Übungsblatt 04

Aufgabe 4.4.

a)

Da beide while Schleifen durchgehend unkoordiniert ausgeführt werden, läuft das Laufband die ganze Zeit, genauso wie Schraubmaschine und die teuren Autos werden zerkratzt.

b)

Spurious Wake Ups, sind unberechtigte Aktivierungen von Threads. Obwohl ein anderer Thread das Signal zum Wakeup nicht gegeben hat ist er trotzdem aufgewacht. Um das zu umgehen werden die waits in while-Schleifen implementiert, welche eine Bedingungsvariable überprüfen, die vom signalisierenden Thread gegeben wird. Die pthread Bibliothek lässt nämlich zu, dass waits auch für andere Signale aufwachen können. Es wäre zu umständlich die gesamte pthread Bibliothek neu zu schreiben, weshalb man while-Schleifen nutzt.

c)

Global:

```
State s=belt;                    //Werte:'belt'oder'screw'
Mutex belt_move_mutex
Mutex state_mutex
Signal band_wartet_signal
Signal screwing_finished

Beförderungsband-Thread:
    while(1) {
        lock(state_mutex);
        while(s!=belt){
            unlock(state_mutex);
            wait(screwing_finished);
            lock(state_mutex);
        }
    belt_move(1);                //Bandnachvornebewegen
    s = screw;
```

```
            signal(band_steht_signal);
            unlock(state_mutex);
        }
```

Schraubarm-Thread:

```
        while(1) {
            lock(state_mutex);
            while(s!=screw){
                unlock(state_mutex);
                wait(band_steht_signal);
                lock(state_mutex);
            }

        arm_dock();                 //Schraubmaschineandocken
        screw();                    //Schraubenfestdrehen
        arm_undock();               //Schraubmaschineabdocken
        s = belt;
        signal(screwing_finished);
        unlock(state_mutex);
        }
```

d)

Es gibt zwei Arten von expliziter Prozessinteraktion: Konkurrenz und Kooperation
- Konkurrenz: mehrere Prozesse bewerben sich um exklusives Betriebsmittel
→ Synchronisationsmechanismus notwendig um gleichzeitige Zugriffe auf das gleiche BM zu vermeiden
- Kommunikation: mehrere Prozesse teilen gezielt Informationen
→ Synchronisation und Kommunikation(müssen von der Existenz aller beteilgten Prozesse wissen und über ausreichend Informationen verfügen)
(Folien Synchronisation S. 2-6)

In unserem Fall handelt es sich um Kooperation, da die beiden Thread sich über weiterfahren des Bandes abstimmen müssen, das Band darf erst weiterfahren wenn der Schraubvorgang abgeschlossen ist.

Aufgabe 4.5.

a)

Notwendiges Kriterium (Schedulability Test, Feasibility Test):

Es gilt für jeden Periodischen Prozess hier $0<b_i \leq d_i \leq p_i$.

Für alle Prozesse gilt:

$\sum b_i/p_i \leq 1$

$\Leftrightarrow (1+2+1)/(6+6+4) = 1/4 \leq 1$

Demnach muss ein zulässiger Schedule für diese Prozesse existieren.

Quelle: Scheduling Folien S. 37

b)

Takt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
CPU	C	B	B	A	C	X	B	B	C	A	X	X	C	B	B	A	C	X	B	B	C	A	X	X
Queue	B A	A	A				A	A	A				B A	A	A				A	A	A			

Hyperperiode rot gekennzeichnet.

X kennzeichnet Leerzeiten.

c)

Die Hinreichende Bedingung für RMS lautet:

$\sum b_i/p_i \leq n(2^{1/n}-1)$

$\Leftrightarrow 0{,}25 \leq 0{,}78$

Die hinreichende Bedingung ist erfüllt, die Prozesse sind zudem unabhängig und es gilt $p_i=d_i$, die Sollzeitpunkte fallen mit den Perioden zusammen. RMS ist ein gültiger Schedule.

d)

Für die notwendige Bedingung würde gelten:

$\sum b_i/p_i \leq 1$

$\Leftrightarrow (1+2+1+3)/(6+6+4+19) \leq 1$

$\Leftrightarrow 0{,}2 \leq 1$

Somit ist diese erfüllt.

Für die Hinreichende Bedingung des RMS gilt:

$\sum b_i/p_i \leq n(2^{1/n}-1)$

$\Leftrightarrow 0{,}2 \leq 0{,}76$

Diese ist ebenfalls erfüllt, somit ist RMS ein gültiger Schedule.

Das kleinste gemeinsame Vielfache aller Perioden ist 228. Somit würde die Hyperperiode sich über 228 Takte erstrecken.