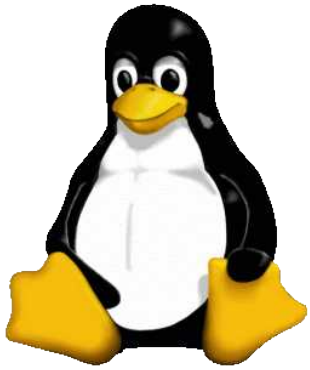
lxconf – LunxBOX Config

Administrations-Scripte für den LunaBOX-Server



Projektarbeit ScriptSprachen

NTA FH Isny
12. Info

David Mayr

1 Inhaltsverzeichnis

2 Einführung

LunaBOX Network Solutions[1] ist ein kleines Webhosting-Unternehmen des Autors mit einem dedizierten 64bit-Server in einem Berliner Hochsicherheits-Rechenzentrum.

In dem hier beschriebenen Projekt geht es darum, auf diesem bestehenden WebHosting-Serversystem mit Hilfe eines Perl-Scripts einfach neue Kunden-Konten (Domains, Benutzer, Speicherplatz, ...) anlegen zu können.

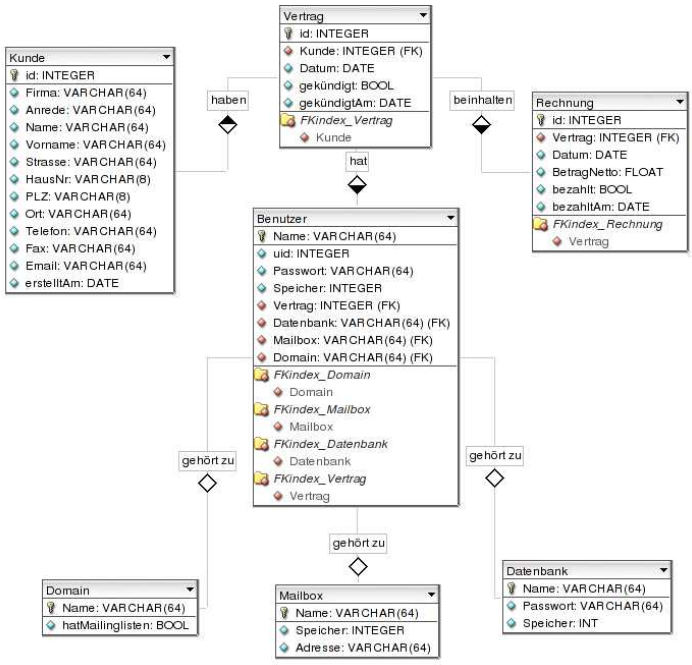
Dabei muss zwischen zwei Arten von "Kunden-Konten" unterschieden werden:

- Kunden-Konten **mit eigener Domain**,
  sog. Hauptkonto/-benutzer oder Domainuser
- Kunden-Konten **ohne eigene Domain**,
  gehören immer zu einer bestehenden Domain und sind Zusatz-Benutzer

Was diese beiden Arten genau unterscheidet ist weiter unten beschrieben.

Zusätzlich soll diese Arbeit ein zuvor von mir angefertigtes Projekt, in dem ein Kundenverwaltungssystem[2] im Fach Datenbanken erstellt wurde, erweitern. In dem damaligen Projekt wurde eine graphische Oberfläche in QT erstellt, mit der sich die Kundendaten[3] in einer MySQL-Datenbank verwalten lassen. Die Datenbank-Struktur ist rechts zu sehen.

Das neue Perl-Script soll neu angelegte Kunden und Aufträge auch in dieser Kundendatenbank speichern, sodass die Daten auch in der graphischen Oberfläche verfügbar sind.



1 http://lunabox.de
2 http://davey.de -> Studium -> Projekte -> Datenbanken / bzw. http://davey.de/db.html
(zum besseren Verständnis wird empfohlen die Dokumentation dieses Projekts zuvor zu lesen.)
3 Tabellen der Datenbank: Kunden, Verträge, Rechnungen, Benutzer, Mailboxen, Domains, Datenbanken

3 Voraussetzungen

Auf dem LunaBOX-Server laufen folgende Dienste, die hier in diesem Zusammenhang von Bedeutung sind:

- LDAP-Server — für System-Benutzer/-Gruppen und eMail-Routing
- MySQL-Datenbank — für Kundendaten und Datenbanken der Kunden
- Apache mit PHP — als Web-Server für die Kunden-Domains
- Postfix — als SMTP-Server, stellt eMails zu
- Cyrus-IMAPd — IMAP- & POP-Server, verwaltet eMails / Mailboxen
- Bind DNS — primärer DNS-Server für Kunden-Domains
- AWStats — erstellt Statistiken über Web-Zugriffe
- LogRotate — komprimiert und archiviert log-Dateien

Des weiteren ist beim Anlegen neuer Konten neben der Konfiguration von Diensten noch folgendes nötig:

- Dateisystem-Quota — begrenzt den für Kunden nutzbaren Speicherplatz
- Homeverzeichnis — muss angelegt werden

Da die Erklärung aller am System beteiligten Dienste viel zu umfangreich für dieses kleine Projekt wäre, wird an dieser Stelle darauf verzichtet. Im wesentlichen reicht zum Verständnis des Projekts auch folgende Information zusätzlich zum Quellcode.

Bei vielen der o.g. Dienste besteht die Konfiguration lediglich daraus, eine bestehende Konfigurations-Vorlage (ASCII-Datei) mit bestimmten Platzhaltern – z.B. `###_DOMAIN_###` für den Domainnamen – einzulesen, Platzhalter zu ersetzen und in eine neue Datei zu schreiben. Diese Datei landet in einem Unterordner des zu konfigurierenden Dienstes, der diese wiederum in seine Konfiguration per `include` einbindet.

4 Umsetzung

Bei der Umsetzung wurde als Programmiersprache Perl gewählt und alles in einer Datei `/etc/lxconf/lxconf` untergebracht, auf die der Symlink `/usr/local/bin/lxconf` zeigt.

Alle anderen Dateien, die dieses Projekt betreffen sind in eigenen Ordner unter `/etc/lxconf/` untergebracht. Die Ordnerstruktur gestaltet sich wie rechts abgebildet.

Die wichtigsten zwei Ordner sind `vorlagen` und `jobs`.

4.1 Grundstruktur

Um mögichst flexibel zu bleiben, wurde die Hauptaufgabe des Scripts – neue Konten anzulegen – in zwei Teile unterteilt: Aufträge erstellen und Aufträge ausführen.

Das Erstellen eines Auftrags führt dazu, dass zum einen die eingegeben Daten in der Kundendatenbank (MySQL) für die graphische Oberfläche gespeichert werden, und zum anderen eine spezielle Job-Datei zur späteren Ausführung angelegt wird.

Sinn dieser Teilung ist, dass Aufträge später evtl. auch von anderen Programmen (z.B. PHP-Bestellformular) angelegt werden können, und nach Prüfung durch den Administrator mit diesem Script auch direkt ohne Umwege ausgeführt werden können.

Zusätzlich können mit diesem Script auch neue Kunden in der Datenbank angelegt werden.

Je nachdem, um welche Art Kunden-Konto es sich handelt, sind verschiedene Schritte zum ausführen eines neuen Auftrags nötig:

- Kunden-Konten **mit eigener Domain**
  - LDAP-Benutzer und -Gruppe anlegen
  - Homeverzeichnis mit entsprechenden Eigentümer-Rechten und Verzeichnissen erstellen und Platzhalter-WebSeite kopieren.
  - Dateisystem-Quota (Speicherplatzbeschränkung) vergeben
  - WebServer-Konfiguration erweitern und PHP-Wrapper anlegen
  - LogRotate-Konfiguration erweitern
  - AWStats-Konfiguration erweitern
  - DNS-Eintrag für Domain erstellen

- SMTP-Server Konfiguration erweitern
- Mailbox anlegen und Quota vergeben
- Je nach Kundenwunsch evtl. MySQL-Datenbank anlegen
- Je nach Kundenwunsch evtl. Mailinglisten aktivieren

- Kunden-Konten **ohne eigene Domain**
  - LDAP-Benutzer anlegen
  - Homeverzeichnis mit entsprechenden Eigentümer-Rechten und Verzeichnissen erstellen
  - Dateisystem-Quota (Speicherplatzbeschränkung) vergeben
  - Mailbox anlegen und Quota vergeben

4.2 Hauptmenü

Das Hauptmenü wurde so gestaltet, dass es jederzeit leicht erweiterbar ist. Es erhält seine Menüpunkte und den Namen der zugehörigen Perl-Funktionen aus der Konfiguration am Anfang der Datei.

```
my $menulist = {
      "kunde_neu"         => "Kunde anlegen",
      "auftrag_neu"       => "Auftrag / Vertrag anlegen",
      "auftrag_aufuehren" => "Auftrag ausführen"
};
```

Zusätzlich zu diesen Punkten sind durch die verwendete Funktion `ask()` noch die weiter unten beschriebene Listenfunktionen aufrufbar. Damit können sowohl im Hauptmenü, als auch an fast jeder anderen Stelle des Programms folgende Listen über die Tastenkürzel k,v,d und a ausgegeben werden: Kunden, Verträge, Domains und Auftragsdateien.

4.3 Hauptfunktionen

Hauptfunktionen werden direkt über das Menü aufgerufen.

4.3.1 Kunde anlegen

Mit diesem Menüpunkt wird für alle Felder der Tabelle `Kunde` in der Kunden-Datenbank die weiter unten beschriebene Funktion `ask()` aufgerufen und in die Hash-Referenz `$kunde` gespeichert.

```
my $kunde = {
      "id"       => "",    # Kunden-Nr/-id
      "Firma"    => "",    #
      "Anrede"   => "",    #
      "Name"     => "",    #
      "Vorname"  => "",    #
      "Strasse"  => "",    #
      "HausNr"   => "",    #
      "PLZ"      => "",    #
      "Ort"      => "",    #
      "Telefon"  => "",    #
      "Fax"      => "",    #
      "Email"    => "",    # alternative eMailadresse
      "erstelltAm" => ""   # Erstellungsdatum wird autom. mit datum() gesetzt
};
```

Bei der Eingabe der `id` wird die nächste freie Kundennummer vorgeschlagen.

Anschliessend werden alle Elemente des Hashs in die Datenbank-Tabelle `Kunde` übertragen. Somit stehen die Eingaben sofort dem Eingangs erwähnten graphischen Kundenverwaltungssystem zur Verfügung.

4.3.2 Auftrag anlegen

Um einen Auftrag anzulegen sind folgende Daten nötig, die zunächst abgefragt und in der Hash-Referenz `$job` gespeichert werden:

```
my $job = {
      "MKdomain" => 1,    ## MAKE Domain - Domain mit diesem Auftrag erstellen?
      "MKdb"     => 0,    ## MAKE DB -- Datenbank mit diesem Auftrag erstellen?
      "v_id"     => 999,   # Vertrags-Nr/-id – nächste freie wird vorgeschlagen
      "k_id"     => 999,   # Kunden-Nr/-id – nächste freie wird vorgeschlagen
      "u_id"     => 999,   # User-ID (unix-uid) – nächste freie wird vorgeschlagen
      "u_name"   => "",    # User-Name
      "u_pass"   => "",    # User-Passwort
      "u_quota"  => 100,   # User FS-Quota
      "m_name"   => "",    # Mailbox Name
      "m_quota"  => 100,   # Mailbox Speicher
      "m_adr"    => "",    # Mailbox Haupt eMail-Adresse
      "d_name"   => "",    # Domain Name
      "d_mlist"  => 0,     # Domain hatMailinglisten?
      "db_name"  => undef, # MySQL-Datenbank Name
      "db_pass"  => undef, # MySQL-Datenbank Passwort
      "db_quota" => 10     # MySQL-Datenbank SoftQuota
};
```

Anschliessend werden die eingegebenen Daten in den verschiedenen Datenbank-Tabellen (`Vertrag`, `Benutzer`, `Mailbox`, `Domain`, `Datenbank`) gespeichert und zusätzlich eine sog. job-Datei angelegt.

Diese Job-Datei enthält die Hash-Referenz `$job` in der Zuweisungsform, so dass der Inhalt dieser Datei von Perl evaluiert werden kann und hinterher die Werte in `$job` wiederzufinden sind:

```
##### JOB-DATEI für lxconf [testdomain123.com] START ####

$job={
  "MKdb" => '1',
  "MKdomain" => '1',
  "d_mlist" => '1',
  "d_name" => 'testdomain123.com',
  "db_name" => '2061',
  "db_pass" => 'ganzgeheim',
  "db_quota" => '10',
  "k_id" => '1',
  "m_adr" => 'webmaster@testdomain123.com',
  "m_name" => 'testdomain123.com',
  "m_quota" => '100',
  "u_id" => '2061',
  "u_name" => 'testdomain123.com',
  "u_pass" => 'ganzgeheim',
  "u_quota" => '100',
  "v_id" => '1048'
};

##### JOB-DATEI für lxconf [testdomain123.com] ENDE ####
```

4.3.3 Auftrag ausführen

Beim Menüpunkt "Auftrag ausführen" muss zunächst die gewünschte Job-Datei gewählt werden, wobei die zuletzt erstellte automatisch vorgeschlagen wird. Dann wird erst der Inhalt der Datei zur Kontrolle am Bildschirm ausgegeben. Diese Kontrolle soll den Administrator davor schützen unbekannten Perl-Code zu interpretieren, da das System diese Auftragsdatei später von möglicherweise unzuverlässigen Quellen (PHP-Script) bekommten könnte.

Ist der Bediener mit dem Inhalt der Job-Datei einverstanden, wird er von Perl evaluiert. Dadurch stehen dem Script die wichtigsten Daten des früher angelegten Auftrags in Form der Hash-Ref. `$job` zur Verfügung.

Nachdem noch einige zusätzliche Daten aus der Datenbank, wie z.B. der Kunden-Name aus der Datenbank nachgeholt wurden beginnt das eigentliche Ausführen des Auftrags: das Anlegen der einzelnen Komponenten eines Kontos. Alle diese Schritte werden von Hilfsfunktionen ausgeführt, die weiter unten kurz zusammengefasst sind.

4.4 Listenfunktionen

Um z.B. bei der Eingabe der Kundennummer zu einem neuen Vertrag nicht extra in der Datenbank nachsehen zu müssen, welche Kundennummer der Kunde XY hat, wurden in die allgemeine Abfrage-Funktion `ask()` ein paar Tastenkürzel zum Aufruf von Listenfunktionen eingebaut.

- k -> Kundenliste: Kundenliste aus der Kunden-Datenbank anzeigen
- v -> Vertragsliste: Vertragsliste aus der Kunden-Datenbank anzeigen
- d -> Domainliste: Domainliste aus der Kunden-Datenbank anzeigen
- a -> Auftragsliste: Aufträge anhand vohandener Job-Dateien zeigen

4.5 Hilfsfunktionen, Allgemein

Hier eine kurze Zusammenfassung der verwendeten allgemeinen Hilfsfunktionen:

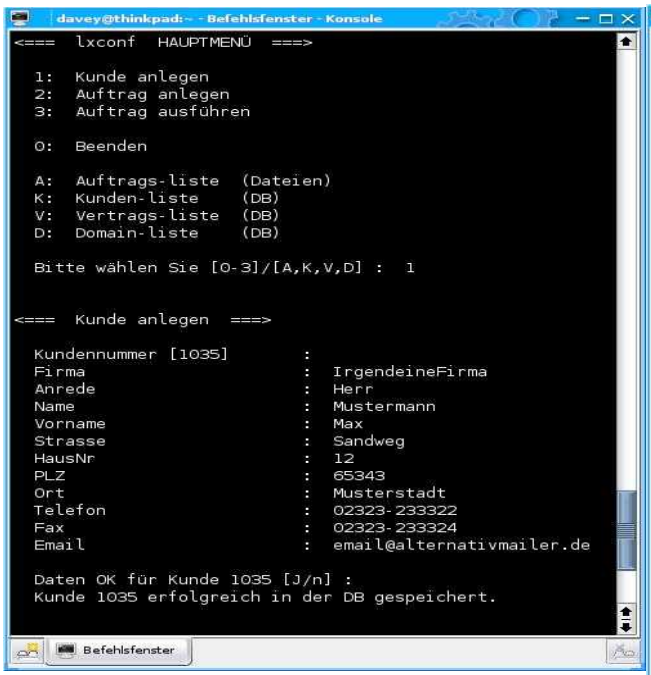
```
ask($$);                # Abfrageroutine (meldung, defaultwert)
datum();                # Datum zurückgeben
dns_serial();           # DNS-Serial in der Form YYYYMMDDxx erzeugen
domain2uid($);          # Gibt UID eines DomainUsers zurück (domainname)
slappasswd($);          # Passwort für LDAP mit slappasswd konvertieren
hole_kundendaten($);    # Alle Daten einer Kunden-Nr aus DB ermitteln

next_uid($);            # Nächste UID zurückgeben (bool:istDomainUser)
next_kdnr();            # Nächste Kunden-id zurückgeben
next_vnr();             # Nächste Vertrags-id zurückgeben

check_domain($);        # Prüfen, ob Domain in DB existiert (domainname)
check_kunde($);         # Prüfen, ob Kunde in DB existiert (kunden-id)
check_vertrag($);       # Prüfen, ob Vertrag in DB existiert (vertrags-id)

pass_vorschlag();       # Passwort-Vorschläge ausgeben
auftrag_in_datei($);    # Auftrag in Datei schreiben ($job)
auftrag_in_db($);       # Auftrag in DB sichern ($job)
```

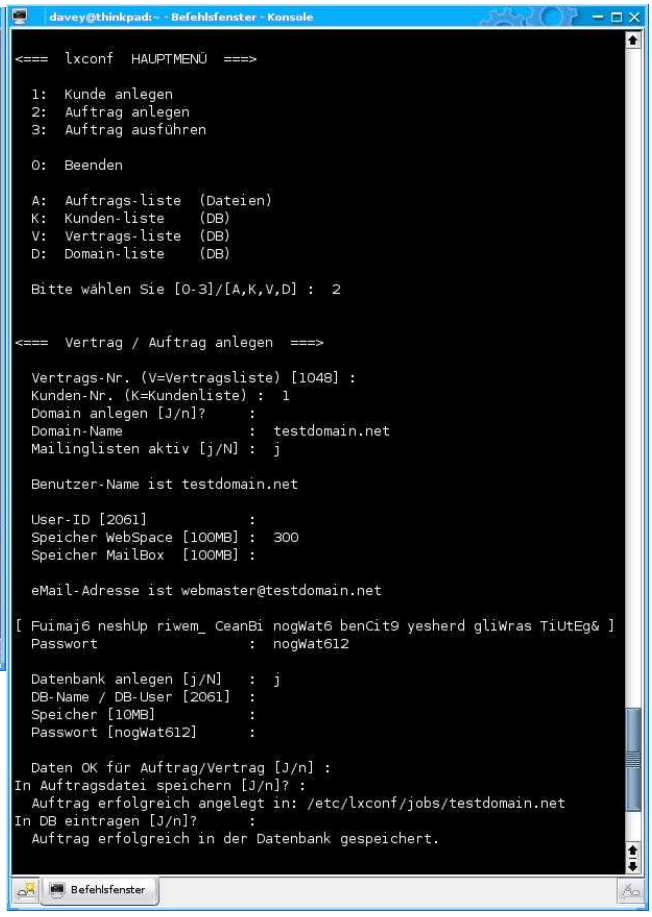
4.6 Hilfsfunktionen, Auftrag ausführen

Hier eine kurze Zusammenfassung der verwendeten Hilfsfunktionen, die nur von der Hauptfunktion "Auftrag ausführen" aufgerufen werden. Alle Funktionen verlassen sich auf die Daten in der Hash-Referenz `$job`.

```
vorlage2config($$$);  # Allgemeine Hilfsfunktion für die folgenen Funktionen,
                      # zum Erstellen von Konfigurationsdateien aus einer
                      # Vorlagendatei mit Platzhaltern. Parameter:
                      # (vorlagenname, zielverzeichnis, zieldateiname)
add_ldap_domain();    # Trage Domain-User/-Gruppe in LDAP-Server ein
mkhomedir_domain();   # Homeverzeichnis für Domain-/Haupt-User anlegen
add_ldap_user();      # Trage Zusatz-User in LDAP-Server ein
mkhomedir_user();     # Homeverzeichnis für Zusatz-User anlegen
add_mailbox();        # Cyrus-Mailbox anlegen
add_quota();          # Dateisystem-Quota (Speicher) einstellen
mkconfig_apache();    # Apache-Config für diese Domain erweitern
mkconfig_logrotate(); # LogRotate Config-Datei erstellen
mkconfig_awstats();   # AWStats Config-Datei erstellen
mkconfig_dns();       # DNS-Config erweitern
mkconfig_postfix();   # Postfix-Config erweitern
add_mysql();          # MySQL-DB anlegen
add_mailman();        # Mailman Mailingliste aktivieren
```

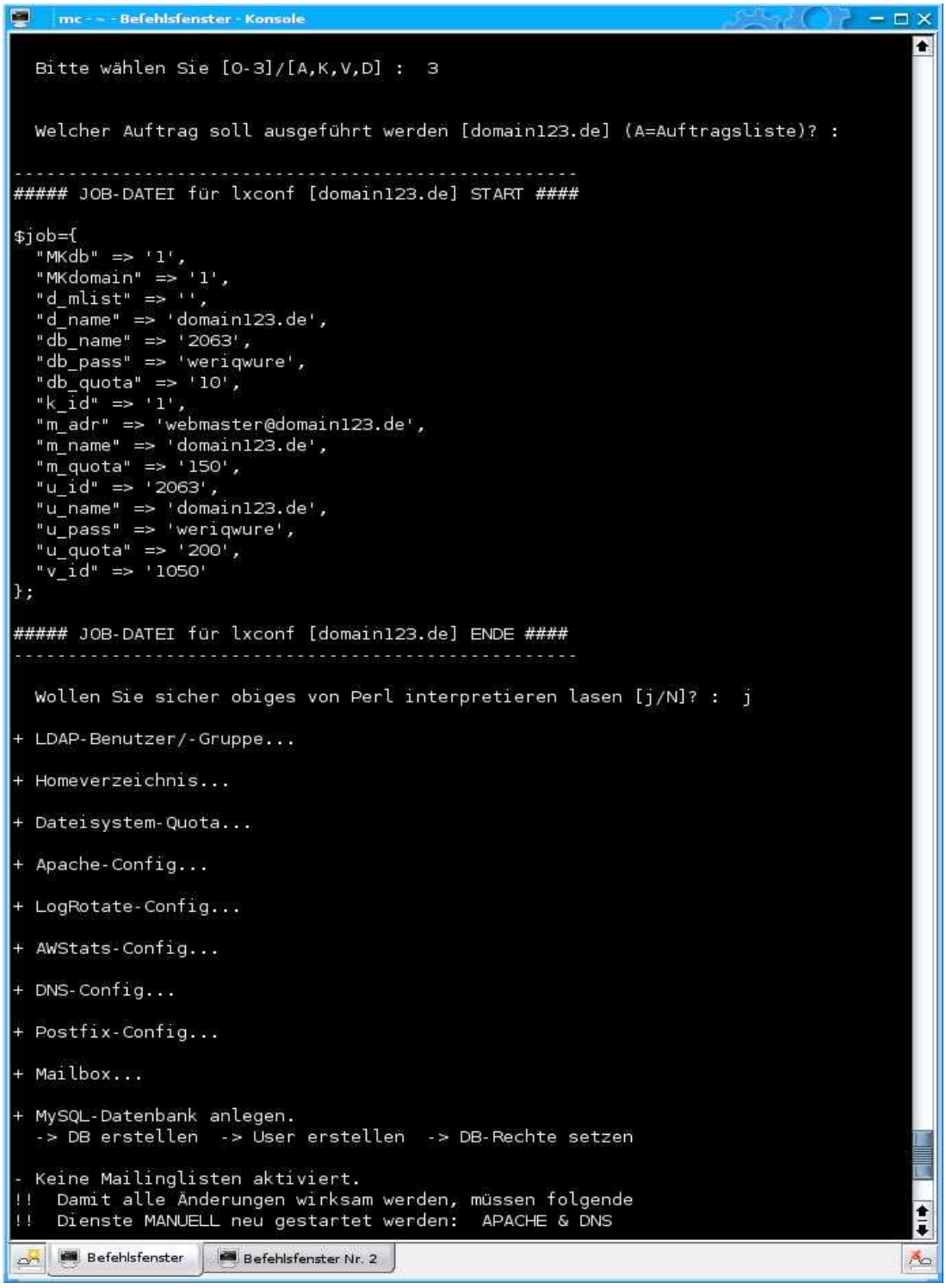
5 Beispiel-Sitzungen





Links: Kunde anlegen

Rechts: Auftrag anlegen



Auftrag ausführen für eine Domain mit Datenbank

6 Fazit

Durch weniger als 2000 Zeilen Perl-Code lässt sich eine menge Arbeit beim Anlegen von Kunden-Konten abnehmen und zusätzlich die Daten aller Kunden immer aktuell in einer graphischen Oberfläche angenehm verwalten.

Nach dem Erstellen eines Kunden-Kontos mit Domain ist nur noch die Anmeldung bei einem Registrar nötig, so dass der gesamte Bestellvorgang einer Kunden-Domain in weniger als 3 Minuten möglich ist.

7 Quellenangaben

7.1 Literatur

- Vorlesungsscript ScriptSprachen v. Marcus Schäfer
- O'Reilly: Perl, kurz und gut

7.2 Internet

Diverse Seiten (gefunden mit google), u.a.:

- http://www.elektronikschule.de/~grupp/perlcgi/mysql/
- http://www.infos24.de/perle/handbuch/22_dbi_modul.htm
- http://www.oreilly.com/catalog/msql/chapter/ch10.html
- http://www.rexswain.com/perl5.html