

Aufgabe 8: jbuffer (12.0 Punkte)

Programmieren Sie eine Bibliothek *jbuffer* (job buffer), 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 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, 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/i4sp/pub/aufgabe8* zur Vermeidung von Über- und Unterlaufsituations, 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 *__sync_bool_compare_and_swap()*, 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 Vorgabezeichnis.

c) Statisches Binden

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.

d) Dynamisches Binden

Erweitern Sie das Makefile so, dass das Ringpuffer- und das Semaphor-Modul zu einer Programmmbibliothek *libjbuffer-dynamic.so* zusammengebunden werden und unter Verwendung dieser Bibliothek ein dynamisch gebundenes Testprogramm *jbuffer-test-dynamic* erstellt wird. Die beiden Bibliotheksmodule sollen hierbei als *Position-Independent Code* übersetzt werden – Sie müssen also für die 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.
- Die Pthread-Funktionen sind in einer speziellen Funktionsbibliothek (*libpthread*) zusammengefasst, die Sie beim Komplizieren bzw. Binden Ihres Programms mit angeben müssen (Option *-pthread*).
- *__sync_bool_compare_and_swap()* ist eine sogenannte Builtin-Funktion des GCC, die vom Compiler auf entsprechende atomare CPU-Instruktionen abgebildet wird. Die Dokumentation finden Sie unter http://gcc.gnu.org/onlinedocs/gcc-4.7.2/gcc/_005f_005fsync-Builtins.html.
- 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.
- Beim Laden von dynamisch gebundenen Programmen werden die benötigten Bibliotheken aus Sicherheitsgründen normalerweise nur in den Systempfaden */usr/local/lib* und */usr/lib* gesucht. Um das aktuelle Arbeitsverzeichnis in die Liste der Suchpfade aufzunehmen, muss bei der Ausführung die Umgebungsvariable *LD_LIBRARY_PATH* entsprechend gesetzt werden: *LD_LIBRARY_PATH=.. ./jbuffer-test-dynamic*

Hinweise zur Abgabe:

Erforderliche Dateien: *jbuffer.c*, *sem.c*, *Makefile*

Bearbeitung: Einzeln

Bearbeitungszeit: 10 Werkstage

Abgabetermin: 17:30 Uhr