Betriebssysteme I Vorlesung 06: Prozesse in Windows und LINUX

Multitasking-Betriebssystem (Windows oder LINUX)

In jedem Multitasking Betriebssystem sind nach dem Start mehrere Prozesse gestartet.

Fragen:

- Was sind Kenngrößen der Prozesse
- Wie verwaltet das Betriebssystem die Prozesse
- Was sind die Unterschiede zwischen LINUX und Windows bzgl. den Prozessen

zur Wiederholung: **Ein Prozess ist ein Programm in Ausführung**

Unter einem Programm verstehen wir die Installationsumgebung incl. der ausführbare Datei, Konfigurationsdateien, etc.

In Ausführung ist das Programm, wenn die ausführbare Datei (z.B. die .exe-Datei) in den Hauptspeicher geladen und von der CPU abgearbeitet wird

Kenngrößen eines Prozesses

Die Prozess-ID PID

Eindeutige Zahl zur Identifikation von Prozessen. Die PID ändert sich zur Laufzeit des Prozesses nicht

Der Besitzer eines Prozesses (OWNER)

Der Owner eines Prozesses ist der Benutzer des Betriebssystems, der den Prozess gestartet hat und dem der Prozess gehört. Der Owner des Prozesses ist entscheidend für die Zugriffsrechte auf die Resourcen des Betriebssystems, insbesondere des Dateisystems. Jede Datei und jeder Ordner des Dateisystems besitzt Zugriffsrechte. Die Zugriffsrechte entscheiden, wie und welche Prozesse auf die Datei zugreifen können

Wichtiges Grundprinzip zur Sicherheit (SECURITY) eines Betriebssystems

Prozesse besitzen außerdem eine Vielzahl von Kenngrößen für statistische Zwecke

CPU-Zeit

Allgemeines über Multitasking und Prozessen

Multitasking aus Benutzersicht

Ein Rechner mit einer CPU kann mehrere Aufgaben gleichzeitig erledigen, d.h. mehrere Prozesse gleichzeitig bearbeiten

Multitasking in der Realität

CPU schaltet zwischen den Prozessen hin und her. Ca. 10-100 mal pro Sekunde wechselt die CPU den zu bearbeitenden Prozess

(Da die meisten Prozesse in einem schlafenden Zustand sind, betrifft das Hin- und Herschalten nur die aktiven Prozesse; das sind nur sehr wenige)

Scheduler

Der Scheduler ist der Teil des Betriebssystems, der das Umschalten zwischen den Prozessen bewerkstelligt

Das Scheduling wird noch Gegenstand einer späteren Vorlesung sein

Wie verwaltet das Betriebssystem die Prozesse

Der Kernel besitzt eine sog. **Prozesstabelle** in der alle wichtigen Kenngrößen der Prozesse stehen

Zu den Prozessen gibt es zwei Hauptaufgaben

- zeitliche Management der Prozesse (Scheduling)

Wann bekommt ein Prozess die CPU, d.h. wann werden die Prozesse hin- und hergeschaltet. Das Umschalten der Prozesse heisst auch **Context-Switch**

- örtliche Management der Prozesse (Speichermanagement im Betriebssystem)

Wie stehen die Prozesse im Hauptspeicher (RAM). Die Problematik ist die, dass der RAM in der Regel zu klein, um alle Prozesse vollständig zu laden. Das Betriebssystem muss dann ggf. Teile des Prozesses während der Laufzeit in den RAM nachladen.

Multitasking System und Echtzeit

Echtzeit im Ingenieur-Sinn bedeutet: Man kann genaue Zeitangaben machen zu den Prozessen (z.B. man kann auf die Micro-Sekunde genau sagen, wie lange eine Aufgabe benötigt)

Multitasking Betriebssysteme (z.B. LINUX bzw. Windows) können diese Form der **Echtzeit im Ingenieur-Sinn** nicht garantieren.
Der Grund dafür ist das Hin- und Her Schalten der Prozesse, d.h. man kann keine genauen Zeitaussagen in Microsekunden für das Scheduling und Speicherverwaltung machen.

Es gibt aber ein LINUX RT (PREEMPT_RT), RT steht für Real Time, d.h. Echtzeit, welches einen anderen Kernel besitzt,
dieser ist aber für den normalen Gebrauch eines PCs oder Servers nicht geeignet.

Prozesse in Windows

- Task Manager

Prozesse in LINUX

- Besitzen Vater-Kind Beziehung
- lassen sich als Baum darstellen (pstree)

auch mit owner Informationen (pstree -u)

Gesamt-Informationen der Prozesse: top

Beenden von Prozessen: das kill-Kommando

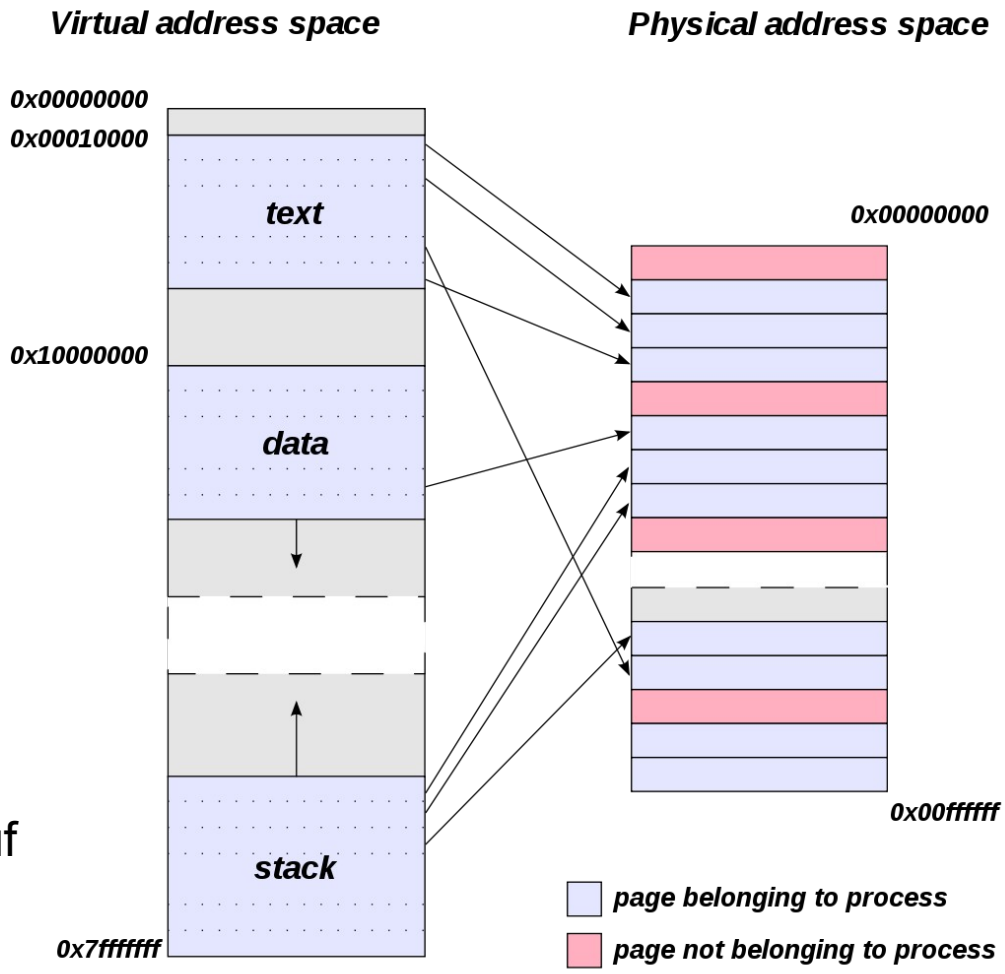
Man muss sich vorstellen: Der Windows Task Manager oder das LINUX top Kommando ist eine Ansicht der Prozesstabelle im Kernel.

Virtueller Addressraum eines Prozesses

Grundlage eines Prozesses ist natürlich sein zugehöriges C-Programm

Im C-Programm stehen **Variablen**, **Funktionen**, …

Wenn das C-Programm geladen wird, d.h. ein Prozess
entsteht, müssen natürlich die Variablen, Funktionen usw.
sich im **RAM** befinden, sodass die CPU darauf zugreifen kann



Virtueller Addressraum eines Prozesses

Welcher Platz im RAM aber gerade frei ist, kann das C-Programm, bzw. der Compiler nicht wissen. Vielleicht wird das Programm später auf einem ganz anderen Rechner gestartet.
Deswegen nutzt der Compiler den gesamten möglichen Adressbereich, um die Variablen, Funktionen, etc. des Programms zu plazieren.
Bei einem 32bit Betriebssystem sind das ca. 4 Mrd. Adressen
Dieser Adressbereich ist der **Virtuelle Addressraum** eines Prozesses.
Virtuell deshalb, weil es ihn in Wirklichkeit gar nicht gibt.

Aus der Sicht des Prozesses hat der Prozess bei einem 32bit Betriebssystem einen Hauptspeicher von 4 GByte, auch wenn der Rechner in Wirklichkeit wesentlich weniger RAM besitzt.

Physikalischer Hauptspeicher (Memory bzw. RAM)

Abstraktion

Ein Prozess kann den gesamten möglichen Speicherbereich belegen (virtueller Speicher)

Aus der Sicht des Prozesses bearbeitet die CPU nur diesen Prozess

Speicherverwaltung

Der Virtuelle Adressraum eines Prozesses wird vom Betriebssystem in den physikalischen Hauptspreicher abgebildet. Dies ist eine komplexe Abbildung, da der virtuelle Speicher wesentlich größer ist als der reale Speicher (RAM). Nach welchen Algorithmen diese Abbildung geschieht ist Gegenstand des Kapitels Speicherverwaltung.

Die Lösung wird sein, dass nur Teile des Virtuellen Speichers sich im RAM befinden

Diese Abbildung muss so erfolgen, dass das Multitasking, d.h. die quasi gleichzeitige Ausführung von Programmen möglichst optimal unterstützt wird

Physikalischer Hauptspeicher (Memory bzw. RAM)

IPC (Inter-Process-Communication)

Ein Prozess kann nicht auf den RAM eines anderen Prozesses zugreifen. Das ist vom Betriebssystem verboten. Das wäre auch eine große Sicherheitslücke.
Deswegen benötigt man eine andere Möglichkeit der Kommunikation.

Da ein Prozess nur auf seinen eigenen virtuellen Speicher direkt zugreifen kann, ist es nur möglich über **Systemaufrufe** mit anderen Prozessen zu kommunizieren. Ein Prozess kann nicht auf den virtuellen Speicher oder den RAM eines anderen Prozesses zugreifen. Die Kommunikation zwischen unterschiedlichen Prozessen ist Gegenstand des Kapitels IPC Inter-Process-Communication