

## Definitionen:

- **Prozeß:** (Vorgang der) Programmausführung
- **Interaktion:** aufeinander bezogenes Handeln
- **Kommunikation:** Bidirektionaler (allg.: multidirektionaler) Austausch von Information
- **Information:** Daten, die in Entscheidungen einfließen
- **Daten:** Angaben, die etwas kennzeichnen



- **'Angaben':** Nennungen, Behauptungen, Schilderungen, Aussagen, Äußerungen  
über gemessene, erhobene, geschätzte, angenommene Zahlenwerte, Zeichen, Funktionen, ...  
(‘ABC’; oder: ‘quadratisch mit der Zeit’)
- **'etwas':** Objekt, Gegenstand, Ereignis, Prozeß, Ablauf, ...  
(z.B.: ‘elektrisches Potential’ oder ‘Gefährdungspotential’)
- **'kennzeichnen':** Interpretation macht aus Angaben Daten  
(“SCHWARZ” als Personennamen oder Haarfarbe)



Karl Valentin

- Mit der Möglichkeit zur Interaktion und Kommunikation stellt sich die Frage nach **Nebenläufigkeit** (*concurrency*), **Parallelisierung** u. **Synchronisation** betrachteter Prozesse.
  - **Nebenläufigkeit:** Eignung von Vorgängen zur voneinander unabhängigen Ausführung
  - **Parallelisierung:** Vorbereitung zur quasi-gleichzeitigen oder (im Mehrprozessor-Betrieb) gleichzeitigen Ausführung
  - **Synchronisation:** Erzwingung des Zeitpunkts und/oder der Reihenfolge der Ausführung von Anweisungen unterschiedlicher Prozesse (meist zur Ermöglichung einer kollisionsfreien parallelen Ausführung)
- Wichtige Grundmuster der Prozeßsynchronisation:
  - Nachrichtenübertragung (*message passing*) – vgl. Callback-Übg
  - Schranke (Hürde, Barriere; *barrier*) – vgl. Double Buffering
  - Wechselseitiger Ausschluß (*mutual exclusion*) – s.u.

Anspruch korrekter Prozeßsynchronisation:

Bei sequentiellen Programmen hauptsächlich zwei Arten logischer Fehler:

- undefinierte Operationen

```
FILE *memo;  
char ch;  
/* ... */  
fprintf(memo, "A"); fflush(memo);  
fseek (memo, -1L, SEEK_CUR);  
fscanf(memo, "%1c", &ch);
```

und

- Nichtterminierung.

```
clock_t tNow, tEnd;  
tEnd=clock()+10*CLOCKS_PER_SEC;  
while(tEnd!=tNow) tNow=clock();
```

Durch Parallelität verändern sich Ressourcen-Verfügbarkeit und Daten laufend; dadurch kann neben Daten-Inkonsistenz ein Fehler besonderer Art entstehen:

**Blockierung:** Warten auf Ereignisse, die nicht eintreten.

...wollen /  
...können

## Anmerkungen:

- Ungesteuerte Nutzung gemeinsamer Ressourcen kann bewirken, daß Prozeß-Ergebnisse von **Zeitpunkt und Dauer der Ressourcen-Zuweisung** abhängen; solche Situationen nennt man **Wettbewerbsbedingungen** (*Race Conditions, RC*).
- Die Verhinderung bzw. Behebung von *RC* ist eine wichtige Aufgabe der Prozeßsynchronisation; dazu werden sog. **kritische Abschnitte** (auch: krit. Bereiche – *Critical Sections / Regions, CS*) eines Prozesses identifiziert, d.h. Programmteile (=Anweisungsfolgen), die **in Konkurrenz mit anderen** Prozessen dieselben exklusiv nutzbaren Ressourcen beanspruchen.
- Die Prozeßsynchronisation konzentriert sich dann auf die Regelungen zum **Eintritt / Verbleib** von Prozessen in ihren kritischen Abschnitten.
- Die Realisierung einer kontrollierten Ressourcen-Zuweisung kann u.U. die Entstehung der **3 Blockierungsarten** begünstigen:

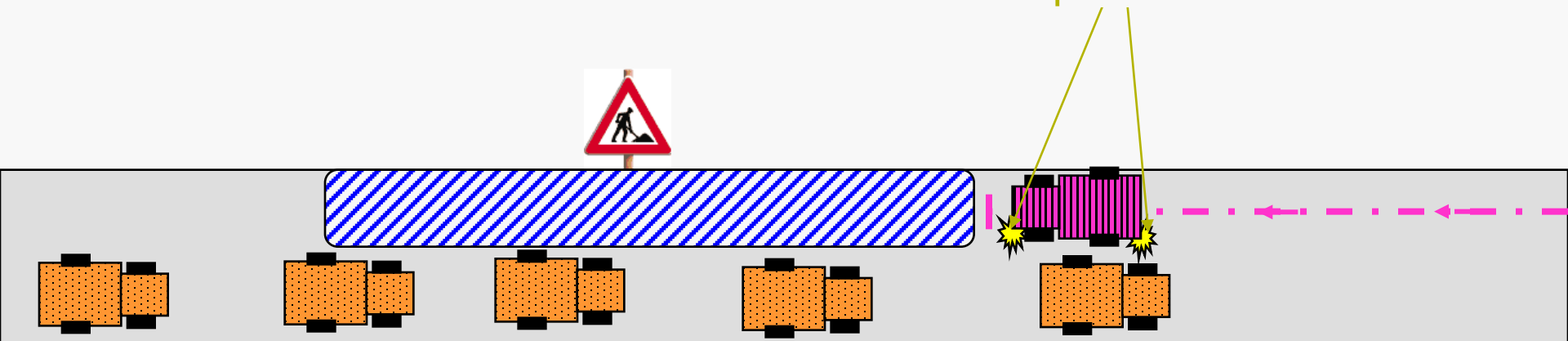
- Erhält ein Prozeß keine Zuteilung wichtiger System-Ressourcen, so sagt man, daß er **verhungert** (*to starve, starvation*).

auch: „Hase-und-Igel-Spiel“:

meist: „durch Prozesse höherer Priorität ausgehungert.“

Charakteristika: oft von endlicher Dauer;

Aktivität beschränkt sich auf **Akquisitionsversuche**



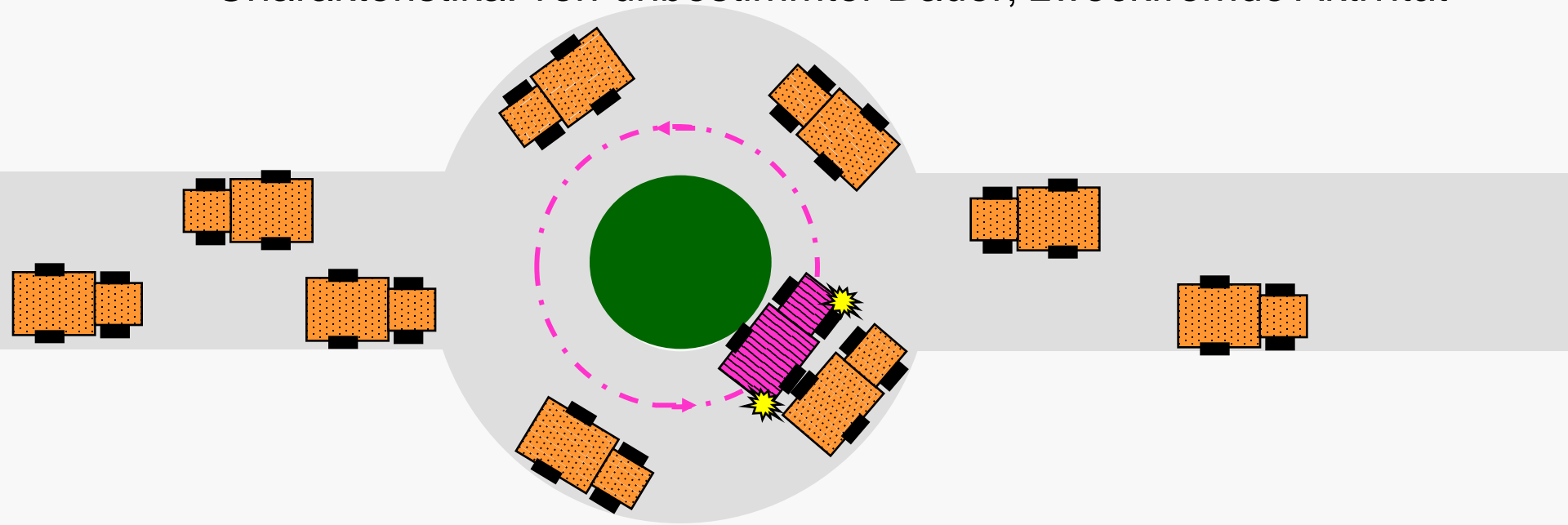
- Eine Situation, in der ein Prozeß beständig um die Nutzung von Systemressourcen gebracht wird, nennt man eine *Blockade* oder einen **Livelock**.

auch: „nicht abbrechende Ausführung“:

meist: Interprozeß-Kommunikation, lineare Wartebeziehungen

„Prozeß verwaltet nur noch sich selbst.“

Charakteristika: von unbestimmter Dauer; zweckfremde Aktivität



- Eine beständige zyklische Wartebeziehung bezeichnet man als *Verklemmung* oder (einen) **Deadlock**.

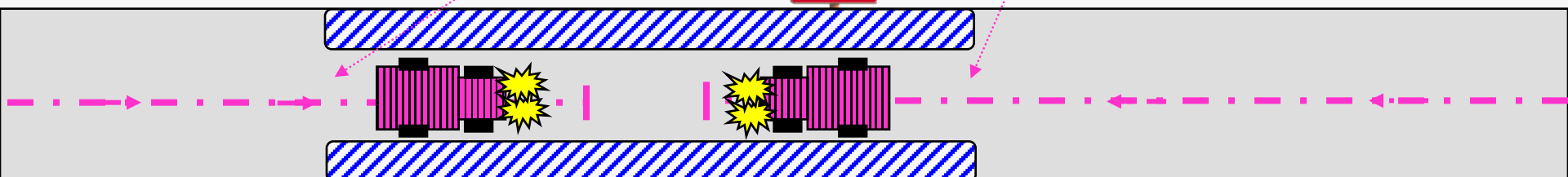
auch: „nicht abbrechende Blockierung“

meist: Einzel-Reservierung gemeinsam benötigter **Ressourcen**

„Prozeß wartet auf Ereignisse, die nicht eintreten können.“

Charakteristika:

1. **Stillstand:** Beteiligte Prozesse können ihr Vorhaben nicht fortsetzen (blockiert).
2. **Permanenz:** Keine spontane Auflösung; zusätzliche Maßnahmen notwendig.
3. **Zufälligkeit:** Auftreten bei manchen Ausführungen, bei anderen dagegen nicht.



Gleichzeitiges Auftreten von vier Bedingungen ist notwendig und hinreichend für einen Deadlock:

- 1. Exklusiv nutzbare Ressourcen** (*serially reusable*)  
Betriebsmittel, deren Nutzung zu einem konkreten Zeitpunkt höchstens einem Prozeß vorbehalten ist und nicht unterbrochen werden darf (Wechselseitiger Ausschluß – engl. *mutual exclusion*).
- 2. Nichtentziehbarkeit** (*non-preemption*)  
Einmal zugewiesene Betriebsmittel werden von Prozessen selbständig o. bei Terminierung zurückgegeben u. können nicht entzogen werden.
- 3. Inkrementelle Akquisition**  
Prozesse können Betriebsmittel halten, während sie auf die Freigabe weiterer Betriebsmittel durch andere Prozesse warten.
- 4. Zyklische Wartebeziehung** (*circular wait*)  
Es gibt eine geschlossene Kette von Prozessen, in der jeder Prozeß auf die Freigabe eines Betriebsmittels durch seinen Nachfolger wartet.

statisch

dynamisch



## Bemerkungen zu den vorausgegangenen Folien:

Prof. Dr. A. Christidis • WS 2014/15

- **Interaktion** – einfacher: Tun unter Berücksichtigung u. zum Auslösen fremden Tuns.
- Es gibt evtl. eine allgemein akzeptierte Definition für **Kommunikation** (I.A. Richards GB, 1928); sie basiert auf der gegenseitigen geistigen Beeinflußung und besagt, daß Komm. dort stattfindet, wo Erfahrungen eines Individuums mental weitergegeben werden, so daß andere Individuen ohne diese Erfahrungen so handeln, als hätten sie diese gehabt.
- Die Information-Def. stammt von Marshall C. Yovits, Emeritus von Indiana University, USA
- **Kollision** ist die gleichzeitige Nutzung exklusiv (auch: seriell, s.u.) nutzbarer Ressourcen durch mehr als einen Prozeß.
- **Synchronisation** spielt eine wichtige Rolle auch bei Applikationen mit zeitdiskreten Datensätzen; darunter fallen Telemetrie, Navigation, Grafik, Bild und Ton bei (z.B. synthetisch erzeugten) Zeichentrickfilmen oder Simulationen, medizinische Apparate, Buchungssysteme u.a. Anwendungen, bei denen die Kollisionsfreiheit nicht durch bloße Verdrängung gelöst werden kann, weil es um eine erwünschte Interaktion unter den beteiligten (u. ggf. kollidierenden) Prozessen geht.
- Beim **Livelock** wird ein Prozeß nicht unbedingt um die Zuweisung von Ressourcen, sondern definitiv um deren Nutzung gebracht (die Straße rechts vom Kreisverkehr kann immer wieder frei werden, wenn das Fahrzeug gerade auf der linken Seite fährt).
- „**Zyklisch**“ sind Wartebeziehungen, bei denen die Wartenden auf einander warten. Ist das Warten nicht gegenseitig, so spricht man von „linearen“ Wartebeziehungen.
- System-Eigenschaften, die stets gelten, werden **statisch** genannt; solche, die erst zur Laufzeit auftreten können und daher bei bestimmten Ausführungen erfüllt, bei anderen wiederum nicht erfüllt sind, werden als **dynamische Deadlockbedingungen** bezeichnet.
- E.G.Coffman, M.J.Elphick und A.Shoshani lieferten 1971 erstmalig den theoretischen Beweis für die o.a. vier Bedingungen zur Entstehung eines Deadlocks.

**Deadlock-Freiheit** ist schwer zu gewährleisten und kaum zu beweisen (keine allgemein akzeptierte Prüfmethode)!

## Anti-Deadlock-Strategien:

- **Vorbeugung** (*Prevention*)

Prinzipielle Verhinderung der Erfüllung aller 4 Voraussetzgn:  
präemptive Vergabe, gruppenweise Akquisition, Prioritäten

- **Vermeidung** (*Avoidance*)

Registrierung bereits erfüllter Voraussetzungen  
ggf. Nichtgewährung benötigter Ressourcen

- **Erkennung & Auflösung** (*Detection & Recovery*)

„Gewaltsamer“ Eingriff von „autorisierter“ Stelle:  
Ressourcen-Entzug, Prozeß-Beendigung



## **Anti-Deadlock-Maßnahmen:**

Verhinderung zyklischer Wartebeziehung:

- Zentrale Warteschlange für Ressourcen-Zuweisung
- Zufallsabhängige Behandlung beteiligter Prozesse
- Einteilung von Prozeß-Gruppen zur Symmetrie-Brechung
- Zuteilung ungleicher Prioritäten

Verhinderung inkrementeller Akquisition:

- Gruppenweise Ressourcen-Reservierung
- Schutz kritischer Abschnitte

## Drei Forderungen an die Behandlung kritischer Abschnitte:

- Wechselseitiger Ausschluß (*mutual exclusion*):

Zu keinem Zeitpunkt darf sich mehr als **ein Prozeß** (Thread) in seinem kritischen Abschnitt aufhalten.

- Fortschreiten (*progress*):

Der Prozeß, der als **nächster** in seinen krit. Abschnitt eintreten darf, soll unter den bereits **wartenden** ausgewählt werden.

- Begrenztes Warten (*bounded waiting*):

Einem wartenden Prozeß darf nur **begrenzt viele** Male der Eintritt in seinen kritischen Abschnitt verwehrt werden.

## Lösungsansätze:

- Software-Lösungen (algorithmisch, z.B. innerhalb d. Sw-Umgebung)
- Hardware-Lösungen (Anwendung spezieller Hardware-Funktionen)
- Betriebssystem-unterstützte Lösungen (spezielle BS-Funktionen)

## Anmerkungen:

### ● **Software-Lösungen...**

... implizieren **aktives Warten** und daher Effizienz-Probleme

⇒ ... sind bei Ermangelung effizienterer Lösungen einzusetzen

### ● **Hardware-Lösungen...**

... als Interrupt-Sperrung bei Ein-Prozessor-Systemen

... vereiteln auch den **Wechsel** zu anderen Tasks (Multitasking)

... sind fatal bei **Programmabstürzen** u. Mißbrauch durch Programmierer

... als Interrupt-Sperrung bei Mehrprozessor-Systemen

... sind praktisch **wirkungslos**, wenn auf nur einen Prozessor beschränkt

... mit speziellen Hardware-Funktionen für Ein-Prozessor-Systeme

... ermöglichen wechselstg. Ausschl. mit **atomaren read-modify-write-Fkt.**

... verursachen bei anderen Prozessen **aktives Warten** (Effizienz-Probl.)

... lassen die Maßnahmen für Fortschreiten & begrenztes Warten **offen**

... mit speziellen Hardware-Funktionen für Mehrprozessor-Systeme

... können zudem wg. Alleinverfügung über d. **Bus** d. System **blockieren**

⇒ ... sind nützlich vor allem innerhalb des Betriebssystems