Betriebssysteme und Systemsoftware *Andrés Montoya, 405409*
Übungsblatt 3 *Til Mohr, 405959*
Tutorium 1 *Marc Ludevid Wulf, 405401*

Vorlesungen 6.2.c - 6.2.e sind immer noch nicht auf Moodle, 2 Tage vor der Abgabe....

Aufgabe 5.1 Nein, die Implementierung funktioniert nicht wie gewünscht: Folgendes Beispiel könnte nicht wie gewünscht funktionieren: Angenommes es kommt zuerst ein Politiker p1 und dann ein Jornalist j1 an. Dann wird durch das signal von p1 j1 im zweiten wait sein. Wenn dann ein weiterer Jornalist j2 und dann ein weiterer Politiker p2 ankommt können zwei Szenarien passieren: entweder das signal von p2 befreit j1 und somit funktioniert das Programm wie gewünscht, oder j2 konsumiert das signal und somit sind beide Jornalisten im zweiten wait. Das ist nicht gewünscht. Die Lösung ist einen Mutex reporterWaiting die sicherstellt, dass nur ein Journalist in dem kritischen Bereich befindet. Also muss man lock(reporterWaiting) als erste und release(reporterWaiting) als letzte Anweisung der Methode reporterArrives() einfügen.

Aufgabe 5.2

a) Sei `use` ein Zählsemaphore, der auf den Wert `0` initialisiert wurde. Seien `empty` und `full` Zählsemaphore, die auf `n` und `0` initialisiert werden und beide Maximalwerte von `n` haben.

```
void enqueue(element) {
        wait(empty);
        wait(use);
        queue.add(element);
        signal(use);
        signal(full);
}

element dequeue() {
        wait(full);
        wait(use);
        element := queue.pop();
        signal(use);
        signal(empty);
        return element;
}
```

b) Es können zwei Wagen gleichzeitig einfahren, falls 2 Wagen aufs einfahren Warten (`while`-Schleife Zeile 5), `eingefahreneneWagen` auf `0`

gesetzt wird, dann zuerst der eine Wagen die `while`-Schleife verlässt, dann der andere Wagen, bevor der erste Wagen überhaupt `eingefahreneneWagen` auf `1` setzen konnte.

Es können mehrere Besucher gleichzeitig einen Wagen betreten: Sei 1 Wagen gerade eingefahren, 3 Besucher warten aufs Betreten (`while`-Schleife Zeile 17). Wenn wie oben alle 3 Besucher direkt hintereinander deren `while`-Schleifen verlassen, ohne Zeile 19 einmal auszuführen, können diese 3 Besucher den Wagen betreten.

In beiden Szenarien sind die Anzahlen nur beispiele, es kann für beliebiges $n \in \mathbb{N}$ auftreten.

Zudem ist wird nie sichergestellt, dass sowohl Besucher als auch Wagen der Reihe nach betreten/einfahren. Gerade bei den Wagen kann dies fatal werden, wenn sie auf einer festen Spur fahren und ein hinterer Wagen einfahren will und dabei andere Wagen zerstört.

c) Man braucht einen Mutex wagenPlatz der auf 0 initialisiert wird, eine Zählsemaphore besucherPlatz das auf 0 initialisiert wird und einen Maximalwert von der maximalen Länge der Warteschlange der Achterbahn hat und eine Zählsemaphore besucherEingetreten das auf 0 initialisiert wird und einen Maximalwert von 2 hat.

```
void AnkunftWagen() {
        wait(wagenPlatz);
        fahreAufPlattform();
        oeffnetTueren();
        signal(besucherPlatz);
        signal(besucherPlatz);
        wait(besucherEingetreten);
        wait(besucherEingetreten);
        schliesseTueren();
        verlassePlattform();
        signal(wagenPlatz);
}

void AnkuftBesucher() {
        wait(besucherPlatz);
        betreteWagen();
        signal(besucherEingetreten);
```

```
}
```

d) Ja, man benutzt eine mit einem Mutex geschützte priorityQueue. VIP-Besucher erhalten eine niedrigere Priority als normale Besucher. Dadurch würden diese immer zuerst drankommen.

Aufgabe 5.3

a) Siehe `restaurant_geruest.c`

b) Siehe `restaurant_geruest_b.c`