Disclaimer

Die Übungen die hier gezeigt werden stammen aus der Vorlesung *Advanced Operating Systems*! Für die Korrektheit der Lösungen wird keine Gewähr gegeben.

1. Funktionale und nichtfuntionale Eigenschaften
   (a) Was ist eine nichtfunktionale Eigenschaft? Finden Sie Beispiele für sowohl funktionale als auch nichtfunktionale Eigenschaften
   - eines Flugzeugs,
   - eines Smartphones,
   - eines Betriebssystems.

   Solution:

   - Flugzeugs
     - F: fliegen, bremsen,
     - NF: Innentemperatur halten, automatisierte Steuerung
   - Smartphones
     - F: telefonieren ermöglichen, Internetzugang
     - NF: klein, leicht, energiesparend, strahlungsarm, umweltfreundlich,
   - Betriebssystems
     - F: den Zugriff auf Daten ermöglichen
     - NF: leicht zu bedienen, skalierbar, offen, performant,

   (b) Charakterisieren Sie den Unterschied zwischen Laufzeiteigenschaften und Evolutionseigenschaften anhand je einer beispielhaften Eigenschaft.

   Solution: Laufzeiteigenschaften Verfügbarkeit: während das System aktiv ist (läuft) muss es verfügbar sein. Überwachbar für ein einzelnes System nur über kurze Zeit.
   Evolutionseigenschaft Wartbarkeit: das System kann über eine längere Zeit überarbeitet/verbessert werden auch ohne dass dieses aktiv ist oder laufen muss. Überwachbar über eine Reihe von Systemen und nur eine lange Zeit

2. Anforderungen an Betriebssysteme
   (a) Welche grundlegenden funktionalen Eigenschaften muss jedes Betriebssystem qua definitione besitzen?

   Solution:

   - Harware Multiplexen
   - Hardware Schutz
   - Harware Abstraktion

(b) Die Erfüllung spezieller nichtfunktionaler Eigenschaften kann Auswirkungen auf den gesamten Hard- und Softwarestack eines IT-Systems haben. Kennen Sie Anwendungsfälle für Betriebssysteme, die

- keinen Scheduler,
- kein Paging,
- keinen privilegierten Prozessormodus

benutzen? Recherchieren und begründen Sie, warum solche Designs möglich und sinnvoll sein können!

Solution: Embedded Systems mit nur einem Programm verwenden kein Scheduler oder Paging

3. Sparsamkeitsbegriff

(a) Erläutern Sie die Begriffe „Sparsamkeit" und „Effizienz". Nennen und begründen Sie in diesem Zusammenhang mögliche Ziele eines sparsamen Betriebssystems.

Solution:

(b) Ist Sparsamkeit grundsätzlich als Laufzeit- oder als Evolutionseigenschaft anzusehen? Begründen Sie anhand Ihrer Antwort auf Frage a).

Solution:

(c) Begründen Sie anhand selbstgewählter Anwendungsszenarien, wann Effizienz im Umgang

- mit Energie
- mit Speicherplatz

eine zentrale nichtfunktionale Eigenschaft eines Betriebssystems ist.

Solution:

(d) Sparsamkeit ist auch eine mögliche Anforderung an den Betrieb eines IT-Systems. Können Sie sich vorstellen, warum dabei „Sparsamkeit mit Funktionalität" oder „Sparsamkeit mit Code" eine Rolle spielen könnte? Bewerten Sie den aktuellen Linux-Mainline Kernel1 nach diesen Kriterien. . .

Solution:

4. Energieeffizienz

(a) Welche hardwareseitigen Voraussetzungen müssen für energieeffizienten Betrieb eines Rechnersystems vorliegen? Welche Rolle spielt das Betriebssystem hierbei?

Solution: Hardware kann Energiesparmodi bereitstellen (sleep mode) und das Betriebssystem muss diese einstellen/umstellen können

(b) Erklären Sie die Begriffe Reaktivität und Nutzererfahrung vor dem Hintergrund Ihrer Antwort auf Frage a).

Solution: Aus dem Energiesparenden Modi wird die Reaktivität niedriger sein aber kann durch das Betriebssystem für die jeweilige Anwendung angepasst wieder in einen Aktiven Modi versetzt werden. Z.B. anpassung der Framerate einer GUI oder Grafik-intensiven Anwendung

5. Speichereffizienz

(a) Erläutern Sie den Begriff Fragmentierung bei Realspeicherverwaltung. Warum kann dies auch für virtuelle Speicherverwaltung zum Problem werden?

Solution: Fragmentierung = bei blockorientierten Datenträgern die Verteilung zusammengehöriger Daten auf nicht aufeinander folgende Datenblöcke. Allgemein die Zerstückelung oder Zergliederung von Speicherbereichen

(b) Welchen Einfluss hat die Seitentabelle für virtuelle Speicherverwaltung auf den unmittelbaren Speicherbedarf eines Betriebssystem-Kernels? Welchen haben etwaige Gerätetreiber? Und welche Möglichkeiten hat ein BS-Entwickler, mit beiden Problemen umzugehen?

Solution:

6. Betriebssystemarchitekturen für Sparsamkeit und Effizienz

Welche Vor- und Nachteile haben die beiden Architekturkonzepte Makro- und Mikrokernel für Entwurf und Implementierung energie- und speichereffizienter Betriebssysteme? Diskutieren Sie anhand der Betriebssysteme TinyOS und RIOT!

Solution:

7. Energieeffiziente Dateizugriffe Nehmen Sie an, eine Anwendung referenziert eine Folge von Festplattenblöcken: A, B, C, A, C, D, E, A. Zur Optimierung von sowohl Performanz als auch Energieverbrauch beim Zugriff auf diese Blöcke soll energieeffizientes Prefetching zum Einsatz kommen. Nehmen Sie weiterhin an, dass ein Blockzugriff (access) jeweils konstant 4 Zeiteinheiten (ZE) dauert, ein fetch oder prefetch je 1 ZE sowie dass der verwendete Festplattencache maximal 3 Blöcke fasst. Die gegebene Referenzfolge können Sie als gesichert annehmen (d. h. es sind keine Abweichungen im tatsächlichen Verhalten der Anwendung zu erwarten).

(a) Entwickeln Sie zwei Zugriffsabläufe für diese Referenzfolge: einen, bei dem traditionelles Prefetching angewandt wird, sowie einen weiteren mit dem in Vorlesung besprochenen Verfahren zum engergieeffizienten Prefetching.

Solution:

(b) Begründen Sie anhand des Vergleichs beider Abläufe, welche Vorteile das energieeffizente Verfahren in diesem Beispiel hat. Kann es Referenzfolgen geben, bei denen beide Verfahren (nahezu oder völlig) gleichen Energieverbrauch verursachen? Unter welchen Umständen, bzw. warum nicht?

Solution:

8. Energieeffizientes Scheduling

Schedulingalgorithmen, welche die optimale Reaktivität eines interaktiven Systems zum Ziel haben, basieren in der Regel auf Round Robin (RR) unter Einbeziehung von Prioritäten. Wenn zugleich Strategien zum sparsamen Umgang mit Prozessorenergie durchgesetzt werden, können solche Algorithmen ihr Ziel jedoch verfehlen.

(a) Begründen Sie obige These anhand einer geeignet konstruierten, beispielhaften Prozessmenge.

Solution:

(b) Das Schedulingziel von RR („maximale Reaktivität") lässt sich als die möglichst faire Verteilung von Rechenzeit auf Threads beschreibt. Finden Sie eine analoge Beschreibung des Schedulingziels „Energieeffizienz" im Sinne der Antwort auf Frage a). Definieren Sie hierfür einen formalen Schedulingparameter analog zur Zeitscheibenlänge T bei RR.

Solution:

(c) Implementieren Sie die einfache RR-Strategie (ohne Prioritäten) so, dass beide in Frage b)beschriebenen Schedulingziele verfolgt werden. Beachten Sie insbesondere, dass

- stark interaktive Threads durch verlängerte Antwortzeiten behindert sowie
- energiesparsame Threads durch relativ geringere Prozessorzeit benachteiligt werden können.

Solution:

Als Ablaufumgebung zur Simulation einer Prozessmenge können Sie entweder ein kleines multi-threaded Programm schreiben oder den Scheduling-Simulator PSSAV verwenden, den Sie evtl. noch aus der zweiten Übung im Grundlagenfach Betriebssysteme kennen. Im letzteren Fall können Sie bereits auf eine RR-Implementierung zurückgreifen; diese und die vom Simulator vorgesehenen Prozessinformationen müssen natürlich modifiziert werden. Stellen Sie Ihren Algorithmus (in knappem Pseudocode) im Seminar vor und zeigen Sie möglichst eine kleine Demonstration. Gehen Sie dabei auf die in der Vorlesung gestellten Implementierungsfragen ein. Tipps: Zur Simulation des Energieverbrauchs eines ablaufenden Prozesses dürfen Sie annehmen, dass dieser linear abhängig von der Prozesslaufzeit ist. Sie müssen daher dieses Prozessmerkmal lediglich als Faktor repräsentieren, der bspw. das verbleibende Energiebudget zum Zeitpunkt des Ablaufs einer Zeitscheibe errechnet oder bestimmt, wann einzum Prozessstart gegebenes Energiebudget aufgebraucht sein wird (als zusätzlichen Schedulingzeitpunkt).

9. Begriffe

(a) Was verstehen wir unter einem „Ausfall"? Erläutern Sie, wie es im abstrakten Fehlermodell nach Laprie[1] zu einem Ausfall kommen kann. Wo können demzufolge Betriebssystemmechanismen zum Erreichen von Robustheit ansetzen?

Solution:

(b) In Politik und Gesellschaft ist oft vom Datenschutz die Rede, wenn es um die Sicherheitseigenschaften von IT-Systemen geht. Analysieren Sie diesen Begriff aus technischer Sicht: Was verstehen Sie (als Informatiker) unter „Daten" und deren „Schutz"? Welche Differenzen ergeben sich zum Wortsinn von „Datenschutz"? Welche grundlegenden Sicherheitsziele sind aus Ihrer Sicht damit gemeint?

Solution:

10. Isolation: Welche Isolationsmechanismen in Betriebssystemen kennen Sie? Diskutieren Sie für jeden bislang behandelten Isolationsmechanimus,

(a) welche konkreten Ziele dieser verfolgt,

Solution:

(b) auf welcher Idee bzw. welchem Prinzip er basiert,

Solution:

(c) welche Kosten er mit sich bringt.

Solution:

11. Referenzmonitorprinzipien: In der Vorlesung wurden bereits die Referenzmonitorprinzipien als Kriterien dafür diskutiert, wie weit die Politikimplementierung der Linux-Sicherheitsarchitektur die nicht-funktionale Eigenschaft Security unterstützt. Wiederholen Sie nun diese Diskussion von

- Unumgehbarkeit,
- Manipulationssicherheit,
- Verifizierbarkeit

für die SELinux-Sicherheitsarchitektur.

Solution:

12. Zugriffssteuerungsmechanismen: Traditionell werden IBAC-Sicherheitspolitiken mittels Zugriffssteuerungsmatrizen (ACMs) modelliert. Zu deren effizienter Implementierung in Betriebssystemen kommen aber verteilte Zugriffssteuerungslisten (ACLs) zum Einsatz.

[1] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing

(a) Implementieren Sie am Beispiel eines Unix-Dateisystems eine IBAC-Zugriffssteuerungspolitik mittels ACLs. Ihre Lösung soll enthalten:

- eine rudimentäre I-Node-Liste, welche lediglich I-Node-IDs Ihrer ACL-Datenstruktur zuordnet (Sie sollen selbstverständlich kein Dateisystem implementieren!),
- eine rudimentäre PCB-Datenstruktur, welche genau die für die Zugriffssteuerung relevanten Metainformationen enthält,
- eine Implementierung der Zugriffsoperationen read (lesen), write (schreiben) und chmod (Rechte ändern)
- Denken Sie an die Unix-Besonderheiten hinsichtlich der Nutzer-ID 0 für root!

Solution:

(b) Wo liegt der Unterschied zwischen einer zentralen, zweidimensionalen (tabellarischen) Datenstruktur zur direkten Implementierung einer ACM und der dezentralen, listenartigen Datenstruktur zur Implementierung von ACLs? Diskutieren Sie jeweils Vor- und Nachteile anhand Ihrer Implementierung!

Solution:

13. Echtzeitsysteme

(a) Was ist ein Echtzeitsystem? Wie unterscheiden sich Echtzeit- von Nichtechtzeitsystemen?

Solution:

(b) Ist Performanz (im Sinne einer möglichst kurzen Antwortzeit) ein hinreichendes, notwendiges, oder gar kein Kriterium für Echtzeitfähigkeit?

Solution:

(c) Welche Arten von Fristen kennen Sie? Nennen Sie je eine typische Beispielanwendung.

Solution:

14. Echtzeit-Scheduling

(a) Ist Round Robin (RR) eine geeignete Strategie zum Echtzeitscheduling? Warum/warum nicht?

Solution:

(b) Würde sich Ihre Antwort auf die vorherige Frage ändern, wenn man RR mit einem Prioritätenschema versieht? Falls ja: unter welchen Bedingungen? Falls nein: begründen Sie.

Solution:

(c) Welchen Parameter müssen Echtzeit-Schedulingstrategien optimieren?

Solution:

15. Scheduling periodischer Prozesse

(a) Vergleichen Sie die Eigenschaften von EDF und RM: Welche Vorteile (Nachteile?) hat der Einsatz der dynamischen Strategie gegenüber der statischen?

Solution:

(b) Überprüfen Sie die Planbarkeit folgender periodischer Prozessmenge durch RM anhand der oberen Schranke des Prozessorauslastungsfaktors (U_{lub}):

	C_i	T_i
τ_1	2	7
τ_2	1	10
τ_3	3	8

Solution:

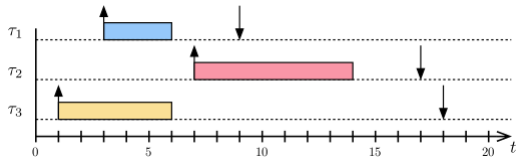
(c) Ergänzen Sie die Prozessmenge der vorherigen Teilaufgabe um einen Prozess τ_4 mit $C_4 = 3$,$T_4 = 20$. Wiederholen Sie die Planbarkeitsanalyse und interpretieren Sie das Ergebnis.

Solution:

(d) Entwickeln Sie für die Prozessmenge aus Teilaufgabe 2 jeweils den Schedule für EDF und RM. Modifizieren Sie anschließend Ihre Lösung, unter Berücksichtigung Ihrer Planbarkeitsanalyse aus Teilaufgabe 3, indem Sie τ_4 aufnehmen.

Solution:

16. Prioritätsumkehr: Die unten abgebildeten aperiodischen Prozesse $\tau_1 ... \tau_3$ sind nach EDF planbar (aufsteigende Pfeile bedeuten Ankunft, absteigende Pfeile Frist eines Prozesses).



(a) Entwickeln Sie mittels EDF einen Schedule für diese Prozessmenge.

Solution:

(b) Sowohl τ_1 als auch τ_3 möchten denselben kritischen Abschnitt betreten, beispielsweise in denselben Puffer eines Geräteregisters schreiben. Der Eintritt in den kritischen Abschnitt erfolgt bei τ_1 nach 2 Zeiteinheiten (ZE) Rechenzeit, bei τ_3 nach 1 ZE Rechenzeit.

Die Verweildauer im kritischen Abschnitt beträgt bei τ_1 1 ZE, bei τ_3 4 ZE. Modifizieren Sie den Schedule unter Berücksichtigung der hierdurch entstandenen kausalen Abhängigkeiten. Diskutieren Sie an diesem Beispiel das Problem der Prioritätsumkehr und seine Folgen.

Solution:

(c) Lösen Sie das in Teilaufgabe 2 beschrieben Problem mittels Prioritätsvererbung und modifizieren Sie Ihren Schedule entsprechend.

Solution:

17. Cyclic Asynchronous Buffers: Als Datenstruktur für echtzeitkritische Kommunikation, etwa als Alternative zu linearen Message-Queues, werden in Betriebssystemen sog. `cyclic asynchronous buffers` (CAB, dt. „Ringpuffer") implementiert. Ihre Aufgabe ist es nun, einen solchen CAB der Länge n zu implementieren. Demonstrieren Sie anschließend, über eine kleines (Kommandozeilen-) Programm das Auslesen einer Folge von $> n$ Nachrichten durch

    - 1 Empfänger, welcher schneller liest als der Sender schreibt
    - 1 Empfänger, welcher langsamer liest als der Sender schreibt
    - $>$ 1 Empfänger mit unterschiedlichen relativen, optional sogar variablen Geschwindigkeiten.

    Um die Anschaulichkeit zu verbessern, benutzen Sie Nachrichtenfolgen der Länge $i * n (i > 1, i \in \mathbb{N})$ oder lassen Sie die einzelnen Threads endlos rechnen und interaktiv bzw. mittels Prozessmanagement abbrechen.
    Diskutieren Sie anhand Ihrer Implementierung Voraussetzungen und Besonderheiten eines CAB verglichen mit linearen, endlichen Warteschlangen. Gehen Sie dabei insbesondere auf die Notwendigkeit von Synchronisationsoperationen ein (vgl. klassisches Reader-Writer-Problem) sowie auf die Eignung hinsichtlich echtzeitkritischer Prozesse als Kommunikationspartner.

    Solution:

18. Adaptivität und komplementäre nichtfunktionale Eigenschaften

    (a) Erklären Sie anhand selbstgewählter Anwendungsbeispiele, warum Adaptivität zur Umsetzung der komplementären nichtfunktionalen Eigenschaften

    - Echtzeitfähigkeit,
    - Robustheit,
    - Wartbarkeit und Portabilität

    beiträgt.

    Solution:

(b) Nennen Sie jeweils eine für diese Eigenschaften geeignete, adaptive Architektur eines Anwendungs-Laufzeitsystems. Ordnen Sie diese in die grundlegende, funktionale Schichtung aus Hardware, Betriebssystem (-kernel), API, User-Space-Ressourcen und Anwendungen ein.

Solution:

19. Exokernel und Virtualisierung

(a) Ein erklärtes Ziel der Exokernel-Architekturen besteht darin, die im Kernel implementierten Betriebssystemfunktionen so klein und wenig komplex wie möglich zu halten. Warum wurde hierfür ein neues Architekturparadigma geschaffen, anstatt die mit einem ähnlichen Ziel geschaffenen Mikrokernarchitekturen zu nutzen?

Solution:

(b) Wodurch unterscheidet sich ein Paravirtualisierungs-Hypervisor von klassischen Typ-1-Hypervisoren? Welche (über Typ-1-Virtualisierung hinaus gehenden) Ziele verfolgt man dabei, und wo liegen die Kosten dieses Ansatzes?

Solution:

(c) Diskutieren Sie, in wieweit sich die Ideen von TinyOS und MirageOS unterscheiden. Gehen Sie dabei auf Gemeinsamkeiten und Unterschiede in deren Zielen sowie in den für beide Betriebssysteme notwendigen Hard- und Softwarevoraussetzungen ein.

Solution:

20. Parallelität in Betriebssystemen

(a) Wo liegt der prinzipielle Unterschied zwischen den Parallelisierungstechniken der Superskalarität und Multithreading?

Solution:

(b) Welche prinzipiellen Aufgaben hat ein Betriebssystem bei der Parallelisierung von Anwendungen?

Solution:

(c) Am Beispiel der Linux-Kernels haben wir verschiedene Implementierungen von Locks als Synchronisationsmechanismus kennen gelernt. Dabei waren atomare Einzeloperationen (atomic_*) die mit Abstand feingranularste, effizienteste und zugleich fehlerunanfälligste Lösung. Warum benutzt man überhaupt andere und komplexere Arten von Locks? Wo liegen deren jeweilige Stärken/Schwächen?

Solution:

(d) Die in der Vorlesung beispielhaft am Linux-Kernel vorgestellten Arten von Locks finden sich in ähnlicher Form auch in anderen monolithischen Betriebssystemen (beispielsweise kennt MacOS die Lock-Arten Mutex, Semaphore und Spinlock). Für Mikrokernarchitekturen hingegen wurde gezeigt, dass bei geschicktem Kerneldesign ein globales Lock (big kernel lock, BKL) den feingranularen Alternativen in puncto Performanz sogar überlegen ist [2]. Warum?

Solution:

21. Parallelisierbarkeit und Effizienz
In der Vorlesung haben Sie das Gesetz von Amdahl kennen gelernt, welches die theoretische Verringerung der Berechnungszeit für ein Problem mit gegebenem sequenziellem Anteil f beschreibt. Auf dieser Basis wurde die Beschleunigung (Speedup) $S(n)$ als Quotient aus dem jeweiligen Zeitbedarf voll sequenzieller ($T(1)$) und n-fach paralleler Bearbeitung des Problems definiert, also $S(n) = \frac{T(1)}{T(n)} = \frac{1}{f+\frac{1-f}{n}}$. In dieser Aufgabe sollen Sie die Auswirkungen dieses Zusammenhangs auf ein vereinfachtes Problem aus der Praxis untersuchen. Gegeben sei folgendes Szenario:

Ein Unternehmen bietet als Cloud-basierte Dienstleistung die Berechnung großer Datenmengen (z.B. Marktforschungsinformationen) für einen festen Kundenstamm an. Hierfür wurde Hardware für das hauseigene Rechenzentrum ausgewählt und so dimensioniert, dass höchstens x Rechenaufträge pro Zeiteinheit innerhalb der jedem Kunden zugesicherten Latenzen bearbeitet werden können (Durchsatzgarantie). Durch die Einstellung eines branchenerfahrenen Marketingexperten konnte kürzlich die Anzahl der Neukunden jedoch überdurchschnittlich gesteigert werden, so dass für das kommende Geschäftsjahr neue technische Herausforderungen erwartet werden: unter Beibehaltung sämtlicher Durchsatzgarantien muss das Unternehmen nun bis zu 3 x Rechenaufträge pro Zeiteinheit bearbeiten können. Dieses Problem soll durch höhere Parallelisierung gelöst werden.

(a) Bestimmen Sie den Faktor n, um den die Hardware des Rechenzentrums leistungsfähiger werden muss, um die Durchsatzgarantien weiter erfüllen zu können (z. B. durch die Ver-n-fachung der eingesetzten Prozessoren). Bestimmen Sie hierfür zunächst den erforderlichen Speedup, berechnen Sie dann n unter der Annahme, dass jeder Rechenauftrag einen vertraglich vereinbarten sequenziellen Anteil von 20% nicht überschreitet.
Hinweis: Behandeln Sie für diese Aufgabe die Summe aller Rechenaufträge so, als wäre es eine einzige, zu parallelisierende Berechnung.

Solution:

(b) Wie effizient ist der Einsatz der zusätzlichen Prozessoren aus Teilaufgabe 1? Quantifizieren Sie mittels des in der Vorlesung definierten Leistungsmaßes Wirkungsgrad ($E(n)$).

Solution:

(c) Angenommen, Ihre Antwort auf Teilaufgabe 2 erscheint der Unternehmensleitung zu unwirtschaftlich. Als Kompromiss sollen die Durchsatzgarantien - und damit der zu erzielende Speedup - reduziert werden, um einen optimierten Faktor n_{opt} an Prozessoren

[2] Sean Peters, Adrian Danis, Kevin Elphinstone, and Gernot Heiser. For a Microkernel, a Big Lock Is Fine. In Proceedings of the 6th Asia-Pacific Workshop on Systems

einzusetzen. Bestimmen Sie n_{opt} sowie $S(n_{opt})$ anhand des Maximums der Beschleunigungseffizenz η_{max}.

Solution:

(d) Beurteilen Sie, wie realistisch die in den bisherigen Teilaufgaben erzielten Ergebnisse für das gegebene Problem in der Praxis sind. Welche bisher nicht berücksichtigten Aspekte realer Systeme können Speedup sowie Effizienz der Parallelisierung beeinflussen?

Solution: