Aufgabe V6: Semaphore im Einsatz

Eine Pufferverwaltung auf Anwendungsebene erlaubt es, Puffer fester Größe anzufordern und wieder freizugeben. Die Anzahl der Puffer wird einmalig bei der Initialisierung festgelegt. Die Puffer sind ab 0 durchnummeriert. Die Verwaltung verwendet einen Bitvektor entsprechender Länge, in dem 0 einen freien und 1 einen belegten Puffer anzeigt. Die Operationen laufen wie folgt ab:

Anfordern: *kein Argument, Rückgabewert ist eine Puffernummer oder ein Fehler*

1. im Bitvektor nach einer 0 suchen
2. nichts gefunden ⇒ Fehler zurückgeben
3. gefundenes Bit auf 1 setzen
4. entsprechende Puffernummer zurückgeben

Freigeben: *Argument ist eine Puffernummer, kein Rückgabewert*

8. im Bitvektor das entsprechende Bit auf 0 setzen

Die Pufferverwaltung ist noch nicht für parallele Aufrufe durch mehrere Prozesse (engl.: *threads*) gleichzeitig ausgelegt. Gehen Sie davon aus, dass die Anwendung nur gültige Aufrufe durchführt, also nur zuvor angeforderte Puffer freigibt.

a) Bestimmen Sie die kritischen Abschnitte beider Operationen.

b) Sichern Sie die Pufferverwaltung mit einem Semaphor `M` so ab, dass höchstens ein kritischer Abschnitt gleichzeitig ablaufen kann. Mit welchem Wert wird `M` initialisiert? Wo müssen Aufrufe von `M.P()` bzw. `M.V()` stattfinden?

c) Ändern Sie die Pufferverwaltung mit einem zweiten Semaphor `Z` so, dass das Anfordern auf einen freien Puffer wartet, statt einen Fehler zurückzugeben. Mit welchem Wert wird `Z` initialisiert? Wo müssen Aufrufe von `Z.P()` bzw. `Z.V()` stattfinden?

d) Was kann schiefgehen, wenn zwei Prozesse gleichzeitig die ungesicherte Pufferverwaltung nutzen, also ohne das Semaphor `M`?

- beide rufen **Anfordern**
- beide rufen **Freigeben**
- einer ruft **Anfordern**, der andere **Freigeben**