UNIVERSITÄT BERN

Übungsstunde

6. März 2018

Mikael Gasparyan
Institut für Informatik
Universität Bern

Vorlesung Betriebssysteme, FS 2018

Heute

- Infos Kommunikationswege
- Rückblick Übungen Kapitel 01+02
- Ausblick Übungen Kapitel 03
- Fragen?

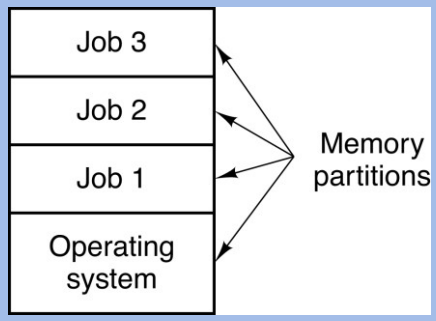
Infos Kommunikationswege

UNIVERSITÄT BERN

- Fragen zur Korrektur der praktischen Übungen
  - Nathalie Froidevaux
  - nathalie.froidevaux@students.unibe.ch
- Fragen zur Korrektur der Theorieübungen
  - Jakob Schärer
  - jakob.schaerer@students.unibe.ch
- Fragen während des Lösens
  - Bitte im Forum

Rückblick Übungen Kapitel 01+02

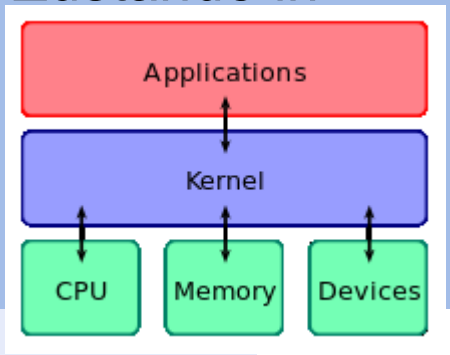
- Welches ist der Unterschied zwischen Timesharing- und Multiprogramming-Systemen? Welche sind als Teilmenge der anderen zu verstehen und wieso? Benutze die Definitionen aus Tanenbaum.
- Tanenbaum: S. 11 / 12
  - *This desire for quick response time paved the way for **timesharing**, a variant of multiprogramming, in which each user has an online terminal. In a timesharing system, if 20 users are logged in and 17 of them are thinking or talking or drinking coffee, the CPU can be allocated in turn to the three jobs that want service.*



Rückblick Übungen Kapitel 01+02

- Benötigt man mehrere Prozessoren, um Multiprogramming implementieren zu können? Begründen sie Ihre Antwort.
  - Nein
  - Scheduling von parallelen Prozessen
  - Aufteilung der CPU-Zeit auf verschiedene Programme

- Was ist der Unterschied zwischen Kernel- und User-Mode? Erkläre, wieso das Definieren dieser zwei Zustände in Betriebssystemen sinnvoll ist.



Rückblick Übungen Kapitel 01+02

Was macht die trap-Instruktion? Erkläre ihren Nutzen. (1)

- Tanenbaum, S. 22:
- *To obtain services from the operating system, a user program must make a **system call**, which traps into the kernel and invokes the operating system. The TRAP instruction switches from user mode to kernel mode and starts the operating system. When the work has been completed, control is returned to the user program at the instruction following the system call.*

Rückblick Übungen Kapitel 01+02

UNIVERSITÄT BERN

- Beschreiben Sie kurz die zwei Konzepte Top-Down und Bottom-Up: Warum werden sie so genannt? Welche von den beiden Sichten ist transparent?

- Transparent
  - Ein System/Modul ist transparent wenn man es als Blackbox betrachten kann
- Top-Down: man schaut von einer abstrakten Ebene auf das System (Systemdetails sind nicht relevant).
  - Transparent
- Bottom-Up: das System ist eine Sammlung von Komponenten, die zusammen funktionieren müssen.

Rückblick Übungen Kapitel 01+02

UNIVERSITÄT BERN

- Benutzerprogramme rufen Systemfunktionen meist nicht direkt auf und wählen den "Umweg" über Bibliotheken. Nennen Sie einige (min. 2) Vorteile dieser Vorgehensweise.

- Einfachere Schnittstelle
  - Systemaufruf-Semantik : komplizierte Assemblerbefehle
  - Bibliotheksfunktionen: einfacher zu bedienende Schnittstelle

- Effizienz
  - Systemaufrufe: aufwendige Kontextwechsel
  - Bibliotheksfunktionen: weniger Aufrufe, durch geeignetes Caching (z.B. bei Dateizugriffen).

Rückblick Übungen Kapitel 01+02

- Was sind Vor- und Nachteile von Mikrokernen?
  - Vorteile
    - Erweiterbarkeit
    - Modularität
    - Prozessisolation
  - Nachteil
    - Zusätzliche Kosten für die Kommunikation zwischen einer Anwendung und dem Dienst

Ausblick Übungen Kapitel 03

UNIVERSITÄT BERN

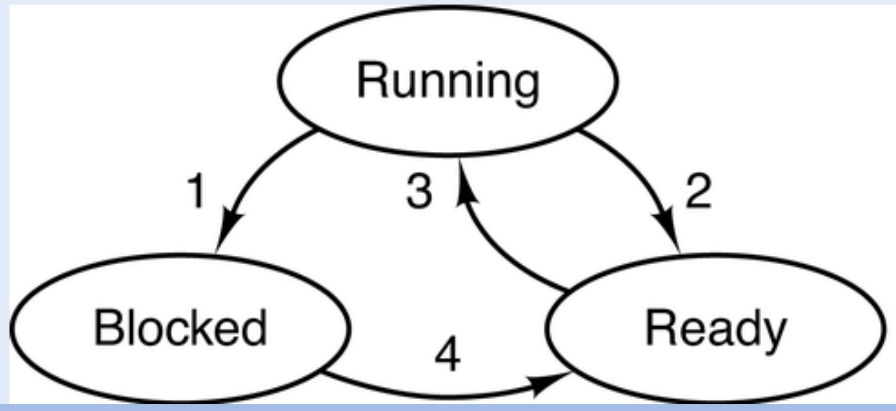
- Abgabetermin: Sonntag 11. März, **23h59**

Ausblick Übungen Kapitel 03

- Was ist die Motivation für Threads, wo doch schon Prozesse die Funktionalität der Aufgabenteilung bieten? Gib mindestens 2 Gründe für die Notwendigkeit von Threads.

- Warum ist ein Threadwechsel schneller als eine Prozessumschaltung?

Ausblick Übungen Kapitel 03



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

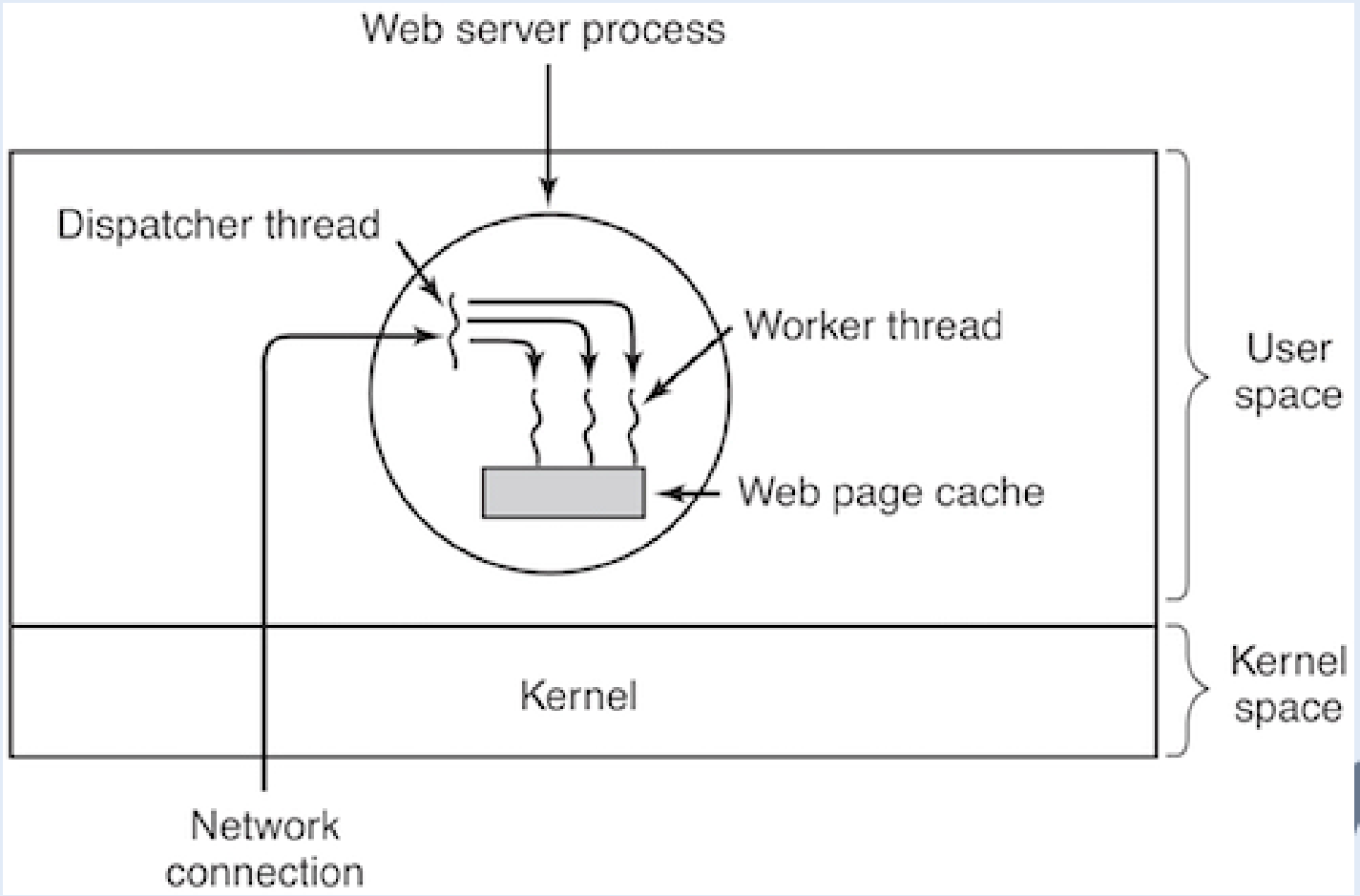
- Auf dem oben angezeigten Bild sind drei Prozesszustände angezeigt. Eigentlich sollten also 6 Pfeile, welche je einen Übergang eines Zustands in einen anderen definieren, gezogen werden können. Welcher der nachfolgenden Pfeile ist jedoch **unmöglich**?
  - Blocked -> Running
  - Ready -> Blocked

Ausblick Übungen Kapitel 03

In UNIX müssen neue Prozesse mittels des ***fork***-Systemaufrufs erzeugt werden. Was passiert beim Aufruf von ***fork***? Wähle die am meisten korrekte Antwort.

- ○ fork erzeugt einen neuen Prozess mit einer neuen PID
- ○ fork erzeugt eine exakte Kopie des aufrufenden Prozesses, mit neuer Prozessnummer, und passt dann den Speicherbereich oder führt allenfalls ein neues Programm aus.
- ○ fork erzeugt eine exakte Kopie des aufrufenden Prozesses, mit neuer Prozessnummer.

Ausblick Übungen Kapitel 03



Obige Grafik zeigt einen Web Server Prozess mit mehreren Threads. Wenn die einzige Möglichkeit, um eine Datei zu lesen, der blockierende ***read*** Systemaufruf ist es geeigneter User- oder Kernel-Threads zu verwenden? Wieso?

Ausblick Übungen Kapitel 03

- Wieso sollte ein Thread freiwillig mittels Aufruf ***thread_yield*** die CPU freigeben? Schliesslich kann es sein, dass es kein periodisches Clock-Interrupt gibt und der Thread nie die CPU zurückerhalten wird.

- Webserver-Prozesse (u.a. Apache) verwenden häufig eine Technik namens "preforking", um Anfragen so schnell als möglich bearbeiten zu können.
  Worin besteht diese Technik?

Fragen?

> **Schöne Woche! ☺**