Grundlagen der Systemsoftware

Übungsblatt 2 \* Gruppe G02-A \* Back, Behrendt, Stäger \* SoSe 2015

**Aufgabe 1: Grundlagen von Betriebssystemen**

**1. Um welche zwei Grundaufgaben von Betriebssystemen handelt es sich hierbei? Bitte erläutern Sie kurz die jeweils entstehende „Sicht“ auf Betriebssysteme.**

Das Betriebssystem muss sich zum einen um die Verteilung der zur Verfügung stehenden Ressourcen (CPU-Zeit, Speicher, Netzwerk) kümmern, als auch gleichzeitig eine Abstraktion der unterliegenden Details bereit zu stellen, um einen komfortablem Umgang zu ermöglichen.

**2. Welche konkreten Einzelaufgaben für ein Betriebssystem ergeben sich aus diesen zwei unterschiedlichen Sichten auf das Betriebssystem? Bitte nennen Sie mindestens zwei Einzelaufgaben für jede Sicht.**

Aus Sicht der Verwaltung muss das System die Ressourcen an die einzelnen Prozesse und Threads zu verteilen und die angeschlossene Peripherie zu verwalten.

Aus Sicht der Abstraktion muss der Mehrbenutzerbetrieb sichergestellt werden – am besten für den Benutzer nachvollziehbar – und die Hardware entsprechend über allgemeine APIs bereitgestellt werden.

**3. (optional) Vergleichen Sie monolithische und hierarchische Organisation von Betriebssystemen Was sind die Vor- und Nachteile?**

In monolithischen organisierten Betriebssystemen laufen alle Aufgaben mit den gleichen, maximalen Rechten. Dadurch kann die maximale Performance verwendet werden, jedoch existieren keinerlei Schutzmechanismen. In hierarchisch organisierten Betriebssystem werden verschiedene Sicherheitslevel verwendet, um den inneren Kern vor unautorisierten Veränderungen zu schützen. Diese Schutzmaßnahmen resultieren schlechterer Performance, da die Zugriffe überwacht werden müssen.

**4. (optional) Beschreiben Sie kurz die Unterschiede zwischen Netzwerk-Betriebssystemen und verteilten Betriebssystemen. Gehen Sie dabei auch auf die unterschiedliche Ausprägung der Verteilungstransparenz ein.**

Ein Netzbetriebssystem besteht aus einem oder mehreren Servern, die jeweils eigenständig ein oder mehrere Dienste bereitstellen. Im Gegensatz dazu unterstützt ein verteiltes Betriebssystem alle angeschlossenen Computer und steuert deren Abläufe. Für den Benutzer ist dies intransparent.

Aufgabe 2: Prozesse und Threads

1. Beschreiben Sie in jeweils max. 2 Sätzen die Begriffe Programm, Prozess und Thread und setzen Sie dabei die Begriffe in Beziehung zueinander.

Ein Programm ist eine binäre Datei, das eine Abfolge von Maschinenbefehlen enthält um eine Tätigkeit bereit zu stellen.

Ein Prozess wird vom Betriebssystem angelegt um einem Programm auf dem Computer einen Identifier, Berechtigungen und Ressourcen zuzuteilen.

Ein Thread ist die verarbeitende Einheit um die Maschinenbefehle des Prozesses tatsächlich auszuführen. Jeder Prozess hat einen Masterthread und kann mehrere weitere Threads starten.

1. (optional) Wie können sich verschiedene Threads gegenseitig negativ beeinflussen?

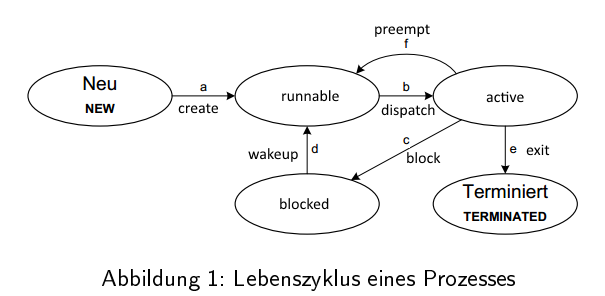
Threads greifen auf den gleichen lokalen Adressraum zu und können damit die Daten verändern, die ein anderer Thread noch braucht. Es kann zu auch zu race-conditions kommen.

1. (optional) Beschreiben Sie zwei Fälle, in denen es sinnvoll ist, nebenläufige Programmteile mit Hilfe von Threads anstatt von Prozessen zu implementieren.

Bei Programmen die zuerst eine große Menge von Daten laden bzw. berechnen müssen, um einen Algorithmus anzuwenden, lohnt sich die Verwendung von Threads bei wiederkehrenden Aufgaben. Beispiel wäre ein Navigationsprogramm, das erst die Kartendaten lesen muss. Hierdurch wird an Speicherplatz gespart, da die Daten nur einmal geladen werden müssen.

Auch bei Servern, die auf Verbindungen von mehreren Klienten gleichzeitig reagieren müssen, sind Threads notwendig, da so die interne Kommunikation viel leichter abgewickelt werden kann und eben echt gleichzeitig auf die verschiedenen Klienten reagiert werden kann.

1. Lebenzyklus

Ein Prozess wird als erstes erzeugt (create) und geht dann in den Status runnable über. In diesem Status arbeitet Prozess jedoch noch nicht. Er ist nur bereit. Sobald der Prozess dispatcht wird, arbeitet er seine Befehle im Status active ab – dem Prozess stehen erst jetzt die Ressourcen zur Verfügung. Durch preempt kann er zurück in den Status runnable gebracht werden oder eben er blockiert sich, kommt in den Status blocked (z.B. um auf eine andere Ressource zu warten) und wird wieder aufgeweckt, sobald die Ressource bereit ist. Er bleibt dann allerdings runnable, und wird nicht direkt wieder gestartet. Aus dem Status active kann der Prozess nach beenden noch beenden (terminiert).