Wichtig: Lesen Sie auch den Teil "**Hinweise zur Aufgabe**" auf diesem Blatt; Spezifikationen in diesem Teil sind ebenfalls einzuhalten!

Aufgabe 4: jbuffer (14.0 Punkte)

Programmieren Sie eine Bibliothek *jbuffer* (**j**ob **b**uffer), die `int`-Werte in einem *Bounded Buffer* verwaltet. Der Puffer soll für einen Schreiber-Thread und mehrere konkurrierende Leser-Threads ausgelegt sein.
Der Puffer wird in der nächsten Aufgabe als Warteschlange eingesetzt werden, um Aufträge (Netzwerkverbindungen), die von einem Haupt-Thread angenommen werden, zur Abarbeitung an mehrere Arbeiter-Threads zu verteilen.

a) Ringpuffer-Modul

Implementieren Sie im Modul `jbuffer.c` einen Ringpuffer mit FIFO-Eigenschaft, der für einen einzelnen Produzenten-Thread und mehrere Konsumenten-Threads konzipiert ist. Verwenden Sie die vorgegebene Semaphor-Implementierung (`sem.o`, `sem.h`) aus dem Verzeichnis `/proj/i4sp2/pub/aufgabe4` zur Vermeidung von Über- und Unterlaufsituationen, so dass Produzent beziehungsweise Konsumenten beim Einfügen in einen vollen beziehungsweise bei der Entnahme aus einem leeren Puffer blockieren.
Die Konsumenten sollen untereinander nicht-blockierend koordiniert werden. Benutzen Sie hierfür die *CAS*-Funktion `atomic_compare_exchange_strong()`, so dass mehrere Konsumenten gleichzeitig den kritischen Abschnitt durchlaufen können (keine Locks!).

b) Semaphor-Modul

Programmieren Sie nun selbst auf der Basis von Mutex- (**pthread_mutex_init(3)**) und Condition-Variablen (**pthread_cond_init(3)**) einen zählenden Semaphor in der Datei `sem.c`. Die Schnittstelle des Moduls finden Sie im Vorgabeverzeichnis.

c) Statische Bibliothek

Erstellen Sie ein Makefile, welches das Ringpuffer- und das Semaphor-Modul zu einer statischen Bibliothek `libjbuffer.a` zusammenbindet. Unter Verwendung dieser Bibliothek soll das Testprogramm `jbuffer-test` aus der vorgegebenen Quelldatei `jbuffer-test.c` erzeugt werden. Die Pthread-Funktionen sind in einer speziellen Funktionsbibliothek (*libpthread*) zusammengefasst, die Sie beim Kompilieren und beim Binden Ihres Programms mit angeben müssen (Option `-pthread`).
Das Makefile soll die Standard-Pseudotargets `all` und `clean` enthalten. Bei Änderungen an einer Quelldatei soll sichergestellt sein, dass nur die benötigten Module neu übersetzt werden. Achten Sie darauf, alle Abhängigkeiten inklusive der Header-Dateien korrekt anzugeben! Nutzen Sie die auf der Webseite genannten Compilerflags.

d) Dynamische Bibliothek

Erweitern Sie das Makefile so, dass das Ringpuffer- und das Semaphor-Modul zu einer Programmbibliothek `libjbuffer-dynamic.so` zusammengebunden werden und unter Verwendung dieser Bibliothek ein dynamisch gebundenes Testprogramm `jbuffer-test-dynamic` erstellt wird. Die beiden Module der dynamischen Bibliothek (und nur diese) sollen als *Position-Independent Code* übersetzt werden – Sie müssen also für die beiden entsprechenden Objektdateien andere Namen verwenden als in Teilaufgabe c) und gesonderte Targets erstellen.

Hinweise zur Aufgabe:

- Die zu implementierenden Funktionen sollen im Fehlerfall weder Meldungen ausgeben noch das Programm beenden, sondern ausschließlich einen Fehlercode zurückliefern – die Fehlerbehandlung soll dem Aufrufer überlassen bleiben.
- `atomic_compare_exchange_strong()` ist eine Funktion aus der *Atomic operations library* die mit C11 standardisiert wurde. Die Dokumentation finden Sie unter `https://en.cppreference.com/w/c/atomic/atomic_compare_exchange`.
- Dynamische Bibliotheken werden aus Sicherheitsgründen normalerweise nur in den Systempfaden `/usr/local/lib`, `/usr/lib` usw. gesucht. Um das Testprogramm aus Teilaufgabe d) starten zu können, muss der Lader durch Setzen der Umgebungsvariable `LD_LIBRARY_PATH` (siehe **ld.so(8)**) angewiesen werden, auch im aktuellen Arbeitsverzeichnis nach Bibliotheken zu suchen. Hierfür muss das Programm folgendermaßen aufgerufen werden:
  `LD_LIBRARY_PATH=. ./jbuffer-test-dynamic`

Hinweise zur Abgabe:

Erforderliche Dateien: `jbuffer.c` (6 Punkte), `sem.c` (6 Punkte), `Makefile` (2 Punkte)
Bearbeitung: Einzeln
Bearbeitungszeit: 10 Werktage (ohne Wochenenden und Feiertage)
Abgabezeit: 17:30 Uhr