Arbeitsgruppe
Eingebettete Systeme und Betriebssysteme

Vorlesung Betriebssysteme

Übungsblatt 2

Abgabetermin ab 22.11.2010

Aufgabe 1

Wie können Betriebssystemdienste in einem Betriebssystem aufgerufen werden?

Aufgabe 2

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

Aufgabe 3

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

Aufgabe 4

Was versteht man unter einem kritischen Abschnitt? Erläutere das Problem am Beispiel einer Queue.

Aufgabe 5

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.

a) Die Funktionen des Ringpuffers sind noch nicht vor Unterbrechungen oder zeitgleiche Ausführung durch einen anderen Prozess geschützt. Zeige an zwei Beispielen, dass dadurch Probleme auftreten können.

b) Der 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 6

Wodurch kann der Status eines Programms beschrieben werden?

Aufgabe 7

Was ist Preemption, wofür wird sie verwendet und was kann durch Preemption erreicht werden? Gibt es Alternativen zur Preemption? Diskutieren Sie Vor- bzw. Nachteile der Preemtion.

Aufgabe 8

Worin besteht der Unterschied zwischen Routinen, Coroutinen, Threads und Prozessen und was muss bei einem Coroutinenwechsel geschehen?

Aufgabe 9

Nennen sie typische Funktionen des Betriebssystemkerns im Hinblick auf Prozessverwaltung in einem Unix System!

Aufgabe 10

Welche Ereignisse können zu einem Prozesswechsel in einem Unix Betriebssystem führen?

```
#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;
}
```