Übungsblatt 7

Abgabe bis spätestens 21.12.2023 in Stud.IP

Allgemeiner Hinweis

Das in StudIP zur Verfügung gestellte Archiv `ueb7-vorgabe.zip` enthält TeX-Quellen, die Ihr bei Bedarf für die Bearbeitung von Aufgabe 2 nutzen könnt.

Aufgabe 1 (4 Punkte)

Warteschlangen/Puffer in Gerätecontrollern sollen dabei helfen, das jeweilige Gerät von der CPU zu entkoppeln. Dies sollt Ihr anhand einer Beispielrechnung illustrieren:

Ein gedachter Gerätecontroller kann ein angeschlossenes Gerät mit einer Datenrate von 400000 Bit/s versorgen. Die zu sendenden Daten nimmt der Gerätecontroller vom Betriebssystem entgegen (das die Daten typischerweise von einem Anwendungsprozess erhält). Der Controller verwendet *Double-Buffering*, d. h., der nachfüllende Prozess kann in einen Puffer schreiben, während der Gerätecontroller Daten aus einem zweiten Puffer liest. Erst, wenn dieser Puffer leer ist, wird auf den anderen Puffer umgeschaltet. Die Dauer für das Umschalten kann in der Rechnung vernachlässigt werden.

Erreicht der Controller beim Versenden der Daten aus dem Puffer die Low-Watermark, löst er einen Interrupt aus, damit der zuständige Anwendungsprozess vom Betriebssystem aufgeweckt wird und neue Daten an den Controller gesendet werden. Das Aufwecken des Prozesses samt Nachfüllen der Warteschlange im Controller dauert vereinfachend immer 120 µs.

Welche effektive Datenrate (in Bit/s) zum angeschlossenen Gerät ergibt sich in folgenden Fällen?

a) Größe des Puffers: 1 Byte. Low-Watermark: 0 Byte.
b) Größe des Puffers: 50 Byte. Low-Watermark: 0 Byte.
c) Größe des Puffers: 100 Byte. Low-Watermark: 8 Byte.
d) Welchen Wert (in ganzen Bytes) müsste die Low-Watermark **mindestens** haben, damit die Netto-Datenrate des Geräts erreicht wird?

Stellt zusätzlich zu den Ergebnissen Euren Rechenweg dar. Ihr dürft im letzten Schritt der Berechnung das Ergebnis runden. Das Rechnen mit gerundeten Zwischenergebnissen führt zu Punktabzug. Bedenkt bei euren Berechnungen, dass in der Warteschlange immer mit ganzen Bytes gearbeitet wird!

Aufgabe 2 (3 Punkte)

Zwei voneinander unabhängige Prozesse A und B arbeiten nebenläufig zueinander die beiden nachfolgend skizzierten Programmstücke ab. Alle Systemaufrufe sind erfolgreich, d. h. es treten keine Fehler bei den Dateioperationen auf.

Prozess A

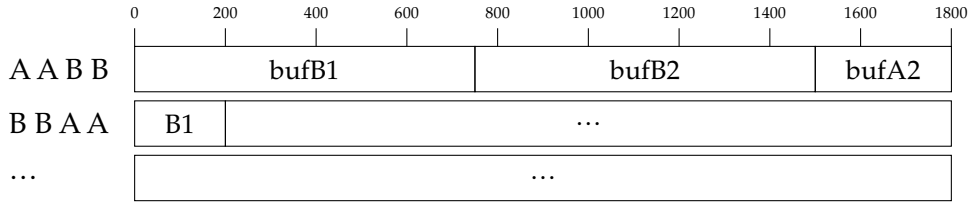
```
int fd = open("/home/ti2/23/ueb7.md", ...);
lseek(fd, 200, SEEK_CUR);
write(fd, bufA1, 1200); /* A1 */
lseek(fd, 1350, SEEK_SET);
write(fd, bufA2, 450); /* A2 */
close(fd);
```

Prozess B

```
int fd = open("/home/ti2/23/ueb7.md", ...);
write(fd, bufB1, 750); /* B1 */
write(fd, bufB2, 750); /* B2 */
close(fd);
```

a) Wieso ist das Ergebnis der Ausführungen nicht deterministisch?

b) Welche verschiedenen Ergebnisse kann es für den Inhalt der Datei geben? Skizziert die möglichen Reihenfolgen entsprechend der unten angefügten Grafik. (Ihr könnt davon ausgehen, dass `write()` atomar arbeitet. Der Buffer Cache bzw. mögliche Auswirkungen durch diesen müssen nicht bedacht werden. Fügt soviele Zeilen hinzu wie notwendig.)



Aufgabe 3 (3 Punkte)

Erklärt das Erzeuger-/Verbraucher-Problem anhand eines geeigneten Szenarios in der physischen Welt (außerhalb eines Computers). **Bezieht Euch dabei auf die in der Vorlesung vorgestellten Konzepte**. Erläutert, an welchen Stellen das Szenario möglicherweise nicht ganz geeignet ist, das Problem zu beschreiben.

a) Wie sieht das Szenario aus (kurze Beschreibung der wesentlichen Eigenschaften)? Wo gibt es hier einen kritischen Abschnitt? Welche Ressource muss geschützt werden? Welche Probleme können entstehen?

b) Wie könntet Ihr Abhilfe schaffen, also was muss in dem kritischen Abschnitt gewährleistet sein? Wie realisiert Ihr das in Eurem Szenario?

c) Welches neue Problem kann durch einen schlecht konzipierten Schutzmechanismus entstehen? Wie verhindert Ihr es in Eurem Szenario?

Abgabe

Bis 23:59 Uhr am 21.12.2023 digital in StudIP. **Es gelten die vereinbarten Scheinbedingungen (siehe StudIP).** Bitte beachtet unsere ergänzenden Hinweise ebenda.

Ladet die abgaberelevanten Dateien in DoIT hoch.

Eure Ansätze und der gewählte Lösungsweg müssen nachvollziehbar sein. Achtet insofern auf eine saubere Dokumentation im Quelltext und im abgegebenen PDF-Dokument. Benennt alle von Euch verwendeten Quellen, auch Zusammenarbeit mit anderen Gruppen und verwendete Unterlagen aus früheren Jahrgängen.

Programmieraufgaben sind im Zweifel in C++20 zu entwickeln. Die Korrektheit der Lösung bzw. deren Grenzen sind grundsätzlich in der Abgabe nachzuweisen. Dies geschieht neben der

Dokumentation des Programmcodes in den Quelldateien und zusätzlich in dem abgegebenen PDF-Dokument mit der Lösungsbeschreibung durch geeignete Tests, deren Auswahl und Eignung begründet werden müssen. Als **Referenzplattform** gelten die Linux-Rechner im Rechnerpool in MZH E0.