt controller*, (dt.) programmierbare Unterbrechungssteuereinheit.

PID *process identifaction*, (dt.) ↑*Prozessidentifikation*.

PL/1 *programming language # 1*, prozedurale, imperative Programmiersprache (1964).

Planer ↑*Programm* zur Erstellung und Verwaltung von einem ↑*Ablaufplan*, das nach zwei verschiedenen Modellen zum Einsatz kommt: vor (*off-line*, statisch) oder zur (*on-line*, dynamisch) ↑*Laufzeit* der Abarbeitung des Ablaufplans. Auch eine Kombination beider Varianten ist möglich. In dem Fall reflektiert der statische Ablaufplan für jeden einzelnen darin aufgeführten ↑*Prozess* Vorwissen zu Daten- und Kontrollflussabhängigkeiten, das dem ↑*Betriebssystem* initial für eine Gruppe von Prozessen übergeben wird. Zur Laufzeit wird dieser Plan dann entsprechend der dynamischen Vorgänge im ↑*Rechensystem* fortgeschrieben. Der statische Ansatz findet bevorzugt bei Spezialsystemen Verwendung, sofern das benötigte Vorwissen vorhanden ist und die Aufstellung des Plans in angemessener Zeit möglich ist: die Erstellung eines solchen Plans ist typischerweise ein NP-schweres Entscheidungsproblem, das in Abhängigkeit von der Prozessanzahl wie auch dem Grad der Prozessabhängigkeiten gegebenenfalls eine unvertretbar hohe Berechnungszeit bedingt. Demgegenüber ist der dynamische Ansatz in Spezial- und Universalsystemen verbreitet, in letzteren wegen dem dort oft nicht vorhandenem Vorwissen eben auch unabdinglich. In diesem Fall der (mit den zu planenden Prozessen) im ↑*Betriebssystem* mitlaufenden Ablaufplanung wird letztere als ↑*Unterprogramm* aufgerufen, wann immer Gründe für eine Aktualisierung des Plans vorliegen. Diese Gründe sind die Zulassung, Bereitstellung, Blockierung, Unterbrechung, Verdrängung, Suspendierung oder Beendigung eines Prozesses.

preemption (dt.) ↑*Verdrängung*.

pre-paging (dt.) Seitenvorabruf.

privater Semaphor ↑*Semaphor*, bei dem ↑P nur für denjenigen ↑*Prozess* möglich ist, der Eigentümer dieses Semaphors ist. Demgegenüber ist ↑V jedem anderen Prozess möglich. Originär ein ↑*allgemeiner Semaphor* mit Wertebereich $[-1, 1]$, implizit vorbelegt mit dem Wert 0. Alternativ kann auch ein ↑*binärer Semaphor* die Basis bilden, mit `false` als Initialwert. Hauptzweck dieses Semaphors besteht in der ↑*Synchronisation* eines Prozesses auf ein ↑*Ereignis*, das den weiteren Verlauf des Prozesses bestimmt. Das Ereignis kann von einem beliebigen Prozess, auch ihm selbst, signalisiert werden, oft als Ergebnis der bei einer Berechnung erreichten Zustandsänderung. Varianten erlauben die Speicherung von mehr als einem Ereignis und umfassen demzufolge den Wertebereich $[-1, n]$, mit $n \geq 1$.

privilegierter Befehl ↑*Maschinenbefehl*, dessen direkte Ausführung durch eine (reale/virtuelle) Maschine nur bedingt tolerierbar ist. Ein ↑*sensitiver Befehl*, wenn die (insbesondere reale) Maschine, die ihn ausführen soll, in einem unprivilegierten ↑*Betriebsmodus* läuft.

process descriptor (dt.) Prozessdeskriptor, ↑*Prozesskontrollblock*.

process scheduling (dt.) ↑*Prozesseinplanung*.

process table (dt.) ↑*Prozesstabelle*.

Programm Festlegung einer Folge von Anweisungen für einer ↑*Prozessor*, nach der die zur Bearbeitung einer (durch einen Algorithmus wohldefinierten) Handlungsvorschrift erforderlichen Aktionen stattfinden sollen; Konkretisierung eines Algorithmus. Von statischer Natur, erst durch den Prozessor kommen die Anweisungen zur Ausführung.

Programmablauf ↑*Prozess*.

Programmiermodell Der in einem ↑*Programm* direkt sicht-, nutz- oder programmierbare ↑*Registersatz* eines Prozessors.

PROM *programmable ROM*, (dt.) programmierbarer ↑*ROM*.

Prozedurfernaufruf Aufruf, der den aktuellen ↑*Handlungsstrang* verlässt und ein Unterprogramm in einem gegebenenfalls anderen, dem ↑*Speicherschutz* unterliegenden ↑*Adressraum* zur Ausführung bringt. Ursprünglich ein Konzept ausschließlich für kooperiende Programme, die verteilt auf einem Rechnernetz ablaufen. In abgewandelter Form jedoch auch anwendbar zur lokalen Kommunikation zwischen Handlungssträngen desselben oder verschiedener Adressräume desselben Rechners.

Prozess ↑*Programm* in Ausführung durch einen ↑*Prozessor*. Ein sich über eine gewisse Zeit erstreckender Programmablauf, eine ↑*Aktionsfolge*. Von dynamischer Natur, bestimmt durch das Programm und den erst zur ↑*Laufzeit* bekannten Eingabedaten.

Prozessadressraum ↑*Adressraum* von einem ↑*Prozess*. In Abhängigkeit von der ↑*Betriebsart* unterliegt dieser Adressraum gegebenenfalls dem ↑*Speicherschutz*.

Prozessdomäne Herrschaftsbereich von einem ↑*Prozess*, definiert durch das ihn bestimmende ↑*Programm* und repräsentiert durch seinen ↑*Adressraum*. Bei zusätzlichem ↑*Speicherschutz* (optional) ist es dem Prozess unmöglich, seine Domäne unkontrolliert zu verlassen und in die eines anderen Prozesses einzudringen.

Prozesseinplanung ↑*Ablaufplanung*, die auf Grundlage einer ↑*Prozessinkarnation* operiert und (relativen) Zeitpunkt wie auch Reihenfolge der ↑*Einlastung* der ↑*CPU* damit festlegt.

Prozessidentifikation Nummer (auch: Name) von einem ↑*Prozess*.

Prozessinkarnation ↑*Exemplar* von einem ↑*Prozess*. Dem ↑*Adressraum* des Prozesses ist (virtueller) ↑*Speicher* und dem Prozess ein ↑*Laufzeitkontext* zugewiesen.

Prozesskontrollblock Datenstruktur zur Beschreibung einer ↑*Prozessinkarnation*; ↑*PCB*. Zentrales Objekt, um für einen ↑*Prozess* alle von ihm belegten oder benötigten ↑*Betriebsmittel*, seinen ↑*Prozesszustand*, seinen ↑*Laufzeitkontext*, seine Rechte und seine Verwandtschaftsbeziehungen zu anderen Prozessen zu erfassen. Muss ein Prozess auf die Zuteilung eines Betriebsmittels oder aus anderen Gründen warten, erfasst der PCB den Grund des Wartens und enthält gegebenenfalls auch entsprechende Attribute für seine Platzierung auf einer ↑*Warteschlange*. Um einen Prozess erzeugen, das heißt, eine Prozessinkarnation einrichten zu können, muss das ↑*Betriebssystem* noch einen PCB im ↑*Hauptspeicher* belegen können. Für jeden existierenden Prozess gibt es eine solche Datenstruktur. Typischerweise sind diese alle in der ↑*Prozesstabelle* des Betriebssystems zusammengefasst.

Prozessor Verarbeitungseinheit; aktive Komponente eines Rechensystems, die eine Transformation auf ihren Daten vornimmt. Die Vorschrift für diese Transformation liefert ein ↑*Programm*, das in Form von Hard-, Firm- oder Software vorliegt.

Prozessorauslastung Zeitanteil der produktiven Arbeit von einem ↑*Prozessor*. Unproduktiv ist der Prozessor typischerweise, wenn er sich im ↑*Leerlauf* befindet.

Prozessorstatus Zustand eines Prozessors entsprechend dem ↑*Programmiermodell* der ↑*CPU*.

Prozesstabelle Feld begrenzter Länge im ↑*Betriebssystem*, wobei jedes Feldelement ein ↑*Prozesskontrollblock* ist. Normalerweise eine statische Datenstruktur, deren Größe (d.h., Anzahl der Feldelemente) ein *Systemparameter* zur Konfigurierung des Betriebssystem bildet. Ist jeder einzelne Tabelleneintrag für eine ↑*Prozessinkarnation* belegt, kann ein weiterer ↑*Prozess* erst dann erzeugt und eingerichtet werden, wenn ein anderer Prozess terminiert, seine Prozessinkarnation gelöscht und dadurch ein Tabelleneintrag frei wurde.

Prozesswechsel ↑*Aktionsfolge*, die den aktuellen ↑*Prozess* aus- und einem ausgewählten Prozess einsetzt. Die Aktionsfolge sichert den ↑*Laufzeitkontext* des aktuellen Prozesses und stellt dann den Laufzeitkontext des einzusetzenden Prozesses wieder her. Darüberhinaus ist der ↑*Prozesszeiger* zu aktualisieren: je nach Implementierung der Operation zum Wechseln des Laufzeitkontextes ist diesem Zeiger die Adresse vom ↑*PCB* des einzusetzenden (davor) beziehungsweise eingesetzten (danach) Prozesses zuzuweisen. Normalerweise erfolgt der Wechsel vom aktuellen hin zu einen anderen Prozess. Jedoch kann es auch Situationen geben, wo beide Prozesse identisch sind: beispielsweise wenn der aktuelle Prozess als ↑*Leerlaufprozess* agiert, also logisch blockierte, jedoch durch seine Deblockierung zwischenzeitlich wieder bereitgestellt und vom ↑*Planer* als nächster einzusetzender Prozess ausgewählt wurde. Ob ein Prozess zu sich selbst wechseln kann, hängt ganz von der Schnittstelle und der Implementierung von dem ↑*Unterprogramm* ab, das typischerweise für solch einen Wechsel aufzurufen ist. Dieses Unterprogramm benötigt dann zwei Referenzparameter, die jeweils einen ↑*PCB* adressieren, und es muss den PCB des aktuellen Prozesses nach erfolgter Kontextsicherung aktualisiert haben, bevor auf den PCB des einzusetzenden Prozesses zugegriffen wird. Auch wenn technisch möglich: Ein Prozesswechsel zu sich selbst bringt für den aktuellen Prozess keinen Fortschritt und bedeutet daher nur Unkosten.

Prozesszeiger Zeiger auf den ↑*Prozesskontrollblock* von dem ↑*Prozess*, der gegenwärtig auf einem ↑*Rechenkern* stattfindet. Im Allgemeinen verwaltet ein ↑*Betriebssystem* pro Rechenkern einen solchen Zeiger.

Prozesszustand Situation, in der sich ein ↑*Prozess* augenblicklich befindet. Der Prozess ist bereit (*ready*) zur ↑*Einlastung*, wenn alle von ihm benötigten ↑*Betriebsmittel* bis auf den ↑*Prozessor* zur Verfügung stehen und letzteren aber mit anderen Prozessen teilen muss. In solch einem Fall muss der Prozess innehalten, bis der für ihn geeignete Prozessor frei wird. Ein Prozess,

der stattfindet, ist laufend (*running*) auf einem Prozessor. Dieser Prozess kann zugunsten eines anderen Prozesses pausieren, dem dann der Prozessor zugeteilt wird. Der damit einhergehende ↑*Prozesswechsel* geschieht freiwillig oder unfreiwillig (*präemptiv*). Benötigt ein stattfindender Prozess ein weiteres Betriebsmittel, das in dem Moment jedoch nicht zur Verfügung steht, wird er blockiert (*blocked*). Dem folgt ein Prozesswechsel, wenn der Prozess sich den Prozessor mit anderen Prozessen teilen muss und wenigstens einer der anderen Prozesse in Bereitschaft steht. Steht in dem Moment kein anderer Prozess zur Einlastung bereit, wird der Prozessor untätig (↑*idle*). In der Situation wird nur ein externer Prozess dem Prozessor wieder eine Tätigkeit verschaffen können, beispielsweise durch eine ↑*Unterbrechungsanforderung*. Mit Zuteilung des benötigten Betriebsmittels wird ein blockierter Prozess deblockiert und wieder bereitgestellt. Diese Zustandsübergänge kontrolliert ein im ↑*Betriebssystem* mitlaufender (*on-line*) ↑*Planer*.

Pseudobefehl Anweisung an einen ↑*Assemblierer*. Beispielsweise zur Angabe des Programmabschnitts (Text, Daten, ↑*BSS*), auf die sich die nachfolgenden Anweisungen beziehen, um den ↑*Adresszähler* des aktuellen Segments auf eine bestimmte Grenze auszurichten, einem ↑*Symbol* einen Wert zuzuweisen oder Informationen für den ↑*Binder* aufzubereiten.

Quellmodul Eingabedatei eines Übersetzers (Kompilierer, Assemblierer), Ausgabedatei eines Instruments zur Programmerzeugung, das heißt, eines Editors, Übersetzers oder Generators.

race condition (dt.) ↑*Wettlaufsituation*.

race hazard (dt.) Laufgefahr, ↑*Wettlaufsituation*.

RAM *random access memory*, (dt.) Direktzugriffsspeicher.

read-modify-write Bezeichnung für einen aus drei Teilschritten bestehenden Zyklus bei der Ausführung von einem komplexen ↑*Maschinenbefehl*: erstens, ein ↑*Speicherwort* auslesen und in den ↑*Prozessor* laden; zweitens, den geladenen Wert im Prozessor verändern; drittens, den geänderten Wert in dasselbe Speicherwort zurückschreiben. Der Maschinenbefehl beschreibt eine ↑*Aktionsfolge*, die zwar eine logisch zusammenhängende Einheit bildet, jedoch keine ↑*atomare Operation* darstellt. Das bedeutet, nach jedem Teilschritt kann ein anderer Prozessor denselben Maschinenbefehl angewendet auf demselben Maschinenwort ausführen. Folge davon ist eine möglicherweise inkonsistente Berechnung.

ready list (dt.) ↑*Bereitliste*.

Rechenkern Prozessorkern einer ↑*CPU*, auf dem ein ↑*Prozess* stattfinden kann. Eine ↑*CPU* kann mehrere solcher Kerne enthalten und so Prozesse gleichzeitig stattfinden lassen.

Rechensystem Einheit aus technischen Anlagen, Bauelementen und Programmen zur Verarbeitung von Daten und Durchführung von Berechnungen. Zentrales Konzept ist der ↑*Rechner*, der allein oder im Verbund vernetzt mit anderen (gleichen/ungleichen) Einheiten seiner Art eine bestimmte Funktion für einen bestimmten Anwendungsfall ausübt. Dieser Anwendungsfall kann in besonderem Maße auf einen ganz bestimmten Zusammenhang ausgerichtet sein oder verschiedenste Bereiche umfassen, das heißt, einerseits ein Spezialsystem (*special-purpose system*) oder anderseits ein Universalsystem (*general-purpose system*) bedingen.

Rechner Programmgesteuerte, elektronische Rechenanlage (↑*computer*).

re-entrance (dt.) Wiedereintritt.

re-entrant (dt.) ablaufinvariant.

reference string (dt.) Referenzfolge.

Referenzfolge Reihe von zeitlich aufeinanderfolgenden, von einem ↑*Prozess* generierte Referenzen auf ↑*Speicherworte* im ↑*Arbeitsspeicher*. Der Verlauf dieser Reihe ist bestimmt durch die Struktur des Programms des jeweils betrachteten Prozesses und seiner Funktion in Abhängigkeit von den Eingabedaten. Jede Referenz entspricht einer ↑*Adresse*.

Registersatz Menge aller Register eines Prozessors. Typischerweise differenziert nach folgenden Arten: Datenregister, zur Speicherung von Operanden und Rechenergebnissen; Adressregister, zur Adressierung von Operanden und Maschinenbefehlen; Spezialregister, zur Betriebsstandanzeige (↑*Statusregister*) oder Adressierung spezieller Programmsegmente. Neben den dem ↑*Programmiermodell* zugerechneten Registern umfasst die Menge auch interne Register, auf die von einem ↑*Programm* heraus nicht zugegriffen werden kann.

RISC *reduced instruction set computer* (dt.) Rechner mit vermindertem (primitiven) Befehlssatz.

ROM *read-only memory*, (dt.) Festwertspeicher.

schedule (dt.) ↑*Ablaufplan*.

scheduler (dt.) ↑*Planer*.

scheduling (dt.) ↑*Ablaufplanung*.

Schlafzustand Systemzustand, der die einem ↑*Prozessor* bereitgestellte Energie, ihm angelegte Leistung beschreibt. Je tiefer dieser Zustand, desto geringer der Energieverbrauch beziehungsweise die Leistungsaufnahme des Prozessors, umso länger dauert seine Aufwachphase und umso umfassender sind die Maßnahmen in einem ↑*Betriebssystem* zur Aufrechterhaltung von Hardwarekontext (insb. ↑*CPU*, ↑*Zwischenspeicher* und ↑*Hauptspeicher*). Der Grad der Abstufung ist prozessorabhängig, heutige (2016) Prozessoren beinhalten eine bis fünf Stufen. Das Betriebssystem bringt den Prozessor zum Schlafen, sobald für ihn ↑*Leerlauf* festgestellt wurde. Der Prozessor weckt wieder auf, wenn er eine ↑*Unterbrechungsanforderung* erhält.

Schutzfehler Zugriffsfehler in Bezug auf das Schutzsystem einer (realen/virtuellen) Maschine. Betrifft die Verletzung von ↑*Speicherschutz*, aber auch von Rechten, die ein ↑*privilegierter Befehl* von einem ↑*Prozess* erfordert. Stellt eine ↑*Ausnahmesituation* dar.

schwergewichtiger Prozess ↑*Prozess*, der als ↑*Systemkernfaden* allein im eigenen und durch ↑*Speicherschutz* isolierten ↑*Adressraum* stattfindet. Auf ↑*Multics* zurückgehendes Verständnis einer ↑*Prozessinkarnation*, nämlich der Gleichsetzung von ↑*Prozess* und physisch abgeschottetem Adressraum.

Segment Unterteilung im ↑*Prozessadressraum*, wobei der ↑*Adressbereich* dieser Unterteilung auf den ↑*Speicher* eines Rechensystems abgebildet ist, genauer: dem ↑*Vorder-* und ↑*Hintergrundspeicher*. In diesem Bereich liegt Programmtext oder -daten, auch bezeichnet als ↑*Textsegment* und ↑*Datensegment*. Ein solcher Adressraum kann mehrere dieser Bereiche umfassen, die sich nicht überlappen und von fester oder variabler Größe sind. Beispiele für Segmente sind grobkörnige Bereiche wie ganze Dateien oder der gesamte Text- oder Datenbereich eines Programms oder feinkörnige Gebilde wie einzelne ↑*Unterprogramme*, Datenstrukturen, Objekte oder Variablen.

Seite Speichereinheit, deren üblicherweise feste Größe ganzzahlige Vielfache der Größe von einem ↑*Speicherwort* ist. Die konkrete Größe hängt ab von verschiedenen Faktoren: Umfang der ↑*Seitentabelle*, verfügbarer Platz im ↑*Übersetzungspuffer*, ↑*interne Fragmentierung* und der ↑*Zugriffszeit* auf den ↑*Hintergrundspeicher*.

Seitenabrufer ↑*Programm* zur ↑*Seitenumlagerung*. Intern im Betriebssystem (Systemebene) oder extern dazu oberhalb (Benutzerebene), gegebenenfalls sogar in jedem ↑*Maschinenprogramm*, angesiedelte Funktion zur Verwaltung von ↑*virtuellem Speicher*. Abgerufen werden Seiten aus dem ↑*Hintergrundspeicher* (Einlagerung) oder dem ↑*Vordergrundspeicher* (Auslagerung), jeweils im Moment des Zugriffs durch einen ↑*Prozess* oder vorausschauend.

Seitenfehler Zugriffsfehler in Bezug auf ↑*seitennummerierten virtuellen Speicher*. Im Moment des Zugriffs auf ein ↑*Speicherwort* im virtuellen Speicher stellt der ↑*Prozessor* fest, dass die Seite, die dieses Wort umfasst, nicht im ↑*Hauptspeicher* eingelagert ist. Der Prozessor stellt daraufhin eine ↑*Ausnahmesituation* fest, die vom Betriebssystem (↑*Seitenabrufer*) behandelt wird und zur ↑*Seitenumlagerung* führt: die Seite, auf der das referenzierte Wort steht, wird eingelagert.

seitennummerierter virtueller Speicher ↑*Virtueller Speicher*, der linear in Form von ↑*Seiten* organisiert ist. Dabei bildet eine Seite die kleinste Verwaltungseinheit.

Seitenumlagerung Technik beim ↑*seitennummerierten virtuellen Speicher*, die es erreicht, bei der Programmausführung referenzierte, aber nicht auf Seiten im ↑*Hauptspeicher* (Vordergrund) vorliegende, Programmteile vom ↑*Hintergrundspeicher* einzulagern und dazu, um gegebenenfalls Platz für die Einlagerung zu schaffen, im Hauptspeicher vorliegende Seiten vorher auf den Hintergrundspeicher auszulagern. Für den dabei anfallenden Transfer aller betroffenen Seiten im Vorder- und Hintergrundspeicher sorgt das Betriebssystem.

Seitenvorabruf ↑*Seitenumlagerung*, die vorausschauend funktioniert und nicht erst im Moment des Zugriffs auf eine ↑*Speicherstelle*. Das Betriebssystem hat exaktes oder heuristisches Wissen darüber, welche Seite als nächste bei der Programmausführung referenziert wird.

semantische Lücke Bedeutungsbezogener Unterschied zwischen den Beschreibungen von zwei Sprachebenen in einem mehrschichtig organisierten Rechensystem. Ursprünglich begründet in der Diskrepanz zwischen den komplexen Operationen der Konstrukte einer höherer Programmiersprache und den einfachen Operationen des Befehlssatzes einer CPU.

Semaphor Mittel zur ↑*Koordination* unterschiedlicher Art (↑*binärer Semaphor*, ↑*allgemeiner Semaphor*); ein spezieller Datentyp zum Zwecke der ↑*Synchronisation*, auf dem im Wesentlichen die beiden Operationen ↑*P* und ↑*V* definiert sind. Mit P synchronisiert sich ein ↑*Prozess* auf ein bestimmtes ↑*Ereignis*, dessen Auftreten durch V entweder von ihm selbst (↑*unilaterale Synchronisation*) oder von einem anderen (uni- und ↑*multilaterale Synchronisation*) Prozess angezeigt wird. Da beide Operationen insbesondere auch von verschiedenen Prozessen benutzt werden, müssen sie einer ↑*Nebenläufigkeitssteuerung* unterliegen: P und V gehören zur Klasse der ↑*Elementaroperation*, deren Durchführung jedoch auch bei ↑*Parallelverarbeitung* ein konsistentes Ergebnis liefern muss — das bedeutet allerdings nicht, sie jeweils als ↑*atomare Operation* auslegen zu müssen. Zentrale Bedeutung für den Fortschritt von einem Prozess in dem Zusammenhang hat die jeweils in P und V formulierte ↑*Ablaufsteuerung*. Diese kann einen Prozess in P blockieren lassen, bis er durch ein V wieder deblockiert wird; sie stellt sich für einen binären Semaphor etwas anders dar als für einen allgemeinen Semaphor. Beiden gemeinsam ist aber, dass die in P blockierten Prozesse eine ↑*Warteschlange* bilden, die durch V abgebaut wird. Diese Reihe von wartenden Prozessen ist entweder als dynamische Datenstruktur ein Attribut des Semaphordatentyps, damit pro ↑*Exemplar* vorhanden und wird auch nach eigenen Kriterien verwaltet, oder sie wird im Moment der (durch V veranlassten) Deblockierung eines Prozesses durch den ↑*Planer* nach seinen Kriterien berechnet. Erstere Variante ist anfällig für ↑*Interferenz* mit dem Planer, da die Bedienungsverfahren der Warteschlange und der ↑*Bereitliste* in aller Regel verschieden sind. Letztere Variante ist anfällig für Varianzen in der zur Deblockierung anfallenden ↑*Latenzzeit*, da die ↑*Prozesstabelle* erst nach Einträgen abgesucht werden muss, die mit bestimmten Exemplaren des Semaphordatentyps verknüpft sind. Wichtiges Merkmal — weder Fehler noch Unzulänglichkeit, wie gelegentlich kolportiert — beider Operationen (P und V) darüberhinaus ist, dass sie von jedem Prozess gleichberechtigt genutzt werden können. Anderenfalls gestaltet sich die Koordinierung von Prozessen in nicht wenigen Fällen schwierig (z.B. Hoare-/Hansen-Typ von ↑*Monitor*) oder gar unmöglich (z.B. Erzeuger-Verbraucher-Beziehung, Planer als ↑*kritischer Abschnitt*). Gleichwohl haben sich spezielle Semaphorvarianten herausgebildet, wie etwa ↑*privater Semaphor* und ↑*Mutex*, durch die bestimmte Muster der Synchronisation unterstützt werden und die helfen, Programmierfehler zu vermeiden oder zu erkennen.

semaphore (dt.) ↑*Semaphor*; Winker, Signalmast, Formsignal; (gr.-nlat.) Zeichenträger.

sensitiver Befehl ↑*Maschinenbefehl*, dessen direkte Ausführung durch eine ↑*virtuelle Maschine* nicht tolerierbar ist. Ein solcher Befehl gilt als *störungsempfindlich*, wenn er nämlich den Zustand der Systemsteuerung oder reservierter Prozessorregster beziehungsweise Speicherstellen ändert/abfragt, das Schutzsystem für Programme, die privilegiert ablaufen müssen, referenziert oder Ein-/Ausgabe tätigt.

sequentielles Programm ↑*Programm*, das ausschließlich Konstrukte zur Formulierung sequentieller Abläufe verwendet. Diese Konstrukte sind in der Programmiersprache enthalten, in der das Programm formuliert ist. Jedoch ist zu beachten, dass ein und derselbe Programmablauf auf einer Abstraktionsebene (höhere) sequentiell und einer anderen (tiefere) parallel sein kann: nämlich wenn auf höherer Ebene ein ↑*Unterprogramm* einer tieferen Ebene aufgerufen wird und das Unterprogramm ein ↑*nichtsequentielles Programm* darstellt.

Simultanverarbeitung Eigenschaft eines Betriebssystems zur ↑*Parallelverarbeitung* von Programmen. Manifestiert sich vor allem in speziellen Verfahren bei der Ein-/Umplanung von Prozessen (↑*partielle Virtualisierung*).

sleep state (dt.) ↑*Schlafzustand*.

Softwareschicht Sammlung von ↑*Programmen* auf einer bestimmten Ebene in einem System, das eine ↑*hierarchische Struktur* aufzeigt. Dabei müssen nicht alle Ebenen dieses Systems in Software vorliegen, das Gesamtsystem kann einen Komplex aus Soft-, Firm- und Hardware bilden. Programme, die in einer Schicht zusammengefasst sind, teilen sich nicht zwingend dasselbe Wissen über die dortigen Datenstrukturen, ↑*Modul* und Schicht sind zwei voneinander unabhängige Konzepte: Eine Schicht kann ein oder mehrere Module enthalten, ein Modul kann sich über eine oder mehrere Schichten erstrecken. So definiert vor allem die umfassende hierarchische Struktur, welche Programme zusammen eine Schicht ausmachen.

source module (dt.) Quellmodul.

Speicher Vorrichtung in einem Rechensystem zur Aufbewahrung von Informationen auf einem ↑*Speichermedium*.

Speicherhierarchie Gesamtheit der in einer Rangordnung stehenden ↑*Speicher* eines Rechensystems. Die Methode der Aufteilung des Speichers, und zwar aus Sicht einer ↑*Aktion*, besteht in der Festlegung einer Anordnung geordnet nach sinkender ↑*Zugriffszeit* (Nano-, Mikro-, Millisekunde, Sekunde, Minuten), größerer Zugriffseinheit (Speicherwort, Zwischenspeicherzeile, Seite, Segment, Datei), steigender Kapazität (Bits, Bytes, Kibi-, Mebi-, Gibi-, Tebi-, Exbi-, Zebi-, Yobibyte) und geringeren Produktkosten pro Bit. Damit ist die Art der Relation zwischen den verschiedenen Speicherebenen eine *Ordnungsrelation* im Sinne einer „kleiner-gleich“-Beziehung in Bezug auf bestimmte nichtfunktionale Merkmale.

Speichermedium Träger für Informationen, Datenträger. Die Informationen sind fotografisch (Film), mechanisch (Lochkarte/-streifen), magnetisch (Band, Platte), optisch (CD, DVD) oder elektronisch (Halbleiter) gespeichert.

Speicherschutz Vorrichtung, die zum Schutz gegen unautorisierte Zugriffe auf den ↑*Arbeitsspeicher* konstruiert ist. Technische Grundlage ist zumeist eine ↑*MMU* oder ↑*MPU*, die vom Betriebssystem entsprechend seines Schutzmodells so programmiert wird, dass ein ↑*Prozess* nur auf die ihm jeweils zugewiesenen Speicherbereiche zugreifen kann. Neben solchen hardwarebasierten Techniken gibt es auch softwarebasierte Ansätze, die ↑*Typsicherheit* von Programmabläufen voraussetzen. In solch einem Fall stellt der ↑*Kompilierer* sicher, dass das von ihm generierte Programm zur ↑*Laufzeit* (logisch) keine unautorisierten Speicherzugriffe hervorrufen kann.

Speicherstelle Ort, lokalisierbarer Bereich, in einem ↑*Speicher*.

Speicherwort Organisationseinheit in einem ↑*Speicher* (genauer: ↑*Hauptspeicher*), wobei jeder dieser Einheit eine ↑*Adresse* zugeordnet ist. Größe und Struktur einer solchen Einheit sind einem ↑*Maschinenwort* entsprechend ausgelegt.

spinlock (dt.) ↑*Umlaufsperre*.

spurious interrupt (dt.) unechte, unberechtigte ↑*Unterbrechung*.

SRAM statischer ↑*RAM*.

Stapelsegment ↑*Datensegment* mit dem Laufzeitstapel von einem ↑*Prozess*.

Stapelspeicher ↑*Speicher*, der als Stapel (dynamische Datenstruktur) organisiert und nach ↑*LIFO* betrieben wird.

Statusregister Register der ↑*CPU*, in dem eine Sammlung von Zustandsanzeigen (*status-flag bits*) vermerkt ist. Diese Anzeigen werden als Nebeneffekt bei der Ausführung von einem ↑*Maschinenbefehl* aktualisiert, etwa bei arithmetisch-logischen Operationen (typisch) oder beim Laden von einem ↑*Speicherwort* in ein Prozessorregister (untypisch, MOS Technology 6502); sie können aber auch explizit durch spezielle Maschinenbefehle gelöscht oder gesetzt werden. Der jeweilige Zustand wird durch ein ↑*Markierungsbit* in diesem Register dargestellt, wobei folgende Reihe von Bits typisch ist: Übertrag (*carry*), Überlauf (*overflow*), Vorzeichen (*sign*), Null (*zero*) und Parität (*parity*). Auf Basis dieser Zustandsanzeigen lassen sich in ↑*Assembliersprache* bedingte Anweisungen beziehungsweise Sprünge formulieren und so Fallunterscheidungen und verschiedene Formen von Schleifen in einem Programm umsetzen.

supervisor (dt.) ↑*Hauptsteuerprogramm*.

swapping (dt.) ↑*Umlagerung* eines Programms.

utility (dt.) Dienstprogramm.

Symbol Sinnbild einer ↑*Adresse*, deren symbolhafte Bezeichnung.

symbolische Adresse ↑*Adresse*, die als ↑*Symbol* für ein Strukturmerkmal eines Programms (d.h., ein Sprungziel oder der Platzhalter einer Variablen oder Konstanten) steht und zeichenhaftig (mnemonisch) dargestellt ist. Das Symbol ist durch einen ↑*Binder* in eine Nummer aufzulösen, bevor ein direkter Zugriff über die damit verknüpfte Adresse möglich ist: eine ↑*Bindung* ist herzustellen. Diese Nummer ist ein Element im ↑*Adressraum* von dem ↑*Prozess*, der durch eben dieses Programm bestimmt ist und den Zugriff bewirkt. Dabei kann die Auflösung vor oder zur ↑*Ladezeit* beziehungsweise sogar erst zur ↑*Laufzeit* des Programms stattfinden, die Bindung ist damit statisch oder dynamisch.

symbolischer Maschinenkode Zeichenhaftige, mnemonische Darstellung von ↑*Maschinenkode*, indem jedem Befehl seine Bedeutung durch eine Merkhilfe als ↑*Mnemon* gegeben wird. Beispielsweise steht `ADD EAX` für die Addition mit dem 32-Bit breiten Akkumulator eines x86-kompatiblen Prozessors.

Symboltabelle Datenstruktur, die jedem ↑*Symbol* eines Programms Attribute zuordnet, die dann vom ↑*Binder* ausgewertet werden. Die Attribute geben Hinweise beispielsweise auf Typ, Größe, Gültigkeitsbereich (lokal/global) und Lage (absolut/relativ) des mit dem Symbol bezeichneten Text-/Datenteils eines Programms.

synchrone Ausnahme ↑*Ausnahme*, die von dem ↑*Prozess* selbst verursacht wurde. Bleiben das ↑*Programm*, das den Prozess in statischer Hinsicht bestimmt, und die Eingabedaten, die den Prozess in dynamischer Hinsicht bestimmen, unverändert, wird der Prozess immer wieder an derselben Stelle (↑*Adresse* in seinem ↑*Adressraum*) in dieselbe Falle (↑*trap*) tappen: die ↑*Ausnahmesituation* ist vorhersehbar und reproduzierbar. Beispiele dafür sind ein ungültiger ↑*Maschinenbefehl*, eine falsche ↑*Adressierungsart*, ein ↑*Schutzfehler* oder auch ein ↑*Seitenfehler*, nämlich wenn ↑*virtueller Speicher* eine ↑*lokale Ersetzungsstrategie* benutzt.

Synchronisation Ergebnis der ↑*Synchronisierung*. Jeder beteiligte ↑*Prozess* unterliegt der Koordination der Kooperation und Konkurrenz zwischen seinesgleichen.

Synchronisierung Vorgänge zeitlich aufeinander abstimmen, sie in einer bestimmten Reihenfolge stattfinden lassen. Die Maßnahmen dazu können eine ↑*blockierende* oder ↑*nichtblockierende Synchronisation* der betroffenen Vorgänge bewirken.

system-call dispatcher (dt.) Systemaufrufzuteiler.

system-call stub (dt.) Systemaufrufstumpf.

Systemaufruf Aufruf eines ↑*Unterprogramms* im Betriebssystem, initiiert durch eine in einem ↑*Maschinenprogramm* kodierte ↑*Aktion*.

Systemaufrufnummer ↑*Operationskode*, der den ↑*Systemaufruf* an ein Betriebssystem identifiziert und normalerweise als fortlaufende Nummer dargestellt ist. Diese Nummer wird vom ↑*Systemaufrufzuteiler* auf eine interne Funktion des Betriebssystems abgebildet.

Systemaufrufparameter Argument für ein ↑*Unterprogramm*; hier: aktueller Parameter für einen ↑*Systemaufruf* und der ↑*Systemfunktion*, die die mit diesen Aufruf bezweckte Operation im Betriebssystem implementiert.

Systemaufrufstumpf Programmabschnitt, der einen ↑*Systemaufruf* absetzt und das dazu erforderliche Protokoll zwischen dem umfassenden ↑*Maschinenprogramm* und dem aufzurufenden Betriebssystem abwickelt. Der Systemaufruf wird kodiert, die ↑*Systemaufrufparameter* werden zur Übergabe ans Betriebssystem bereitgestellt, die ↑*CPU* wird angewiesen, die ↑*partielle Interpretation* des Maschinenprogramms durch das Betriebssystem zu ermöglichen und nach Rückkehr vom Systemaufruf wird auf eine ↑*Ausnahmesituation* geprüft. Dieser Programmabschnitt entspricht einer ↑*Blattprozedur* des Maschinenprogramms.

Systemaufrufzuteiler Gegenstück zu einem ↑*Systemaufrufstumpf*; Programmabschnitt im Betriebssystem, der die ↑*Aktionsfolge* festlegt, um einen ↑*Systemaufruf* anzunehmen und abzuwickeln: den Systemaufruf total und das ↑*Maschinenprogramm* partiell zu interpretieren. Das bedeutet, erstens, den Systemaufruf dekodieren und auf Gültigkeit prüfen. Ein ungültiger Systemaufruf begründet eine ↑*Ausnahmesituation*. Zweitens, die Weitergabe der ↑*Systemaufrufparameter* an das aufzurufende ↑*Unterprogramm*, welches die angeforderte Systemfunktion implementiert. Abhängig vom ↑*Operationsprinzip* des Betriebssystems erfolgt die Überprüfung der Parameter hier bei ihrer Weitergabe oder später im Moment ihrer Verwendung. Ungültige Parameter begründen eine Ausnahmesituation. Drittens, die durch den Systemaufuf ausgelöste ↑*partielle Interpretation* des Maschinenprogramms beenden und die ↑*CPU* anweisen, die Interpretation des Maschinenprogramms wieder aufzunehmen. Dieser Programmabschnitt ist eine ↑*Wurzelprozedur* des Betriebssystems.

Systemfunktion ↑*Unterprogramm* im Betriebssystem. Ein solches Programm implementiert beispielsweise die durch einen ↑*Systemaufruf* vom ↑*Maschinenprogramm* angeforderte Operation. Im Falle von ↑*UNIX* ist für jede mit `man(1)` unter Abschnitt 2 „*System calls*“ gelistete Funktion wenigstens ein Unterprogramm des Betriebssystems verknüpft. Darüberhinaus enthält ein Betriebssystem Unterprogramme, die nicht als Folge von Systemaufrufen zur Ausführung kommen, also ihren Ursprung in einer ↑*Aktion* des Maschinenprogramms haben, sondern bei denen der Aufruf originär von der Hardware ausgelöst wird, nämlich bei einer ↑*Unterbrechungsanforderung* oder allgemein aufgrund einer ↑*Ausnahmesituation*. Ferner kann ein im Betriebssystem autonom stattfindender ↑*Prozess*, eventuell sogar mehrere davon, als ↑*Dämon* Anlass für solche Aufrufe sein.

Systemkernfaden ↑*Faden* im ↑*Maschinenprogramm*, der durch eine eigene ↑*Prozessinkarnation* im ↑*Betriebssystemkern* implementiert ist. Sämtliche für solch einen Faden erforderlichen Betriebsmittel sowie für seine Verwaltung nötigen Funktionen stellt das ↑*Betriebssystem*. Dies

betrifft den ↑*Laufzeitkontext* des Fadens und die Funktionen zur ↑*Ablaufplanung*, ↑*Einlastung* und ↑*Synchronisation*: All diese Funktionen, insbesondere auch der ↑*Prozesswechsel*, verlaufen unter Kontrolle des Betriebssystemkerns und erfordern daher einen ↑*Systemaufruf*, wenn innerhalb des Maschinenprogramms vom aktuellen zu einem anderen Faden umgeschaltet werden soll. Damit ist jeder dieser Fäden aus Sichtweise des Maschinenprogramms vom Bautyp her ein ↑*leichtgewichtiger Prozess*, da (im Gegensatz zum ↑*Anwendungsfaden*) zwar vergleichsweise aufwändige Systemaufrufe zu seiner Kontrolle erforderlich sind, jedoch bei Fadenwechsel im selben Maschinenprogramm derselbe gegebenenfalls unter ↑*Speicherschutz* stehende ↑*Adressraum* aktiv bleibt. Für das Betriebssystem ist ein solcher Faden ein ↑*Objekt erster Klasse*, das heißt, der durch den Faden konkretisierte ↑*Prozess* ist insbesondere auch dem Betriebssystemkern bekannt. Eine vom Maschinenprogramm aus beanspruchte ↑*Systemfunktion* läuft damit immer im Namen dieses Fadens ab. Wird dieser dann durch die Systemfunktion blockiert (↑*Prozesszustand*), kann der Betriebssystemkern von selbst zu einen anderen Faden dieser Art im selben Maschinenprogramm umschalten.

task (dt.) ↑*Aufgabe*.

tatsächlicher Parameter Argument, das einem ↑*Unterprogramm* beim Aufruf tatsächlich übergeben wird. Ein dazu korrespondierender ↑*formaler Parameter* bestimmt den Typ des Wertes, der übergeben werden kann.

Team Gruppe von Vorgängen an einer gemeinsamen ↑*Aufgabe*. Logisch entspricht jeder Vorgang einem ↑*Prozess*, der physisch als ↑*Faden* im ↑*Maschinenprogramm* in Erscheinung tritt und im ↑*Betriebssystem* durch eine ↑*Ablaufplanung* kontrolliert wird. Wesentliches Merkmal ist die Repräsentation als ↑*nichtsequentielles Programm*, das die Basis für den gemeinsamen ↑*Adressraum* der Fäden legt. Unwesentlich ist die Repräsentation des Fadens im Betriebssystem, solange sichergestellt ist, dass ein Faden in einer sich für ihn abzeichnenden Entwicklung zuvorkommend (*präemptiv*) zur Ausführung gelangen kann. Letzteres bedeutet keinesfalls, den Faden als ↑*Systemkernfaden* implementieren zu müssen. Andere Formen sind ebenso möglich, beispielsweise eine ↑*Fortsetzung*.

Teilaufgabe Teil einer ↑*Aufgabe*.

Teilinterpretation siehe ↑*partielle Interpretation*.

Termineinhaltung Vereinbarung eines festgelegten Zeitpunkts, bis zu dem ein Berechnungsergebnis vorliegen soll oder muss. Der Termin ist typischerweise abgestuft klassifiziert wie folgt: weich (schwach), wenn die Terminverletzung tolerierbar ist, mit jeder weiteren Verzögerung das Berechnungsergebnis aber an Wert verliert; fest, wenn die Terminverletzung tolerierbar ist, das Berechnungsergebnis aber keinen Wert mehr hat und die ↑*Aufgabe* daher abgebrochen werden muss; hart (strikt), wenn die Terminverletzung wegen sonst drohender Gefahr für Leib und Leben oder hohem Risiko für wirtschaftlichen Verlust nicht tolerierbar ist, eine ↑*Ausnahmesituation*, auf die die Anwendung rechtzeitig reagieren können muss. Charakteristisches Merkmal für ein ↑*Maschinenprogramm*, das unter ↑*Echtzeit* ablaufen muss.

Textsegment ↑*Segment*, das Programmtext speichert.

thread (dt.) ↑*Faden*.

time slice (dt.) ↑*Zeitscheibe*.

timer (dt.) ↑*Zeitgeber*.

TLB *translation lookaside buffer*, (dt.) Übersetzungspuffer.

Transaktion ↑*Aktion* oder ↑*Aktionsfolge*, die eine *logische Einheit* bildet und als Ganzes entweder gelingt oder scheitert. Gelingt sie, wurde die Berechnung vollständig korrekt durchgeführt und der damit verbundene Datenbestand im konsistenten Zustand hinterlassen. Scheitert sie, bleibt die Aktion/Aktionsfolge ohne Auswirkung, als wenn sie nie stattgefunden hätte.

trap (dt.) Fangstelle, abfangen; ↑*Abfangung*.

Typfehler Folge einer ↑*Typverletzung*.

Typsicherheit Ausmaß der Verhinderung von ↑*Typfehler* durch eine Programmiersprache beziehungsweise einen ↑*Kompilierer*. Die Maßnahmen dazu greifen statisch (bei der ↑*Kompilierung*), dynamisch (zur ↑*Laufzeit*) oder in Kombination. Je nach Stärke der Typisierung der Programmiersprache wird der Kompilierer mehr oder weniger viel Typkonflikte unbeanstandet durchlassen, die dann jedoch durch generierte Anweisungen zur Typkontrolle zur Laufzeit abgefangen werden und als ↑*Ausnahmesituation* enden.

Typverletzung Unstimmigkeit bei Operationen auf verschiedenen Datentypen. Beispielsweise eine Ganzzahl (`int`) als Zeiger auf ein Zeichen (`char*`) oder als Universalzeiger (`void*`) behandeln — dadurch eine beliebige ↑*Adresse* konstruieren und (logisch) unautorisiert auf den ↑*Arbeitsspeicher* zugreifen können.

Überlagerung Programmteil (Text oder Daten), dessen Hauptspeicherbereich wechselweise über die Zeit von mehreren solcher Teile gemeinsam belegt wird.

Überlagerungsspeicher Bereich in einem peripheren ↑*Speicher*, in dem ↑*Überlagerungen* eines Programms abgelegt sind.

Übersetzer Dienstprogramm, das ein in einer bestimmten Sprache formuliertes Programm in ein semantisch äquivalentes Programm einer anderen Sprache umwandelt. Typisch dafür sind ↑*Kompilierer* und ↑*Assemblierer*.

Übersetzungseinheit Quellmodul für einen ↑*Übersetzer*, seine Eingabe.

Umlagerung Übertragung von Speicherinhalten zwischen verschiedenen Ebenen der ↑*Speicherhierarchie* eines Rechensystems. Typisch dafür ist etwa die Auslagerung eines kompletten Programms (d.h., all seiner Text- und Datenbereiche) vom Vordergrund- in den Hintergrundspeicher beziehungsweise umgekehrt die Einlagerung.

Umlaufsperre Sperre (*lock*), deren Gültigkeit von einem ↑*Prozess* durch Kreiseln (*spin*) anhaltend überprüft wird; ↑*blockierende Synchronisation*. Der Prozess betreibt ↑*geschäftiges Warten*, bis die Sperre aufgehoben ist. Ein Verfahren zur ↑*Synchronisation* von Prozessen auf einem ↑*Multi-*/↑*Mehrkernprozessor*. Dabei sollte darauf geachtet werden, dass jeder der kreiselnden Prozesse auf einen eigenen ↑*Prozessor* stattfindet und der die Sperre haltende Prozess keine ↑*Verdrängung* erfährt. Ansonsten drohen erhebliche Leistungseinbußen durch die zusätzliche Verzögerung der Prozesse anderer Prozessoren durch einen einzelnen (von seinem Prozessor verdrängten) Prozess.

unilaterale Synchronisation ↑*Synchronisation*, die sich nur in einer bestimmten Rolle blockierend für einen ↑*Prozess* auswirken kann; auch ↑*logische Synchronisation* oder ↑*Bedingungssynchronisation*. Zwischen den Prozessen besteht eine Erzeuger-Verbraucher-Beziehung in Bezug auf ein ↑*konsumierbares Betriebsmittel*. Der Erzeugerprozess zeigt das ↑*Ereignis* an, ein Betriebsmittel zur Verfügung gestellt zu haben und muss zur Durchführung dieser Anzeige (Signalisierung) selbst nicht warten. Demgegenüber ist der Verbraucherprozess in seinem Vorankommen von einem solchen Ereignis abhängig. Er wird blockieren, wenn das Ereignis (Signalisierung) noch nicht stattgefunden hat. Solch ein Muster der Kooperation von Prozessen, formuliert als ↑*nichtsequentielles Programm*, definiert die konsequente ↑*Aktionsfolge* für einen bestimmten Sachzusammenhang, das daher auch als logische Synchronisation bezeichnet wird. Darüberhinaus ist ein derartiges Ereignis auch Beispiel für die Entkräftung der Wartebedingung eines Verbraucherprozesses, der mit dieser Betrachtungsweise dann der Bedingungssynchronisation unterliegt. Grundlage für dieses Muster der Synchronisation bildet ein ↑*allgemeiner Semaphor*, der das konsumierbare Betriebsmittel repräsentiert. Die Anzahl der mit ↑*V* durch den Erzeugerprozess freigesetzten Einheiten diese Betriebsmittels ergibt

sich durch den positiven Wert des Semaphors. Mit ↑*P* wird diese Anzahl durch den Verbraucherprozess reduziert, er konsumiert eine entsprechende Einheit des Betriebsmittels. Der Absolutwert eines negativen Werts des Semaphors gibt die Anzahl von Betriebsmitteleinheiten an, auf deren Bereitstellung Verbraucherprozesse warten.

UNIX Mehrplatz-/Mehrbenutzerbetriebssystem. Erste Installation im Jahr 1969 (DEC PDP-7), zunächst programmiert in ↑*Assembliersprache*, später (1972) umgearbeitet unter Verwendung von ↑*C*. In der Anfangsphase (1969) unter dem Namen UNICS geführt, als Abkürzung für „*Uniplexed Information and Computing Service*“ und Wortspiel in Bezug auf ↑*Multics*. Ursprung vieler Betriebssystementwicklungen, insbesondere ↑*Linux* und ↑*Darwin*.

unteilbarer Grundblock Programmabschnitt, auch Basisblock genannt, der einen linearen und der ↑*Unteilbarkeit* unterzogenen Kontrollfluss beschreibt. Linearität des Kontrollflusses bedeutet, dass der Block nur einen Einsprungspunkt und einen Ausstieg aufweist.

Unteilbarkeit Umstand, der die Aufsplittung einer ↑*Aktion* oder ↑*Aktionsfolge* und Verteilung der Einzelmaßnahmen auf verschiedene Zeitintervalle verhindert. Die (Schrittfolge einer) Aktion beziehungsweise Aktionsfolge findet scheinbar gleichzeitig statt, sie ist atomar, kann sich nicht mit einem weiteren Exemplar ihrer selbst zeitlich überlappen.

untellbares Betriebsmittel Exklusivität beanspruchendes ↑*wiederverwendbares Betriebsmittel*, dessen Einheiten zu einem Zeitpunkt nur von maximal einen ↑*Prozess* erworben (*acquire*) und für die Zeitdauer bis zur Freisetzung (*release*) belegt werden dürfen. Typisches Beispiel eines solchen Betriebsmittels ist jede Form von Drucker: für die Dauer eines Druckvorgangs steht das Druckgerät keinem anderen Druckprozess zur Verfügung, ansonsten können unlesbare oder unverständliche Ausdrucke die Folge sein. In diese Kategorie ist aber auch ein einzelnes ↑*Speicherwort* einzuordnen, wenn nämlich durch ↑*Parallelverarbeitung* nicht tolerierbare Lese-Schreib-Konflikte für eine an der ↑*Adresse* dieses Worts liegende und *gemeinsam benutzte Variable* auftreten können. Im übertragenen Sinn passt auch ein ↑*kritischer Abschnitt* dazu, bei dem letztlich ja die von seinem Programmtext belegten Speicherworte — nicht etwa die von dem Text referenzierten Daten! — zeitweilig der exklusiven Nutzung nur durch einen Prozess unterzogen sind (↑*unteilbarer Grundblock*).

Unterbrechung Störung von einem ↑*Prozess*, einer ↑*Aktionsfolge* oder ↑*Aktion*, verursacht durch ein in Bezug auf diese Handlungen externes Ereignis. Das Ereignis findet asynchron zu der Handlung statt, unvorhersagbar und nicht reproduzierbar. Bestenfalls lässt sich noch die ↑*Zwischenankunftszeit* der Ereignisse abschätzen, womit jedoch nur eine Aussage über die Frequenz und nicht über den Zeitpunkt möglich ist.

Unterbrechungsanforderung Forderung der ↑*Peripherie* an die ↑*CPU*, eine ↑*Unterbrechungsbehandlung* einzuleiten und durchzuführen.

Unterbrechungsbehandlung Vorgang zur Analyse und Bearbeitung der bei Ausführung von einem ↑*Maschinenprogramm* aufgetretenen Störung (↑*Unterbrechung*). Handelt es sich um eine berechtigte (echte) Störung, wird diese bearbeitet und die Ausführung des unterbrochenen Programms anschließend wieder aufgenommen. Im Falle einer unberechtigten Störung (↑*spurious interrupt*), wird diese vermerkt. Zuviele unberechtigte Störungen innerhalb kurzer Zeit deuten auf einen Fehler in der Hardware hin und könnten ↑*Panik* auslösen. Anderenfalls wird die Programmausführung fortgesetzt.

Unterbrechungshandhaber ↑*Unterprogramm* zur ↑*Unterbrechungsbehandlung*; dem Deutschen Patentwesen entnommen, das (en.) *handler* fachbegrifflich als (dt.) *Handhaber* übersetzt.

Unterbrechungslatenz ↑*Latenzzeit* ab Zeitpunkt der Wahrnehmung der ↑*Unterbrechung* in der ↑*CPU* bis zum Zeitpunkt der Aufnahme der ↑*Unterbrechungsbehandlung* im Betriebssystem. Wurde keine ↑*Unterbrechungssperre* angefordert, ist die Verzögerungszeit bestimmt durch die restliche Ausführungszeit von dem ↑*Maschinenbefehl*, der im Moment der ↑*Unterbrechung*

aktiv war. Wurde jedoch eine Unterbrechungssperre gesetzt, ergibt sich die Verzögerung aus der Restlaufzeit bis zur Aufhebung der Sperre durch das Betriebssystem.

Unterbrechungsprioritätsebene Attribut des aktuellen Systemzustands, das die Priorität der momentan von der ↑*CPU* akzeptierten ↑*Unterbrechungsanforderung* anzeigt. Typischerweise werden Anforderungen mit einer Priorität kleiner oder gleich der Priorität der CPU nicht akzeptiert, diese sind gesperrt. Im Normalbetrieb läuft die CPU auf Prioritätsebene 0 und jede Unterbrechungsanforderung hat eine Priorität größer als 0. Mit jeder akzeptierten Unterbrechungsanforderung wird die Priorität der CPU auf die Ebene dieser Anforderung gehoben. Bei Rückkehr aus der ↑*Unterbrechungsbehandlung* kehrt die Priorität der CPU wieder auf die vorige Ebene zurück. Im ↑*Unterbrechungszyklus* führt die CPU damit implizit eine ↑*Unterbrechungssperre* für die Prioritätsebene durch, die mit der Unterbrechungsanforderung assoziiert ist. Eine solche Sperre kann jedoch auch explizit gesetzt werden, um der ↑*Unterbrechung* eines Programmablaufs vorzubeugen (↑*kritischer Abschnitt*). Darüber hinaus kann diese Sperre im Rahmen der Unterbrechungsbehandlung explizit aufgehoben werden, um noch vor Rückkehr daraus auf Unterbrechungsanforderungen reagieren zu können und damit die ↑*Unterbrechungslatenz* zu verkürzen. Läuft die Unterbrechung jedoch über ↑*Pegelsteuerung* und hat der ↑*Unterbrechungshandhaber* vor Aufhebung der Sperre die Unterbrechung noch nicht am ↑*Peripheriegerät* bestätigt, gilt die zugehörige Anforderung immer noch als anhängig und es kommt zum indirekt rekursiven Aufruf des Handhabers durch die CPU. Dies kann ↑*Panik* auslösen, da davon auszugehen ist, dass ein solcher Kreislauf im Unterbrechungshandhaber nicht durchbrochen wird beziehungsweise werden kann.

Unterbrechungssperre Vorkehrung zur Abwehr einer ↑*Unterbrechung*, stellt ↑*Synchronisation* her. Implementiert als ↑*privilegierter Befehl*, der nur lokale Auswirkung auf seinen ↑*Prozessor* hat und eine an ihn gestellte ↑*Unterbrechungsanforderung* der ↑*Peripherie* nicht durchlässt. Dazu muss diese Anforderung am Prozessor maskierbar sein (↑*IRQ*). Eine nichtmaskierbare Anforderung (↑*NMI*) dagegen kann nicht gesperrt werden: Gegebenenfalls lässt sich die Peripherie jedoch zeitweilig umprogrammieren, einen NMI nicht anzufordern. Zur Aufhebung der Sperre kommt eine entsprechende inverse Operation zum Einsatz. Bei der Unterbrechungsabwehr besonders zu beachten ist die Art der Zustellung des Digitalsignals, das heisst, ob die zu sperrende Unterbrechung der ↑*Pegel-* oder ↑*Flankensteuerung* unterliegt.

Unterbrechungszyklus Phase im ↑*Abruf- und Ausführungszyklus*, wird typischerweise am Ende der Befehlsausführung durchlaufen und umfasst folgenden *Vorspann* an Teilschritten, um die ↑*Unterbrechungsbehandlung* — allgemein: ↑*Ausnahmebehandlung*, denn für die ↑*Abfangung* gilt dasselbe — einzuleiten: (1) den ↑*Prozessorstatus* sichern, mindestens davon die Inhalte von ↑*Statusregister* und ↑*Befehlszähler*, (2) den Befehlszähler mit der ↑*Adresse* des ↑*Unterbrechungshandhabers* laden und (3) in den privilegierten ↑*Betriebsmodus* wechseln, sofern von der ↑*CPU* implementiert. Der *Nachspann* der Unterbrechungsbehandlung kommt durch einen speziellen ↑*Maschinenbefehl* in ↑*Aktion*, der ganz normal im Abruf- und Ausführungszyklus verarbeitet wird und folgenden wesentlichen Schritt macht: (5) den Prozessorstatus wieder herstellen. Hier ist zu beachten, dass die Schritte (2) und (5) jeweils den Befehlszähler verändern und damit für den nächsten Durchlauf vom Abruf- und Ausführungszyklus vorgeben, von welcher Adresse der nächste Maschinenbefehl zur Ausführung abgerufen werden soll. Diese Schritte bewirken letztlich den Hinsprung (2) zur Unterbrechungsbehandlung und den Rücksprung (5) zu dem ↑*Prozess*, der unterbrochen/abgefangen wurde. Da der Betriebsmodus der CPU in ihrem Statusregister vermerkt ist, wird mit Wiederherstellung vom Prozessorstatus implizit zu dem Betriebsmodus umgeschaltet (5), der zum Zeitpunkt der ↑*Unterbrechung* aktiv war. Zur Sicherung (1) und Wiederherstellung (5) all dieser Statusdaten findet typischerweise ein ↑*Stapelspeicher* Verwendung.

Unterprogramm ↑*Programm*, dessen Aufruf in demselben (Rekursion), einem anderen oder mehrerer anderer Programme kodiert ist. Je nach Programmiersprache und -paradigma technisch verschieden umgesetzt und unterschiedlich bezeichnet: Operation, Abschnitt, Routine, Subroutine, Prozedur, Funktion, Methode oder Makro.

user thread (dt.) ↑*Anwendungsfaden*.

V Abk. für (hol.) *verhoog*, (dt.) erhöhen; synonym mit (en.) *up*, *signal*, *release*.

verdeckter Kanal Variante von einem Sicherheitsangriff auf ein ↑*Rechensystem*, die es ermöglicht, Informationen von einem ↑*Prozess* abzugreifen und zu einem anderen Prozess zu transferieren, ohne dass beide zur direkten Kommunikation berechtigt wären. Der Kanal ist nicht zum Informationstransfer bestimmt: er benutzt keine der von einem ↑*Betriebssystem* angebotenen legitimen Mechanismen zum Datentransfer, sondern zweckentfremdet andere legitime und per regulärem ↑*Systemaufruf* angebotene Mechanismen dafür. Beispiel ist der durch bestimmte Berechnungen eines Prozesses bewirkte Effekt auf die Systemlast, die von einem anderen Prozess gemessen werden kann. Die Lastschwankungen geben Anlass zu Spekulationen über die in dem Moment stattfindenden Vorgänge beziehungsweise werden gezielt zur Informationskodierung genutzt.

Verdrängung ↑*Aktionsfolge*, durch die ein ↑*wiederverwendbares Betriebsmittel* den Besitzer wechselt. Dem ↑*Prozess*, der dieses ↑*Betriebsmittel* gerade besitzt, wird es entzogen und einem anderen Prozess zugeteilt, der dann zum neuen Besitzer wird. Typisches Beispiel eines solchen Betriebsmittels ist die ↑*CPU* oder ein einzelner ↑*Rechenkern*, wenn nämlich die ↑*Prozesseinplanung* nach einem ↑*Zeitteilverfahren* arbeitet.

Virtualisierungssystem ↑*Interpretersystem* zur Nachbildung einer (realen) Maschine, typischerweise realisiert durch einen ↑*VMM*. Der VMM ist ein für eine ↑*Wirtsmaschine* bestimmtes Steuerprogramm, er erschafft die Ablaufumgebung für eine ↑*virtuelle Maschine*. Unterschieden wird zwischen *Typ I* und *Typ II*. Ein Typ I VMM läuft auf der bloßen Hardware (Wirtsmaschine), direkt auf der ↑*CPU*, wohingegen ein Typ II VMM zusätzlich noch ein ↑*Wirtsbetriebssystem* benutzt. Zentrale Funktion von einem VMM ist es, die direkte Ausführung eines „störungsempfindlichen Befehls“ (↑*sensitiver Befehl*) durch die virtuelle Maschine zu verhindern. Ein solcher Befehl wird vom VMM abgefangen (↑*trap*) und speziell behandelt (↑*partielle Interpretation*). Typisches Beispiel ist die ↑*Unterbrechungssperre*. Wird diese durch eine ↑*Aktion* in einem ↑*Gastbetriebssystem* veranlasst, darf nur eine ↑*Unterbrechungsanforderung* an die virtuelle Maschine, die eben dieses Gastbetriebssystem ausführt, maskiert werden. In solch einem Fall fängt der VMM den entsprechenden ↑*Maschinenbefehl* (`cli`, x86) ab und unterbindet die mögliche ↑*Unterbrechung* einer auf der virtuellen Maschine stattfindenden Aktion. Der Zustand der ↑*Wirtsmaschine* bleibt diesbezüglich unverändert, das heißt, für sie gilt keine Unterbrechungssperre. Kommt der inverse Befehl (`sti`, x86) zur Ausführung durch die virtuelle Maschine, agiert der VMM dementsprechend.

virtuelle Maschine Maschine, die nicht in Wirklichkeit existiert, aber echt (real) erscheint. Der ↑*Prozessor* dieser Maschine ist entweder ein ↑*Interpreter* oder ein ↑*Übersetzer*. Ersterer implementiert ein ↑*Virtualisierungssystem* bestimmter Art, das ein ↑*Programm* dieser Maschine direkt ausführen kann, wohingegen letzterer das auszuführende Programm in ein Programm einer anderen (realen/virtuellen) Maschine transformiert, um es durch diese ausführen zu lassen.

virtueller Speicher ↑*Arbeitsspeicher*, der „unecht“ vorhanden ist, nicht in Wirklichkeit existiert, aber echt erscheint. Wird durch ein Betriebssystem bereitgestellt und auf den im Rechensystem verfügbaren Vorder- und Hintergrundspeicher abgebildet.

VM/370 Einplatz-/Einbenutzerbetriebssystem für eine ↑*virtuelle Maschine* im ↑*Mehrprogrammbetrieb*. Erste Installation 1972 (IBM System/370).

VMM *virtual machine monitor*, (dt.) Steuerprogramm für eine ↑*virtuelle Maschine*, zentraler Bestandteil von einem ↑*Virtualisierungssystem*.

volatile register (dt.) ↑*flüchtiges Register*.

Vordergrundspeicher ↑*Hauptspeicher*; auch Primärpeicher. Flüchtiger ↑*Speicher*, der direkt über Lese-/Schreiboperationen des Prozessors bedient wird.

Vorhersagbarkeit Grad, in dem eine korrekte Prädiktion (Vorhersage) des Zustands, Verhaltens oder Platz-, Energie- oder Zeitbedarfs von einem ↑*Rechensystem* möglich ist, in qualitativer oder quantitativer Hinsicht. Diese unterliegt immer bestimmten und vorher zu treffenden Annahmen. Insbesondere für ↑*Echtzeit* sind dabei folgende Aspekte bedeutsam: die Granularität eines Termins und Laxheit (weich, fest, hart) einer ↑*Aufgabe*, die Striktheit (weich, fest, hart) eines Termins, Zuverlässigkeitsanforderungen, die Größe des Systems und der Grad an Interaktion (↑*Koodination*) zwischen den Komponenten, sowie die Charakteristik der Umgebung, in der das System operieren muss.

Warteschlange Konstrukt für aufgelaufene, unerfüllte und zeitweilig zurückgestellte Aufträge. Bildet sich, wenn in einem Zeitintervall mehr Aufträge eintreffen, als in demselben Intervall verarbeitet werden können. Ist ein solcher Auftrag etwa ein ↑*Prozess*, so bedeutet beispielsweise eine gefüllte ↑*Bereitliste*, dass zu wenig ↑*Betriebsmittel* der Art ↑*Prozessor* zur Verfügung stehen. Hinzufügen weiterer Prozessoren kann zum Abbau der Liste beitragen, oder die gerade verfügbaren Prozessoren werden im ↑*Zeitteilverfahren* entsprechend einer bestimmten Strategie bedient, bis die Liste geleert werden konnte. Die Strategie ist im ↑*Einplanungsalgorithmus* verankert.

Wartezeit Zeitspanne von der Unterbrechung bis zur Fortsetzung einer Operation oder Operationsfolge.

wechselseitiger Ausschluss Form von ↑*multilaterale*, ↑*blockierende Synchronisation*. Sorgt dafür, dass ein ↑*kritischer Abschnitt* stets nur von höchstens einen ↑*Prozess* betreten und durchstreift werden kann. Jeder der beteiligten Prozesse erwartet *aktiv* (↑*Umlaufsperre*) oder *passiv* (↑*binärer Semaphor*, ↑*Mutex*) das ↑*Ereignis*, dass der von einem Prozess erworbene kritische Abschnitt wieder freigesetzt wird. Ein Prozess wartet aktiv, wenn er durch anhaltendes Abfragen (*polling*) seiner Wartebedingung während seiner Wartezeit beschäftigt bleibt (↑*busy waiting*). Im Gegensatz dazu wartet ein Prozess passiv, wenn er blockiert (↑*Prozesszustand*) und dem ↑*Ereignis*, dass der kritische Abschnitt verlassen wird, „schlafend“ entgegensieht.

Wettlaufsituation Umstand, bei dem ein ↑*nichtsequentielles Programm* eine ↑*Aktion* oder ↑*Aktionsfolge* mit unbestimmten zeitlichen Verhalten zulässt. Konsequenz daraus können inkorrekte Programmabläufe sein, die Fehlverhalten verursachen. Dem zugrunde liegen asynchrone Ereignisse, die die Programmabläufe nicht nur unvorhersagbar, sondern auch nicht reproduzierbar machen. Zur Vorbeugung des möglichen Fehlverhaltens sind Maßnahmen zur ↑*Nebenläufigkeitssteuerung* erforderlich.

Wiedereintritt Moment, in dem ein ↑*Programm* erneut, und zwar verschachtelt, zur Ausführung kommt. Ursache ist eine vorangegangene ↑*Ausnahme*, erhoben während der Programmausführung, wobei die ↑*Ausnahmebehandlung* Bestandteil des Programms selbst ist. Im Zuge dieser Behandlung wird ein in seiner Ausführung zuvor unterbrochener Programmabschnitt (einer indirekten Rekursion nicht unähnlich) wiederholt betreten. Zur korrekten Ausführung ist für das einen solchen Abschnitt enthaltene Programm ↑*Ablaufinvarianz* gefordert.

wiederverwendbares Betriebsmittel ↑*Betriebsmittel*, das nur in begrenzter Anzahl vorhanden ist und von dem es eine feste Anzahl von Einheiten gibt. Jede dieser Einheiten ist entweder verfügbar oder belegt, das heißt, keinem oder einem ↑*Prozess* zugewiesen. Mit erfolgter Zuweisung hat ein Prozess die von ihm angeforderte Betriebsmitteleinheit erworben (*acquire*). Ist die Mitbenutzung desselben Betriebsmittels ausgeschlossen (↑*unteilbares Betriebsmittel*), können Einheiten davon zu einem Zeitpunkt nur maximal einem Prozess zugewiesen sein. Ein Prozess kann beliebige von ihm belegte Einheiten von Betriebsmitteln freisetzen (*release*), sofern er seinerseits nicht den Erwerb einer Betriebsmitteleinheit erwartet und demzufolge

blockiert ist. Erworbene Einheiten können dem Prozess nicht entzogen werden, sie sind erst wieder verfügbar, nachdem sie durch den Prozess explizit freigesetzt wurden.

Wirtsbetriebssystem ↑*Betriebssystem*, das ein ↑*Gastbetriebssystem* bewirtet. Letzteres kann in der Bereitstellung einer eigenen ↑*Systemfunktion* von den (per ↑*Systemaufruf* zugänglichen) Systemfunktionen des Wirtssystems profitieren. Darüberhinaus kann mehr als ein Gastsystem von dem Wirtssystem bedient werden, ohne dass ein ↑*Prozess* des Gastsystems dies in funktionaler Hinsicht wahrnimmt. Eine solche Mehrfachnutzung kann sich *homogen* (gleichartige Gastsysteme) oder *heterogen* (verschiedenartige Gastsysteme) darstellen, ohne dass dies wiederum dem Wirtssystem bewusst ist.

Wirtsmaschine Maschine realer oder virtueller Natur, die eine virtuelle Maschine bewirtet: letztere benutzt (↑*Benutzthierarchie*) erstere. Dies bedeutet in erster Linie, dass das ↑*Programmiermodell* der benutzten Maschine ebenfalls für die benutzende Maschine gilt. Typischerweise läuft der ↑*VMM* als einziges ↑*Programm* auf dieser Maschine, um eben eine oder mehrere virtuelle Maschinen zu realisieren.

Wortbreite Definiert die Bitanzahl in einem ↑*Maschinenwort* einer ↑*CPU*. Typisch waren oder sind 8, 9, 12, 16, 18, 24, 32, 36, 39, 40, 48, 60 und 64 Bits pro Wort (Stand 2016).

Wurzelprozedur Oberprogramm, auch übergeordnetes Programm, das ein ↑*Unterprogramm* aufruft, aber selbst nicht als Unterprogramm aufgerufen wird.

x86 Prozessorarchitektur, 16/32-Bit, abwärtskompatibel zum Intel 8086 (1978).

Zeitgeber ↑*Peripheriegerät*, mit dem zeitlich gebundene Vorgänge geregelt werden können. Das Gerät sendet eine ↑*Unterbrechungsanforderung* an die ↑*CPU*, wenn ein vom ↑*Gerätetreiber* vorprogrammiertes Zeitintervall abgelaufen ist.

Zeitscheibe Zeitintervall, für das ein ↑*Prozess* den ↑*Prozessor* vom ↑*Planer* zugeteilt bekommt.

Zeitteilverfahren ↑*Multiplexverfahren*, das jedem ↑*Prozess* immer nur für ein bestimmtes Zeitintervall (↑*time slice*) den ↑*Prozessor* zuteilt. Technische Grundlage dafür bildet ein von dem ↑*Planer* benutzter ↑*Zeitgeber*. Der Ablauf des Zeitintervalls bewirkt eine ↑*Unterbrechung* des laufenden Prozesses, woraufhin der Planer als Teil der ↑*Unterbrechungsbehandlung* aufgerufen wird. In dieser Situation wird der Planer dem unterbrochenen Prozess bedingt den Prozessor entziehen, das heißt, eventuell einen ↑*Prozesswechsel* erzwingen: der unterbrochene Prozess wird vom Prozessor verdrängt (↑*preemption*). Dies geschieht jedoch nur, wenn neben dem logisch noch laufenden (↑*Prozesszustand*), aber tatsächlich unterbrochenen, Prozess wenigstens ein weiterer Prozess bereit zur ↑*Einlastung* zur Verfügung steht und dieser keine niedrigere Priorität als der unterbrochene Prozess besitzt. In dem Fall wird der unterbrochene Prozess auf die ↑*Bereitliste* gesetzt und ein anderer Prozess (mit gleicher/höherer Priorität), zu dem dann als nächster gewechselt werden soll, davon herunter genommen. Gibt es keinen solchen Prozess, wird der unterbrochene Prozess für ein weiteres Zeitintervall fortgesetzt. Je nach Verfahren ist das Zeitintervall fest oder variabel.

Zentraleinheit Mittelpunkt eines Rechensystems, durch dem alle anderen Komponenten der ↑*Peripherie* kontrolliert und gesteuert werden. Synonym für ↑*CPU*.

Zugriffszeit Zeitspanne von der Auslösung bis zur Ausführung einer Operation oder Operationsfolge.

Zustandssicherung Maßnahme zum Sicherstellen der augenblicklichen physischen Beschaffenheit von einem ↑*Prozess* in Bezug auf den ihm zugeteilten ↑*Prozessor*. Die betrifft wenigstens den ↑*Prozessorstatus*, kann jedoch auch bestimmte Datenbestände umfassen, die über den ↑*Adressraum* eines Prozesses zugänglich sind. Beispiel für letzteres ist das Ziehen einer Sicherungskopie (*backup*) von im ↑*Hauptspeicher* liegende Datenstrukturen, um ein Zurückrollen

(*rollback*) des Prozesses im Rahmen der Erholung (*recovery*) nach dem Scheitern einer ↑*Aktion* oder einem Systemausfall duchführen zu können. Im einfachsten Fall wird mit der Maßnahme nur der Prozessorstatus gesichert, um einen unterbrochenen Prozess vom Prozessor wegschalten und später auf demselben oder einem anderen Prozessor, jedoch mit demselben ↑*Programmiermodell*, wieder fortsetzen zu können.

Zwischenankunftszeit Zeitspanne zwischen zwei aufeinanderfolgenden Aufträgen an einer Bedienstation (↑*Prozessor*, ↑*Peripheriegerät*), allgemein zwischen zwei Ereignissen derselben Art (↑*Unterbrechung*).

Zwischenspeicher Schneller Pufferspeicher zur Verbergung der ↑*Latenzzeit* für Zugriffe auf den im Vergleich zur ↑*Zykluszeit* des Prozessors langsameren ↑*Arbeitsspeicher*. Die Puffergröße ist ganzzahlige Vielfache der Größe einer ↑*Zwischenspeicherzeile*.

Zwischenspeicherzeile Kleinste Verwaltungseinheit im ↑*Zwischenspeicher*, deren Größe ganzzahlige Vielfache der Größe von einem ↑*Speicherwort* ist. Jede dieser Einheit ist die Kopie des Inhalts eines entsprechend großen Bereichs im ↑*Hauptspeicher*. Der Zugriff auf den Inhalt eines nicht im Zwischenspeicher liegenden Speicherworts bewirkt den blockweisen Transfer der assoziierten Verwaltungseinheit. Wichtiger Aspekt für die ↑*Performanz* dabei ist die ↑*Granularität* der Zeile und die jeweiligen Eigentümerschaften der Originalspeicherworte in ihr (↑*false sharing*).

Zykluszeit Zeitspanne vom Start bis zur Vollendung einer Operation oder Operationsfolge.