Übungsblatt 10

Ü 10.1 Geräte (devices) in Linux

Beantworten Sie die folgenden Fragen und Aufgaben:

1) Was versteht man unter einer Gerätedatei in Linux?
2) Listen Sie alle Gerätedateien in Ihrem Linux System auf.
3) Was ist der Unterschied zwischen Character- und Block-Devices?
4) Wie viele Character-Devices bzw. wie viele Block-Devices sind auf Ihrem Rechner im Einsatz?
5) Was versteht man unter den Major- und Minor-Nummern von Gerätedateien und wozu werden diese benutzt?
6) Bestimmen Sie die Major- und Minor-Nummern von /dev/random und /dev/null. Handelt es sich bei den Geräten um Character- oder Block-Devices?

Ü 10.2 Threads - Barrieren Bingo

Machen Sie sich mit den folgenden Funktionen vertraut: `pthread_barrier_init`, `pthread_barrier_wait`, `pthread_testcancel`, `pthread_cancel`

Implementieren Sie ein einfaches, nicht-interaktives Bingo-Spiel mittels Threads (verwenden Sie `stub10_2.c` als Basis):

- Es gibt N Spielende, die zu Beginn je ein Array mit 10 unterschiedlichen Zufallszahlen von 1 bis 50 befüllen.
- Ein Showmaster zieht maximal einmal pro Sekunde eine Zahl aus diesem Wertebereich, wobei jede Zahl nur einmal gezogen werden kann.
- Die Spielenden überprüfen, ob die gezogene Zahl in ihrem Array enthalten ist. Sollte dies der Fall sein, wird die Zahl mit 0 überschrieben.
- Nachdem alle Spielenden überprüft haben, ob die Zahl vorhanden ist, gibt ein Monitor-Thread den aktuellen Spielstand auf der Konsole aus: es werden die Zahlen jedes Spielenden ausgegeben, die noch nicht gezogen wurden. Verwenden Sie für die Synchronisierung der Spielenden und des Monitor-Threads `pthread_barrier_wait`.
- Wurden alle Zahlen eines Spielenden gezogen, hat dieser gewonnen. Er beendet alle anderen Threads mit `pthread_cancel` und beendet sich schließlich selbst.

- Stellen Sie sicher, dass der Showmaster erst dann wieder eine Zahl ziehen kann, wenn alle SpielerInnen die letzte Zahl überprüft haben und der Monitor-Thread den aktuellen Spielstand ausgegeben hat.

Ü 10.3 Stack und Heap Performance

Recherchieren Sie, wann Ihr Programm Speicher am Stack oder am Heap anfordert. Schreiben Sie ein C-Programm welches zwei Funktionen implementiert:

```
StackAllocationTime();

HeapAllocationTime();
```

Beide Funktionen sollen die Ausführungszeit in die Konsole loggen. Natürlich liefert eine einmalige Speicher allokation nicht representative Werte. Somit sollten zumindest **10^6** Allokationen durchgeführt werden.

Mögliche Ausgabe:

```
Time of 1000000 stack allocations: 0.002520 seconds

Time of 1000000 heap allocations: 0.029184 seconds
```

Ü 10.4 Zombies

Recherchieren Sie, was Zombie-Prozesse in Unix-Betriebssystemen sind und wie diese erzeugt werden. Dokumentieren Sie, wie im `struct task_struct` Zombie-Prozesse gekennzeichnet werden und schreiben Sie ein einfaches C-Programm `create_zombie.c`, welches einen Zombie-Prozess erzeugt und danach zumindest 5 Minuten lang weiterläuft.

Starten Sie Ihr Programm nach dem Kompilieren mehrfach, beispielsweise mit dem Befehl `./create_zombie &` , damit die Eingabe im Terminal nicht blockiert ist. Mit dem Befehl `ps aux | grep 'Z'` sollten Sie sehen können, ob Ihre Zombies aktiv sind.

Ü 10.5 Zombie Monitor Kernel Modul

Schreiben Sie ein Linux-Kernel-Modul mit dem Namen `zombie_monitor`, das (einmalig in `init_module`) Informationen über alle Zombie-Prozesse ins Kernel-Log schreibt. Das Modul soll folgende Informationen über alle Zombie Prozesse ausgeben:

- Prozess ID

- Parent-Prozess ID
- Befehl, mit welchem der Prozess gestartet wurde
- Exit-Status des Prozesses
- User-ID
- CPU-Time im Userspace
- CPU-Time im Kernelspace

Testen Sie Ihr Modul indem Sie einen Zombie-Prozess erzeugen und anschließend das Modul `zombie_monitor` laden.

> Tip: Da der Linux-Kernel alle Prozesse in einer doppelt verketteten Liste (Task-List) verwaltet, auf deren Kopf die Variable `struct task_struct init_task` zeigt (definiert in `signal.h`), sollten Sie die Makros `next_task()` oder `for_each_process()` für die Navigation durch die Task-List verwenden. Beachten Sie, dass der letzte Eintrag der Task-List wieder auf den ersten Eintrag zeigt.