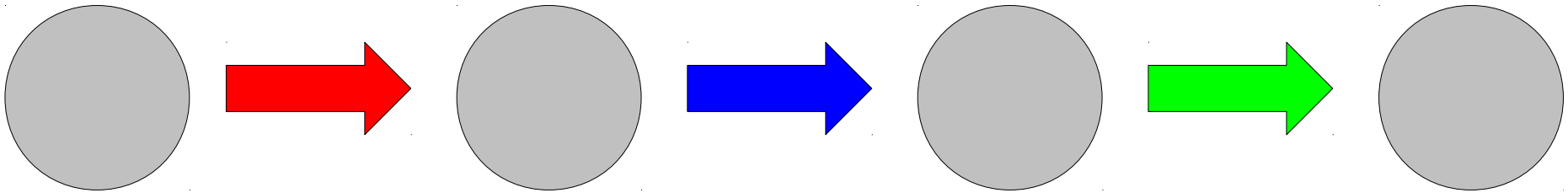
Bedingungsvariablen

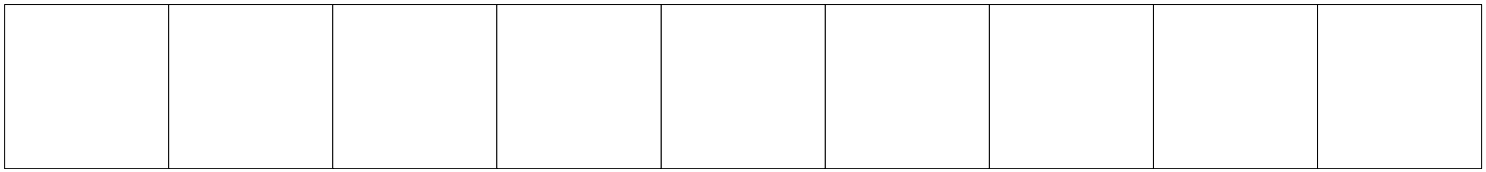
Für das Proseminar:
Konzepte von Betriebssystemkomponenten

Sebastian Rachuj

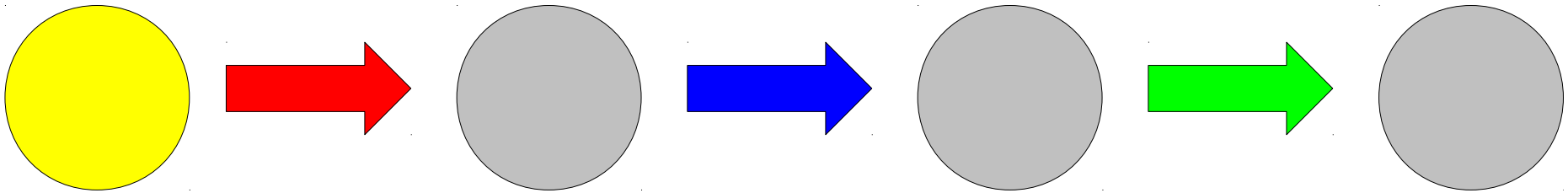
20.06.2013

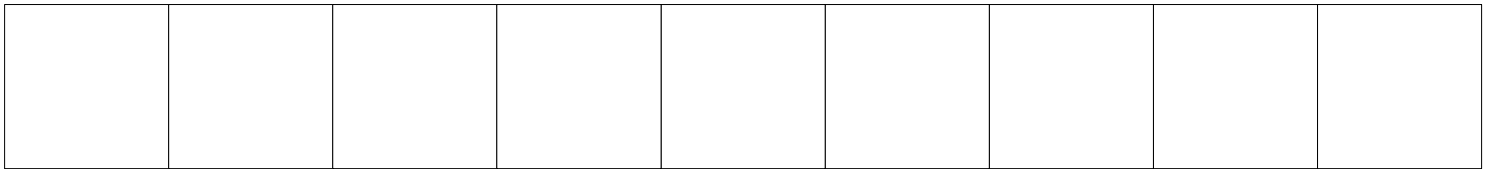
Motivation



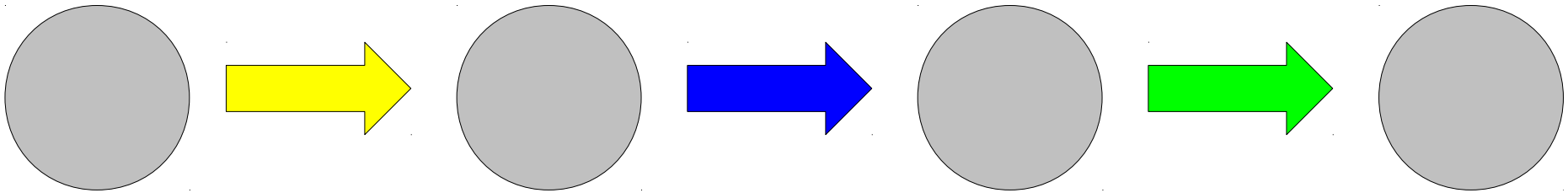


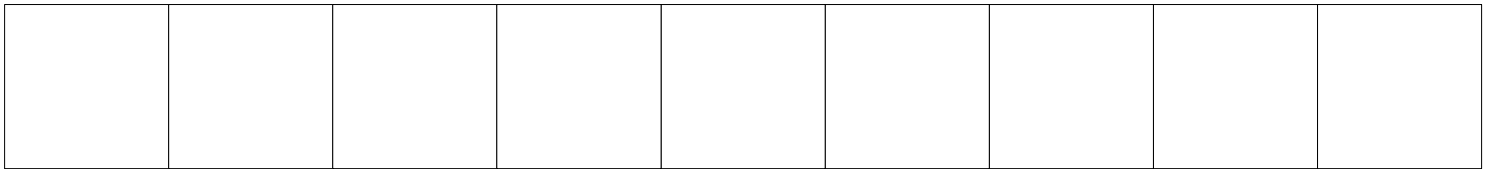
Motivation



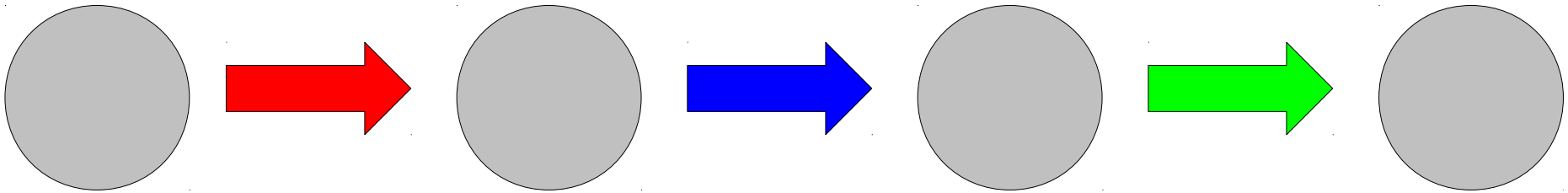


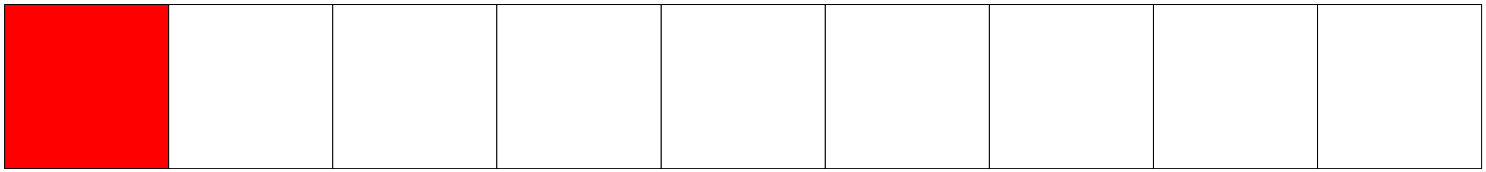
Motivation



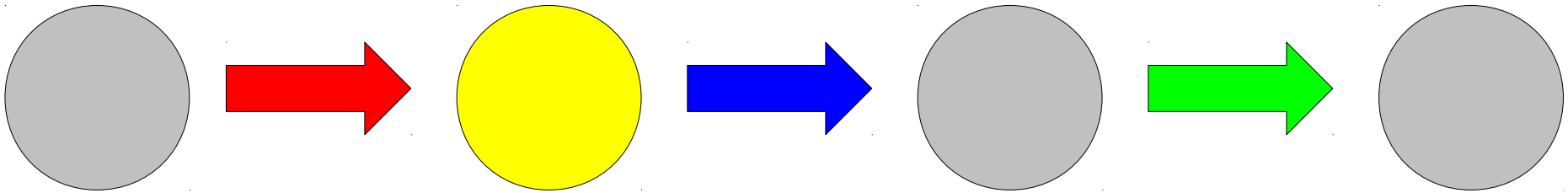


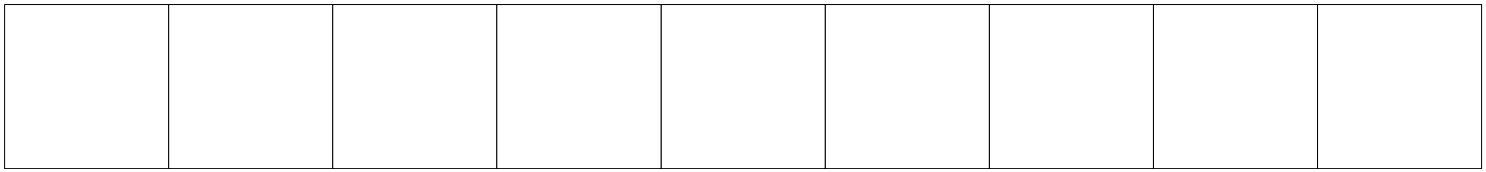
Motivation



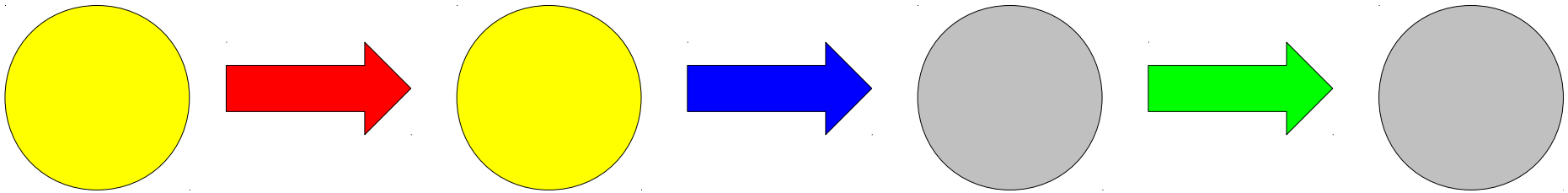


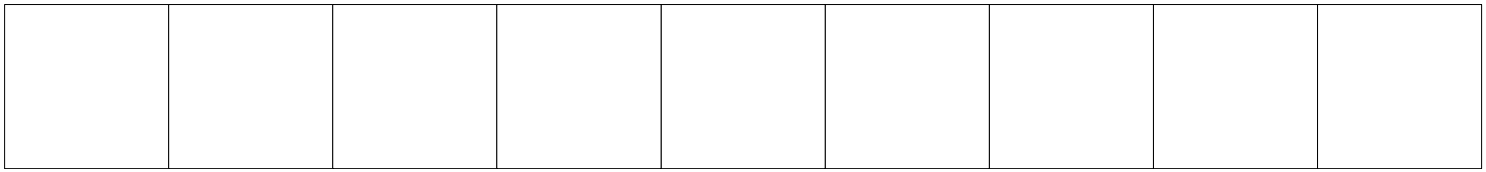
Motivation



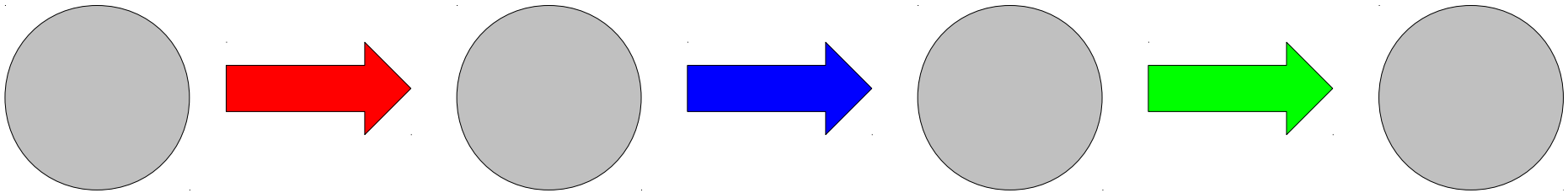


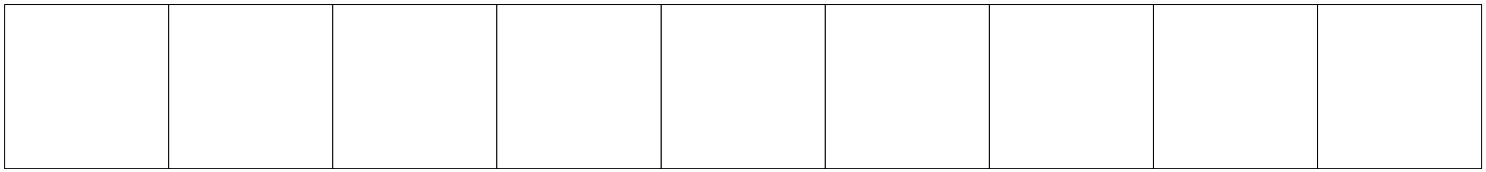
Motivation



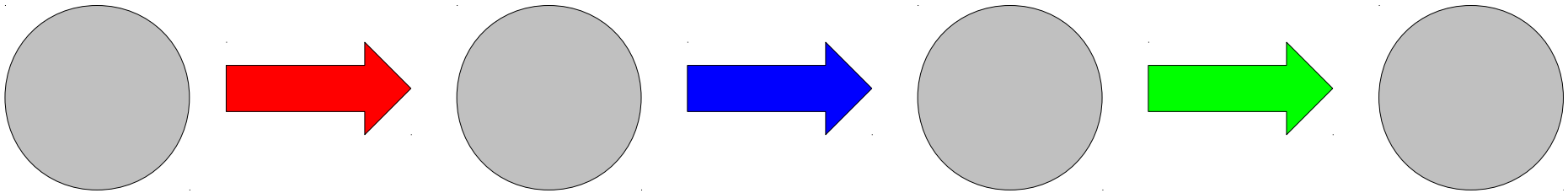


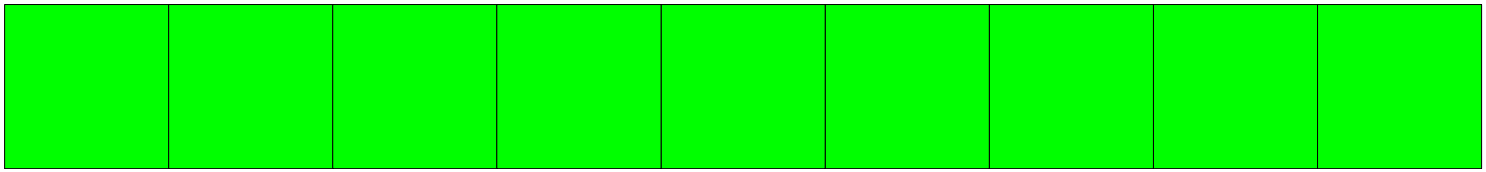
Motivation



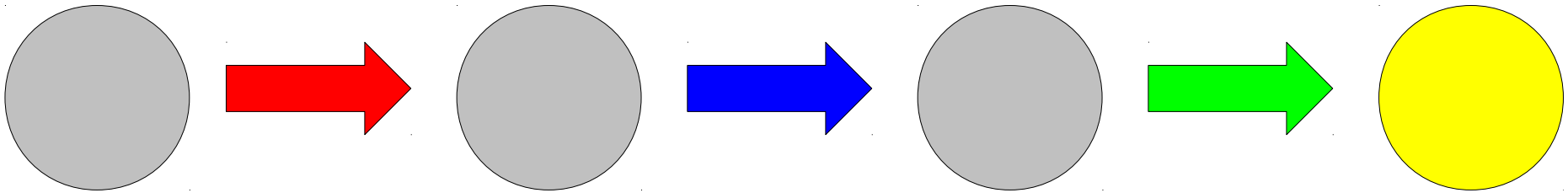


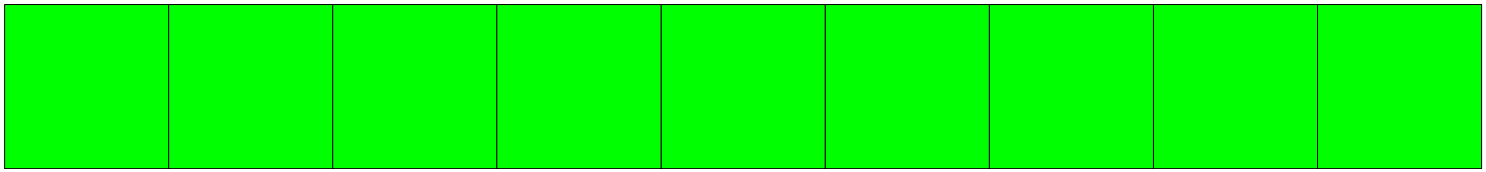
Motivation



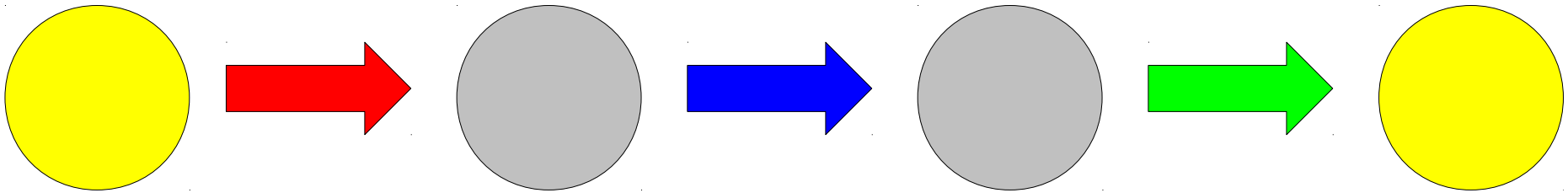


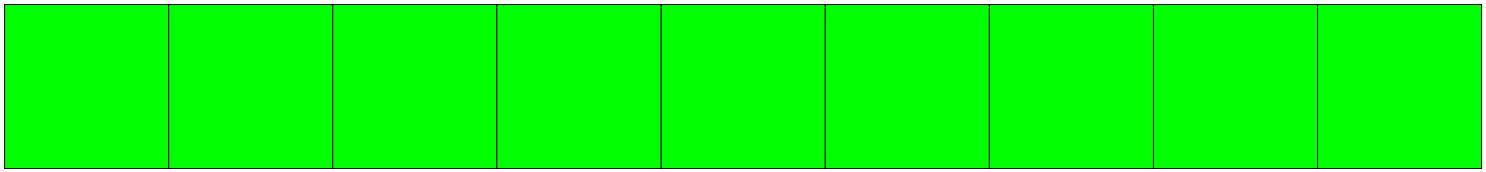
Motivation



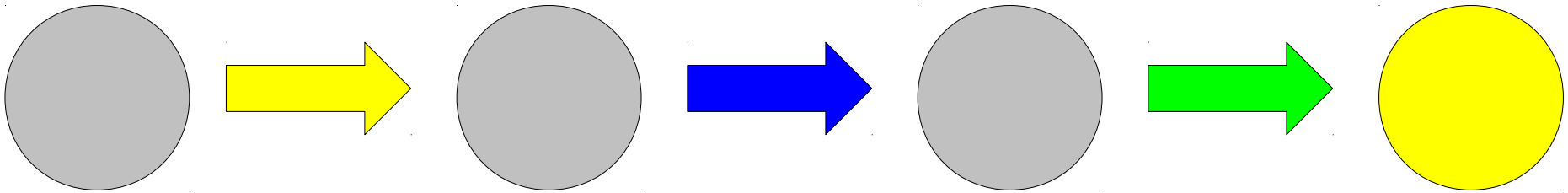


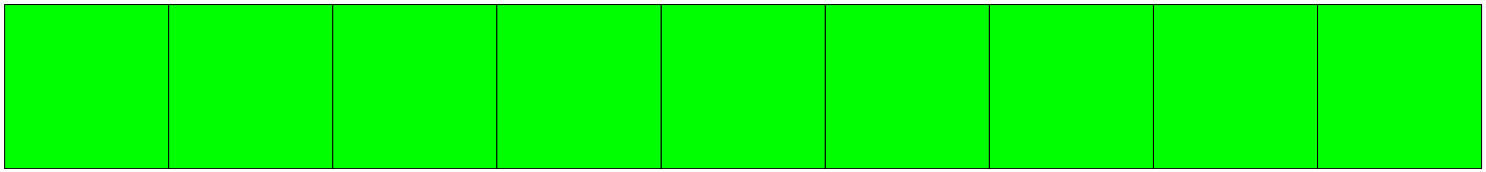
Motivation



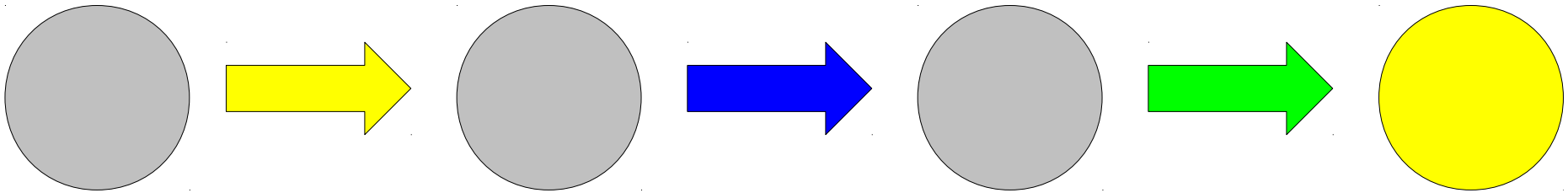


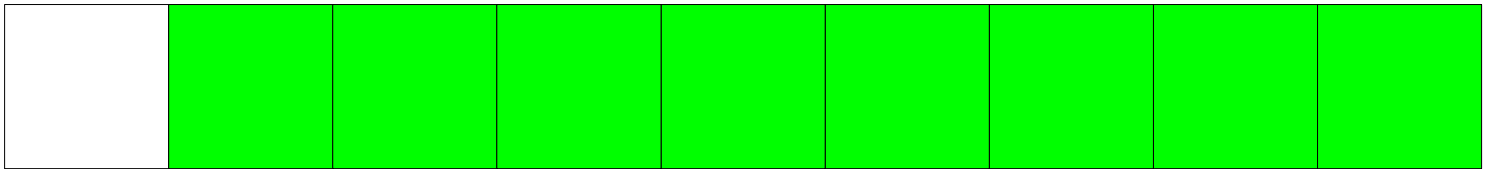
Motivation



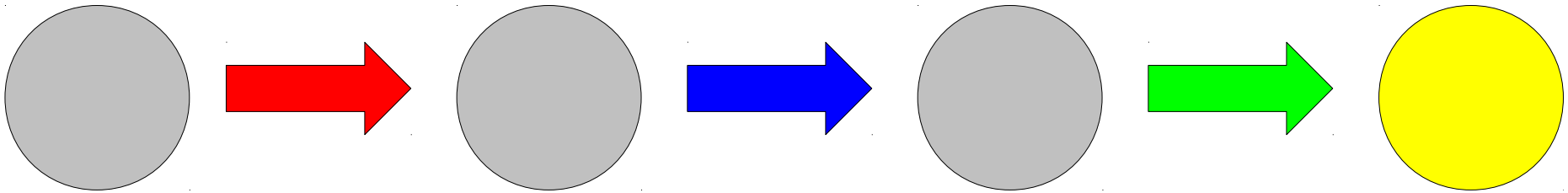


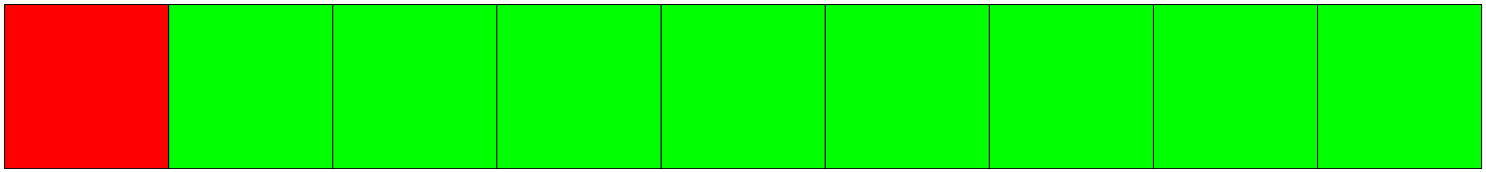
Motivation



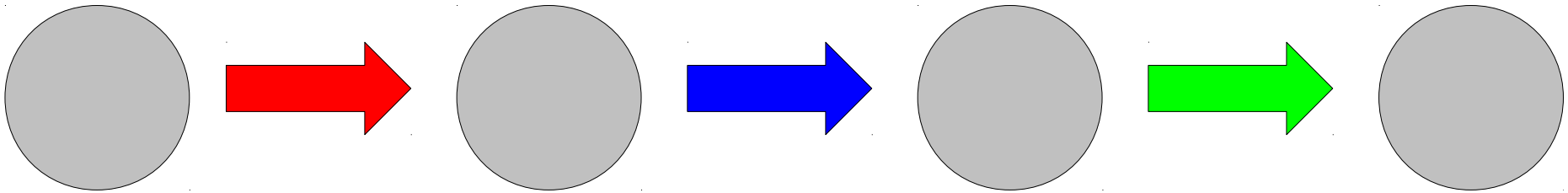


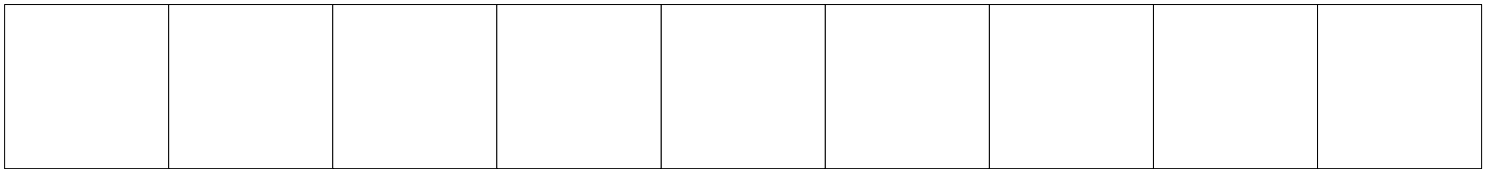
Motivation





Motivation





Inhalt

Sequentielle Prozesse

Sequentielle Prozesse

```
void produce(void* x[], size_t s) {
    for (size_t i = 0; i < s; ++i) {
        step1(x[i]);
        step2(x[i]);
        step3(x[i]);
    }
}
```

Sequentielle Prozesse

```
void produce(void* x[], size_t s) {
    for (size_t i = 0; i < s; ++i) {
        step1(x[i]);
        step2(x[i]);
        step3(x[i]);
    }
}
```

- Ergebnis unabhängig der Ausführungsgeschwindigkeit

Sequentielle Prozesse

```
void produce(void* x[], size_t s) {
    for (size_t i = 0; i < s; ++i) {
        step1(x[i]);
        step2(x[i]);
        step3(x[i]);
    }
}
```

- Ergebnis unabhängig der Ausführungsgeschwindigkeit
- Invarianten möglich

Sequentielle Prozesse

```
void produce(void* x[], size_t s) {
    for (size_t i = 0; i < s; ++i) {
        step1(x[i]);
        step2(x[i]);
        step3(x[i]);
    }
}
```

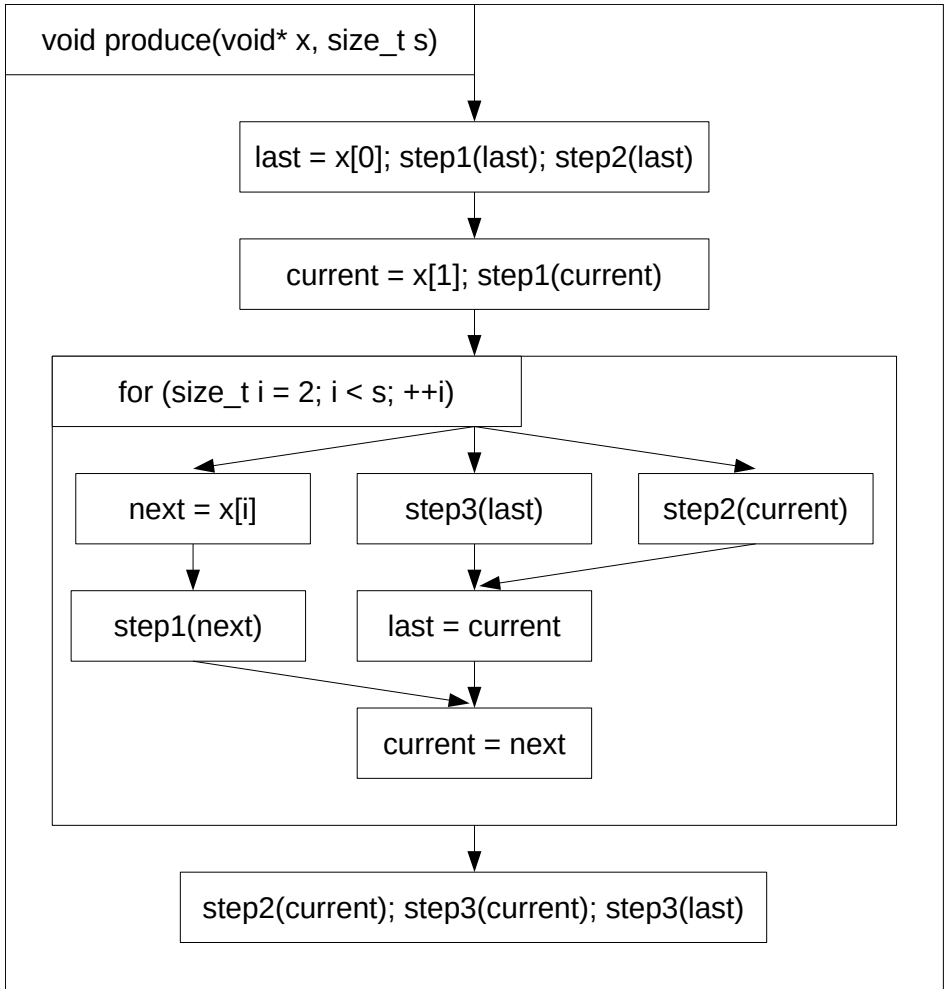
- Ergebnis unabhängig der Ausführungsgeschwindigkeit
- Invarianten möglich
- Gleiche Ausgabedaten bei gleichen Eingabedaten

Parallele Prozesse

Parallele Prozesse

- Strukturierung wünschenswert

Parallele Prozesse



Parallele Prozesse

- Strukturierung wünschenswert

Parallele Prozesse

- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block

Parallele Prozesse

- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block
  - Deterministisch, Invarianten möglich

Parallele Prozesse

- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block
  - Deterministisch, Invarianten möglich
  - Fehlererkennung durch den Compiler

Parallele Prozesse

- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block
  - Deterministisch, Invarianten möglich
  - Fehlererkennung durch den Compiler
    - Kenntnis über Veränderlichkeit von Variablen

Parallele Prozesse

- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block
  - Deterministisch, Invarianten möglich
  - Fehlererkennung durch den Compiler
    - Kenntnis über Veränderlichkeit von Variablen
    - Kenntnis über Ziele von Zeigern

Parallele Prozesse

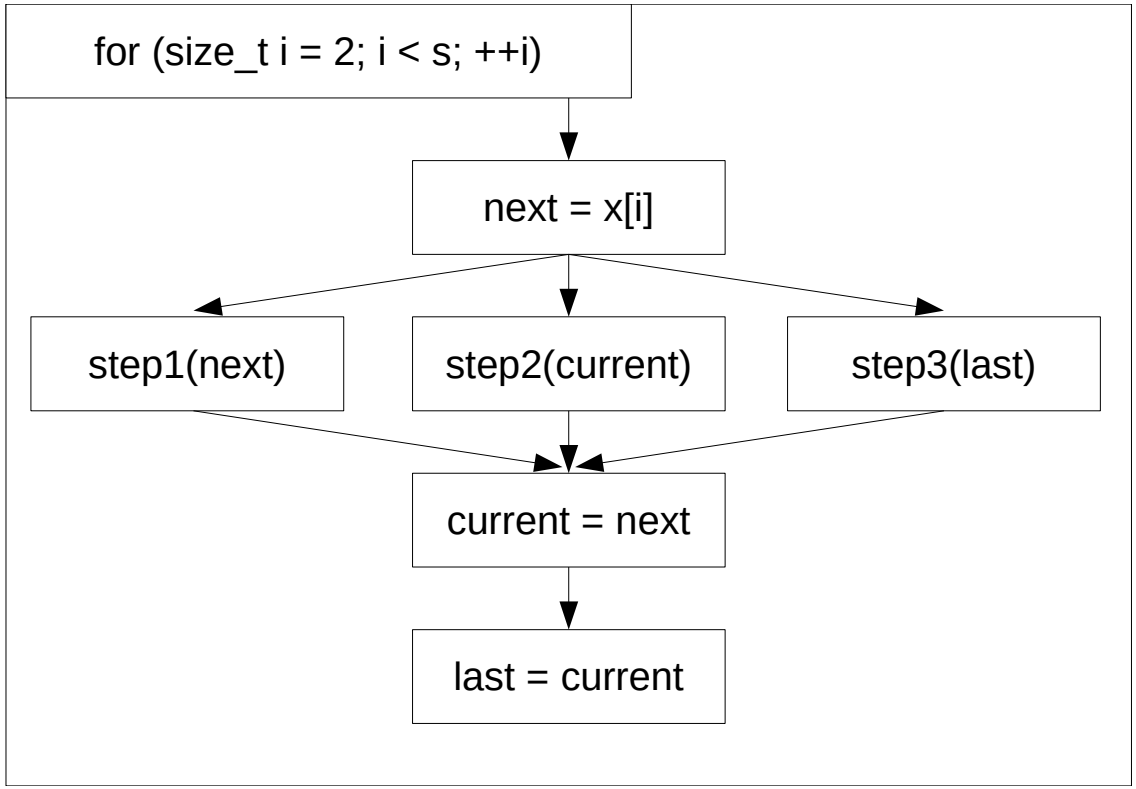
- Strukturierung wünschenswert
- $\Rightarrow$ parallel -Block
  - Deterministisch, Invarianten möglich
  - Fehlererkennung durch den Compiler
    - Kenntnis über Veränderlichkeit von Variablen
    - Kenntnis über Ziele von Zeigern
- Variablenunabhängigkeit

Parallele Prozesse

Parallele Prozesse

```
void produce(void* x[], size_t size) {
    void* next, current = x[0], last = x[1];
    step1(current);
    step1(last);
    step2(last);
    for(size_t i = 2; i < size; ++i) {
        next = x[i];
        parallel {
            step1(next);
            step2(current);
            step3(last);
        }
        last = current;
        current = next;
    }
}
```

Parallele Prozesse



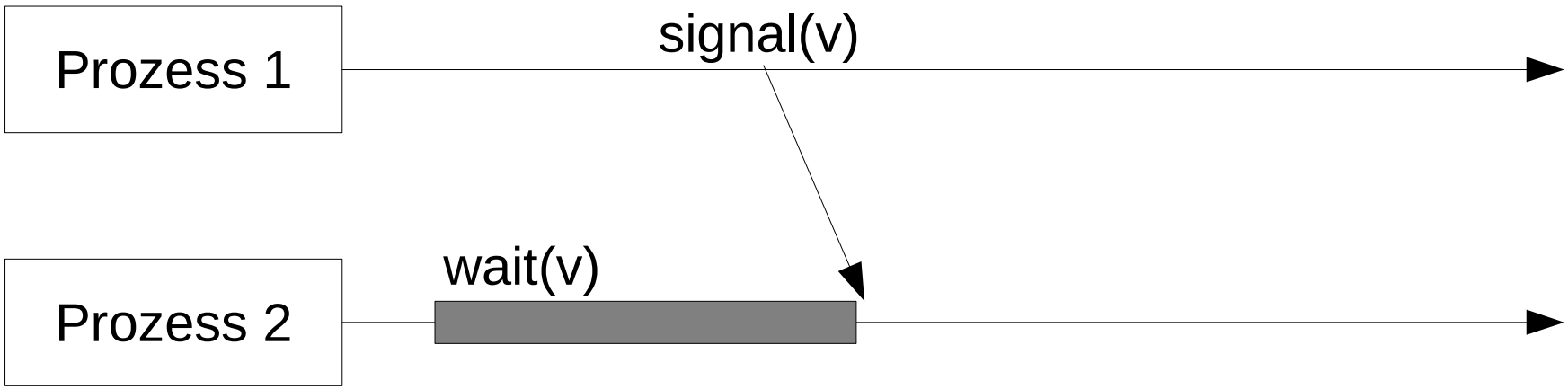
Signale

Signale

- Zulassen von wait und signal auf Variablen

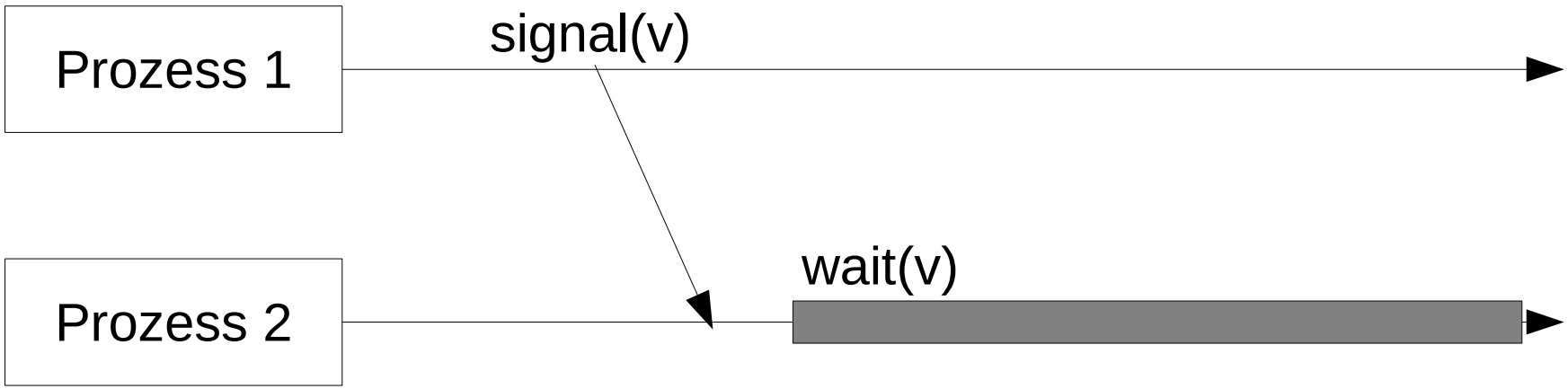
Signale

- Zulassen von wait und signal auf Variablen



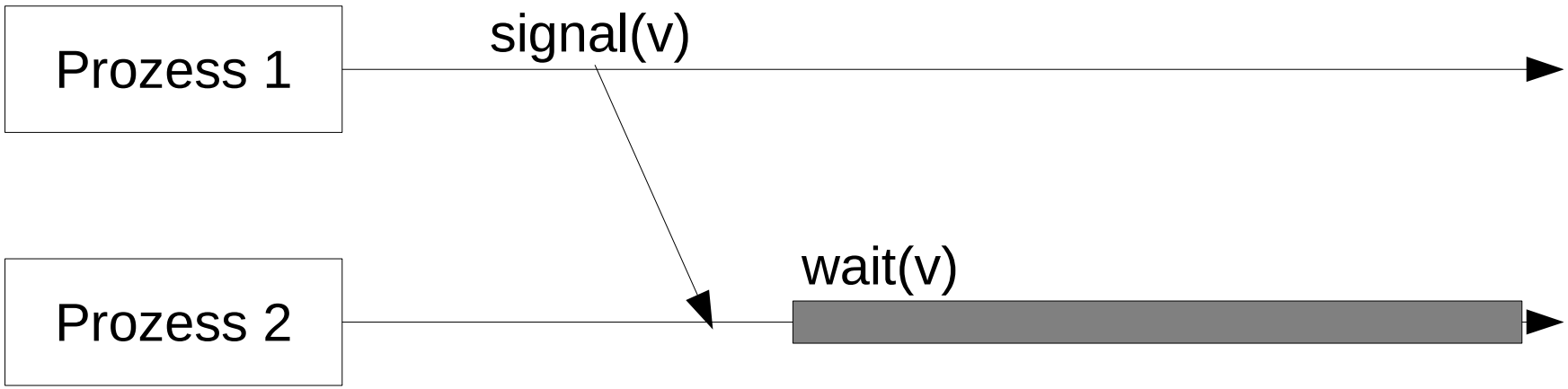
Signale

- Zulassen von wait und signal auf Variablen



Signale

- Zulassen von wait und signal auf Variablen
- Lösungsansatz: Semaphore als Signalvariable



Kritische Abschnitte

Kritische Abschnitte

- Austausch von Daten

Kritische Abschnitte

- Austausch von Daten
- Race Conditions

Kritische Abschnitte

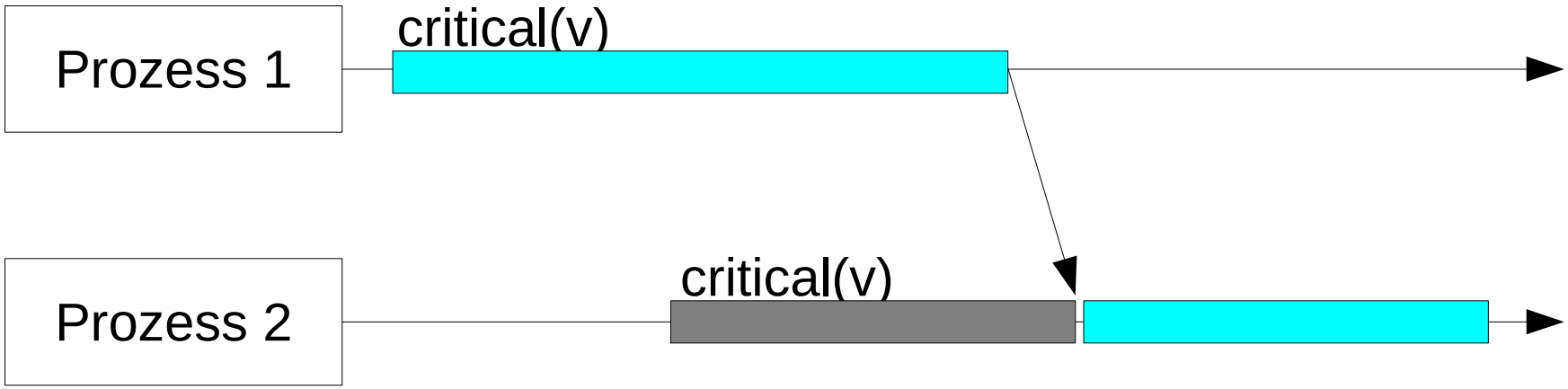
- Austausch von Daten
- Race Conditions
- $\Rightarrow$ Wechselseitiger Ausschluss

Kritische Abschnitte

- Austausch von Daten
- Race Conditions
- $\Rightarrow$ Wechselseitiger Ausschluss
- critical -Block

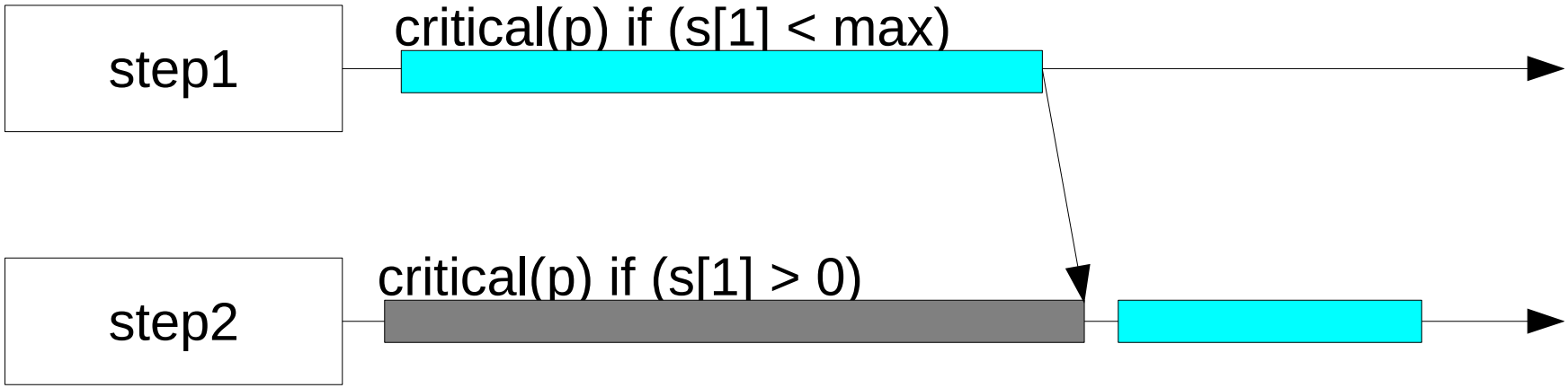
Kritische Abschnitte

- Austausch von Daten
- Race Conditions
- $\Rightarrow$ Wechselseitiger Ausschluss
- critical -Block

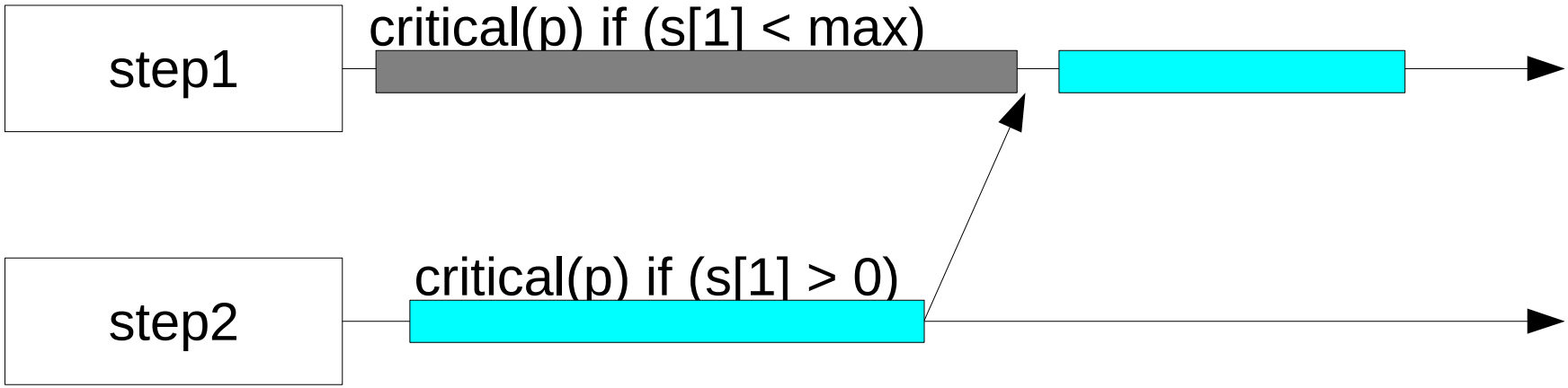


Bedingte kritische Abschnitte

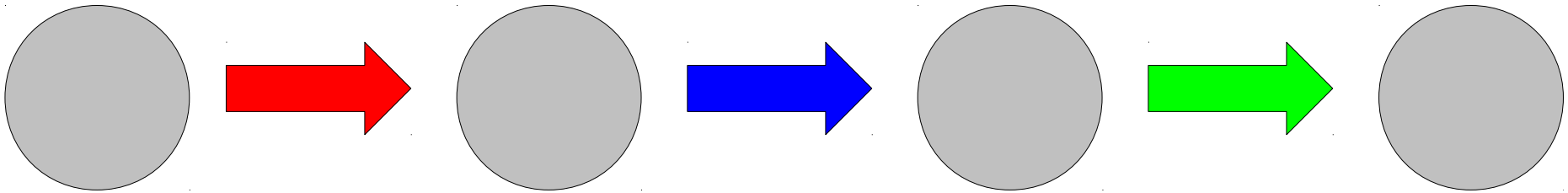
Bedingte kritische Abschnitte

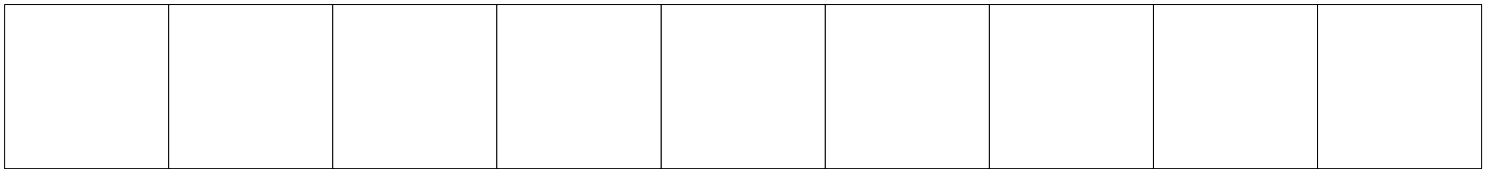


Bedingte kritische Abschnitte



Bedingte kritische Abschnitte





Zusammenfassung

- Strukturierte sequentielle Prozesse
- Strukturierte parallele Prozesse
- Ereignismitteilung mittels Signalen
- Austausch von Daten mit kritischen Abschnitten
- Bedingte kritische Abschnitte (Bedingungsvariablen)

Zusammenfassung

- Strukturierte sequentielle Prozesse
- Strukturierte parallele Prozesse
- Ereignismitteilung mittels Signalen
- Austausch von Daten mit kritischen Abschnitten
- Bedingte kritische Abschnitte (Bedingungsvariablen)

Denkanstöße:

- Gezeigte Sprachkonstrukte in heutigen Sprachen
- Ähnlichkeit des Beispiels zu jbuffer $\Rightarrow$ Wartefreie Synchronisation besser?