

## Lösung von Übungsblatt 8

### Aufgabe 1 (Unterbrechungen)

1. Was sind Interrupts?

*Interrupts sind externe Unterbrechungen. Sie werden durch Ereignisse außerhalb des zu unterbrechenden Prozesses ausgelöst (z.B. ein Ein-/Ausgabe-Gerät meldet ein E/A-Ereignis).*

2. Was ist der Unterbrechungsvektor (Interrupt Vector)?

*Eine Liste mit den Adressen aller Unterbrechungsroutinen im Betriebssystem.*

3. Was sind Exceptions?

*Exceptions sind interne Unterbrechungen oder Ausnahmen. Im Gegensatz zu Interrupts werden Sie vom zu unterbrechenden Prozess selbst ausgelöst.*

4. Was passiert, wenn es während der Behandlung eines Interrupts zu einem weiteren Interrupt kommt?

*Bei sequentieller Interrupt-Verarbeitung werden Interrupts strikt nacheinander bearbeitet und nicht selbst unterbrochen.*

*Bei verschachtelter Interrupt-Verarbeitung werden Prioritäten für die Interrupts festgelegt. Unterbrechungsbehandlungen können unterbrochen werden, wenn eine Unterbrechung mit höherer Priorität auftritt. Unterbrechungen mit niedriger Priorität werden zurückgestellt, bis alle Unterbrechungen mit höherer Priorität abgearbeitet sind.*

### Aufgabe 2 (Schedulingverfahren)

1. Warum existiert in einigen Betriebssystemen ein Leerlaufprozess?

*Ist kein Prozess im Zustand **bereit**, kommt der Leerlaufprozess zum Zug. Der Leerlaufprozess ist immer aktiv und hat die niedrigste Priorität. Durch den Leerlaufprozess muss der Scheduler nie den Fall berücksichtigen, dass kein aktiver Prozess existiert.*

2. Erklären Sie den Unterschied zwischen präemptivem und nicht-präemptivem Scheduling.

*Bei präemptivem Scheduling (verdrängendem Scheduling) kann einem Prozess die CPU vor seiner Fertigstellung entzogen werden.*

*Bei nicht-präemptivem Scheduling (nicht-verdrängendem Scheduling) kann ein Prozess die CPU so lange belegen wie er will.*

3. Nennen Sie einen Nachteil von präemptivem Scheduling.

*Höherer Overhead als nicht-präemptives Scheduling wegen der häufigeren Prozesswechsel.*

4. Nennen Sie einen Nachteil von nicht-präemptivem Scheduling.

*Belegt ein Prozess die CPU, ist es häufig so, dass andere, vielleicht dringendere Prozesse für lange Zeit nicht zum Zuge kommen.*

5. Wie funktioniert statisches Multilevel-Scheduling?

*Die bereit-Liste wird in mehrere Teillisten unterteilt. Für jede Teilliste wird eine andere Scheduling-Strategie verwendet. Die Teillisten haben unterschiedliche Prioritäten oder Zeitmultiplexe.*

6. Wie funktioniert Multilevel-Feedback-Scheduling?

*Es arbeitet mit mehreren Warteschlangen. Jede Warteschlange hat eine andere Priorität oder Zeitmultiplex. Jeder neue Prozess kommt in die oberste Warteschlange und hat damit die höchste Priorität. Innerhalb jeder Warteschlange wird Round Robin eingesetzt. Gibt ein Prozess die CPU freiwillig wieder ab, wird er wieder in die selbe Warteschlange eingereiht. Hat ein Prozess seine volle Zeitscheibe genutzt, kommt er in die nächst tiefere Warteschlange mit einer niedrigeren Priorität.*

7. Welche Schedulingverfahren sind fair?

*Ein Schedulingverfahren ist „fair“, wenn jeder Prozess irgendwann Zugriff auf die CPU erhält.*

- |   |   |
|---|---|
| <input type="checkbox"/> Prioritätengesteuertes Scheduling      | <input type="checkbox"/> Shortest Remaining Time First          |
| <input checked="" type="checkbox"/> First Come First Served     | <input type="checkbox"/> Longest Remaining Time First           |
| <input type="checkbox"/> Last Come First Served                 | <input checked="" type="checkbox"/> Highest Response Ratio Next |
| <input checked="" type="checkbox"/> Round Robin mit Zeitquantum | <input checked="" type="checkbox"/> Earliest Deadline First     |
| <input type="checkbox"/> Shortest Job First                     | <input checked="" type="checkbox"/> Fair-Share                  |
| <input type="checkbox"/> Longest Job First                      |   |

8. Welche Schedulingverfahren arbeiten präemptiv (= *unterbrechend*)?

- |   |  |
|---|--|
| <input type="checkbox"/> First Come First Served                  | <input checked="" type="checkbox"/> Longest Remaining Time First     |
| <input checked="" type="checkbox"/> Round Robin mit Zeitquantum   | <input checked="" type="checkbox"/> Fair-Share                       |
| <input type="checkbox"/> Shortest Job First                       | <input checked="" type="checkbox"/> Statisches Multilevel-Scheduling |
| <input type="checkbox"/> Longest Job First                        | <input checked="" type="checkbox"/> Multilevel-Feedback-Scheduling   |
| <input checked="" type="checkbox"/> Shortest Remaining Time First |  |

9. Bei welchen Schedulingverfahren muss die CPU-Laufzeit (= *Rechenzeit*) bekannt sein?

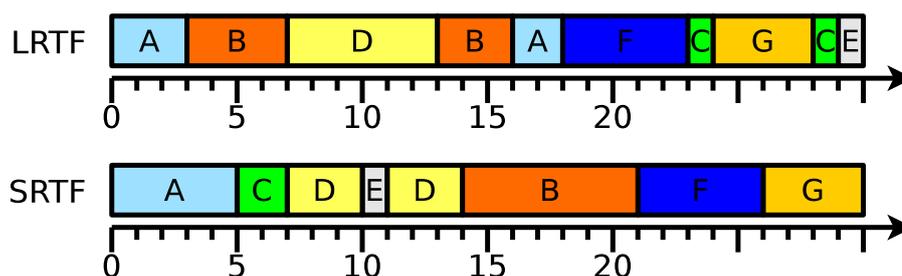
- |  |   |
|--|---|
| <input type="checkbox"/> Prioritätengesteuertes Scheduling | <input checked="" type="checkbox"/> Shortest Remaining Time First |
| <input type="checkbox"/> First Come First Served           | <input checked="" type="checkbox"/> Longest Remaining Time First  |
| <input type="checkbox"/> Last Come First Served            | <input checked="" type="checkbox"/> Highest Response Ratio Next   |
| <input type="checkbox"/> Round Robin mit Zeitquantum       | <input type="checkbox"/> Earliest Deadline First                  |
| <input checked="" type="checkbox"/> Shortest Job First     | <input type="checkbox"/> Fair-Share                               |
| <input checked="" type="checkbox"/> Longest Job First      |   |

### Aufgabe 3 (Scheduling)

Prozess	CPU-Laufzeit [ms]	Ankunftszeit [ms]
A	5	0
B	7	3
C	2	5
D	6	7
E	1	10
F	5	18
G	4	24

1. Auf einem Einprozessorrechner sollen sieben Prozesse mit unterschiedlichen Ankunftszeiten verarbeitet werden. Skizzieren Sie die Ausführungsreihenfolge der Prozesse mit einem Gantt-Diagramm (Zeitleiste) für...

- Longest Remaining Time First und
- Shortest Remaining Time First.



2. Berechnen Sie die mittleren Laufzeiten der Prozesse.

	A	B	C	D	E	F	G
Longest Remaining Time First	18	13	24	6	20	5	4
Shortest Remaining Time First	5	18	2	7	1	8	6

Longest Remaining Time First  $\frac{18+13+24+6+20+5+4}{7} = 12,86 \text{ ms}$

Shortest Remaining Time First  $\frac{5+18+2+7+1+8+6}{7} = 6,71 \text{ ms}$

3. Berechnen Sie die mittleren Wartezeiten der Prozesse.

	A	B	C	D	E	F	G
Longest Remaining Time First	13	6	22	0	19	0	0
Shortest Remaining Time First	0	11	0	1	0	3	2

Longest Remaining Time First  $\frac{13+6+22+0+19+0+0}{7} = 8,57 \text{ ms}$

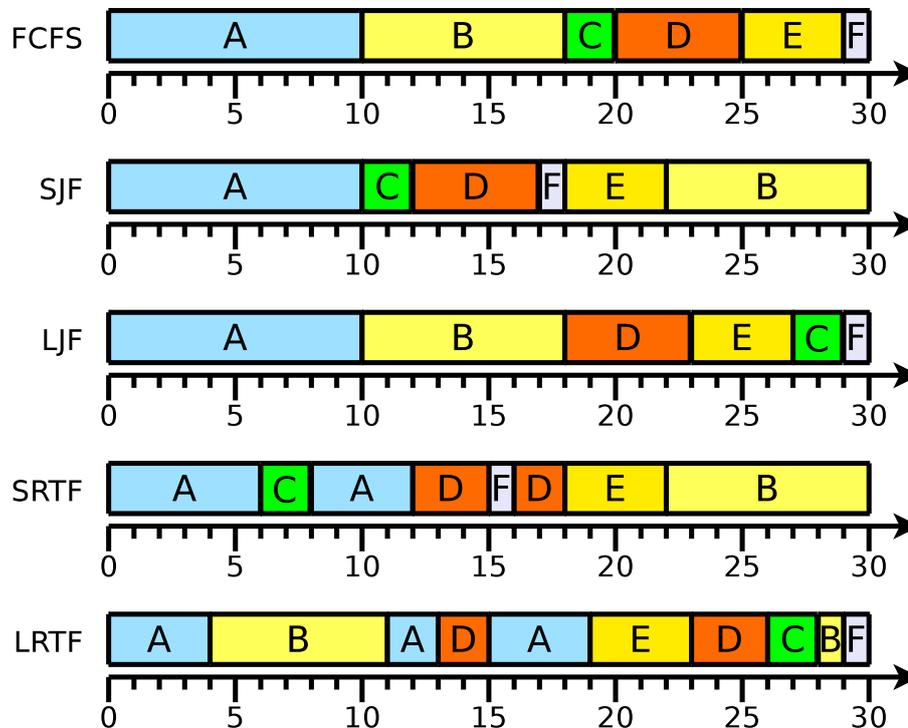
Shortest Remaining Time First  $\frac{0+11+0+1+0+3+2}{7} = 2,43 \text{ ms}$

## Aufgabe 4 (Scheduling)

Prozess	CPU-Laufzeit [ms]	Ankunftszeit [ms]
A	10	0
B	8	4
C	2	6
D	5	11
E	4	13
F	1	15

1. Auf einem Einprozessorrechner sollen sechs Prozesse mit unterschiedlichen Ankunftszeiten verarbeitet werden. Skizzieren Sie die Ausführungsreihenfolge der Prozesse mit einem Gantt-Diagramm (Zeitleiste) für...

- First Come First Served,
- Shortest Job First,
- Longest Job First,
- Shortest Remaining Time First und
- Longest Remaining Time First.



2. Berechnen Sie die mittleren Laufzeiten der Prozesse.

	A	B	C	D	E	F
First Come First Served	10	14	14	14	16	15
Longest Job First	10	14	23	12	14	15
Shortest Job First	10	26	6	6	9	3
Longest Remaining Time First	19	25	22	15	10	15
Shortest Remaining Time First	12	26	2	7	9	1

First Come First Served  $\frac{10+14+14+14+16+15}{6} = 13,8\bar{3}$  ms

Longest Job First  $\frac{10+14+23+12+14+15}{6} = 14,\bar{6}$  ms

Shortest Job First  $\frac{10+26+6+6+9+3}{6} = 10$  ms

Longest Remaining Time First  $\frac{19+25+22+15+10+15}{6} = 17,\bar{6}$  ms

Shortest Remaining Time First  $\frac{12+26+2+7+9+1}{6} = 9,5$  ms

3. Berechnen Sie die mittleren Wartezeiten der Prozesse.

	A	B	C	D	E	F
First Come First Served	0	6	12	9	12	14
Longest Job First	0	6	21	7	10	14
Shortest Job First	0	18	4	1	5	2
Longest Remaining Time First	9	17	20	10	6	14
Shortest Remaining Time First	2	18	0	2	5	0

$$\text{First Come First Served} \quad \frac{0+6+12+9+12+14}{6} = 8,8\bar{3} \text{ ms}$$

$$\text{Longest Job First} \quad \frac{0+6+21+7+10+14}{6} = 9,6\bar{7} \text{ ms}$$

$$\text{Shortest Job First} \quad \frac{0+18+4+1+5+2}{6} = 5 \text{ ms}$$

$$\text{Longest Remaining Time First} \quad \frac{9+17+20+10+6+14}{6} = 12,6\bar{6} \text{ ms}$$

$$\text{Shortest Remaining Time First} \quad \frac{2+18+0+2+5+0}{6} = 4,5 \text{ ms}$$

## Aufgabe 5 (Shell-Skripte)

1. Schreiben Sie ein Shell-Skript, das den Benutzer bittet, eine der vier Grundrechenarten auszuwählen. Nach der Auswahl einer Grundrechenart wird der Benutzer gebeten, zwei Operanden einzugeben. Die beiden Operanden werden mit der zuvor ausgewählten Grundrechenart verrechnet und das Ergebnis in der folgenden Form ausgegeben:

<Operand1> <Operator> <Operand2> = <Ergebnis>

```

1 #!/bin/bash
2 #
3 # Skript: operanden1.bat
4 #
5 echo "Bitte geben Sie den gewünschten Operator ein."
6 echo "Mögliche Eingaben sind: + - * /"
7 read OPERATOR
8 echo "Bitte geben Sie den ersten Operanden ein:"
9 read OPERAND1
10 echo "Bitte geben Sie den zweiten Operanden ein:"
11 read OPERAND2
12
13 # Eingabe verarbeiten
14 case $OPERATOR in
15 +) ERGEBNIS=`expr $OPERAND1 + $OPERAND2` ;;
16 -) ERGEBNIS=`expr $OPERAND1 - $OPERAND2` ;;
17 \*) ERGEBNIS=`expr $OPERAND1 \* $OPERAND2` ;;
18 /) ERGEBNIS=`expr $OPERAND1 / $OPERAND2` ;;
19 *) echo "Falsche Eingabe: $OPERATOR" >&2
20     exit 1
21     ;;
22 esac
23

```

```
24 # Ergebnis ausgeben
25 echo "$OPERAND1 $OPERATOR $OPERAND2 = $ERGEBNIS"
```

2. Ändern Sie das Shell-Skript aus Teilaufgabe 1 dahingehend, dass für jede Grundrechenart eine eigene Funktion existiert. Die Funktionen sollen in eine externe Funktionsbibliothek ausgelagert und für die Berechnungen verwendet werden.

```
1 #!/bin/bash
2 #
3 # Skript: operanden2.bat
4 #
5 # Funktionsbibliothek einbinden
6 . funktionen.bib
7
8 echo "Bitte geben Sie den gewünschten Operator ein."
9 echo "Mögliche Eingaben sind: + - * /"
10 read OPERATOR
11 echo "Bitte geben Sie den ersten Operanden ein:"
12 read OPERAND1
13 echo "Bitte geben Sie den zweiten Operanden ein:"
14 read OPERAND2
15
16 # Eingabe verarbeiten
17 case $OPERATOR in
18 +) add $OPERAND1 $OPERAND2 ;;
19 -) sub $OPERAND1 $OPERAND2 ;;
20 \*) mul $OPERAND1 $OPERAND2 ;;
21 /) div $OPERAND1 $OPERAND2 ;;
22 *) echo "Falsche Eingabe: $OPERATOR" >&2
23     exit 1
24     ;;
25 esac
26
27 # Ergebnis ausgeben
28 echo "$OPERAND1 $OPERATOR $OPERAND2 = $ERGEBNIS"
```

```
1 # Funktionsbibliothek funktionen.bib
2
3 add() {
4     ERGEBNIS=`expr $OPERAND1 + $OPERAND2`
5 }
6
7 sub() {
8     ERGEBNIS=`expr $OPERAND1 - $OPERAND2`
9 }
10
11 mul() {
12     ERGEBNIS=`expr $OPERAND1 \* $OPERAND2`
13 }
14
15 div() {
16     ERGEBNIS=`expr $OPERAND1 / $OPERAND2`
17 }
```

3. Schreiben Sie ein Shell-Skript, das eine bestimmte Anzahl an Zufallszahlen bis zu einem bestimmten Maximalwert ausgibt. Nach dem Start des Shell-Skripts, soll dieses vom Benutzer folgende Parameter interaktiv abfragen:

- Maximalwert, der im Zahlenraum zwischen 10 und 32767 liegen muss.
- Gewünschte Anzahl an Zufallszahlen.

```
1 #!/bin/bash
2 #
3 # Skript: random.bat
4 #
5 echo "Geben Sie den Maximalwert ein: "
6 read MAX
7 echo "Geben Sie an, wie viele Zufallszahlen Sie wünschen: "
8 read ANZAHL
9
10 for ((i=1; i<=${ANZAHL}; i+=1))
11 do
12     echo "Zufallszahl Nr. $i hat den Wert `expr $RANDOM % $MAX`"
13 done
```

4. Schreiben Sie ein Shell-Skript, das die folgenden leeren Dateien erzeugt:

image0000.jpg, image0001.jpg, image0002.jpg, ..., image9999.jpg

```
1 #!/bin/bash
2 #
3 # Skript: dateien_anlegen.bat
4 #
5 for i in {0..9999}
6 do
7     filename="image"$(printf "%04u" $i)".jpg"
8     touch $filename
9 done
```

5. Schreiben Sie ein Shell-Skript, das die Dateien aus Teilaufgabe 4 nach folgendem Schema umbenennt:

BTS\_Übung\_<JAHR>\_<MONAT>\_<TAG>\_0000.jpg  
BTS\_Übung\_<JAHR>\_<MONAT>\_<TAG>\_0001.jpg  
BTS\_Übung\_<JAHR>\_<MONAT>\_<TAG>\_0002.jpg  
...  
BTS\_Übung\_<JAHR>\_<MONAT>\_<TAG>\_9999.jpg