Aufgabe V8: Kanal mit Puffer

Ein Kanal ist ein Treffpunkt für Prozesse, die Nachrichten senden oder empfangen. Trifft ein Sender auf einen Empfänger, überträgt der Kanal die Nachricht. Für beide Operationen bietet der Kanal verschiedene Kopplungsformen an. Er ist als Kernobjekt implementiert und verwendet folgende Attribute:

- Menge **NP** für Tupel (Nachrichtenreferenz, Prozess). Initial leer.
  Der Prozess darf `null` sein.
- Menge **ZP** für Tupel (Zielreferenz, Prozess). Initial leer.
  Der Prozess darf nicht `null` sein.
- Pufferspeicher **PS** für mehrere Nachrichten, Größe begrenzt. Initial leer.
  Man kann den freien Platz abfragen, Nachrichten hinein- und herauskopieren, und sie einzeln aus dem Pufferspeicher löschen.

Ungültige Fälle wären:

1. Sowohl **NP** als auch **ZP** enthalten ein Element.
2. **PS** enthält eine Nachricht, welche **NP** nicht referenziert.

Es gibt eine maximale Nachrichtengröße für jeden Kanal. Bei allen Sendeoperationen übergibt der Aufrufer eine Referenz auf die Nachricht in seinem Adressraum. Bei allen Empfangsoperationen übergibt der Aufrufer eine Referenz auf einen Zielbereich, d.h. einen Empfangspuffer, in seinem Adressraum. Sie können für diese Aufgabe davon ausgehen, dass Empfangspuffer mindestens so groß sind wie die maximale Nachrichtengröße.

a) Beschreiben Sie den Ablauf des synchronen Empfangens. (`receiveSyn`)

- **NP** enthält Element?

ja: Tupel (Nachrichtenref., Prozess) aus **NP** nehmen
Nachricht in Zielbereich kopieren
Prozess deblockieren, falls nicht `null`
Nachricht aus **PS** löschen, falls von dort

nein: Aufrufer blockieren
(Zielreferenz, Aufrufer) in **ZP** legen

Der Nein–Fall von `receiveSyn` bildet das Gegenstück zum Ja–Fall von `sendSyn` aus der Aufgabenstellung. Hier wird ein Element in **ZP** eingefügt, dort entnommen.

b) Beschreiben Sie den Ablauf des asynchronen Sendens (`sendAsyn`) mit Wertablage. Falls nicht genug Pufferspeicher frei ist, scheitert die Operation.

- Nachricht zu groß? ⇒ Abbruch mit Fehler
- **ZP** enthält Element?

ja: Tupel (Zielreferenz, Prozess) aus **ZP** nehmen
Nachricht in Zielbereich kopieren
Prozess deblockieren

nein: Nicht genug Platz in **PS**? ⇒ Abbruch mit Fehler
Nachricht in **PS** kopieren
(Kopiereferenz, `null`) in **NP** legen

Der Nein–Fall von `sendAsyn` bildet das Gegenstück zu den zwei bedingten Schritten in `receiveTry` aus der Aufgabenstellung. Hier wird ein Element in `NP` eingefügt, bei dem der Prozess `null` ist und die Nachricht in **PS** liegt.

c) Was bedeuten die beiden ungültigen Fälle?

Fall 1: Der Kanal kennt einen Sender und einen Empfänger, überträgt aber die Nachricht nicht.

Fall 2: Eine Nachricht belegt noch Pufferspeicher, obwohl sie zugestellt wurde oder nie mehr zugestellt werden kann.

d) Die Operation `receiveSyn` wird von n Prozessen in Dauerschleife aufgerufen. Schätzen Sie den Speicherbedarf des Kanals dafür ab.

Der Kanal belegt für Empfangsoperationen weder Speicher in **NP** noch in **PS**. In **ZP** liegt pro blockiertem Empfänger ein Element. Für die n Prozesse braucht der Kanal also höchstens n Elemente.

Da Prozesse nur an einer Operation gleichzeitig blockieren, egal an welchem Kernobjekt, kann man den Speicher für die Verwaltung in Wartemengen schon beim Erzeugen eines Prozesses reservieren. In Cosy *war das so gelöst.*

e) Die Operation `sendAsyn` wird von m Prozessen in Dauerschleife aufgerufen. Schätzen Sie den Speicherbedarf des Kanals dafür ab.

Der Kanal belegt für Sendeoperationen keinen Speicher in **ZP**. Pro wartender Nachricht braucht er ein Element in **NP** sowie Platz für den Inhalt in **PS**. In **PS** könnte auch etwas Verwaltungsinformation pro gepufferter Nachricht liegen.

Da es um asynchrones Senden geht, kann jeder Prozess mehrere Nachrichten im Kanal puffern. Der Platz in **PS** ist beschränkt, was zu einer indirekten Beschränkung der Anzahl an Nachrichten führt. Bei kleinen Nachrichten ist diese aber wahrscheinlich ein hohes Vielfaches von m. Falls leere Nachrichten erlaubt sind und keine Verwaltungsinformation in **PS** belegen, ist die Anzahl der Elemente in **NP** unbeschränkt.

*Es empfiehlt sich, die Anzahl der Elemente in **NP** explizit zu beschränken.*