Vorlesung Betriebssystemarchitektur SS 2006
Aufgabenblatt 2 vom 18. Mai 2006
(Vorstellung der Lösungen bei den Tutoren bis zum 31. Mai 2006)

Aufgabe 2.1: (20 Punkte)
Untersuchen Sie Eigenschaften des Betriebssystems Windows XP mit Hilfe des Performance Monitors (`perfmon`)! Beobachten Sie die Zähler für *privileged time, interrupts/second, processor time* und *user time*!

- Geben Sie Auskunft über die aktuelle Messung. Welche Charakteristiken des Betriebssystems werden durch diese Werte erkennbar?
- Erklären Sie die Änderungen die durch das Ausführen einer Shell (`cmd.exe`), Maus- oder Tastaturereignisse auftreten. Versuchen Sie die gemessenen Werte zu maximieren.

Aufgabe 2.2: (20 Punkte)
Untersuchen Sie die Eigenschaften des Betriebssystem Linux! Lesen Sie dazu die Manualseiten: *top(1)* und *ps(1)*!

- Woran erkennt man die Prozesse die als Dämonen laufen?
- Wo lassen sich die von *ps* angezeigten Informationen im Dateisystem finden?
- Welche Bibliotheken werden von ihrer aktuellen Shell benutzt?

Aufgabe 2.3: (20 Punkte)
Beantworten Sie folgende Fragen und erläutern Sie die Antworten Ihrem Tutor.

- Was ist „*busy waiting*“?
- Welche anderen Möglichkeiten „zu warten“ gibt es in einem Betriebssystem?
- Kann „*busy waiting*“ immer vermieden werden?
- An welcher Stelle verwendet der Windows-Kernel „*busy waiting*“?

Aufgabe 2.4: (20 Punkte)
In der Vorlesung wurde der „Bakery“-Algorithmus zur Realisierung von wechselseitigem Ausschluss vorgestellt. Implementieren Sie den Algorithmus innerhalb des bereitgestellten Programmrahmens unter Windows!

- `bakery.h` enthält die Interfacedeklarationen der zu implementierenden Funktionen.
- `main.c` enthält ein Testprogramm.

Erläutern Sie Ihrem Tutor Ihre Implementierung und führen Sie Ihr Programm vor!
Erstellen Sie ein Makefile für **`nmake`** mit dem Ziel **`bakery.exe`**. Verpacken Sie alle Source-Code Dateien (inklusive Makefile) in eine ZIP-Datei (blatt2.zip). Diese ZIP-Datei muss mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden.

Aufgabe 2.5: (20 Punkte)
Zwei parallel ablaufende Prozesse (der Erzeuger und der Verbraucher) verwenden einen gemeinsamen, begrenzten Speicherbereich. Der Erzeuger schreibt Daten in den gemeinsamen Speicher, der Verbraucher entnimmt Daten.
Kann der Erzeuger kein neues Datenelement speichern, verwendet er die `sleep`-Funktion und wartet, bis er vom Verbraucher wieder aufgeweckt wird nachdem dieser ein Datenelement entfernt hat.
Analog wartet der Verbraucher wenn keine Datenelemente vorhanden sind bis er vom Erzeuger geweckt wird.

Nachfolgend sind (Pseudo-Code) Implementierungen für den Erzeuger und den Verbraucher dargestellt:

```
#define N 100     // Puffergroesse

int count = 0;    // Anzahl der Puffereintraege

void producer(void) {

  int item;

  while (TRUE) {
    produce_item(&item);  // erzeuge naechsten Eintrag
    if (count == N)       // Puffer voll? -> schlafen
      sleep();
    insert_item(item);    // schreibe Eintrag in Puffer
    count = count+1;      // Anzahl der Eintraege erhoehen
    if (count == 1)       // war Puffer leer? -> consumer wecken
      wakeup(consumer);
  }
}

void consumer(void) {

  int item;

  while (TRUE) {
    if (count == 0)       // Puffer leer? -> schlafen
      sleep();
    remove_item(&item);   // Eintrag aus Puffer lesen
    count = count-1;      // Anzahl der Eintraege vermindern
    if (count == N-1)     // war Puffer voll? -> producer wecken
      wakeup(producer);
    consume_item(item);   // Eintrag verarbeiten
  }
}
```

Analysieren Sie das (fehlerhafte) Programmfragment und beantworten Sie Ihrem Tutor folgende Fragen:

- Da der Erzeuger- und der Verbraucherprozess parallel ablaufen, können sie vom Scheduler unterbrochen werden. Finden Sie zwei unterschiedliche parallele Programmabläufe, die dazu führen, dass sowohl der Erzeuger- als auch der Verbraucherprozess mittels `sleep` warten!
- Markiert man den gesamten „while(TRUE)“-Schleifenrumpf als kritischen Abschnitt wird dieses Problem nicht gelöst. Warum?
- Beschreiben und begründen Sie, wie dieses Problem mit Hilfe von Semaphoren gelöst werden kann!