**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)**

Prozesse sind dynamische Objekte, die sequentielle Aktivitäten in einem System repräsentieren. Prozesse gelten als Beschreibungseinheit von System- und Anwendungsprogrammen und sind somit auch eine: `Instanz eines Programmes´.

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

Parallelität in Prozess Abläufen geschieht dann, wenn Prozesse miteinander verzahnt werden. Dies bedeutet, dass wirklich gleichzeitig mehrere Prozesse auf verschiedenen Kernen (CPU´s) ausgeführt werden können und somit parallel zu einander laufen.

Nebenläufigkeit beschreibt den sequentiellen Programm Ablauf. Dies bedeutet, dass Programme als parallel laufend erscheinen können, in Wirklichkeit wechseln sich jedoch die verschiedenen Prozesse innerhalb in ihrer Laufzeit ab. Dies führt dazu, das Prozesse als parallel laufend erscheinen könnten, obwohl diese aber nur sequentiell laufen.

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

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