OPDRACHT 5. PRODUCER/CONSUMER

In deze opdracht onderzoek je hoe je een data producerend proces (of thread) en een data consumerend proces (of thread) via een eindig buffer kunt laten communiceren.

1. Voorbereidingen

Bestudeer Stallings paragraaf 5.3 *The Producer/Consumer Problem*.

2. Taken

2.1 Bounded buffer

Start de applet http://www.doc.ic.ac.uk/~jnm/book/book_applets/BoundedBuffer.html.

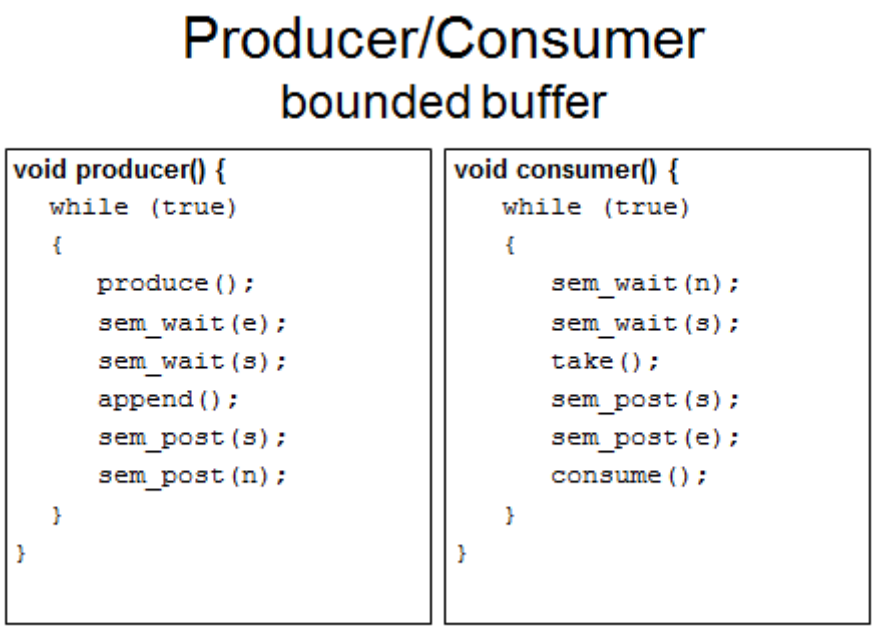
Opdrachten:

- Start beide processen
- Laat een van beide processen pauzeren totdat de ander ook niet meer verder kan
- En daarna het andere proces

Beschrijf je bevindingen.

2.2 Producer/Consumer

We gaan het Producer/Consumer probleem uit het boek van Stallings (zie paragraaf 5.3, *figuur 5.13: A Solution to the Bounded-Buffer Producer/Consumer Problem Using Semaphores*) bekijken, en wel eerst voor 1 producer en 1 consumer.



Maak een implementatie m.b.v. een circulair array. Schrijf twee programma's: `producer` en `consumer` die je als twee losse processen opstart (dus niet met threads).

In het boek worden de operaties `append()` en `take()` wel aangeroepen, maar niet geïmplementeerd. Dat moet je zelf doen.
Het schrijvende proces `producer` gebruikt in de functie `append()` een variabele (geen shared!) `in` die bijhoudt welke de volgende plaats is waar geschreven dient te worden.
Het lezende proces `consumer` gebruikt in de functie `take()` een variabele (geen shared!) `out` die aangeeft welke de volgende plaats is die gelezen dient te worden.

Beide processen maken gebruik van een array `b[]` van `unsigned long`s, dat in shared memory wordt geplaatst.
De `producer` schrijft opeenvolgende getallen (1, 2, 3, …) en (let op!) de `consumer` schrijft een 0 in elke zojuist gelezen plaats.
Maak in `producer` een check dat er daadwerkelijk een 0 staat in een veld dat hij wil gaan vullen (om aan te tonen dat de vorige waarde inderdaad gelezen is door `consumer`).
Laat `consumer` de ingelezen waarde uitprinten (dan dienen de getallen 1, 2, 3, … weer te verschijnen).
Je oplossing hoeft *niet* bestand te zijn tegen integer-overflow.

Kies voor de grootte van het array zelf een waarde, bijvoorbeeld 7.
Check ook of je implementatie werkt bij andere buffergroottes, bijvoorbeeld 1, of 2, of 100, of 10000.

Vraag: Is de semaphore `s` (zoals beschreven in het boek) nodig? Zoniet, verwijder deze dan en test opnieuw.

Extra's:

- Breid je programma's uit zodat meerdere `producer` processen en `consumer` processen kunnen worden opgestart. (`in` en `uit` worden dan wel shared variabelen).
- Genereer random extra belasting bij `producer` en `consumer` (bijvoorbeeld met busy-wait loops). Zorg er dan voor dat:
    - soms heeft `producer` het erg druk
    - soms heeft `consumer` het erg druk
    - soms loopt `producer` een beetje sneller dan `consumer`
    - soms loopt `consumer` een beetje sneller dan `producer`.

  beschrijf hoe je dit gerealiseerd hebt, en hoe je het getest hebt

3. Opleveren

Je dient een document op te leveren waarin je de bovenstaande experimenten uitwerkt. Voeg, waar nodig, screendumps toe om een en andere duidelijk te maken. Lever ook de bijbehorende source code in van de programma’s die je gemaakt of aangepast hebt.