ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert
Fabian Berstecher

2. Februar 2018
Blatt 14

Systemnahe Software I (WS 2017/2018)

Abgabe bis zum 9. Februar 2018

Lernziele:

- Atomizität und gegenseitiger Ausschluss im Umgang mit Dateien

Aufgabe 15: Eine kleine Auktion

Ihre Aufgabe ist es, ein Programm zur Durchführung einer kleinen Auktion zu schreiben (nennen wir es `bid`). Das Programm soll von jedem Auktionsteilnehmer aufgerufen werden und das Auktionsverzeichnis und den Namen des Bieters als Parameter erhalten. Darauf hin gibt das Programm in einer Schleife den aktuellen Gebotsstatus (aktuelles Höchstgebot und den Namen des Bieters) aus und bietet dem Benutzer die Möglichkeit, selbst Gebote abzugeben. Der Benutzer kann dann wahlweise einen Betrag eingeben oder auch nur ENTER drücken, um eine aktuelle Gebotsinformation zu erhalten.

Hier ist ein beispielhafter Programmablauf:

```
thales$ bid /tmp/auction Max
No bid yet.
Your bid: 100
Ok.
100 by Max
Your bid:
120 by John
Your bid: 130
Sorry, someone else was faster.
140 by Alice
Your bid: 150
Ok.
150 by Max
Your bid:
```

Das aktuelle Gebot und der Name des Bieters soll dabei in einer Datei mit dem Namen *current_bid* in dem als Parameter übergebenen Verzeichnis abgelegt werden. Es sind dabei folgende Richtlinien einzuhalten:

- Wenn diese Datei existiert, muss sie immer ein gültiges Gebot enthalten (Höhe des Gebots und Name des Bieters), etwa in der Form „100 by Max“.
- Ein exklusiver Zugang auf die Datei wird mit Hilfe der temporären Datei *current_bid.tmp* erreicht. D.h. es darf immer nur ein Prozess diese Datei erzeugen und beschreiben. Der exklusive Zugang darf nur kurzfristig in Anspruch genommen werden.
- Die Datei mit dem aktuellen Gebot darf nur dann (atomar!) aktualisiert werden, wenn sie entweder vorher noch nicht existierte oder sichergestellt ist, dass das neue Gebot höher ist als das alte.
- Man muss davon ausgehen, dass mehrere Programme gleichzeitig versuchen, Gebote abzugeben. Über diese Programme ist nichts bekannt, außer dass sie das nach den hier aufgeführten Regeln tun.
- Es ist sicherzustellen, dass keine zwei Bieter mit dem gleichen Namen arbeiten. Dazu sollte exklusiv eine nach dem Bieter benannte Datei im gemeinsamen Verzeichnis erzeugt werden.

Hinweis: Die Vorgehensweise ist recht ähnlich zu der in den Vorlesungsbeispielen *unique.c* und *unique2.c*. Während dort jedoch zuerst mit Hilfe der temporären Datei ein exklusiver Zugang gesucht wird, bevor über die Aktualisierung nachgedacht wird, läuft es bei dieser Aufgabe anders ab. Dem interaktiven Benutzer ist nur ein älteres Gebot bekannt, das möglicherweise in der Zwischenzeit aktualisiert worden ist. Wenn er beschließt, ein neues Gebot zu geben, muss zuerst der exklusive Zugang erreicht werden, um dann festzustellen, ob die Aktualisierung überhaupt zulässig ist. Wenn sich dann herausstellt, dass das aktuelle Gebot höher ist, muss der Aktualisierungsversuch abgebrochen und die temporäre Datei wieder entfernt werden. Der Benutzer ist dann auf das neue, höhere Gebot hinzuweisen. Nur wenn das neue Gebot tatsächlich höher ist, darf die Zieldatei atomar aktualisiert werden.

Dies entspricht dem Muster vieler atomarer Operationen, die nur dann durchgezogen werden, wenn eine bestimmte Bedingung erfüllt ist.

Reichen Sie bitte Ihre Lösung mit folgendem Kommando ein:

```
thales$ submit ss1 15 bid.c
```

Viel Erfolg!