3 Hausaufgaben

3.1 Synchronisation beim Ressourcenzugriff: Mutex (6 Punkte)

Wenn mehrere Prozesse oder Threads sich die Standardausgabe ("stdout") des Terminals teilen, können sich die Ausgaben überlagern und schwer lesbar werden. Effektiv stellt stdout eine gemeinsam genutzte Ressource dar, die die Prozesse oder Threads sinnvollerweise nur in kompletten Ausgabeblöcken nutzen sollen, die nicht unterbrochen werden. Wir wollen das an einem praktischen Beispiel veranschaulichen.

Nehmen Sie als Ausgangspunkt Ihr Programm `threadit` vom letzten Übungsblatt (inkl. der Listenverwaltung). Lassen Sie maximal 10 Threads zu. Die Anzahl der zu startenden Threads soll wieder auf der Kommandozeile (Parameter `-n`) übergeben werden. Erweitern Sie Ihre Threadfunktion so, dass diese die laufende Nummer des Threads k (beginnend bei 0) als Parameter übergeben bekommt und dann folgende Aktionen durchführt:

- die Datei *k*`.txt` mittels `open()` öffnet,
- den Inhalt der Datei in Einheiten von 64 Zeichen mittels `read()` liest und die laufende Nummer i der Einheit (beginnend bei 0) festhält,
- einen Ausgabepräfix erzeugt, der k und i der Ausgabe voranstellt, und zwar im Format `"[%02d] %03d"`, wobei erst k und dann i ausgegeben werden soll, gefolgt von einem Tab `\t`,
- dann die 64 Zeichen gefolgt von einem Newline ausgibt, wobei angenommen werden kann, dass alle Zeichen in der Datei darstellbar sind.

Verwenden Sie drei Mal den `write()`-Systemaufruf: 1. für den Präfix, 2. für die Einheit von 64 (oder weniger) Bytes aus der Datei und 3. für das Newline. (Das ist nicht effizient, dient aber der Übung.)

Wir stellen hierfür Beispieldateien bereit, die `0.txt`, `2.txt`, ..., `9.txt` benannt sind und sich im aktuellen Arbeitsverzeichnis von blatt05 befinden sollen.

Starten Sie das Programm wiederholt, und betrachen Sie die Ausgabe. Kann man in allen Fällen alle Zeilen gut lesen?

Stellen Sie nun zwei sinnvolle Gruppierungen bereit, die (nicht gleichzeitig) mit der Option `-l` (Line) oder `-f` (File) ausgewählt werden.

- `-l` fasst alle drei `write()`-Anweisungen in einem kritischen Abschnitt zusammen, so dass immer nur eine ganze Zeile von einem Thread atomar ausgegeben werden kann.
- `-f` fasst alle Ausgaben einer Datei zu einem kritischen Abschnitt zusammen, so dass immer nur der gesamte Dateiinhalt am Stück ausgegeben wird.

Schützen Sie diese kritischen Abschnitt gegen gleichzeitiges Betreten durch mehrere Threads, indem Sie ein Mutex benutzen.

Syntax: `syncem [-n <# Threads>] [-l|-f]`

Quellen: `syncem.c list.c list.h`

Executable: `syncem`

Beispielausgaben:

Ohne Synchronisation

```
$ ./syncem -n 10
[00] 000        0000000000000000000000000000000000000000000000000000000000000000000000000000
[01] 000        1111111111111111111111111111111111111111111111111111111111111111111111111111[00] 001
0000000000000000000000000000000000
[01] 001        11111111111111111111111111111111
[02] 000        22222222222222222222222222222222222222222222222222222222222222222222
[02] 001        22222222222222222222222222222222
[03] 000        33333333333333333333333333333333333333333333333333333333333333333333333333
[03] 001        333333333333333333333333333333333
[04] 000        [05] 000        4444444444444444444444444444444444444444444444444444444444444444444455555555555555555

[05] 001        [04] 001        5555555555555555555555555555555554444444444444444444444444444444444

[06] 000        66666666666666666666666666666666666666666666666666666666666666
[06] 001        6666666666666666666666666666666
[07] 000        77777777777777777777777777777777777777777777777777777777777777
[08] 000        [09] 000        88888888888888888888888888888888888888888888888888888888888888888888888888888888999999999999999999
[09] 001        99999999999999999999999999999999

[08] 001        888888888888888888888888888888888
[07] 001        7777777777777777777777777777777777
```

Synchronisation pro Zeile

```
$ ./syncem -n 10 -l
[00] 000        0000000000000000000000000000000000000000000000000000000000000000000000000000
[02] 000        22222222222222222222222222222222222222222222222222222222222222222222
[03] 000        33333333333333333333333333333333333333333333333333333333333333333333333333
[04] 000        44444444444444444444444444444444444444444444444444444444444444444444
[00] 001        00000000000000000000000000000000
[05] 000        55555555555555555555555555555555555555555555555555555555555555555555
[06] 000        66666666666666666666666666666666666666666666666666666666666666666666
[01] 000        11111111111111111111111111111111111111111111111111111111111111111111
[07] 000        77777777777777777777777777777777777777777777777777777777777777777777
[08] 000        88888888888888888888888888888888888888888888888888888888888888888888
[09] 000        99999999999999999999999999999999999999999999999999999999999999999999
[02] 001        22222222222222222222222222222222
[03] 001        333333333333333333333333333333333
[04] 001        44444444444444444444444444444444
[05] 001        55555555555555555555555555555555
[06] 001        66666666666666666666666666666666
[01] 001        11111111111111111111111111111111
[07] 001        77777777777777777777777777777777
[08] 001        88888888888888888888888888888888
[09] 001        99999999999999999999999999999999
```

Synchronisation pro Thread

```
$ ./syncem -n 10 -f
[02] 000        2222222222222222222222222222222222222222222222222222222222222222
[02] 001        22222222222222222222222222222222
[00] 000        0000000000000000000000000000000000000000000000000000000000000000
[00] 001        00000000000000000000000000000000
[01] 000        1111111111111111111111111111111111111111111111111111111111111111
[01] 001        11111111111111111111111111111111
[03] 000        3333333333333333333333333333333333333333333333333333333333333333
[03] 001        33333333333333333333333333333333
[04] 000        4444444444444444444444444444444444444444444444444444444444444444
[04] 001        44444444444444444444444444444444
[05] 000        5555555555555555555555555555555555555555555555555555555555555555
[05] 001        55555555555555555555555555555555
[06] 000        6666666666666666666666666666666666666666666666666666666666666666
[06] 001        66666666666666666666666666666666
[07] 000        7777777777777777777777777777777777777777777777777777777777777777
[07] 001        77777777777777777777777777777777
[08] 000        8888888888888888888888888888888888888888888888888888888888888888
[08] 001        88888888888888888888888888888888
[09] 000        9999999999999999999999999999999999999999999999999999999999999999
[09] 001        99999999999999999999999999999999
```

3.2 Synchronisation beim Ressourcenzugriff: Conditional Variables (4 Punkte)

Schreiben Sie ein Progamm `syncorder`, welches Ihr Programm `syncem` um eine weitere Option `-o` ergänzt und dafür sorgt, dass die Ausgaben der Threads in aufsteigender Reihenfolge der laufenden Thread-Nummern erfolgt:

- `-o` fasst alle `write()`-Anweisungen eines Threads in einem kritischen Abschnitt zusammen und garantiert die Ausgabe alle Threads in aufsteigender Reihenfolge nach Threadnummer k.

Ersetzen Sie hierzu den Systemaufruf `write()` durch eine von Ihnen geschriebene Funktion `write_buffer()`, die wie folgt definiert ist:

- `int write_buffer (long thread, char *buffer, int len);`

Die Funktion erhält als ersten Parameter die Laufnummer des Threads und danach (analog zu `write()`) einen Pointer auf den Puffer und die Pufferlänge. Die Funktion soll prüfen, ob der aufrufende Thread mit seiner Ausgabe an der Reihe ist. Falls ja, soll sie den Puffer mittels `write()` auf STDOUT ausgeben. Falls nicht, soll sie den Thread mit Hilfe einer Conditional Variable schlafen legen. Vergessen Sie nicht, ihn wieder aufzuwecken, wenn der aktuelle berechtigte Thread alle seine Ausgaben abgeschlossen hat.

Um zu erkennen, dass ein Thread mit seinen Ausgaben fertig ist, ruft er die Funktion `write_buffer` mit seiner Thread-Nummer, dem Pointer NULL und der Länge 0 auf.

Syntax: `syncorder [-n <# Threads>] [-l|-f|-o]`

Quellen: `syncorder.c list.c list.h`

Executable: `syncorder`

Beispielausgabe:

```
$ ./syncem -n 10 -o
[00] 000        00000000000000000000000000000000000000000000000000000000000000000000
[00] 001        00000000000000000000000000000000000
[01] 000        11111111111111111111111111111111111111111111111111111111111111111111
[01] 001        11111111111111111111111111111111
[02] 000        22222222222222222222222222222222222222222222222222222222222222222222
[02] 001        22222222222222222222222222222222
[03] 000        33333333333333333333333333333333333333333333333333333333333333333333
[03] 001        33333333333333333333333333333333
[04] 000        44444444444444444444444444444444444444444444444444444444444444444444
[04] 001        44444444444444444444444444444444
[05] 000        55555555555555555555555555555555555555555555555555555555555555555555
[05] 001        55555555555555555555555555555555
[06] 000        66666666666666666666666666666666666666666666666666666666666666666666
[06] 001        66666666666666666666666666666666
[07] 000        77777777777777777777777777777777777777777777777777777777777777777777
[07] 001        77777777777777777777777777777777
[08] 000        88888888888888888888888888888888888888888888888888888888888888888888
[08] 001        88888888888888888888888888888888
[09] 000        99999999999999999999999999999999999999999999999999999999999999999999
[09] 001        99999999999999999999999999999999
```