Dr. Johannes Mayer
Axel Blumenstock
Christian Ehrhardt

Allgemeine Informatik III / Systemnahe Software I (WS 2005/2006)

Abgabetermin: 14. Februar 2006

Schlussverkauf (10 Punkte)

Da Sie ja nun den gesamten Stoff dieser Vorlesung im Kopf haben, sich andererseits Ihren verdienten Urlaub finanzieren müssen, schreiben Sie nun noch ein kleines Programm, mit dem Sie u.a. Ihren (hoch begehrten) Vorlesungsbegleiter versteigern können.[1]
Dieses Programm (nennen wir es `auction`) soll von jedem Auktionsteilnehmer aufgerufen werden und als einzigen Parameter den *Artikelnamen* erhalten. Daraufhin gibt das Programm in einer Schleife den aktuellen Gebotsstatus aus (mindestens den Betrag und Zeitpunkt des Höchstgebots sowie den Namen des zugehörigen Bieters) und erfragt vom Benutzer ein (weiteres) Gebot. Der Benutzer kann dann wahlweise einen Betrag eingeben oder auch nur ENTER drücken, um eine aktuelle Gebotsinformation zu erhalten.
Die Gebotshistorie wird in einem Verzeichnis namens `/tmp/auction/`*Artikelname* gehalten. Dieses Verzeichnis soll angelegt werden, falls es noch nicht existiert. Für jedes akzeptierte Gebot erzeugt das Programm darin eine leere Datei, die nach der Gebotshöhe in Cent benannt ist. Der Name des Bieters sowie der Gebotszeitpunkt sind damit automatisch als Eigenschaften dieser Datei vermerkt. Nutzen Sie zum Anlegen dieser Datei den Systemaufruf `open()`, und zwar so, dass er fehlschlägt, wenn eine Datei gleichen Namens bereits existiert. Auf diese Weise kann der bei einer Angebotskollision Unterlegene sofort informiert werden. Ermitteln Sie den Gebotsstatus durch Maximumbildung über die numerisch interpretierten Dateinamen.

Beispiel

Unter Berücksichtigung eines Ablaufzeitpunkts (Zusatzaufgabe) könnten wir etwa folgenden spannenden Auktionsverlauf erleben:

```
$ whoami
miri
$ ./auction Vorlesungsbegleiter
Angebot Vorlesungsbegleiter endet in 60 Sekunden
  es liegen noch keine Gebote vor.
Gebot (in Cent) eingeben oder <ENTER> fuer Aktualisierung: 5

Angebot Vorlesungsbegleiter endet in 51 Sekunden
  derzeitiges Hoechstgebot (Miri Sübel um 10:25:39): 0.05 Euro
Gebot (in Cent) eingeben oder <ENTER> fuer Aktualisierung:

Angebot Vorlesungsbegleiter endet in 32 Sekunden
  derzeitiges Hoechstgebot (Frank Reich um 10:25:55): 3.12 Euro
Gebot (in Cent) eingeben oder <ENTER> fuer Aktualisierung: 517

Gebot liegt nicht ueber aktuellem Hoechstgebot
Angebot Vorlesungsbegleiter endet in 12 Sekunden
  derzeitiges Hoechstgebot (Resi Stenz um 10:26:16): 14.20 Euro
Gebot (in Cent) eingeben oder <ENTER> fuer Aktualisierung: 1567

Angebot Vorlesungsbegleiter
  wurde ersteigert von Frank Reich fuer 19.99 Euro

$ ls -l /tmp/auction/Vorlesungsbegleiter
-rw-------    1 resi     users           0 2006-02-14 10:26 1420
-rw-------    1 frank    users           0 2006-02-14 10:26 1999
-rw-------    1 frank    users           0 2006-02-14 10:25 312
-rw-------    1 miri     users           0 2006-02-14 10:25 5
```

[1] Sollten Sie die Laufbahn zum Papst anstreben, empfiehlt sich möglicherweise, noch ein paar Jahre mit der Versteigerung zu warten.

Zusatzaufgabe (5 Punkte)

Sorgen Sie nun noch dafür, dass die Auktionen zu einem festen Zeitpunkt nach Beginn (markiert durch das Anlegen des Gebotsverzeichnisses) ablaufen. Die verbleibende Zeit soll nun ebenfalls im Rahmen des Gebotsstatus ausgegeben werden. Nach Ablauf sind keine weiteren Gebote mehr möglich, und das Programm wird mit einer Meldung des Auktionsergebnisses beendet.
Für das Hinterlegen des Zeitpunkts des Auktionsbeginns suchen Sie sich ein geeignetes Dateisystemobjekt aus `;-)`
Machen Sie sich schließlich noch Gedanken, wie das System besser vor Manipulation geschützt werden könnte. Die Auktionsteilnehmer sollten keine Angebote löschen können (auch nicht die eigenen) und auch keinen Einfluss auf den Endzeitpunkt der Auktion haben.

Fingerzeige zur Umsetzung

Sie können Ihr Programm beispielsweise mit folgenden Funktionen strukturieren:

`char *createOfferDir(char *offerName)` stellt sicher, dass das Verzeichnis für den gegebenen Artikelnamen existiert. Achten Sie darauf, dass die Anwender (etwa mit Punkten oder Schrägstrichen im Artikelnamen) keinen Unfug anstellen können `;-)`

`int getMaxBid(char *offerDir)` ermittelt das Maximum unter den numerisch interpretierten Dateinamen in `offerDir`.

`int getOfferStatus(char *offerDir, int bid, struct stat *statP)` liefert Informationen zum Gebot der Höhe `bid`.

`int printOfferStatus(char *offerDir, char *offerName)` gibt den Gebotsstatus auf der Standardausgabe aus.

`void placeBid(char *offerDir, int bid)` überprüft, ob das Gebot `bid` das aktuelle Höchstgebot überschreitet und legt ggf. eine entsprechende Datei im Gebotsverzeichnis an. Eine perfekte Synchronisation ist an dieser Stelle nicht erforderlich.

`void run(char *offerName)` realisiert das zyklische Ausgeben des Gebotsstatus und das Einlesen des nächsten Gebots.

Nützliche Bibliotheksfunktionen und System Calls

- `struct passwd *getpwuid(int uid)` schlägt in `/etc/passwd` die Informationen nach, die zur Benutzer-ID `uid` gehören. Dazu zählt insbesondere der Login- und ggf. der tatsächliche Name des Benutzers.

- `struct tm *mtime = localtime(int time)` erhält als Parameter die Zahl der Sekunden seit 1970 (wie in den Zeitfeldern von `struct stat` zu finden) und zerlegt sie gemäß der aktuellen Zeitzone in die gewohnen Felder für Stunde, Minute, etc.

- `int open(char *pathName, int flags)` öffnet die durch `pathName` gegebene Datei. Der zweite Parameter ist ein bitweises ODER verschiedener Flags, die über das genaue Verhalten bestimmen. Bei Erfolg wird ein Filedeskriptor zurückgegeben, über den die Datei beim nachfolgenden `close()` identifiziert wird.

  Dies sind Systemaufrufe; die (lesenswerte) Dokumentation findet sich also in Kapitel 2 der Manualseiten.

- `int mkdir(char *pathName, int mode)` legt das Verzeichnis names `pathName` mit den angegebenen Rechten an. Achtung: Die Rechte neu angelegter Dateisystemobjekte werden mit der bitweisen Negation der `umask` UND-verknüpft. Sollten also die Zugriffsrechte eines neuen Verzeichnisses nicht Ihren Vorstellungen entsprechen, können Sie entweder `umask` mit dem Systemaufruf `umask()` ändern, oder Sie rufen nach dem Anlegen des Verzeichnisses `int chmod(char *pathName, int mode)` auf.

Viel Erfolg!