Aufgabe 3:

Programmieren Sie ein Erzeuger-Verbraucher-System, das Daten 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 folgende 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 Homedirectory 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 (Erzeuger schreibt bis zum Ende des Puffers und fängt dann wieder vorne an) 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. Die Schreiboperationen auf dem Ringpuffer müssen über Semaphore geeignet mit den Leseoperationen des Verbrauchers koordiniert werden!

Sollte bereits ein Erzeuger-Prozeß laufen, sollen weitere Erzeuger-Starts mit einer Fehlermeldung abbrechen.

Shared-Memory-Segment und Semaphore bleiben auch nach dem Ende des Programmlaufs, in dem sie erzeugt wurden, im Betriebssystem liegen. Sie müssen deshalb dafür sorgen, daß bei Beendigung des Erzeugers (auch bei Abbruch durch die Signale SIGHUP, SIGINT, SIGQUIT und SIGTERM!) aufgeräumt wird (***shmctl(2)***, ***semctl(2)***). Ein evtl. vorhandener Verbraucherprozeß soll im Rahmen der Aufräumaktion mit dem Signal SIGPIPE beendet werden. Dabei muß allerdings gewartet werden, bis der Verbraucher alle Daten aus dem Ringpuffer gelesen und ausgegeben hat.

Mit Hilfe des Kommandos ***ipcs(1)*** können Sie sich ausgeben lassen, welche dieser Betriebsmittel angelegt wurden, mit ***ipcrm(1)*** können Sie im Ernstfall (z. B. wenn Ihr Programm mit Segmentation Fault abgebrochen ist) über eine Kommandozeile aus der Shell aufräumen.

Achten Sie beim Erzeugen von Shared-Memory und Semaphoren darauf, daß Sie ausreichend Zugriffsrechte vergeben (am besten Read für alle).

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 (Koordinierung hier nicht vergessen, es könnten mehrere Verbraucher gleichzeitig loslaufen!) und 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, das Verbraucher-Programm neu zu starten. Dieser Prozeß klemmt sich wieder am Puffer an und gibt dessen Inhalt weiter aus.

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 inkonsistenten Zustand geraten!