| h_da | Embedded SW-Praktikum | |
|---|---|---|
| Termin 5 | Interprozesskommunikation | Prof. Akelbein |

## Teil 1: Vorbereitung

Datenaustausch ist der wichtigste Aspekt eines Programmablaufs – dies wird jedoch erschwert, wenn der Programmablauf von Daten abhängig ist, die von anderen Prozessen (zur Laufzeit) erzeugt werden. In diesem Praktikumstermin widmen wir uns der Inter-Process-Communication (abk. IPC) und erarbeiten ein Beispiel.
Zur Vorbereitung für den Programmierteil beantworten Sie folgende Fragen vor dem Praktikumstermin:

1. Welche Arten des Datenaustauschs zwischen Prozessen kennen Sie bereits? Welche Möglichkeiten bietet uns Linux an?
2. Welchen Unterschied haben Sie beim Datenaustausch zwischen
   Thread ← Daten → Thread und Prozess ← Daten → Prozess?
   Was erschwert diesen für Prozesse?
3. Warum bündeln wir unseren Code nicht in mehrere Threads im gleichen Prozess, um den Austausch der Daten zwischen Prozessen zu vermeiden? Was ist der Vor- und Nachteil von IPC?
4. Machen Sie sich mit dem gegebenen Code vertraut, welches Ihnen bei diesem Termin beiliegt. Sie sollten in der Lage sein, die Funktionsweise der verwendeten Klassen CMessage, CBinarySemaphore und CCommQueue zu verstehen und einzusetzen. Lesen Sie die entsprechende Abschnitte des Buchs „Automotive Embedded Systeme“.

## Teil 2: Producer / Consumer

Sie erarbeiten ein IPC-Beispiel, welches den Datenaustausch über den Shared Memory vollzieht. Hierzu nutzen Sie eine sogenannte MessageQueue, welche Sie im Shared Memory erstellen. Machen Sie die Abläufe über Terminalausgaben sichtbar.

1. Schreiben Sie ein Programm, dass ein Kind-Prozess mittels fork() erzeugt. Vergewissern Sie sich, dass Ihr Kind-Prozess startet, indem Sie jeweils am Anfang des Eltern- und Kind-Prozess‘ eine Ausgabe schalten.
2. Öffnen Sie ein (Posix-) Shared Memory und mappen dieses jeweils in den Adressraum des Eltern- und Kind-Prozesses. Eine Übersicht über die hierzu nötigen Funktionsaufrufe finden Sie durch den Konsolenbefehl:
   *$ man shm_overview*
3. Welcher Prozess ist verantwortlich für das Erzeugen des Shared Memory? Welcher Prozess muss diesen wieder schließen?
4. Erzeugen Sie nun jeweils ein CBinarySemaphore- und *CCommQueue*-Objekt im Shared Memory. Hierzu verwenden Sie das sogenannte Placement-New, um die Objekte im nun gemappten Shared Memory zu erstellen (anstatt auf dem Heap/Stack). Der Semaphor dient dazu, einem blockierten Thread zu signalisieren, dass die Queue nicht mehr leer ist und ausgelesen werden kann.
5. Erstellen Sie nun im Eltern-Prozess beliebig viele Nachrichten, indem Sie ein CMessage-Objekt erstellen und dieses mit sinnvollen Daten füllen. Eine CMessage sollte anhand ihrer Daten **eindeutig** identifizierbar sein. Fügen Sie diese Messages nun in die Queue ein.
6. Im Kind-Prozess lesen Sie nun sämtliche Nachrichten aus der Queue aus, bis entweder eine von Ihnen definierte Anzahl von Nachrichten gelesen oder eine andere Abbruchvoraussetzung erfüllt ist. Nutzen Sie den vorher erstellten Signal-Semaphor dafür, den Kind-Prozess zu blockieren, solange in der Queue keine Nachrichten vorhanden sind.
7. Sorgen Sie in jedem Fall dafür, dass **beide** Prozesse beendet werden und das Shared Memory aufgelöst wird.

## Teil 3: Zeitmessung und Scheduling

Sie können nun komplexere Daten zwischen Prozessen tauschen. Schauen Sie sich nun an, in welchem zeitlichen Rahmen die Nachrichten ausgetauscht werden und wie die Queue-Tiefe sowie das gewählte Scheduling Einfluss auf diese Zeit nimmt.

1. Geben Sie jeder CMessage, welche Sie in die Queue schreiben, einen Zeitstempel mit geeigneter Zeitauflösung mit. Lesen Sie diesen Zeitstempel im Kind-Prozess und bilden Sie eine Zeitdifferenz mit der Empfangszeit, um die Transferdauer ausgeben zu können. Geben Sie am Ende zudem die Durchschnittstransferdauer aus.
2. Schreiben Sie 10.000 Nachrichten in die Queue und vergleichen die Transferzeiten mit jeweils der Queue-Tiefe 1, 4, 16, … . Was stellen Sie fest? Wie erklären Sie sich den Effekt?
3. Ändern Sie das Scheduling der beiden Prozesse. Können Sie einen Unterschied feststellen?