**Aufgabe 2.3: Prozesse und Threads (1 Punkt) (Theorie1)**

1. **a)  Erklären Sie, was Prozesse sind und wofür sie benötigt werden. Gehen Sie dabei auf die Aussage**

**”Prozesse sind instanziierte Programme“ ein. (0,3 Punkte)**

Ein Prozess ist eine Instanziierung eines Programms für die Ausführung innerhalb eines Rechensystems. Prozesse sind dynamisch, da sie zum Beispiel von unterschiedlichen Systemen auf unterschiedliche Arten ausgeführt werden.

Prozesse werden benötigt, um ein Programm auf dem jeweiligen Rechensystem ausführen zu können. Um auf die Aussage „Prozesse sind instanziierte Programme“ einzugehen, treffen wir folgende Aussage: Ein Prozess entsteht genau dann, wenn ein Befehl eines Programmes von dem Prozessor ausgeführt wird.

Quelle: Christian Ullenboom: Java ist auch eine Insel. Einführung, Ausbildung, Praxis. 11., aktualisierte und überarbeitete Auflage, Galileo Computing: Bonn, 2014, S. 902.

1. **b)  Erklären Sie die Begriffe *Parallelität* und *Nebenläufigkeit*? (0,2 Punkte)**

Parallelität:

* + Voraussetzung für mehrere Prozesse
  + Jedem Prozess wird eine Prozessor zugewiesen
  + Daraus folgt: mehrere Prozesse werden zur gleichen Zeit ausgeführt

Nebenläufigkeit:

* + Um den Prozessor voll auszulasten und bei dem Benutzer den Eindruck zu erwecken, dass Prozesse gleichzeitig ablaufen, werden verschiedene Prozesse verzahnt ausgeführt (resign)
  + Unterbrechung (block) eines laufenden Prozesses (z.B. Hardware Interrupt) führt zur Ausführung eines anderen Teilprozesses bis ein weiterer Prozess aufgenommen wird.

Quelle: Foliensatz Prozesse Kao/Nordholdz (Folie 12)

1. **c)  Nennen Sie zwei Ereignisse, die eine Prozessumschaltung zur Folge haben und erklären Sie, wie dieses Umschalten realisiert wird. Gehen Sie dabei auch auf den Begriff ”Process Control Block“ (PCB) ein. (0,2 Punkte)**

Mögliche Ereignisse für Prozessumschaltung:

* + Zeitscheibe verbraucht
  + Blockierender Systemaufruf

Das Umschalten wird realisiert, indem die Registerinhalte des laufenden Prozesses im PCB abgespeichert werden (inklusive Befehlszähler und Stelle an dem der jeweilige Prozess unterbrochen wurde).

Danach wird der Prozesszustand auf blockiert gestellt. Sobald der Prozess wieder zum Laufen gebracht wird, werden Inhalte aus dem PCB gelesen und der Prozessorzustand auf laufen gesetzt. Fortsetzung des Befehls an hinterlegter Stelle im PCB.

Quelle: Foliensatz Prozesse Kao/Nordholdz (Folie 18)

1. **d)  Was unterscheidet einen User Level Thread von einem Prozess? (0,3 Punkte)**

Ein Prozess besteht aus mindestens einem User-Level Thread.

**NOCH NICHT VOLLSTÄNDIG!**

Quelle: Foliensatz Prozesse Kao/Nordholdz (Folie 32, 33)

**Aufgabe 2.4: Parallelisierung II (1 Punkt) (Theorie1) Gegeben ist das folgende nicht-parallele C-Programm. Die Funktionen jobA, jobB, ..., jobK wurden zuvor**

**im Programm implementiert und enthalten länger laufende Berechnungen.**

**int main(void) {**

**int a, b, c, d, e, f, g, h, i, j, k; a = jobA();  
b = jobB(a);  
c = jobC(a);**

**d = jobD(c);**

**e = jobE();**

**f = jobF(c, e);**

**g = jobG(d, f);**

**h = jobH();**

**i = jobI(g);**

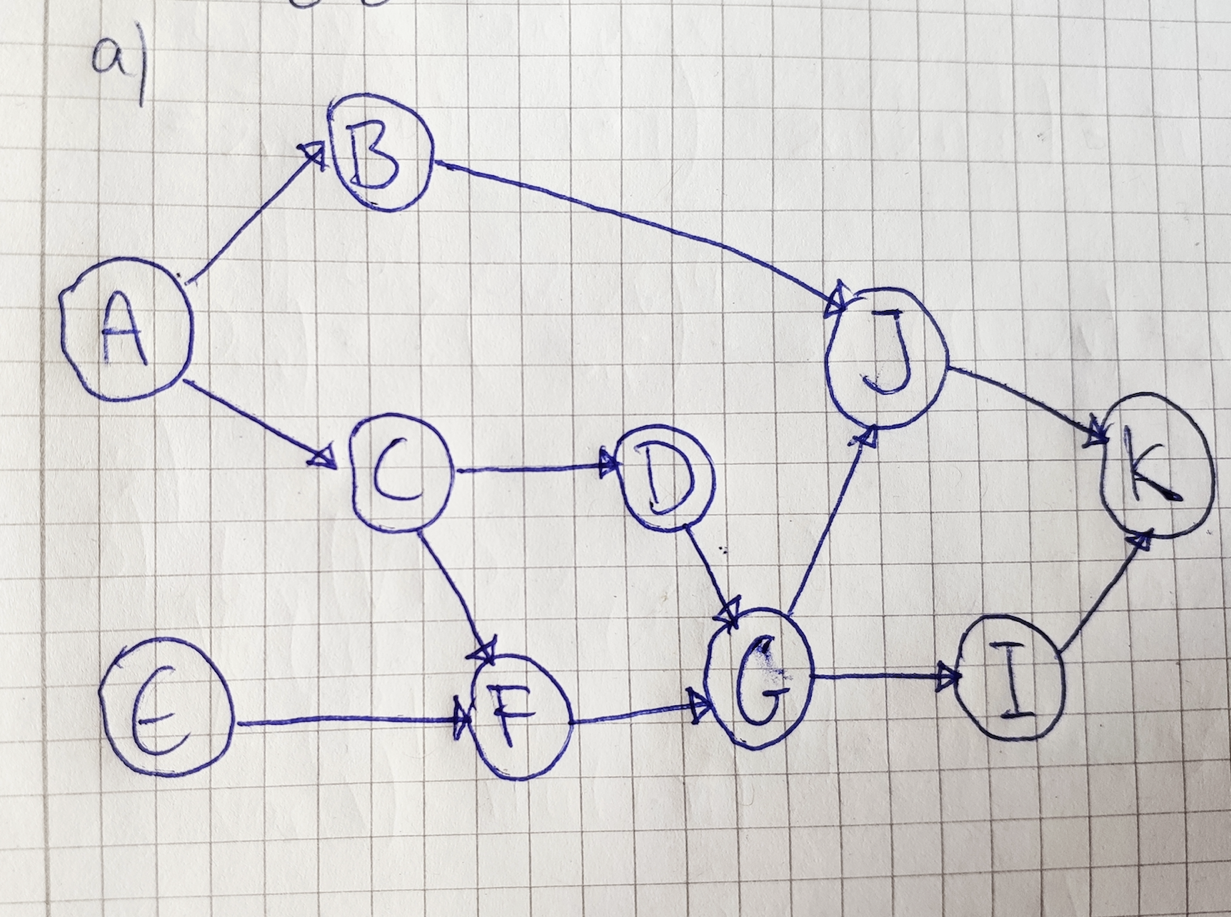
**j = jobJ(b, g);**

**k = jobK(i, j);**

**return k;**

**}**

1. **a)  Zeichnen Sie einen Prozessabhängigkeitsgraphen, der die Abhängigkeiten der einzelnen Jobs dar- stellt. Jede aufgerufene Funktion soll dabei einem Task bzw. Prozess/Thread entsprechen. (0,5 Punkte)**



1. **b)  Schreiben Sie basierend auf dem Graphen ein Programm in Pseudocode mit fork/join, das die Jobs möglichst effizient abarbeitet. (0,5 Punkte)**

fork(H)

fork(E)

A

Fork(B)

C

Fork(D)

Join(E)

F

Join(D)

G

Fork(I)

Join(B)

J

Join(I)

K