POSIX/C Multithreading und Synchronisation (joins und barriers)

Aufgabe 1: Übersetzen Sie das gegebene Programm **`no01Vorgabe.c`** und starten Sie es.

- Studieren Sie den Code, um die Funktionsweise zu verstehen.
- Variieren Sie die Parameter der beiden **`usleep`**`(...)` Aufrufe.
- Versuchen Sie zu verstehen, warum die Ausgabe `Main beendet` niemals erscheint. Hilft das Entfernen des `pthread_exit` Aufrufs?

Aufgabe 2: Erstellen Sie eine neue Version des obigen Programms: **`no02.c`**

- Entfernen Sie den `pthread_exit` Aufruf.
- Der *main-thread* soll sich jetzt per **`pthread_join(...)`** mit den beiden threads *rauf* und *runter* synchronisieren, sodass die Meldung `Main beendet` erst nach Beendigung von *rauf* und *runter* erscheint.
- Versehen Sie die thread-Funktionen *rauf* und *runter* mit einem Parameter `delayMs`, um die Verzögerungszeit in Millisekunden angeben zu können. Der letzte Parameter beim `pthread_create` - Aufruf zeigt auf den Wert von `delayMs`.
- Testen Sie das Verhalten jetzt auch mit unterschiedlichen delay-Einstellungen für *rauf* und *runter*.

Aufgabe 3: Übersetzen, starten und studieren Sie das Programm **`no03Vorgabe.c`**. Es zeichnet 5000 kleine blaue Quadrate in ein Fenster.
Es wird die SDL 1.2 Library verwendet, die möglicherweise auf Ihrem System erst installiert werden muss (siehe Kommentar im Dateikopf).

Das Programm soll wie folgt erweitert werden:

- Mit zwei weiteren threads soll die SH-Flagge gezeichnet werden. Ein Blau-thread, ein Weiß-thread und ein Rot-thread zeichnen gleichzeitig jeweils ihre Quadrate.
- Ändern Sie `zeichneBlaueQuadrateAnZufaelligerPosition` in `zeichneQuadrateAnZufaelligerPosition` und fügen Sie einen Parameter für die Farbe hinzu. Die thread-Funktion kann jetzt für alle drei threads verwendet werden.
- Ändern Sie die Größe der Quadrate in z.B.: `min/10`
- Ändern Sie die Anzahl der Quadrate in z.B. `500` (pro Farbe)
- Die drei threads sollen jetzt wie folgt nur in bestimmte Bereiche zeichnen, so dass eine SH-Flagge entsteht:
  - blau zeichnet in das obere Drittel
  - weiß zeichnet in das mittlere Drittel
  - rot zeichnet in das untere Drittel

  Hierzu muss nur der Aufruf von `getZufallsZahl` für die `r1.y`-Berechnung entsprechend begrenzt werden. Dies kann abhängig vom Farbwert erfolgen: blau -> oberes Drittel, weiss -> mittleres, ...
- Setzen Sie die Verzögerung z.B. auf 10 Millisekunden.
  Der *Grafik-Main-Loop* im main-thread wird jetzt erst erreicht, wenn die drei Farbthreads beendet sind. Vorher wird kein Tastendruck zum Beenden akzeptiert!
- Geben Sie eine Meldung aus, wenn alle Quadrate gezeichnet sind.

So etwa sieht die Flagge dann aus:

Aufgabe 4: Das Programm **`no04.c`** soll mit **`pthread_join`** folgende Sequenz erzwingen:

- zuerst werden alle blauen Quadrate gezeichnet
- dann werden alle weißen Quadrate gezeichnet
- dann werden alle roten Quadrate gezeichnet

Wenn alle Quadrate gezeichnet sind, soll wieder eine Meldung ausgegeben werden.

Um den *Grafik-Main-Loop* nicht weiter mit den joins zu blockieren, erstellen Sie einen weiteren thread **`startUndSync`**, der jetzt den Start der drei Farb-threads und die *join-Synchronisation* übernimmt. Der main-thread kann nun ungestört den *Grafik-Main-Loop* ausführen, während **`startUndSync`** durch joins blockiert ist.

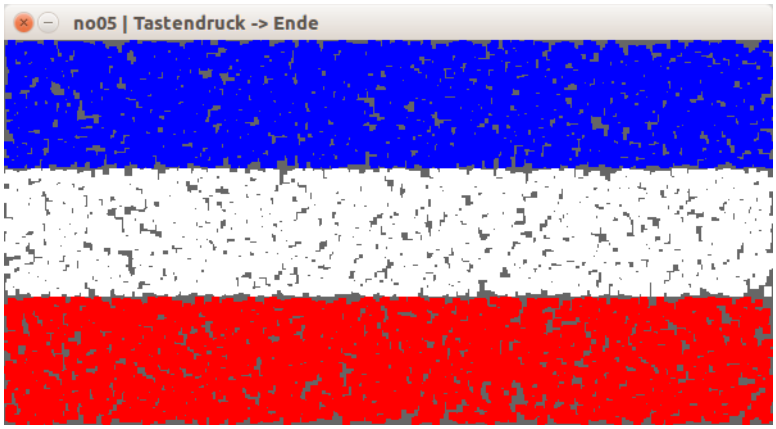
Aufgabe 5: In **`no05.c`** soll mit **`pthread_barrier_wait`** eine Abfolge wie in **`no04`** entstehen, die allerdings mehrfach durchlaufen wird, also:

von 1 bis AnzahlDurchläufe:
- N blaue Quadrate zeichnen
- N weiße Quadrate zeichnen
- N rote Quadrate zeichnen

Außerdem:

- Es soll ein Array von z.B. 100 Zeichenthreads erstellt werden. Diese threads verwenden die `zeichneQuadrate...` -Funktion jetzt ohne Farbparameter. Sie lesen die Farbe aus einer global definierten Struktur.
- Nachdem die 100 threads z.B. je drei Quadrate gezeichnet haben (N ist jetzt 300), warten sie an einer Barriere. Der **`startUndSync`**-thread schreibt die neue Farbe in die globale Struktur und gibt die Barriere wieder frei.
- Die obige Schleife „*von 1 bis AnzahlDurchläufe*“ wird von **`startUndSync`** ausgeführt.
- Weitere Parameter etwa:
  - Größe der Quadrate: `min/50`
  - Anzahl der Durchläufe (immer Vielfaches von 3): `3 * 15`
  - Verzögerung zwischen den Farbwechseln: `200` Millisekunden

Insgesamt werden dann *N * 3 Farben * 45 Durchläufe = 40500* Quadrate gezeichnet und so sieht es dann etwa aus: ------>


no05 | Tastendruck -> Ende