Bearbeitung des 4. Aufgabenblattes 2. Januar 2023

Autoren: Elena Frank, Stefan Vesovic

Aufgabe 4-1 (Nachrichten unterschiedlicher Länge)

- Ein Kommunikationsobjekt muss die Daten oder einen Zeiger auf diese Daten speichern. Zusätzlich dazu muss auch noch die Länge der Daten gespeichert werden. Wenn die Werte mit "call by value"übertragen werden, hängen die Auswirkungen davon ab, wo das Kommunikationsobjekt befindet. Wenn es im Adressraum des sendenden Prozesses sich befindet, gibt es keine Auswirkungen auf das Betriebssystem, da der Sender dafür selbst zuständig ist. Im Falle, dass es sich im Kernel befindet, muss für eine komplette Speicherung der Daten dynamisch Speicher alloziert werden. Dabei muss auf die Größe der Daten geachtet werden, da diese den ganzen Speicher besetzen können, wenn diese groß genug wird. Als Letztes kann das Objekt sich auch im Adressraum des empfangenden Prozesses befinden. Hier muss das Betriebssystem darauf achten, das die Daten nicht außerhalb Ihres eigenen Adressraumes schreiben und damit andere Sachen überschreiben.
- In dem Falle muss beim Empfänger geprüft werden, dass die Längenangaben stimmen und wirklich nur so viele Daten kopiert werden wie auch vom Empfänger verarbeitet werden können. Wenn der Sender mehr sendet, muss dieser entweder blockiert werden, bis wieder empfangen werden kann oder die Operation abgebrochen und zurückgegeben, wie viele Daten kopiert wurden. Der Sender weiß dadurch, was schon gesendet wurde und was nicht und kann dann versuchen, noch einmal in einem zweiten Anlauf die restlichen Daten zu übertragen. Es kann auch eine Maximalgröße für die Daten festgelegt werden und dann Speicherbereiche dieser Größe statisch zugewiesen werden. Dadurch kann jedoch viel platz verschwendet werden wenn die Daten sehr klein sind im vergleich zu der Maximalgröße.

Aufgabe 4-2 (Kontextwechsel und präemptives Multitasking)

Git: `https://github.com/elenaf9/WS2022-betriebssysteme`
Branch: `assignment-4`

- Idle thread läuft im busy-wait wenn es keine Threads zum Ausführen gibt.
- Scheduler für den Kontextwechsel implementiert über den *Periodic Interval Timer*.
  Zeitscheibenlänge für Threads konfigurierbar durch Ändern des `ST_INTERVAL` in driver/system_timer.c
- Threads werden im Rounnd-Robin gescheduled.
- Anzahl der Threads configurierbar via `NUM_THREADS` in system/threads.c.