Betriebssysteme Praktikum Aufgabe 2

Zugehörigkeitstabelle

<table>
<tr><td></td><td colspan="2">rb_mutex</td></tr>
<tr><td rowspan="6">Condition_Var./
Entry_Condition

(signal on CV if EC true)</td><td>CV</td><td>EC</td></tr>
<tr><td>not_empty_condvar</td><td>count != 0</td></tr>
<tr><td>not_full_condvar</td><td>count <= MAX</td></tr>
<tr><td>prod_1_restart</td><td>prod_1_stopped == 1</td></tr>
<tr><td>prod_2_restart</td><td>prod_2_stopped == 1</td></tr>
<tr><td>cons_restart</td><td>cons_stopped == 1</td></tr>
<tr><td>Threads</td><td colspan="2">producer_1
producer_2
consumer</td></tr>
<tr><td>Tests</td><td colspan="2">a) if producer_1 is writing, producer_2 and consumer have to be blocked or inactive.
b) if producer_2 is writing, producer_1 and consumer have to be blocked or inactive.
c) if consumer is writing, producer_1 and producer_2 have to be blocked or inactive.
d) If rb is full, producer_1 and producer_2 have to blocked or inactive.
e) If rb is full, consumer has to get a char from rb and unblock producer_1 and producer_2.
f) If rb is empty, consumer has to be blocked.
g) If rb is empty, a producer hast o unblock the consumer after putting a char into rb.</td></tr>
</table>

Kommentar zu Abbildung f)

In der Abbildung (f) ist ein Overflow des Ringbuffers zu beobachten. Es wird also ein Element mehr rein geschrieben, als eigentlich vom Ringbuffer gehalten werden kann. Somit wird das älteste Element des Ringbuffer mit dem neuen Element überschrieben. Sowohl In- als auch Out-Pointer werden dabei einen weiter gesetzt, um auf das nun älteste Element zu zeigen, um den „normalen“ Ablauf weiter zu betreiben.

Pointer-Semantik

Zum Ringbuffer gehören zwei Pointer – `p_in` und `p_out`.
`p_in` zeigt auf die als nächstes zu beschreibende Speicherzelle im Array des Ringbuffers.
`p_out` zeigt auf die als nächstes auszulesende Speicherzelle.
Wird eine Leseoperation auf den `RB` ausgeführt, wird im Anschluss der Operation, `p_in` auf die Adresse der nächsten Speicherzelle gesetzt. Wird eine Schreiboperation auf `RB` ausgeführt, wird im Anschluss `p_out` auf die Adresse der nachfolgenden Speicherzelle gesetzt. Außerdem darf `p_in` nicht `p_out` überholen, da sonst nicht gelesene Zeichen überschrieben würden. Ist einer der beiden Pointer im Begriff, durch eine Lese- bzw. Schreiboperation auf dem Ringbuffer, auf eine Adresse über die Grenze des Ringbuffer-Speichers hinweg gesetzt zu werden, wird der entsprechende Pointer auf die Startadresse (Adresse der Startspeicherzelle) des Ringbuffers zurückgesetzt, um den Zyklus zu gewährleisten und einer Speicherzugriffsverletzung vorzubeugen.

Beobachtungen

Als Ausgabe bekommt man standardmäßig das Alphabet mit abwechselnd großen und kleinen Buchstaben, also: a, A, b, B, c, C,… Es kann aber auch sein, dass erst der große Buchstabe ausgegeben wird und dann der Kleine, falls die Priorität verändert wurde, sodass der zweite Producer eine höhere hat, als der erste Producer. Da der Consumer für die Ausgabe zuständig ist, wird nichts mehr ausgegeben, falls er angehalten wird. Sobald man diesen wieder startet, geht es mit der Ausgabe weiter, wo dieser vorher aufgehört hat. Stoppt man einen Producer, wird man nicht unbedingt sofort einen Unterschied sehen, da der Producer schon mehrere Zeichen in den Ringbuffer geschrieben hat, die der Consumer zuerst einmal abarbeiten muss. Nach einer bestimmten Zeit (je nachdem, wie viele Zeichen noch übrig waren) wird man dann jedoch nur noch die Zeichen sehen, die vom anderen Producer in den Ringbuffer geschrieben werden. Wenn man nun beide Producer stoppt, erhält man, wie vorher, noch für eine bestimmte Zeit die restlichen eingespeicherten Zeichen und die Ausgabe versiegt, bis erneut ein Producer gestartet wird. Diese Ausgaben gehen so lange weiter, bis das Programm beendet wird.

Kommunikationsdiagramm