

# Grundlagen - Betriebssysteme und Systemsoftware IN0009, WiSe 2023/24

#### Übungsblatt 3

6. November 2023 - 10. November 2023

Hinweis: Mit \* gekennzeichnete Teilaufgaben sind ohne Lösung vorhergehender Teilaufgaben lösbar.

### Aufgabe 1 Scheduling

Es seien die 3 Prozesse  $P_1$  bis  $P_3$  gegeben. Die jeweilige Ankunftszeit im System sowie die benötigte Rechenzeit entnehmen Sie bitte folgender Tabelle:

Prozess	Ankunftszeit	Benötigte Rechenzeit						
$P_1$	0	7						
$P_2$	5	3						
P <sub>3</sub>	2	4						

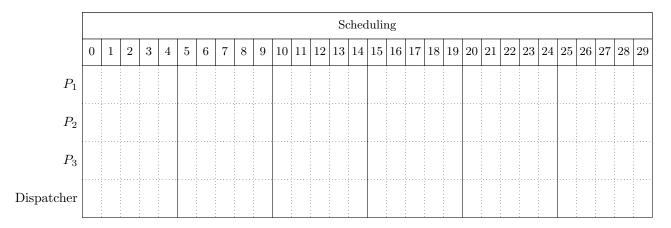
Die Aktivierung des Schedulers beanspruche in dieser Aufgabe keine Zeiteinheit. Sollte jedoch ein Prozesskontextwechsel nötig sein, so dauere dieser eine Zeiteinheit.

Modellieren Sie den Scheduler/Dispatcher in der untersten Zeile.

**Gantt-Diagramm:** Verwenden Sie die x-Achse des Diagramms für die zeitliche Dimension und die y-Achse für die Prozesse. Stellen Sie für jeden Prozess die rechenwillige Wartezeit mit einem - (beginnend mit der Ankunftszeit des Prozesses), und die Rechenzeit mit einem **X** dar. Berücksichtigen Sie die Prioriäten in Ihrem Diagramm.

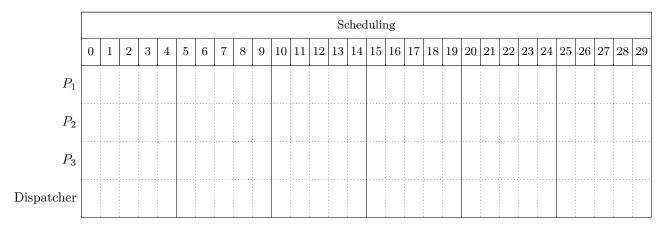
Skizzieren Sie unter diesen Annahmen den Ablauf der Prozesse in einem Gantt-Diagram für folgende Schedulingstrategien. **Hinweis**: Vernachlässigen Sie den initialen Kontextwechsel. Beginnen Sie im ersten Zeitslot mit dem ersten rechnenden Prozess. Sollten die Kriterien zur Auswahl des nächsten rechnenden Prozesses einmal nicht eindeutig sein, treffen Sie geeignete Annahmen und skizzieren Sie diese.

**a)**\* First-Come-First-Served (FCFS): Non-preemptive, Prozesse werden in der Reihenfolge ihrer Ankunftszeiten abgearbeitet.

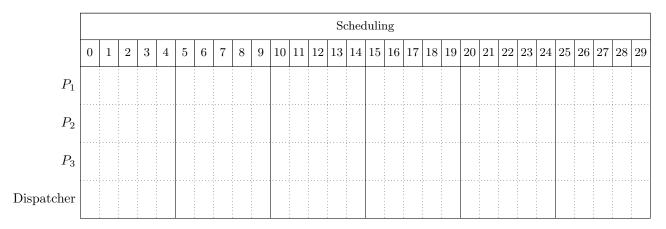




**b)**\* Shortest Remaining Time Next (SRTN): Preemptive, Auswahl des Prozesses mit der kürzesten verbleibenden Rechenzeit, Unterbrechungen erfolgen nur beim Eintreffen eines neuen Prozesses.



c) Round-Robin mit einem Zeitquantum von einer Zeiteinheit und zyklischer Abarbeitung der Prozesse (gleiche und nicht veränderbare Prioritäten, Sortierung nach der PID (=Index))



**d)**\* Round-Robin mit einem Zeitquantum von 2 Zeitenheiten und zyklischer Abarbeitung der Prozesse (gleiche und nicht veränderbare Prioritäten, Sortierung nach der PID (=Index))

	Scheduling																													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
$P_1$																														
$P_2$																														
$P_3$																														
Dispatcher																														



# Aufgabe 2 Priority Scheduling

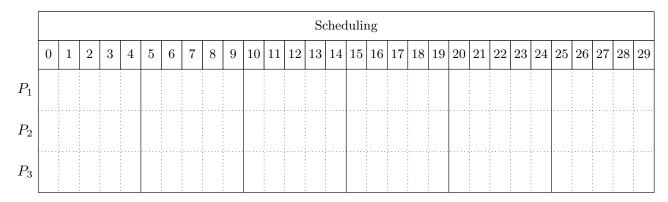
Ein Scheduler verwendet ein **priorisiertes** Round Robin Scheduling Verfahren (Priority Scheduling) mit dynamischen Prioritäten:

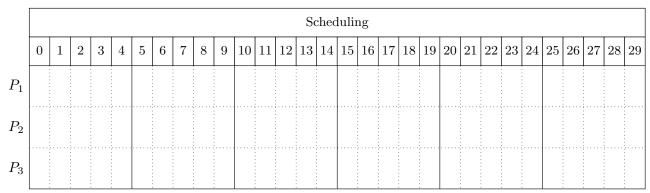
- Das Quantum beträgt q = 2 Zeiteinheiten.
- Jeder Prozess  $P_i$  besitzt eine Initialpriorität  $I_i$ .
- Im rechnenden Zustand wird die Priorität des Prozesses je nach 1 Zeiteinheit um 2 erniedrigt.
- Im rechenwilligen Zustand wird die Priorität des Prozesses alle 2 Zeiteinheiten um 1 erhöht.
- Prioritäten reichen von 0 bis 20, wobei 0 die niedrigste und 20 die höchste Prioritäten darstellen.
- In jedem Zeitquantum wird der Prozess mit der höchsten Priorität ausgewählt. Bei Gleichstand wird der Prozess mit dem niedrigsten Index vorgezogen.
- Bei einem I/O-Aufruf wird ein Prozess blockiert. Nehmen Sie an, dass die I/O-Daten dann nach 5 Zeiteinheiten bereitstehen. Anschließend ist der blockierte Prozess wieder rechenwillig.

Nun seien die drei Prozesse  $P_1$ ,  $P_2$  und  $P_3$  gegeben:

Prozess	Ankunftszeit	Benötigte Rechenzeit	Initiale Priorität	I/O-Interrupt nach n Rechenzeiten						
<i>P</i> <sub>1</sub>	0	6	10	einmalig: n = 3						
P <sub>2</sub>	2	6	9	kein I/O						
P <sub>3</sub>	0	8	14	kein I/O						

**a)**\* Zeichnen Sie die Ausführung von  $P_1$ ,  $P_2$  und  $P_3$  in das Gantt-Diagramm ein. Skizzieren Sie sowohl die Ausführung (×) als auch die Wartenzeiten (–). In einer Zeiteinheit, in der ein Prozess auf I/O-Daten wartet, wird die Zelle leer gelassen. Vernachlässigen Sie die Zeit, die durch den Scheduler und Dispatcher verbraucht wird.





- **b)** Berechnen Sie die mittlere Wartezeit  $\overline{W} = \frac{\sum_{i=1}^{n} w_i}{n}$  und die mittlere Verweilzeit  $\overline{V} = \frac{\sum_{i=1}^{n} v_i}{n}$  für dieses Szenario.
- c)\* Was ist der Vorteil von dynamischen Prioritäten gegenüber statischen Prioritäten?



# Aufgabe 3 Noch mehr C

**a)**\* Betrachten Sie die nachfolgende Implementierung einer Bibliotheksfunktion. Um welche Funktion handelt es sich? Was ist natürlichsprachlich die Abbruchbedingung?

```
void fct(char *s, const char *t) {
    while(*s++ = *t++);
}
```

b)\* Wie unterscheiden sich die folgenden Typdeklarationen? Es gilt: sizeof(void\*)==8 und sizeof(short)==2

```
      struct v1 {
      struct v2 {

      char a;
      struct v2 *o;

      short h;
      short h;

      struct v1 *o;
      char a;

      }
      Listing 1: Variante 1

Listing 2: Variante 2
```

Gehen Sie hierbei auch auf das Ergebnis des Operators sizeof ein.

**c)**\* Betrachten Sie folgendes C-Programm, welches die *n*-te harmonische Zahl  $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}$  berechnet. Beschreiben Sie etwaige Programmierfehler, die in diesem Programm gemacht wurden und erklären Sie kurz, wie sie sich auf das Programm auswirken.

```
#include <stdio.h>
2
    #include <stdlib.h>
3
    // Returns the first n harmonic numbers
4
5
    double* harmonic_numbers(unsigned int n) {
6
        double result[n];
7
        result [0] = 1.0;
8
9
        for (unsigned int i = 1; i < n; i++) {
            result[i] = result[i-1] + (1.0 / (double)(i+1));
10
11
12
        return result;
13
   }
14
15
    void print_harmonics(unsigned int n) {
16
        if (n == 0) return;
        double *result = harmonic_numbers(n);
17
18
        for (unsigned int i = 0; i < n; i++) {
            printf("%f\n", result[i]);
19
20
21
   // [...]
```