Schmid Kevin
1710418023

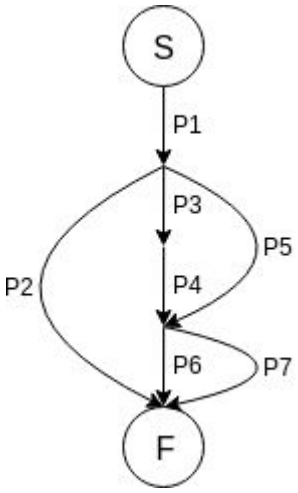
Prozesse

Verwaltung und Erzeugung

1 Prozessdesign

1. Transformation: S(p1, P(p2, S(P (S(p3, p4), p5), P(p6, p7))))

a) Prozessgraphen



b) cobegin/coend Pseudocode

```
cobegin
    p1;
    cobegin
        p2
        //
        cobegin
            cobegin
                cobegin
                    cobegin
                        p3;
                        p4
                coend
                //
                p5
            coend;
            cobegin
                p6
                //
                p7
```

```
            coend
        coend
    coend
coend
```

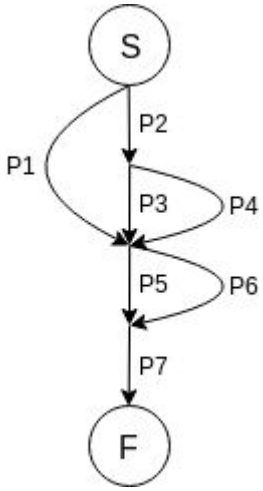
c) fork / join / quit Systemfunktionen

```
t1;
k := 1;
m := 1;
n := 2;
o := 3;
fork p2;
fork p3;
fork p5;
join k, p3; quit;
p2 : t2; join o; quit;
p3 : t3, fork p4; join m, p4; quit;
p4 : t4; join n, p6; quit;
p5 : t5; join n, p6; quit;
p6 : fork p7, t6; join o; quit;
p7 : t7; join o; quit;
```

2. Transformation

p1 := w = x_1 * x_2
p2 := v = x_3 * x_4
p3 := y = v * x_5
p4 := z = v * x_6
p5 := y = w * y
p6 := z = w * z
p7 := a = y + z

a) Prozessgraphen



Schmid Kevin
1710418023

b) cobegin/coend Pseudocode

```
cobegin
    cobegin
        p1
        //
        cobegin
            p2;
            cobegin
                p3
                //
                p4
            coend
        coend
    coend;
    cobegin
        cobegin
            p5
            //
            p6
        coend;
        p7
    coend
coend
```

c) fork / join / quit Systemfunktionen

```
m := 3;
n := 2;
fork p2;
fork p5;
fork p7;
w = x1 * x2, join m, p5; quit;
p2 : v = x3 * x4, fork p3, fork p4; quit;
p3 : y = v * x5, join m, p5; quit;
p4 : z = v * x6, join m, p5; quit;
p5 : fork p6, y = w * y, join n, p7; quit;
p6 : z = w * z, join n, p7; quit;
p7 : a = y + z; quit;
```

Schmid Kevin
1710418023

2 Prozessverwaltung

1. Prozesskontextblock mit ps

a) Lesen Sie eine einfache Prozesstabelle mit dem Kommando ps aus

```
kevin@Kevin-XPS15:~$ ps
  PID TTY          TIME CMD
13760 pts/0    00:00:00 bash
13775 pts/0    00:00:00 ps
```

b) Wie kann eine übersichtliche Prozess-Hierarchie ausgegeben werden?

```
kevin@Kevin-XPS15:~$ pstree
systemd─┬─Discord─┬─Discord───Discord───23*[{Discord}]
        │         ├─Discord───5*[{Discord}]
        │         └─29*[{Discord}]
        ├─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─agetty
        ├─avahi-daemon───avahi-daemon
        ├─bluetoothd
        ├─boltd───2*[{boltd}]
        ├─chrome─┬─chrome───chrome─┬─3*[chrome───15*[{chrome}]]
        │        │                 ├─13*[chrome───12*[{chrome}]]
        │        │                 ├─10*[chrome───13*[{chrome}]]
        │        │                 ├─chrome───5*[{chrome}]
        │        │                 ├─chrome───19*[{chrome}]
        │        │                 ├─2*[chrome───14*[{chrome}]]
        │        │                 ├─chrome───16*[{chrome}]
        │        │                 ├─chrome───17*[{chrome}]
        │        │                 └─chrome───22*[{chrome}]
        │        ├─chrome───7*[{chrome}]
        │        └─36*[{chrome}]
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───2*[dbus]
        ├─dbus-daemon
        ├─fwupd───4*[{fwupd}]
```

Schmid Kevin
1710418023

```
kevin@Kevin-XPS15:~$ ps fx
  PID TTY      STAT   TIME COMMAND
 2238 ?        Ssl    0:00 /usr/lib/gnome-session/gnome-session-binary --session=budgie-desktop
 2419 ?        Ss     0:00  \_ /usr/bin/ssh-agent /usr/bin/im-launch /usr/bin/budgie-desktop
 2454 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-keyboard
 2455 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-media-keys
 2456 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-mouse
 2457 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-power
 2458 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-print-notifications
 2459 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-screensaver-proxy
 2462 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-smartcard
 2465 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-sound
 2468 ?        Sl     0:01  \_ /usr/lib/gnome-settings-daemon/gsd-sharing
 2470 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-rfkill
 2482 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-wacom
 2483 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-xsettings
 2493 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-a11y-settings
 2502 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-color
 2506 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-clipboard
 2507 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-housekeeping
 2512 ?        Sl     0:00  \_ /usr/lib/gnome-settings-daemon/gsd-datetime
 2734 ?        Sl     3:56  \_ budgie-wm
 2743 ?        Sl     1:52  \_ budgie-panel
 2849 ?        Sl     0:01  \_ budgie-daemon
 2853 ?        Sl     0:00  \_ budgie-polkit-dialog
 2912 ?        Sl     0:01  \_ nautilus-desktop
 2914 ?        Sl     0:01  \_ nm-applet --no-indicator
 2919 ?        Sl     0:00  \_ /usr/lib/x86_64-linux-gnu/indicator-application/indicator-application-service
 2927 ?        Sl     0:08  \_ /usr/bin/python /usr/bin/solaar
 2930 ?        Sl     0:00  \_ /usr/lib/gnome-disk-utility/gsd-disk-utility-notify
 2939 ?        Sl     0:04  \_ plank
 2948 ?        Sl     0:00  \_ /usr/lib/evolution/evolution-data-server/evolution-alarm-notify
 4163 ?        Sl     0:18  \_ usr/share/jetbrains-toolbox/jetbrains-toolbox --minimize
 4203 ?        Sl     0:01  |   \_ /tmp/.mount_jetbraifdHn6/usr/share/jetbrains-toolbox/CEF/jetbrains-toolbox-ce
 4219 ?        S      0:00  |       \_ /tmp/.mount_jetbraifdHn6/usr/share/jetbrains-toolbox/CEF/jetbrains-toolbo
 4240 ?        Sl     0:01  |       |   \_ /tmp/.mount_jetbraifdHn6/usr/share/jetbrains-toolbox/CEF/jetbrains-to
 4232 ?        Sl     0:00  |       \_ /tmp/.mount_jetbraifdHn6/usr/share/jetbrains-toolbox/CEF/jetbrains-toolbo
 4469 ?        Sl     0:00  \  update-notifier
```

c) Lesen Sie detaillierte Informationen mit ps –aF oder ps –aux aus und interpretieren Sie das Ergebnis. Welcher Wert beschreibt die Rechenzeit, den belegten Speicher?

```
kevin@Kevin-XPS15:~$ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 193224  9480 ?        Ss   11:18   0:02 /sbin/init nogpumanager
root           2  0.0  0.0      0     0 ?        S    11:18   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   11:18   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   11:18   0:00 [rcu_par_gp]
```

PID: Prozess ID
%CPU: Verwendete CPU-Zeit dividiert durch die Zeit, in der der Prozess ausgeführt wurde (CPU-Zeit/Echtzeit-Verhältnis), ausgedrückt in Prozent
%MEM: Verhältnis von RSS zum physikalischen Speicher auf der Maschine, ausgedrückt in Prozent
VSZ: Virtuelle Speichergröße in KiB
RSS: Der nicht ausgelagerte physische Speicher in KB
TIME: kumulierte CPU-Zeit

d. Was ist der Unterschied zwischen PID und PPID?

PID: Prozess ID
PPID: Prozess ID des Elternprozesses, der den aktuellen Prozess gestartet hat

e. Wie können alle Threads ausgeben werden?

ps -T

```
kevin@Kevin-XPS15:~$ ps -T -p 6313
  PID  SPID TTY          TIME CMD
 6313  6313 ?        00:00:10 Discord
 6313  6436 ?        00:00:00 sandbox_ipc_thr
 6313  6438 ?        00:00:03 Chrome_IOThread
 6313  6439 ?        00:00:00 Discord
 6313  6440 ?        00:00:03 Discord
 6313  6441 ?        00:00:03 Discord
 6313  6442 ?        00:00:03 Discord
 6313  6443 ?        00:00:00 Discord
 6313  6444 ?        00:00:00 NetworkChangeNo
 6313  6445 ?        00:00:00 D-Bus thread
```

2. Prozessmonitoring mit top

a. Testen Sie top und vergleichen Sie die Ergebnisse mit ps

```
top - 14:43:06 up  3:24,  1 user,  load average: 0,41, 0,64, 0,65
Tasks: 379 total,   1 running, 378 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,5 us,  1,1 sy,  0,0 ni, 97,1 id,  0,2 wa,  0,0 hi,  0,1 si,  0,0 st
MiB Mem :  15680,2 total,   7941,0 free,   4045,9 used,   3693,3 buff/cache
MiB Swap:    980,0 total,    980,0 free,      0,0 used.  10495,9 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1399 root      20   0  656600 148160 123636 S  10,6   0,9   6:05.23 Xorg
 2734 kevin     20   0 1160968  62516  43340 S   8,9   0,4   4:40.28 budgie-wm
17618 kevin     20   0  512656  34416  26420 S   3,3   0,2   0:00.28 gnome-screensho
 4912 kevin     20   0 4104488 529452 134280 S   2,3   3,3   7:37.29 chrome
 2743 kevin     20   0 1653800 103992  46336 S   1,3   0,6   2:14.84 budgie-panel
 6501 kevin     20   0 1262980 232856  86344 S   1,3   1,5   3:02.82 Discord
 2495 kevin      9 -11 3012736  20028  15144 S   0,7   0,1   2:28.37 pulseaudio
 4203 kevin     20   0 1931244  77228  64912 S   0,7   0,5   0:01.44 jetbrains-toolb
 6313 kevin     20   0  997704 125268  85696 S   0,7   0,8   0:24.73 Discord
```

Das Top-Programm bietet eine dynamische Echtzeitansicht eines laufenden Systems. Es kann zusammengefasste Systeminformationen sowie eine Liste von Prozessen oder Threads anzeigen, die derzeit vom Linux-Kernel verwaltet werden.

b. Finden Sie mit top die Prozessnummer von top selbst heraus und beenden Sie aus einer anderen Shell das Programm mit dem kill Signal

```
top - 14:57:57 up  3:39,  1 user,  load average: 0,89, 0,82, 0,76
Tasks: 379 total,   1 running, 378 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,2 us,  0,8 sy,  0,0 ni, 97,8 id,  0,1 wa,  0,0 hi,  0,1 si,  0,0 st
MiB Mem :  15680,2 total,   7800,8 free,   4149,2 used,   3730,2 buff/cache
MiB Swap:    980,0 total,    980,0 free,      0,0 used.  10373,1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
18916 kevin     20   0   31904   4252   3284 R   0,3   0,0   0:00.60 top
```

in top mit 'o' den Filter öffnen und den Filter 'COMMAND=top' angeben

```
kevin@Kevin-XPS15:~$ kill -9 18916
kevin@Kevin-XPS15:~$ _
```

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
18916 kevin     20   0   31904   4252   3284 R   0,3   0,0   0:01.24 top
Killed
kevin@Kevin-XPS15:~$
```

c. An welchem Parameter ist der Zustand der Prozesse ersichtlich? Welche Möglichkeiten gibt es?

Parameter S

```
29. S  --  Process Status
    The status of the task which can be one of:
        D = uninterruptible sleep
        I = idle
        R = running
        S = sleeping
        T = stopped by job control signal
        t = stopped by debugger during trace
        Z = zombie
```

d. Kann das Scheduling der Prozesse konfiguriert bzw. beeinflusst werden?

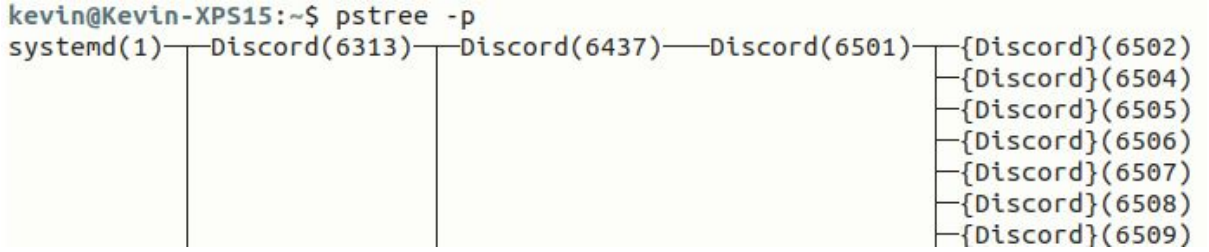
Über den 'nice' Wert. Ein negativer Wert bedeutet höhere Priorität, während ein positiver Wert niedrigere Priorität bedeutet.

3. Prozesshierarchie mit pstree

a. Welcher Prozess steht immer an der ersten Stelle der Hierachie?

Der init Prozess mit der PID 1.

b. Wie kann die Prozesshierarchie mit PIDs ausgegeben werden?



c. Wie kann die Prozesshierarchie mit den entsprechenden Elternprozessen ausgegeben werden?

pstree zeigt die Hierarchie der Prozesse an, somit werden die Elternprozesse immer ausgegeben.

d. Wie kann die Prozesshierarchie eines Benutzers ausgegeben werden?

Indem man den Namen des Benutzers als letztes Argument an pstree übergibt

```
kevin@Kevin-XPS15:~$ pstree kevin
Discord─┬─Discord───Discord───24*[{Discord}]
        ├─Discord───5*[{Discord}]
        └─31*[{Discord}]

chrome─┬─chrome───chrome─┬─14*[chrome───13*[{chrome}]]
       │                 ├─13*[chrome───12*[{chrome}]]
       │                 ├─3*[chrome───15*[{chrome}]]
       │                 ├─chrome───5*[{chrome}]
       │                 ├─chrome───20*[{chrome}]
       │                 ├─chrome───16*[{chrome}]
       │                 ├─chrome───18*[{chrome}]
       │                 └─chrome───14*[{chrome}]
       ├─chrome───7*[{chrome}]
       └─36*[{chrome}]

gnome-keyring-d───3*[{gnome-keyring-d}]

gsd-printer───2*[{gsd-printer}]

ibus-daemon─┬─ibus-dconf───3*[{ibus-dconf}]
            ├─ibus-engine-sim───2*[{ibus-engine-sim}]
            ├─ibus-extension-───3*[{ibus-extension-}]
            ├─ibus-ui-gtk3───3*[{ibus-ui-gtk3}]
            └─2*[{ibus-daemon}]

ibus-x11───2*[{ibus-x11}]

jetbrains-toolb───{jetbrains-toolb}
```

3 Prozesserzeugung mit fork

3.1 Prozesserzeugung

```
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ ./s
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55cf7e0bc010
lokal_var  = 1 Speicheradresse : 0x7ffead163bd0
--- Im Kindprozess ---
global_var = 1 Speicheradresse : 0x55cf7e0bc010
lokal_var  = 1 Speicheradresse : 0x7ffead163bd0
--- Im Kindprozess ---
global_var = 2 Speicheradresse : 0x55cf7e0bc010
lokal_var  = 2 Speicheradresse : 0x7ffead163bd0
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55cf7e0bc010
lokal_var  = 1 Speicheradresse : 0x7ffead163bd0
```

Schmid Kevin
1710418023

1. Was ist auf den ersten Blick das Bemerkenswerte bei der Ausgabe von Eltern- und Kindprozess?

Das obwohl die Speicheradresse ident ist, die Werte der Variablen sich unterscheiden.

2. Synchronisiert die Systemfunktion sleep() die Ausgabe oder verzögert sie diese nur

Verzögert

3. Eine eigene Kopie eines Speichersegmentes wird wirklich erst erstellt, wenn in diese durch den Kindprozess auch geschrieben wird, wodurch zeitaufwändiges Kopieren erspart wird – ist diese Aussage richtig oder falsch?

Richtig

4. Nicht kopiert werden die Speichersegmente Stack und Heap – ist diese Aussage richtig oder falsch?

Richtig (gleicher Grund als in Punkt 3.1.3)

5. Interpretieren Sie die Reihenfolge der Ausgabe dieses und auch des Programmes oben in Bezug auf Scheduling und Zustandsdiagramm. Wer legt die Reihenfolge fest und durch was kann diese beeinflusst werden?

Der nice Wert kann die Reihenfolge beeinflussen (siehe 2.2.d). Die Ausgabe am Terminal ist jedoch vom Puffer abhängig.

Schmid Kevin
1710418023

6. Die Reihenfolge der Ausgabe kann sich bei mehrfachen Aufrufen unterscheiden. Dahinter steckt neben dem Prinzip des Scheduling auch das der Pufferung, welches mit der Ausgabe in eine Datei verhindert werden kann

a. Leiten Sie die Ausgabe in eine Datei um und beachten Sie den Unterschied

```
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ ./s
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x563e44af3010
lokal_var  = 1 Speicheradresse : 0x7ffe67a40e40
--- Im Kindprozess ---
global_var = 1 Speicheradresse : 0x563e44af3010
lokal_var  = 1 Speicheradresse : 0x7ffe67a40e40
--- Im Kindprozess ---
global_var = 2 Speicheradresse : 0x563e44af3010
lokal_var  = 2 Speicheradresse : 0x7ffe67a40e40
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x563e44af3010
lokal_var  = 1 Speicheradresse : 0x7ffe67a40e40
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ ./s > out
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ cat out
--- Im Kindprozess ---
global_var = 1 Speicheradresse : 0x55fbc0807010
lokal_var  = 1 Speicheradresse : 0x7ffe60c14400
--- Im Kindprozess ---
global_var = 2 Speicheradresse : 0x55fbc0807010
lokal_var  = 2 Speicheradresse : 0x7ffe60c14400
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55fbc0807010
lokal_var  = 1 Speicheradresse : 0x7ffe60c14400
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55fbc0807010
lokal_var  = 1 Speicheradresse : 0x7ffe60c14400
```

Schmid Kevin
1710418023

b. Wie könnte der Code mit fflush(stdout)ergänz werden, dass die Ausgabe ohne Pufferung funktioniert?

```c
int main (void) {
   pid_t pid;
   int lokal_var = 1;
   switch (pid = fork ()) {

   case -1:
      printf ("Fehler bei fork()\n");
      fflush(stdout);
      break;

   case 0:
      sleep (1);
      printf ("--- Im Kindprozess ---\n");
      printf ("global_var = %d Speicheradresse : %p\n",
               global_var, &global_var);
      printf ("lokal_var  = %d Speicheradresse : %p\n",
               lokal_var, &lokal_var);
      fflush(stdout);

      ++global_var;
      ++lokal_var;

      printf ("--- Im Kindprozess ---\n");
      printf ("global_var = %d Speicheradresse : %p\n",
               global_var, &global_var);
      printf ("lokal_var  = %d Speicheradresse : %p\n",
               lokal_var, &lokal_var);
      fflush(stdout);
      break;

   default:
      printf ("--- Im Elternprozess ---\n");
      printf ("global_var = %d Speicheradresse : %p\n",
               global_var, &global_var);
      printf ("lokal_var  = %d Speicheradresse : %p\n",
               lokal_var, &lokal_var);
      fflush(stdout);

      sleep (2);

      printf ("--- Im Elternprozess ---\n");
      printf ("global_var = %d Speicheradresse : %p\n",
               global_var, &global_var);
      printf ("lokal_var  = %d Speicheradresse : %p\n",
               lokal_var, &lokal_var);
      fflush(stdout);
      break;
   }
   return EXIT_SUCCESS;
```

Schmid Kevin
1710418023

```
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ ./s > out
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ cat out
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55603c44b010
lokal_var  = 1 Speicheradresse : 0x7fff65059130
--- Im Kindprozess ---
global_var = 1 Speicheradresse : 0x55603c44b010
lokal_var  = 1 Speicheradresse : 0x7fff65059130
--- Im Kindprozess ---
global_var = 2 Speicheradresse : 0x55603c44b010
lokal_var  = 2 Speicheradresse : 0x7fff65059130
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55603c44b010
lokal_var  = 1 Speicheradresse : 0x7fff65059130
```

Schmid Kevin
1710418023

7. Erweitern Sie den Code so, dass der Kindprozess einen neuen Kindprozess erzeugt, dieser also zum Elternprozess wird und der Elternprozess Großvater wird

```c
int main (void) {
    pid_t pid_child;
    pid_t pid_grandchild;
    int lokal_var = 1;
    int is_grandchild = 0;
    switch (pid_child = fork ()) {

    case -1:
        printf ("Fehler bei fork()\n");
        fflush(stdout);
        break;

    case 0:
        switch(pid_grandchild = fork()) {
            case -1:
                printf ("Fehler bei fork()\n");
                fflush(stdout);
                break;
            case 0:
                sleep(2);
                printf ("--- Im Enkelkindprozess ---\n");
                printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
                printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
                fflush(stdout);

                global_var += 2;
                lokal_var += 2;

                printf ("--- Im Enkelkindprozess ---\n");
                printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
                printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
                fflush(stdout);

                is_grandchild = 1;
                break;
        }
        if(!is_grandchild) {
            sleep(1);
            printf ("--- Im Kindprozess ---\n");
            printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
            printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
            fflush(stdout);

            ++global_var;
            ++lokal_var;
            sleep(2);

            printf ("--- Im Kindprozess ---\n");
            printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
            printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
            fflush(stdout);
        }
        break;

    default:
        printf ("--- Im Elternprozess ---\n");
        printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
        printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
        fflush(stdout);

        sleep (4);

        printf ("--- Im Elternprozess ---\n");
        printf ("global_var = %d Speicheradresse : %p\n", global_var, &global_var);
        printf ("lokal_var  = %d Speicheradresse : %p\n", lokal_var, &lokal_var);
        fflush(stdout);
        break;
    }
    return EXIT_SUCCESS;
}
```

Schmid Kevin
1710418023

```
kevin@Kevin-XPS15:~/FH/S4/sysprog/ue2$ ./s
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55b5b17a3010
lokal_var  = 1 Speicheradresse : 0x7ffc171860b8
--- Im Kindprozess ---
global_var = 1 Speicheradresse : 0x55b5b17a3010
lokal_var  = 1 Speicheradresse : 0x7ffc171860b8
--- Im Enkelkindprozess ---
global_var = 1 Speicheradresse : 0x55b5b17a3010
lokal_var  = 1 Speicheradresse : 0x7ffc171860b8
--- Im Enkelkindprozess ---
global_var = 3 Speicheradresse : 0x55b5b17a3010
lokal_var  = 3 Speicheradresse : 0x7ffc171860b8
--- Im Kindprozess ---
global_var = 2 Speicheradresse : 0x55b5b17a3010
lokal_var  = 2 Speicheradresse : 0x7ffc171860b8
--- Im Elternprozess ---
global_var = 1 Speicheradresse : 0x55b5b17a3010
lokal_var  = 1 Speicheradresse : 0x7ffc171860b8
```