Die fork()-Funktion in Unix/Linux

Prof. Dr. Carsten Vogt

Technology
Arts Sciences
TH Köln

Fakultät für
Informations-,
Medien- und
Elektrotechnik

Die fork()-Funktion in Unix/Linux

```
pid_t fork(void);
```

- **Kein Parameter**
- **Erzeugt vollwertigen nebenläufigen Prozess**

$\rightarrow$ **Wie funktioniert das?**

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

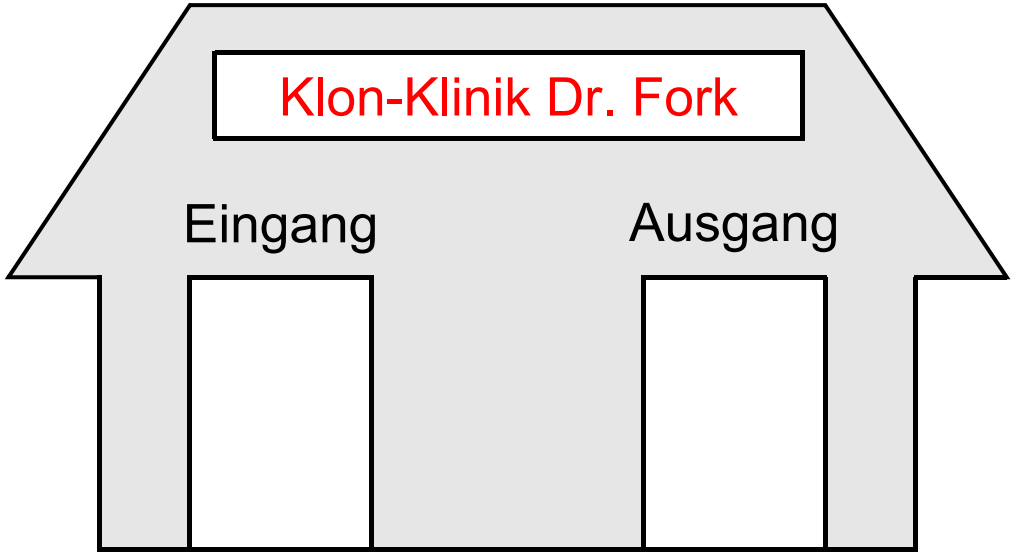
Die fork()-Funktion in Unix/Linux

1.) Analog-Beispiel

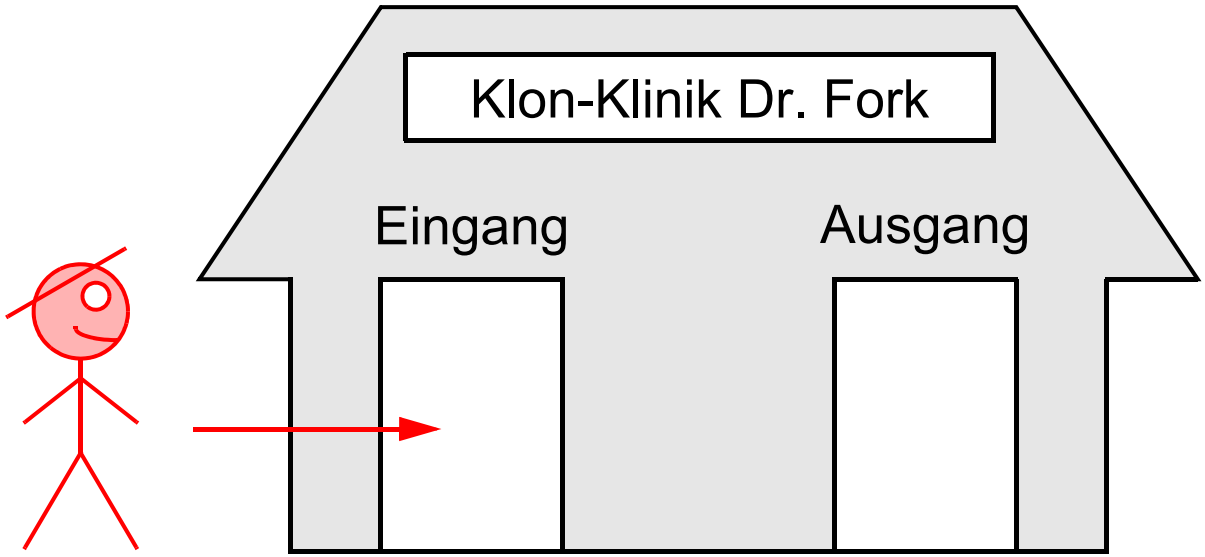
2.) Arbeitsweise

3.) Speicherkonzept

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



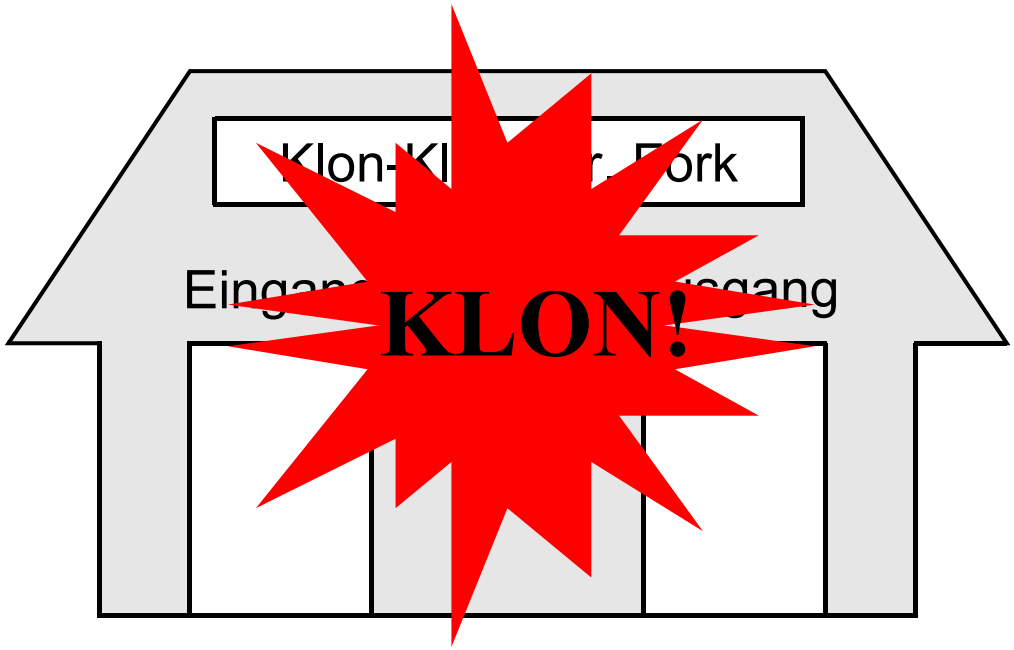
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



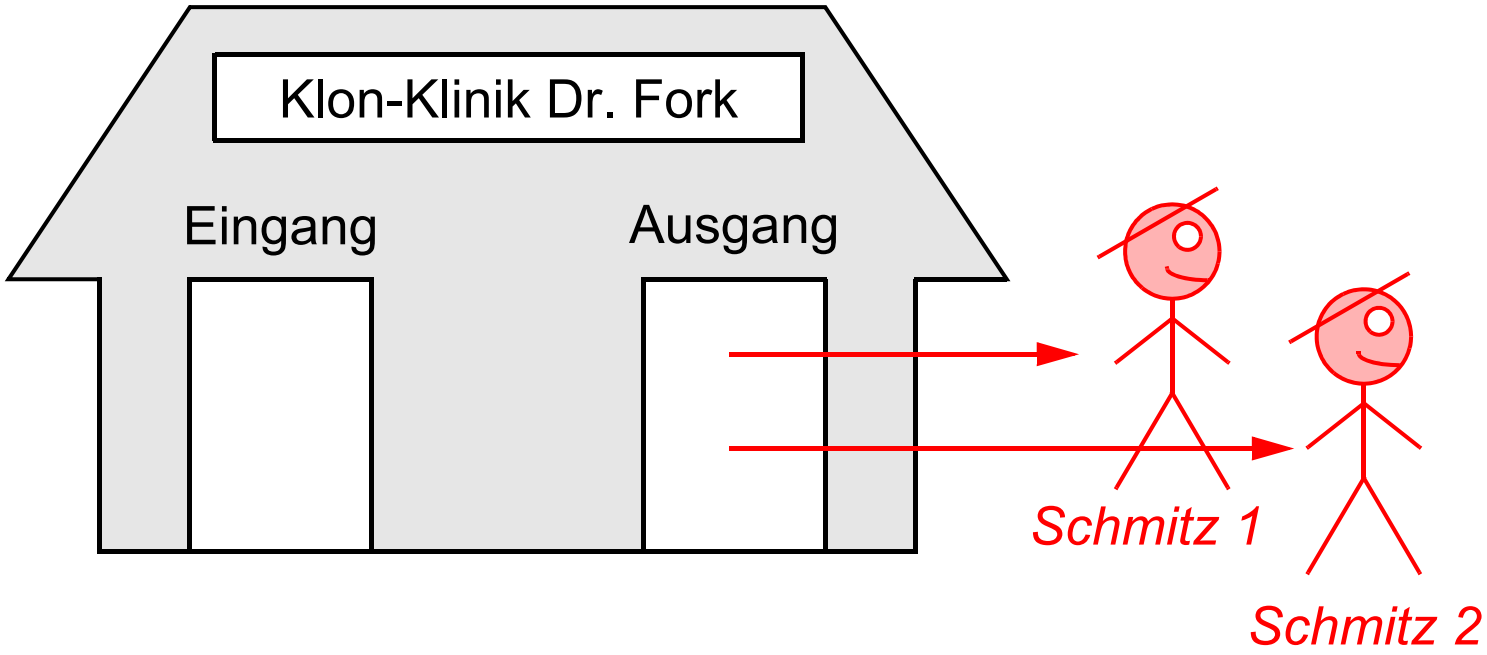
Schmitz 1

Eine Person betritt die Klinik,
um sich klonen zu lassen.

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



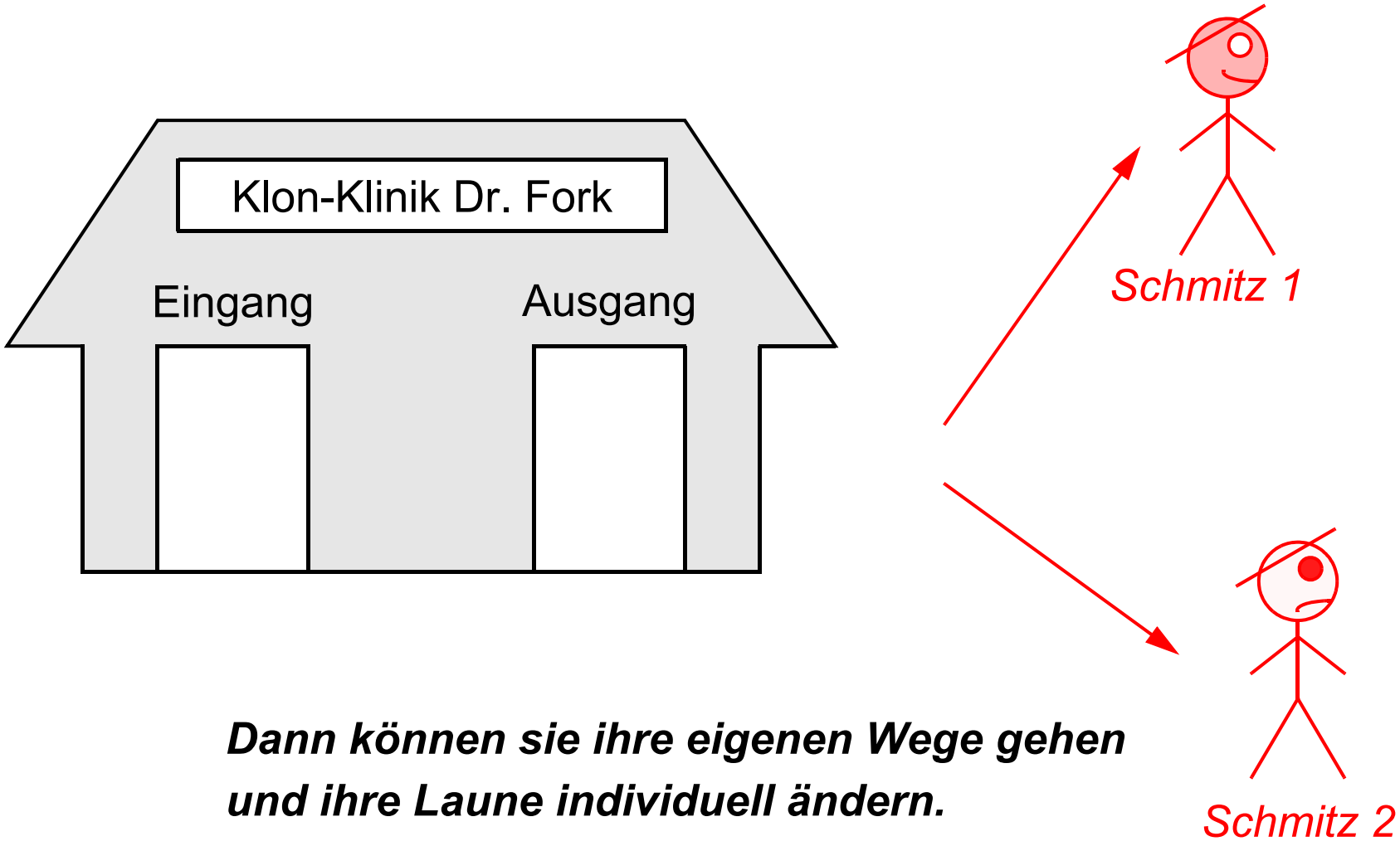
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



Zwei identische Personen verlassen die Klinik:

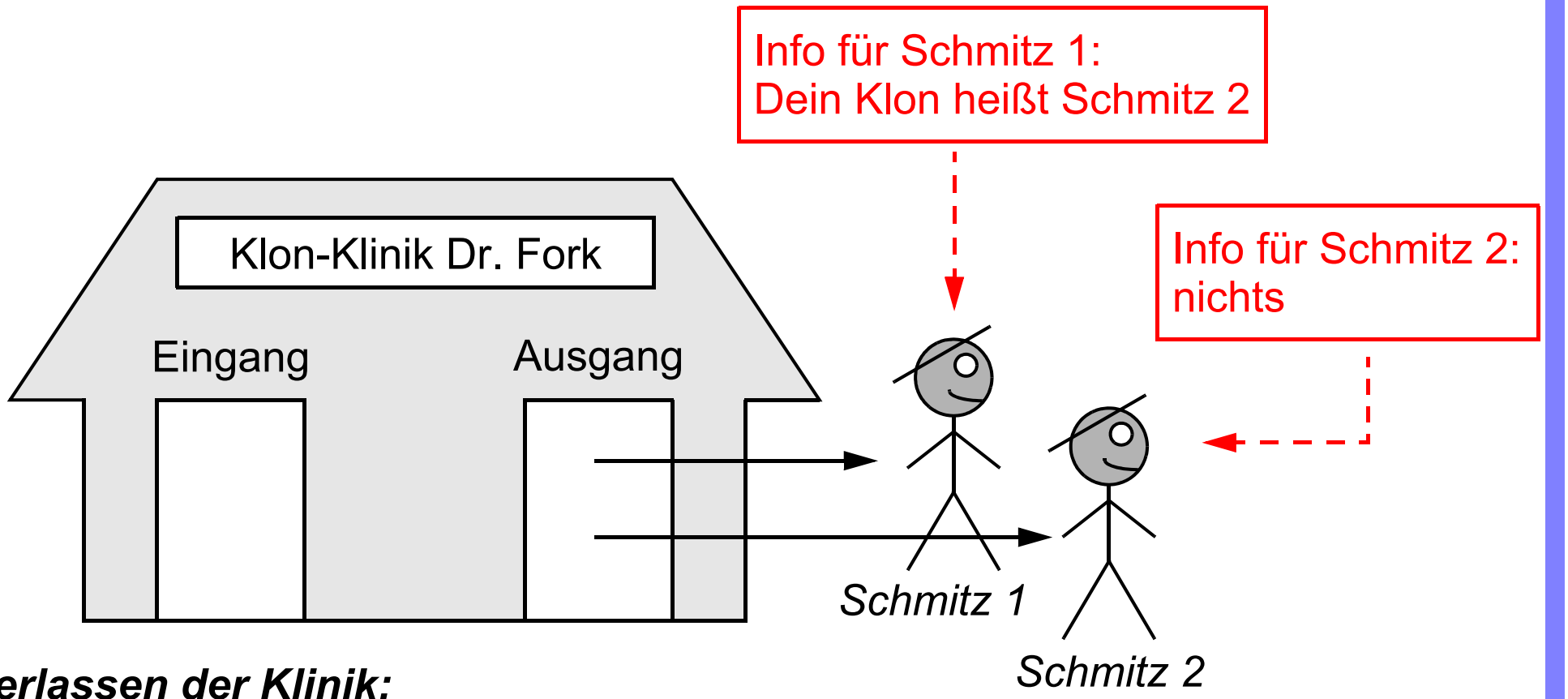
Sie tun zunächst dasselbe am selben Ort,
sie haben zunächst dieselbe Laune,
sie sind aber zwei eigenständige Individuen mit eigenen Namen.

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



Dann können sie ihre eigenen Wege gehen und ihre Laune individuell ändern.

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



Beim Verlassen der Klinik:

Die "Original-Person" wird informiert, wie ihr Klon heißt.
Der Klon erhält keine solche Information.
Hieran kann eine Person auch erkennen,
ob sie das Original oder der Klon ist,
und ihr Leben entsprechend fortsetzen.

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

Die fork()-Funktion in Unix/Linux

1.) Analog-Beispiel

2.) Arbeitsweise

3.) Speicherkonzept

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

Die fork()-Funktion in Unix/Linux

```
pid_t fork(void);
```

- Erzeugt eine Kopie des aufrufenden Prozesses
  - “Sohn/Child-Prozess” entsteht als Kopie des “Vater/Parent-Prozesses”
- Gibt dem Vater die PID des Sohns zurück
- Gibt dem Sohn den Wert 0 zurück

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

Vater-Prozess beginnt die Programmausführung

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

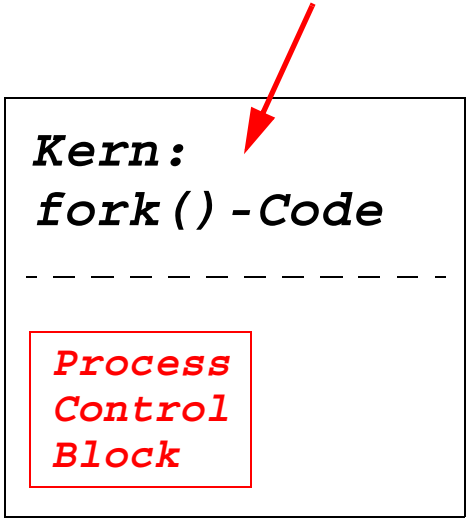
fork()-Aufruf: Vater-Prozess springt in den Unix/Linux-Kern

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

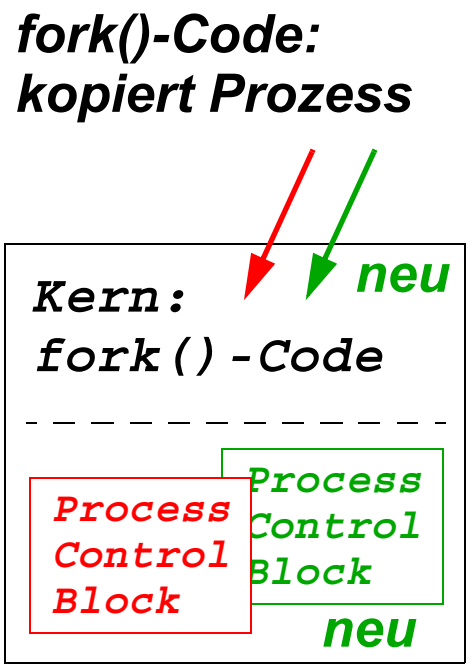
```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

im Kern:
fork()-Code
wird ausgeführt



© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

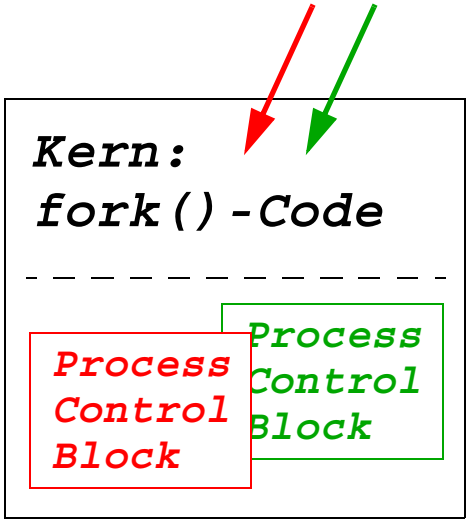
```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```



© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

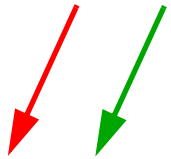
```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

zwei identische Prozesse ...



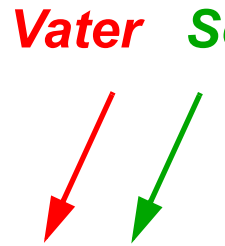
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

... kehren aus dem Kern zurück



```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

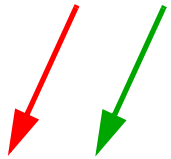
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

fork()-Rückgabe an den Vater: PID des Sohns != 0



```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

also: Vater springt hinter die if-Anweisung

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

fork()-Rückgabe an den Sohn: 0

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

```
if (fork()==0) {
  /* Code des Sohns */
  printf("Hier ist der Sohn\n");
  exit(0); /* Ende des Sohns */
}
/* Code des Vaters */
printf("Hier ist der Vater\n");
```

also: Sohn springt in die if-Anweisung

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

Die fork()-Funktion in Unix/Linux

1.) Analog-Beispiel

2.) Arbeitsweise

3.) Speicherkonzept

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

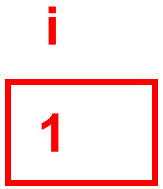
Die fork()-Funktion in Unix/Linux

- **Sohn-Speicher**
  entsteht als Kopie des Vater-Speichers
- **Aber: Zwei getrennte Speicher!**

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

```
int i = 1;

if (fork()==0) {
  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);
}

sleep(2);
printf("i im Vater: %d\n",i);
```

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/

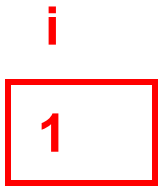



```
int i = 1;
```

Vater erzeugt und initialisiert Variable i

```
if (fork()==0) {
  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);
}
sleep(2);
printf("i im Vater: %d\n",i);
```

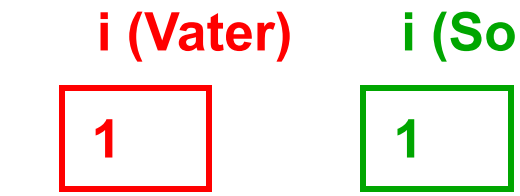
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

fork(): Sohn entsteht

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



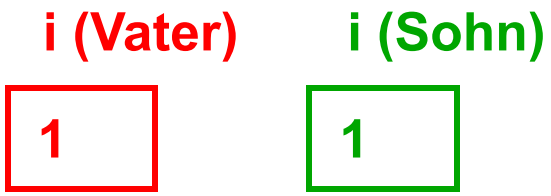
fork() kopiert den Vater:

Sohn erhält eigene Variable i

Anfangswert wird aus der Variablen des Vaters übernommen

```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

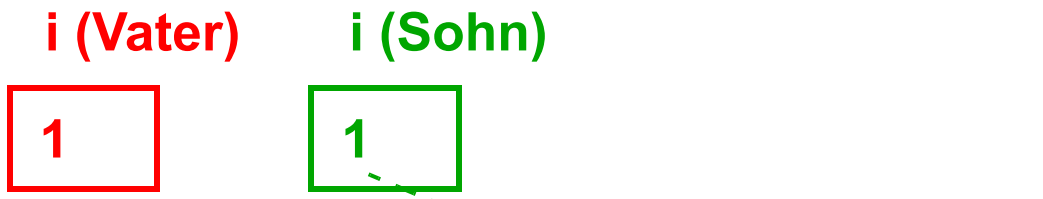
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

Vater schläft zunächst

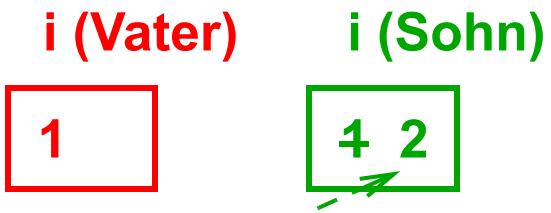
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

Sohn-Ausgabe: 1

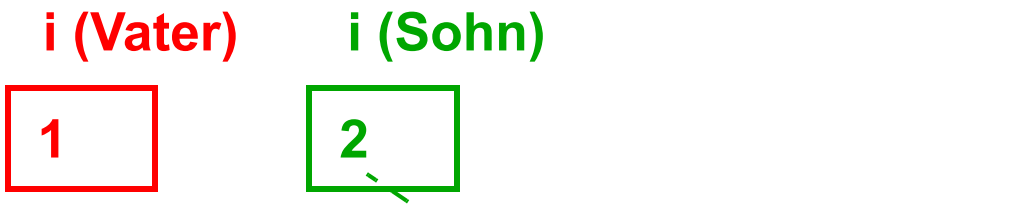
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

Sohn ändert Inhalt seiner Variablen

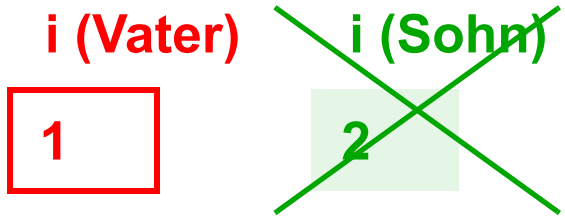
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);

}

sleep(2);
printf("i im Vater: %d\n",i);
```

Sohn-Ausgabe: 2

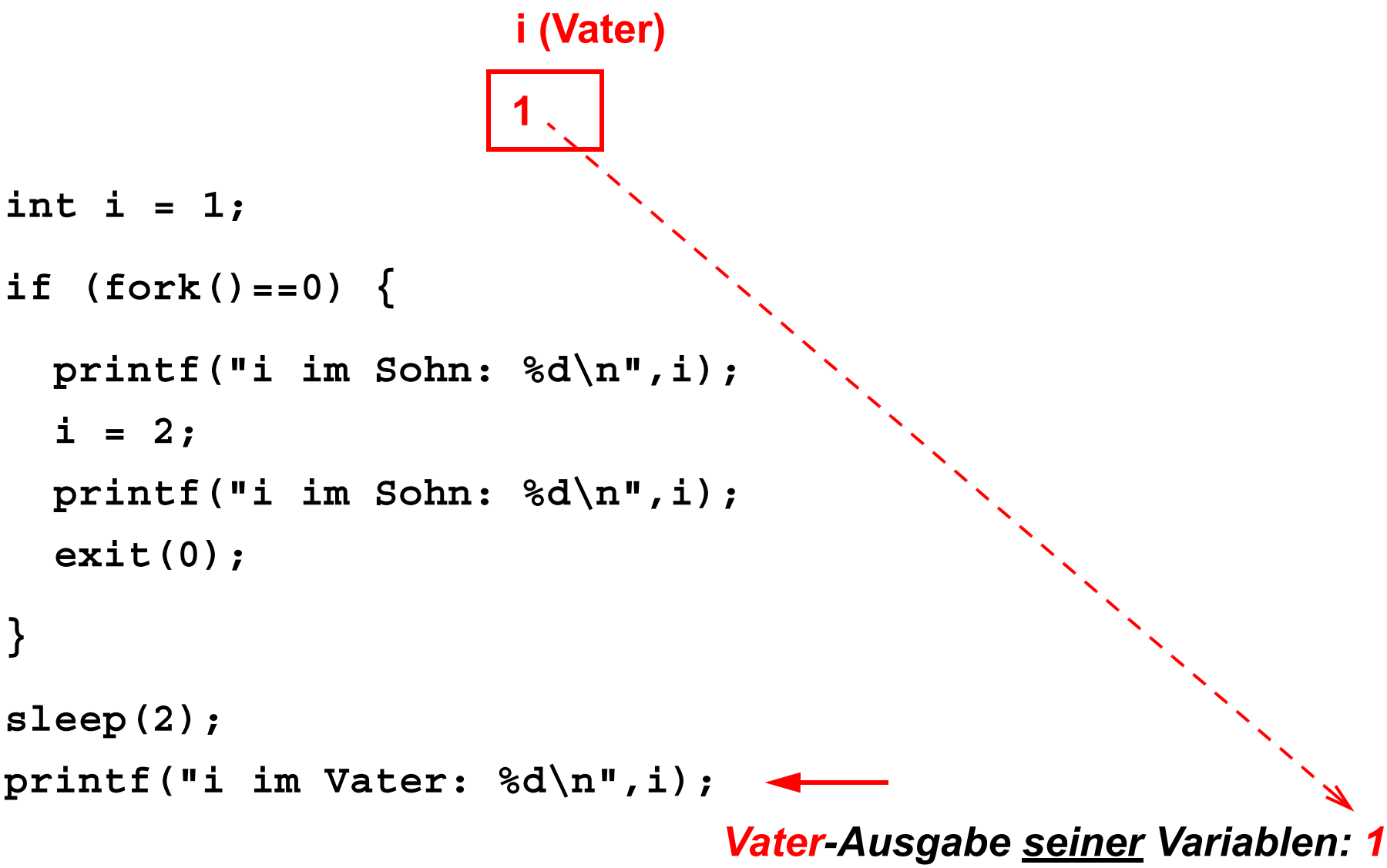
© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/




```
int i = 1;

if (fork()==0) {

  printf("i im Sohn: %d\n",i);
  i = 2;
  printf("i im Sohn: %d\n",i);
  exit(0);
}

sleep(2);
printf("i im Vater: %d\n",i);
```

Sohn beendet sich, Variable wird gelöscht

© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/



© Prof. Dr. Carsten Vogt
http://www.nt.th-koeln.de/vogt/