OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Institut für Intelligente Kooperierende Systeme (IKS)
Prof. Dr. David Hausheer

Technische Informatik 2 - SS 2018 - Theoretische Übung 2

Abgabe: 06.05.2018

Technische Informatik II

Übung 2: Prozesse und Threads

Aufgabe 1: Prozesse und Threads

a) Wie verhält sich eine Applikation die aus mehreren Prozessen bzw. aus mehreren Userspace-Threads besteht bei einem blockierenden Diskzugriff?

b) Welche der folgenden Ressourcen werden von allen Threads eines Prozesses geteilt und welche bestehen pro Thread? Program-Counter, Heap-Speicher, globale Variablen, Stack, CPU-Register, geöffnete Dateien, Accounting- und Benutzer-Informationen.

Aufgabe 2: Scheduling

a) Was versteht man unter Langzeit-Scheduling (Job-Scheduling), was unter Kurzzeit-Scheduling (CPU-Scheduling)?

b) Warum ist es für den Scheduler sinnvoll zwischen *I/O-bound* und *CPU-bound* Prozessen zu unterscheiden?

c) Was sind die Vor- und Nachteile von *preemptiv* gegenüber *non-preemptiv* Multitasking.

d) Gegeben seien die folgenden Prozesse die alle zur Zeit $t = 0$ erstellt und gestartet werden.

Prozess	A	B	C	D	E
Priorität	2.	5.	3.	1.	4.
Bedienzeit	100	10	20	10	50

Berechnen Sie für die unten aufgeführten Scheduling-Algorithmen die mittlere Wartezeit (*Waiting-time*) $\overline{WT}$ und die Ausführungszeit (*Turnaround Time*) $\overline{TT}$ und stellen Sie den Ablauf grafisch dar.

First Come First Server (FCFS)

	5			20			40			60			80			100			120			140			160			180			200	
A																																
B																																
C																																
D																																
E																																

$\overline{WT} =$

$\overline{TT} =$

Shortest Job First (SJF)

	5			20			40			60			80			100			120			140			160			180			200	
A																																
B																																
C																																
D																																
E																																

$\overline{WT} =$

$\overline{TT} =$

Priority Scheduling (PS)

	5			20			40				60				80				100				120				140				160				180				200	
A																																								
B																																								
C																																								
D																																								
E																																								

$\overline{WT} =$

$\overline{TT} =$

Round Robin (RR) mit einer Zeitscheibe von 5

	5			20			40				60				80				100				120				140				160				180				200	
A																																								
B																																								
C																																								
D																																								
E																																								

$\overline{RT} =$

$\overline{TT} =$

Aufgabe 3: Prozess Synchronisation, Deadlock und Starvation

a) Erörtern Sie die Begriffe *Race Condition* anhand des folgenden Beispiels und markieren sie alle *Critical Sections*. Was für Probleme können bei der Ausführung auf mehreren Prozessen auftreten und wie können diese (z.B. mit Hilfe von Semaphoren oder Monitoren) behoben werden?

```
global const buf_size := 10
global var buf[buf_size]
global var num := 0

function boolean push(a)
    if (num < buf_size)
        buf[num] := a
        num := num + 1
        return true
    else
        return false
    end
end

function pop()
    if  (num > 0)
        num := num - 1
        return buf[num]
    else
        return null
    end
end
```

b) Beschreiben Sie mit Hilfe von Pseudocode die atomaren Operationen *TestAndSet*, *Swap* und *FetchAndAdd*.

c) Gegeben sei die folgende atomare Instruktion:

```
atomar function TestAndAdd(a, b, c)
    if (a == b)
        a := a + c
        return true
    else
        return false
    end
end
```

Zeigen Sie wie mit deren Hilfe eine binäre Semaphore implementiert werden kann.

d) Eine einfache Lösung des Dining-Philosophers-Problem besteht darin, dass einer der Philosophen die Gabeln in der umgekehrten Reihenfolge aufnimmt als alle anderen. Zeigen Sie, dass diese Lösung deadlockfrei ist, in dem Sie angeben welche Deadlock-Voraussetzung(en) nicht erfüllt ist/sind. Ist die Lösung auch gerecht, das heisst starvationfrei?

Aufgabe 4: Prozess Synchronisation, Deadlock und Starvation

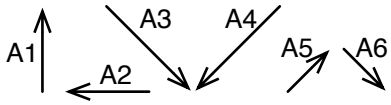
Gegeben sei ein Programm aus drei Prozessen (`P1`, `P2`, `P3`), die ein Haus zeichnen sollen (siehe unten). Es kommt kooperatives Multitasking zum Einsatz. Prozesswechsel passieren nur in der `down()` Funktion und nur, falls der laufende Prozess warten müsste.

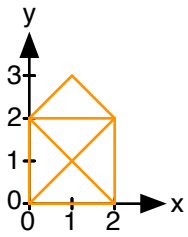
Dazu können folgende Operationen ausgeführt werden:

```
(A) draw(dx, dy)          (B) move(x, y)
```

Wobei draw eine Linie relativ zur aktuellen Position zeichnet und move den Zeichenkopf zur angegebenen Position bewegt, aber nichts zeichnet. Die Ausführung einer Operation dauert *eine Sekunde*.
Es sind folgende Operationen implementiert:

```
(A1) draw(0, 2)           (B1) move(2, 0)
(A2) draw(-2, 0)
(A3) draw(2, -2)
(A4) draw(-2, -2)
(A5) draw(1, 1)
(A6) draw(1, -1)
```





Das Programm ist unten abgebildet. Eine Programmzeile ist durch ihre Zeilennummer und durch die entsprechende Operation gekennzeichnet.

```
Init:
100 (B1)
101 semaphore S1=0
102 start(P1, P2, P3)

P1:                 P2:                 P3:
200 down(S1)        300 down(S1)        400 down(S1)
205 (A1)            305 (A1)            405 (A5)
210 (A4)            310 (A2)            410 (A6)
215 (A3)            315 up(S1)          415 (A2)
220 up(S1)                              420 up(S1)
```

1. An welcher Koordinate befindet sich der Zeichenkopf 20 Sekunden nachdem das Programm gestartet wurde?

   _____

2. Passen Sie den Gebrauch der Semaphore im obigen Programm so an, dass ein korrektes Haus gezeichnet wird. Verwenden Sie genau drei Semaphore und vergessen Sie nicht, die Semaphore zu initialisieren.

```
Init:
100 (B1)
101
102
103
104
105

P1:                     P2:                     P3:
200                     300                     400
201                     301                     401
202                     302                     402
203                     303                     403
204                     304                     404
205 (A1)                305 (A1)                405 (A5)
206                     306                     406
207                     307                     407
208                     308                     408
209                     309                     409
210 (A4)                310 (A2)                410 (A6)
211                     311                     411
212                     312                     412
213                     313                     413
214                     314                     414
215 (A3)                315                     415 (A2)
216                     316                     416
217                     317                     417
218                     318                     418
219                     319                     419
220                     320                     420
```

3. Das Programm wird auf einem Rechner ausgeführt wo Priority Scheduling verwendet wird. Prozess P1 hat die höchste Priorität und P3 die kleinste ($P(P1) > P(P2) > P(P3)$). (Prozesswechsel passieren nach wie vor nur in der `down()` Funktion).

   i) Wie viele Semaphore werden jetzt mindestens gebraucht, damit das Programm korrekt funktioniert?

   ---

   ii) Passen Sie den Gebrauch der Semaphore so an, dass mit der minimalen Anzahl Semaphore deterministisch ein korrektes Haus gezeichnet wird.

```
100 (B1)
101
102
103
104
105

P1:                    P2:                    P3:
200                    300                    400
201                    301                    401
202                    302                    402
203                    303                    403
204                    304                    404
205 (A1)               305 (A1)               405 (A5)
206                    306                    406
207                    307                    407
208                    308                    408
209                    309                    409
210 (A4)               310 (A2)               410 (A6)
211                    311                    411
212                    312                    412
213                    313                    413
214                    314                    414
215 (A3)               315                    415 (A2)
216                    316                    416
217                    317                    417
218                    318                    418
219                    319                    419
220                    320                    420
```