Übungsblatt 5

Aufgabe 1 (Schedulingverfahren)

1. Erklären Sie, warum in einigen Betriebssystemen mindestens ein Leerlaufprozess existiert.

2. Wie viele Leerlaufprozesse gibt es in einem modernen Linux-System?

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

4. Nennen Sie einen Nachteil von präemptivem Scheduling.

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

6. Beschreiben Sie wie Multilevel-Feedback-Scheduling funktioniert.

7. Welche Schedulingverfahren sind fair? (*Ein Schedulingverfahren ist „fair“, wenn jeder Prozess irgendwann Zugriff auf die CPU erhält.*)

   ☐ Prioritätengesteuertes Scheduling
   ☐ First Come First Served
   ☐ Round Robin mit Zeitquantum
   ☐ Earliest Deadline First
   ☐ Completely Fair Scheduler

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

   ☐ First Come First Served
   ☐ Round Robin mit Zeitquantum
   ☐ Completely Fair Scheduler
   ☐ Multilevel-Feedback-Scheduling

9. Beschreiben Sie, welches Problem es bei der Verwendung von (statischem) prioritätsgesteuertem Scheduling geben kann.

10. Die beiden Prozesse P_A (4 ms CPU-Rechenzeit) und P_B (26 ms CPU-Rechenzeit) sind zum Zeitpunkt 0 beide im Zustand `bereit` und sollen nacheinander ausgeführt werden.

    Schreiben Sie die fehlenden Werte in die Tabelle

    Hinweise:
    Rechenzeit ist die Zeit, die der Prozess Zugriff auf die CPU benötigt, um komplett abgearbeitet zu werden.
    Laufzeit = „Lebensdauer“ = Zeitspanne zwischen dem Anlegen und Beenden eines Prozesses = (Rechenzeit + Wartezeit)

Reihenfolge	**Laufzeit** P_A	P_B	**Durchschnittl. Laufzeit**	**Wartezeit** P_A	P_B	**Durchschnittl. Wartezeit**
P_A, P_B						
P_B, P_A						

11. Beschreiben Sie, welche Erkenntnisse sich aus den Werten, die Sie in der Tabelle eingetragen haben, herleiten lassen.

12. Nennen Sie die Scheduling-Methode, die moderne Windows-Betriebssysteme verwenden.

13. Nennen Sie die Scheduling-Methode, die moderne Linux-Betriebssysteme verwenden.

Aufgabe 2 (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>
   ```

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.

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.