Universität Konstanz
Fachbereich Informatik & Informationswissenschaft
Prof. Dr. Marcel Waldvogel / Tino Klingebiel

Betriebssysteme
WS 2016/2017

5. Übungsblatt

Abgabe: 24. November 2016, 4 Uhr

Aufgabe 1: Allgemein — 8 Punkte

1. Welches sind die zwei getrennten und potenziell unabhängigen Merkmale, die durch das Prozesskonzept verkörpert werden? (1)
2. Welche Ressourcen werden in der Regel gemeinsam von allen Threads eines Prozesses genutzt? (1)
3. Wird ein Prozess beendet und es existieren noch laufende Threads dieses Prozesses, können diese weiterlaufen? (1)
4. Nennen Sie je zwei Vor- und Nachteile von Benutzer-Threads gegenüber Kernel-Threads. (2)
5. Gibt es einen Unterschied zwischen Thread- und Prozesswechseln bezüglich ihrem (Zeit)Aufwand? Geben Sie zwei mögliche Gründe für das von Ihnen vermutete Verhalten an. (3)

Aufgabe 2: Race Condition — 4 Punkte

Betrachten Sie das folgende Programm:

```
1  const int n=50;
2  int anzGesamt;
3
4  void total(){
5    int count;
6    for(count=0; count < n; count++) {
7      if (count % 2 == 0)
8        anzGesamt++;
9      else
10       anzGesamt--;
11   }
12 }
13
14 int main(void) {
15   anzGesamt=0;
16   parbegin(total(), total());
17   write(anzGesamt);
18   return 0;
19 }
```

(a) Bestimmen Sie die korrekte untere und obere Grenze des Endwerts der Ausgabe der gemeinsamen Variablen *anzGesamt* durch dieses nebenläufige Programm. Begründen Sie ihre Ergebnisse. Nehmen Sie dafür an, dass Prozesse mit jeder beliebigen relativen Geschwindigkeit ausgeführt werden können und dass ein Wert nur verändert werden kann, nachdem er mit Hilfe eines separaten Maschinenbefehls in ein Register geladen wurde.

(b) Nehmen Sie an, dass unter Anwendung der Voraussetzungen in Teil (a) eine willkürliche Anzahl dieser Prozesse parallel ausgeführt werden darf. Welche Auswirkungen hat diese Änderung auf die Bandbreite der Endwerte von *anzGesamt*? Begründen Sie ihre Ergebnisse.

Anmerkung:
parbegin($P_1, P_2, \ldots, P_n$) bedeutet: Suspendiere die Ausführung des Main-Programms, starte die Prozeduren $P_1, P_2, \ldots, P_n$ nebenläufig und führe das Main-Programm weiter aus, nachdem alle $P_1, P_2, \ldots, P_n$ terminiert sind.

Aufgabe 3: User-Level-Threads **3 Punkte**

Betrachten Sie ein System, in dem die Threads vollständig im Benutzeradressraum realisiert sind und das Laufzeitsystem einmal in der Sekunde ein Zeitsignal (Timer-Interrupt) bekommt. Nehmen Sie an, dass ein Zeitsignal auftritt während ein Thread im Laufzeitsystem ausgeführt wird. Ein mögliches Problem in diesem Szenario wäre, daß das Laufzeitsystem gerade dabei sein könnte die Schedulingqueues zu bearbeiten, wenn der Interrupthandler des Zeitsignals seine Arbeit aufnimmt. Warum? Können Sie eine Lösung vorschlagen?

Hinweis: Solche Interrupts werden meistens für Schedulingoperationen genutzt, so daß jedem Thread/ Prozess nur ein fester Zeitabschnitt zugestanden wird, bevor er wieder hinten in die Queue einsortiert wird und ein weiterer den Zuschlag bekommt.

Aufgabe 4: Threadkontrollblock **5 Punkte**

Im Anhang befindet sich eine Übersicht mit typischen Elementen eines Prozesskontrollblocks eines ungethreadeten Betriebssystems. Welche dieser Elemente sollten in einem Multithreadingsystem zu einem Threadkontrollblock, und welche zum Prozesskontrollblock gehören?

Process Identification

Identifiers
Numeric identifiers that may be stored with the process control block include

- Identifier of this process
- Identifier of the process that created this process (parent process)
- User identifier

Processor State Information

User-Visible Registers
A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

Control and Status Registers
These are a variety of processor registers that are employed to control the operation of the processor. These include

- **Program counter:** Contains the address of the next instruction to be fetched
- **Condition codes:** Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)
- **Status information:** Includes interrupt enabled/disabled flags, execution mode

Stack Pointers
Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.

Process Control Information

Scheduling and State Information
This is information that is needed by the operating system to perform its scheduling function. Typical items of information:

- **Process state:** Defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
- **Priority:** One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable).
- **Scheduling-related information:** This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
- **Event:** Identity of event the process is awaiting before it can be resumed.

Data Structuring
A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent–child (creator–created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

Interprocess Communication
Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.

Process Privileges
Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

Memory Management
This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.

Resource Ownership and Utilization
Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.