Aufgabe 8:

10.12.2002

Shared-Memory und Semaphore

Programmieren Sie ein Erzeuger-Verbraucher-System, das einzelne Zeichen über einen Ringpuffer überträgt. Der Erzeugerprozess liest von der Standard-Eingabe und schreibt in den Ringpuffer, der Verbraucherprozess liest aus dem Ringpuffer und schreibt auf die Standard-Ausgabe. Im einzelnen sollen Erzeuger und Verbraucher die nachfolgende Funktionalität haben.

Erzeuger:

Der Prozess erzeugt ein Shared-Memory-Segment und die für die Koordinierung benötigten Semaphore. Der Key für diese Datenstrukturen soll mittels ***ftok(3)*** aus dem Directory

`/proj/i4sp/<`*`login`*`>/aufgabe8`

gewonnen werden. Am Anfang des Shared-Memory-Segments soll eine Verwaltungsdatenstruktur angelegt werden, die alle Daten enthält, die für die Verwaltung der Kommunikation benötigt werden. Außerdem liegt der Kommunikationspuffer im Shared-Memory. Der Puffer soll als Ringpuffer benutzt werden. Die Größe des Ringpuffers soll über eine Macro-Definition in einer include-Datei festgelegt werden. Testen Sie Ihr Programm mit den Pufferlängen 1024, 12, 2 und 1, für die Abgabe ist Puffergröße 12 einzustellen. Die Schreiboperationen auf dem Ringpuffer müssen so mittels Semaphore koordiniert werden, dass der Ringpuffer nicht überläuft und konkurrierende Zugriffe von Erzeuger und Verbraucher nicht zum Verlust von Daten führen!

Beim Start des Erzeugers überprüft dieser, ob bereits ein Erzeuger existiert und terminiert in diesem Fall. Beim Beenden wartet der Erzeuger, bis alle Daten aus dem Ringpuffer gelesen wurden, terminiert dann den Verbraucher durch das Signal SIGPIPE und löscht sowohl das Shared-Memory Segment als auch die Semaphore.

Verbraucher:

Der Verbraucherprozess *attached* das Shared-Memory-Segment und überprüft, dass noch kein Verbraucher existiert. Existiert bereits ein Verbraucher, terminiert er mit einer Fehlermeldung. Existiert noch kein Verbraucher, schreibt er seine Prozess-Id in die Verwaltungsstruktur, beginnt aus dem Ringpuffer zu lesen und die Daten auf dem Standardausgabekanal auszugeben. Am Ende der Datenübertragung wird er vom Erzeuger mit dem Signal SIGPIPE terminiert. Er muss sich nicht weiter um Aufräumarbeiten kümmern.

Es soll möglich sein, den Verbraucher mit dem Signal SIGINT abzubrechen. In diesem Fall meldet sich der Verbraucher ab (er signalisiert mit Prozess-Id "-1" in der Verwaltungsstruktur, dass kein Verbraucher mehr vorhanden ist!) und terminiert. Danach soll es möglich sein einen neuen Verbraucher zu starten, der mit der Ausgabe fortfährt.

Beachten Sie hierbei, dass SIGINT an beliebiger Stelle eintreffen kann, auch während gerade Daten aus dem Ringpuffer gelesen werden. Die Verwaltungsdaten (z. B. Schreib/Lese-Indizes/Semaphore) dürfen dabei auf keinen Fall in einen inkonsisten Zustand geraten

© Universität Erlangen Nürnberg, Informatik 4, 2002
Bei Fragen zur Aufgabenstellung bitte Mail an *i4sp@informatik.uni-erlangen.de*

Hinweise:

Da sowohl Erzeuger als auch Verbraucher Semphore und Shared-Memory benötigen, sind die dafür notwendigen Funktionen in je zwei eigenständige Dateien auszulagern (**sem.c** und **shm.c**). Die Prototypen dafür sollen in der Headerdatei **common.h** abgelegt werden. Erzeuger und Verbraucher sind entsprechend in **erzeuger.c** und **verbraucher.c** zu implementieren. Zur automatischen Übersetzung ist ein Makefile zu erstellen, bei dem durch "**make all"** beide Programme erzeugt werden können.

Die Koordinierung auf dem Shared-Memory (Puffer und Verwaltungsstruktur) muss in der Datei **doc/sem.doc** knapp aber präzise dokumentieren werden.

Außerdem ist der Aufbau der Verwaltungsstruktur zu dokumentieren und zu begründen (warum wurde welche Strukturkomponente eingeführt).

Abgabe: bis spätestens Donnerstag, 16.01.2003, 15:30 Uhr

© Universität Erlangen Nürnberg, Informatik 4, 2002
Bei Fragen zur Aufgabenstellung bitte Mail an *i4sp@informatik.uni-erlangen.de*