Übung 02

Betriebs- und Kommunikationssysteme
Sommersemester 2021
Prof. Dr.-Ing. Jochen Schiller

Abgabe 07.05.2021, 10:00

Aufgabe 1: Syscalls & Multitasking

Beantworten Sie Fragen und geben Sie ggf. Ihre Quellen an.

1. Bis ein Hardware-Interrupt über einen Interrupt-Controller tatsächlich an die CPU gemeldet wurde ist bereits eine (sehr geringe!) Verzögerung eingegangen. In manchen Systemen ist dies nicht tolerierbar (Airbag o. Ä.), wie könnte man das Problem umgehen und warum ist diese Methode schneller?

2. Trotz aller Optimierungen sind auch Systemaufrufe durchaus aufwändig, weshalb viele verbreitete Betriebssysteme versuchen, bestimmte System-Funktionalität ohne Systemaufrufe bereitzustellen. Linux nutzt dazu „vDSO“ [1]. Dies wird dadurch ermöglicht, dass die Daten – bspw. die (Uhr-)Zeit – direkt als sich stetig verändernde globale Variablen in jedem Nutzerprozess eingeblendet sind.

   Warum bietet das sich nicht für alle Systemaufrufe an bzw. was unterscheidet die in dem verlinkten Artikel genannten Systemaufrufen von anderen?

3. Früher haben viele Betriebssysteme kein „echtes“ (präemptives) Multitasking implementiert, sondern kooperatives Multitasking. Damit also zwischen Programmen gewechselt werden konnte, mussten diese spezielle Systemaufrufe tätigen (hoffentlich häufig genug!) um die Kontrolle von sich heraus abzugeben und andere Prozesse laufen zu lassen.

   Heutzutage ist auf dem PC präemptives Multitasking vorherrschend, hier unterbricht das Betriebssysteme – oft u. A. durch einen von einem Timer ausgelösten Interrupt – die laufenden Prozesse regelmäßig und wechselt diese durch.

   Nennen Sie einen *Vorteil* von kooperativem Multitasking.

4. Für präemptives Multitasking nutzen wir Verwaltungsstrukturen wie den **Process Controll Block** (PCB), die einen Schnappschuss eines Prozesses zu einem bestimmten Zeitpunkt darstellen. Welche Informationen müssen in dem PCB gespeichert werden, wann wird auf diesen zugegriffen und dieser aktualisiert?

5. Während ein Prozess eine Programminstanz ist, repräsentiert ein Thread eine Ausführung eines Prozesses. Somit hat jeder Prozess zumindest einen Thread. Jedoch ist auch möglich Prozesse mit mehreren Threads laufen zu lassen, also innerhalb einer Programminstanz.

   Wann ist es sinnvoll eher mehrere Threads zu starten an Stelle von mehreren Prozessen um ein paralleles Problem zu bearbeiten? Denken Sie besonders an die Inter-Process-Communication (IPC) im Vergleich zur Abstimmung zwischen zwei Threads!

[1] `https://lwn.net/Articles/615809/`

Aufgabe 2: `bc(1)` mit ISO-C Streams

Die bisherigen Funktionen `open`, `read` und `write` sind Teil von POSIX und dem Low-Level Interface um auf UNIX-oiden System Ein- und Ausgabe zu realisieren.

Der C-Standard spezifiziert ein zusätzliches abstrakteres Interface um ausschließlich auf Dateien zuzugreifen (im Gegensatz zu `read` / `write` die auch bspw. für IPC genutzt werden können). Dieses Interface arbeitet mit File-Streams an Stelle von File Deskriptoren und die Funktionen sind im generellen etwas „bequemer“ zu benutzen.

Implementieren Sie mit diesen Funktionen (aus dem Header `stdio.h`) einen **basic calculator** in Anlehnung an das Programm `bc(1)`. Dieses soll – wenn ohne Argumente aufgerufen – wieder auf `stdin` einlesen. Andernfalls soll es durch die Argumente iterieren, diese als Dateien öffnen und von diesen einlesen.

In jedem Fall soll das Programm die Eingabe nach dem Schema Operand-Operator-Operand parsen, wobei die Operanden Ganzzahlen in Basis 10 sind und als Operatoren einfache Integer-Arithmetik (`+`, `-`, `*`, `/`) unterstützt werden soll. Eine valide Eingabe wäre also bspw. `4 * 8`.

Nach jedem erfolgreichen „matchen“ eines Ausdrucks soll dieser berechnet werden und das Ergebnis ausgegeben werden. Bei der interaktiven Benutzung könnte das bspw. so aussehen:

```
$ ./bc
4 * 8
32
3 / 3
1
1 + 8
9
0 * 9
0
```

Fehler wie nicht-existente Dateien sollen in dieser Aufgabe auch abgefangen werden!

Hinweis Nutzen Sie zum Parsen die Funktionen aus der `scanf`-Familie und zur Ausgabe `printf`. Sie finden die Dokumentation dazu in den Man-Pages auf Sektion 3.