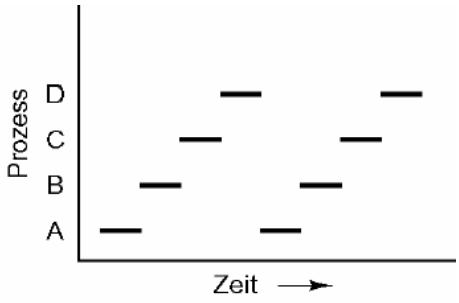
Informationssysteme
Semesterwoche 7

A) Studienelement Betriebssysteme: Fragen

1. Wie ist es möglich, dass auf einer einzigen CPU mehrere Prozesse parallel ablaufen?
→ Die einzelnen Prozesse werden quasi-parallelisiert. Das bedeutet, dass ein Ressourceverteiler (sog. Scheduler) den Prozessen in sehr kurzen Zeitabschnitten Rechenzeit zur Verfügung stellt.

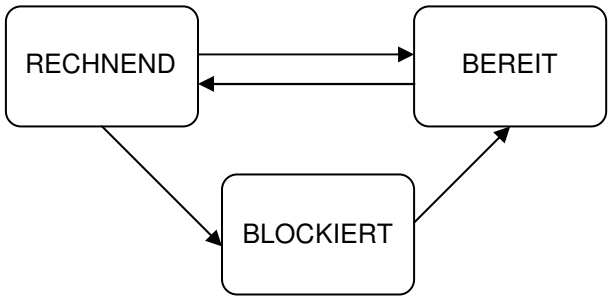


2. Wie werden, technisch gesehen, Prozesse erzeugt?
→ Durch einen Systemaufruf.

Konkret heisst das:
→ Anwendungen werden als Prozesse gestartet (manuell).
→ Dienste starten Prozesse beim Systemstart (automatisch).
→ Prozesse starten andere Prozesse.

3. Nennen Sie die drei Zustände im Leben eines Prozesses
→ rechnend
→ bereit
→ blockiert

4. Zeichnen Sie – möglichst auswendig – die Zustandsübergänge eines Prozesses.



Grundsätzlich kann der Status BLOCKIERT nur dann eintreten, wenn ein Prozess Rechenoperationen durchführt.

5. Kann mehr als ein Prozess gleichzeitig rechenbereit sein?
→ Ja. Grundsätzlich sind alle Prozesse, die aktuell keine Operationen ausführen „BEREIT“ oder eben „RECHENBEREIT“.

6. Kann mehr als ein Prozess gleichzeitig rechnend sein?
→ Nein, auf 1-Prozessor-/1-Core Systemen nicht.

7. Beschreiben Sie die Semantik eines Semaphors.
→ P (reserviert)
→ V (freigegeben)

B) Studienelement Betriebssysteme: Theoriefragen

Bezeichnen Sie bei den folgenden Fragen alle richtigen Antworten mit einem Kreuz.

a) Welche Aufgaben erfüllen Prozesse in einem Programmsystem?
☐ Sie können wie Prozeduren oder Methoden gerufen werden.
☐ Sie stellen über eine Interfaceliste Routinen und Variablen der Umgebung zur Verfügung.
☑ Sie sind die Träger der Aktivitäten im Programmsystem
☑ Sie dienen der dynamischen Strukturierung der zu lösenden Aufgabe.

b) Was versteht man unter dem Begriff MUTEX?
☐ Eine Programmiersprache für Parallele Prozesse.
☑ Den wechselseitigen Ausschluss beim parallelen Zugriff auf gemeinsame Betriebsmittel.
☐ Eine virtuelle Maschine zur Ausführung Paralleler Prozesse.
☐ Eine Maschineninstruktion zur Erzeugung nebenläufiger Prozesse.

c) Welche Aussagen treffen zu:
☐ Damit Prozesse parallel ablaufen können, dürfen sie überhaupt nicht miteinander verkoppelt sein.
☑ Prozesse stellen fast unabhängige, lose gekoppelte Recheneinheiten dar.
☑ Parallele Prozesse können auch auf virtuellen Maschinen laufen.
☑ Parallele Programme lassen sich in kritische Abschnitte und unkritische Abschnitte aufteilen.

d) Was passiert bei einem V(Semaphor) wenn kein Prozess darauf wartet?
☐ Das Signalisieren hat keine Wirkung, es verpufft ins Leere.
☑ Der dem Semaphor zugeordnete Wert wird verändert. Es wird kein Prozess aufgeweckt.
☐ Der dem Semaphor zugeordnete Wert wird verändert. Es wird ein beliebiger schlafender Prozess aufgeweckt.
☐ Es entsteht ein Laufzeitfehler, der vom System abgefangen wird.

e) Auf wie viele verschiedene Varianten können n=5 Prozesse eine einzige Instruktion ausführen?
☐ Auf 1 Art
☐ Auf 5 Arten
☑ Auf 120 Arten
☐ Auf 240 Arten

f) Auf wie viele verschiedene Arten kann ein Prozess ein Programm mit n = 5 Instruktionen ausführen?
☐ Auf n ! (n Fakultät) = 5*4*3*2*1 = 120 Arten
☐ Das kann nicht generell beantwortet werden, es kommt darauf an, in wie viele atomare Instruktionen die n=5 Instruktionen zerlegt werden.
☑ Auf genau eine Art: sequentiell.
☐ Die Anzahl hängt vom Initialwert des Semaphors ab..

C) Studienelement Betriebssysteme: Aufgaben

1 Parallele Programmausführung *
Nennen Sie alle möglichen Schlussresultate der Variablen „x“. Erklären und dokumentieren Sie Ihren Lösungsweg !
"//" bedeutet, dass die drei Prozesse quasi-parallel in Ausführung sind.

```
var s2: semaphore(0); s1: semaphore(1); X:=3;
concurrent_begin
   Prozess1:: P(s2); P(s1); x:=x*2; V(s1)
// Prozess2:: P(s1); x:=x*x; V(s1)
// Prozess3:: P(s1); x:=x+5;V(s2); V(s1)
concurrent_end
```

Ohne Semaphore wären 3*2*1=6 verschiedene Abläufe möglich.
Dank den Semaphoren sieht das ganze einwenig anders aus: Weil S2=0 ist, kann Prozess1 NIE als erster laufen. Prozess2 oder Prozess3 startet also zuerst.
- Wenn Prozess3 durchläuft, schaltet er S2=1, sodass auch Prozess1 die Möglichkeit hat, abzulaufen.
- Prozess 2 kommt vor oder nach Prozess 3.
- Prozess 1 kommt immer nach Prozess3.

Möglichkeit1	**Möglichkeit2**	**Möglichkeit3**	→	**Möglichkeit1**	**Möglichkeit2**	**Möglichkeit3**
Prozess2	Prozess3	Prozess3		9	8	8
Prozess3	Prozess2	Prozess1		14	64	16
Prozess1	Prozess1	Prozess2		28	128	256

2 Addiere Parallel (Fehleranalyse)
Gegeben ist folgender Programmausschnitt, der in einer gemeinsamen Variablen, die Anzahl der BesucherInnen zählt, die durch zwei Drehkreuze (Prozesse) ein Museum betreten:

```
class Counter {
int value=0;
……
void increment() {
int temp = value; // Lesen
Simulate.HWinterrupt(); // Scheduling Erzwingen
value = temp + 1; // Addieren, Zurückspeichern
}
}
……
Counter people = new Counter();
```

Durch jedes der beiden Drehkreuze betreten 20 BesucherInnen das Museum, d.h. zwei Prozesse rufen parallel (gleichzeitig) people.increment();
Was gibt das Resultat? Erklären Sie! Ist das Programm korrekt ? Falls das Programm fehlerhaft ist, geben Sie ein präzises Beispiel (ein Szenario) einer fehlerhaften Programmausführung an und korrigieren Sie das Programm!

→ Es wird nicht sichergestellt, wenn die beiden Drehkreuze gleichzeitig den Counter erhöhen wollen. So kann es vorkommen, dass Counter1 während der Ausführung durch Counter2 unterbrochen wird. Wenn Counter1 wieder von der CPU bedient wird, zählt diesen einen veralteten Wert hoch (denselben Wert, welcher schon von Counter2 erhöht wurde).
→ Die beiden Drehkreuze Counter1 und Counter2 müssen durch MUTEX (Mutual Exclusion, „wechselseitiger Ausschluss“) geschützt werden.
→ Ein Counter ist ein kritischer Prozess und darf während der Ausführung nicht unterbrochen werden.

D) Studienelement Netzwerke

1. Nennen Sie einige Anwendungen für UDP.
→ RTP für Audio-/Video-Streaming, SIP (Voice)
→ WOL (WakeOnLAN)
→ DHCP, DNS, WINS, TFTP

2. Worin besteht für ein Programm der Unterschied beim Aufruf einer lokalen Prozedur im Vergleich zu einer Prozedur auf einem anderen Rechner mittels RPC.
→ Für das Programm ansich besteht kein Unterschied. Lediglich das zeitliche Verhalten kann als Unterschied bezeichnet werden (Latenzzeit).

3. Wofür braucht man das RTP?
→ Das Real-time Transport Protocol wird zur Übertragung von audiovisuellen Daten genutzt. Es ist somit ein typisches Streamin-Protokoll in IP-basierten Netzwerken.
http://de.wikipedia.org/wiki/Real-Time_Transport_Protocol

4. Nennen Sie die vordefinierten Portnummern der Dienste HTTP, SMTP, Telnet, FTP und POP-3.
→ HTTP Port 80
→ SMTP Port 25
→ Telnet Port 23
→ FTP Port 21
→ POP3 Port 110

5. Warum unterstützt TCP kein Broadcast oder Multicast?
→ TCP ist verbindungsorientiert. Das bedeutet, die kommunizierenden Endgeräte bauen vor einer Datenübertragung eine Verbindung auf. Das funktioniert nur für Unicast-Verbindungen. End-zu-End-Verbindungen verlangen zudem ein Handshake beim Verbindungsaufbau.
Der Overhead für eine multicastfähige-TCP Verbindung wäre enorm…

6. Warum braucht man UDP? Reicht es nicht aus, wenn die Benutzerprozesse direkt IP-Pakete senden und empfangen?
→ Nein. UDP gehört zum OSI Layer 4 (Netzwerkschicht) und hat somit die Aufgabe der Zuweisung von Datagrammen zur jeweiligen Anwendung. Diese Nummerierung über "Ports" ist auf Layer 3 (IP) nicht vorgesehen.
→ OSI Layer 4 übernimmt zudem auch noch die Fehlerkorrektur (CRC Checksumme).

7. UDP und TCP benutzen Portnummern, um einen Prozess zu adressieren. Geben Sie zwei Gründe an, warum nicht zur Vereinfachung die bereits vorhandenen Prozesskennungen benutzt werden und stattdessen bei der Definition von UDP und TCP als neue abstrakte Kennung die Portnummern eingeführt wurden.
→ Process IDs varieren ständig und sind zudem abhängig vom Betriebssystem. Die Port-Adressierung muss jedoch Hersteller-unabhängig sein.
→ Wird beispielsweise ein Http-Service neugestartet, hätte dies die Folge, dass ein Client ein anderer Port adressieren müsste.

8. Ein Rechner sendet vollständige TCP Frames mit 65.535 Bytes über einen Kanal mit 1-Gbit/s. Der Kanal hat eine Übertragungsverzögerung von 10 ms in eine Richtung.

a) Welcher maximale Durchsatz kann erreicht werden?

→ Durchsatz = 524280bit / (10ms + 10ms) = 26.214 Mbit/s
f

→ Size s_{max} = 65535 Byte/Frame = 524280 bit/Frame
→ Best Effort e_{max} = 1'000'000'000 bit/s
→ Frameübertragungszeit t_{frame} = 524280 bit/Frame / 1'000'000'000 bit/s = 0.00052428 s/Frame
→ Verzögerungszeit t_{delay} = 10ms = 0.01s

→ Effort $e_{eff} = 1 / (t_{delay} + t_{frame}) = 1 / (0.01s + 0.00052428s) =$ 95 Frames/s

b) * Welche Leitungseffizienz ergibt sich?
→ Effizienz = Durchsatz / Bandbreite = 26.214 Mbit/s / 1000 Mbit/s = 2.62%

→ Best Frame-Effort Fe_{max} = 1907 Frames/s
→ Effort e_{eff} = 95 Frames/s

→ Leistungseffizienz $\eta = e_{eff} / Fe_{max} = 0.05$ → 5% Wirkungsgrad

9. Wie goss kann die Leitungsgeschwindigkeit maximal sein, mit der ein Host TCP-Nutzdaten von 1.500 Bytes und einer maximalen Paketlebensdauer von 120s übertragen kann, ohne dass Folgenummern mehrmals verwendet werden? Berücksichtigen Sie den Overhead von TCP, IP und Ethernet. Gehen Sie davon aus, dass Ethernet-Rahmen fortlaufend gesendet werden. (Hinweis: Der Overhead eines IP-Paketes beträgt 20 Bytes.)
→ In einem LAN ist die Leitungsgeschwindigkeit die Übertragungsrate der Netzwerkkarte.
→ Overhead: TCP: 20Bytes, IP: 20Bytes, Ethernet: 26Bytes.
→ Totaler Framesize = 1566 Bytes
→ Sequenznummer ist eine 32bit Zahl. Mögliche Anzahl Nummern = 2^{32}
→ Maximal 23860 Frames à 1566 Bytes
→ Max.Geschwindigkeit = 299Mbit/s

→ TCP- und IP-Protokoll definieren jeweils einen Header von 20 Bytes. Für die Nutzdaten bleiben in einem TCP/IP-Paket also 1460 (Nutzdaten= 1500 Byte – 20 Byte – 20 Byte) Bytes übrig.
→ 1/0.120 * (12000 bit – 160 – 160) = 97333.33 bit/s