Lösungen zum Übungsblatt

Allgemeine Fragen

1. Das Programm ist die statische Aufschreibung von Befehlen zur Steuerung des technischen Prozesses. Der Rechenprozeß ist die dynamische Ausführung des Programms in Abhängigkeit der Anforderungen des zu bedienenden technischen Prozesses.

2. Unter Multitasking versteht man die quasi gleichzeitige Ausführung mehrerer Rechenprozesse auf einem Rechner. Die quasi gleichzeitige Ausführung wird durch Stückelung und Verteilung der verfügbaren Rechenzeit auf die einzelnen Rechenprozesse erreicht. Das Umschalten von einem Rechenprozeß auf den nächsten nennt man Kontextswitch. Dieser Kontextswitch benötigt Rechenzeit, die den eigentlichen Rechenprozessen verloren geht. Das Verhältnis der Kontextswitchzeit zur maximal verfügbaren Rechenzeit inklusive Kontextswitchzeit nennt man Overhead.

3. Overhead = $t_{kontext} / t_{Zeitscheibe}$

- $t_{kontext}$ ist in erster Linie rechnersystemabhängig: Overhead verkleinern durch $t_{kontext}$ verkleinern durch Kauf eines schnelleren Rechners
- $t_{Zeitscheibe}$ ist die mittlere Zeitdauer zwischen zwei Kontextswitchzeitpunkten. Ein Rechenprozeß kann davon die Zeitdauer $t_{zeitscheibe}$-$t_{kontext}$ ungestört am Stück den Rechner nutzen. Overhead verkleinern durch Vergrößern von $t_{Zeitscheibe}$.
- Zu beachten:
  - Kauf eines schnelleren Rechners: sprengt gegebenenfalls den vorgegebenen finanziellen Rahmen beim Auswahl des Rechners
  - Das Vergrößern der $t_{Zeitscheibe}$ hat eine Grenze. Die Zeit darf maximal so groß gewählt werden, daß die Echtzeitbedingung des zu steuernden technischen Prozesses nicht verletzt wird.

4. Unter Zuverlässigkeit versteht man die Fähigkeit der Anlage unter vorgegebener Zeitdauer und zulässigen Betriebsbedingungen die spezifizierte Funktion zu erbringen. Die Sicherheit ist das Nichtvorhandensein einer Gefahr für den Menschen, die Umwelt und den Sachwert. Von einer Anlage kann Gefahr für den Menschen ausgehen obwohl die Anlage zuverlässig ihre Funktion erfüllt. Beispielsweise ist das Fehlen eines Schutzgitters an einer hydraulischen Presse im Betrieb ein Sicherheitsmangel und birgt damit die Möglichkeit unbeabsichtigt Menschen im Falle einer Unachtsamkeit zu verletzen, trotzdem kann die Presse fehlerfrei funktionieren.

5. Realzeitbetrieb ist, wenn der Rechner mit den Vorgängen des technischen Prozessses, die er erfassen und steuern soll, trägheitslos Schritt halten kann.

6. Aufgabe eines Realzeitbetriebsystems ist es den einzelnen Rechenprozessen zu jedem Zeitpunkt die prozeßbedingte erforderliche Rechenzeit just-in-time zur Verfügung zu stellen.

7. Jeder Rechenprozeß hat eine von den anderen verschiedene Wichtigkeit; Priorität genannt. Die Rechnerzuteilung, das Scheduling, erfolgt immer so, daß der Rechenprozeß mit der höchsten Priorität die CPU erhält.

8. Jeder Rechenprozeß bekommt zyklisch wiederkehrend die CPU für eine maximale Zeidauer in Vielfachen einer Zeitscheibe zugeteilt. Er kann seine Zeitscheibe aber freiwillig vorzeitig beenden.

9. Ein Event dient zur Tasksynchronisation. Eine Task kann auf das Auftreten eines Eventwertebereichs, z.B. dem Ablauf einer festen Wartezeit, warten und verbraucht während dieser Zeit keine CPU-Rechenleistung. Mit Auftreten des Ereignisses -Ablauf der Wartezeit- wird die Task geweckt und startet sofort bei ausreichend hoher Task-Priorität ihre Berarbeitungsaufgabe, z.B. die Berechnung und Ausgabe des nächsten Steuerungswertes eines Reglers.

10. Eine Semaphore ist eine besondere Variable, die dazu dient den Zugriff auf exklusiv verfügbare Betriebsmittel zu steuern. Das Belegen und das Freigeben einer Semaphore ist eine unteilbare Operation. Ist das Belegen erfolgreich, so steht das damit verbundene Betriebsmittel dem Besitzer der Semaphore bis zur erneuten Freigabe zur Verfügung. Eine Deadlock-Situation kann bei der Verwendung zweier Semaphore entstehen, wenn die erfolgreiche gleichzeitige Belegung beider Semaphore Bedingung für den weiteren Ablauf zweier unterschiedlicher Tasks ist, die Semaphore nur nacheinander belegt werden können, jeweils eine Task bereits eine der beiden Semaphore belegt hat und nun auf das Freiwerden des jeweils anderen Semaphors wartet.

11. siehe Vorlesungsunterlagen

12. Der Taskkontrollblock beinhaltet die Laufdaten einer Task, die vom RBS benötigt werden um die Task im System zu verwalten. Die Laufdaten umfassen: beschreibende Parameter -z.B. TaskID, Priorität,Rechenzeit- , Laufparameter - Prozeßkontext = Speicherplatz für die CPU-Register , Verwaltungsdaten -Verkettungszeiger für Wartelisten-

13. Der **neue Rechenprozeß in OS9 wird identifiziert über einen eineindeutigen Modulnamen**.

a) Befindet sich bereits ein Programm mit **gleichem Namen im Modulverzeichnis**, so wird der **bereits geladene Programmcode** benutzt, sofern die **Benutzungsrechte** es erlauben, dann ein **neuer Datenbereich** im Arbeitsspeicher (ASP) gesucht und belegt, ein **Taskkontrollblock** neu **angelegt** und in die **Verwaltungstabellen des Scheduler**s **eingetragen**. Danach wird der **Scheduler aufgerufen** und falls die neue Task in diesem Augenblick die höchste Priorität hat, erhält sie die CPU.

b) Befindet sich das **Modul nicht bereits im Arbeitsspeicher so wird eine Programmdatei gleichen Namens entlang des sog. Execution-Pfades** auf dem Massenspeicher **gesucht** und falls vorhanden **vollständig** in den **ASP geladen -wenn die Zugriffsrechte und die Benutzungsrechte es erlauben**. Aus dem **Modulkopf wird der Platzbedarf an Datenbereich** entnommen und der ASP belegt. Der weitere Ablauf ist wie a).

Ist zu irgendeinem Zeitpunkt nicht mehr ausreichender ASP vorhanden, so wird der Vorgang abgebrochen.

14. Ein Petri-Netz /SDL-Diagramm sind graphische Hilfsmittel um zeitliche und logische Abhängigkeiten nebenläufiger Prozesse zu beschreiben.

15. Durch programmgesteuertes Verändern der Priorität einer anderen Task ist es möglich den Zeitpunkt zu bestimmen wann eine Task eine Aufgabe abarbeiten soll. Erhöht man die Priorität so, daß sie über allen anderen liegt und diese Task ist auch noch rechenwillig, so erhält sie durch den Prioritätsveränderungsaufruf sofort die CPU. Erniedrigt man die Priorität so weit, daß sie unter allen anderen liegt, so wird diese Task nie die CPU erhalten, vorausgesetzt es gibt immer andere Tasks die rechenwillig sind.

16. Durch das Signalisieren/Setzen eines Events durch eine Task, mittels RBS Aufruf, werden andere Tasks, die darauf warten, geweckt und rechenwillig. Entsprechend ihrer Priorität erhält dan die höchstpriore Task die CPU. Somit wird durch das Signalisieren/Setzen eines Events der Zeitpunkt gesteuert wann eine Task eine Aufgabe abarbeiten soll.

17. **Shared-Memory:** gemeinsamer Speicherbereich, den sich mehrere Tasks teilen, der Zugriff wird über Semaphore gesteuert. *Vorteil*: sehr schnell, da die Nachricht nicht mehrfach kopiert wird *Nachteil*: in verteilten Systemen, die keinen gemeinsamen Arbeitsspeicher besitzen aufwendig zu realisieren da jedes System immer eine aktuelle Kopie besitzen muß.

Message-Buffers: Das RBS erhält den Auftrag einem Empfänger eine Nachricht zu übermitteln Ł RBS kopiert die Nachricht in einen Message-Buffer, speichert und verwaltet alle in einer Queue bis sie von den Empfängern abgeholt werden.Ł der Empfänger erhält eine Kopie des Messagebuffers. *Vorteil*: einfacher Mechanismus, der auch in vernetzten Systemen ohne Änderung funktioniert Ł Empfangsadresse beinhaltet dann auch indirekt die Empfangssystemkennung. Nachrichtenverwaltung und -speicherung übernimmt das RBSŁ Eine Nachricht kann auch an mehrere Empfänger gleichzeitig geschickt werden. *Nachteil:* Messages müssen kopiert werden, langsamer als shared-memory und Messages die länger sind als die max. Nachrichtengröße müssen irgendwie durch die Sendertask oder RBS zerstückelt werden und beim Empfänger wieder zusammengesetzt werden.

Pipes: Pipes sind spezielle Dateien, die sich wie eine FIFO-Queue beim Lesen und Schreiben verhalten. Schreibt eine Task nacheinander in eine Pipe Daten, so werden diese werden in einer FIFO-Queue gespeichert. Liest eine andere

Task Daten von der Pipe, so erhält sie die Daten in derselben Reihenfolge in der sie eingeschrieben wurden. *Vorteil*: Benutzung wie eine normale Datei des Dateisystems, zeitliche Reihenfolge und Strom -keine Stückelung in Nachrichten nötig- der Daten bleibt erhalten, einfache zeitliche Entkoppelung eines Senders und Empfängers, die über längere Zeit hinweg betrachtet dieselbe mittlere Sende-/Empfangsdatenrate haben, aber kurzfristig durchaus unterschiedliche Datenraten haben können (Burstsendebetrieb, kontinuierliche Empfangsabarbeitung) *Nachteil:* Pipemechanismus ist langsamer als shared-memory und eine Pipe funktioniert nur zwischen einem/mehreren Sender(n) und einem! Empfänger.

18. Die Echtzeittask gibt aktiv -freiwillig- die CPU auf in dem sie auf das Auftreten eines internen oder externen Ereignisses wartet und somit eine andere Task auch mit ggf. niederer Priorität die CPU zugeteilt bekommt. Der RBS warte-Aufruf aktiviert den Scheduler.
Ein externes Ereignis -Interrupt- führt zu einer Unterbrechung der Echtzeittask und in der Interruptroutine wird eine Task geweckt die eine höhere Priorität besitzt wie die gerade laufende Task. In der Endebehandlung des Interrupts durch das RBS wird der Scheduler aufgerufen der dann der höchstprioren rechenwilligen Task die CPU zuteilt.

19. siehe Vorlesung, Erkärung Formeln

20. siehe Vorlesung

21. Ein Signal ist ein Zahlenwert der von einem Sender an einen Empfänger verschickt wird. Es wird hierzu eine vom Empfänger zu installierende Signal-Empfangsroutine, unabhängig vom Taskzustand der Empfangstask, aufgerufen, der der Signalwert als Parameter zur weiteren Bearbeitung übergeben wird. Danach wird die Empfangsroutine der Empfängertask verlassen.

22