## 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 (Betriebsmittelverwalter), als auch gleichzeitig eine Abstraktion der unterliegenden Details bereitzustellen (da jetzt erweiterte / virtuelle Maschine), 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 verteilen und die angeschlossene Peripherie 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 in 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 (lose gekoppelt). Im Gegensatz dazu unterstützt ein

verteiltes Betriebssystem alle angeschlossenen Computer und steuert deren Abläufe (eng gekoppelt). 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 bereitzustellen.
Ein Prozess wird vom Betriebssystem angelegt um einem Programm auf dem Computer einen Identifier, Berechtigungen und Ressourcen zuzuteilen. Außerdem sorgt das Betriebssystem dafür, dass die Verarbeitungsschritte in eine höhere Speicherhierarchie geladen werden.
Ein Thread ist die verarbeitende Einheit um die Maschinenbefehle des Prozesses tatsächlich auszuführen. Jeder Prozess hat einen Masterthread und kann mehrere weitere Threads starten.

**2. (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.

**3. (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.

**4. Ein Prozess durchläuft in seinem Lebenszyklus (siehe Abbildung 1) verschiedene Zustände, beginnend mit der Erzeugung des Prozesses und endend mit dessen**

**Terminierung. Vervollständigen Sie den Zustandsübergangsgraphen indem Sie die fehlenden Zustände X, Y und Z benennen und in jeweils max. 2 Sätzen deren Bedeutung beschreiben. Erläutern Sie außerdem die jeweilige Bedingung (bzw. Ereignis) a-f, die zu einem Zustandsübergang führt (max. 2 Sätze pro Bedingung).**

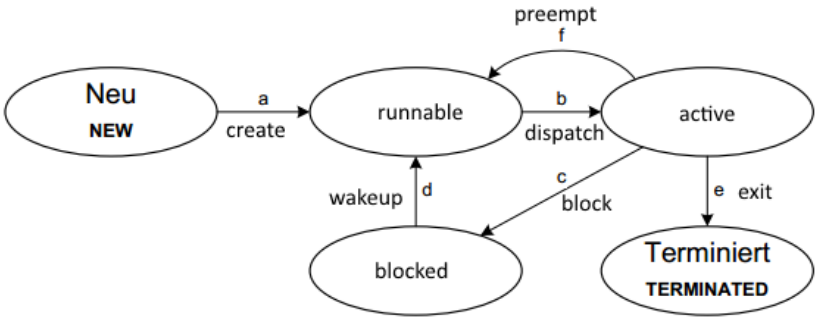


Abbildung 1: Lebenszyklus eines Prozesses

Ein Prozess wird als erstes erzeugt (create) und geht dann automatisch in den Status runnable über. In diesem Status arbeitet der Prozess jedoch noch nicht. Er ist nur bereit. Sobald dem Prozess Ressourcen zugeteilt werden, wird er dispatched und arbeitet seine Befehle im Status active ab. Per preempt werden dem Prozess wieder die Ressourcen entzogen und der Prozess wird in den Status runnable überführt. Alternativ kann er in den Status blocked wechseln, falls er auf eine andere Ressource warten muss. Er 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 über die exit-Kante beenden (terminiert).

## Aufgabe 3: n-Adressmaschine

**Der Ausdruck $R = \frac{a_1+a_2}{a_3} + \frac{b_1-b_2}{b_3}$ sei direkt (d.h. ohne Umformung bzw. Optimierung) mit folgenden Maschinenarten zu berechnen:**

a) **Eine 2-Adressmaschine (mit Überdeckung des zweiten Operanden)**
b) **(optional) eine 1-Adressmaschine (mit Überdeckung des Akkumulator-Inhaltes AC)**
c) **(optional) eine 0-Adressmaschine (Keller-Rechner)**

**Die Benutzung von Hilfsspeicherzellen (keine Register) sei bei der Berechnung erlaubt. Ermitteln Sie – in Anlehnung an die Beispiele für die Arbeitsweise einen n-Adressmaschine im Abschnitt B1b der GSS-Vorlesung – jeweils**

- **die Befehlsfolge mit zugeordneter Wirkung (bitte skizzieren),**
- **die Anzahl der Leseaufträge an den Speicher (bei jeweils direkter Adressierung der Hauptspeicherzellen),**

- **die Anzahl der Schreibaufträge an den Speicher (bei jeweils direkter Adressierung der Hauptspeicherzellen) sowie**
- **die Berechnungszeit für den Ausdruck (unter den vereinfachenden Annahmen, dass die Ausführungszeit eines Maschinenbefehls nur 5% eines Speicherzugriffs beträgt und die Befehle für Addition, Subtraktion und Division gleich viel Zeit kosten).**

a) 2-Adressmaschine:

| | ***Befehlsfolge*** | | | ***Wirkung*** |
|---|---|---|---|---|
| (1) | **->** | >a3< | >H1< | H1 := a3 |
| (2) | **->** | >a2< | >H2< | H2 := a2 |
| (3) | **+** | >a1< | >H2< | H2 := a1 + H2 |
| (4) | **/** | >H2< | >H1< | H1 := H2 / H1 |
| (5) | **->** | >b3< | >H2< | H2 := b3 |
| (6) | **->** | >b2< | >H3< | H3 := b2 |
| (7) | **-** | >b1< | >H3< | H3 := b1 - H3 |
| (8) | **/** | >H3< | >H2< | H2 := H3 / H2 |
| (9) | **+** | >H2< | >H1< | H1 := H2 + H1 |
| (10) | **->** | >H1< | >R< | R := H1 |

***15 Leseaufträge*** an den Speicher

***10 Schreibaufträge*** an den Speicher

Dieses Programm benötigt 25,25 Zeitschritte, da ein Lese- und Schreibauftrag jeweils als ein Zeitschritt gelten und jede der fünf arithmetischen Operationen 5% eines Zeitschrittes dauert, also insgesamt 25%.

b) 1-Adressmaschine:

| | ***Befehlsfolge*** | | ***Wirkung*** |
|---|---|---|---|
| (1) | **lade** | >a1< | AC := a1 |
| (2) | **+** | >a2< | AC := AC + a2 |
| (3) | **/** | >a3< | AC := AC / a3 |
| (4) | **speichere** | >H1< | H1 := AC |
| (5) | **lade** | >b1< | AC := b1 |
| (6) | **-** | >b2< | AC := AC - b2 |
| (7) | **/** | >b3< | AC := AC / b3 |
| (8) | **+** | >H1< | AC := AC + H1 |
| (9) | **speichere** | >R< | R := AC |

***7 Leseaufträge*** an den Speicher

***9 Schreibaufträge*** an den Speicher

Dieses Programm benötigt 16,25 Zeitschritte, da es 16 Lese- bzw. Schreibaufträge durchführt und wie oben fünf arithmetische Operationen durchführt.

c) 0-Adressmaschine:

| | ***Befehlsfolge*** | ***Kellerinhalt*** | | |
|---|---|---|---|---|
| (1) | **PUSH** >a1< | a1 | | |
| (2) | **PUSH** >a2< | a1 | a2 | |
| (3) | **ADD** | a1+a2 | | |
| (4) | **PUSH** >a3< | a1+a2 | a3 | |
| (5) | **DIV** | (a1+a2)/a3 | | |
| (6) | **PUSH** >b1< | (a1+a2)/a3 | b1 | |
| (7) | **PUSH** >b2< | (a1+a2)/a3 | b1 | b2 |
| (8) | **SUB** | (a1+a2)/a3 | b1-b2 | |
| (9) | **PUSH** >b3< | (a1+a2)/a3 | b1-b2 | b3 |
| (10) | **DIV** | (a1+a2)/a3 | (b1-b2)/b3 | |
| (11) | **ADD** | (a1+a2)/a3+ (b1-b2)/b3 | | |
| (12) | **POP** >R< | | | |

***6 Leseaufträge*** an den Speicher

***12 Schreibaufträge*** an den Speicher

Dieses Programm benötigt 18,25 Zeitschritte, da es insgesamt 18 Lese- und Schreibaufträge durchführt und wie bisher 6 arithmetische Operationen besitzt.