Übungen zur Vorlesung
Betriebssysteme II
Sommersemester 2007

AG BS
Jan Peleska
Christof Efkemann

Blatt 2

CSP-Kanalkommunikation im Kernel

In dieser Übung sollen eine Reihe von Systemaufrufen entwickelt werden, die es mehreren Prozessen erlauben, synchron über einen Kanal zu kommunizieren. Diese sollen gemäß der Semantik von CSP-Kanalkommunikation arbeiten.

```
int bs2_channel_open(char *name, size_t length);
int bs2_channel_read(int id, char *data);
int bs2_channel_write(int id, char *data);
```

Mit `bs2_channel_open` öffnet ein Prozess einen Kommunikationskanal mit dem Namen `name`. Ist dieser noch von keinem anderen Prozess geöffnet, wird er neu angelegt und die Größe der zu kommunizierenden Nachrichten auf `length` festgelegt. Existiert der Kanal hingegen bereits, schlägt der Aufruf fehl, falls sich `length` von der bereits festgelegten Nachrichtengröße unterscheidet. Im Erfolgsfall gibt der Aufruf eine (positive) Kanal-ID zurück.
Mit `bs2_channel_read` signalisiert ein Prozess, dass er zum Lesen einer Nachricht auf dem Kanal mit der ID `id` bereit ist – `bs2_channel_write` hingegen signalisiert die Bereitschaft zum Schreiben.
Im Falle von `bs2_channel_read` zeigt `data` auf einen Puffer, der groß genug sein muss, um eine Nachricht der Größe des Kanals aufzunehmen. Bei `bs2_channel_write` zeigt `data` auf die zu sendende Nachricht.
Da die Kommunikation über alle Prozesse, die einen Kanal geöffnet haben, synchronisiert ablaufen soll, blockieren diese beiden Aufrufe, bis alle diese Prozesse sich entweder im `read`- oder `write`-Aufruf befinden.
Falls zwei Prozesse gleichzeitig auf den Kanal schreiben wollen, so gelingt dies nur, wenn sie die gleichen Daten schreiben. Andernfalls führt dies wie bei CSP zu einem Deadlock.
Falls alle Prozesse, die über den Kanal kommunizieren, lesen wollen, und keiner schreibt, führt dies ebenfalls zum Deadlock.
Jeder Prozess kann zu einem Zeitpunkt nur zur Kommunikation über höchstens einen Kanal bereit sein, das heißt es existiert (noch) keine Choice-Operation.

Aufgabe 1: Datenstrukturen (10%)

Überlegen Sie sich geeignete Datenstrukturen, mit denen Sie die Kanalkommunikation im Kernel verwalten wollen. Was muss hierbei alles berücksichtigt werden? Geben Sie eine kurze Erläuterung Ihrer Entscheidungen.

Aufgabe 2: Implementation der Systemaufrufe (60%)

Implementieren Sie die oben genannten Systemaufrufe nach dem gleichen Prinzip wie in Übung 1. Benutzen Sie hierbei

- `kmalloc`, `kfree` zur dynamischen Speicherverwaltung
- `mutex_lock`, `mutex_unlock` für gegenseitigen Ausschluss
- `wait_event_interruptible`, `wake_up_all` um Prozesse schlafen zu legen und aufzuwecken

Stellen Sie außerdem sicher, dass die an einer Kanalkommunikation teilnehmenden Prozesse (auch im Falle eines Deadlocks) noch durch Signale abgebrochen werden können. Sorgen Sie dafür, dass alle angeforderten Ressourcen wieder freigegeben werden, sobald der letzte an der Kommunikation teilnehmende Prozess beendet wird.

Aufgabe 3: Testprogramme (30%)

Schreiben Sie 5 kurze Testprogramme, mit denen die Kanalkommunikation getestet werden kann:

1. Leser-Programm: Liest Text aus einer Datei und schreibt ihn stückweise in seinen Ausgabekanal.
2. Uppercase-Programm: Liest aus seinem Eingabekanal Nachrichten des Leser-Programms, ändert jeden Kleinbuchstaben einer Nachricht in den entsprechenden Großbuchstaben und schreibt das Ergebnis in seinen Ausgabekanal.
3. ROT13-Programm: Liest aus seinem Eingabekanal Nachrichten des Leser-Programms, wendet auf alle Buchstaben den ROT13-Algorithmus an und schreibt das Ergebnis in seinen Ausgabekanal.
4. Lowercase-Programm: Liest aus seinem Eingabekanal Nachrichten des Leser-Programms, ändert jeden Großbuchstaben einer Nachricht in den entsprechenden Kleinbuchstaben und schreibt das Ergebnis in seinen Ausgabekanal.
5. Schreiber-Programm: Liest aus seinem Eingabekanal Nachrichten und schreibt diese in eine Datei.

Die zu benutzenden Kanalnamen sollen den Programmen als Kommandozeilenparameter übergeben werden. Alle nicht weiter erwähnten Zeichen werden von den mittleren drei Programmen unverändert weitergegeben.
Starten Sie drei Schreiber-Programme, dann jeweils ein Uppercase-, ROT13- und Lowercase-Programm, die mit jeweils einem der Schreiber kommunizieren, und anschließend ein Leser-Programm, das mit den drei Konvertierungsprogrammen über einen gemeinsamen Kanal kommuniziert.

Abgabe: Bis Mittwoch, 30.05.2007, in der Vorlesung.

Sowohl bei der schriftlichen Lösung als auch im Source-Code die Namen aller Gruppenmitglieder nicht vergessen!