GSS Abgabe 02

Carolin Konietzny, Paul Bienkowski, Julian Tobergte, Oliver Sengpiel, Lars Thoms

29. April 2015

1 (Grundlagen von Betriebssystemen)

a)
- **Abstraktion**: Das Betriebssystem bietet ein „schönes" (aufgeräumt, übersichtlich, verständlich, standardisiert) Interface zur Kommunikation mit der Hardware. Dies macht es Anwendungsentwicklern einfacher damit zu interagieren.
- **Resourcenmanagement**: Das Betriebssystem weist den Programmen Speicher und Rechenzeit zu. Es verwaltet Gerate und Dateien.

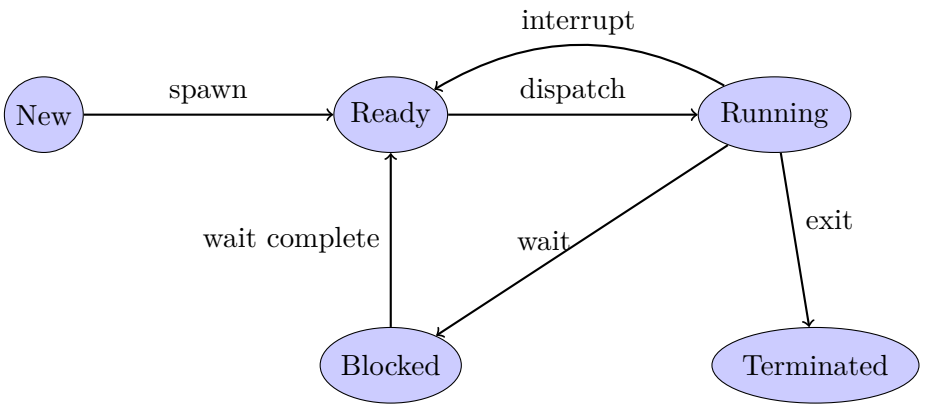
b)
- **Abstraktion**: Verschiedene Hardwaremodule des gleichen Typs (z.B. USB-Sticks versch. Hersteller oder WLAN/Ethernetkarte als verschiedene Netzwerkadapter) können über die gleiche Schnittstelle angesprochen werden. Damit wird die Komplexität der eigentlichen Maschine versteckt.
- **Resourcenmanagement**: Der Hauptspeicher wird verwaltet, den Programmen werden verschiedene Speicherbereiche zugewiesen. Die Festplatte wird in logische Segmente unterteilt, das Dateisystem verwaltet Dateien als logische, benannte Einheiten.

2 (Prozesse und Threads)

a) Programm, Prozess, Thread

Ein Programm ist eine vorher eindeutig definierte Abfolge an Anweisungen an die Maschine, die zuvor von einem Programmierer in der Programmiersprache seiner Wahl niedergeschrieben wurde. Zur Ausführung eines Programmes wird ein Prozess benötigt, welcher vom Betriebssystem erzeugt wird und ihm einen eigenen exklusiven Speicherbereich zuweist. Das Betriebssystem verwaltet auch die Rechenzeit mit Hilfe von Interrupts. Ein Prozess kann explizit durch die Programmierung aus mehreren Threads bestehen, welche auf dem selben Speicherbereich parallel arbeiten können.

d) Lebenszyklus eines Prozesses



Durch die Erzeugung eines Prozesses gelangt dieser erst einmal in den Readystate. Von diesem aus können ihm nun Aufgaben zugeteilt werden (Dispatcher lädt Kontext) und er wechselt in den Runningstate, d.h. er "läuft". Von dort aus kann er entweder unterbrochen (interrupt), angehalten (wait), oder terminiert (exit) werden. Wait unterscheidet sich vom Interrupt insofern, als dass der Prozess beim Wait erst die Nachricht über die Vollendung des Wartens erhalten muss, bevor er wieder wie beim Interrupt in den Readystate gelangt.

3 (n-Adressmaschine)

Wir dürfen keine Hilfsregister verwenden, benötigen aber in einer 2-Adress-Maschine mindestens 2 Register um Berechnungen durchzuführen. Daher bennen wir unsere Register R1 und R2, als Hilfsspeicherzelle verwenden wir H1.

Befehl			R_1	R_2	H_1	Z
LOAD	a_1	R_1	a_1			
LOAD	a_2	R_2	$\vdots$	a_2		
ADD	R_1	R_2	$\vdots$	$a_1 + a_2$		
LOAD	a_3	R_1	a_3	$\vdots$		
DIV	R_2	R_1	$\frac{a_1+a_2}{a_3}$	$\vdots$		
STORE	R_1	H_1	$\vdots$	$\vdots$	$\frac{a_1+a_2}{a_3}$	
LOAD	b_1	R_1	b_1	$\vdots$	$\vdots$	
LOAD	b_2	R_2	$\vdots$	b_2	$\vdots$	
ADD	R_1	R_2	$\vdots$	$b_1 + b_2$	$\vdots$	
LOAD	b_3	R_1	b_3	$\vdots$	$\vdots$	
DIV	R_2	R_1	$\frac{b_1+b_2}{b_3}$	$\vdots$	$\vdots$	
LOAD	H_1	R_2	$\vdots$	$\frac{a_1+a_2}{a_3}$	$\vdots$	
ADD	R_2	R_1	$\frac{a_1+a_2}{a_3} + \frac{b_1+b_2}{b_3}$	$\vdots$	$\vdots$	
STORE	R_1	Z	$\vdots$	$\vdots$	$\vdots$	$\frac{a_1+a_2}{a_3} + \frac{b_1+b_2}{b_3}$

Leseoperationen: 7
Schreiboperationen: 2
Rechenbefehle: 5
Berechnungszeit: $(7 + 2) * 20 + 5 = 185$ FLOP