Arbeitsgruppe
Eingebettete Systeme und Betriebssysteme

Vorlesung Betriebssysteme


---

Übungsblatt 2

ab 19.11.2007

---

## Aufgabe 1

Was ist unter Diensten und Betriebsmitteln zu verstehen und wie werden sie verwendet?

## Aufgabe 2

Wie können Betriebssystemdienste in einem Betriebssystem aufgerufen werden?

## Aufgabe 3

Worin besteht der Unterschied zwischen Traps und Interrupts und wie arbeitet ein Interrupt?

## Aufgabe 4

Welche Probleme entstehen, wenn Interrupts über einen längeren Zeitraum deaktiviert (maskiert) werden?

## Aufgabe 5

Was versteht man unter einem kritischen Abschnitt? Erläutere das Problem am Beispiel eines Druckerspoolers.

## Aufgabe 6

Der Quellcode am Ende des Aufgabenblatts implementieren die Funktionen eines Ringpuffers. Mit `put()` kann ein Prozess ein Zeichen in den Puffer eintragen und mit `get()` holt er eines heraus. Allerdings sind diese Funktionen noch nicht vor Unterbrechungen oder zeitgleiche Ausführung durch einen anderen Prozess geschützt. Zeige an zwei Beispielen, dass dadurch Probleme auftreten können.

## Aufgabe 7

Der schon in Aufgabe  beschriebene Ringpuffer soll eingesetzt werden, um Tastencodes, die der Interrupthandler von der Tastatur abholt, zwischenzuspeichern, bis sie von einem Anwendungsprozesss, der auf dem gleichen Prozessor ausgeführt wird, benötigt werden. Wie können die Datenstrukturen des Ringpuffers in diesem Fall geschüzt werden?

## Aufgabe 8

Wodurch kann der Status eines Programms beschrieben werden?

## Aufgabe 9

Was ist Preemtion, wofür wird sie verwendet und was kann durch Preemtion erreicht werden? Gibt es Alternativen zur Preemtion?

## Aufgabe 10

Welche grundlegenden Ansätze zur Umsetzung eines Betriebssystemkerns gibt es und worin unterscheiden sich diese Ansätze (*Beschreibe Vor- und Nachteile, die sich aus den Unterschieden ergeben*)?

```
#define BUF_SIZE 10

char buffer[BUF_SIZE];
int empty = 1;
int full = 0;
int posr = 0;
int posw = 0;

char put (char c) {
  if (!full) {
    buffer[posw] = c;
    posw = (posw + 1) % BUF_SIZE;
    empty = 0;
    if (posw == posr) full = 1;
    return c;
  }
  return 0;
}

char get () {
  char c;
  if (!empty) {
    c = buffer[posr];
    posr = (posr + 1) % BUF_SIZE;
    full = 0;
    if (posr == posw) empty = 1;
    return c;
  }
  return 0;
}
```