

# Order/Lager/Fakturering för Linuxsystem

Teknisk manual

Version 0.47

2004-10-31

```
*****
*
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*****
```

# Innehåll

Målgrupp.....	10
Filosofin bakom konstruktionen .....	11
OLFIX grundkonstruktion.....	11
Strukturen för det grafiska gränsnittet.....	12
Programnamnsval.....	12
Kataloger.....	12
Säkerhet.....	13
Dokumentation.....	13
Förutsättningar för OLFIX.....	14
Installationsanvisningar.....	15
Bibliotek.....	15
Installation.....	15
Skapa databasen olfix.....	16
Script.....	16
Behörighet.....	17
Resursfilen .olfixrc.....	18
Administration.....	19
Upplägg av nya användare.....	19
Tilldela behörighet till användare.....	19
Tillägg av nya program.....	19
Tillägg av nya funktioner.....	19
GUI-program i OLFIX .....	20
OLFIXW.....Grafiskt menyprogram.....	22
Behörighetskrav:.....	24
ADDBARW.....Nytt bokföringsår.....	25
Behörighetskrav:.....	27
ADDBETVW.....Nytt betalningsvillkor.....	28
Behörighetskrav:.....	30
ADDFNCW.....Ny funktion .....	31
Behörighetskrav:.....	33
ADDFTGW.....Nya företagsdata .....	34
Behörighetskrav:.....	37
ADDINKW.....Registrera inköpsorder.....	38
Behörighetskrav:.....	43
ADDKSTW.....Nytt kostnadsställe.....	44
Behörighetskrav:.....	46
ADDKTOW.....Nytt Konto .....	47
Behörighetskrav:.....	49
ADDKUW ..... Ny kund.....	50
Behörighetskrav:.....	53
ADDLEVW ..... Ny leverantör.....	54
Behörighetskrav:.....	56
ADDLEVPW ..... Ny standardleveransplats.....	57
Behörighetskrav:.....	59
ADDLEVSW ..... Nytt leveranssätt.....	60
Behörighetskrav:.....	62
ADDLEVWV ..... Nytt leveransvillkor.....	63
Behörighetskrav:.....	65
ADDRGTW.....Ny behörighet .....	66
Behörighetskrav:.....	68

ADDTXTW.....Ny text till TEXTREG .....	69
Behörighetskrav:.....	71
ADDUSRW.....Ny användare .....	72
Behörighetskrav:.....	74
ADDVALW.....Ny valuta .....	75
Behörighetskrav:.....	77
ATTBETW.....Lista obetalda leverantörsfakturer .....	78
Behörighetskrav:.....	81
BOKFORSW.....Bokföring standard.....	82
Behörighetskrav:.....	86
BYTFTGW ..... Byta företag.....	87
Flera företag.....	87
CHGBARW.....Ändra bokföringsårsdata.....	89
Behörighetskrav:.....	91
CHGBETVW.....Ändra betalningsvillkor.....	92
Behörighetskrav:.....	95
CHGFTGW.....Ändra företagsdata.....	96
Behörighetskrav:.....	98
CHGLEVW.....Ändra leverantörsdata.....	99
Behörighetskrav:.....	101
CHGUSRW.....Ändra användardata.....	102
Behörighetskrav:.....	104
CHGKTOW.....Ändra kontodata.....	105
Behörighetskrav:.....	107
CHGKUW.....Ändra kunddata.....	108
Behörighetskrav:.....	110
CHGVALW.....Ändra valutadata.....	111
Behörighetskrav:.....	113
DAGBOKW.....Dagbok .....	114
Behörighetskrav:.....	116
Exempel.....	117
DELRGTW.....Radera behörighet .....	118
Behörighetskrav:.....	120
DELXTW.....Radera text. ....	121
Behörighetskrav:.....	123
DELUSRW.....Radera användare .....	124
Behörighetskrav:.....	127
DELVALW.....Radera valuta .....	128
Behörighetskrav:.....	130
DSPINKW.....Visa en inköpsorder.....	131
Behörighetskrav:.....	133
DSPKSTW.....Visa kostnadsställdata.....	134
Behörighetskrav:.....	136
DSPKUW.....Visa kunddata.....	137
Behörighetskrav:.....	139
DSPLEVW.....Visa leverantörsdata.....	140
Behörighetskrav:.....	142
DSPUSRW.....Visa användardata.....	143
Behörighetskrav:.....	145
DSPVALW.....Visa valutainformation.....	146
Behörighetskrav:.....	148
HUVBOKW.....Huvudbok.....	149

Behörighetskrav:.....	151
LEVFAKTW.....Registrera fakturor .....	152
Behörighetskrav:.....	156
LSTFNCW.....Lista funktioner .....	157
Behörighetskrav:.....	159
LSTINKW.....Beställningsstock .....	160
Behörighetskrav:.....	162
LSTKSTW.....Lista kostnadsställen.....	163
Behörighetskrav:.....	165
LSTKTOW.....Lista konton .....	166
Behörighetskrav:.....	168
LSTKUW.....Lista kunder.....	169
Behörighetskrav:.....	171
LSTLEVSW.....Lista leveranssätt.....	172
Behörighetskrav:.....	174
LSTLEVWV.....Lista leveransvillkor.....	175
Behörighetskrav:.....	177
LSTRGTW.....Lista behörigheter.....	178
Behörighetskrav:.....	180
LSTUSRW.....Lista alla användare .....	181
Behörighetskrav:.....	183
LSTVALW.....Lista alla valutor.....	184
Behörighetskrav:.....	186
OLFIXHLP.....Online hjälp.....	187
PRTINKW.....Skriv ut beställning.....	188
Behörighetskrav:.....	191
RPTGENW.....Rapportgenerator.....	192
Behörighetskrav:.....	194
SDOLISW.....Saldolista.....	195
Behörighetskrav:.....	202
Flödesschema.....	203
Konsolprogram i OLFIX.....	205
ADMIN.....	206
BOKF .....	212
FTGADM.....	213
OLFIX (konsolprogram) .....	216
REDOV.....	219
Standardrapporter i OLFIX.....	226
Leverantörsfakturer förfallna till betalning.....	226
Kontorapport, endast på skärm.....	226
Leverantörsreskontrarapport, endast skärm.....	226
Huvudbok.....	226
Saldolista.....	226
Funktionerna i OLFIX.....	227
ARADD.....	233
Interface:.....	233
ARCHK.....	235
Interface:.....	235
ARLST.....	236
Interface:.....	236
ARICHK.....	237
ATTBET.....	238

Interface:.....	238
BARADD.....	240
Interface:.....	240
BARCHG.....	242
Interface:.....	242
BARCHK.....	244
Interface:.....	244
BARDSP.....	245
Interface:.....	245
BARFND.....	247
Interface:.....	247
BETADD.....	248
Interface:.....	248
BETCHG.....	249
Interface:.....	249
BETDSP.....	250
Interface:.....	250
BETLST.....	251
Interface:.....	251
DBOKRPT.....	252
Interface:.....	252
FTGADD.....	253
Interface:.....	253
FTGDSP.....	254
Interface:.....	255
FTGLIS.....	256
Interface:.....	257
FTGLST.....	258
Interface:.....	258
FTGUPD.....	259
Interface:.....	259
HBOKRPT.....	260
Interface:.....	260
INKADD.....	261
Interface:.....	262
INKLST.....	263
Interface:.....	264
INKRADD.....	265
Interface:.....	266
INKHDSP.....	267
Interface:.....	268
INKRLST.....	269
Interface:.....	269
KSTADD.....	270
Interface:.....	270
KSTCHK.....	271
Interface:.....	271
KSTDSP.....	272
Interface:.....	272
KSTLST.....	273
Interface:.....	273
KTOADD.....	274

Interface:.....	274
KTOCHG.....	275
Interface:.....	275
KTOCHK.....	277
Interface:.....	277
KTODSP.....	278
Interface:.....	278
KTOLST.....	280
Interface:.....	280
KTORPT.....	281
Interface:.....	282
KTOUPD.....	283
KTOVIEW.....	284
Interface:.....	284
KUADD.....	286
Interface:.....	287
KUCHG.....	288
Interface:.....	289
KUCHK.....	290
Interface:.....	290
KUDSP.....	291
Interface:.....	291
KULST.....	293
Interface:.....	293
LEVADD.....	294
Interface:.....	295
LEVCHG.....	296
Interface:.....	297
LEVDSP.....	298
Interface:.....	298
LEVLST.....	300
Interface:.....	300
LEVSADD.....	301
Interface:.....	301
LEVSDSP.....	302
Interface:.....	302
LEVSLST.....	303
Interface:.....	303
LEVVADD.....	304
Interface:.....	304
LEVVDSP.....	305
Interface:.....	305
LEVVLST.....	306
Interface:.....	306
LRESADD.....	307
Interface:.....	308
LRESRPT.....	309
Interface:.....	310
PRGLST.....	311
Interface:.....	312
PRTAPI.....	313
RGTADD.....	314

Interface:.....	314
RGTCHK.....	315
Interface:.....	315
RGTDEL.....	316
Interface:.....	316
RGTDSP.....	317
Interface:.....	318
RGTLST.....	319
Interface:.....	319
RPTCRE.....	320
Interface:.....	320
SLPADD.....	321
Interface:.....	321
STYRMAN.....	322
Interface:.....	323
TRHDADD.....	324
Interface:.....	324
TRNSADD.....	325
Interface:.....	325
TRNSLST.....	326
Interface:.....	326
TXTADD.....	327
Interface:.....	327
TXTDEL.....	328
Interface:.....	328
TXTDSP.....	329
Interface:.....	329
USERADD.....	330
Interface:.....	330
USERCHG.....	331
Interface:.....	331
USERDEL.....	332
Interface:.....	332
USERDSP.....	333
Interface:.....	333
USERLST.....	334
Interface:.....	334
VALADD.....	335
Interface:.....	335
VALCHG.....	336
Interface:.....	336
VALDEL.....	337
Interface:.....	337
VALDSP.....	338
Interface:.....	338
VALLST.....	339
Interface:.....	339
VERUPD.....	340
Interface:.....	341
WRREC ..... vernr.txt.....	342
Tabeller i OLFIXs databas .....	344
ARTIKELREG.....	346

BOKFAR.....	347
BETVILKOR.....	348
FTGDATA.....	349
INKREG.....	351
INKRADREG.....	352
LEVREG.....	353
LEVRESK.....	354
KSTALLE.....	355
KTOPLAN.....	356
KUNDREG.....	357
LAGERSTELLEREG.....	358
LEVVILLKOR.....	359
LEVSETT.....	360
PROGRAM.....	361
RIGHTS.....	363
STDLEVPLATS.....	364
TEXTREG.....	365
TRANSID.....	366
TRHD.....	367
USR.....	368
VALUTA.....	369
VERHUVUD.....	370
VERRAD.....	371
Credits.....	372
Appendix A.....	373
MySQL databasens fysiska läge.....	373
Installation av MySQL .....	373
MySQL Navigator.....	373
Appendix B.....	374
Uppläggning av användare och behörigheter med konsolprogram.....	374
Appendix C.....	378
Felmeddelanden.....	378
Appendix D.....	380
Manuell skapande av databasen olfix.....	380



## **Målgrupp**

Målgruppen för OLFIX är fåmansföretag och små företag med en eller flera samtidiga användare.

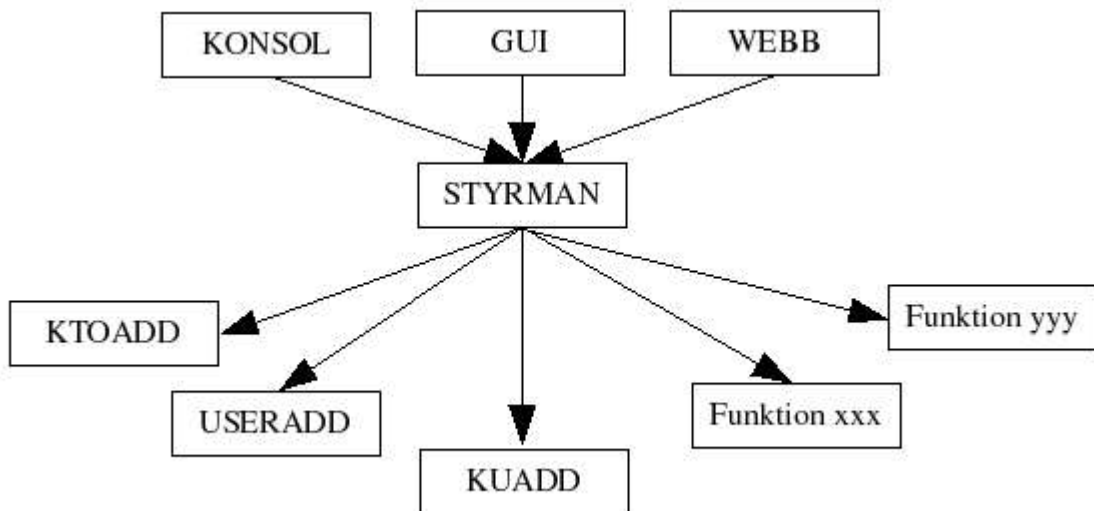
## Filosofin bakom konstruktionen

Det finns flera skäl till valet att använda små programmoduler (här kallade funktioner) .

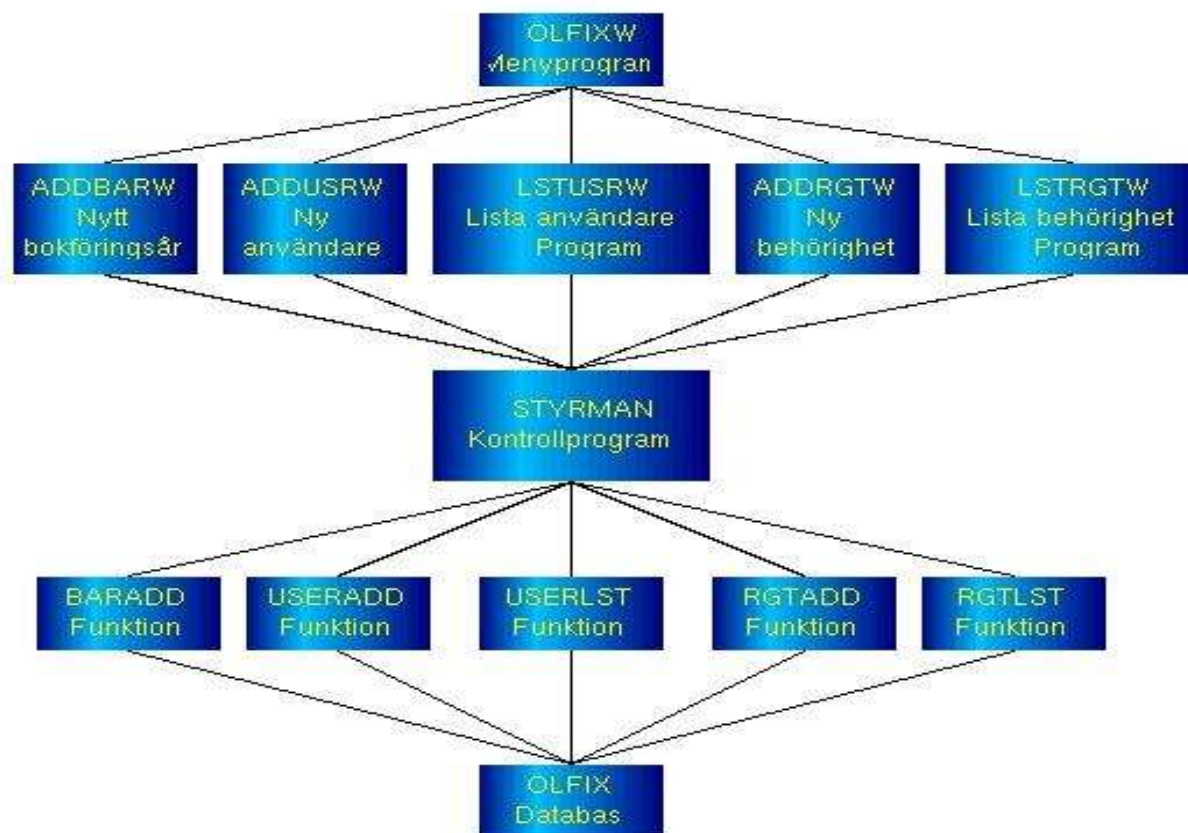
- Det förenklar införandet av nya funktioner.
- Det ger möjlighet till att utveckla program och funktioner i andra programspråk.
- Det underlättar felrättning.
- Det medger, inte minst viktigt, användares möjlighet att modifiera funktionerna efter sitt önskemål.
- Lätt att lägga till nya funktioner. Det krävs inga omfattande omprogrammeringar.

Skriv en ny funktion och registrera den i tabellen TRANSID samt lägg upp behörigheten i tabellen RIGHTS, så vips har man ny funktionalitet i OLFIX.

### OLFIX grundkonstruktion.



## Strukturen för det grafiska gränssnittet.



## Programnamnsval.

Programnamn och namn på funktioner har satts med versaler. Program anropar andra program och funktioner och använder då programnamnen skrivna i versaler. Programnamnen finns också upplagda i databasen så att menyprogrammet OLFIXW kan kontrollera att programmen/funktionerna existerar. Längden på programnamnen får icke överskrida 8 tecken.

## Kataloger.

Alla program måste ligga i samma katalog på grund av att anrop till funktioner och andra program sker med relativt (./PROGRAM) till var det aktuella programmet befinner sig (i vilken katalog).

Eftersom OLFIXW är menyprogram som anropar andra program måste dessa ligga i samma katalog som OLFIXW. Dessa andra program anropar också i sin tur funktioner och program relativt sin egen position.

OLFIX anser vidare att alla filer som tillhör programpaketet OLFIX ska hållas samlade under ett huvudkatalog t ex /usr/local/olfix/ eller /opt/olfix/. Detta görs med programpaketet Acrobat med flera. Det är ett oskick att sprida filer som tillhör programpaketet över hela systemet, därför hålls hela paketet ihop.

## Säkerhet.

I detta fall avser säkerhet informationssäkerhet. I många företag vill man av olika anledningar begränsa de anställdas tillgång till data. Det kan gälla ekonomiska transaktioner som endast ekonomerna ska ha tillgång till. Det kan gälla priser av olika slag så som tillverkningskostnad mm. Därför har OLFIX ett eget behörighetssystem. Behörigheterna läggs upp i programmet ADDRGTW. Men för att förhindra möjligheten att starta ett program i en konsol så måste operativsystemet hjälpa till att förhindra detta.

Genom att ha en användare som heter olfix samt en grupp som också heter olfix och bara tillåter olfix att exekvera alla filerna (utom OLFIXW som alla får exekvera) förhindrar man att användare kan exekvera program via konsol. Se vidare under **Installationsanvisningar**.

## Dokumentation

Dokumentationen är uppdelad på två (tre) delar:

(Utvecklingsdokumentation)

Teknisk manual

Användarmanual

## Förutsättningar för OLFIX

För att kunna använda OLFIX fördras följande:

Operativsystemet **Linux** 2.4.19 eller senare

Programvaran **Qt** version 3.0.5 eller senare (Trolltech AS) finns installerad.

Ifall KDE är installerat så finns automatiskt också Qt.

Databashanteraren **MySQL** version 3.23.52 eller senare (MySQL AB)

KDE:Kugar

KDE:Kspread

Skaffa gärna ett grafiskt administrationsverktyg för MySQL, t ex MySQLCC.

# Installationsanvisningar

Förutsättningen för att kunna installera OLFIX är att MySQL finns installerad på datorn.

## Bibliotek.

OLFIX behöver följande bibliotek:

```
/usr/local/olfix/bin      (Här ska alla binärfiler ligga)
/usr/local/olfix/data
/usr/local/olfix/doc
/usr/local/olfix/doc/helpfiles
/usr/local/olfix/doc/helpfiles/usermanual      (online-hjälp)
/usr/local/olfix/doc/image
/usr/local/olfix/include
/usr/local/olfix/lib
/usr/local/olfix/report   (Här ligger alla templates för rapporter, Kugar)
/usr/local/olfix/script
/usr/local/olfix/sql
/usr/local/olfix/src
/usr/local/olfix/util
```

Följande behörigheter bör du ha:

```
/usr/local/olfix/data  chmod +rw *
/usr/local/olfix/doc   chmod +rw *
/usr/local/olfix/sql    chmod +rw *
/usr/local/olfix/src   chmod +rwx *
```

## Installation.

Se till att du är "root".

```
shell> cd /usr/local
shell> gunzip < /path/to/OLFIX-VERSION.tar.gz | tar xvf -
shell> cp /usr/local/olfix/util/.olfixrc $HOME/.olfixrc
shell> cp /usr/local/olfix/util/start_olfix $HOME/start_olfix
```

För installation med flera användare.

```
shell> cd /usr/local
shell> chown -R olfix olfix
shell> chgrp -R olfix olfix
shell> cd /usr/local/olfix/bin
shell> chmod 0010 *
shell> chmod 2111 OLFIXW
```

## Skapa databasen olfix.

Att skapa och uppdatera olfix databas från början.

Tips!

Om du väljer att **inte** installera OLFIX enligt förslag ska du editera filerna i biblioteken

..../olfix/sql

..../olfix/data

så att de passar dig.

1. Lägg upp användaren (usern) **olfix** i operativsystemet (annars funkar inte anropen till databasen olfix).
2. Se till att bli root.
3. shell> cd /usr/local/olfix/sql
4. Editera filen **add\_user\_DittNamn.sql**

Leta reda på detta avsnitt;

```
)  
VALUES  
(  
  "localhost",  
  "DittUserid",  
  PASSWORD("DittPassword"),  
  "Y",  
  "Y",  
  "Y",
```

Skriv in ditt eget userid, med gemener (små bokstäver), och ditt eget password.

Jag förordar att du utelämnar password och går in i databasen och ändrar password vilket då kommer att bli krypterat.

Detta kan göras med följande procedur:

```
shell>mysql -u root  
shell>use mysql  
shell>update user set password=PASSWORD('DittPassword') where User='DittUserid';  
shell>FLUSH PRIVILEGES;
```

5. Shell> cd /usr/local/olfix/data
6. Editera filen **USRdata.txt**  
"OLFIX", "Olfix Superuser", "IT", "Stab"  
"USERID", "Ditt Namn", "Avd", "Sektion"

## Script.

7. Kopiera filen **/usr/local/olfix/util/.olixrc** till \$HOME.  
**.olfixrc** måste finnas i \$HOME hos alla användare som ska köra OLFIX.  
Ifall du **inte** valt standardinstallation behöver filen editeras så den passar dig.  
**OBS!** VTMP=/tmp/ får icke följas av ett returenstecken. Måste ligga sist i filen.
8. Kopiera filen **/usr/local/olfix/util/start\_olfix** till varje användares hembibliotek (\$HOME).  
**start\_olfix** är ett script för att starta OLFIXW.

### **Behörighet.**

Om du har ett fleranvändarsystem behöver behörigheterna på filerna sättas enligt följande.

#### 9. Behörigheterna på binärfilerna ska göras enligt följande:

Här måste du ha rootbehörighet.

Sätt uid=olfix och gid=olfix

```
shell> cd /usr/local/olfix/bin
```

```
shell> chown olfix *
```

```
shell> chgrp olfix *
```

#### Övriga program

```
shell> chmod 0010 program
```

```
-----x--- 1 olfix olfix 58239 jan 29 05:09 program
```

#### Programmet OLFIXW

```
shell> chmod 2111 OLFIXW
```

```
---x--s--x 1 olfix olfix 68637 feb 10 15:12 OLFIXW
```



## Resursfilen .olfixrc

Filen används i/för OLFIX.

Hur Unix/Linux hanterar "current directory".

En process (ett körande program) har alltid ett "current directory" associerat med sig, och allt som det programmet gör med filer/filsystem utgår från "current directory". Så när man startar från ett konsolfönster är kommandotolkens "current dir" samma som den katalog man 'står' i.

Men när man kör från Konqueror eller startar från ett annat program så blir "current dir" "\$HOME" eftersom det var därifrån man startar X (och därifrån X startade KDE som startar Konqueror). För att säkerställa att alla processer hittar sina "underprogram" låter vi programmen läsa \$HOME/.olfixrc som innehåller path till det bibliotek där OLFIX program ligger (PATH).

WRREC och VERUPD hämtar sökvägen till vernr.txt (VTMP) från \$HOME/.olfixrc.

Filen .olfixrc

```
PATH=/usr/local/olfix/bin/  
HOST=localhost  
DATABASE=olfixtst  
REPORT=/usr/local/olfix/report/  
VTMP=/tmp/
```

**OBS!** Inget "New Line" efter VTMP=/tmp/

Värdet för DATABASE kan skifta. Om det står olfixtst så kommer du att arbeta med testföretaget.

Om värdet är olfix så anger det att du arbetar med det "skarpa", ordinarie företaget.

Detta är inte något du ska ändra själv. Använd programmet BYTFTGW för att byta mellan olika företag..

Filen levereras i /usr/local/olfix/util

Editera filen så den passar dig.

Kopiera filen till \$HOME.

Alla användare som skall ha tillgång till OLFIX ska ha ett exemplar av .olfixrc i sitt hembibliotek, \$HOME.

Om filen **.olfixrc** inte finns i hemmakatalogen så skapas den den första gången man kör OLFIXW.

# Administration

## Allmänt

Normalt är det tänkt att en duktig användare (superuser) ska ta hand om administrationen av OLFIX för att avlasta systemadministratören.

En superuser kan ta hand om upplägg av nya användare i applikationen OLFIX samt administrera behörigheter i OLFIX.

Uppdatering av nya program och nya funktioner är dock av sådan art att det bör skötas av systemadministratören därför finns det inga GUI-program.

## Upplägg av nya användare.

En ny användare registreras med hjälp av programmet ADDUSRW.

## Tilldela behörighet till användare.

För att en användare ska kunna utnyttja OLFIX behöver han/hon behörighet till ett antal funktioner.

Behörigheter tilldelas per funktion **och** per program.

Att använda ett visst program kan innebära behov av behörighet till ett flertal funktioner.

Vilka behörigheter som krävs för ett visst program framgår av respektive program.

Upplägg av behörighet görs med programmet ADDRGTW.

## Tillägg av nya program.

I samband med installation av ny program för OLFIXW (GUI) måste tabellen PROGRAM uppdateras.

Detta görs lämpligast genom att editera filen /usr/local/olfix/data/PROGRAMdata.txt.

Kolumn 1 är ett löpnummer och tillika primary key.

Kolumn 2 anger huvudmeny.

Kolumn 3 anger submeny.

Kolumn 4 är ledtext för programmet.

Kolumn 5 är programnamn.

Därefter exekveras filen /usr/local/olfix/sql/LoadPROGRAM.sql.

## Tillägg av nya funktioner.

När man inför nya funktioner i OLFIX måste tabellen TRANSID uppdateras.

Detta görs med programmet ADDFNCW.

Det kan också göras genom att editera filen /usr/local/olfix/data/TRANSIDdata.txt.

Kolumn 1 är funktionsid/transaktionsid.

Kolumn 2 är beskrivning av transaktionen.

Därefter exekveras filen /usr/local/olfix/sql/LoadTRANSID.sql

## GUI-program i OLFIX

<b>OLFIXW</b>	Huvudprogram för OLFIX, menyprogram.
<b>ADDARW</b>	Lägga upp en ny artikel
<b>ADDBARW</b>	Lägga upp nytt bokföringsår.
<b>ADDBETVW</b>	Lägga upp nytt betalningsvillkor.
<b>ADDFNCW</b>	Lägga upp ny funktion.
<b>ADDINKW</b>	Registrera inköpsorder.
<b>ADDKSTW</b>	Lägga upp nya kostnadställen.
<b>ADDKTOW</b>	Lägga upp nytt konto.
<b>ADDKUW</b>	Lägga upp en ny kund.
<b>ADDLEVPW</b>	Lägga upp en ny standardleveransplats.
<b>ADDLEVSW</b>	Lägga upp ett nytt leveranssätt.
<b>ADDLEVWV</b>	Lägga upp ett nytt leveransvillkor.
<b>ADDLEVW</b>	Lägga upp en ny leverantör.
<b>ADDLPLW</b>	Lägga kompletterande artikeldata per lagerplats.
<b>ADDRGTW</b>	Lägga upp ny behörighet.
<b>ADDTXTW</b>	Lägga upp nya texter i TEXTREG.
<b>ADDUSRW</b>	Lägga upp ny användare.
<b>ADDVALW</b>	Lägga upp en ny valuta.
<b>ATTBETW</b>	Lista leverantörsfakturer förfallna till betalning.
<b>BOKFORSW</b>	Bokföring, registrera verifikat (Standardprogram)
<b>BYTFTGW</b>	Att byta företag.
<b>CHGBARW</b>	Ändra data för bokföringsår.
<b>CHGBETVW</b>	Ändra data för betalningsvillkor.
<b>CHGFTGW</b>	Ändra företagsdata
<b>CHGKTOW</b>	Ändra kontodata.
<b>CHGKUW</b>	Ändra kunddata.
<b>CHGLEVW</b>	Ändra leverantörsdata.
<b>CHGUSRW</b>	Ändra data på användare.
<b>CHGVALW</b>	Ändra information på en valuta.
<b>DAGBOKW</b>	Skriva ut dagboksrapport.
<b>DELRGTW</b>	Ta bort en behörighet från en användare.
<b>DELXTW</b>	Radera texter ur textregistret.
<b>DELUSRW</b>	Ta bort användare samt dennes behörigheter
<b>DELVALW</b>	Ta bort en valuta.
<b>DSPFTGW</b>	Visa företagsdata.
<b>DSPINKW</b>	Visa en inköpsorder/beställning
<b>DSPKSTW</b>	Visa information på ett kostnadställe.
<b>DSPKTOW</b>	Visa information om enstaka konto.
<b>DSPKUW</b>	Visa kundinformation.
<b>DSPLEVW</b>	Visa information om en leverantör.
<b>DSPUSRW</b>	Visa användardata samt dennes behörigheter.
<b>DSPVALW</b>	Visa information om envaluta.

<b>HUVBOKW</b>	Skriva ut huvudboksrapport.
<b>LEVFAKTW</b>	Registrera leverantörsfakturor.
<b>LEVRESKW</b>	Visa obetalda leverantörsfakturor, leverantörsreskontra.
<b>LSTFNCW</b>	Lista funktioner (transaktionstyper)
<b>LSTINKW</b>	Beställningsstock.
<b>LSTKSTW</b>	Lista information på alla kostnadställen.
<b>LSTKTOW</b>	Lista konton.
<b>LSTKUW</b>	Lista kunder på skärm, kundnr och namn.
<b>LSTRGTW</b>	Lista behörigheter.
<b>LSTUSRW</b>	Lista användare.
<b>LSTVALW</b>	Lista alla valutor.
<b>OLFIXW</b>	Grafiskt menyprogram för OLFIX.
<b>PRTINKW</b>	Utskrift av beställning. Använder sig av Kugar.
<b>RPTGENW</b>	Generell rapportgenerator.
<b>RPTKTOW</b>	Kontorapport.
<b>SDOLISW</b>	Saldolista

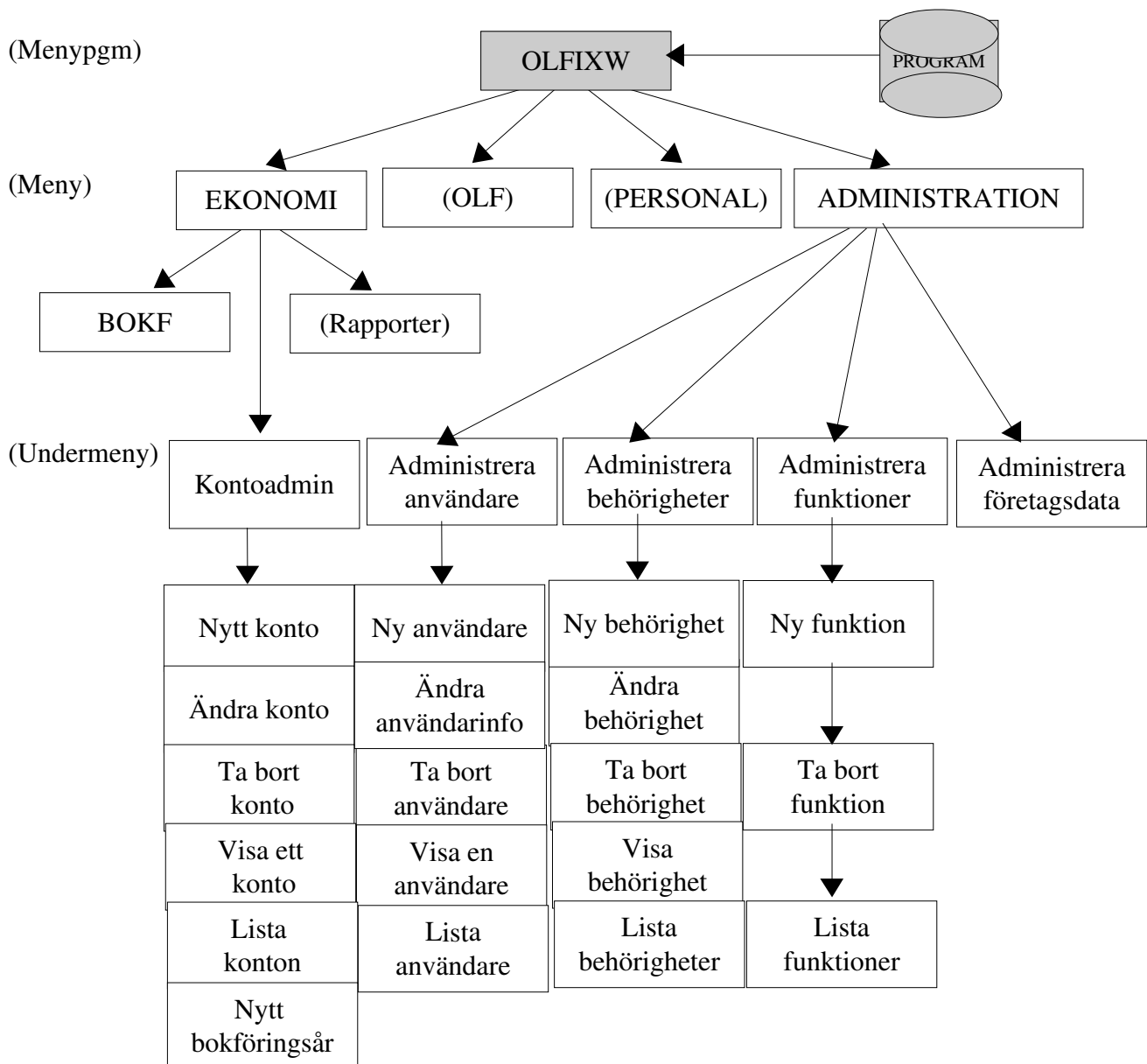
## OLFIXW.....Grafiskt menyprogram

OLFIXW är ett grafiskt menyprogram.

Programmet plockar upp userid från environment.

Genom sin konstruktion kan nya program läggas till i menyn utan omprogrammering.

Nya program läggs in genom registrering i tabellen PROGRAM.



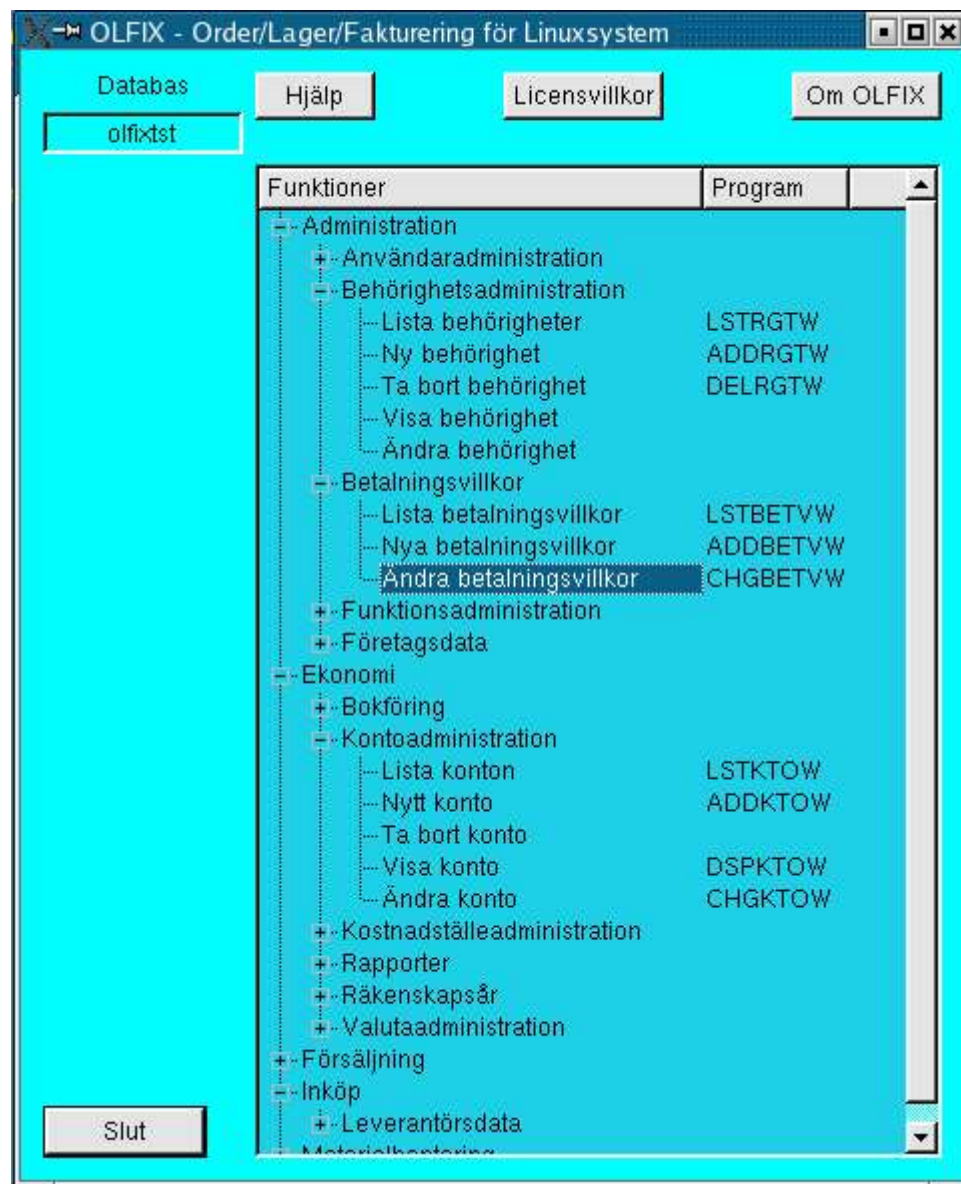


Bild 1.

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Innan OLFIXW startar ett program görs kontroll att användaren har behörighet att använda programmet. Detta gör OLFIXW genom att hämta användarid från environment och sedan jämföra med behörigheten i tabellen RIGHTS. Till sin hjälp att göra detta använder OLFIXW funktionen RGTCHK.

Om användaren inte har behörighet att använda önskat program visas denna messagebox:



Bild 2.

I detta fall har användaren inte behörighet att använda programmet LSTRGTW, men det kan också vara så att användaren inte har behörighet till funktionen RGTLSL som används av programmet LSTRGTW (Lista behörigheter).

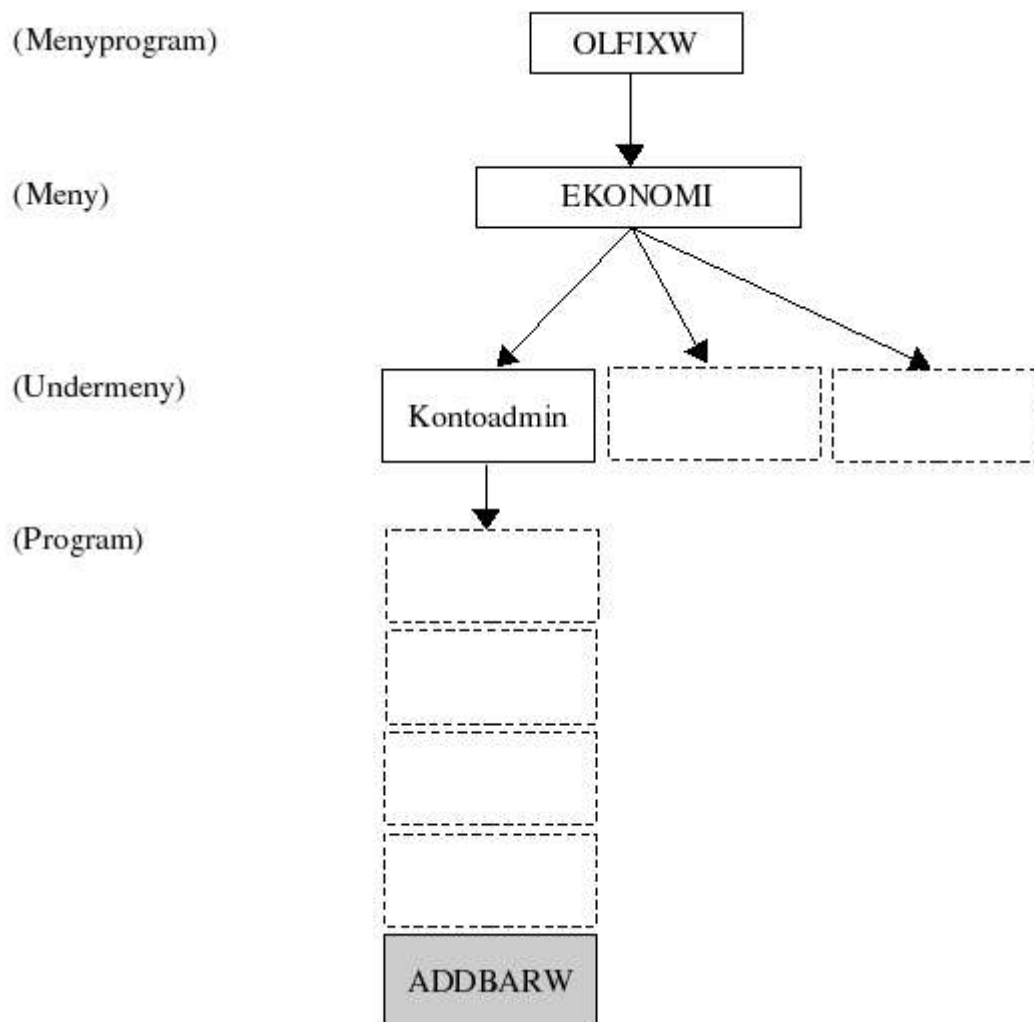
Test av behörigheten görs mot både **programmet** och **funktionen** .

#### **Behörighetskrav:**

För att kunna köra OLFIXW behövs behörighet till PRGLST.

## ADDBARW.....Nytt bokföringsår

ADDBARW, ett grafiskt program för att registrera nytt bokföringsår.





ADDBARW Lägga upp nytt bokföringsår

Bokföringsår: (arid, 2 teck.) Obligatoriskt.

Benämning:

Startdatum: Obligatoriskt.

Slutdatum: Obligatoriskt.

Beskattningsår: Obligatoriskt.

Senaste ver.datum:

Nästa ver.nummer: 1

Kontoplan: Obligatoriskt.

OK Avsluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDBARW.

ADDBARW anropar BARADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("BARADD");               // OLFIX funktion
process->addArgument(arid);
process->addArgument(benamn);
process->addArgument(arstart);
process->addArgument(ar slut);
process->addArgument(arlast);
process->addArgument(beskattar);
process->addArgument(senverdat);
process->addArgument(vernr);
process->addArgument(ktoplan);
```

Detta blir:

```
./STYRMAN userid BARADD arid benamning arstart ar slut arlast beskattningsar
senverdat vernr ktoplan.
```

ARLAST sätts till "N"

### **Behörighetskrav:**

För att kunna köra ADDBARW behövs behörighet till  
PRGLST  
BARADD

## ADDBETVW.....Nytt betalningsvillkor

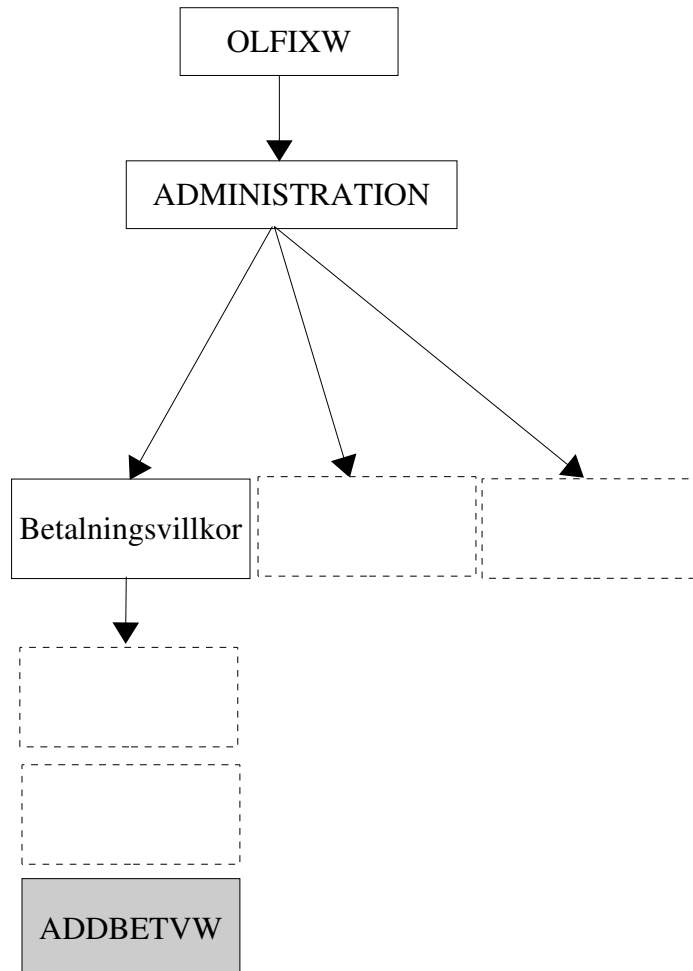
ADDBETVW, ett grafiskt program för att registrera ett nytt betalningsvillkor.

(Menyprogram)

(Meny)

(Undermeny)

(Program)



ADDBETVW Lägga upp nytt betalningsvillkor

Betalningsvillkor: 003

Antal dagar 45

Beskrivning 45 dagar netto

OK Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDBETVW.

ADDBETVW anropar BETADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("BETADD");                 // OLFIX funktion
process->addArgument(betvillk);
process->addArgument(dagar);
process->addArgument(beskrivning);
```

Detta blir:

```
./STYRMAN userid BETADD betvillk dagar beskrivning.
```

### **Behörighetskrav:**

För att kunna köra ADDBETVW behövs behörighet till  
PRGLST  
BETADD

## ADDFNCW.....Ny funktion

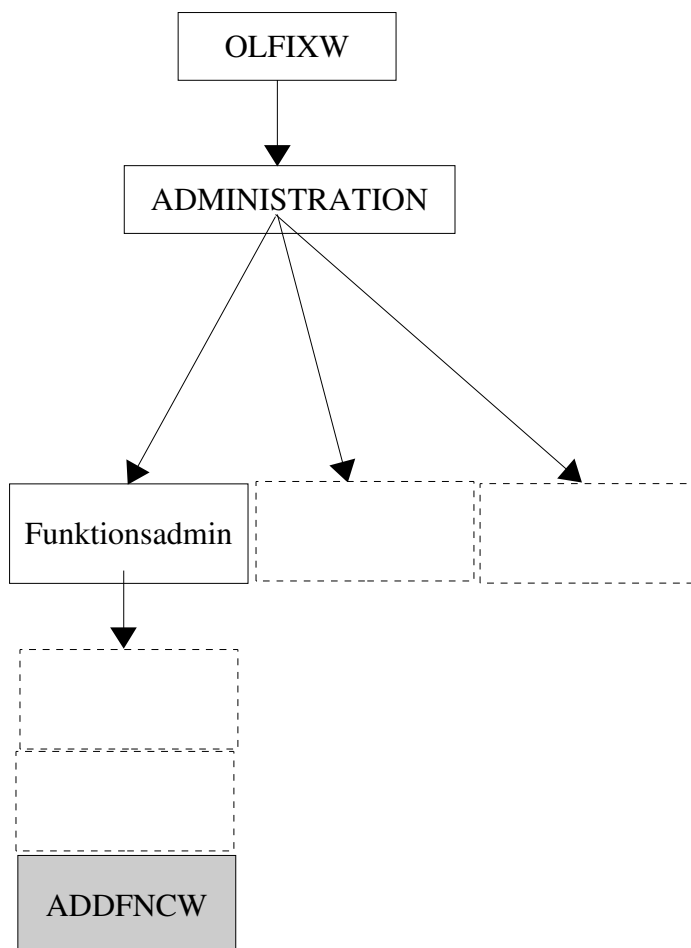
ADDFNCW, ett grafiskt program för att lägga upp en ny funktion.

(Menyprogram)

(Meny)

(Undermeny)

(Program)





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDFNCW.

ADDFNCW anropar TRNSADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // Userid
process->addArgument("TRNSADD");              // OLFIX funktion
process->addArgument(Func);
process->addArgument(LedText);
```

Detta blir:

```
./STYRMAN userid TRNSADD trnsid trnstxt
```

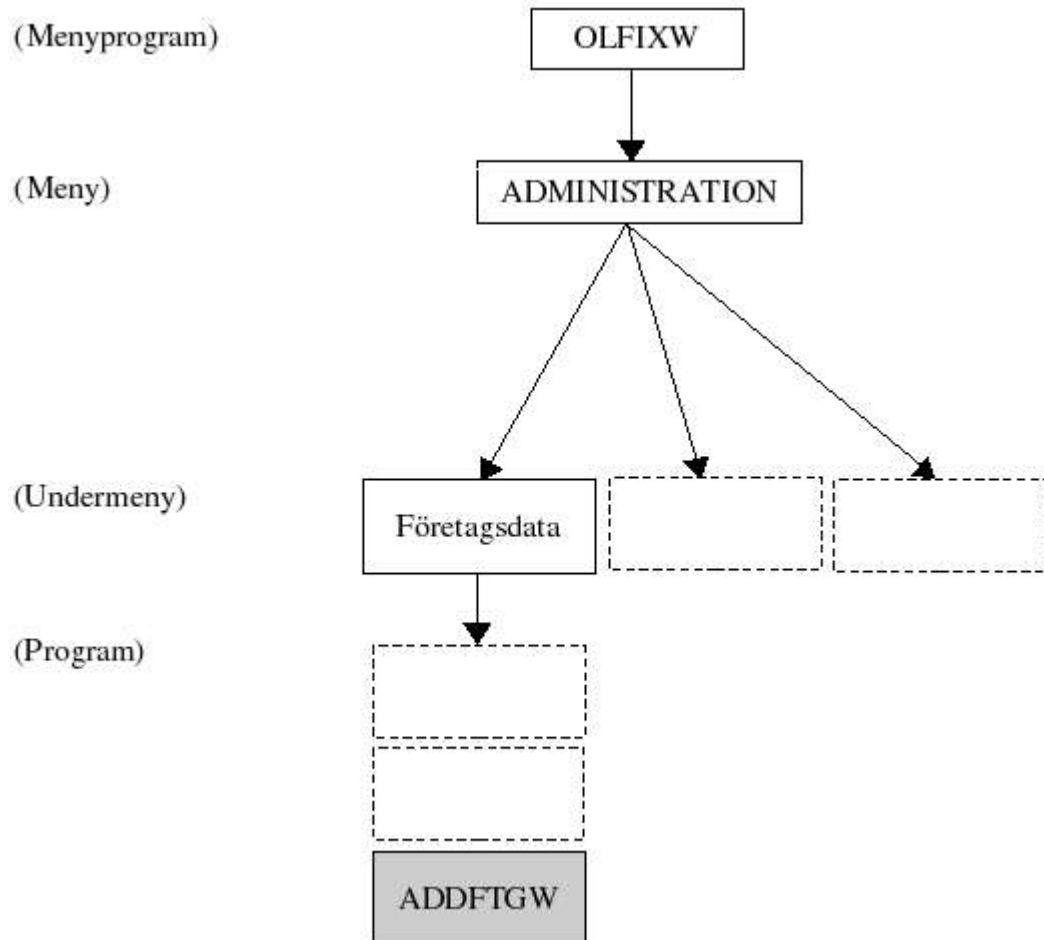
### **Behörighetskrav:**

För att kunna köra ADDFNCW behövs behörighet till  
PRGLST  
TRNSADD



## ADDFTGW.....Nya företagsdata

ADDFTGW, ett grafiskt program för att lägga upp grunddata till företaget.



**Funktionsbeskrivning.**

Vid uppstart läses tabellen FTGDATA och eventuellt befintliga data presenteras. Om inget data finns så skrivs texten (null) ut i fältet, se fälten **Momsats 4** och **Momssats 5**. Genom att fylla i respektive fält och trycka Enter fylls data på och kommer att sparas i tabellen när man klickar på **Spara** .

Uppdateringen sker post för post, se FTGUPD.

ADDFTGW - Nya företagsdata

Företagsnamn:	PROGRAM AB		Organisationsnr:	550101-5555	
Postadress:	Datagatan 2	Postnummer:	199 01	Ort:	DATASTAD
Besöksadress:	Programmerarstigen 3	Postnummer:	198 10	Ort:	DATASTAD
Godsadress:	Godsvägen 31	Postnummer:	198 25	Ort:	DATASTAD
Telefonnummer:	09-109910	Mobiletelefon:	070-991199	Telefax:	09-109919
e-mailadress:	info@progre.se			Telex:	99999
Momssats 1:	25	Momssats 2:	12	Momssats 3:	6
Momssats 4:	(null)	Momssats 5:	(null)		
Momskonto, ingående moms:	2641				
Momskonto, utgående moms:	2611				
Automatkontering J/N :	J				
<div>Spara    Avbryt</div>					

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDFTGW.

ADDFTGW anropar TRNSADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("FTGUPD");                // OLFIX funktion
process->addArgument(posttyp);
process->addArgument(ftgdata);
```

Detta blir:

```
./STYRMAN userid FTGUPD posttyp ftgdata
```

Detta upprepas för varje posttyp:

```
slotUpdateFtgdata("FNAMN",ftgnamn);
slotUpdateFtgdata("FTGNR",ftgnr);
slotUpdateFtgdata("ADR1",postadr);
slotUpdateFtgdata("ADR2",postnr1);
slotUpdateFtgdata("ADR3",postort);
slotUpdateFtgdata("ADR4",besoksadr);
slotUpdateFtgdata("ADR5",postnr2);
slotUpdateFtgdata("ADR6",besoksort);
slotUpdateFtgdata("ADR7",goodsadr);
slotUpdateFtgdata("ADR8",postnr3);
slotUpdateFtgdata("ADR9",godsort);

slotUpdateFtgdata("TFN1",tfnnr);
slotUpdateFtgdata("TFN2",mobiltfnnr);

slotUpdateFtgdata("TFAX",telefax);
slotUpdateFtgdata("TELEX",telex);
slotUpdateFtgdata("EML1",email);

slotUpdateFtgdata("MOMS1",moms1);
slotUpdateFtgdata("MOMS2",moms2);
slotUpdateFtgdata("MOMS3",moms3);
slotUpdateFtgdata("MOMS4",moms4);
slotUpdateFtgdata("MOMS5",moms5);
slotUpdateFtgdata("MOMSI",momsin);           // momkonto ingående moms
slotUpdateFtgdata("MOMSU",momsut);          // momskonto utgående moms
slotUpdateFtgdata("AUTOK",autokonto); // automatisk kontering J/N
```

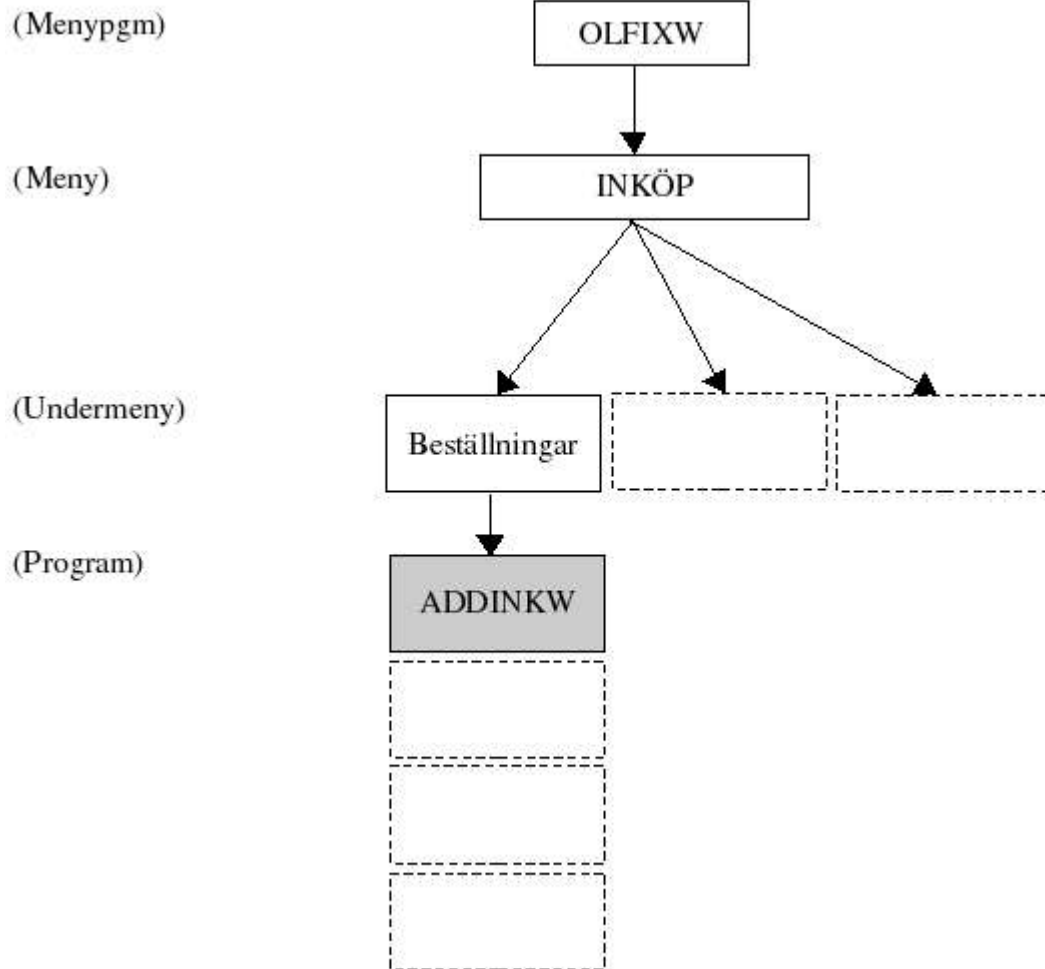
### Behörighetskrav:

För att kunna köra ADDFTGW behövs behörighet till  
PRGLST  
FTGUPD  
FTGLIS



## ADDINKW.....Registrera inköpsorder

ADDINKW, ett grafiskt program för att registrera nya inköpsordrar.  
Programmet plockar upp userid från environment.



**ADDINKW - Registrera inköpsorder**

Leverantörsnr: ..... 126 \* = Obligatoriskt.  
Beställningsnr: ..... 27 Beställningsdatum: ... 2004-02-05

---

**Orderhuvud**

Leverantörens postadress: Mottagarens Leveransadress/Godsadress

Namn: ..... Dataspecialisten AB ..... PROGRAM AB

Adress: ..... Storgatan 1 ..... Verktygsgatan 11

Postnummer: ..... 199 11 ..... 199 97

Postadress: ..... STORSTAD ..... PROGSTAD

Land: ..... Sverige ..... Valuta: SEK

Godsmärke: ..... PELLE

Vår referent: ..... Jan Pihlgren ..... Er referens: ... Ola Norman

Direkttfn: ..... 09-112233 ..... Direktfax: ... 09112239 ..... Best.typ: N Vårt kundnr: ... 567891

Leveransdatum: ..... 2004-02-25 ..... Leveransvillkor: ..... 001 ..... Leveranssätt: 001 Betalningsvillkor: 1

Kommentar: ..... Direkt ..... Eftertext: ..... 001

---

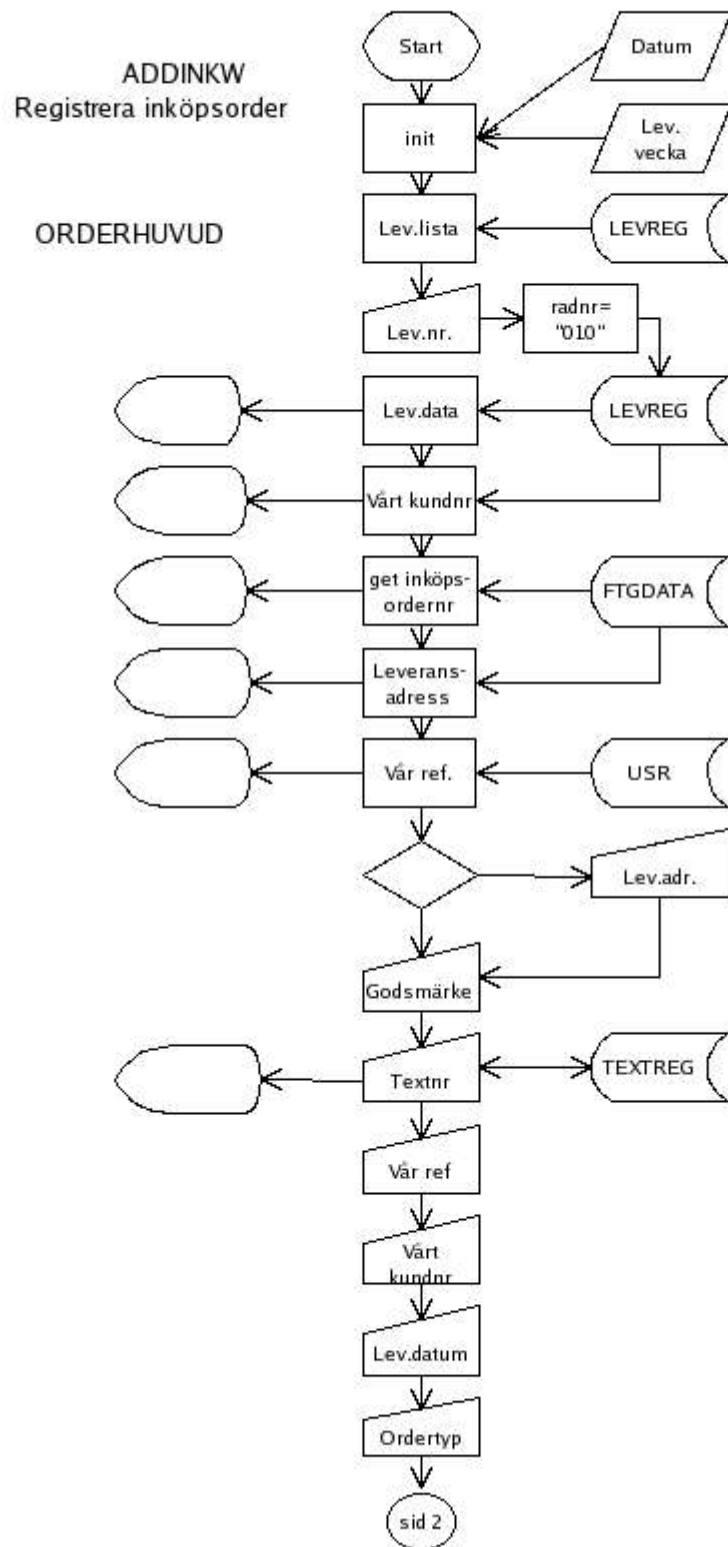
**Detaljrader**

Radnr	Artikelnr *	Benämning	Lev.vecka	Antal *	Pris/st	Summa	Godkänna rad
030	1173-1447	Timerkrets	4093	0.00	5.45		<input type="button" value="Ja"/> <input type="button" value="Nej"/>
010	1173-1175	Spänningsregulator positiv	4093	100	30.00	3000.00	
020	1173-1445	D/A Omvandlare 12-bit	4093	100	95.00	9500.00	

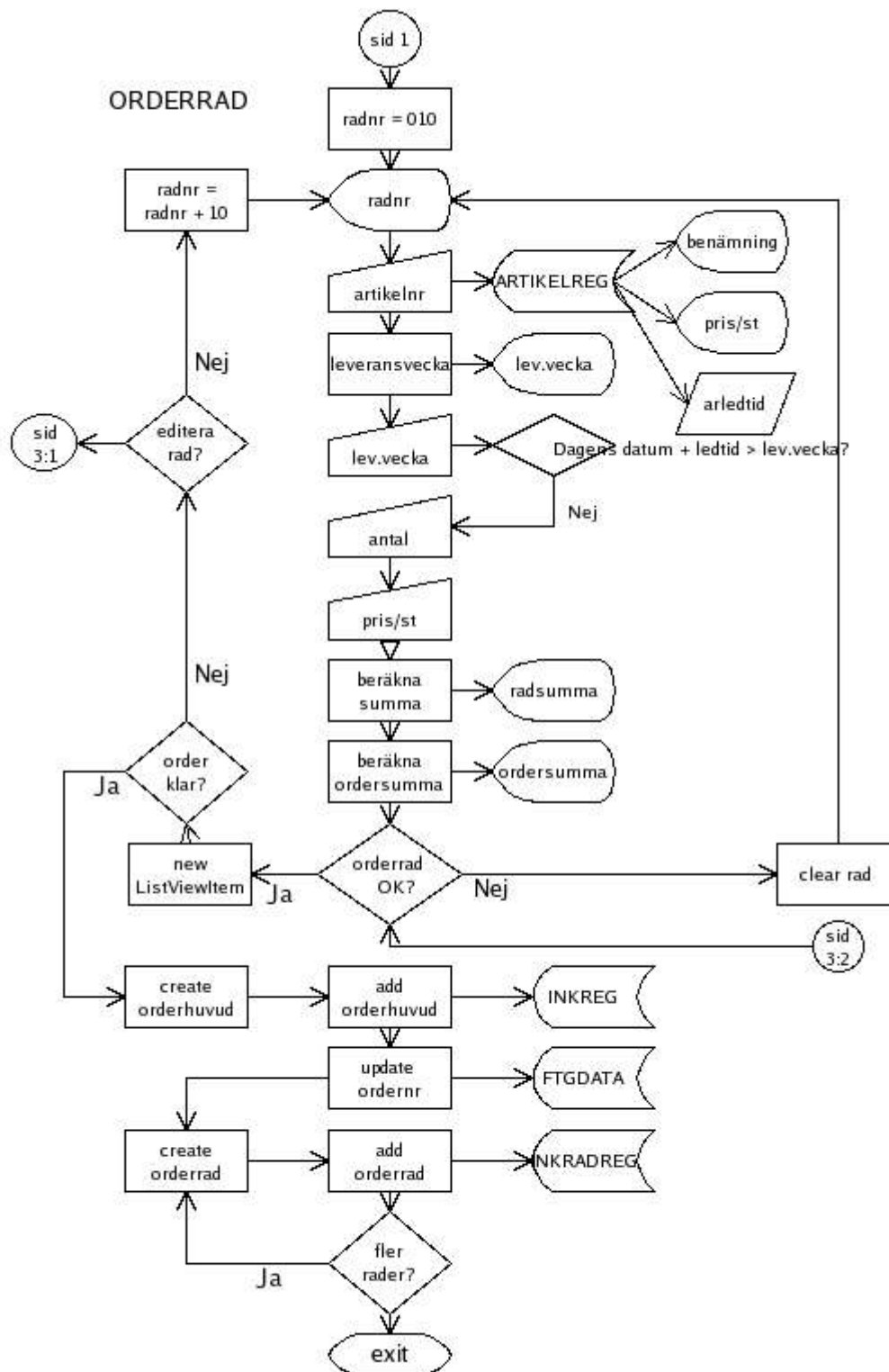
---

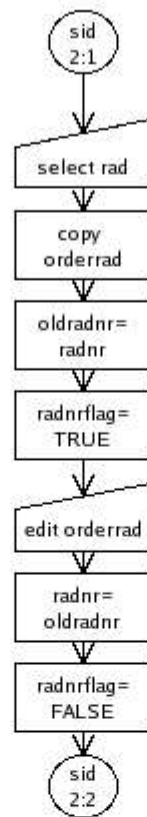
Godkänna beställning

Beställning Total: 12500.00









För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDINKW.

ADDKSTW anropar INKADD och INKRADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // userid
process->addArgument("INKADD");               // OLFIX funktion
process->addArgument(orderhuvud);
```

och

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // userid
process->addArgument("INKRADD");             // OLFIX funktion
process->addArgument(orderraddata);
```

Detta blir:

```
./STYRMAN userid INKADD orderhuvuddata
```

och

```
./STYRMAN userid INKRADD orderraddata
```

### Behörighetskrav:

För att kunna köra ADDINKW behövs behörighet till

ARDSP

ARDSPL

FTGDSP

FTGUPD

INKADD

INKRADD

LEVDSPL

LEVLST

LEVVDSP

LEVSDSP

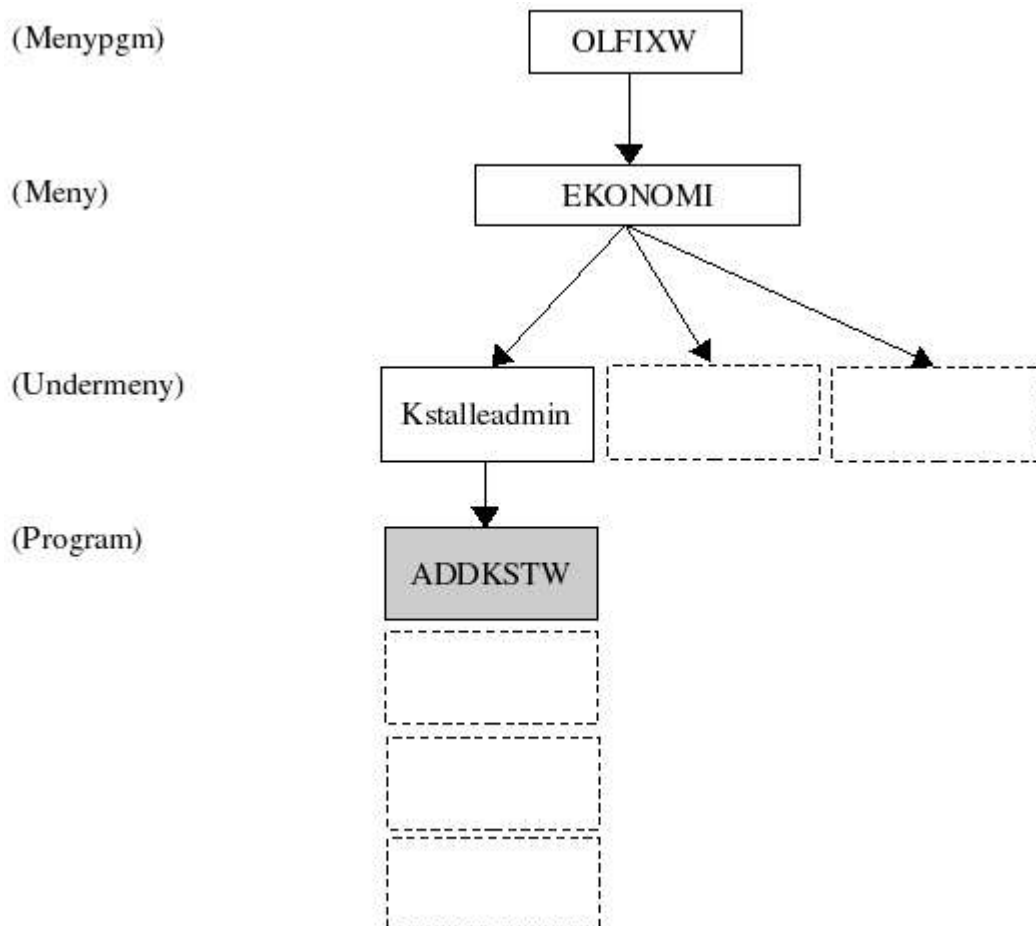
PRGLST

TXTDSP

USERDSP

## ADDKSTW.....Nytt kostnadsställe

ADDKSTW, ett grafiskt program för att registrera nya kostnadsställen. Man måste ange bokföringsår (arid). Programmet plockar upp userid från environment.



OLFIX - ADDKSTW Lägga upp nytt kostnadsställe

Bokföringsår: AD

Kostnadsställe: 9040

Benämning: Projekt OLFIX

OK Avsluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDKSTW.

ADDKSTW anropar KSTADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // userid
process->addArgument("KSTADD");               // OLFIX funktion
process->addArgument(arid);
process->addArgument(kstalle);
process->addArgument(benamn);
```

Detta blir:

```
./STYRMAN userid KSTADD arid kstalle benamn
```

### **Behörighetskrav:**

För att kunna köra ADDKSTW behövs behörighet till  
PRGLST  
KSTADD

## ADDKTOW.....Nytt Konto

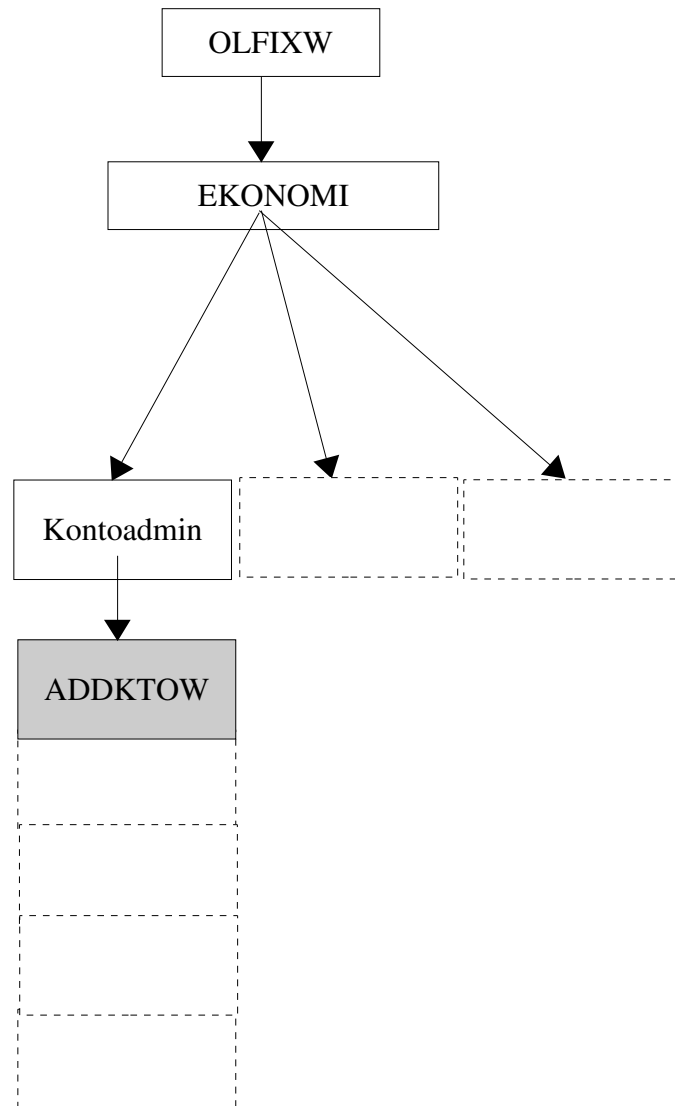
ADDKTOW, ett grafiskt program för att registrera nya konton. Man måste ange bokföringsår (arid). Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



ADDKTOW Lägga upp nytt konto

\* = Obligatorisk

Bokföringsår: \* ZZ  
(arid, 2 teck.)

Kontonummer \* 2440

Benämning: Leverantörsskulder

Manuell (J/N): \* J

Momskod: \* 1

SRUnr: \* 0

Kostnadsställe:

Projekt:

Subkonto:

Kontoplan: \* EUBAS97

IB:

UB:

OK Avsluta



För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDKTOW.

```
if (funk == "KTOADD")
    frmOLFIXW::runProgram("./ADDKTOW");
```

ADDKTOW anropar KTOADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

QString bibl;
bibl.append("./STYRMAN");                     // OLFIX huvudprogram

process = new QProcess();
process->addArgument("./STYRMAN");             // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("KTOADD");                // OLFIX funktion
process->addArgument(arid);
process->addArgument(ktonr);
process->addArgument(benamn);
process->addArgument(manuell);
process->addArgument(momskod);
process->addArgument(srunr);
process->addArgument(kst);
process->addArgument(projekt);
process->addArgument(subkonto);
process->addArgument(ktoplan);
```

Detta blir:

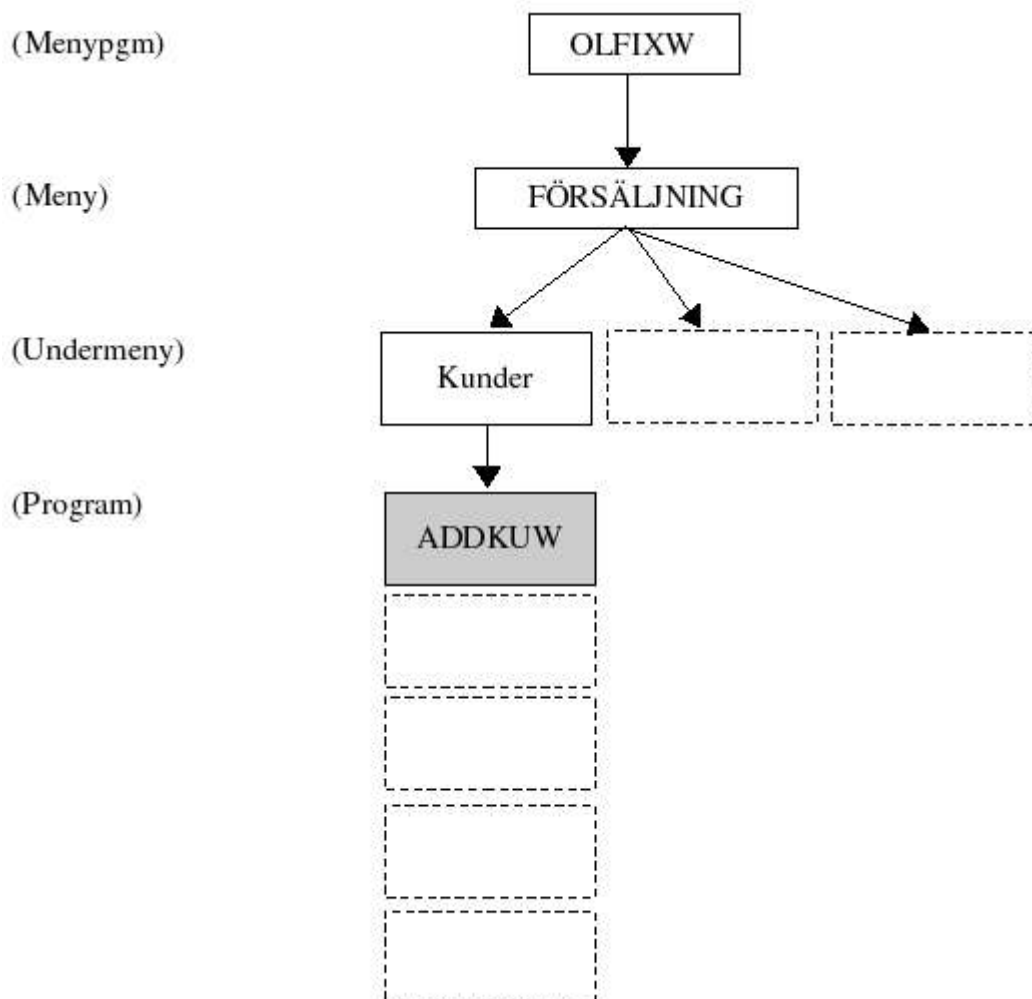
```
./STYRMAN usr KTOADD arid ktonr benamn manuell momskod srunr kst
           projekt subkonto ktoplan
```

### Behörighetskrav:

För att kunna köra ADDKTOW behövs behörighet till  
PRGLST  
KTOADD

## ADDKUW ..... Ny kund

ADDKUW är ett grafiskt program för att lägga upp nya kunder  
Programmet plockar upp userid från environment.



ADDKUW - Lägga upp en ny kund.

\* = Obligatorisk information.

KundID: ..... \* 125 Max 10 tecken.

Kundnamn: ..... \* Testbolaget AB

Kundadress: ..... Storgatan 33

Postnummer: ..... 199 11

Postadress: ..... LILLIBY

Land: ..... Sverige

Telefonnummer: ..... 09-999910

Faxnummer: ..... 09-999919

E-mail: ..... info@testbolaget.se

Er Referent: ..... Lars Andersson

Er Ref's telefonnr: .... 09-999911

Er Ref's e-mailadr: .. landersson@testbolaget.se

Vår säljare: ..... Caroline Seljare

Distrikt: ..... LI Kundkategori: ..... ST

Leveransplats: ..... 002 Leveransvillkor: ..... 001 Leveranssätt: ..... 001

Betalningsvillkor: 1

Valuta: ..... SEK Språkkod: ..... sv

Ordererkännande: .. J Plocklista : ..... J Följesedel: ..... J

Expeditionsavgift: ... J Fraktavgift: ..... J Kravbrev: ..... J

Kreditlimit: ..... 100000

Dröjsmålsränta: ..... J Dröjsmålsfaktura: .. J

Fri text (100 tkn): .... Bästa kunden

OK Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDKUW.

Först skapas *kunddata* enligt följande:

```
QString skilj;
    skilj="_:_";
    kunddata=skilj;
    kunddata.append(kundid);
    kunddata.append(skilj);
    kunddata.append(kundnamn);
    kunddata.append(skilj);
    kunddata.append(kundadress);
    kunddata.append(skilj);
    kunddata.append(postnr);
    kunddata.append(skilj);
    kunddata.append(postadr);
    kunddata.append(skilj);
    kunddata.append(land);
    kunddata.append(skilj);
    kunddata.append(tfnnr);
    kunddata.append(skilj);
    kunddata.append(faxnr);
    kunddata.append(skilj);
    kunddata.append(email);
    kunddata.append(skilj);
    kunddata.append(erref);
    kunddata.append(skilj);
    kunddata.append(erreftfnnr);
    kunddata.append(skilj);
    kunddata.append(errefemail);
    kunddata.append(skilj);
    kunddata.append(seljare);
    kunddata.append(skilj);
    kunddata.append(distrikt);
    kunddata.append(skilj);
    kunddata.append(kundkat);
    kunddata.append(skilj);
    kunddata.append(levplats);
    kunddata.append(skilj);
    kunddata.append(levvillkor);
    kunddata.append(skilj);
    kunddata.append(levsett);
    kunddata.append(skilj);
    kunddata.append(betvillkor);
    kunddata.append(skilj);
    kunddata.append(valuta);
    kunddata.append(skilj);
    kunddata.append(sprakkod);
    kunddata.append(skilj);
    kunddata.append(ordererk);
    kunddata.append(skilj);
    kunddata.append(plocklista);
    kunddata.append(skilj);
    kunddata.append(foljesedel);
    kunddata.append(skilj);
    kunddata.append(expavg);
    kunddata.append(skilj);
    kunddata.append(fraktavg);
    kunddata.append(skilj);
    kunddata.append(kravbrev);
    kunddata.append(skilj);
    kunddata.append(kreditlimit);
    kunddata.append(skilj);
```

```

kunddata.append(drojmalsrenta);
kunddata.append(skilj);
kunddata.append(drofimalsfakt);
kunddata.append(skilj);
kunddata.append(fritext);
kunddata.append(skilj);

```

sedan anropar ADDKUW KUADD via STYRMAN med parametern *kunddata*.

```

const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument( "STYRMAN");           // OLFIX funktion
process->addArgument(usr);
process->addArgument( "KUADD" );
process->addArgument(kunddata);             process->addArgument(momskod);

```

Detta blir:

```
./STYRMAN usr KUADD kunddata
```

### **Behörighetskrav:**

För att kunna köra ADDKUW behövs behörighet till

PRGLST

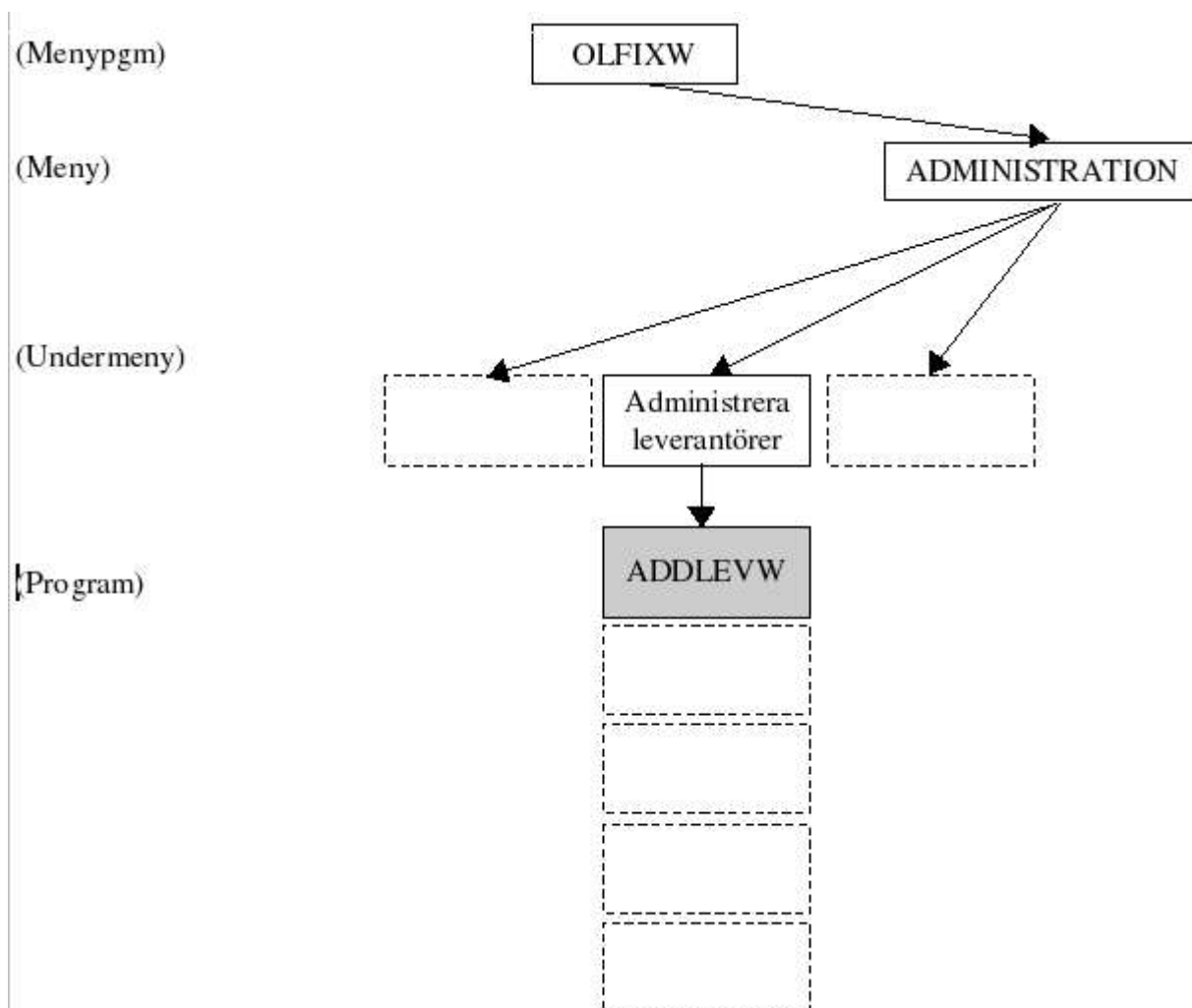
KUADD

KUCHK

SLPADD

## ADDLEVW ..... Ny leverantör

ADDLEVW är ett grafiskt program för att lägga upp nya leverantörer  
Programmet plockar upp userid från environment.



ADDLEW - Lägga upp en ny leverantör.

Leverantörsnummer:	<input type="text"/>	Max 10 tecken.	Obligatoriskt.
Organisationsnr:	<input type="text"/>		
Leverantörsnamn:	<input type="text"/>		Obligatoriskt.
Leverantörsadress:	<input type="text"/>		
Postnummer: .....	<input type="text"/>		
Postadress: .....	<input type="text"/>		
Land: .....	<input type="text"/>		
Telefonnummer: .....	<input type="text"/>		
Faxnummer: .....	<input type="text"/>		
Telex: .....	<input type="text"/>		
E-mail: .....	<input type="text"/>		
Referent: .....	<input type="text"/>		
Ref's telefonnr: .....	<input type="text"/>		
Momskod: .....	<input type="text" value="1"/>		Obligatoriskt.
Kontonummer: .....	<input type="text"/>		
Postgironummer: ....	<input type="text"/>		
Bankgironummer: ...	<input type="text"/>		
Kundnr: .....	<input type="text"/>		

OK Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDLEVW.

ADDLEVW anropar LEVADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER");          // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");             // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("LEVADD");                // OLFIX funktion
process->addArgument(levnr);
process->addArgument(levorgnr);
process->addArgument(levnamn);
process->addArgument(levadress);
process->addArgument(levpostnr);
process->addArgument(levpostadr);
process->addArgument(levland);
process->addArgument(levtfnnr);
process->addArgument(levfaxnr);
process->addArgument(levtelexnr);
process->addArgument(levemail);
process->addArgument(levpgnr);
process->addArgument(levbgnr);
process->addArgument(levref);
process->addArgument(levreftfnnr);
process->addArgument(levmomskod);
process->addArgument(levkontonr);
process->addArgument(levkundnr);
```

Detta blir:

```
./STYRMAN usr LEVADD levnr levorgnr levnamn levadress levpostnr levpostadr
levland levtfnnr levfaxnr levtelexnr levemail levpgnr levbgnr levref
levreftfnnr levmomskod levkontonr levkundnr
```

Turordningen på argumenten är viktig, LEVADD bearbetar dem i denna ordning.

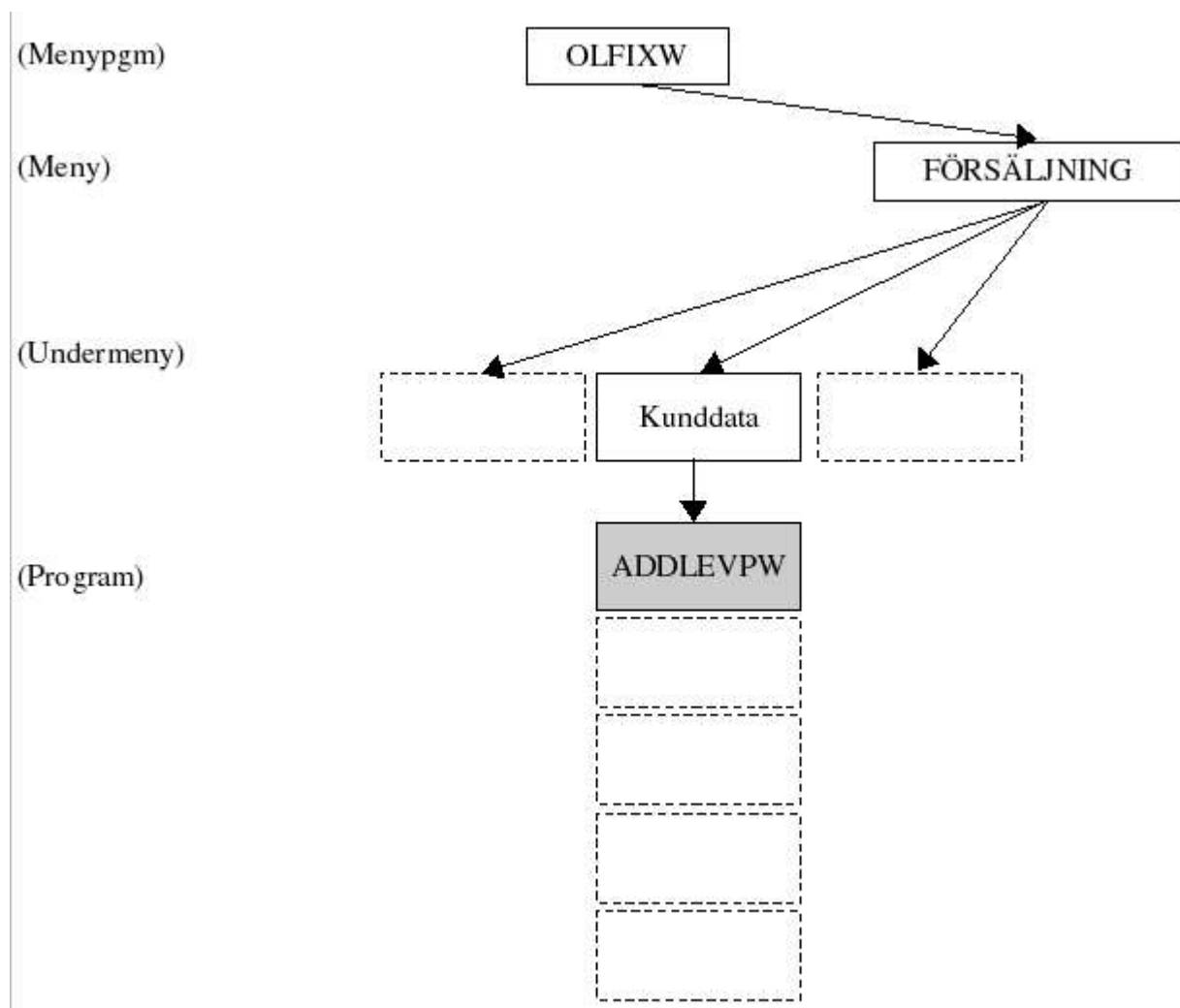
### Behörighetskrav:

För att kunna köra ADDLEVW behövs behörighet till  
PRGLST  
LEVADD



## ADDLEVPW ..... Ny standardleveransplats

ADDLEVPW är ett grafiskt program för att lägga upp nya leveransplatser för kunder. Programmet plockar upp userid från environment.



ADDLEVPW - Ny standardleveransplats

\* = Obligatoriskt.

KundID: \* 1234

Platsnummer: \* 002

Leveransadress: \* Storgatan 44

Postnummer: \* 19999

Postadress: \* STORSTAD

Land: Sverige

OK Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDLEVPW.

ADDLEVPW anropar SLPADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument( "STYRMAN" );             // OLFIX funktion
process->addArgument(usr);
process->addArgument( "SLPADD" );
process->addArgument(kundid);
process->addArgument(levplatsnr);
process->addArgument(levadress);
process->addArgument(postnr);
process->addArgument(postadr);
process->addArgument(land);
```

Detta blir:

```
./STYRMAN usr SLPADD kundid levplatsnr levadress postnr postadr land
```

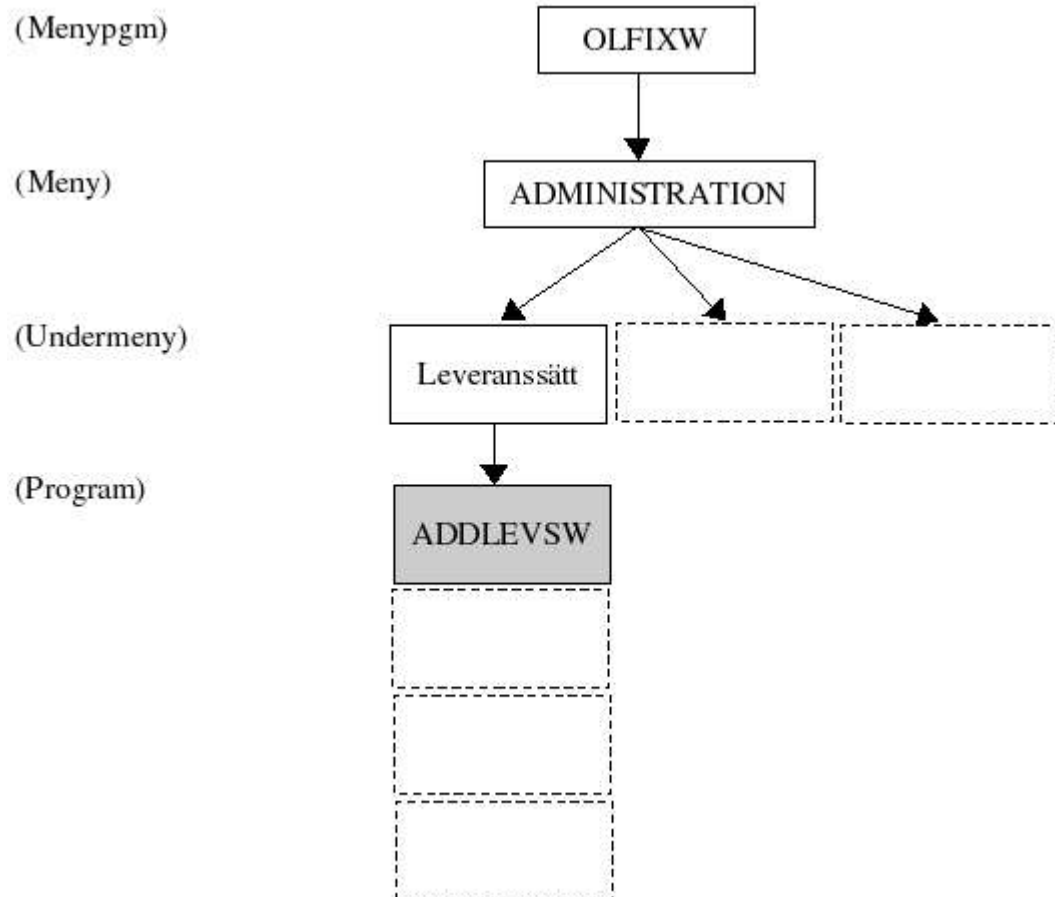
Turordningen på argumenten är viktig, SLPADD bearbetar dem i denna ordning.

### **Behörighetskrav:**

För att kunna köra ADDLEVPW behövs behörighet till  
PRGLST  
SLPADD

## ADDLEVSW ..... Nytt leveranssätt

ADDLEVSW är ett grafiskt program för att lägga upp nya **leveranssätt**.  
Programmet plockar upp userid från environment.





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDLEVSW.

ADDLEVSW anropar LEVSADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument( "STYRMAN");              // OLFIX funktion
process->addArgument(usr);
process->addArgument( "LEVSADD" );
process->addArgument(levsett);
process->addArgument(beskrivning);
```

Detta blir:

./STYRMAN usr LEVSADD levsett beskrivning

Turordningen på argumenten är viktig, LEVSADD bearbetar dem i denna ordning.

#### **Behörighetskrav:**

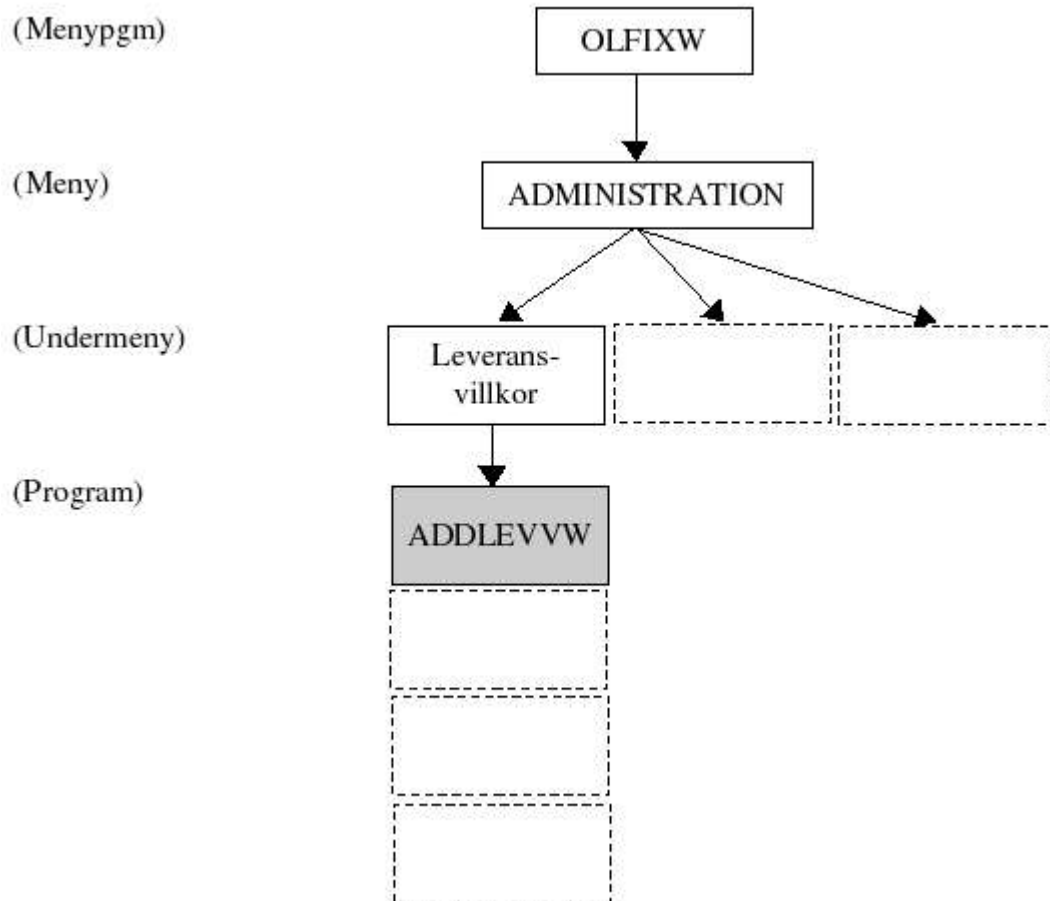
För att kunna köra ADDLEVSW behövs behörighet till

PRGLST

LEVSADD

## ADDLEVW ..... Nytt leveransvillkor

ADDLEVSW är ett grafiskt program för att lägga upp nya **leveransvillkor**.  
Programmet plockar upp userid från environment.







För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDLEVW.

ADDLEVW anropar LEVVADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument( "STYRMAN");              // OLFIX funktion
process->addArgument(usr);
process->addArgument( "LEVVADD" );
process->addArgument(levvillk);
process->addArgument(beskrivning);
```

Detta blir:

./STYRMAN usr LEVVADD levvillkor beskrivning

Turordningen på argumenten är viktig, LEVVADD bearbetar dem i denna ordning.

#### **Behörighetskrav:**

För att kunna köra ADDLEVW behövs behörighet till

PRGLST

LEVVADD

## ADDRGTW.....Ny behörighet

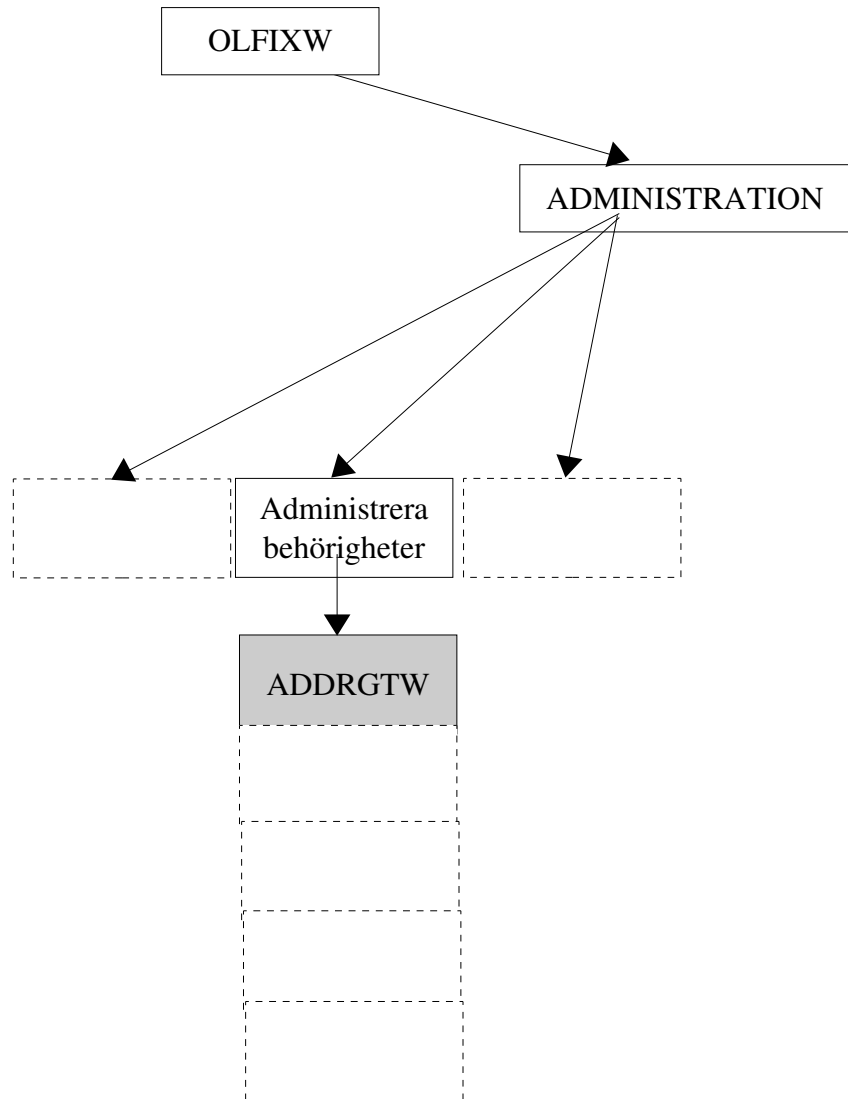
ADDRGTW är ett grafiskt program för att lägga upp nya behörigheter  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



ADDRGTW Lägga upp ny behörighet

Användar-ID:

Behörighet:

OK Avsluta

Funktion	Beskrivning
ARICLK	Kontrollera om visst bokföringsår finns
BARADD	Lägga upp nytt bokföringsår
BARCHG	Ändra data för angivet bokföringsår
BARCHK	Kontrollera om visst bokföringsår finns
BARDSP	Hämta data för angivet bokföringsår
BOKF	Bokföringsprogram

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

**Behörighetskrav:**

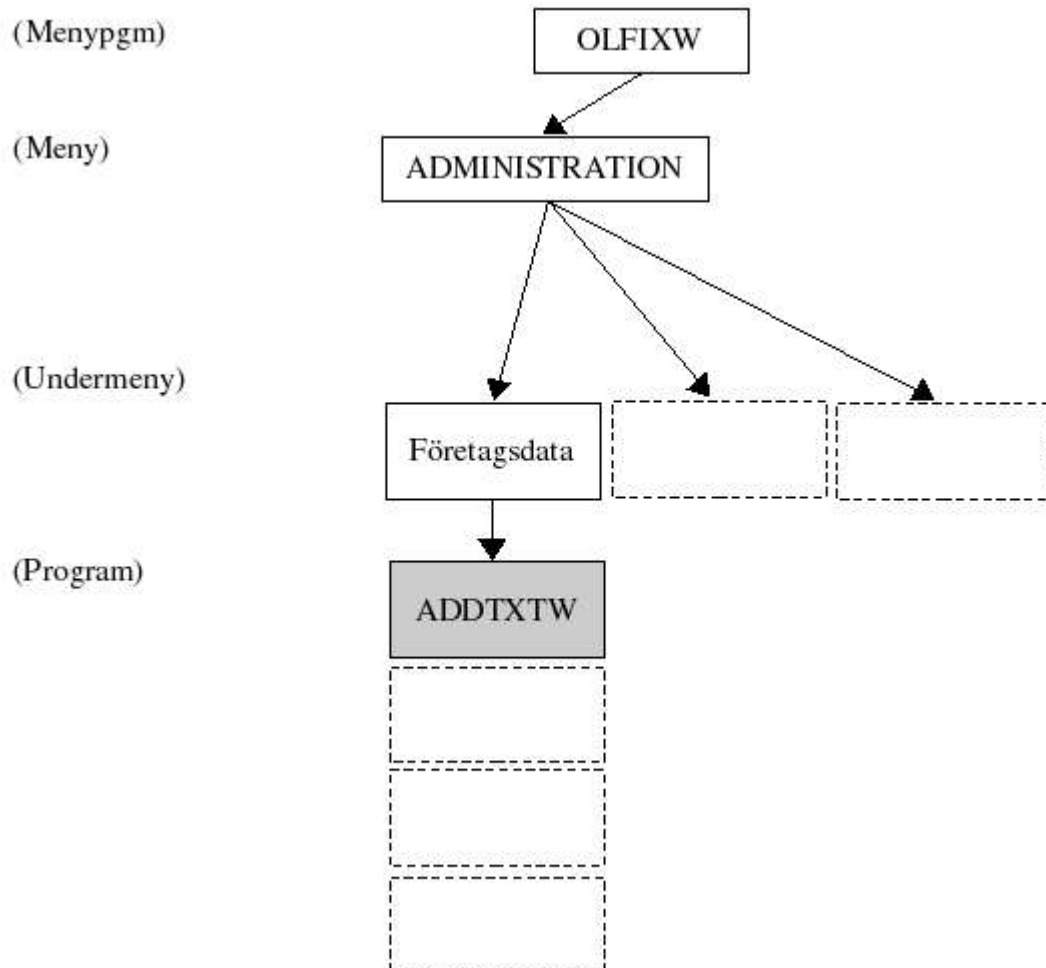
För att kunna köra ADDRGTW behövs behörighet till

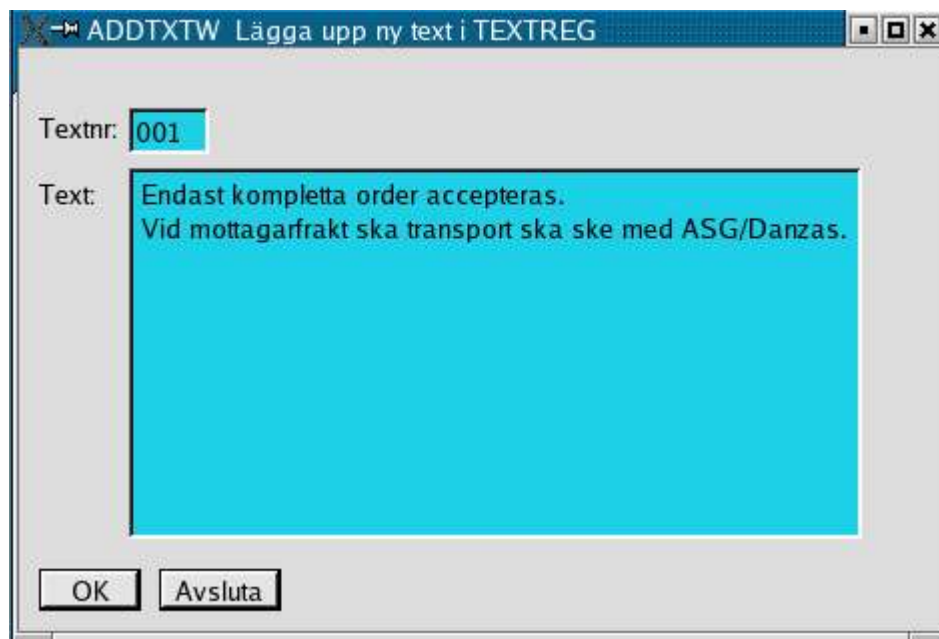
PRGLST

RGTTADD

## ADDTXTW.....Ny text till TEXTREG

ADDTXTW, ett grafiskt program för att lägga till nya texter i TEXTREG.  
Programmet plockar upp userid från environment.





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

**Behörighetskrav:**

För att kunna köra ADDTXTW behövs behörighet till

PRGLST

TXTADD

## ADDUSRW.....Ny användare

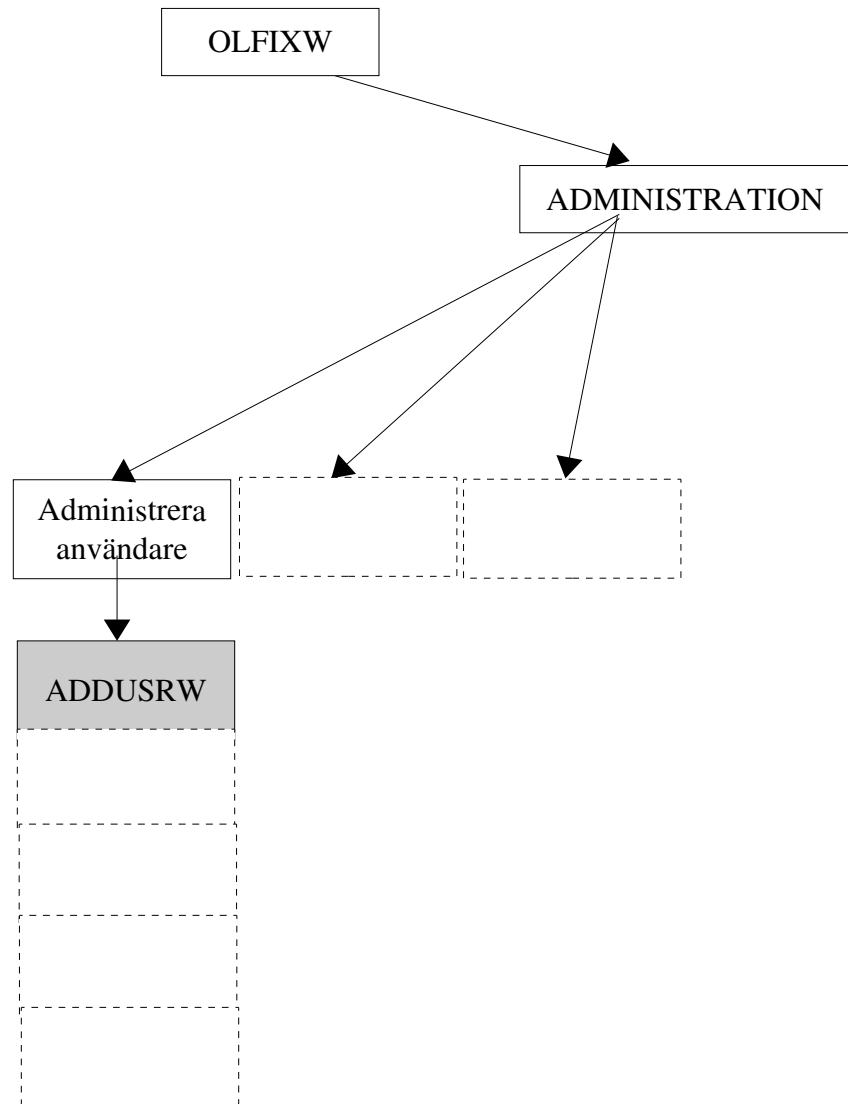
ADDUSRW, ett grafiskt program för att lägga till nya användare.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)





The image shows a Windows-style dialog box titled "ADDUSRW Lägga upp ny användare". It contains four text input fields labeled "Användar-ID:", "Namn:", "Avd:", and "Grupp:". Each field is currently filled with a solid blue color. At the bottom of the dialog, there are two buttons: "OK" and "Avsluta".

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.  
Detta görs i main.cpp.

**Behörighetskrav:**

För att kunna köra ADDUSRW behövs behörighet till  
PRGLST  
USERADD

## ADDVALW.....Ny valuta

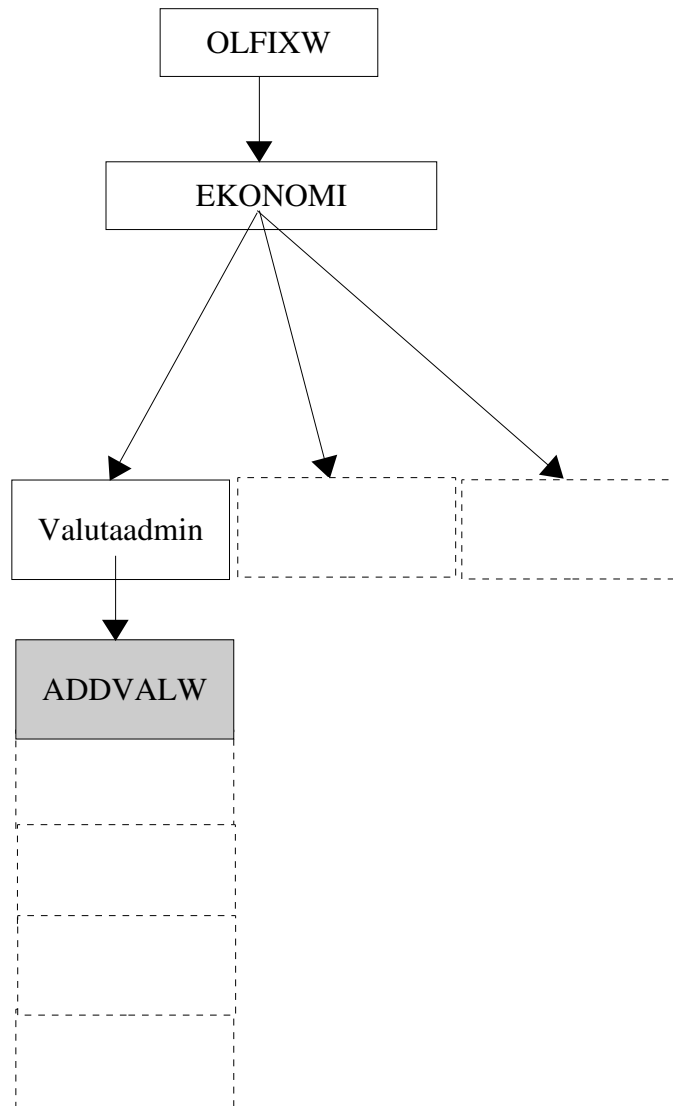
ADDVALW är ett grafiskt program för att lägga upp nya behörigheter  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



OLFIX - ADDVALW Lägga upp ny valuta

Valuta:

Beteckning:

Land:

Köp:

Sälj:

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda ADDVALW.

ADDVALW anropar VALADD via STYRMAN med parametrar.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);

QString bibl;
bibl.append("./STYRMAN"); // OLFIX huvudprogram

process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr); // userid
process->addArgument("VALADD"); // OLFIX funktion
process->addArgument(valuta);
process->addArgument(land);
process->addArgument(salj);
process->addArgument(kop);
process->addArgument(beteckning);
```

Detta blir:

```
./STYRMAN usr VALADD valuta land salj kop beteckning
```

### **Behörighetskrav:**

För att kunna köra ADDVALW behövs behörighet till  
PRGLST  
VALADD

## ATTBETW.....Lista obetalda leverantörsfakturer

ATTBETW är ett grafiskt program för att lista leverantörsfakturer som förfaller till betalning senast ÅÅÅÅ-MM-DD.

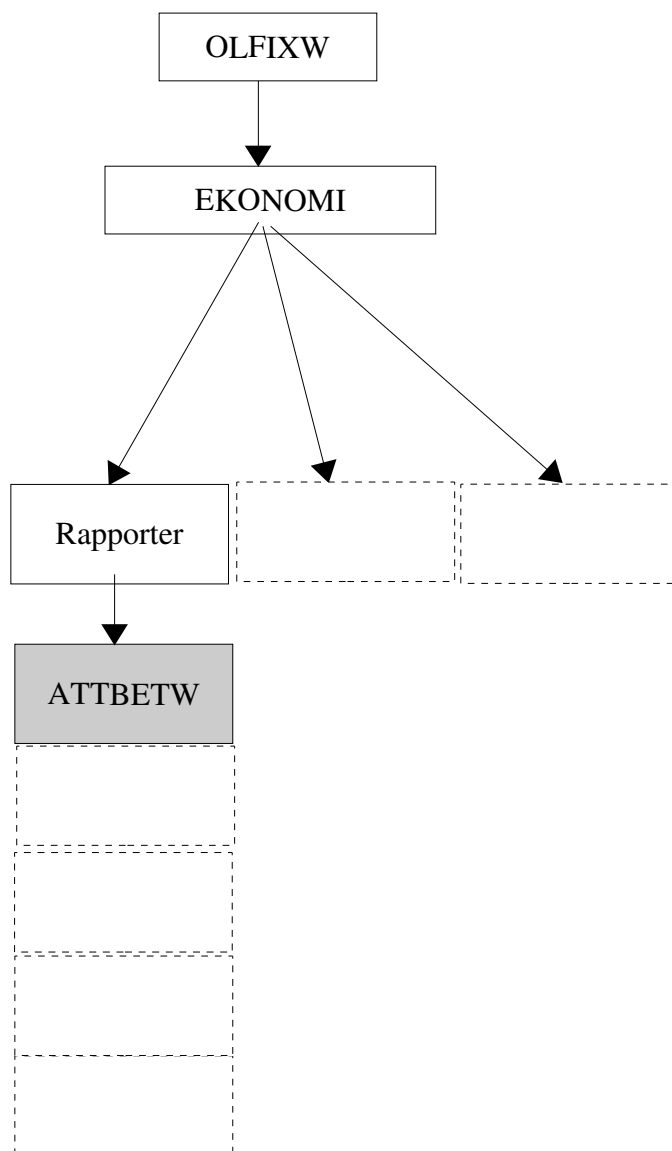
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



## Funktionsbeskrivning.

Programmet ATTBETW skapar en datafil för rapportgeneratoren Kugar, /tmp/AttBetala.kud, i XMLformat.

```
rad1="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
rad2="<!DOCTYPE KugarData [\n";
rad3="    <!ELEMENT KugarData (Row* )>\n";
rad4="    <!ATTLIST KugarData\n";
rad5="        Template CDATA #REQUIRED>\n\n";
rad6="    <!ELEMENT Row EMPTY>\n";
rad7="    <!ATTLIST Row\n";
rad8="        level CDATA #REQUIRED\n";
rad9="        betaldatum CDATA #REQUIRED\n";
rad10="        levnr CDATA #REQUIRED\n";
rad11="        fakturanr CDATA #REQUIRED\n";
rad12="        belopp CDATA #REQUIRED\n";
rad12b="        valuta CDATA #REQUIRED>\n";
rad13="]>\n\n";
rad14="<KugarData Template=\"AttBet.kut\">\n";
```

Eventuell gammal fil raderas innan den nya filen skapas.

När datafilen skapats anropas Kugar med kommando

```
system("kugar -d /tmp/attBetala.kud -r /usr/local/olfix/data/AttBet.kut");
```

AttBet.kut är ett “template” (layout) av rapporten. AttBet.kut kan behöva editeras med rätt företagsnamn vilket kan göras med kommando

```
sed -e 's/PROGRAM AB/KALLES AB/' AttBet.kut > test.kut
```

där “KALLES AB” ersätts med företagets eget namn.

Sedan ändras test.kut till AttBet.kut.

ATTBETW - Rapport över leverantörsfakturer att betala

Förfalldatum: 2003-08-22

Skapa rapport Avbryt

Kugar

Arkiv Kör Inställningar Hjälp

PROGRAM AB

Leverantörsfakturer förfallna till betalning

Förfalldatum	Leverantörsnummer	Fakturanummer	Fakturabelopp	Valuta
2003-08-22	123	1	2000.00	SEK
2003-08-30	123	3321	2000.00	SEK
2003-08-30	123	115599	3000.00	SEK
2003-08-30	123	5522881	3000.00	SEK
2003-08-30	123	665544	3000.00	SEK
2003-08-30	123	556699	3000.00	SEK
2003-08-30	123	55	1000.00	SEK
2003-08-30	123	25	1000.00	SEK
2003-08-30	123	669977	1000.00	SEK
2003-08-30	123	56128745	5000.00	SEK
2003-09-02	123	2	1000.00	SEK



För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda programmet ATTBETW.

ATTBETW anropar ATTBET via STYRMAN. Se ATTBET.

```
const char *userp = getenv("USER");
QString usr(userp);
QString filnamn;

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("ATTBET");              // OLFIX funktion
process->addArgument(datum);
```

### **Behörighetskrav:**

För att kunna köra ATTBETW behövs behörighet till

PRGLST

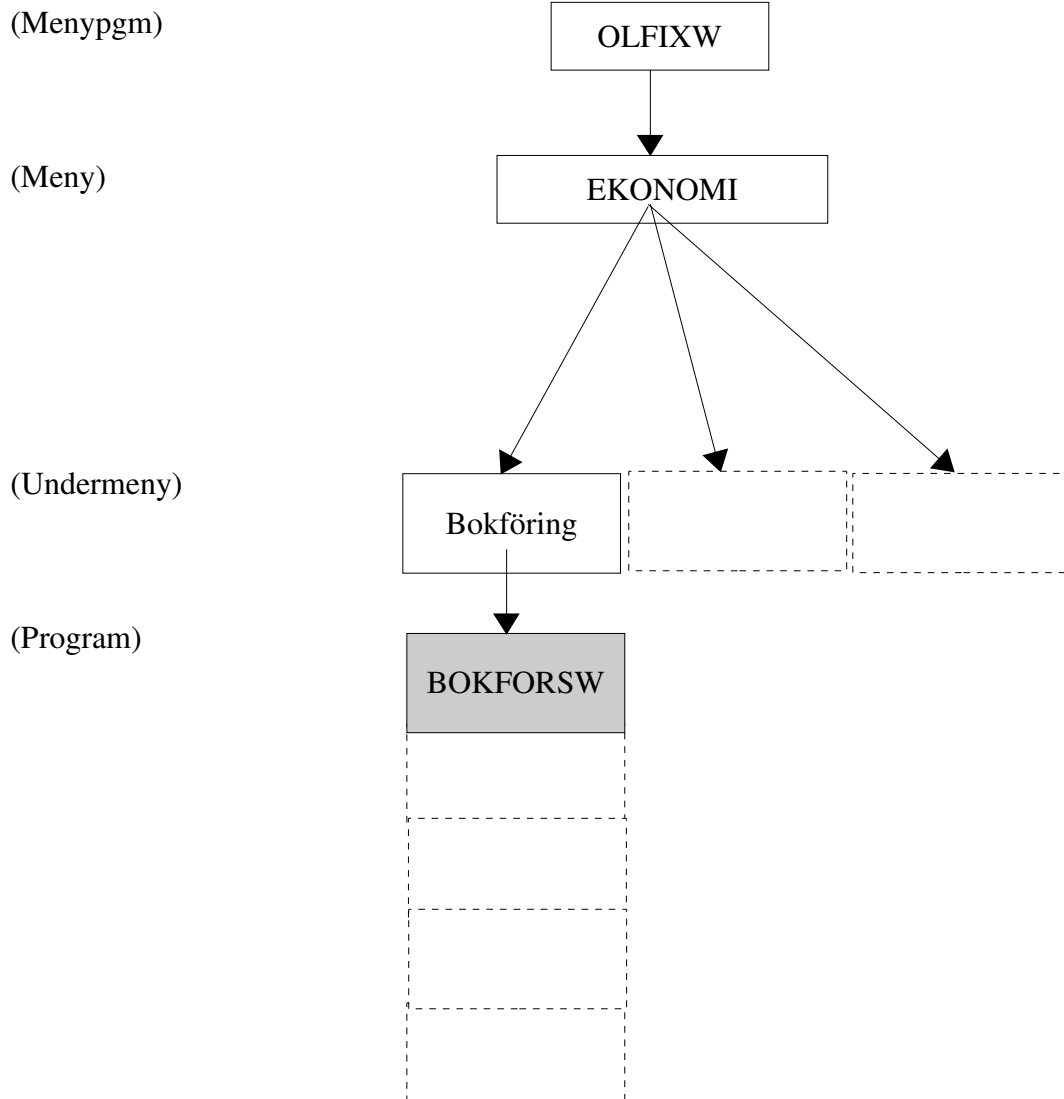
ATTBET

Programmet Kugar

Läs rättigheter i biblioteket /usr/local/olfix/data/

## BOKFORSW....Bokföring standard

BOFARSW, ett grafiskt program för att registrera verifikat (bokföring).  
Programmet plockar upp userid från environment.



**BOKFORSW Registrering av verifikationer. Standardversion. (0.41)**

## OLFIX Bokföring

Datum: 2003-08-10

Bokföringsår:

Vernummer:

Verifikationstext:

Radnr Kontonr D/K Belopp KSTÄLLE SUBKTO Godkänn rad

Diff:

Registrerade verifikationsrader

Radnr	Kontonr	D/K	Belopp	KSTÄLLE	SUBKTO
001	2440	K	50000.00		
002	2641	D	12500.00		
003	4010	D	37500.00		

Godkänn verifikation

Kontoförteckning

Kontonr	Kontotext
2641	Ingående moms
2645	Ingående moms utland
2710	Källskatt (A-skatt)
2920	Upplupna semesterlöner
2941	Upplupna arbetsgivaravgifter
3041	Försäljning Jonaid
3051	Försäljning Zuhaib
3960	Kursvinst rörelsen
3990	Övriga intäkter
4010	Materialkostnad
4056	Varuinköp EU
5010	Lokalhyra
5090	Övriga lokalkostnader
5830	Kost och logi
5900	Reklam och PR
6071	Representation, avdragsgill
6110	Kontorsmateriel
6200	Telefon och Post
6310	Företagsförsäkringar
6970	Tidningar, facklitteratur
6992	Övriga kostnader, ej avdragsgilla
7010	Löner anställda

Efter inmatning av **bokföringsår** görs kontroll att bokföringsåret är upplagt.

**Vernummer** sätts automatiskt.

**Verifikationstext** skrivs in av användaren.

**Radnr** sätts automatiskt.

Inmatning av **Kontonr** kan göras på två sätt;

klicka på önskat kontonr i **Kontoförteckningen** eller  
skriva in kontonr manuellt.

Efter inmatning av **Kontonr** görs kontroll att kontonr finns för aktuellt bokföringsår.

**D/K** skrivs av användaren. D för en debetkontering och K för en kreditkontering.

**Belopp** registreras av användaren. På rad 001 ska verifikatets totalbelopp skrivas.

Om **KSTÄLLE** fylls i görs kontroll att kostnadstället finns registrerat på aktuellt kontonr och aktuellt bokföringsår. Fältet får lämnas tomt.

Om **SUBKONTO** fylls i ska kontroll göras att subkonto finns registrerat på aktuellt kontonr. Fältet får lämnas tomt. (Ej implementerat)

När man godkänt verifikationsrad så uppdateras **Diff**. När rad 001 registreras så förs samma belopp in i Diff. För varje påföljande rad som registreras så minskas värdet i Diff med aktuell rads belopp. Först när Diff är 0:- kan verifikationen godkännas.

I samband med uppdatering av databasen tas **datum**, **userid** med till VERHUVUD och TRHD.

Följande tabeller är involverade:

BOKFAR	läses/uppdateras
KTOPLAN	läses
VERHUVUD	uppdateras
VERRAD	uppdateras
KSTALLE	läses

TRHD

uppdateras

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda programmet BOKFORSW.

BOKFORSW anropar KTOVIEW, BARCHK, BARDSP, KSTCHK, KTOCHK och VERUPD via STYRMAN. Dessutom anropas WRREC 2 gånger.

KTOVIEW

```
const char *userp = getenv("USER");
QString usr(userp);
inradktolist="";
errorrad="";

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KTOVIEW");             // OLFIX funktion
process->addArgument(arid);
```

BARCHK

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("BARCHK");              // OLFIX funktion
process->addArgument(arid);
```

BARDSP

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("BARDSP");              // OLFIX funktion
process->addArgument(arid);
```

KSTCHK

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KSTCHK");              // OLFIX funktion
process->addArgument(arid);
process->addArgument(kstalle);
```

KTOCHK

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KTOCHK");              // OLFIX funktion
```

```

process->addArgument(arid);
process->addArgument(kontonr);

WRREC (1)
const char *userp = getenv("USER");
QString usr(userp);
QString posttyp="H";
if (kstalle == "")
    kstalle=" ";
if(subkto == "")
    subkto=" ";
while (usr.length()<8){
    usr.append(" ");
}
process = new QProcess();
process->addArgument( "./WRREC");           // OLFIX funktion
process->addArgument(posttyp);
process->addArgument(arid);
process->addArgument(vernr);
process->addArgument(radnr);
process->addArgument(kontonr);
process->addArgument(dk);
process->addArgument(belopp);
process->addArgument(kstalle);
process->addArgument(subkto);
process->addArgument(datum);
process->addArgument(usr);
process->addArgument(vertext);

WRREC (2)
QString posttyp="D";

process = new QProcess();
process->addArgument( "./WRREC");           // OLFIX funktion
process->addArgument(posttyp);
process->addArgument(arid);
process->addArgument(vernr);
process->addArgument(radnr);
process->addArgument(kontonr);
process->addArgument(dk);
process->addArgument(belopp);
process->addArgument(kstalle);
process->addArgument(subkto);

VERUPD
const char *userp = getenv("USER");
QString usr(userp);
QString filnamn;
filnamn.append("/tmp/");
filnamn.append(vernr);
filnamn.append(".txt");

process = new QProcess();
process->addArgument( "./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument( "VERUPD");             // OLFIX funktion
process->addArgument(filnamn);

```

### Behörighetskrav:

För att kunna köra BOFORSW behövs behörighet till  
 PRGLST  
 BARCHK

BARDSP  
KTOVIEW  
KTOCHK  
KSTCHK  
VERUPD

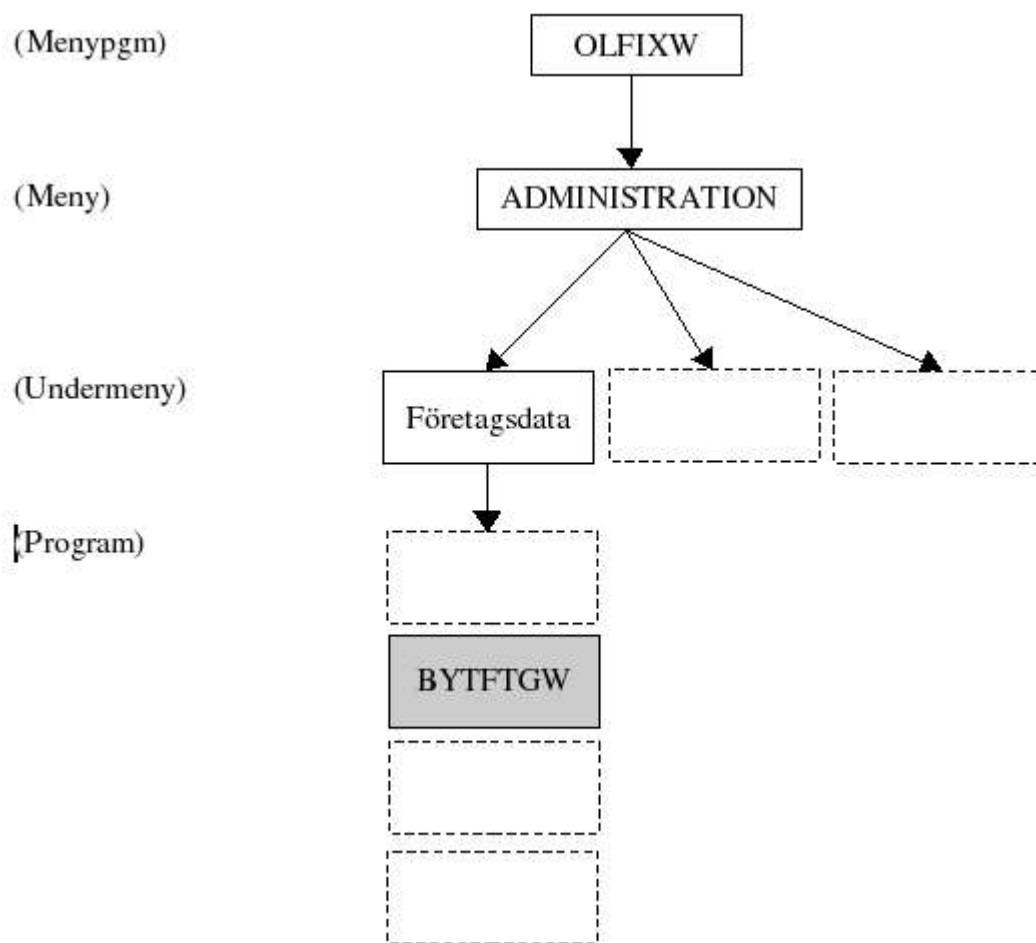
Funktionen WRREC används också.

## BYTFTGW ..... Byta företag.

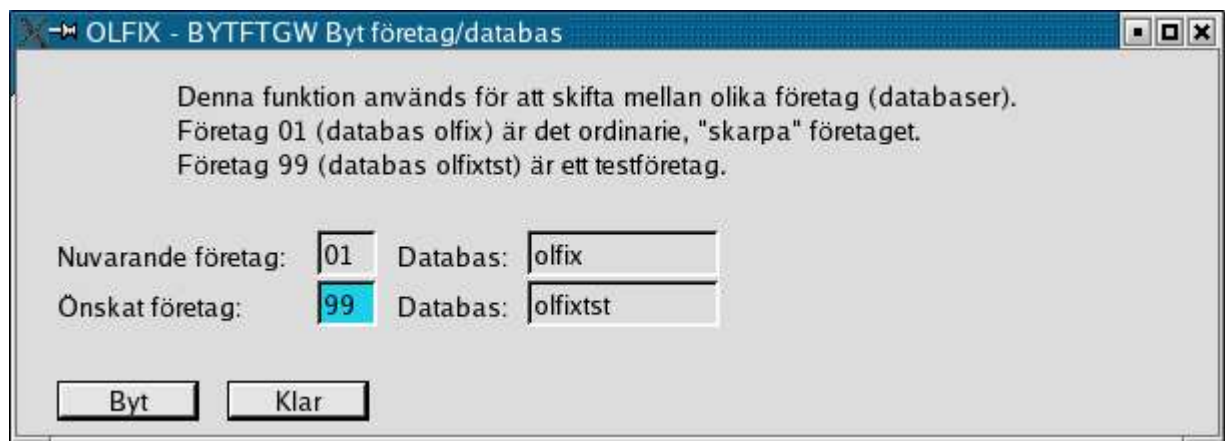
BYTFTGW, ett grafiskt program för att byta mellan olika företag (databaser).

### Flera företag.

OLFIX medger hantering av upp till 99 olika företag. Företag nr 99 är reserverat för testföretaget. OLFIX levereras med ett "skarpt" (ordinarie) företag och ett testföretag.







Programmet ändrar i filen \$HOME/.olfixrc, nämligen värdet på DATABASE.

```
PATH=/home/jan/Utveckling/OLFIX/bin/  
HOST=localhost  
DATABASE=olfixtst  
VTMP=/tmp/
```

Det går för närvarande endast att välja mellan företag 01 och 99.

## CHGBARW....Ändra bokföringsårsdata

CHGBARW, ett grafiskt program för att ändra info för ett bokföringsår. Man måste ange vilket bokföringsår (år) man vill ändra.

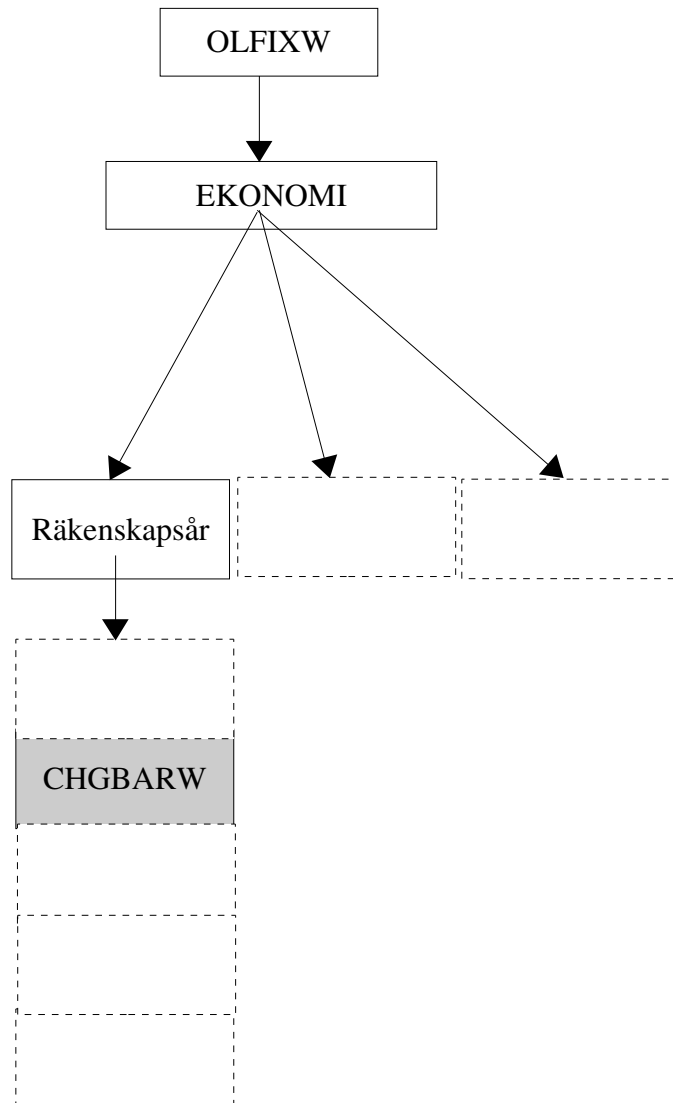
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



CHGBARW Ändra bokföringsår

Bokföringsår: AD Obligatoriskt.  
(arid, 2 teck.)

Benämning: 2003-01-01--2003-12-31

Startdatum: 2003-01-01 Obligatoriskt.

Slutdatum: 2003-12-31 Obligatoriskt.

Beskattningsår: 2003 Obligatoriskt.

Senaste ver.datum: 0000-00-00

Nästa ver.nummer: 1

Kontoplan: AD Obligatoriskt.

OK Avsluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen BARCHG.

CHGBARW anropar BARDSP och BARCHG via STYRMAN.

BARDSP

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // userid
process->addArgument("BARDSP");               // OLFIX funktion
process->addArgument(arid);
```

BARCHG

```
process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr);          // userid
process->addArgument("BARCHG");     // OLFIX funktion
process->addArgument(arid);
process->addArgument(benamn);
process->addArgument(arstart);
process->addArgument(ar slut);
process->addArgument(arlast);
process->addArgument(beskattar);
process->addArgument(senverdat);
process->addArgument(vernr);
process->addArgument(ktoplan);
```

Detta blir:

```
./STYRMAN usr BARDSP arid
```

och

```
./STYRMAN userid BARCHG arid benamning arstart ar slut arlast beskattningsar
senverdat vernr ktoplan.
```

usr är den inloggades userid.

### **Behörighetskrav:**

För att kunna köra CHGBARW behövs behörighet till  
PRGLST  
BARCHG  
BARDSP



## CHGBETVW....Ändra betalningsvillkor

CHGBETVW, ett grafiskt program för att ändra data för ett betalningsvillkor. Man måste ange vilket betalningsvillkor man vill ändra.

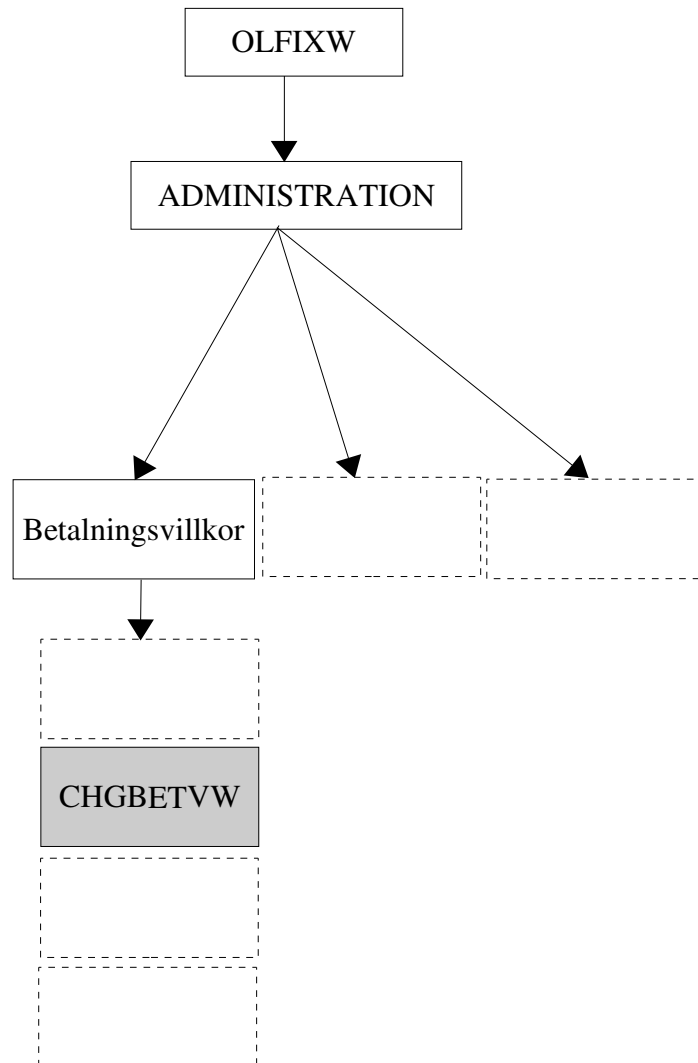
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



CHGBETVW Ändradata för betalningsvillkor

Betalningsvillkor: 001

Antal dagar 100

Beskrivning 10 daga netto

OK Avbryt





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen BETCHG.

CHGBETVW anropar BETDSP och BETCHG via STYRMAN.

BETDSP

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("BETDSP");              // OLFIX funktion
process->addArgument(betvillk);
```

BETCHG

```
process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr);         // userid
process->addArgument("BETCHG");    // OLFIX funktion
process->addArgument(betvillk);
process->addArgument(dagar);
process->addArgument(beskrivning);
```

Detta blir:

```
./STYRMAN usr BETDSP betvillk
```

och

```
./STYRMAN usr BETCHG betvillk dagar beskrivning
```

usr är den inloggades userid.

### **Behörighetskrav:**

För att kunna köra CHGBETVW behövs behörighet till

PRGLST

BETCHG

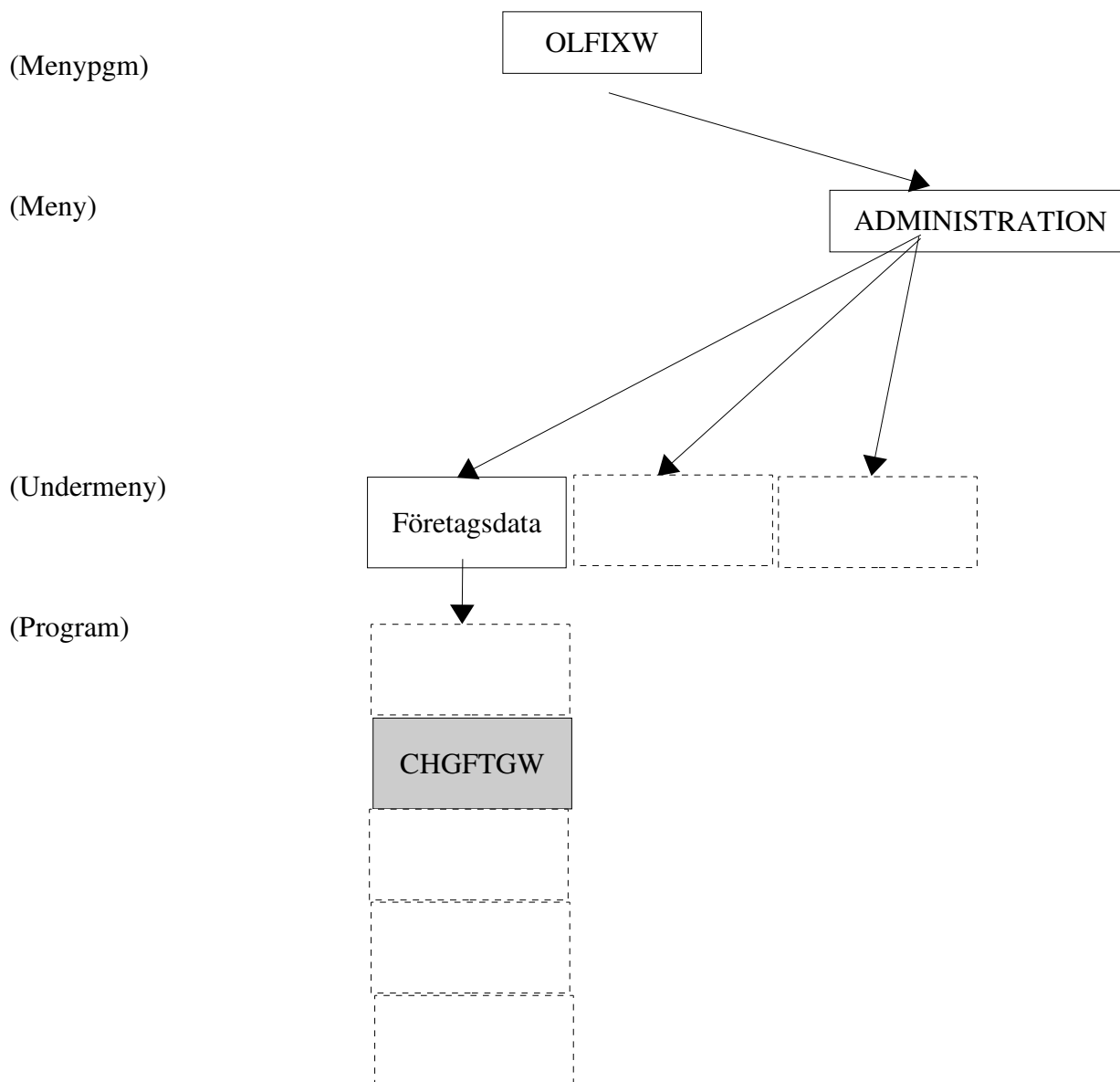
BETDSP

## CHGFTGW.....Ändra företagsdata

CHGFTGW, ett grafiskt program för att ändra information om företaget.

För att använda programmet krävs behörighet till funktionerna FTGLST och FTGUPD.

Programmet plockar upp userid från environment.



CHGFTGW - Ändra företagsdata

Posttyp: **ADR1** **Hämta**

Postadress: **Box 70**

Postnummer: **199 98**

Ort: **PROGSTAD**

**OK** **Uppdatera**

Posttyp	Benämning
ADR1	Postadress
ADR2	Postnummer till Postadress
ADR3	Ort till Postadress
ADR4	Besöksadress
ADR5	Postnr till Besöksadress
ADR6	Ort till Besöksadress
ADR7	Godsadress
ADR8	Postnr till Godsadress
ADR9	Ort till Godsadress
BF1	Bokföringsperiod 1
BF10	Bokföringsperiod 10
BF11	Bokföringsperiod 11
BF12	Bokföringsperiod 12
BF13	Bokföringsperiod 13
BF2	Bokföringsperiod 2

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda programmet CHGFTGW.

CHGFTGW anropar FTGLST och FTGUPD via STYRMAN.

```
const char *userp = getenv("USER");// Hämtar den inloggades userid
QString usr(userp);
QString bibl;

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                // userid
process->addArgument("FTGLST");           // OLFIX funktion
```

och

```
const char *userp = getenv("USER");// Hämtar den inloggades userid
QString usr(userp);

inrad="";
errorrad="";

process = new QProcess();
process->addArgument("./STYRMAN");        // OLFIX styrprogram
process->addArgument(usr);                // userid
process->addArgument("FTGUPD");           // OLFIX funktion
process->addArgument(posttyp);
process->addArgument(ftgdata);
```

### Behörighetskrav:

För att kunna köra CHGFTGW behövs behörighet till  
FTGLST  
FTGUPD

## CHGLEVW.....Ändra leverantörsdata

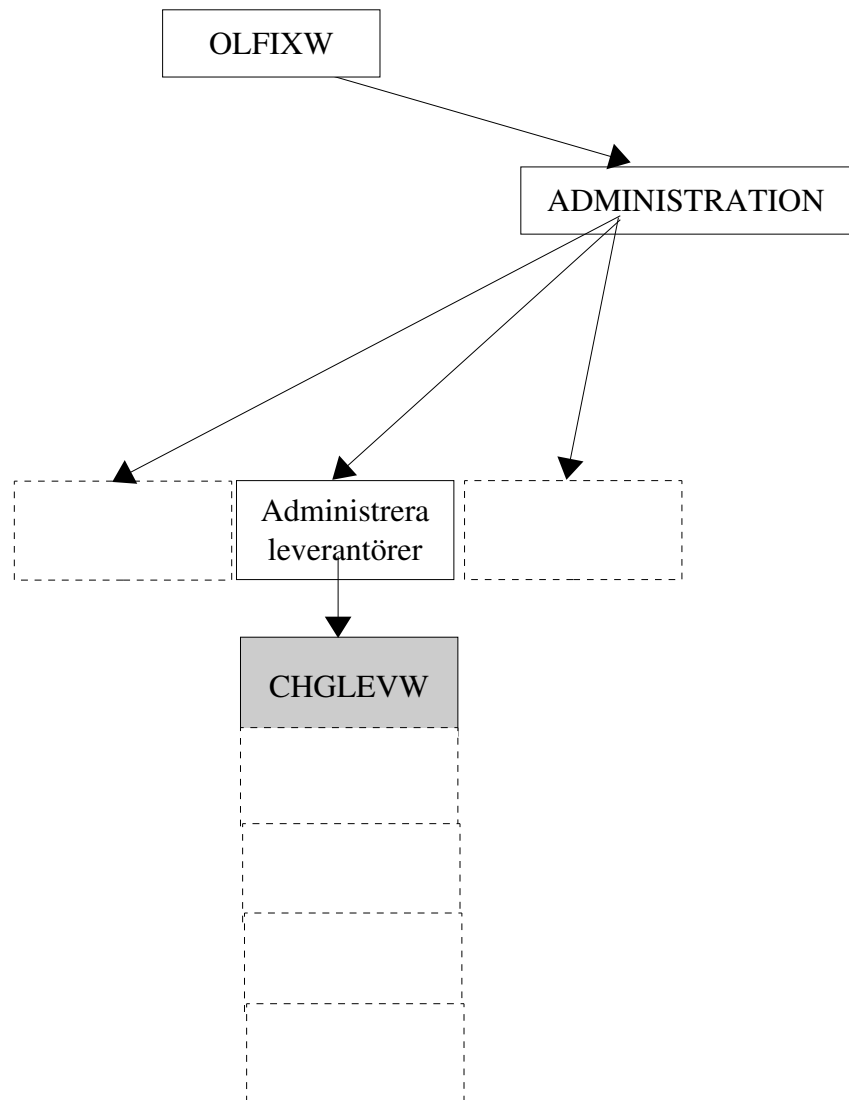
CHGLEVW är ett grafiskt program för att ändra information för en leverantör.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



CHGLEVW - Ändra leverantörsdata.

Leverantörsnummer:	124	Hämta
Organisationsnr:	559999-9999	
Leverantörsnamn:	Distributör AB	
Leverantörsadress:	Försäljningsvägen 27	
Postnummer: .....	199 99	
Postadress: .....	STORSTAD	
Land: .....	Sverige	
Telefonnummer: .....	09-199999	
Faxnummer: .....	09-199998	
Telex: .....	19999	
E-mail: .....	info@distributor.se	
Referent: .....	Per Karlsson	
Ref's telefonnr: .....	09-199996	
Momskod: .....	1	
Kontonummer: .....	2110	
Postgironummer: ....	4559998-9	
Bankgironummer: ...	5999-9998	
Kundnr: .....	991165	

OK Sluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda programmet CHGLEVW.

CHGLEVW anropar LEVDSP och LEVCHG via STYRMAN.

```
const char *userp = getenv("USER");// Hämtar den inloggades userid
QString usr(userp);
QString bibl;

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr); // userid
process->addArgument("LEVDSP"); // OLFIX funktion
process->addArgument(levnr);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

if (levmomskod == ""){
    levmomskod = "1";
}

process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr); // userid
process->addArgument("LEVCHG"); // OLFIX funktion
process->addArgument(levnr);
process->addArgument(levorgnr);
process->addArgument(levnamn);
process->addArgument(levadress);
process->addArgument(levpostnr);
process->addArgument(levpostadr);
process->addArgument(levland);
process->addArgument(levtfnnr);
process->addArgument(levfaxnr);
process->addArgument(levtelexn timer);
process->addArgument(levemail);
process->addArgument(levpgnr);
process->addArgument(levbgnr);
process->addArgument(levref);
process->addArgument(levreftfnnr);
process->addArgument(levmomskod);
process->addArgument(levkontonr);
process->addArgument(levkundnr);
```

Detta blir:

```
./STYRMAN usr LEVCHG levn timer levorgnr levnamn levadress levpostnr levpostadr
levland levtfnnr levfaxnr levtelexn timer levemail levpgnr levbgnr levref
levreftfnnr levmomskod levkontonr levkundnr
```

Turordningen på argumenten är viktig, LEVCHG bearbetar dem i denna ordning.

### Behörighetskrav:

För att kunna köra CHGLEVW behövs behörighet till  
PRGLST

LEV DSP  
LEV CHG



## CHGUSRW.....Ändra användardata

CHGUSRW, ett grafiskt program för att ändra information för en användare.

För att använda programmet krävs behörighet till funktionerna USERDSP och USERCHG.

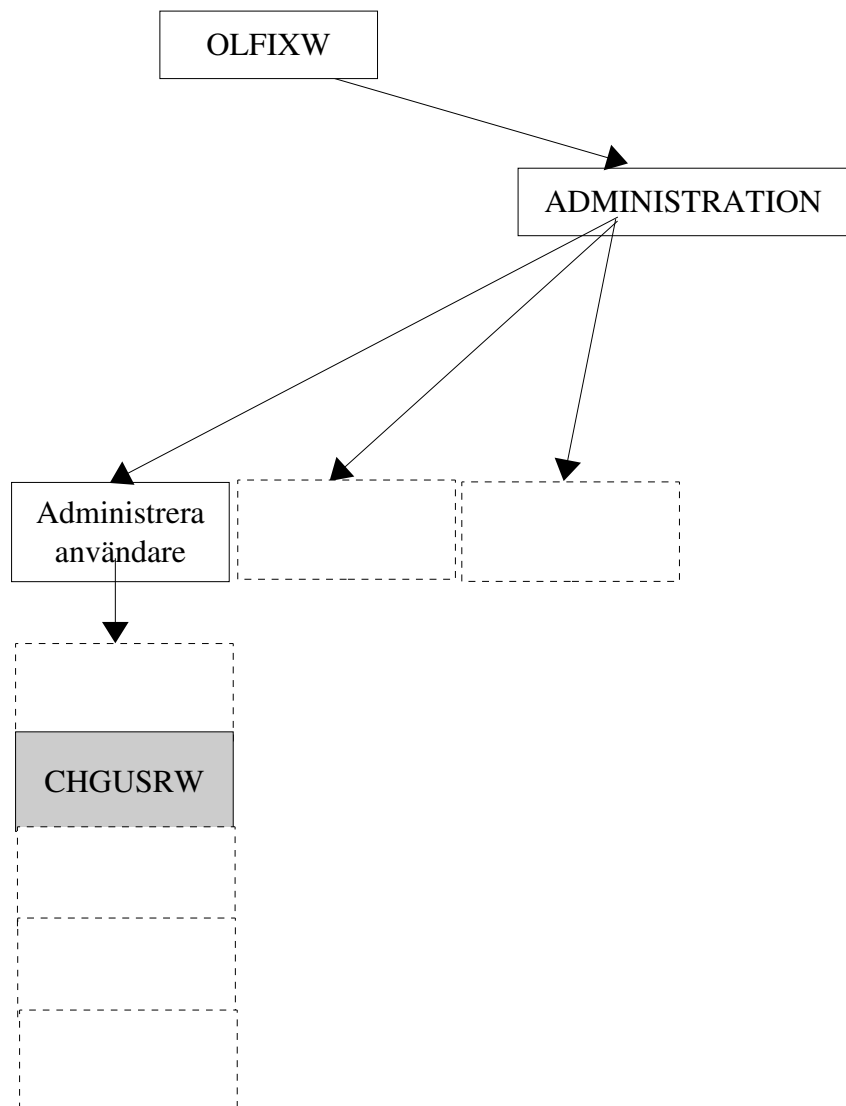
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



CHGUSRW Ändra användare

Användar-ID: KALLE

Namn: Karl Pettersson

Avd: Prod

Grupp: PROD

Hämta Uppdatera Avsluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen USERCHG.

CHGUSRW anropar USERDSP och USERCHG via STYRMAN.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1()); // user OLFIX
process->addArgument("USERDSP"); // OLFIX program
process->addArgument( Userid.toLatin1() );
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr); // Userid
process->addArgument("USERCHG"); // OLFIX funktion
process->addArgument(Userid);
process->addArgument(namn);
process->addArgument(avd);
process->addArgument(grupp);
```

Detta blir:

```
./STYRMAN usr USERDSP Userid
```

och

```
./STYRMAN usr USERCHG Userid namn avd grupp
```

usr är den inloggades userid.

Userid är userid på den användare som man önskar data om.

### Behörighetskrav:

För att kunna köra CHGUSRW behövs behörighet till  
PRGLST  
USERDSP  
USERDSP

## CHGKTOW....Ändra kontodata

CHGKTOW, ett grafiskt program för att ändra info för ett konto. Man måste ange bokföringsår (arid) och kontonummer (ktonr). IB och UB kan inte ändras.

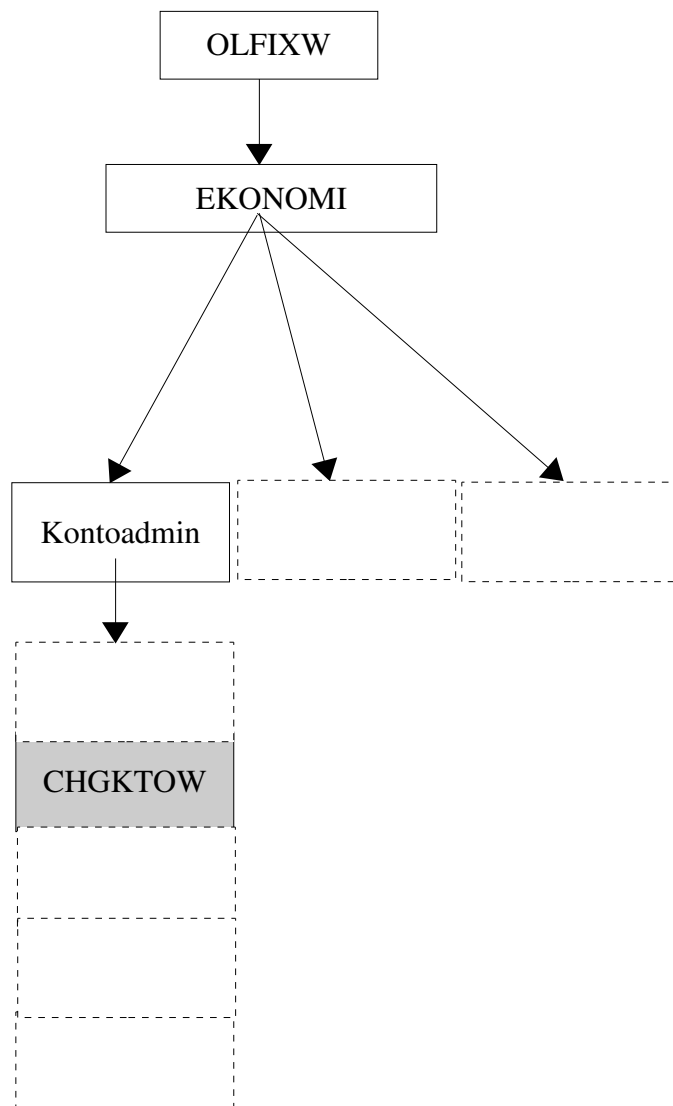
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



CHGKTOW Ändra konto

Bokföringsår:  Obligatoriskt.  
(arid, 2 teck.)

Kontonummer:  Obligatoriskt.

Benämning:

Manuell (J/N):  Obligatoriskt.

Momskod:  Obligatoriskt.

SRUnr:  Obligatoriskt.

Kostnadsställe:

Projekt:

Subkonto:

Kontoplan:  Obligatoriskt.

IB:

UB:

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen KTOCHG.

CHGKTOW anropar KTODSP och KTOCHG via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KTODSP");              // OLFIX funktion
process->addArgument(arid);
process->addArgument(ktonr);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KTOCHG");              // OLFIX funktion
process->addArgument(arid);
process->addArgument(ktonr);
process->addArgument(benamn);
process->addArgument(manuell);
process->addArgument(momskod);
process->addArgument(srunr);
process->addArgument(kst);
process->addArgument(projekt);
process->addArgument(subkonto);
process->addArgument(ktoplan);
```

Detta blir:

```
./STYRMAN usr KTODSP arid ktonr
```

och

```
./STYRMAN usr KTOCHG arid ktonr benamn manuell momskod srunr kst projekt subkonto
ktoplan
```

usr är den inloggades userid.

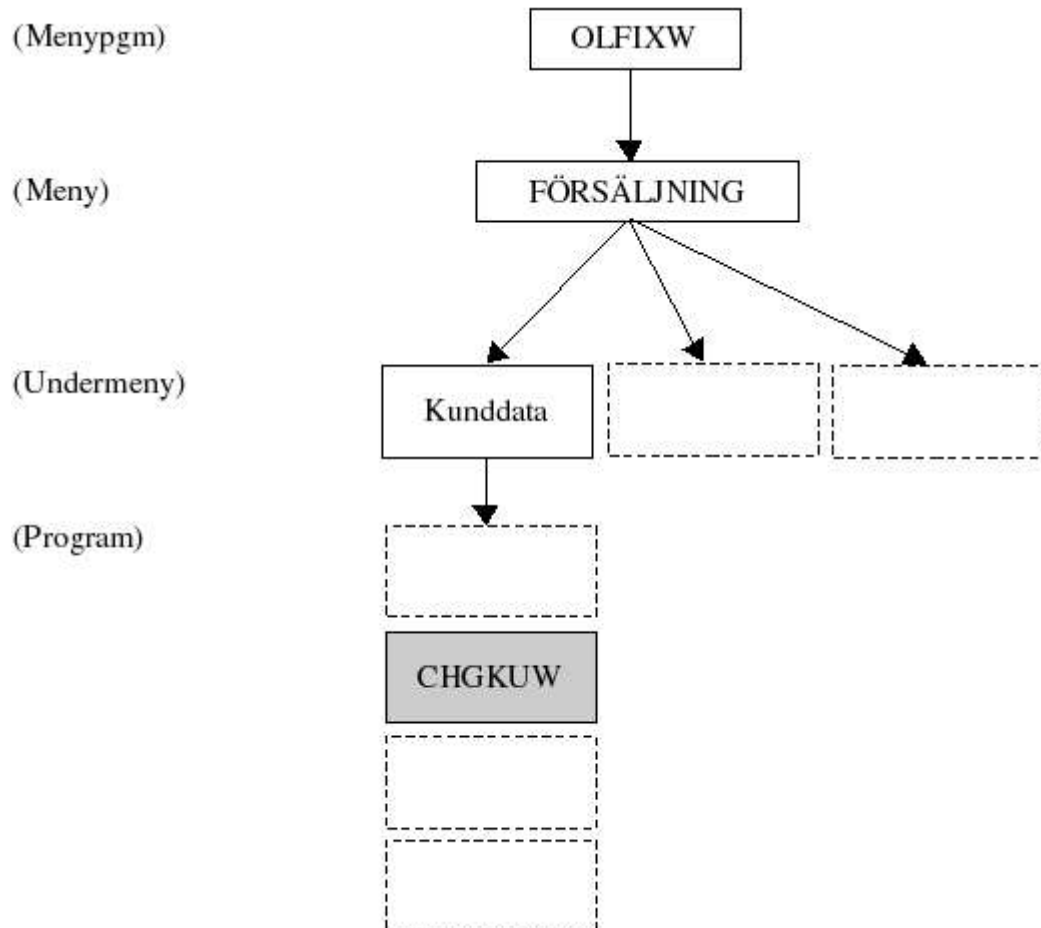
### Behörighetskrav:

För att kunna köra CHGKTOW behövs behörighet till  
PRGLST  
KTOCHG  
KTODSP



## CHGKUW.....Ändra kunddata

CHGKUW, ett grafiskt program för att ändra kunddata. Man måste ange kundnummer (kundid).  
Programmet plockar upp userid från environment.





CHGKUW - Ändra kunddata.

KundID: ..... 4377 Kundlista

Kundnamn: ..... Kund AB

Kundadress: ..... Provgatan 23

Postnummer: ..... 199 97

Postadress: ..... LILLEBY

Land: ..... Sverige

Telefonnummer: ..... 09-999190

Faxnummer: ..... 09-999199

E-mail: ..... info@kund.se

Er Referent: ..... Per Karlsson

Er Ref's telefonnr: .... 09-999191

Er Ref's e-mailadr: .. per.k@kund.se

Vår säljare: ..... Josef Seljare

Distrikt: ..... Öre Kundkategori: ..... Sof

Leveransplats: ..... 001 Leveransvillkor: ..... 001 Leveranssätt: ..... 001

Betalningsvillkor: 1

Valuta: ..... SEK Språkkod: ..... sv

Ordererkännande: .. J Plocklista : ..... J Följesedel: ..... J

Expeditionsavgift: ... J Fraktavgift: ..... J Kravbrev: ..... J

Kreditlimit: ..... 2000.00 Kreditdagar: (null) Kreditkod: ..... (null)

Exportkod: ..... (null) Skattekod: ..... (null) Rabattkod: ..... (null)

Dröjsmålsränta: ..... J Dröjsmålsfaktura: .. J Samlingsfaktura: J

Senaste kravdatum: (null) Skuld: ..... (null)

Orderstock: ..... (null)

Fri text (100 tkn): .... Fritt textfält

Uppdatera Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen KUCHG.

CHGKUW anropar KUDSP och KUCHG via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KUDSP");               // OLFIX funktion
process->addArgument(jundid);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KUCHG");               // OLFIX funktion
process->addArgument(kunddata);
```

Detta blir:

```
./STYRMAN usr KUDSP kundnr
```

och

```
./STYRMAN usr KUCHG kunddata
```

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra CHGKUW behövs behörighet till  
PRGLST  
KUCHG  
KUDSP

## CHGVALW....Ändra valutadata

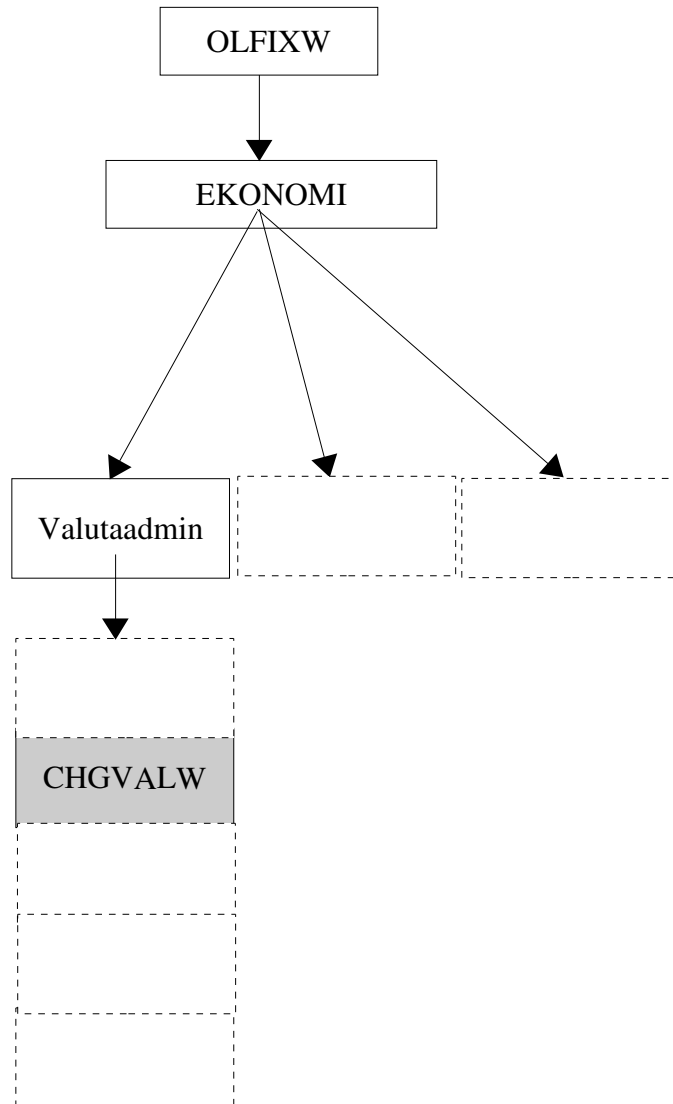
CHGVALW, ett grafiskt program för att ändra info för ett konto.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



OLFIX - CHGVALW Ändra valuta

Valuta:

Beteckning:

Land:

Köp:

Sälj:

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen VALCHG.

CHGVALW anropar VALDSP och VALCHG via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VALDSP");              // OLFIX funktion
process->addArgument(valuta);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VALCHG");              // OLFIX funktion
process->addArgument(valuta);
process->addArgument(land);
process->addArgument(salj);
process->addArgument(kop);
process->addArgument(beteckning);
```

Detta blir:

```
./STYRMAN usr VALDSP valuta
```

och

```
./STYRMAN usr VALCHG valuta land salj kop beteckning
```

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra CHGVALW behövs behörighet till  
PRGLST  
VALCHG  
VALDSP



# DAGBOKW.....Dagbok

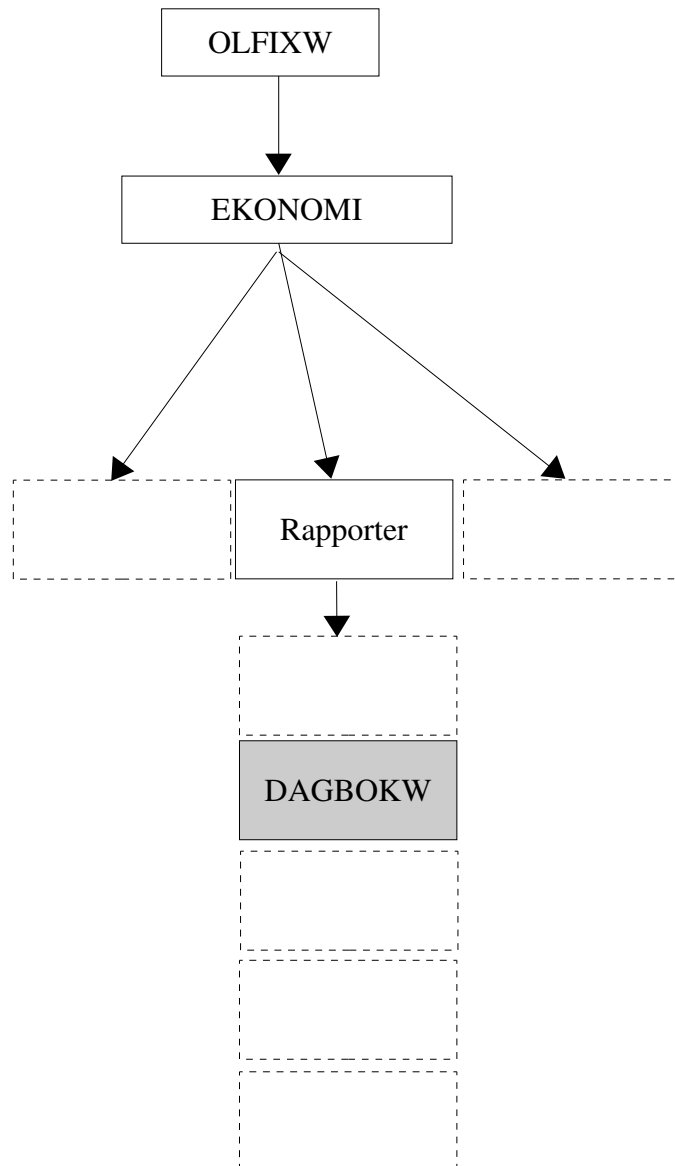
DAGBOKW är ett grafiskt program för att skriva ut dagbok.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



**DAGBOKW - Dagbok**

**Dagbok**

Datum: 2004-04-11

Bokföringsår: **AD** Fr. o. m. datum: **2003-07-30** T.o.m. datum: **2003-09-11**

CSV-format:	<input checked="" type="radio"/>	Startar Kspread med importerade data.
Utskriftsformat:	<input checked="" type="radio"/>	Skapar en rapport med Kugar färdig att skriva ut.

**Skapa rapport** **Avbryt**



För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen DBOKRPT, VERHDSP och FTGDSP.

DAGBOKW anropar DBOKRPT, VERHDSP och FTGDSP via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("DBOKRPT");             // OLFIX funktion
process->addArgument(bar);
process->addArgument(fromdatum);
process->addArgument(tomdatum);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("FTGDSP");              // OLFIX funktion
process->addArgument("FNAMN");
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VERHDSP");             // OLFIX funktion
process->addArgument(bar);                   // Bokföringsår
```

Detta blir:

```
./STYRMAN usr DBOKRPT bar fromdatum tomdatum
```

och

```
./STYRMAN usr FTGDSP "FNAMN"
```

och

```
./STYRMAN usr VERHDSP bar
```

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DAGBOKW behövs behörighet till

PRGLST

DAGBOKW

DBOKRPT

FTGDSP

VERHDSP



Kugar

Arkiv Kör Inställningar Hjälp

2004-04-11

**PROGRAM AB**

Dagbok För perioden 2003-07-30 -- 2003-09-11

Konto	Vemr	Verifikationstext	Debet	Kredit
<b>Ver:</b>	<b>1</b>	<b>2003-07-30 Lån eget kapital</b>		
2081		Aktiekapital	0.00	300000.00
2330		Checkräkningskredit	300000.00	0.00
<b>Ver:</b>	<b>2</b>	<b>2003-07-30 Insättning checkräkningskredit</b>		
2330		Checkräkningskredit	499450.00	0.00
2350		Banklån	0.00	500000.00
8490		Ovriga finansiella kostnader	550.00	0.00
<b>Ver:</b>	<b>3</b>	<b>2003-07-30 Lokalha 1.a kv 2003</b>		
2330		Checkräkningskredit	0.00	60000.00
2641		Ingående moms	12000.00	0.00
5010		Lokalhyra	48000.00	0.00
<b>Ver:</b>	<b>7</b>	<b>2003-08-01 Mässingplåt. 1 mm. 0,5 m3</b>		
2440		Leverantörsskulder	0.00	2000.00
2641		Ingående moms	500.00	0.00
4010		Materialkostnad	1500.00	0.00
<b>Ver:</b>	<b>8</b>	<b>2003-08-02 Al-plåt. 1 mm. 1 m3</b>		
2440		Leverantörsskulder	0.00	2000.00
2641		Ingående moms	500.00	0.00
4010		Materialkostnad	1500.00	0.00
<b>Ver:</b>	<b>9</b>	<b>2003-08-03 Jämplåt. 1 mm. 1 m3</b>		
2440		Leverantörsskulder	0.00	3000.00
2641		Ingående moms	750.00	0.00
4010		Materialkostnad	2250.00	0.00
<b>Ver:</b>	<b>10</b>	<b>2003-08-04 Jämplåt. 0,6 mm. 1,5 m3</b>		
2440		Leverantörsskulder	0.00	3000.00
2641		Ingående moms	750.00	0.00
4010		Materialkostnad	2250.00	0.00
<b>Ver:</b>	<b>11</b>	<b>2003-08-05 Jämplåt. 0,3 mm. 3 m3.</b>		

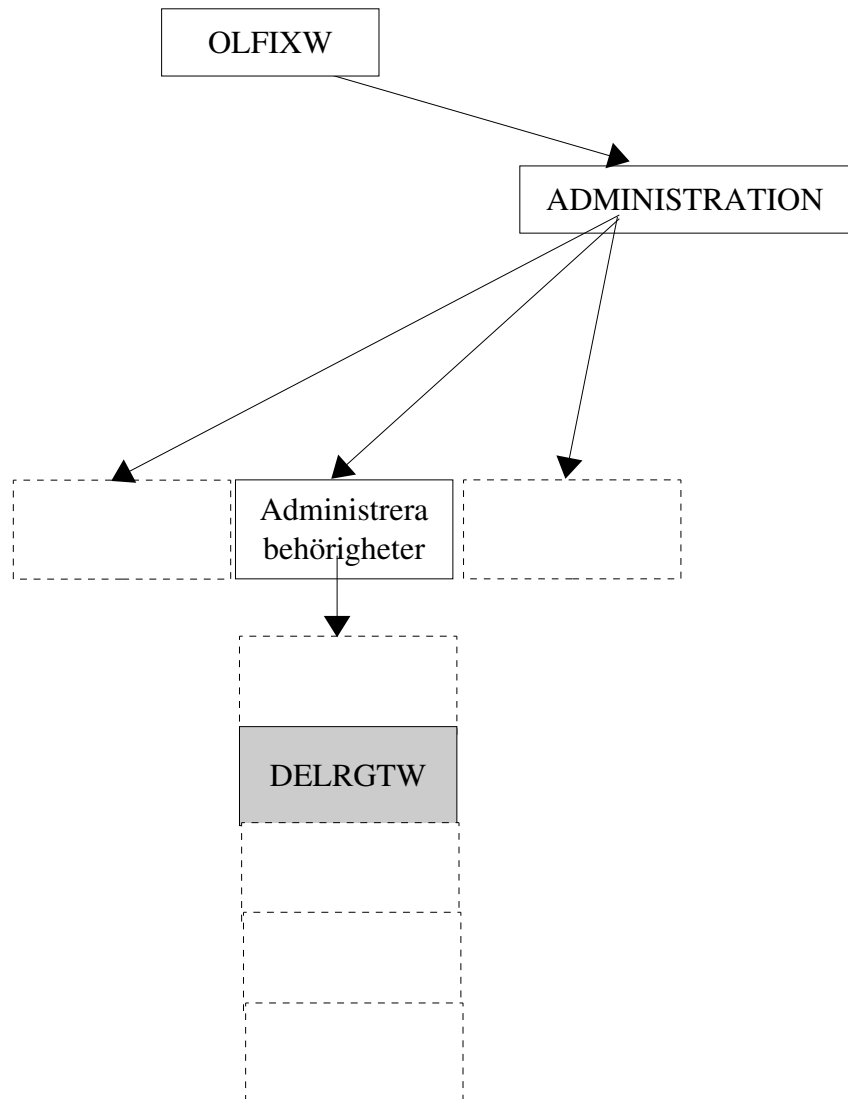
## DELRTW....Radera behörighet

DELRTW är ett grafiskt program för att ta bort en behörighet för en användare.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen RGTCHK och RGTDEL.

RGTDELW anropar RGTCHK och RGTDEL via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("RGTCHK");              // OLFIX funktion
process->addArgument(userid);
process->addArgument(funk);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("RGTDEL");              // OLFIX funktion
process->addArgument(usrid);
process->addArgument(funk);
```

Detta blir:

```
./STYRMAN usr RGTCHK usrid funk
```

samt

```
./STYRMAN usr KTODEL usrid funk
```

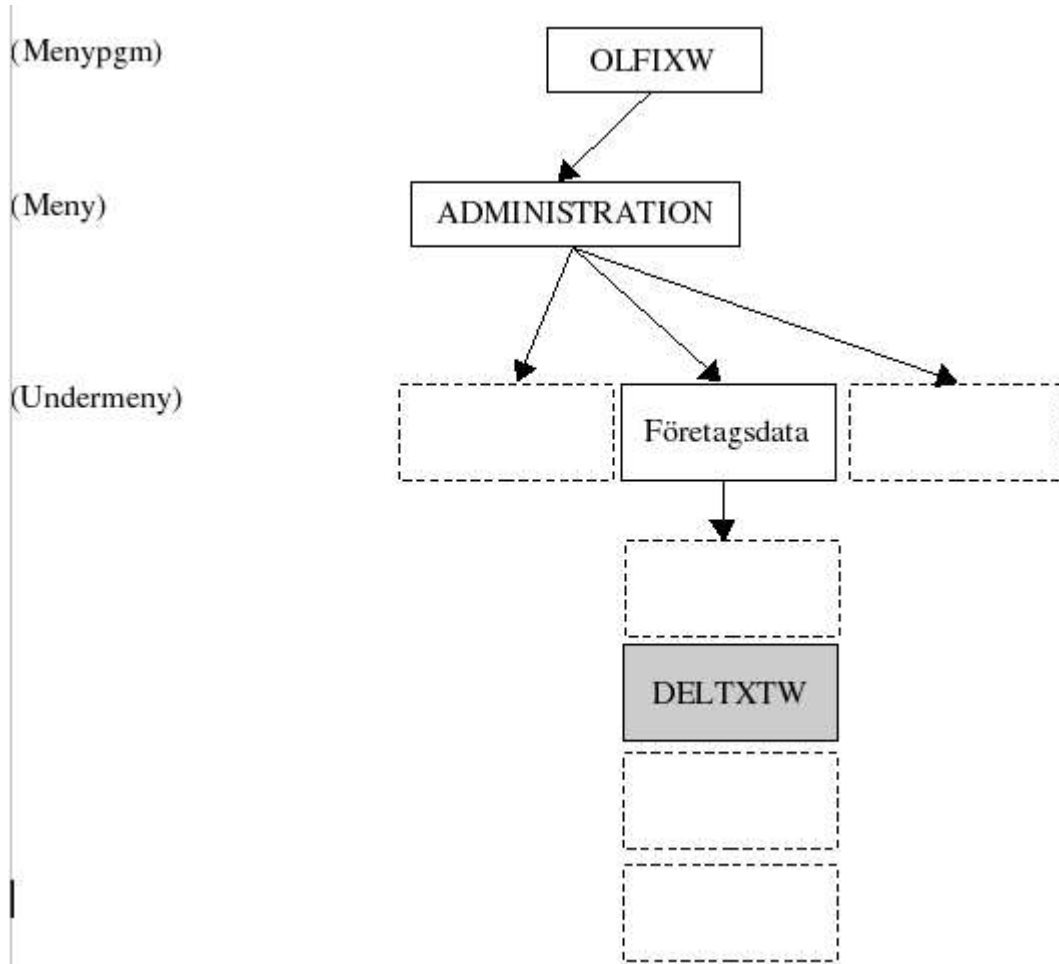
usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DELRGTW behövs behörighet till  
PRGLST  
RGTCHK  
RGTDEL

## DELXTW.....Radera text.

DELXTW är ett grafiskt program för att ta bort en post ur TEXTTREG.  
Programmet plockar upp userid från environment.







För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionerna TXTDSP ochTXTDEL

DELXTW anropar TXTDSP och TXTDEL via STYRMAN.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // Userid
process->addArgument( "TXTDSP");              // OLFIX funktion
process->addArgument(textnr);
```

och

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                    // Userid
process->addArgument( "TXTDEL");              // OLFIX funktion
process->addArgument(textnr);
```

Detta blir:

```
./STYRMAN userid TXTDSP textnr
```

och

```
./STYRMAN userid TXTDEL textnr
```

userid är den inloggades userid.

### **Behörighetskrav:**

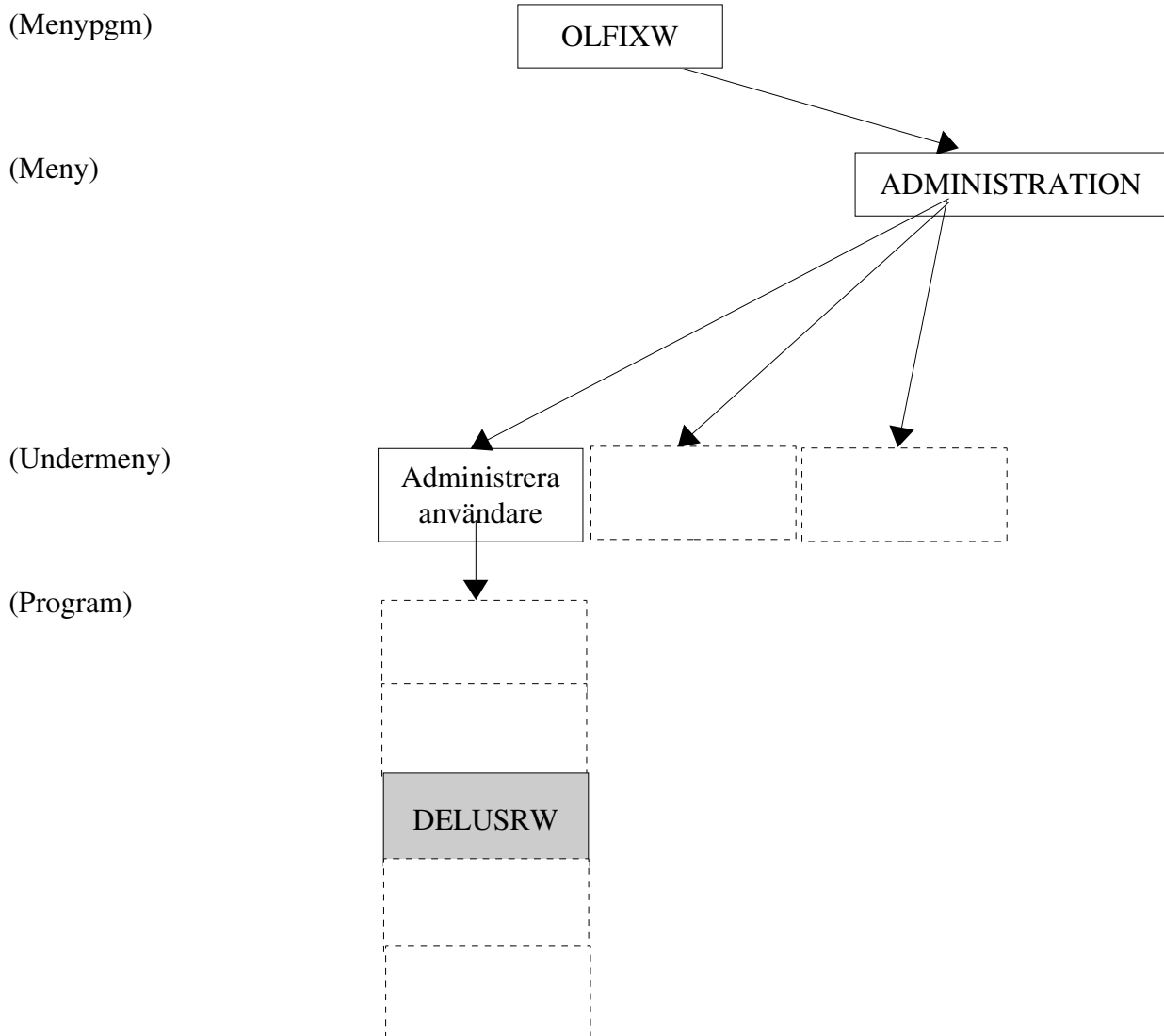
För att kunna köra DELXTW behövs behörighet till  
PRGLST  
TXTDEL  
TXTDSP

## DELUSRW.....Radera användare

DSPUSRW, ett grafiskt program för att visa information för en användare.

För att använda programmet krävs behörighet till funktionerna RGTDSP, USERDSP, RGTDEL och USERDEL.

Programmet plockar upp userid från environment.



DELUSRW Ta bort en användare och dennes behörigheter

UserId:

Namn:

Avd:

Grupp:

Behörigheter

UserId	Behörighet
TESTARE	USERADD
TESTARE	USERDSP

Radera användare

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionerna USERDSP, RGTDSP, RGTDEL och USERDEL.

DELUSRW anropar USERDSP, RGTDSP, RGTDEL och USERDEL via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1());           // user OLFIX
process->addArgument("USERDSP");               // OLFIX program
process->addArgument(Userid.toLatin1());
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1());
process->addArgument("RGTDSP");                // OLFIX
program
process->addArgument(Userid.toLatin1());
```

och

```
const char *userp = getenv("USER");
QString usr(userp);
process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1());
process->addArgument("RGTDEL");                // OLFIX program
process->addArgument(anvID.toLatin1());
process->addArgument(fncID.toLatin1());
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1());
process->addArgument("USERDEL");              // OLFIX program
process->addArgument(anvID.toLatin1());
```

Detta blir:

```
./STYRMAN userid1 RGTDSP userid2
```

och

```
./STYRMAN userid1 USERDSP userid2
```

och

```
./STYRMAN userid1 RGTDEL userid2 trnsid (funktion)
```

och

```
./STYRMAN userid1 USERDEL userid2
```

userid1 är den inloggades userid.

userid2 är userid på den användare som man önskar data om.

**Behörighetskrav:**

För att kunna köra DELUSRW behövs behörighet till

PRGLST

RGTDEL

RGTDSP

USERDEL

USERDSP

## DELVALW.....Radera valuta

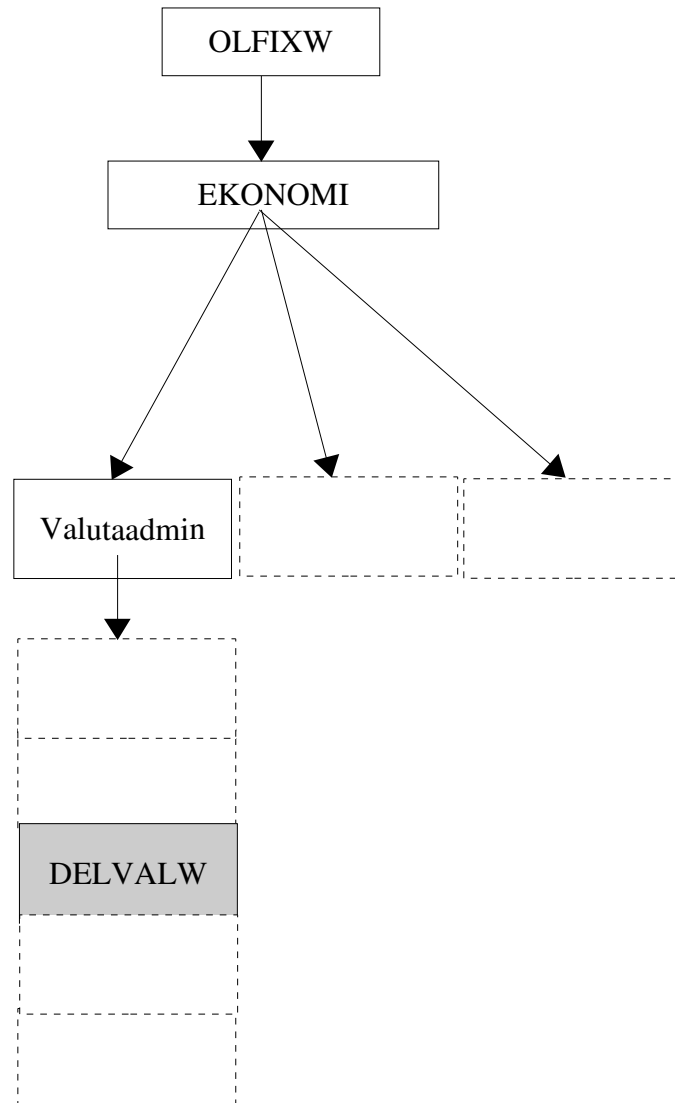
CHGVALW, ett grafiskt program för att ändra info för en valuta.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



OLFIX - DELVALW Ta bort en valuta

Valuta:

Beteckning:

Land:

Köp:

Sälj:

Radera valuta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen VALDEL.

DELVALW anropar VALDSP och VALDEL via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VALDSP");              // OLFIX funktion
process->addArgument(valuta);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VALDEL");              // OLFIX funktion
process->addArgument(mynt.latin1() );       // valuta
```

Detta blir:

```
./STYRMAN usr VALDSP valuta
```

och

```
./STYRMAN usr VALDEL valuta
```

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DELVALW behövs behörighet till

PRGLST

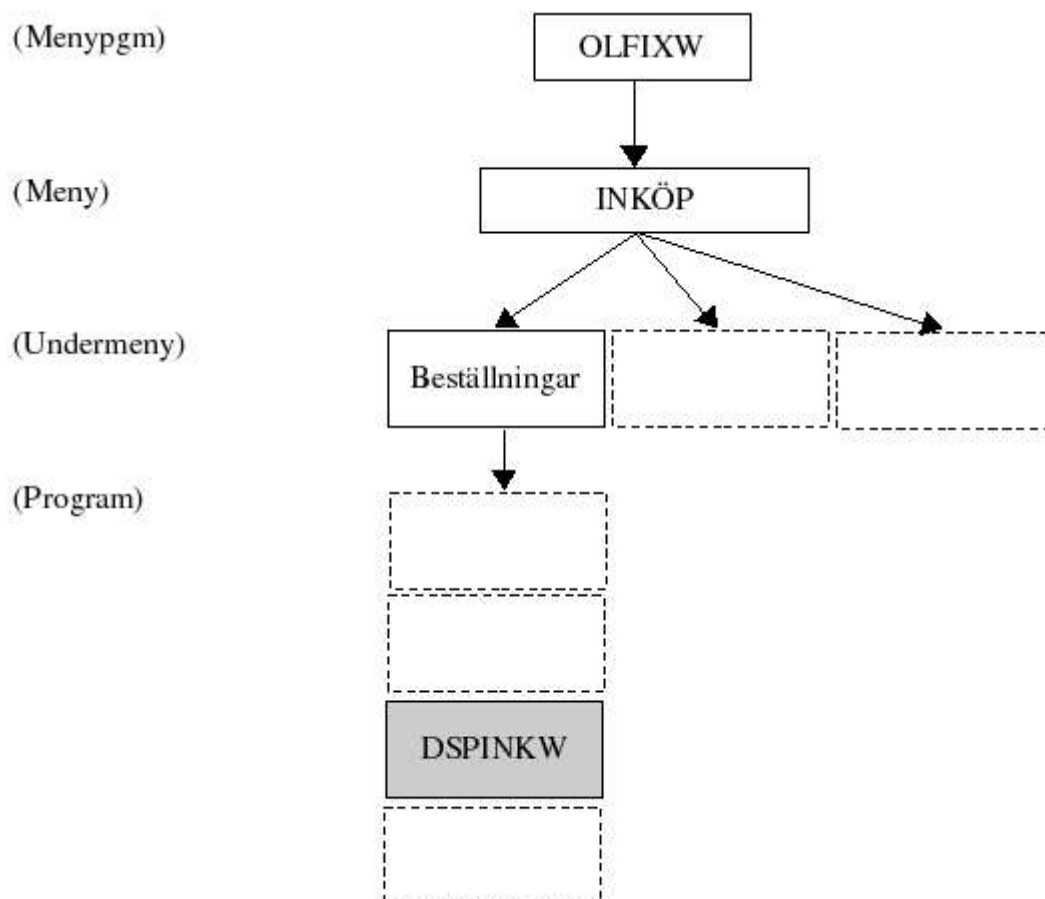
VALDSP

VALDEL



## DSPINKW.....Visa en inköpsorder

DSPINKW, ett grafiskt program för att visa information om en inköpsorder.  
Programmet plockar upp userid från environment.



Leverantörsnr: ..... 9999
Beställningsnr: ..... 18
Beställningsdatum: ... 2003-12-24

Leverantörens postadress
**Orderhuvud**

Mottagarens Leveransadress/Godsadress

Namn: ..... Testleverantör AB
Adress: ..... Delivery Street 1C
Postnummer: ..... 199 99
Postadress: ..... LEVSTAD
Land: ..... Sverige
Godsmärke: ..... P-order
Vår referent: ... Jan Pihlgren
Leveransdatum: ..... 2004-01-10
Kommentar: ..... Kommentar

Vårt kundnr: ... 98765

Er referens: ... Caroline Seljare

Valuta: SEK

Betaltingsvillkor: 30 dagar netto
Leveransvillkor: EXW
Leveranssätt: ASG kundnr:99901111
Eftertext: Ordererkännande önskas inom 3 arbetsdagar (om ej redan bekräftats) Ange alltid vårt artikelnummer på följesedel och faktura.

**Detaljrader**

Radnr	Artikelnr	Benämning	Lev.vecka	Beställt Antal	Lev:Antal	Resterande ant.	Pris	Summa
10	1173-0911	Spänningsregulator positiv	402	10.00	0.00	10.00	1.50	15.00
20	1173-0963	Spänningsregulator negativ	402	10.00	10.00	0.00	5.50	0.00
30	1173-1175	Spänningsregulator positiv	402	10.00	0.00	10.00	30.00	300.00
40	1173-1445	D/A Omvandlare 12-bit	402	10.00	0.00	10.00	95.00	950.00
50	1173-1447	Timerkrets	402	10.00	0.00	10.00	5.45	54.50
60	1173-6910	Quadruple 2input NAND gate	402	10.00	0.00	10.00	1.45	14.50

Stäng

Orderstatus: N
Order bekräftad: E

Ursprunglig ordertotal: 1389.00
Beställning Total: 1334.00

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen INKDSP och INKRLST.

DSPINKW anropar INKDSP och INKRLST via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("INKHDSP");               // OLFIX funktion
process->addArgument(inkordernr);
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");             // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("INKRLST");               // OLFIX funktion
process->addArgument(inkordernr);
```

INKRLST hämtar också benämning 1 från ARTIKELREG med hjälp av funktionens SQL-sats.

Detta blir:

```
./STYRMAN usr INKHDSP inkordernr
```

och

```
./STYRMAN usr INKRLST inkordernr
```

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DSPINKW behövs behörighet till  
PRGLST  
INKDSP  
INKRLST

## DSPKSTW.....Visa kostnadsställdata

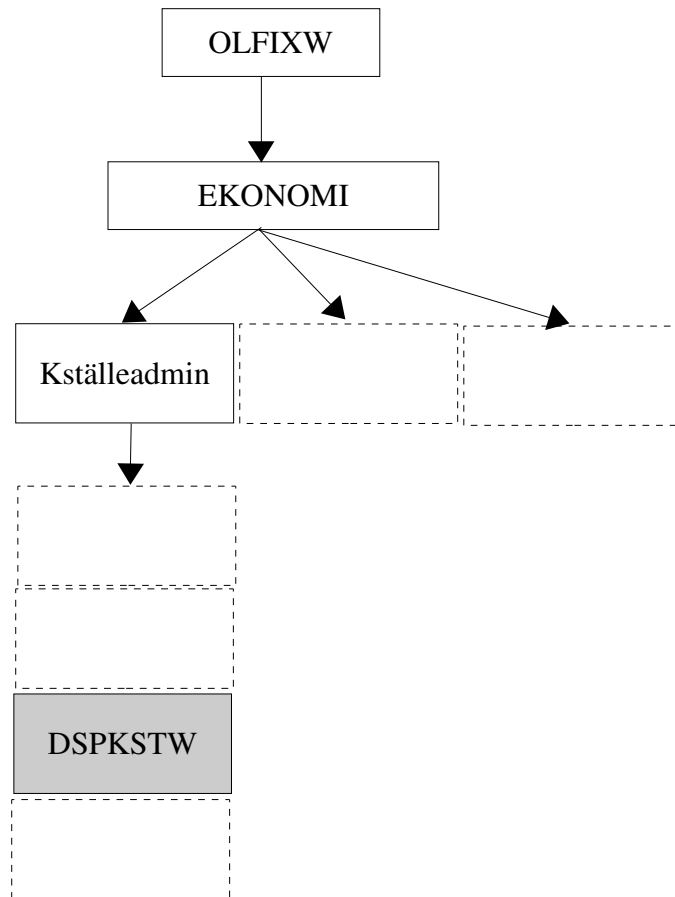
DSPKSTW, ett grafiskt program för att visa information om ett kostnadställe.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



OLFIX - DSPKSTW Visa information för ett kostnadsställ

Bokföringsår: AD

Kostnadsställe 9037 Hämta

Benämning: Projekt Titan

Sluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen KSTDSP.

DSPKSTW anropar KSTDSP via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("KSTDSP");                // OLFIX funktion
process->addArgument(bar);
process->addArgument(kstalle);
```

Detta blir:

```
./STYRMAN usr KSTDSP bar kstalle
```

usr är den inloggades userid.

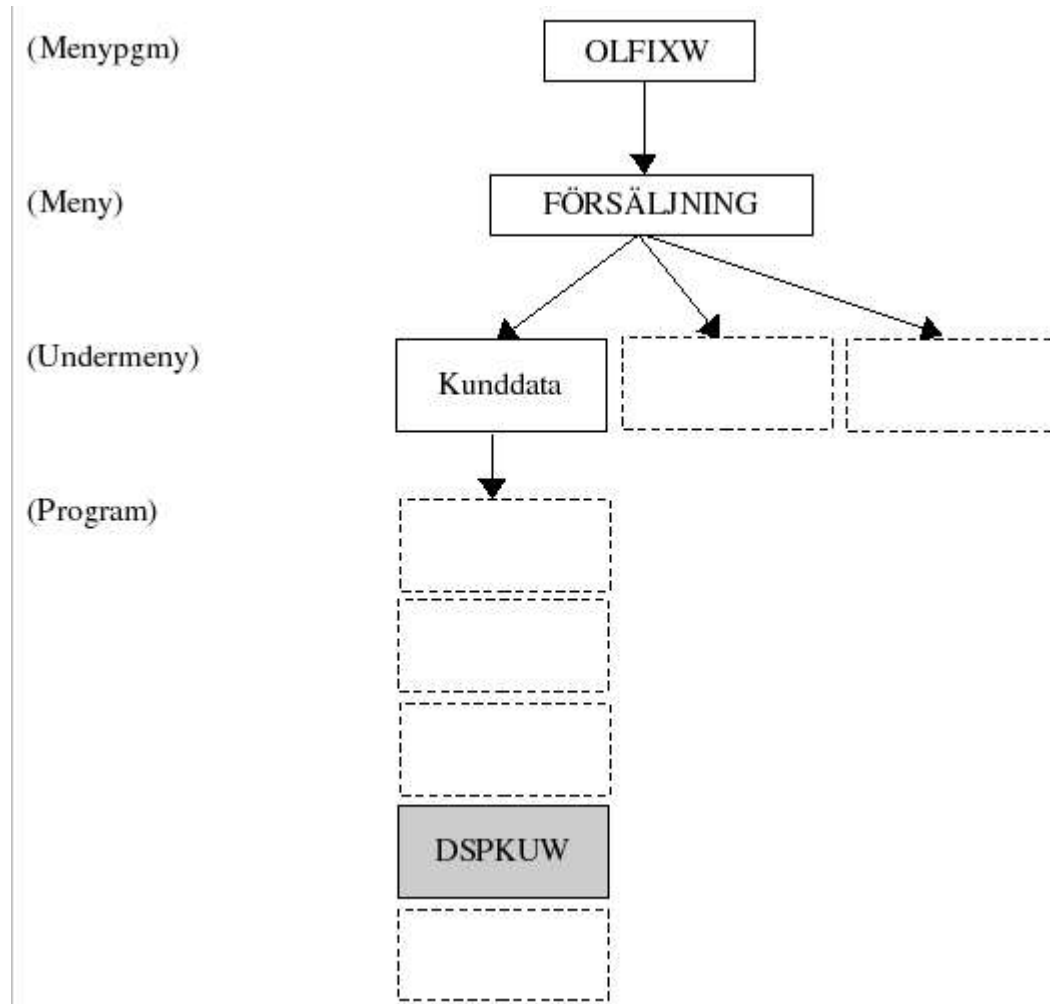
#### **Behörighetskrav:**

För att kunna köra DSPKSTW behövs behörighet till  
PRGLST  
KSTDSP



## DSPKUW.....Visa kunddata

DSPKUW, ett grafiskt program för att visa information om en kund.  
Programmet plockar upp userid från environment.





DSPKUW - Visa kunddata.

KundID: ..... 4379

Kundnamn: ..... Småkund AB

Kundadress: ..... Myrstigen 32

Postnummer: ..... 199 02

Postadress: ..... SMÅSTAD

Land: ..... Sverige

Telefonnummer: ..... 09-129990

Faxnummer: ..... 09-129999

E-mail: ..... info@smakund.se

Er Referent: ..... Lillemor Andrén

Er Ref's telefonnr: .... 09-129991

Er Ref's e-mailadr: .. lillemor.a@smakund.se

Vår säljare: ..... Caroline Seljare

Distrikt: ..... Öre Kundkategori: ..... Sft

Leveransplats: ..... 002 Leveransvillkor: ..... 001 Leveranssätt: ..... 001

Betalningsvillkor: ..... 1

Valuta: ..... SEK Språkkod: ..... sv

Ordererkännande: .. J Plocklista : ..... J Följesedel: ..... J

Expeditionsavgift: ... J Fraktagift: ..... J Kravbrev: ..... J

Kreditlimit: ..... 5000.00 Kreditdagar: 30 Kreditkod: ..... 001

Exportkod: ..... 001 Skattekod: ..... 001 Rabattkod: ..... 001

Dröjsmålsränta: .... J Dröjsmålsfaktura: .. J Samlingsfaktura: J

Senaste kravdatum: (null) Skuld: ..... (null)

Orderstock: ..... (null)

Fri text (100 tkn): .... Den lilla kunden

KLAR Avbryt

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen KUDSP.

DSPKUW anropar KUDSP via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("KUDSP");                 // OLFIX funktion
process->addArgument(kundid);
```

Detta blir:

```
./STYRMAN usr KUDSP kundnr
```

usr är den inloggades userid.

### **Behörighetskrav:**

För att kunna köra DSPKUW behövs behörighet till  
PRGLST  
KUDSP

## DSPLEVW.....Visa leverantörsdata

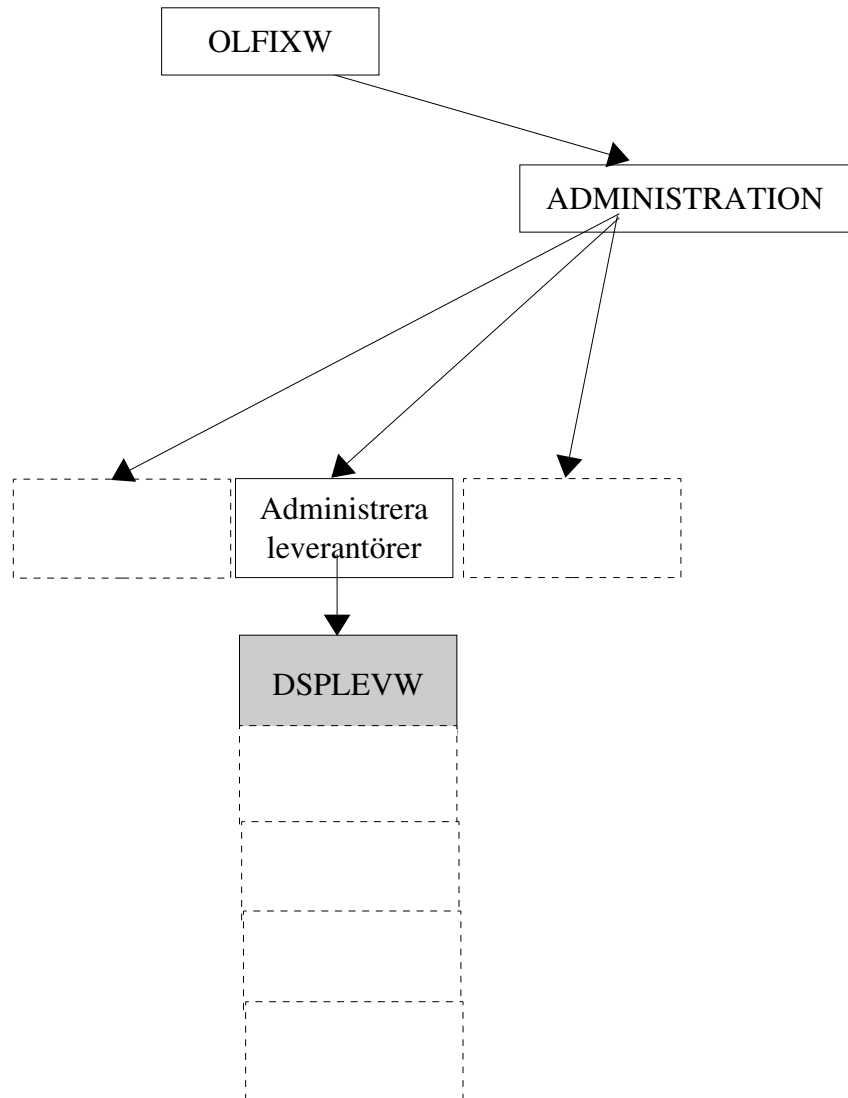
DSPLEVW är ett grafiskt program för att visa data om en leverantör.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



DSPLEWV - Visa leverantörsdata.

Leverantörsnummer:	123	Hämta
Organisationsnr:	559999-1112	
Leverantörsnamn:	Leverantör AB	
Leverantörsadress:	Tillverkningsgatan 3	
Postnummer: .....	199 99	
Postadress: .....	STORSTAD	
Land: .....	Sverige	
Telefonnummer: .....	09-999999	
Faxnummer: .....	09-999998	
Telex: .....	99999	
E-mail: .....	info@leverantor.se	
Referent: .....	Eva Säljare	
Ref's telefonnr: .....	09-999997	
Momskod: .....	2	
Kontonummer: .....	2110	
Postgironr: .....	4559988-8	
Bangironr: .....	5999-1199	
Kundnr: .....	329876	
Leverantörsskuld:	0.00	

OK

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen LEVDSP.

DSPLEVW anropar LEVDSP via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

//      qDebug("levnr=%s",levnr.latin1());

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("LEVDSP");              // OLFIX funktion
process->addArgument(levnr);
```

Detta blir:

```
./STYRMAN usr LEVADD levnr
```

#### **Behörighetskrav:**

För att kunna köra DSPLEVW behövs behörighet till

PRGLST

LEVDSP

## DSPUSRW.....Visa användardata

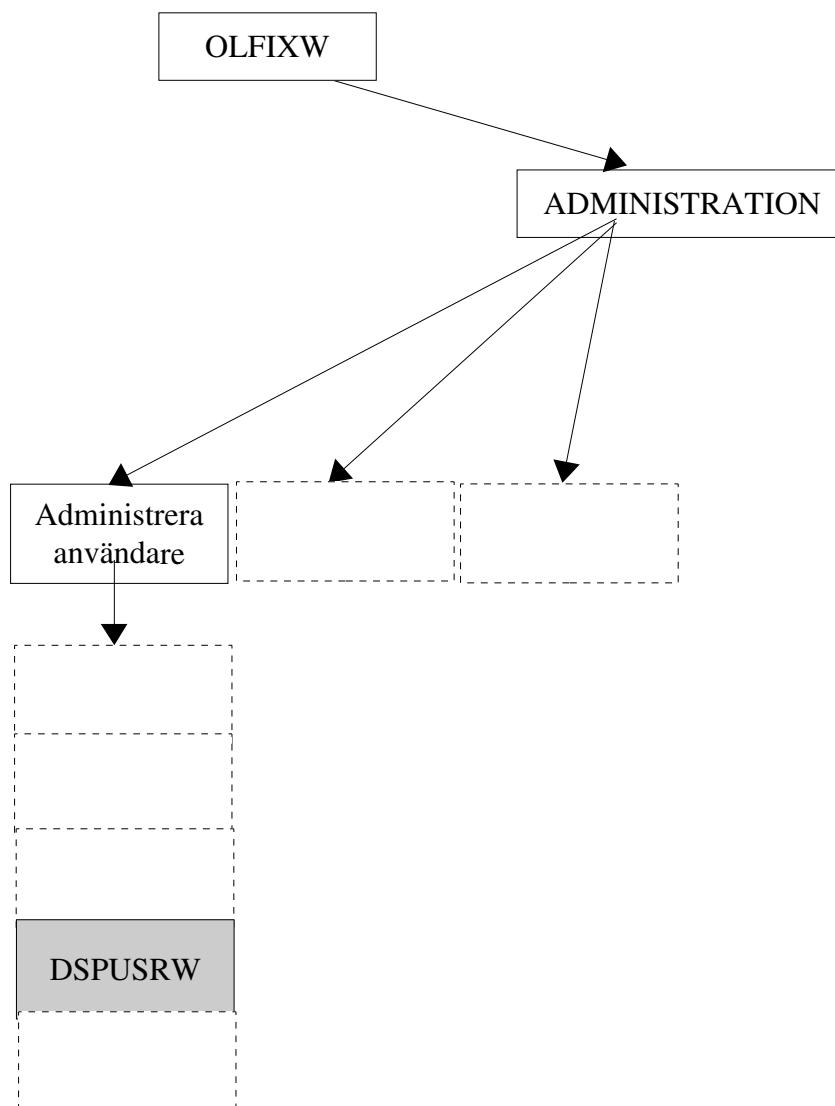
DSPUSRW, ett grafiskt program för att visa information för en användare.  
För att använda programmet krävs behörighet till funktionerna RGTDSP och USERDSP.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Popdown)

(Popdown)



DSPUSRW Visa information för en användare

UserId: JAN

Namn: Jan Pihlgren

Avd: Ekonomi

Grupp: Stab

Behörigheter

JAN	BARADD
JAN	BOKF
JAN	KTOADD
JAN	KTOVIEW
JAN	RGTADD
JAN	RGTDSP
JAN	RGTLST
JAN	USERADD
JAN	USERDSP
JAN	USERLST

OK Sluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen USERDSP.

DSPUSRW anropar USERDSP och RGTDSP via STYRMAN.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1()); // user OLFIX
process->addArgument("USERDSP"); // OLFIX program
process->addArgument(Userid.toLatin1());
```

och

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.toLatin1());
process->addArgument("RGTDSP"); // OLFIX program
process->addArgument(Userid.toLatin1());
```

Detta blir:

```
./STYRMAN userid1 RGTDSP userid2
```

och

```
./STYRMAN userid1 USERDSP userid2
```

userid1 är den inloggades userid.

userid2 är userid på den användare som man önskar data om.

### Behörighetskrav:

För att kunna köra DSPUSRW behövs behörighet till  
PRGLST  
USERDSP  
RGTDSP



## DSPVALW.....Visa valutainformation

DSPVALW, ett grafiskt program för att visa information om ett kostnadställe.  
Programmet plockar upp userid från environment.

(Menypgm)

OLFIXW

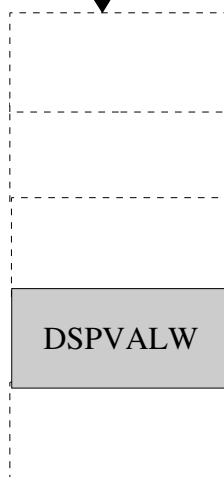
(Meny)

EKONOMI

(Undermeny)

Valutaadmin

(Program)



OLFIX - DSPVALW Visa information för en valut

Valuta: USD

Beteckning: Dollar

Land: USA

Köp: 8.97

Sälj: 8.97

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen VALDSP.

DSPVALW anropar VALDSP via STYRMAN.

```
const char *userp = getenv("USER");          // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                    // user OLFIX
process->addArgument("VALDSP");               // OLFIX funktion
process->addArgument(valuta);
```

Detta blir:

```
./STYRMAN usr VALDSP bar kstalle
```

usr är den inloggades userid.

#### **Behörighetskrav:**

För att kunna köra DSPVALW behövs behörighet till

PRGLST

VALDSP

## HUVBOKW.....Huvudbok

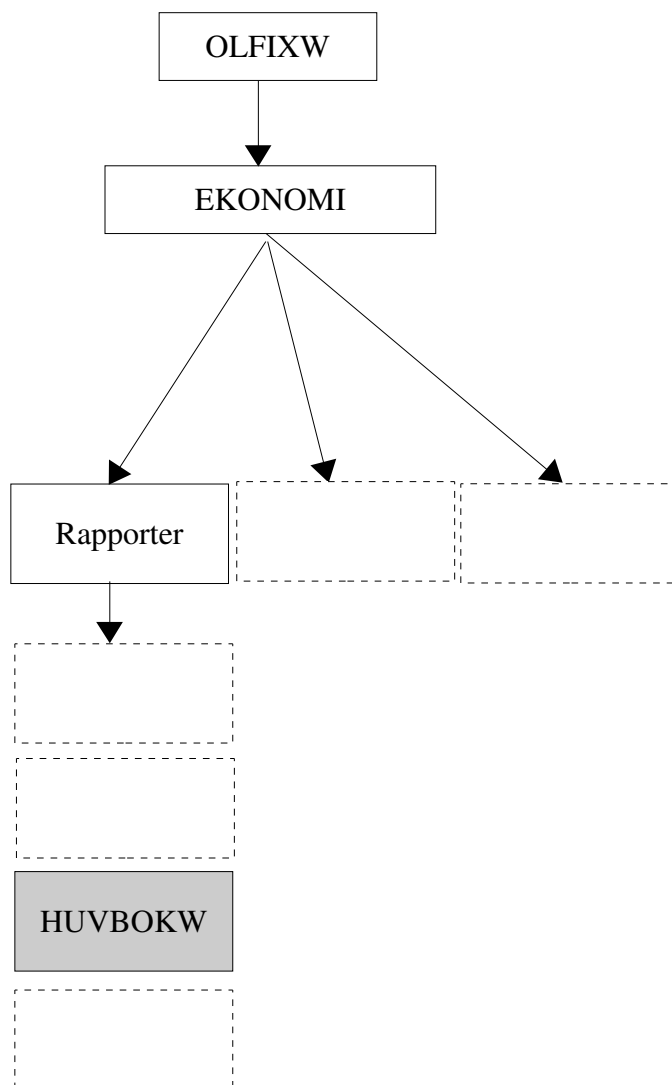
HUVBOKW, ett grafiskt program för att skriva ut huvudboken.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)





HUVBOKW - Huvudbok

### Huvudbok

Datum: 2004-04-09

Bokföringsår: AD Fr. o. m. datum: 2003-07-30 T.o.m. datum: 2003-09-11

CSV-format:		Startar Kspread med importerade data.
Utskriftsformat:		Skapar en rapport med Kugar färdig att skriva ut.

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen HBOKRPT.

HUVBOKW anropar HBOKRPT via STYRMAN.

```
process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr); // user OLFIX
process->addArgument("HBOKRPT"); // OLFIX funktion
process->addArgument(bar); // Bokföringsår
process->addArgument(fromdatum); // Från och med datum
process->addArgument(tomdatum); // Till och med datum
```

Detta blir:

```
./STYRMAN usr HBOKRPT bar fromdatum tomdatum
```

usr är den inloggades userid.

Och VERHDSP

```
process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr); // userid
process->addArgument("VERHDSP"); // OLFIX funktion
process->addArgument(bar); // Bokföringsår
```

Detta blir:

```
./STYRMAN usr VERDSP bar
```

Samt FTGDSP

```
process = new QProcess();
process->addArgument("./STYRMAN"); // OLFIX styrprogram
process->addArgument(usr); // userid
process->addArgument("FTGDSP"); // OLFIX funktion
process->addArgument("FNAME"); // Företagsnamn
```

Detta blir:

```
./STYRMAN usr FTGDSP FNAME
```

### Behörighetskrav:

För att kunna köra HUVBOKW behövs behörighet till

PRGLST

FTGDSP

VERHDSP

HBOKRPT

## LEVFAKTW.....Registrera fakturor

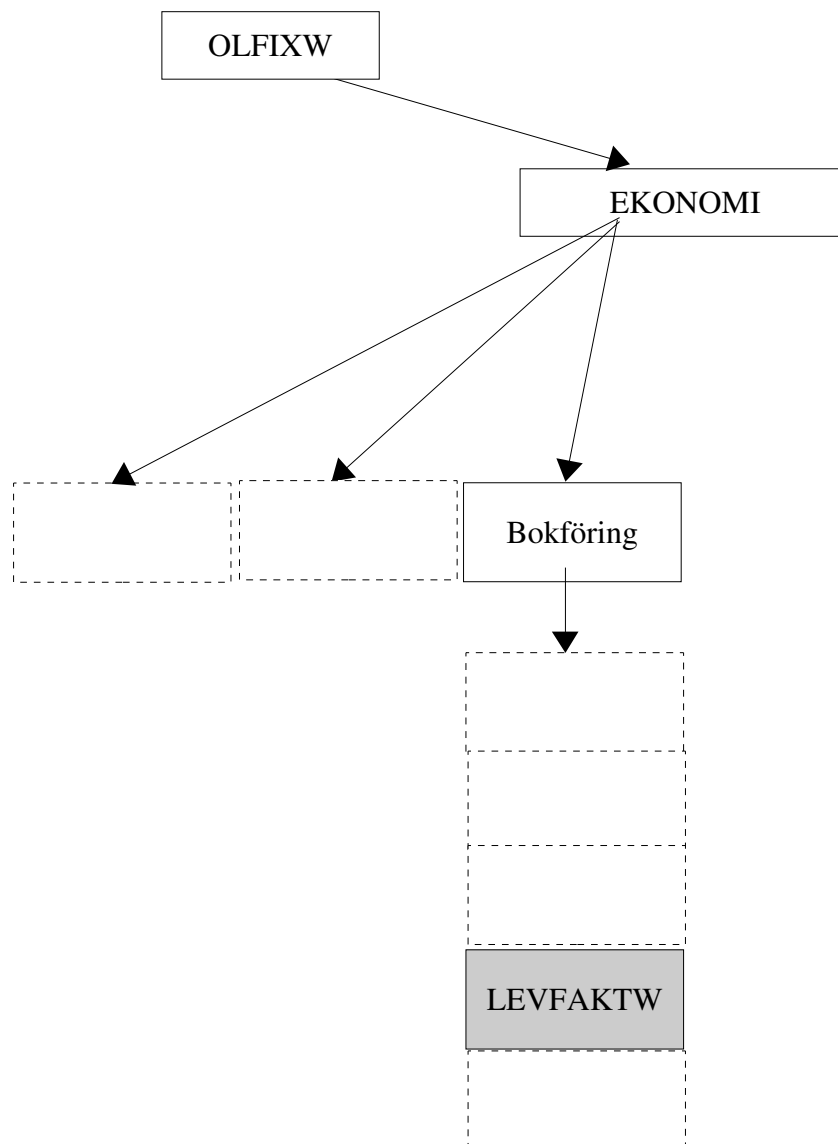
LEVFAKTW, ett grafiskt program för att registrera leverantörsfakturor. Dessutom bokförs fakturorna. Omräkning sker av fakturerat belopp från angiven valuta och valutakurs till SEK. Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



## Funktionsbeskrivning.

När programmet startas plockas dagens datum upp och läggs in som **Registreringsdatum**.

Utifrån aktuellt datum letas aktuellt bokföringsår fram och presenteras i **Bokföringsår**.

Programmet hämtar också flagga för automatkontering. Om flaggan är satt till något annat än **J** så kommer ingen bokföring att ske, bara registrering till leverantörsregistret.

Bokföringsår ligger sedan till grund för vilken kontoplan som listas. Även en förteckning över registrerade leverantörer visas.

Genom att välja en leverantör från listan (genom att klicka på önskad leverantör) presenteras leverantörsID i fältet **LeverantörsID**. LeverantörsID kan också skrivas in.

När man sedan trycker Enter så hämtas aktuella leverantörsdata och presenteras.

Även **Betalningsvillkor** i dagar, aktuell **Valuta** med aktuell kurs, **Kontonr**, **Momskontonr** och **Momskod** hämtas. Med utgångspunkt från momskoden hämtas aktuell momssats (i procent) och presenteras i fältet **Moms %**.

Värdet i **Betalningsvillkor** används för att räkna fram **Förfallodatum**.

Värdet i **Valutakurs** används för att räkna om **Fakturabelopp** i SEK. Det omräknade fakturabeloppet skrivs in i fältet **Bokfört belopp (SEK)**. **Momsbelopp (SEK)** räknas fram med hjälp av Bokförtbelopp och Moms %.

Ifall man ändrar bokföringsår så hämtas kontoplanen för angivet bokföringsår och presenteras i listan.

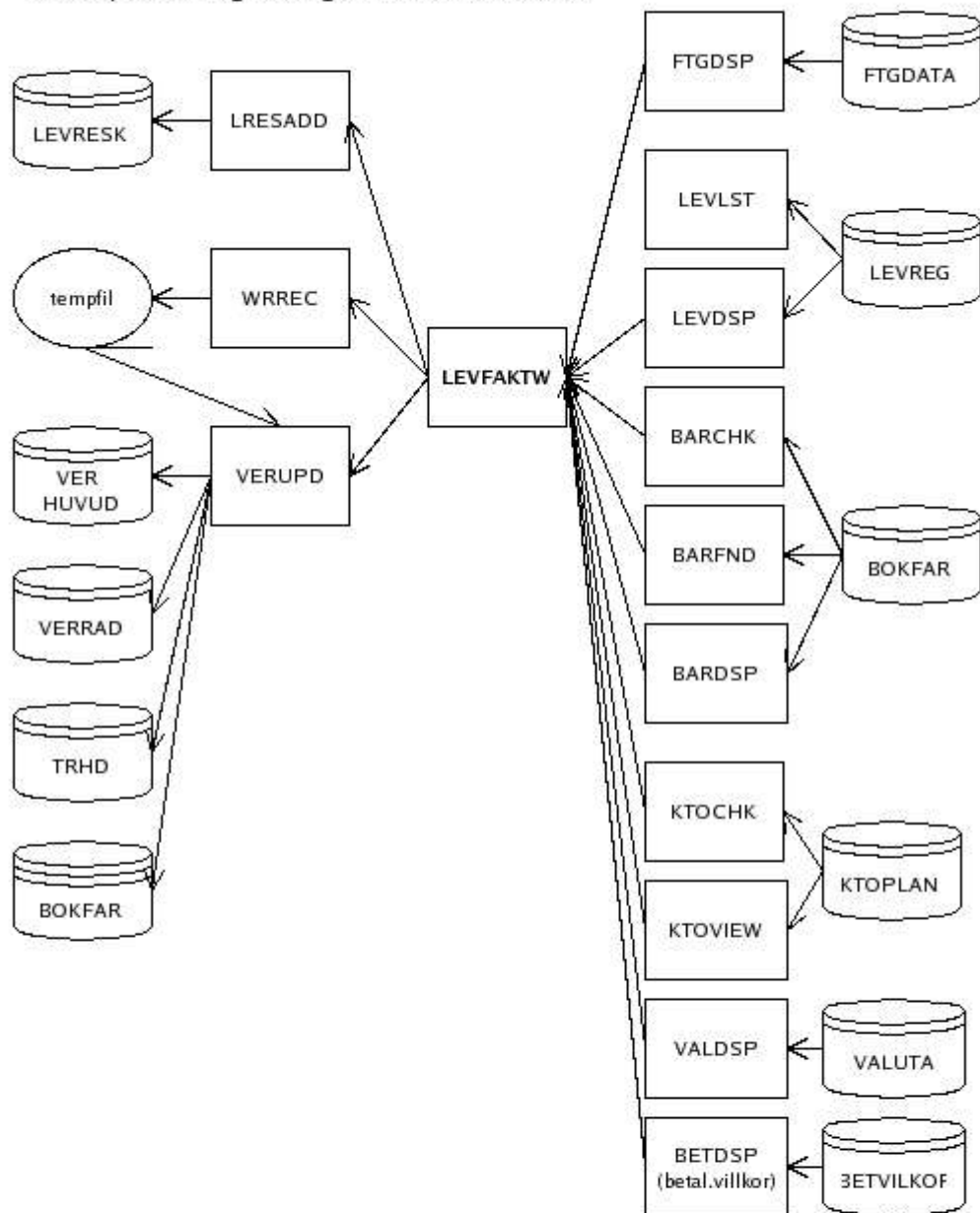
När markören (cursorn) står i ettdera av fälten för kontonummer, kan kontonummer väljas genom att klicka på önskat kontonummer i listan.

När man klickar på **OK** uppdateras leverantörsreskontran och bokföringen (huvudboken).

**Avsluta** avbryter pågående arbete utan uppdatering.



# Flödesplan för registrering av leverantörsfaktura



**LEVFAKTW - Registrera leverantörsfaktura.**

Leverantörsnummer:  Leverantörens organisationsnummer:

Leverantörsnamn:

Adress:  Postnr:

Postadress:  Land:

---

Kundnummer:  Registreringsdatum:

Fakturanummer:

Faktura datum:

Betalningsvillkor:  dagar netto

Förfallodatum:  OCR nummer:

Fakturatext:

---

Bokföringsår:  Verifikationsnr:  Momskod:

Valuta:  Valutakurs:  Moms %:

Fakturabelopp:  i fakturans valuta.

---

Kontonr: (Kredit)  Bokfört belopp (SEK):

Momskontonr:  Momsbelopp (SEK):

Kontonr: (Debet)  Debetbelopp (SEK):

Levnummer	Levnamn
123	Leverantör AB
124	Distributör AB
125	Försäljning AB
126	Dataspecialisten AB

Kontonr	Kontonamn
1010	Kassa
1011	Kassa 2
1012	Kassa 3
1210	Maskiner
1219	Ack avskrivningar maskiner
1220	Inventarier
1229	Ack avskrivningar inventarier

OK Avsluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda programmet LEVFAKTW.

LEVFAKTW anropar LRESADD, LEVDSP, LEVLST, FTGDSP, BARCHK, BARFND, KTOCHK, KTOVIEW, WRREC, VALUTA och VERUPD via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");             // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("LRESADD");               // OLFIX funktion
process->addArgument(levnr);                   // primary key
process->addArgument(fakturanr);               // primary key
process->addArgument(regdatum);
process->addArgument(faktdatum);
process->addArgument(expiredatum);             // förfallodatum
process->addArgument(fakttext);
process->addArgument(bar);                     // bokföringsår
process->addArgument(momsprocent);
process->addArgument(levkontonr);
process->addArgument(faktbelopp);
process->addArgument(momsktonr);
process->addArgument(momsbelopp);
process->addArgument(kreditkto);
process->addArgument(kreditbelopp);
process->addArgument(usr);                     // userid
```

Detta blir:

```
./STYRMAN usr LRESADD levnr fakturanr regdatum expiredatum fakttext bar momsprocent
levkontonr faktbelopp momsktonr momsbelopp debetkto debetbelopp usr
```

usr är den inloggades userid.

För övriga funktioner se respektive funktion.

När det gäller belopp ska det alltid ske en utfyllnad med mellanslag före första siffran så att det blir totalt 12 tecken.

### Behörighetskrav:

För att kunna köra LEVFAKTW behövs behörighet till

PRGLST  
LRESADD  
LEVDSP  
LEVLST  
FTGDSP  
VALUTA  
BARCHK  
BARFND  
KTOCHK  
KTOVIEW

WRREC  
VERUPD

## LSTFNCW....Lista funktioner

LSTFNCW, ett grafiskt program för att lista befintliga funktioner..

För att använda programmet krävs behörighet till funktionerna TRNSLST.

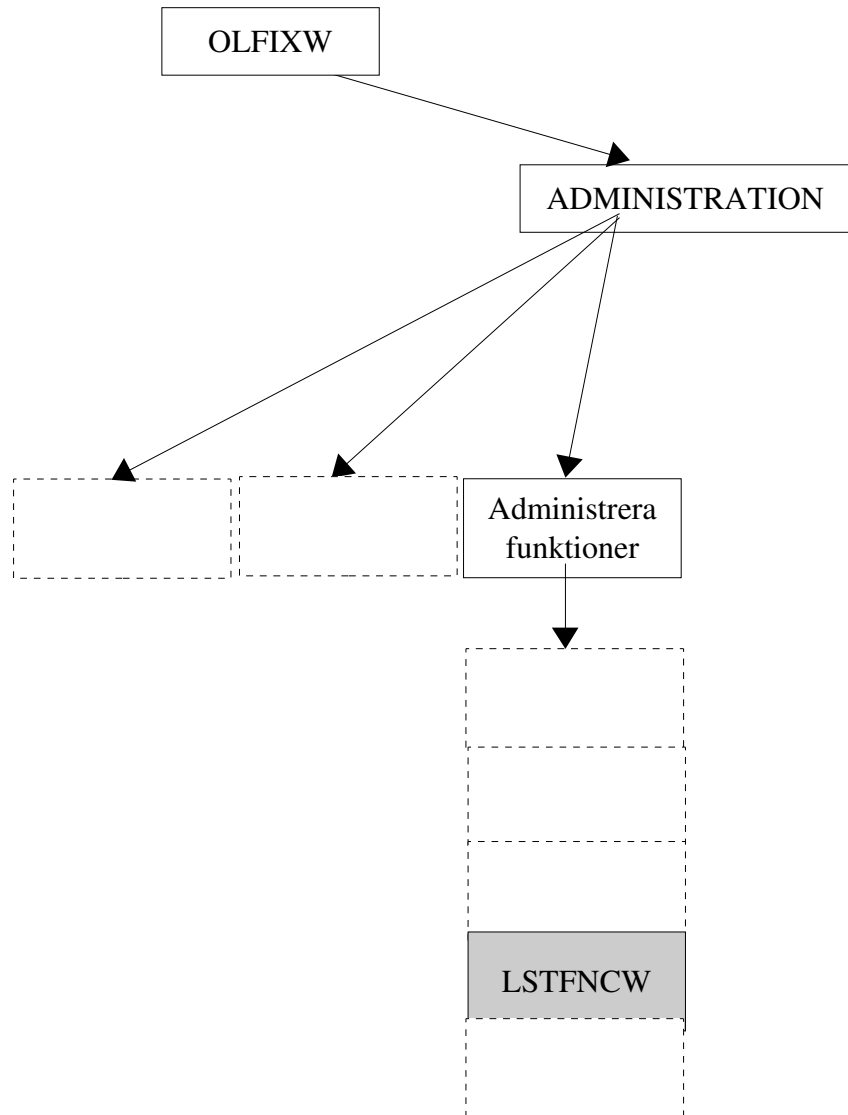
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Popdown)

(Popdown)





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet använder funktionen TRNSLST för att hämta data från databasen. Anropet av TRNSLST sker via STYRMAN.

**Behörighetskrav:**

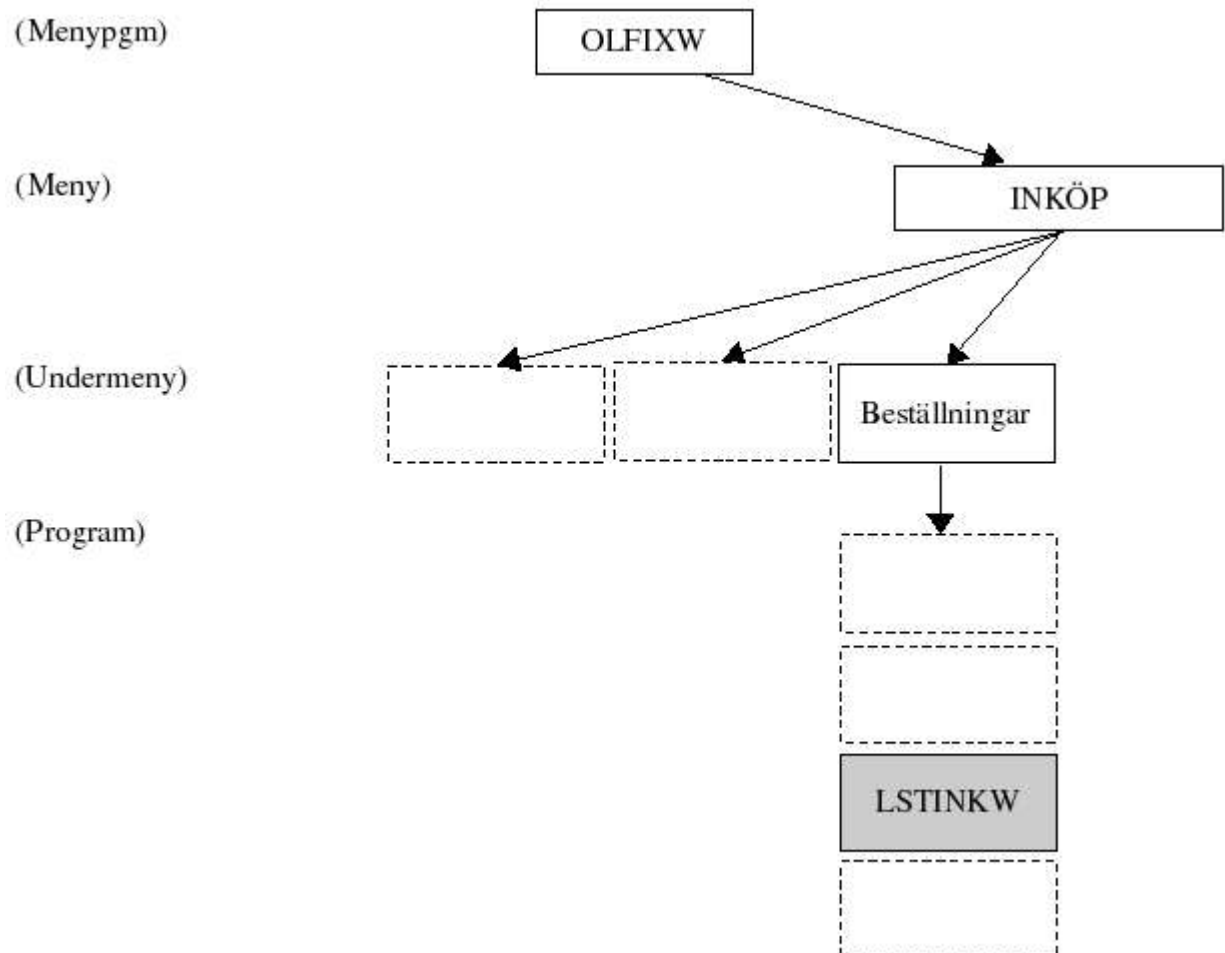
För att kunna köra LSTFNCW behövs behörighet till

PRGLST

TRNSLST

## LSTINKW.....Beställningsstock

LSTINKW, ett grafiskt program för att lista samtliga inköpsorderrader som ej är fullt levererade. Programmet plockar upp userid från environment.





LSTINKW - Beställningsstock											
Ordernr	Radnr	Leverantör	Artikelnr	Benämning	Lev.vecka	B-kod	Beställt Antal	Lev.Antal	Resterande ant.	Pris	Summa
18	10	9999	1173-0911	Spänningsregulator positiv	402	E	10.00	0.00	10.00	1.50	15.00
18	30	9999	1173-1175	Spänningsregulator positiv	402	E	10.00	0.00	10.00	30.00	300.00
18	40	9999	1173-1445	D/A Omvandlare 12-bit	402	E	10.00	0.00	10.00	95.00	950.00
18	50	9999	1173-1447	Timerkrets	402	E	10.00	0.00	10.00	5.45	54.50
18	60	9999	1173-6910	Quadruple 2input NAND gate	402	E	10.00	0.00	10.00	1.45	14.50
19	10	9999	1173-7206	Dual Monostable Multivibrator	402	E	20.00	0.00	20.00	4.75	95.00
19	20	9999	1173-7300	1024x1 bit statiskt RAM	401	E	20.00	10.00	10.00	10.05	100.50
19	30	9999	1173-7300	1024x1 bit statiskt RAM	402	E	20.00	0.00	20.00	10.05	201.00
19	40	9999	1173-7513	Microprocessor	402	E	20.00	5.00	15.00	63.00	945.00
20	10	1000	4008496326389	Batteri LR03/AAA	403	E	100.00	25.00	75.00	34.90	2617.50
20	20	1000	4008496326426	Batteri LR6/AA	403	E	100.00	0.00	100.00	31.50	3150.00
20	30	1000	7310759090508	Lampa Novaline 40W	404	E	200.00	100.00	100.00	10.90	1090.00
21	10	2001	1173-1445	D/A Omvandlare 12-bit	4024	E	10.00	0.00	10.00	95.00	950.00
21	20	2001	Test1	Testartikel	4053	E	10.00	0.00	10.00	125.50	1255.00
21	30	2001	1173-1447	Timerkrets	4053	E	10.00	0.00	10.00	5.45	54.50
6712	10	9999	1173-1445	D/A Omvandlare 12-bit	351	E	25.00	0.00	25.00	95.00	2375.00
Total										14167.50	

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen INKLST.

LSTINKW anropar INKLST via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("INKLST");                // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr INKLST
```

usr är den inloggades userid.

### **Behörighetskrav:**

För att kunna köra LSTINKW behövs behörighet till  
PRGLST  
INKLST

## LSTKSTW.....Lista kostnadsställen

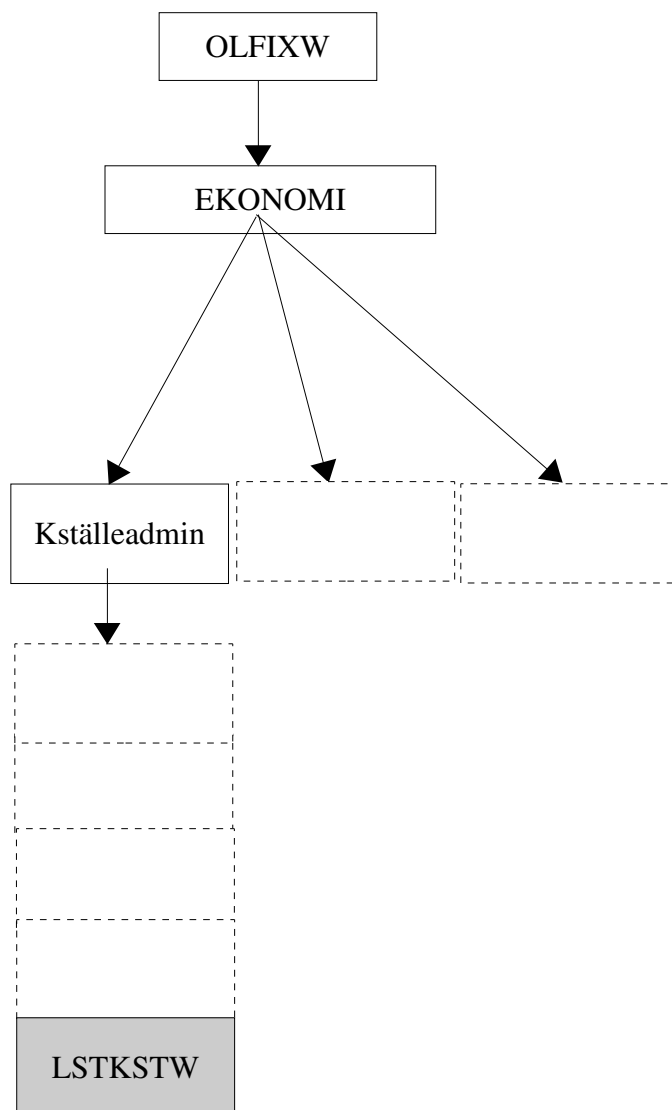
LSTKSTW, ett grafiskt program för att visa information om alla kostnadställen på skärm.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



OLFIX - LSTKSTW. Lista kostnadsställen

Bokf.år	Kställe	Benämning
AD	9037	Projekt Titan
AD	9038	Projekt OMEGA
AD	9040	Projekt OLFIX
AD	9041	Projekt Test
AD	9042	Projekt Prov nr 1
AD	9043	Projekt Prov nr 2

Uppdatera

Sluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen KSTLST.

LSTKSTW anropar KSTLST via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("LSTDSP");                // OLFIX funktion
process->addArgument(bar);
process->addArgument(kstalle);
```

Detta blir:

```
./STYRMAN usr KSTLST bar kstalle
```

usr är den inloggades userid.

#### **Behörighetskrav:**

För att kunna köra LSTKSTW behövs behörighet till  
PRGLST  
KSTLST

## LSTKTOW.....Lista konton

LSTKTOW, ett grafiskt program för att lista konton. Man måste ange bokföringsår (arid).  
Programmet plockar upp userid från environment.

(Menypgm)

OLFIXW

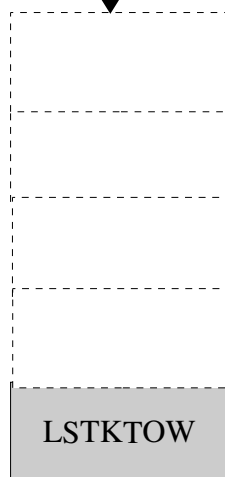
(Meny)

EKONOMI

(Undermeny)

Kontoadmin

(Program)





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda KTOVIEW.

LSTKTOW anropar KTOVIEW via STYRMAN med parametrarn user-id och arid

```
const char *userp = getenv("USER"); // Hämtar den inloggades user-id
QString usr(userp);

QString bibl;
bibl.append("./STYRMAN"); // OLFIX huvudprogram

process = new QProcess();
process->addArgument(bibl); // ./STYRMAN
process->addArgument(usr); // användarens user-id
process->addArgument("KTOVIEW"); // OLFIX funktion
process->addArgument(arid); // bokföringsår
```

Detta blir:

```
./STYRMAN usr KTOVIEW arid
```

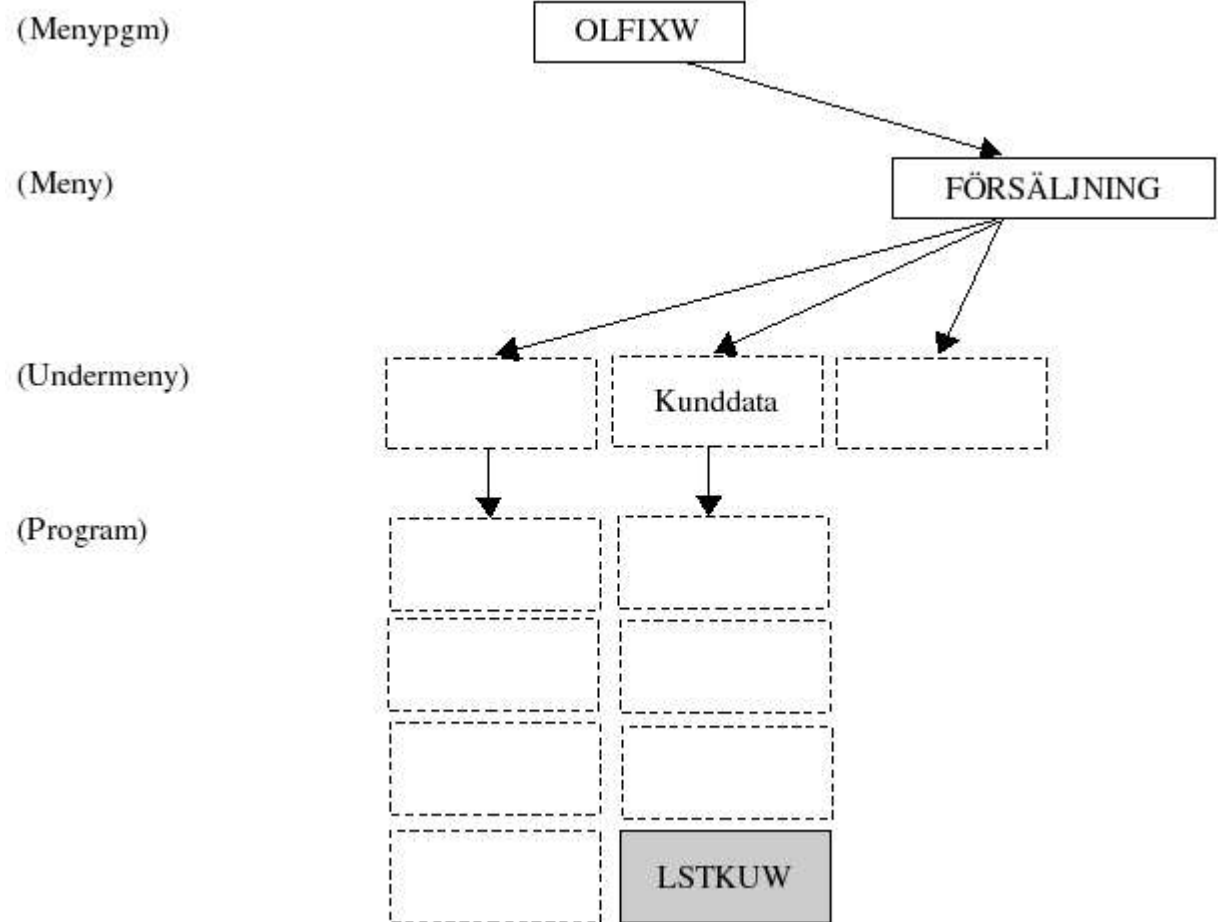
### **Behörighetskrav:**

För att kunna köra LSTKTOW behövs behörighet till  
PRGLST  
KTOVIEW



## LSTKUW.....Lista kunder

LSTKUW, ett grafiskt program för att lista kunder på skärmen, kundnr och namn.





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda LSTKUW.

LSTKUW anropar KULST via STYRMAN utan parametrar.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);
```

```
process = new QProcess();
process->addArgument("STYRMAN");
process->addArgument(usr); // användarens userid
process->addArgument("KULST"); // OLFIX funktion
```

Detta blir:

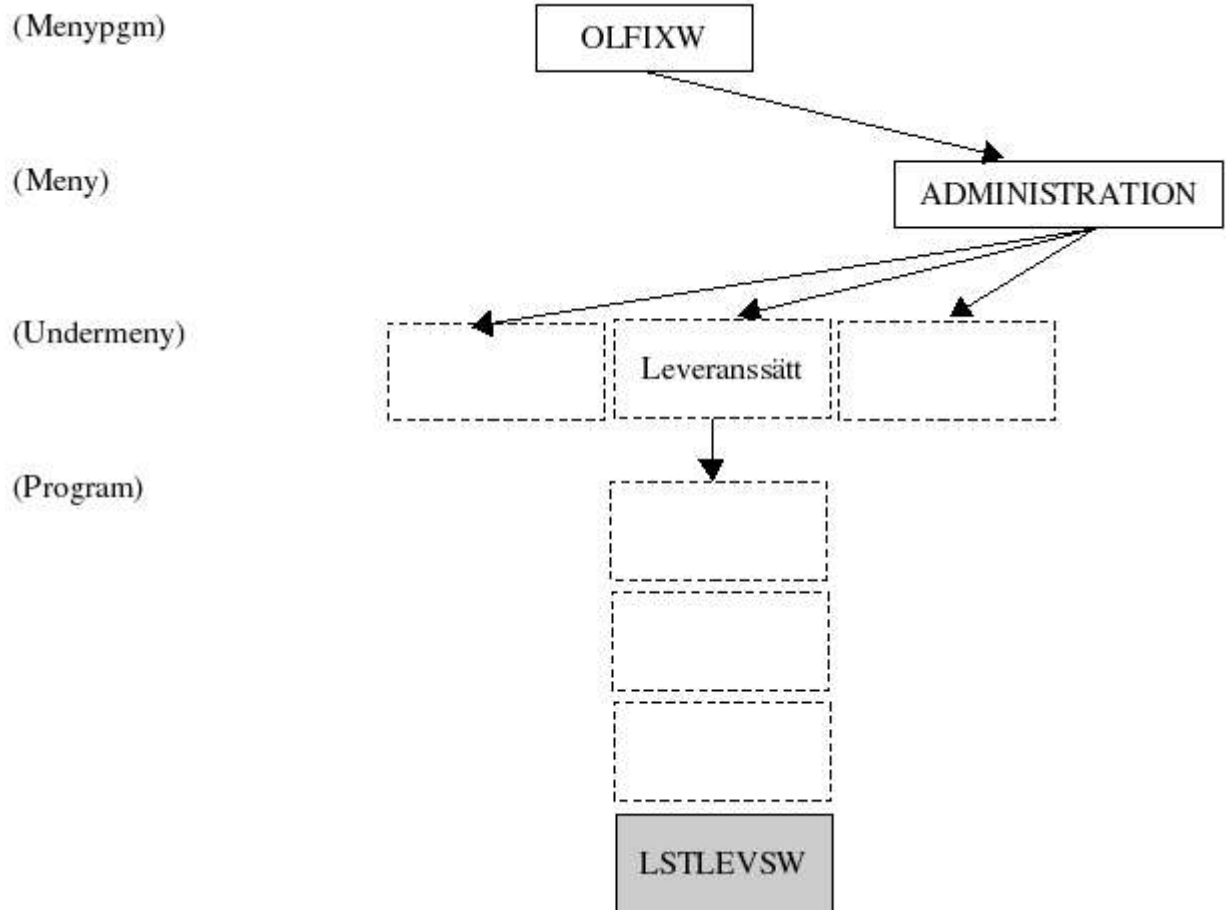
```
./STYRMAN usr KULST
```

#### **Behörighetskrav:**

För att kunna köra LSTKUW behövs behörighet till  
PRGLST  
KULST

## LSTLEVSW.....Lista leveranssätt

LSTLEVSW, ett grafiskt program för att lista **leveranssätt** på skärmen.





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda LSTLEVSW.

LSTLEVSW anropar LEVSLST via STYRMAN utan parametrar.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);
```

```
process = new QProcess();
process->addArgument("STYRMAN"); // OLFIX huvudprogram
process->addArgument(usr); // användarens userid
process->addArgument("LEVSLST"); // OLFIX funktion
```

Detta blir:

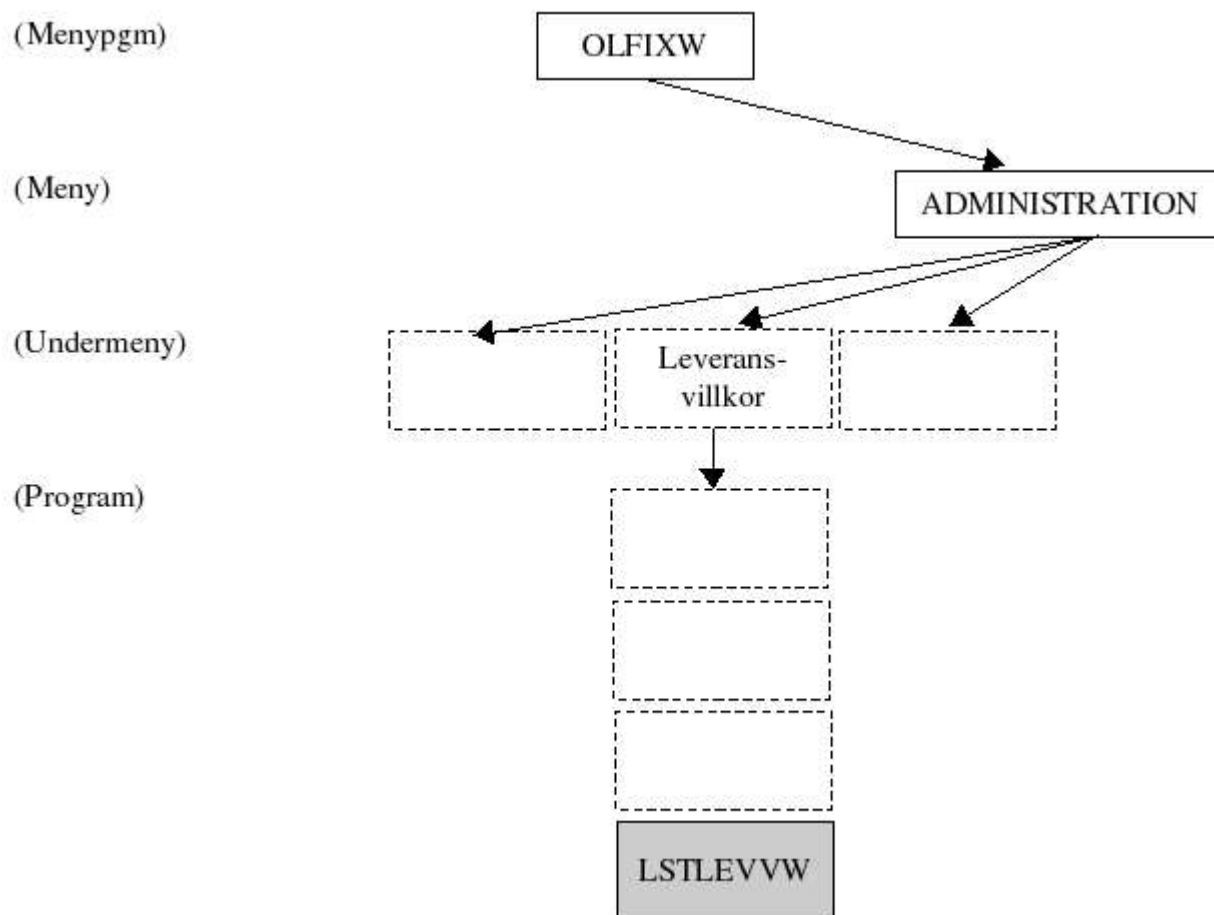
```
./STYRMAN usr LEVSLST
```

#### **Behörighetskrav:**

För att kunna köra LSTLEVSW behövs behörighet till  
PRGLST  
LEVSLST

## LSTLEV VW.....Lista leveransvillkor

LSTLEVS W, ett grafiskt program för att lista **leveransvillkor** på skärmen.







För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda LSTLEVW.

LSTLEVW anropas LEVVLST via STYRMAN utan parametrar.

```
const char *userp = getenv("USER"); // Hämtar den inloggades userid
QString usr(userp);
```

```
process = new QProcess();
process->addArgument("STYRMAN"); // OLFIX huvudprogram
process->addArgument(usr);       // användarens userid
process->addArgument("LEVVLST"); // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr LEVSLST
```

#### **Behörighetskrav:**

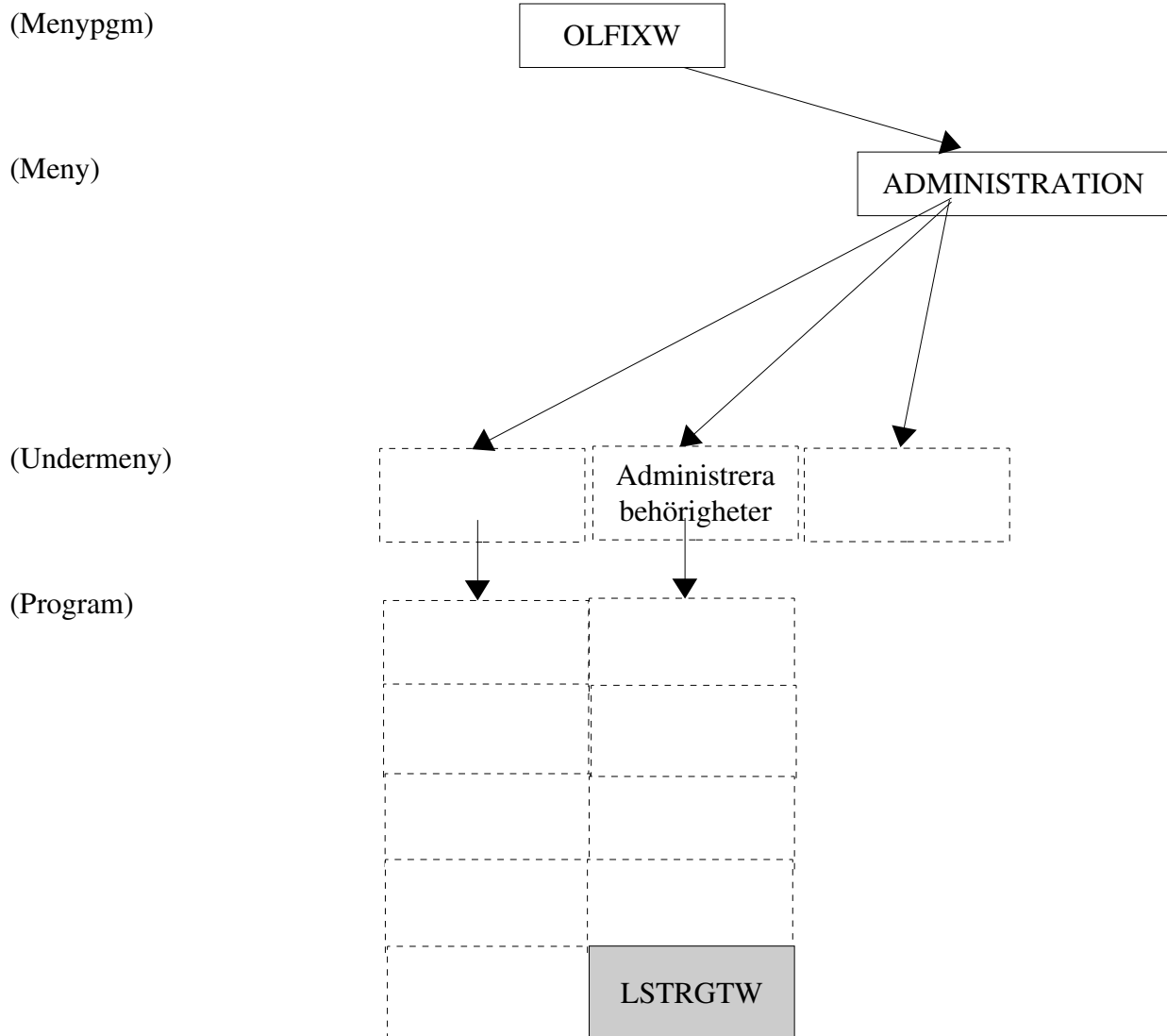
För att kunna köra LSTLEVW behövs behörighet till

PRGLST

LEVVLST

## LSTRGTW.....Lista behörigheter

LSTRGTW, ett grafiskt program för att lista behörigheter på skärmen.





För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet använder funktionen RGTLST för att hämta data från databasen. Anropet av RGTLST sker via STYRMAN.

**Behörighetskrav:**

För att kunna köra LSTRGTW behövs behörighet till

PRGLST

RGTLST

## LSTUSRW.....Lista alla användare

LSTUSRW, ett grafiskt program för att lista användare på skärmen.

(Menypgm)

OLFIXW

(Meny)

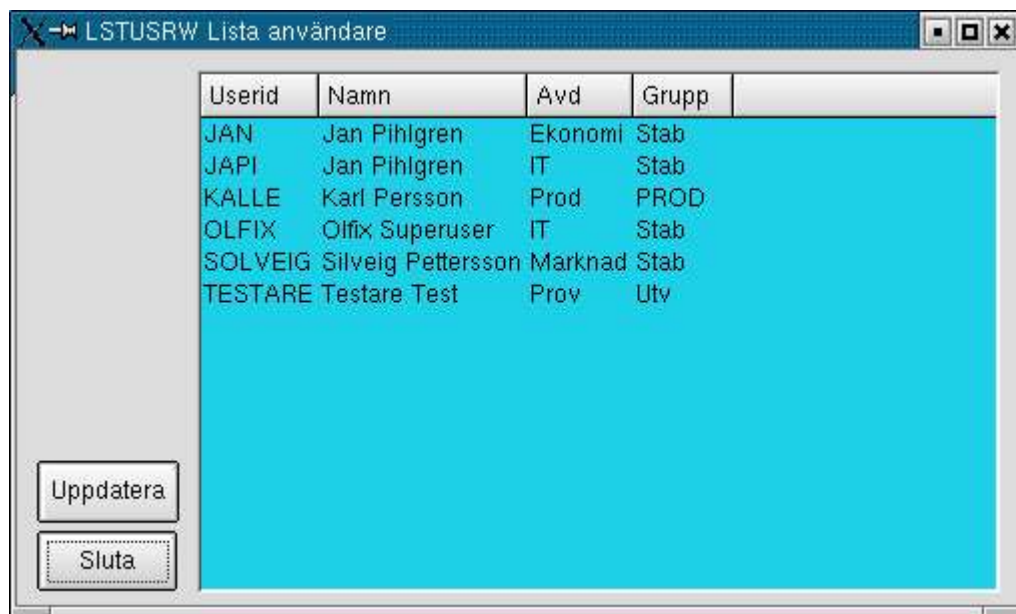
ADMINISTRATION

(Undermeny)

Administrera  
användare

(Program)

LSTUSRW



För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet använder funktionen USERLST för att hämta data från databasen. Anropet av USERLST sker via STYRMAN.

**Behörighetskrav:**

För att kunna köra LSTUSRW behövs behörighet till

PRGLST

USERLST

## LSTVALW.....Lista alla valutor

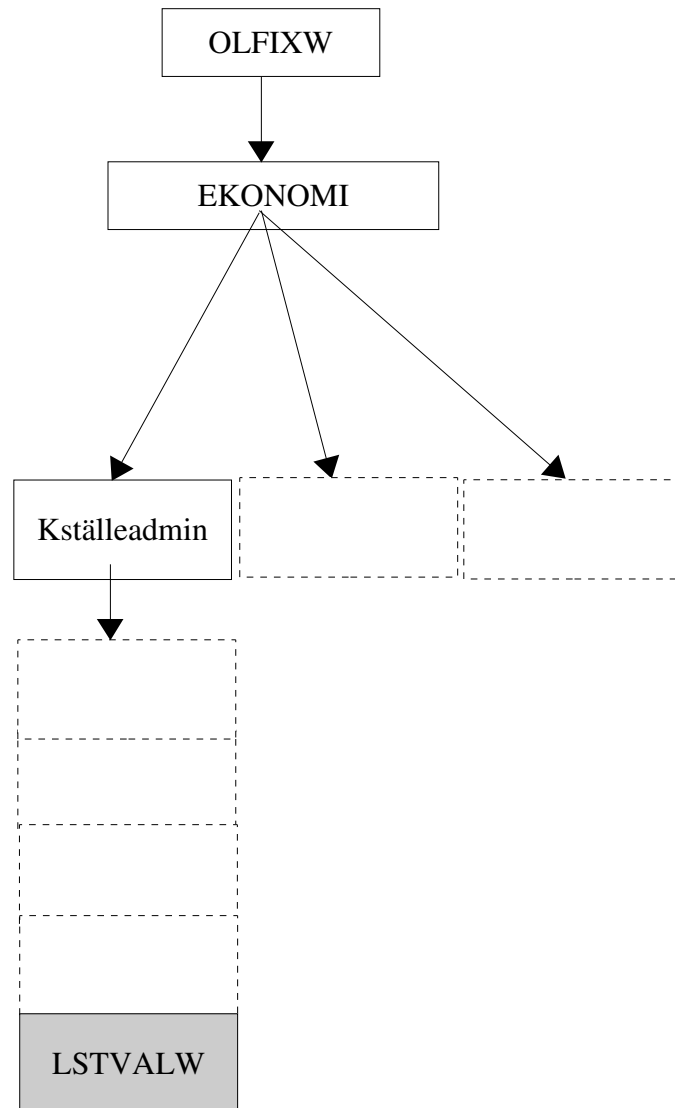
LSTVALW, ett grafiskt program för att visa information om alla valutor på skärm.  
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)





OLFIX - LSTVALW. Lista valutor

Valuta	Land	Sälj	Köp	Beteckning	
AUD	Australien	5.03	5.03	Dollar	
CAD	Kanada	5.66	5.66	Dollar	
CHF	Schweiz	0.00	0.00	France	
DKK	Danmark	1.22	1.22	Kronor	
EEK	Estland	0.59	0.57	Kronor	
EUR	Europa	9.08	9.08	Euro	
GBP	Storbritanien	14.26	14.26	Pund	
HKD	Honkong	0.00	0.00	Dollar	
JPY	Japan	7.38	7.38	Yen	
MYR	Malaysia	2.36	2.36	Ringgit	
NOK	Norge	1.23	1.23	Kronor	
NYZ	Nya Zeeland	4.45	4.45	Dollar	
SAR	Saudiarabien	2.40	2.40	Riyal	
SEK	Sverige	1.00	1.00	Kronor	
SGD	Singapore	5.08	5.08	Dollar	

Uppdatera

Sluta

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen VALLST.

LSTVALW anropar VALLST via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("LSTVAL");                // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr VALLST
```

usr är den inloggades userid.

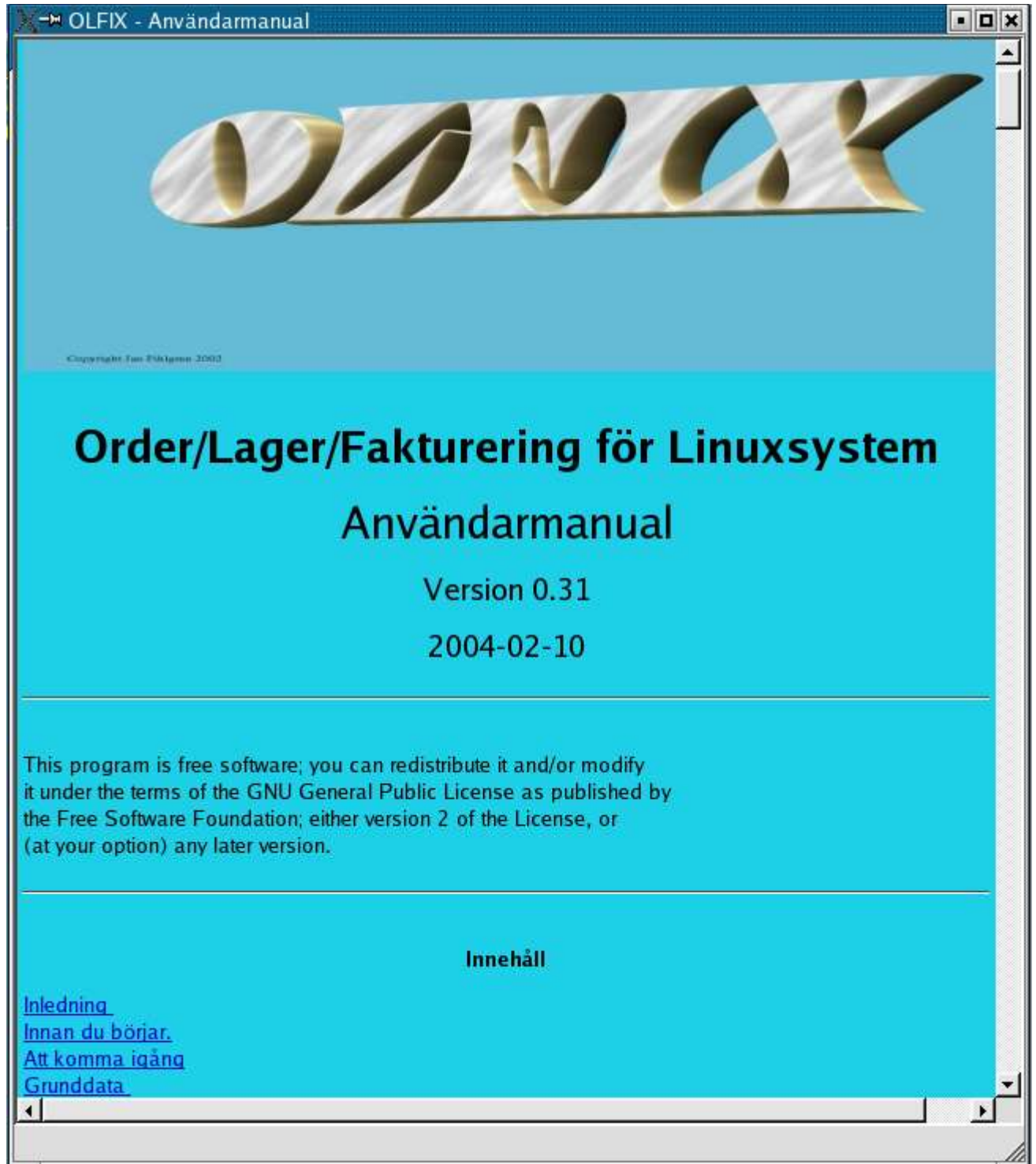
### **Behörighetskrav:**

För att kunna köra LSTVALW behövs behörighet till  
PRGLST  
VALLST

## OLFIXHLP.....Online hjälp

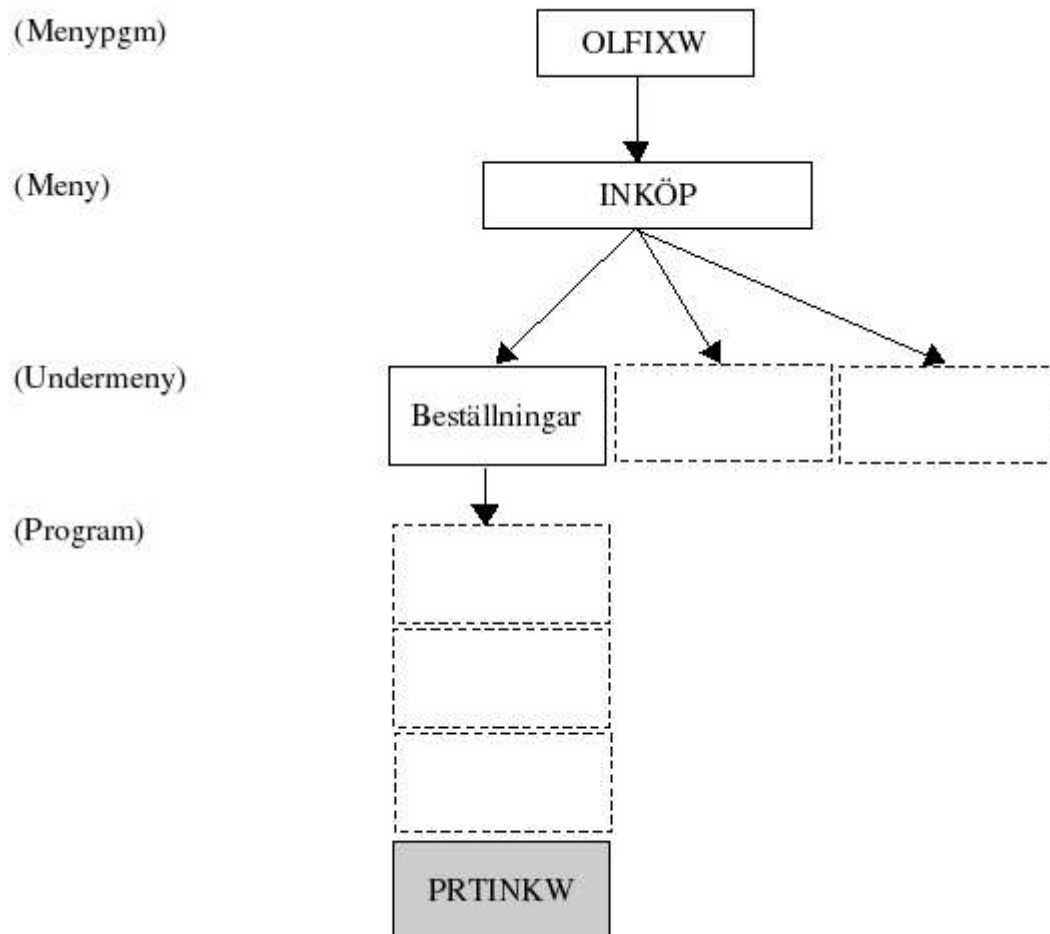
OLFIXHLP, ett grafiskt program för att med hjälp online.  
Programmet anropas från OLFIXW.

Texterna är en html-fil, **UserManual.html** som ligger i biblioteket /usr/local/olfix/doc/helpfiles/usermanual.




## PRTINKW.....Skriv ut beställning

PRTINKW, ett grafiskt program för att med hjälp av programmet Kugar skriva ut beställningar. Programmet plockar upp userid från environment.










Exempel på en rapport presenterad i Kugar.


**Kugar**

Arkiv Kör Inställningar Hjälp

PROGRAM AB

**Beställning**

Datum 2004-02-05 Beställningsnr 26 Sida 1

Leverantörsnr 123

Leveransadress  
PROGRAM AB  
Verktygsgatan 11  
199 97 PROGSTAD

Leverantör AB  
Postgatan 33  
199 99 DATABY  
SVERIGE

Kommentar  
Detta är en kommentar

Telefon: 08-59112449  
Fax: 09-112233

Valuta  
SEK

Er referens  
Per Josefsson

Godsmärke  
KALLE PALL 26

Betalningsvillkor  
20 dagar netto

Leveransvillkor  
EXW

Leveranssätt  
Schenker kundnr:11105232

Vår referens  
Jan Pihlgren

Vårt artikelnr	Ert artikelnr				
Pos	Benämning	Benämning	Antal	Sort	A-pris Leverans
10	1173-1175 Spänningsregulator positiv		100.00		30.00 4074
20	1173-1445 D/A Omvandlare 12-bit		100.00		95.00 4074

Ordererkonfimerande skickas inom 3 arbetsdagar (om ej redan bekräftats)

Ange alltid vårt artikelnummer på flöjesedel och faktura.

För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionerna INKHDSP (hämta inköpsorderhuvud) och INKRLST (inköpsorderrader).

PRTINKW anropar INKHDSP och INKRLST via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("INKHDSP");               // OLFIX funktion
process->addArgument(inkordernr);              // inköpsordernummer
```

och

```
process = new QProcess();
process->addArgument("./STYRMAN");             // OLFIX styrprogram
process->addArgument(usr);                     // userid
process->addArgument("INKRLST");               // OLFIX funktion
process->addArgument(inkordernr);
```

Detta blir:

```
./STYRMAN usr INKHDSP inkordernr
```

och

```
./STYRMAN usr INKRLST inkordernr
```

usr är den inloggades userid.

PRTINKW skapar filen Bestelling.kud i XML-format.

PRTINKW anropar sedan Kugar.

```
if (kugarversion<"1.2.92"){
    kommando="kugar -d /tmp/Bestelling.kud -r "+reportpath+"Bestelling.kut";
    system(kommando);
}else{
    system("kugar /tmp/Bestelling.kud");
}
```

**Behörighetskrav:**

För att kunna köra PRTINKW behövs behörighet till

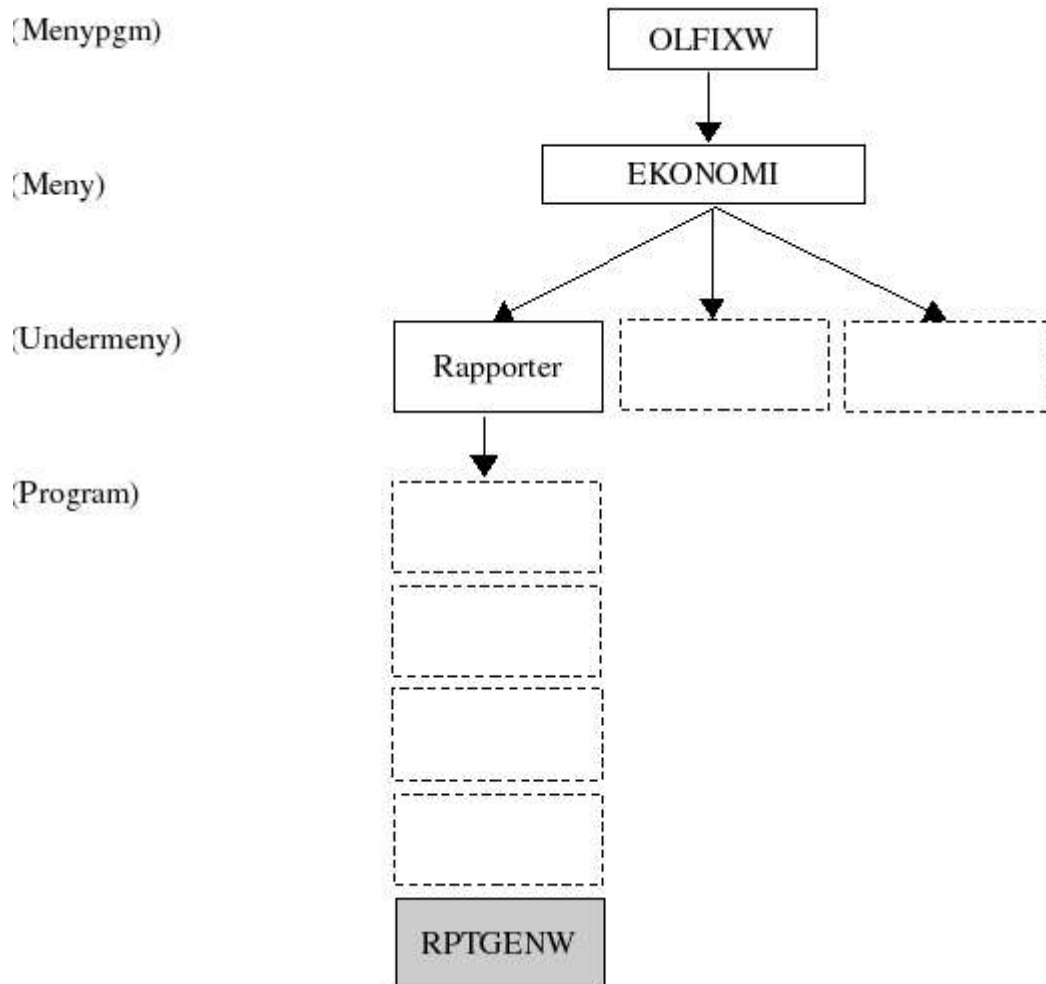
PRGLST

INKHDSP

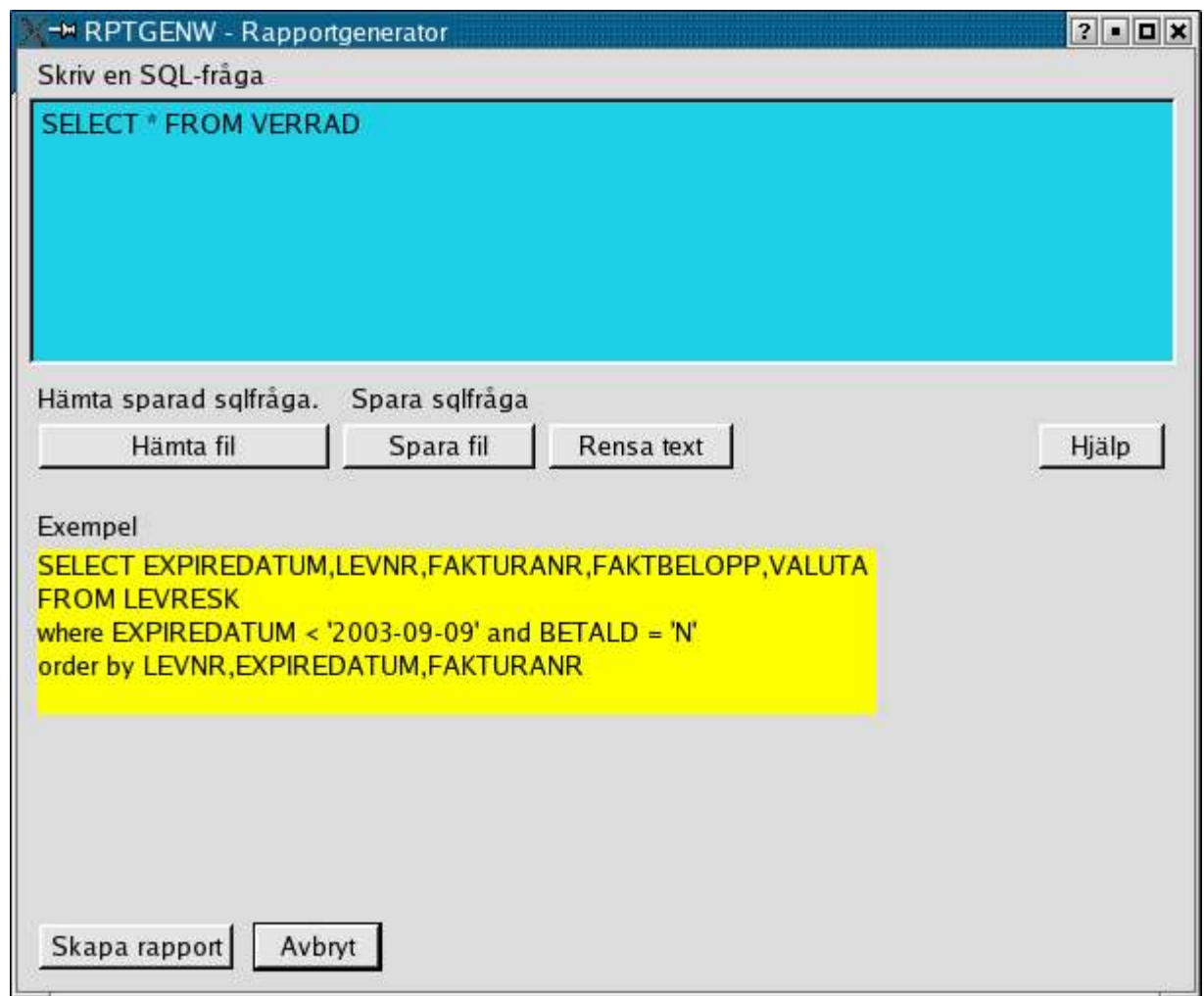
INKRLST

## RPTGENW.....Rapportgenerator

RPTGENW, ett grafiskt program för att skapa valfria rapporter. Rapporterna plockas sedan upp av Kspread. Programmet plockar upp userid från environment.







För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen RPTCRE.

RPTGENW anropar RPTCRE via STYRMAN.

```
const char *userp = getenv("USER");           // Hämtar den inloggades userid
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr);                     // user OLFIX
process->addArgument("RPTCRE");                // OLFIX funktion
process->addArgument(sqlquery);                 // SQLfråga
```

Detta blir:

```
./STYRMAN usr RPTCRE sqlquery
```

usr är den inloggades userid.

RPTCRE skapar filen /tmp/rptcre.txt.

RPTGENW anropar sedan Kspread.

```
system("kspread /tmp/rptcre.txt");
```

### **Behörighetskrav:**

För att kunna köra RPTGENW behövs behörighet till

PRGLST

RPTCRE

## SDOLISW.....Saldolista

SDOLISW, ett grafiskt program för att skapa underlag till en saldolista. Saldolistan kan sedan plockas upp av antingen Kugar eller Kspread.

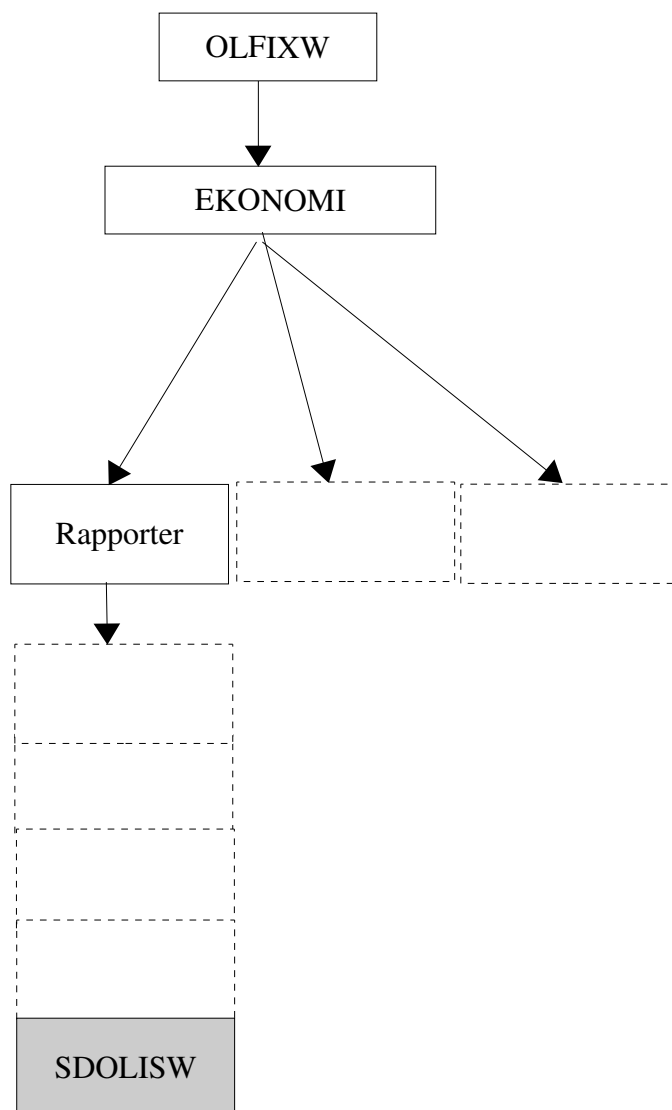
Programmet plockar upp userid från environment.

(Menypgm)

(Meny)

(Undermeny)

(Program)



**SDOLISW - Saldolista**

Datum: 2004-03-25

Bokföringsår: **AD** Fr. o. m. datum: **2003-07-30** T.o.m. datum: **2003-09-11**

☒ CSV-format: Startar Kspread med importerade data.  
☒ Utskriftsformat: Skapar en rapport med Kugar färdig att skriva ut.

file:/home/jan/Utveckling/OLFIX/doc/SaldolistaKspreadExempel.ksp - Kspread

Arkiv Redigera Visa Infoga Format Data Verktyg Inställningar Hjälp

Σ f(x) SUM

Sans 10 A B I U

B15

	A	B	C	D	E
1	1220	Inventarier	18 750,00	0,00	18 750,00
2	2081	Aktiekapital	0,00	300 000,00	-300 000,00
3	2330	Checkräkningskredit	799 450,00	60 000,00	739 450,00
4	2350	Banklån	0,00	500 000,00	-500 000,00
5	2440	Leverantörsskulder	0,00	149 820,00	-149 820,00
6	2641	Ingående moms	49 455,00	0,00	49 455,00
7	4010	Materialkostnad	86 865,00	0,00	86 865,00
8	5010	Lokalhyra	48 000,00	0,00	48 000,00
9	6110	Kontorsmateriel	6 750,00	0,00	6 750,00
10	8490	Övriga finansiella kostnader	550,00	0,00	550,00
11					

Arbetsblad 1

Summa: 0

# PROGRAM AB

2004-03-23

Saldolista				
		För perioden 2003-07-30 -- 2003-09-11		
Konto	Kontonamn	Debet	Kredit	Utg Saldo
1220	Inventarier	18750.00	0.00	18750.00
<b>KLASSTOTAL</b>		<b>18750.00</b>	<b>0.00</b>	<b>18750.00</b>
2081	Aktiekapital	0.00	300000.00	-300000.00
2330	Checkräkningskredit	799450.00	60000.00	739450.00
2350	Banklån	0.00	500000.00	-500000.00
2440	Leverantörsskulder	0.00	190680.00	-190680.00
2611	Utgående moms	10215.00	0.00	10215.00
2641	Ingående moms	49455.00	0.00	49455.00
<b>KLASSTOTAL</b>		<b>859120.00</b>	<b>1050680.00</b>	<b>-191560.00</b>
4010	Materialkostnad	117510.00	0.00	117510.00
<b>KLASSTOTAL</b>		<b>117510.00</b>	<b>0.00</b>	<b>117510.00</b>
5010	Lokalhyra	48000.00	0.00	48000.00
<b>KLASSTOTAL</b>		<b>48000.00</b>	<b>0.00</b>	<b>48000.00</b>

Sida: 1

Konto	Kontonamn	Debet	Kredit	Utg Saldo
6110	Kontorsmateriel	6750.00	0.00	6750.00
<b>KLASSTOTAL</b>		<b>6750.00</b>	<b>0.00</b>	<b>6750.00</b>
8490	Övriga finansiella kostnader	550.00	0.00	550.00
<b>KLASSTOTAL</b>		<b>550.00</b>	<b>0.00</b>	<b>550.00</b>
<b>SALDOTOTALER</b>		<b>1050680.00</b>	<b>1050680.00</b>	<b>0.00</b>

## Funktionsbeskrivning.

Programmet hämtar data från VERHUVUD, VERRAD och KTOPLAN.

SQLsats som används är

```
SELECT VERRAD.KTONR, KTOPLAN.BENAMNING, VERRAD.DK, VERRAD.BELOPP FROM VERRAD
LEFT JOIN KTOPLAN ON KTOPLAN.KTONR = VERRAD.KTONR AND VERRAD.ARID = KTOPLAN.ARID
WHERE VERRAD.ARID = "AC"
ORDER BY KTONR
```

samt

```
SELECT min(VERDATUM) mindate, max(VERDATUM) maxdate FROM VERHUVUD WHERE ARID = "AC"
```

Från och med datum som anges i **“För perioden”** avser datum när den första verifikationen registrerades och till och med datum avser registrerings datum för den senaste verifikationen, allt för angivet bokföringsår. Datum överst på sidan avser utskriftsdatum.

Man kan välja att att få rapporten till Kspread, ett kalkylprogram, eller Kugar som är en rapportgenerator. Mallen, template, till Saldolistan ligger i biblioteket */usr/local/olfix/report* . Mallen är fast eftersom data ut från SDOLISW är skapat för just den mallen.

Utdata till Kugar är i XML-format, ex:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE KugarData [
  <!ELEMENT KugarData (Rowhead* )>
  <!ATTLIST KugarData
    Template CDATA #REQUIRED>

  <!ELEMENT Rowhead EMPTY>
  <!ATTLIST Rowhead
    level CDATA #REQUIRED
    ftgnamn CDATA #REQUIRED
    startdatum CDATA #REQUIRED
    datum CDATA #REQUIRED
>
  <!ELEMENT KugarData (Row* )>
  <!ATTLIST KugarData
    Template CDATA #REQUIRED>

  <!ELEMENT Row EMPTY>
  <!ATTLIST Row
    level CDATA #REQUIRED
    kontonr CDATA #REQUIRED
    kontonamn CDATA #REQUIRED
    debet CDATA #REQUIRED
    kredit CDATA #REQUIRED
    utgsaldo CDATA #REQUIRED
>
  <!ELEMENT KugarData (Delsumma* )>
  <!ATTLIST KugarData
    Template CDATA #REQUIRED>

  <!ELEMENT Delsumma EMPTY>
  <!ATTLIST Delsumma
    level CDATA #REQUIRED
    delsumdebit CDATA #REQUIRED
    delsumkredit CDATA #REQUIRED
    delsumutgsaldo CDATA #REQUIRED
>
  <!ELEMENT KugarData (Blankrad* )>
  <!ATTLIST KugarData
    Template CDATA #REQUIRED>

  <!ELEMENT Blankrad EMPTY>
  <!ATTLIST Blankrad
    level CDATA #REQUIRED
    blank CDATA #REQUIRED
>
  <!ELEMENT KugarData (Totalsumma* )>
  <!ATTLIST KugarData
```

```

Template CDATA #REQUIRED>

<!ELEMENT Totalsumma EMPTY>
<!--ATTLIST Totalsumma
    level CDATA #REQUIRED
    totaldebit CDATA #REQUIRED
    totalkredit CDATA #REQUIRED
    totalutgsaldo CDATA #REQUIRED
-->
]>

<KugarData Template="/home/jan/Utveckling/OLFIX/report/Saldolista.kut">
<Rowhead level="0" ftgnamn="PROGRAM AB" startdatum="2003-07-30 -- 2003-09-11" datum="2004-03-23"/>
<Row level="1" kontonr="1220" kontonamn="Inventarier" debet="18750.00" kredit="0.00" utgsaldo="18750.00"/>
<Delsumma level="2" delsumdebit="18750.00" delsumkredit="0.00" delsumutgsaldo="18750.00"/>
<Blankrad level="3" blank=" "/>
<Row level="1" kontonr="2081" kontonamn="Aktiekapital" debet="0.00" kredit="300000.00"
utgsaldo="-300000.00"/>
<Row level="1" kontonr="2330" kontonamn="Checkräkningskredit" debet="799450.00" kredit="60000.00"
utgsaldo="739450.00"/>
<Row level="1" kontonr="2350" kontonamn="Banklån" debet="0.00" kredit="500000.00" utgsaldo="-500000.00"/>
<Row level="1" kontonr="2440" kontonamn="Leverantörsskulder" debet="0.00" kredit="190680.00"
utgsaldo="-190680.00"/>
<Row level="1" kontonr="2611" kontonamn="Utgående moms" debet="10215.00" kredit="0.00" utgsaldo="10215.00"/>
<Row level="1" kontonr="2641" kontonamn="Ingående moms" debet="49455.00" kredit="0.00" utgsaldo="49455.00"/>
<Delsumma level="2" delsumdebit="859120.00" delsumkredit="1050680.00" delsumutgsaldo="-191560.00"/>
<Blankrad level="3" blank=" "/>
<Row level="1" kontonr="4010" kontonamn="Materialkostnad" debet="117510.00" kredit="0.00"
utgsaldo="117510.00"/>
<Delsumma level="2" delsumdebit="117510.00" delsumkredit="0.00" delsumutgsaldo="117510.00"/>
<Blankrad level="3" blank=" "/>
<Row level="1" kontonr="5010" kontonamn="Lokalhyra" debet="48000.00" kredit="0.00" utgsaldo="48000.00"/>
<Delsumma level="2" delsumdebit="48000.00" delsumkredit="0.00" delsumutgsaldo="48000.00"/>
<Blankrad level="3" blank=" "/>
<Row level="1" kontonr="6110" kontonamn="Kontorsmateriel" debet="6750.00" kredit="0.00" utgsaldo="6750.00"/>
<Delsumma level="2" delsumdebit="6750.00" delsumkredit="0.00" delsumutgsaldo="6750.00"/>
<Blankrad level="3" blank=" "/>
<Row level="1" kontonr="8490" kontonamn="Övriga finansiella kostnader" debet="550.00" kredit="0.00"
utgsaldo="550.00"/>
<Delsumma level="2" delsumdebit="550.00" delsumkredit="0.00" delsumutgsaldo="550.00"/>
<Blankrad level="3" blank=" "/>
<Totalsumma level="4" totaldebit="1050680.00" totalkredit="1050680.00" totalutgsaldo="0.00"/>
</KugarData>

```

Utdata till Kspread är i CSV-format, ex:

```

1220,Inventarier,18750.00,0.00,18750.00
Delsumma,,18750.00,0.00,18750.00
2081,Aktiekapital,0.00,300000.00,-300000.00
2330,Checkräkningskredit,799450.00,60000.00,739450.00
2350,Banklån,0.00,500000.00,-500000.00
2440,Leverantörsskulder,0.00,190680.00,-190680.00
2611,Utgående moms,10215.00,0.00,10215.00
2641,Ingående moms,49455.00,0.00,49455.00
Delsumma,,859120.00,1050680.00,-191560.00
4010,Materialkostnad,117510.00,0.00,117510.00
Delsumma,,117510.00,0.00,117510.00
5010,Lokalhyra,48000.00,0.00,48000.00
Delsumma,,48000.00,0.00,48000.00
6110,Kontorsmateriel,6750.00,0.00,6750.00
Delsumma,,6750.00,0.00,6750.00
8490,Övriga finansiella kostnader,550.00,0.00,550.00
Delsumma,,550.00,0.00,550.00

```



För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.

Detta görs i main.cpp.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen SDOLISW och KTORPT.

Programmet gör summeringar dels per konto och dels per kontoklass.

SDOLISW anropar KTORPT via STYRMAN.

```
const char *userp = getenv("USER");
QString usr(userp);

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("KTORPT");              // OLFIX funktion
process->addArgument(bar);                   // Bokföringsår
```

Detta blir:

```
./STYRMAN usr KTORPT bar
```

usr är den inloggades userid.

bar är bokföringsår.

och

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("FTGDSP");              // OLFIX funktion
process->addArgument("FNAMN");               // Företagsnamn
```

Detta blir:

```
./STYRMAN usr FTGDSP FNAMN
```

FNAMN för att hämta företagsnamnet som skrivs i huvudet på saldolistan.

Dessutom anropas VERHDSP

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument("VERHDSP");             // OLFIX funktion
process->addArgument(bar);                   // Bokföringsår
```

Detta blir:

```
./STYRMAN usr VERHDSP bar
```

Som avslutning anropas PRTAPI som är utskriftsinterfacet.

```
process = new QProcess();
process->addArgument("./PRTAPI");             // OLFIX funktion
process->addArgument(csvflag);                 // Kugar eller Kspread
process->addArgument(printfile);               // Datafilen
process->addArgument(templatefile);            // Utskriftsmallen
```

Detta blir:

```
./PRTAPI csvflag printfile [templatefile]
```

**Behörighetskrav:**

För att kunna köra SDOLISW behövs behörighet till

PRGLST

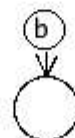
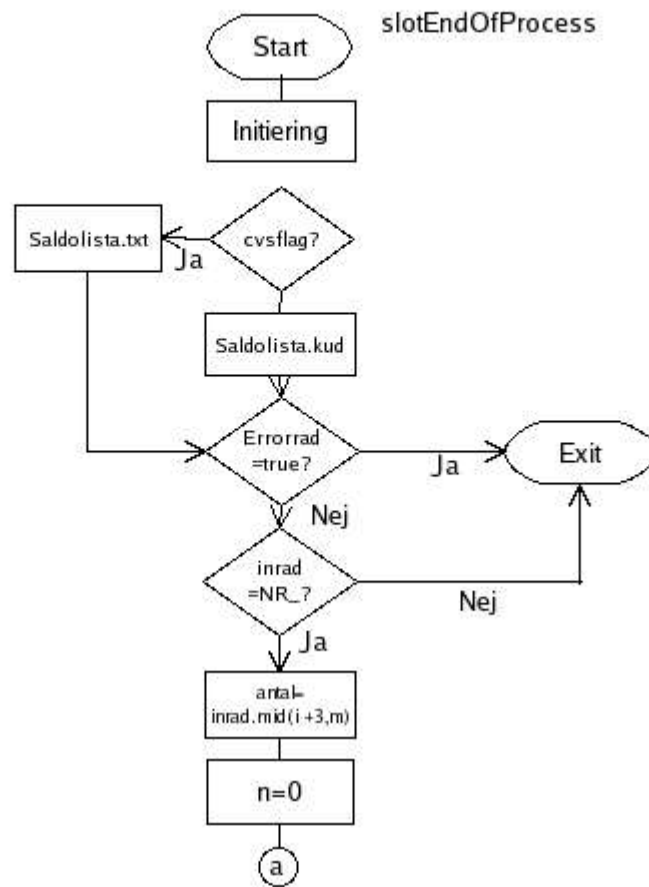
FTGDSP

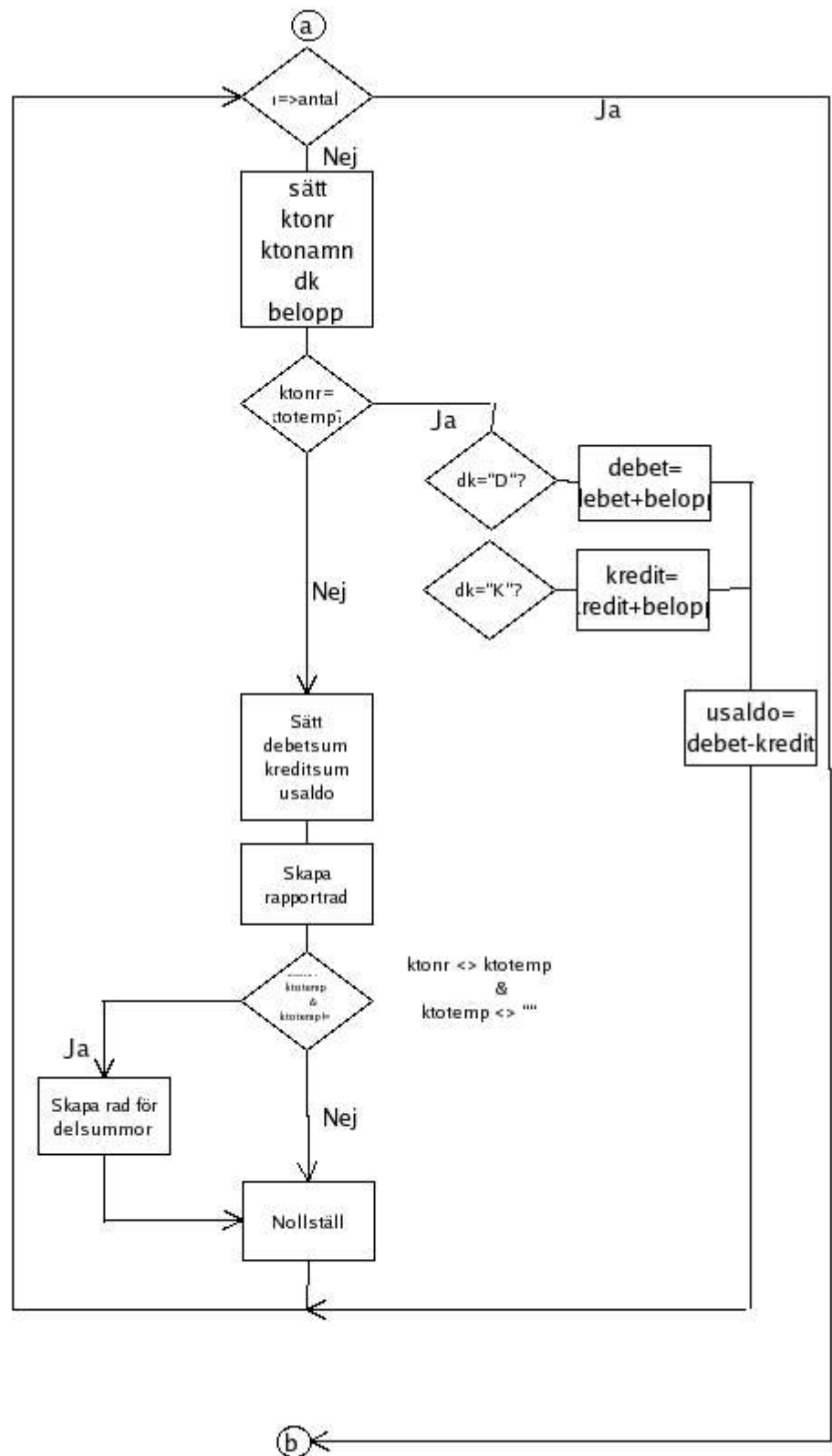
KTORPT

VERHDSP

PRTAPI

## Flödesschema





## Konsolprogram i OLFIX

Konsolprogrammen är icke kompletta och innehåller ingen felhantering.

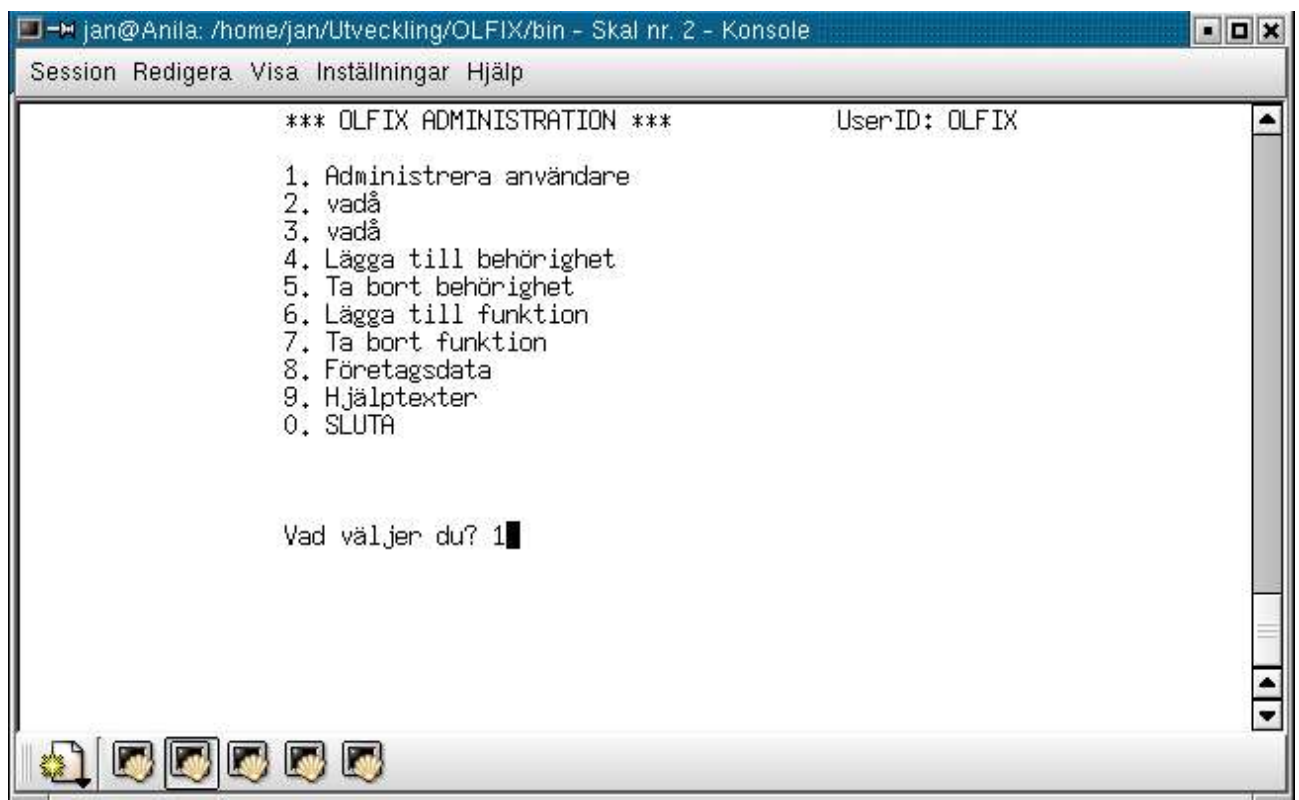
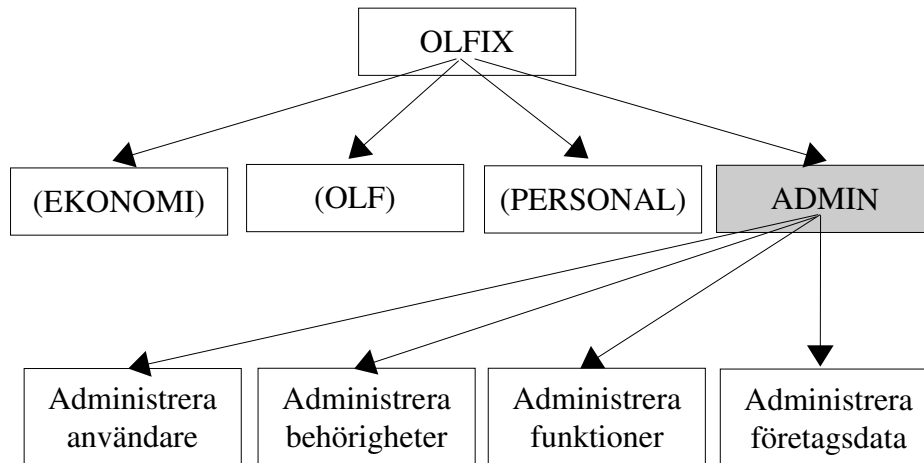
ADMIN	Konsolprogram.
BOKF	Konsolprogram för bokföring.
FORADM	Konsolprogram.
OLFIX	Huvudprogram, konsolprogram.
REDOV	Konsolprogram.

## ADMIN(program)

ADMIN är ett menyprogram för konsol som anropas från konsolprogrammet OLFIX  
Userid följer med från OLFIX.

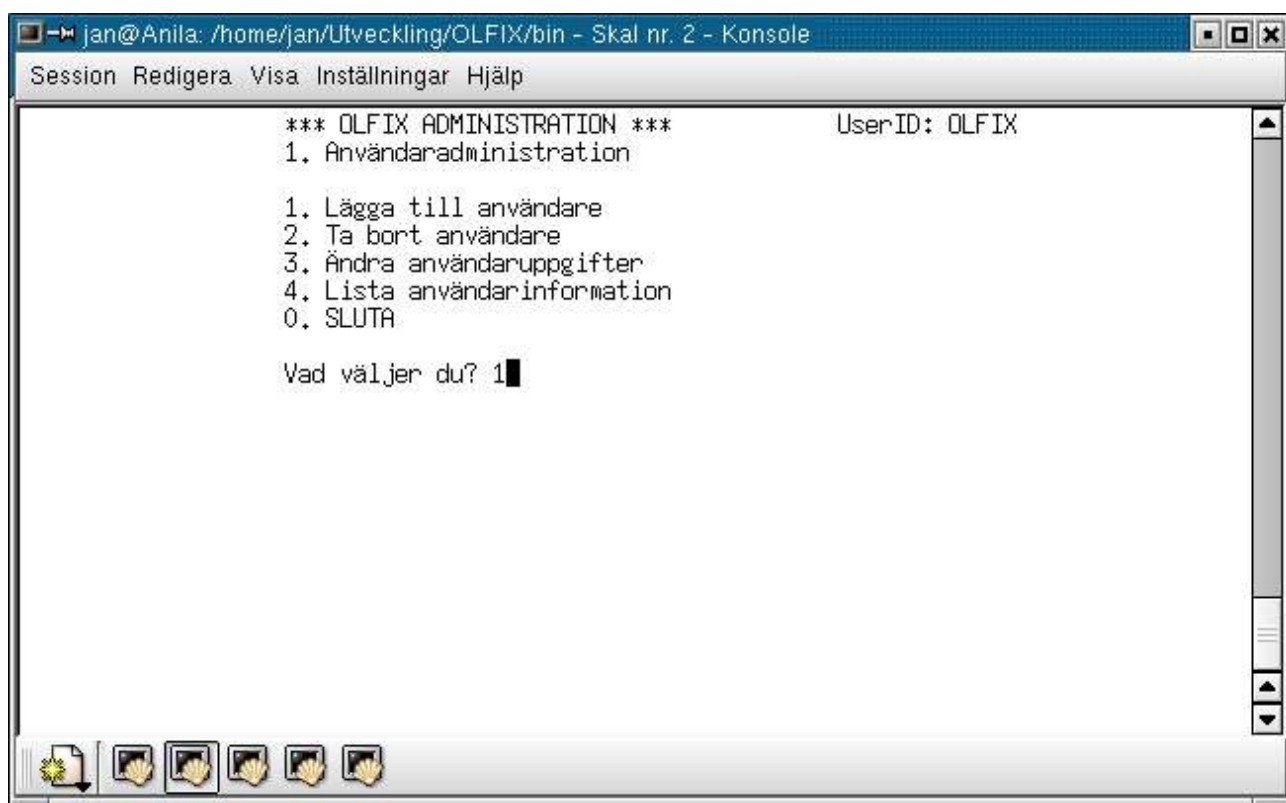
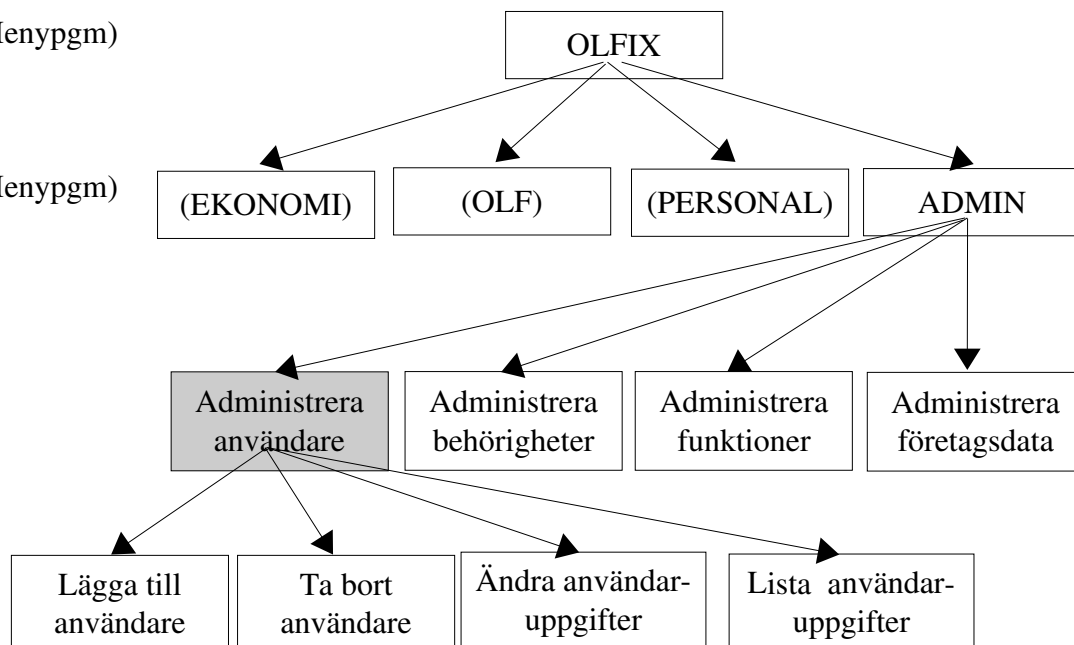
(Menypgm)

(Menypgm)

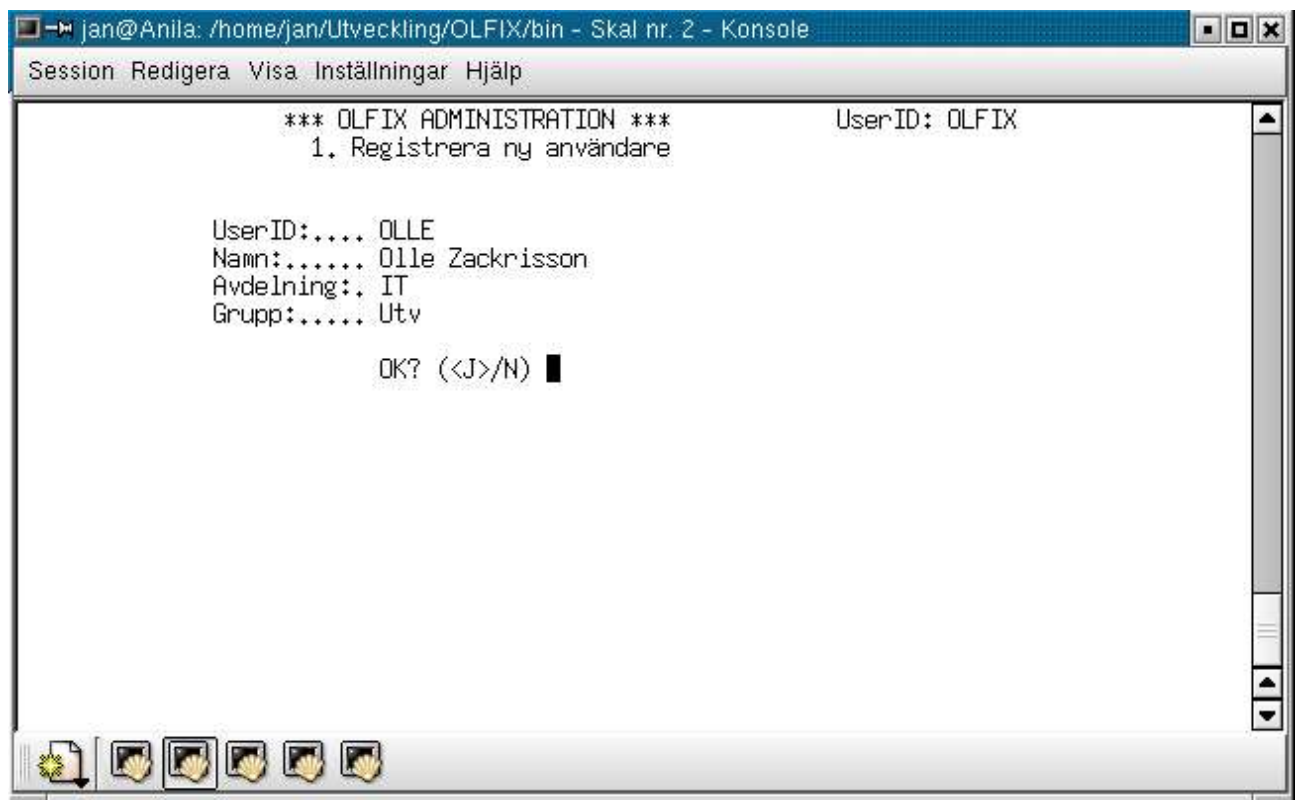


(Menypgm)

(Menypgm)







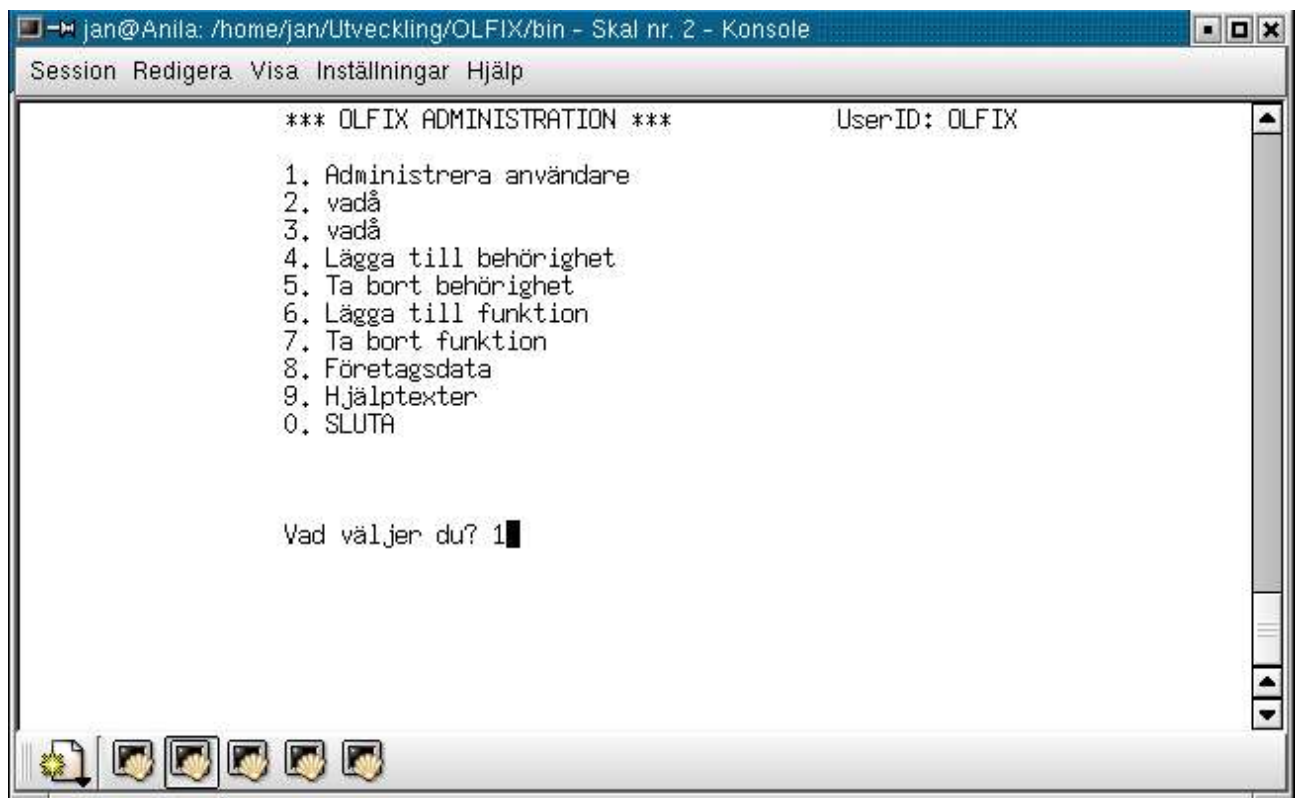
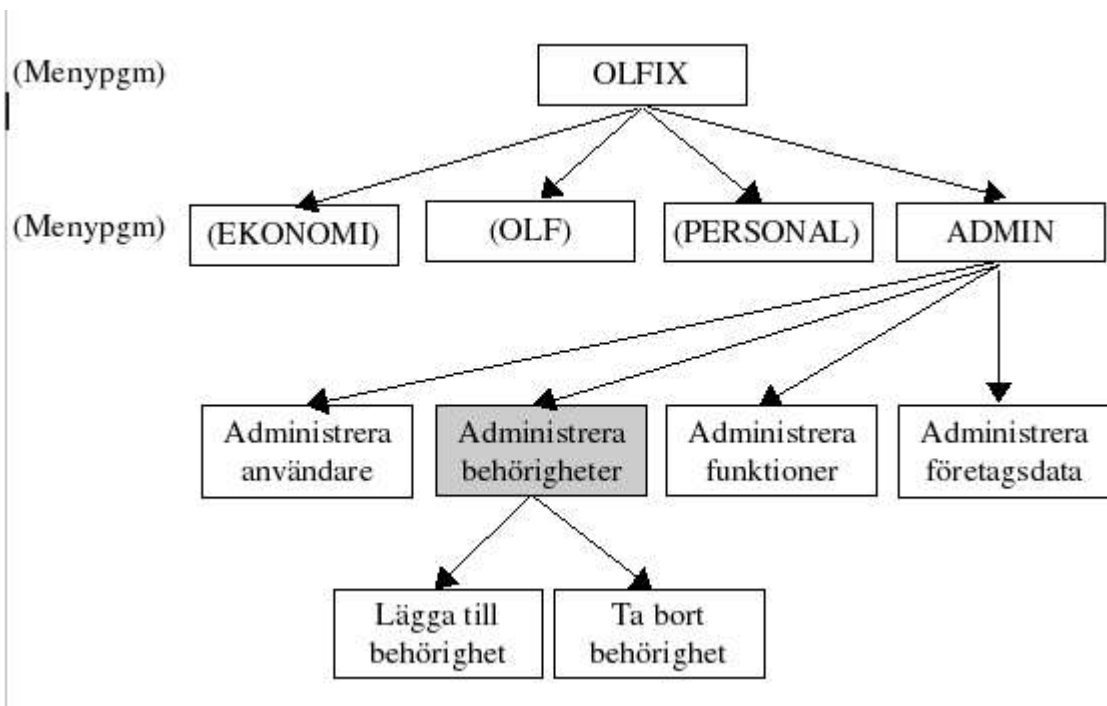
\*\*\* OLFIX ADMINISTRATION \*\*\*

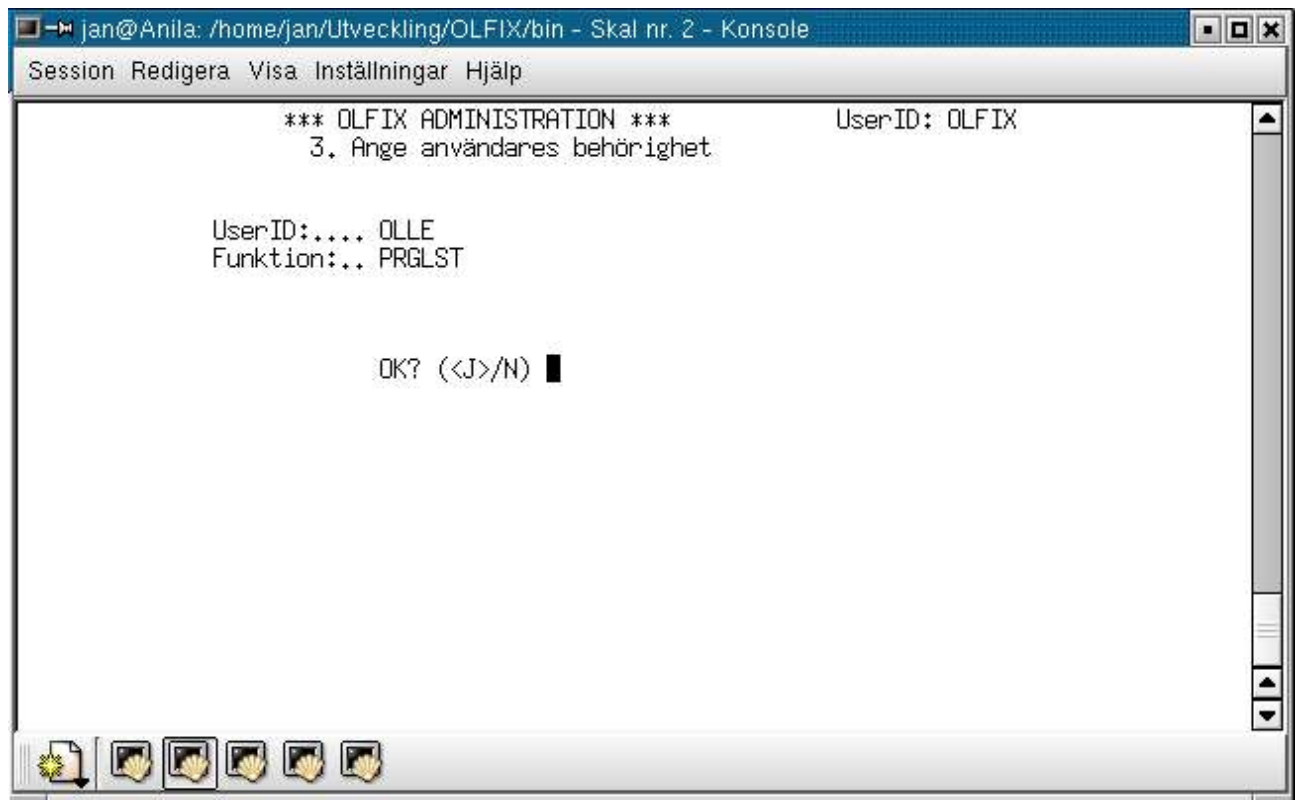
UserID: JAPI

4. Lista användare

UserID	Namn	Avd	Grupp
JAPI	Jan Pihlgren	Prod	Kista
OLFIX	Olfix Superuser	IT	Stab
pelle	Pelle Andersson	Prod	PT
SARA	Sara Johansson	Ekonomi	Stab

OK? (<J>/N)





*** OLFIX ADMINISTRATION ***	UserID: olfix
4. Ta bort användares behörighet	
UserID:....	
Funktion:..	

# BOKF

OBS! Fungerar ej (2003-03-08)

BOKF är ett konsolprogram där bokföringen sker.

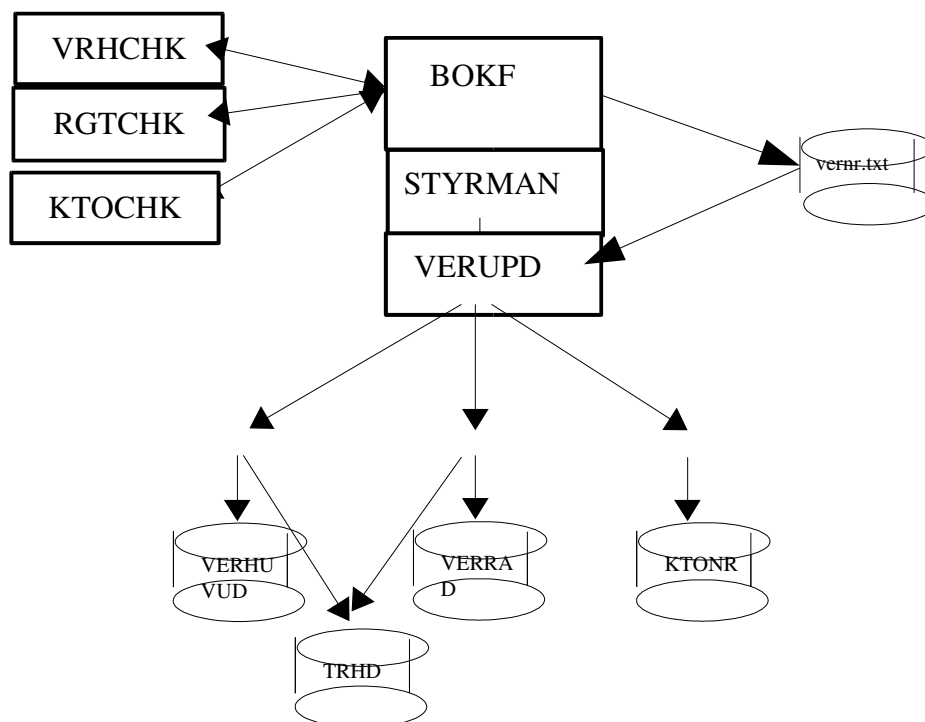
Initialt kontrolleras att user har behörighet att använda programmet.

Posterna lagras i temporärfilen vernr.txt

BOKF kontrollerar fortlöpande saldot på verifikationen vilket registreras på verifikationsrad nr 1.

För varje därpå följande verifikationsrad så minskas saldot med det belopp som konteras på raden. Först när saldot är 0 (noll) så godkänns verifikationen som färdigbehandlad.

Därefter överlämnas arbetet via STYRMAN till funktionen VERUPD som uppdaterar databasen från filen vernr.txt. När alla poster i vernr.txt är behandlade så raderas filen.

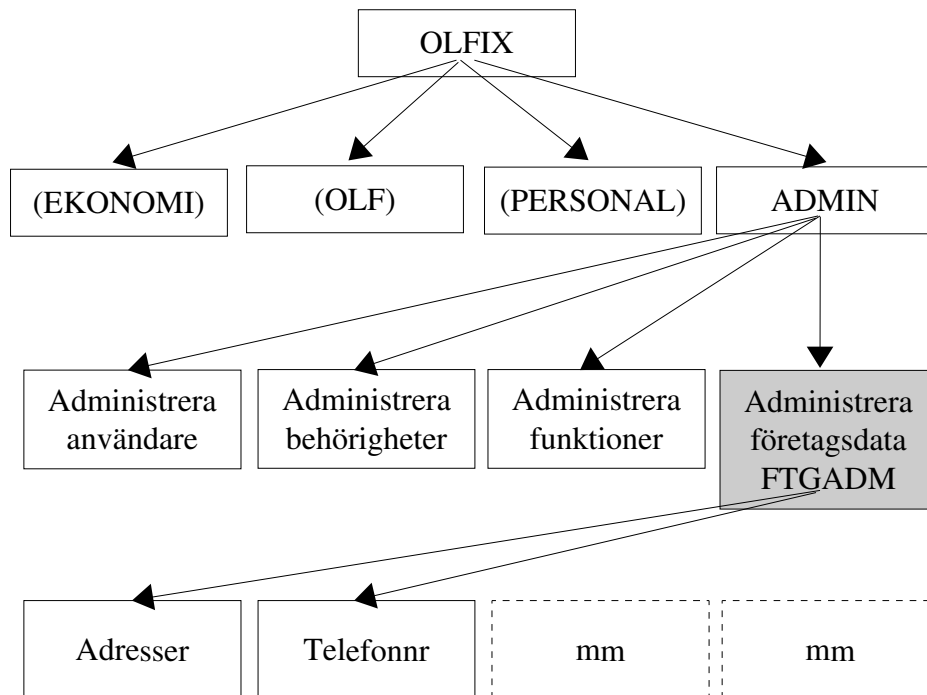


## FTGADMprogram)

FOREG är ett menyprogram för konsol som anropas från konsolprogrammet OLFIX via ADMIN. Userid följer med från OLFIX.

(Menypgm)

(Menypgm)



\*\*\* OLFIX \*\*\*

UserID: JAPI  
Företagsdata

1. Adresser
2. Telefonnummer
3. vadå
4. vadå
5. vadå
6. vadå
7. vadå
8. vadå
9. vadå
0. SLUTA

Vad väljer du?

\*\*\* OLFIX \*\*\*

UserID: JAPI

Företagsdata, Adresser

A. Företagssvar:

Företagets postadress

-----

1. Postadr:

2. Postnr:

3. Ort

Besöksadress

-----

4. Gatuadr:

5. Postnr:

6. Ort

Leveransadress

-----

7. Gatuadr:

8. Postnr:

9. Ort

0. SLUTA

Vilket fält väljer du?



## OLFIX (konsolprogram)

OLFIX är ett menyprogram för konsol.

Initialt uppmanas man att ange sitt userid. I slutversionen av programmet ska userid hämtas från systemet.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=databas existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

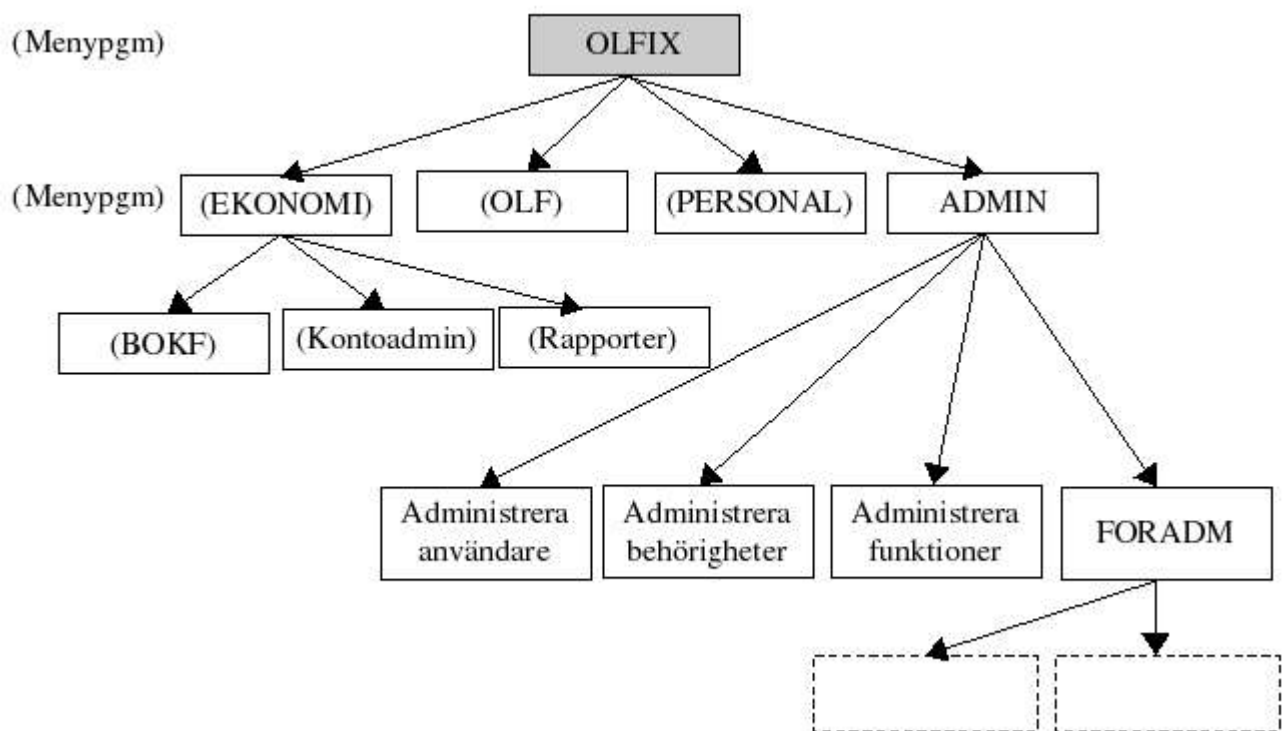
1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

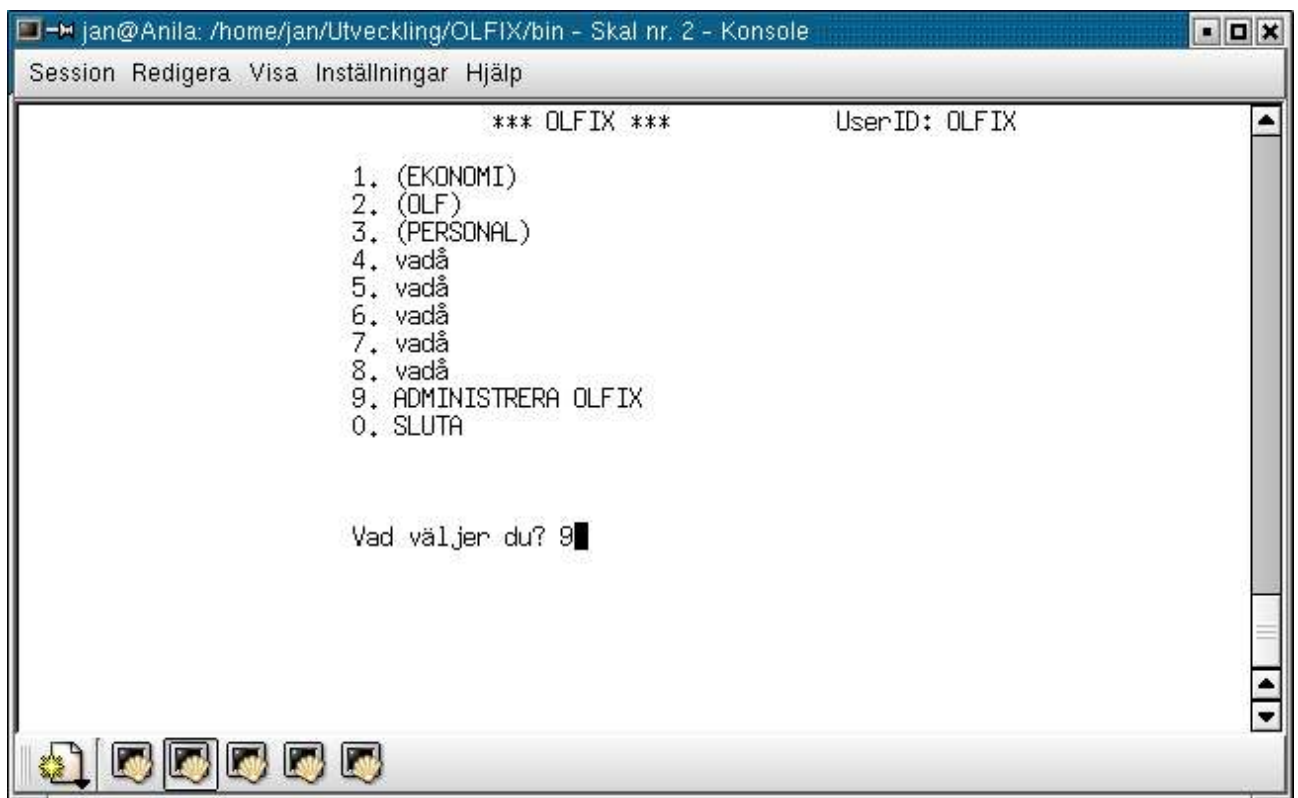
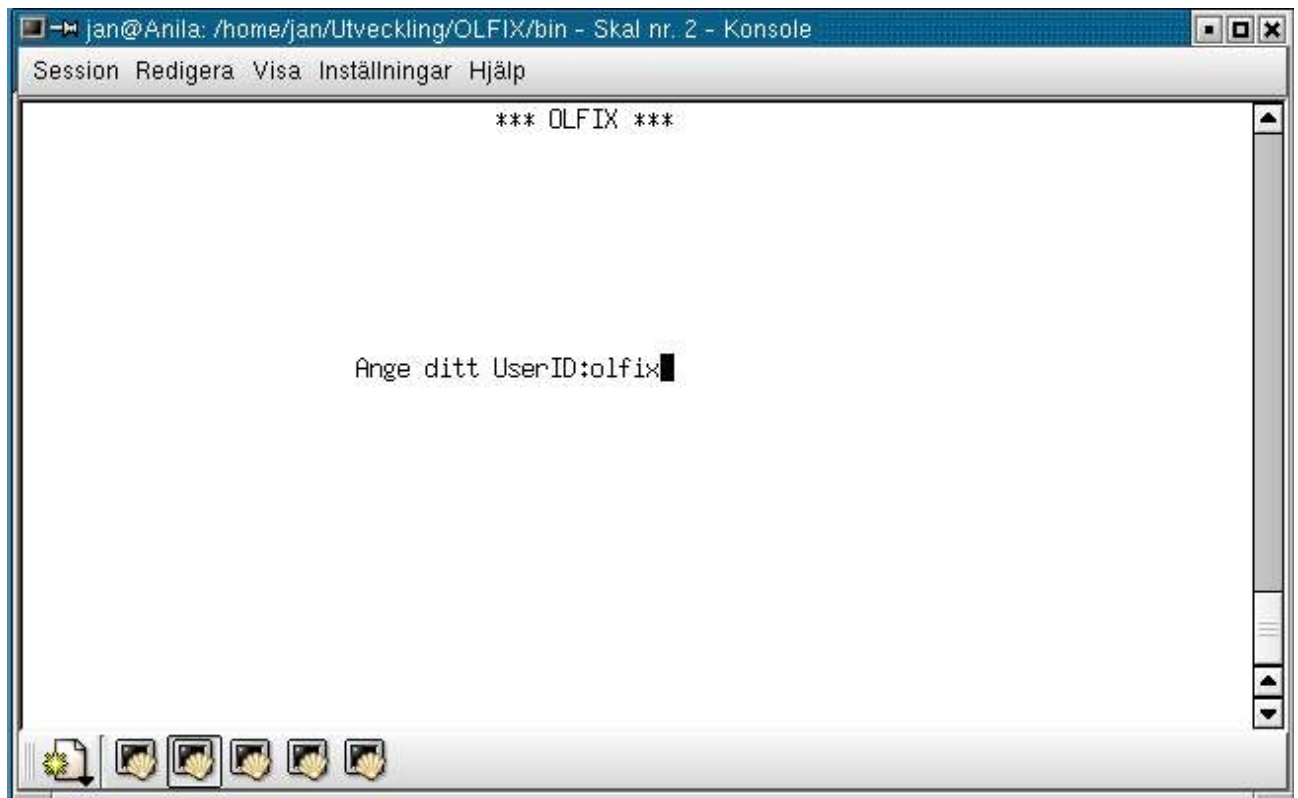
B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.





\*\*\* OLFIX REDOVISNING \*\*\*

UserID: JAPI

1. BOKFÖRING, registrera verifikationer
2. (Dagboksrapport)
3. (Huvudboksrapport)
4. Lägga till konton
5. Ta bort konton
6. Ändra bokföringsperioder
7. Ändra momssatser
8. Automatkontering (J/N)
9. vadå
0. SLUTA

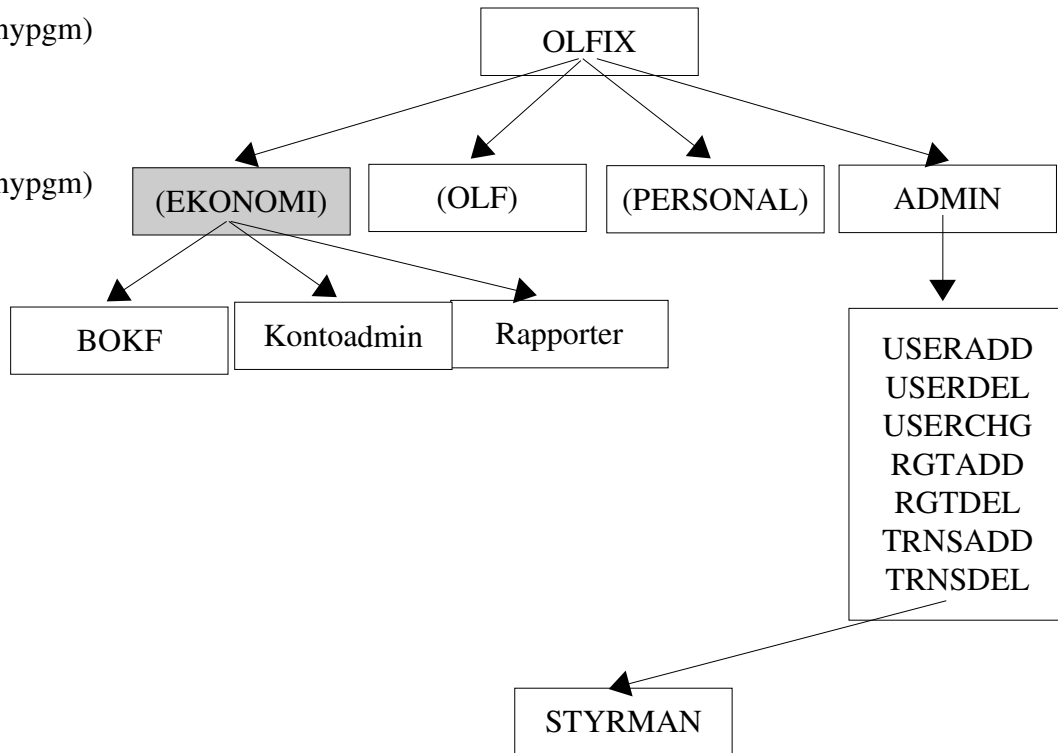
Vad väljer du?

## REDOV(konsolprogram)

REDOV är ett menyprogram för konsol som anropas från konsolprogrammet OLFIX.  
Userid följer med från OLFIX.

(Menypgm)

(Menypgm)



\*\*\* OLFIX REDOVISNING \*\*\*

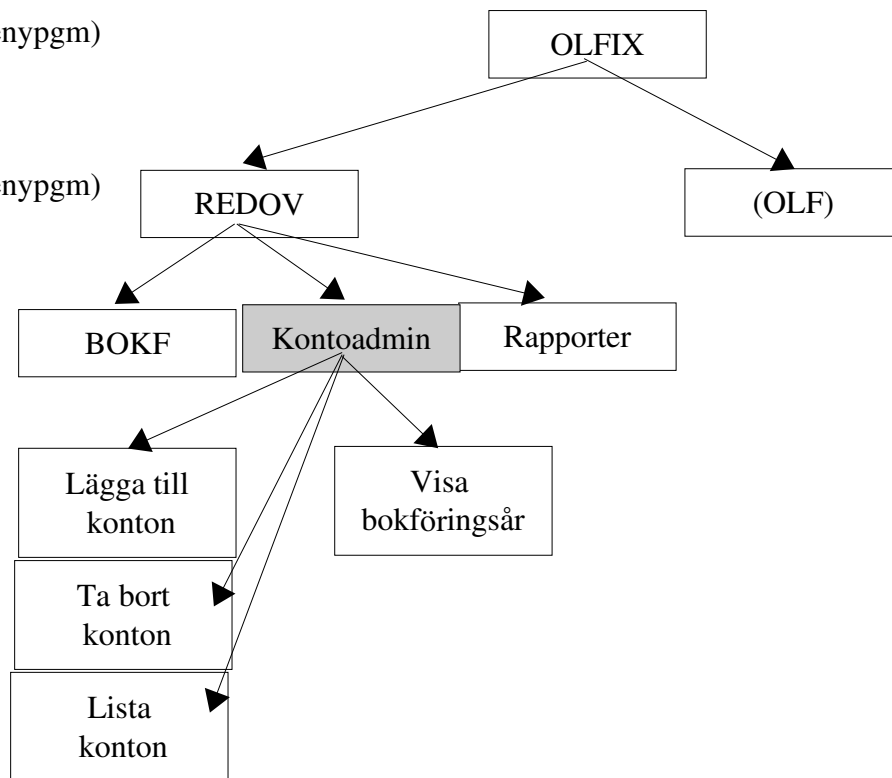
UserID: JAPI

1. BOKFÖRING, registrera verifikationer
2. (Dagboksrapport)
3. (Huvudboksrapport)
4. Kontoadministration
5. vadå
6. vadå
7. Ändra bokföringsperioder
8. Ändra momssatser
9. Automatkontering (J/N)
0. SLUTA

Vad väljer du?

(Menypgm)

(Menypgm)



\*\*\* OLFIX REDOVISNING \*\*\*

UserID: JAPI

4. Kontoadministration

1. Lägga till konton
2. Ta bort konton
3. Lista konton
4. Ändra konto
5. Lägga till ett bokföringsår
6. Visa ett bokföringsår
7. vadå
8. vadå
9. Kopiera kontoplan till nytt bokföringsår
0. SLUTA

Vad väljer du?

\*\*\* OLFIX REDOVISNING \*\*\*

UserID: JAPI

3. Lista konton

Bokför.år:

Kontonr Kontobeskrivning

-----

Ange bokföringsår: AC



3. Lista konton

Kontonr Kontobeskrivning

1010	Kassa
1020	Postgiro
1040	Checkkonto
1050	Bank
1120	Aktier och andelar
1210	Kundfodringar
1230	Belånade kundfodringar (factoring)
1310	Förutbetalda hyreskostnader
1330	Förutbetalda försäkringskostnader
1350	Upplupna hyresintäkter
1360	Upplupna ränteintäkter
1410	Fordringar hos anställda
1430	Fordringar hos leverantörer
1450	Skattefordringar
1470	Ingåendemervardesskatt (moms)
1510	Lager
1530	Produkter i arbete (PIA)

Fortsätta? <J>/N

\*\*\* OLFIX REDOVISNING \*\*\*

6. Visa bokföringsår

Bokföringsår: AC

Benämning: 2002-08-01-2003-07-31

Startdatum: :2002-08-01

Slutdatum: 2003-07-31 Året låst: N

Beskattningsår: 2003

Kontoplan: AC

Senaste verdatum: 0000-00-00

Nästa vernummer: 1

Tryck ENTER!

## Standardrapporter i OLFIX

<b>ATTBETW</b>	Leverantörsfakturor förfallna till betalning senast åååå-mm-dd, även utskrift.
<b>RPTKTOW</b>	Kontorapport, endast på skärm.
<b>LEVRESKW</b>	Leverantörsreskontrarapport, endast skärm.
<b>HUVBOKW</b>	Huvudbok
<b>SDOLISW</b>	Saldolista

## Funktionerna i OLFIX

path/STYRMAN userid betyder “anropa programmet STYRMAN med userid på den som har rätt att utföra den följande funktionen”

Därefter följer vilken funktion som skall utföras, följt av parametrar till funktionen.

Funktionerna testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixstst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixstst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixstst**.

Vilkor C överrider villkoren A och B.

**ARICHK** Kontrollera om bokföringsår finns. (2003-03-08. Ska bort.)  
Path/STYRMAN userid *ARICHK arid*

**ARADD** Lägga upp en ny artikel.  
Path/STYRMAN userid *ARADD artikeldata [databas]*

**ARCHK** Kontrollera om en artikel finns registrerad.  
Path/STYRMAN userid *ARCHK artikelnr [databas]*

**ARLST** Lista artiklarna i artikelregistret. Artikelnummer, benämning 1 och benämning 2.  
Path/STYRMAN userid *ARLST [databas]*

**ATTBET** Lista vilka leverantörsfakturor som förfaller till betalning ÅÅÅÅMMDD  
Path/STYRMAN userid *ATTBET datum [databas]*

**BARADD** Lägga upp nytt bokföringsår i tabellen BOKFAR.  
Path/STYRMAN userid *BARADD arid,benamning, arstart, arslut, arlast, beskattningsar, senverdat, vernr, ktoplan [databas]*.

**BARCHG** Ändra data för ett bokföringsår i tabellen BOKFAR  
Path/STYRMAN userid *BARCHG arid,benamning, arstart, arslut, arlast, beskattningsar, senverdat, vernr, ktoplan*.

**BARCHK** Kontrollera om bokföringsår finns.  
Path/STYRMAN userid *BARCHK arid [databas]*

**BARDSP** Visa/Hämta data för angivet bokföringsår ur tabellen BOKFAR.  
Path/STYRMAN userid *BARDSP arid [databas]*

- BARFND** Söka efter bokföringsår vars startdatum är mindre än *datum* och vars slutdatum är större än *datum*.  
Path/STYRMAN userid *BARFND datum [databas]* (ÅÅÅÅ-MM-DD)
- BETADD** Lägga till en nytt betalningsvillkor i tabellen BETVILKOR.  
Path/STYRMAN userid *BETADD betvilkor dagar beskrivning [databas]*
- BETCHG** Ändra data för ett betalningsvillkor i tabellen BETVILKOR.  
Path/STYRMAN userid *BETADD betvilkor dagar beskrivning [databas]*
- BETDSP** Visa/hämta data för angivet betalningsvillkor ur tabellen BETVILKOR.  
Path/STYRMAN userid *BETDSP betvilkor [databas]*
- BETLST** Hämta alla betalningsvillkor ur tabellen BETVILKOR.  
Path/STYRMAN userid *BETLST [databas]*
- DBOKRPT** Hämta data till dagboksrapport.  
Path/STYRMAN userid *DBOKRPT bar fromdatum tomdatum [databas]*
- FTGADD** Lägga till ny post i företagsregistret (tabell FTGDATA)  
Path/STYRMAN userid *FTGADD posttyp postbeskr fdata [databas]*
- FTGDSP** Hämta data från företagsregistret.  
Path/STYRMAN userid *FTGDSP posttyp*
- FTGLST** Lista förteckning av posttyper från företagsregistret.  
Path/STYRMAN userid *FTGLST*
- FTGLIS** Lista företagsdata från företagsregistret.  
Path/STYRMAN userid *FTGLIS*
- FGTUPD** Uppdatera företagsregistret  
Path/STYRMAN userid *FTGUPD posttyp 'fdata'*
- HBOKRPT** Hämta data till huvudboksrapport.  
Path/STYRMAN userid *HBOKRPT bar fromdatum tomdatum*
- INKADD** Lägga upp en ny inköpsorder.  
path/STYRMAN userid *INKADD orderhuvudata*
- INKRADD** Lägga upp en ny inköpsorderrad.  
path/STYRMAN userid *INKRADD orderraddata*
- INKHDSP** Visa orderhuvud på angiven inköpsorder.  
path/STYRMAN userid *INKHDSP inkordnr*

**INKLST** Visa alla inköpsorderrader i INKRADREG (Beställningsstock).  
path/STYRMAN userid *INKLST*

**INKRLST** Visa alla orderrader på angiven inköpsorder.  
path/STYRMAN userid *INKHDSP inkordnr*

**KSTADD** Lägga upp nytt kostnadställe.  
Path/STYRMAN userid *KSTADD arid kstalle benamning*

**KSTCHK** Kontrollera om kostnadställe finns.  
Path/STYRMAN userid *KSTCHK arid kstalle*

**KSTDSP** Visa information om ett kostnadställe.  
Path/STYRMAN userid *KSTDSP arid kstalle*

**KSTLST** Lista alla kostnadställen.  
Path/STYRMAN userid *KSTLST*

**KTOADD** Lägga upp nya kontonummer.  
path/STYRMAN userid *KTOADD arid ktonr benamning manuell momskod srunk  
kstalle projekt subkto ktoplan*

**KTOCHK** Kontrollera om kontonummer finns.  
*path/STYRMAN userid KTOCHK kontonr*

**KTOLST** Lista kontonr med beskrivning  
path/STYRMAN userid *KTOLST*

**KTORPT** Läs ut data ur VERRAD.  
Path/STYRMAN userid *KTORPT arid*

**KTOUPD** **OBS! Använd ej! Ska få ny funktionalitet!**  
Uppdatera kontonummerr med verifikationsbelopp  
path/STYRMAN userid *KTOUPD kontonr y belopp*

**KTOVIEW** Lista kontonr med beskrivning från tabellen KTOPLAN  
path/STYRMAN userid *KTOVIEW arid*

**KUADD** Lägga upp nya kunder  
Path/STYRMAN userid *KUADD kunddata [databas]*

**KUCHG** Ändra/uppdatera kunddata.  
Path/STYRMAN userid *KUCHG kunddata [databas]*

**KUCHK** Kontrollera om kundnr finns.  
Path/STYRMAN userid *KUCHK kundnr [databas]*

- KUDSP** Hämta kundata  
Path/STYRMAN userid *KUDSP kundnr [databas]*
- KULST** Lista kunder, kundnr och namn.  
Path/STYRMAN userid *KULST [databas]*
- LEVADD** Lägga till nya leverantörer till tabellen LEVREG  
path/STYRMAN userid *LEVADD levnr levorgnr levnamn levadress levpostnr  
levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfnnr  
levmoms kod levskuld levkonto*
- LEVCHG** Ändra data på en leverantör.  
path/STYRMAN userid *LEVCHG levnr levorgnr levnamn levadress levpostnr  
levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfnnr  
levmoms kod levkonto*
- LEVDSP** Visa information om en leverantör.  
path/STYRMAN userid *LEVDSP levnr*
- LEVLST** Lista leverantörer, leverantörsnummer och leverantörsnamn.  
path/STYRMAN userid *LEVLST*
- LEVSADD** Lägga till nya leveranssätt.  
path/STYRMAN userid *LEVSADD levsettnr levsetttext*
- LEVSDSP** Visa leveranssätt.  
path/STYRMAN userid *LEVSDSP levsettnr*
- LEVSLST** Lista alla leveranssätt.  
path/STYRMAN userid *LEVSLST*
- LEVVADD** Lägga till nya leveransvillkor.  
path/STYRMAN userid *LEVVADD villkorsnr villkorstext*
- LEVVDSP** Visa leveransvillkor.  
path/STYRMAN userid *LEVVDSP levvillkornr*
- LEVVLST** Lista alla leveranssätt.  
path/STYRMAN userid *LEVVLST*
- LRESADD** Lägga till en post i leverantörsreskontran.  
path/STYRMAN userid *LRESADD levnr fakturanr regdatum faktdatum expiredatum  
fakttext bar moms procent levktonr faktbelopp moms ktonr momsbelopp kreditkontonr  
kreditbelopp userid*
- LRESRPT** Lista obetalda leverantörsfakturor på skärm.

Path/STYRMAN userid *LRESRPT*.

- PRGLST** Lista programnamn för programmet OLFIXW.  
path/STYRMAN userid *PRGLST*
- PRTAPI** Interface till utskriftsprogram för rapporter.  
path/*PRTAPI csvflag prtfile [prttemplate]*
- RGTADD** Lägga till nya rättigheter för användare  
path/STYRMAN userid *RGTADD userid function*
- RGTCHK** Kontrollera om användare har viss behörighet.  
path/STYRMAN userid *RGTCHK userid funktion*
- RGTDEL** Ta bort rättigheter för användare  
path/STYRMAN userid *RGTDEL userid funktion*
- RGTDSP** Visa rättigheter för en användare  
path/STYRMAN userid *RGTDSP userid funktion*
- RGTLST** Visa rättigheter för alla användare  
path/STYRMAN userid *RGTLST*
- RPTCRE** Rapportgenerator som skapar en CSV-fil utifrån valfri SQL-fråga  
path/STYRMAN userid *RPTCRE sqlquery*
- SLPADD** Lägga till nya standardleveransplatser.  
Path/STYRMAN userid *SLPADD kundnr stdlevplats adress postnr postadress land*
- STYRMAN** Centralt styrprogram  
path/STYRMAN ..... se resp funktion.
- TRHDADD** Logga ekonomiska transaktioner  
path/STYRMAN userid *TRHDADD trnsid "tid" userid "trnsdata"*
- TRNSADD** Lägga till nya transaktionstyper  
path/STYRMAN userid *TRNSADD function "functiontext"*
- TRNSLST** Lista transaktionstyper  
path/STYRMAN userid *TRNSLST*
- TXTADD** Lägga till nya texter i TEXTREG  
path/STYRMAN userid *TXTADD textnr txt*
- TXTDEL** Radera post i TEXTREG  
path/STYRMAN userid *TXTDEL textnr*



**TXTDSP** Visa en post i TEXTREG.  
path/STYRMAN userid *TXTDSP textnr*

**USERADD** Lägga till nya användare  
path/STYRMAN userid *USERADD userid "username" department group*

**USERCHG** Ändra data på en användare  
path/STYRMAN userid *USERCHG userid "username" department group*

**USERDEL** Ta bort användare  
path/STYRMAN userid *USERDEL userid*

**USERDSP** Visa information om en användare  
path/STYRMAN userid *USERDSP userid*

**USERLST** Visa information på alla användare  
path/STYRMAN userid *USERLST*

**VALADD** Lägga till ny valuta.  
Path/STYRMAN userid *VALCHG valuta land salj kop beteckning*

**VALCHG** Ändra information för en valuta.  
Path/STYRMAN userid *VALCHG valuta land salj kop beteckning*

**VALDEL** Ta bort en valuta.  
Path/STYRMAN userid *VALDEL valuta.*

**VALDSP** Visa information om en valuta.  
Path/STYRMAN userid *VALDSP valuta.*

**VALLST** Lista information om alla valutor.  
Path/STYRMAN userid *VALLST*

**VERUPD** Uppdatera VERH, VERD och KONTONR  
path/STYRMAN userid *VERUPD path/vernr.txt (vernr=siffror)*

# ARADD

Funktionen anropas med artikeldata som parameter.  
Funktionen används för att lägga upp en nya artiklar.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/ARADD *artikeldata* [databas]

Exempel:

```
$ ./ARADD artikeldata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN *userid* ARADD *artikeldata* *databas*

userid = userid på den som ställer frågan.

```
artikeldataformat = "_:2345_:Testartikel_:Provprodukt_:ST_:2,5_:200,00_:20_:12345_:67890_:2_:1234567_:DA123_:Mesurment Part_:England_:MP23Z_:2,500_:"
```

\_: = fältavskiljare

Exempel:

```
$ ./STYRMAN JAPI ARADD "_:2345_:Testartikel_:Provprodukt_:ST_:2,5_:200,00_:20_:12345_:67890_:2_:1234567_:DA123_:Mesurment Part_:England_:MP23Z_:2,500_:" databas
```

SQLsats som används;

```
INSERT INTO ARTIKELREG
(ARTIKELNR,ARBENEMNING1,ARBENEMNING2,ARENHET,ARFPRIS,ARLEDTID,
ARPRODKLASS,ARPRODKTO,ARLEVNr1,ARLEVNr2,ARLEVNr3,ARNETTOVIKT,ARARTTTYP,
ARSTRUKT,ARURBENEMNING,ARURLAND,ARURARTNR,ARTULLTAX,ARVOLYM) VALUES (
"2345","Testartikel","Provprodukt","ST","2,5","200,00","20","12345","67890",
"2"," ","1234567","DA123"," "," ",
"Mesurment Part","England","MP023Z","2,500")
```

## Interface:

INPUT:

artikeldata

OUTPUT:

```
fprintf(stdout,"OK: ARADD Inserted %lu rows",
        (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: ARADD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: ARADD Connection failed\n");
fprintf(stderr,"Error: ARADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# ARCHK

Funktionen anropas med artikelnr som parameter.

Funktionen används för att kontrollera om angivet artikelnummer finns registrerat.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/ARCHK artikelnr [databas]

Exempel:

```
$ ./ARCHK artikelnr [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARCHK artikelnr [databas]

userid = userid på den som ställer frågan.

Exempel:

```
$ ./STYRMAN JAPI ARCHK 2345
```

SQLsats som används;

```
SELECT ARTIKELNR FROM ARTIKELREG WHERE ARTIKELNR = "2345"
```

## Interface:

INPUT:

ARTIKELNR

OUTPUT:

```
fprintf(stdout,"OK: ARCHK Status = %d\n",status);
fprintf(stderr,"Error: ARCHK Artikelnr %s finns inte!\n",artikelnr);
fprintf(stderr,"Error: ARCHK SELECT error: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: ARCHK Retrieval error: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: ARCHK Connection failed\n");
fprintf(stderr,"Error: ARCHK Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# ARLST

Funktionen anropas utan parameter eller, som option, *databas*.  
Funktionen används för att lista artikelnummer, benämning 1 och benämning 2.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [*databas*] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixst**, testföretag
  - B. Om parameter [*databas*] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixst**, testföretag
    - 2. Annars används värdet i [*databas*]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/ARLST [*databas*]

Exempel:

```
$ ./ARLST [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARLST [*databas*]

userid = userid på den som ställer frågan.

Exempel:

```
$ ./STYRMAN JAPI ARLST [olfix]
```

SQLsats som används;

```
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2 FROM ARTIKELREG ORDER BY ARTIKELNR
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stdout,"\n");
fprintf(stderr,"Error: ARLST SELECT error: %s\n",
           mysql_error(&my_connection));
fprintf(stderr,"Error: ARLST Retrieval error: %s\n",
           mysql_error(&my_connection));
fprintf(stderr,"Error: ARLST Connection failed\n");
fprintf(stderr,"Error: ARLST Connection error %d: %s\n",
           mysql_errno(&my_connection), mysql_error(&my_connection));
```

## ARICHK

2003-03-08. Ska bort.

Funktionen anropas med parametern ARID.

Kontrollera om bokföringsår ARID finns.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/ARICHK arid

Exempel:

```
$ ./ARICHK AC
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARICHK arid

userid = userid på den som ställer frågan.

arid = det eftersökta bokföringsårets id.

Exempel:

```
$ ./STYRMAN JAPI ARICHK AC
```

Returvärde från ARICHK = 0 om arid finns i tabellen BOKFAR annars returneras värdet -1.

Funktionen är tänkt att användas för att kontrollera om bokföringsåret finns.

SQLsats som används;

```
SELECT ARID FROM BOKFAR WHERE ARID = "AC"
```

# ATTBET

Funktionen anropas med datum som parameter.

Funktionen listar leverantörsfakturer som förfaller till betalning till och med **datum**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/ATTBET datum [databas]

Exempel:

```
$ ./ATTBET "2003-08-21" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ATTBET datum [databas]

userid = userid på den som ställer frågan.

datum = ÅÅÅÅ-MM-DD.

Exempel:

```
$ ./STYRMAN JAPI ATTBET "2003-08-21" [olfix]
```

SQLsats som används;

```
SELECT EXPIREDATUM,LEVNR,FAKTURANR,FAKTBELOPP,VALUTA FROM LEVRESK where EXPIREDATUM <=
'2003-08-21' AND BETALD = 'N' ORDER BY EXPIREDATUM,LEVNR
```

## Interface:

INPUT:

DATUM

OUTPUT:

```
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);

fprintf(stderr,"Error: ATTBET SELECT error: %s\n",mysql_error
(&my_connection));
```



```
fprintf(stderr, "Error: ATTBET Retrieval error: %s\n",
mysql_error(&my_connection));
fprintf(stderr, "Error: ATTBET Connection failed\n");
fprintf(stderr, "Error: ATTBET Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# BARADD

Funktionen används för att lägga upp en nya konton.

Funktionen anropas med parametrarna ARID, BENAMNING, ARSTART, ARSLUT, ARLAST, BESKATTNINGSAR, SENVERDAT, VERNR, KTOPLAN [databas]

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/BARADD arid,ktonr,benamning,manuell,momskod,srunr,kstalle,projekt,subkto,ktoplan [databas]

Exempel:

```
$ ./BARADD SS 2003-01-01-2003-12-31 2003-01-01 2003-12-31 N 2003 0000-00-00
987124 EUBAS97 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BARADD arid benamning arstart arslut arlast beskattningsar senverdat vernr ktoplan [olfix].

ARLAST sätts till "N"

Exempel:

```
$ ./STYRMAN jan BARADD SS 2003-01-01-2003-12-31 2003-01-01 2003-12-31 N 2003 0000-00-00
987124 EUBAS97 [olfix]
```

SQLsats som används:

```
INSERT INTO BOKFAR(ARID,BENAMNING,ARSTART,ARSLUT,BESKATTBINGSAR,SENVERDAT,
VERNR,KTOPLAN)
VALUES ("SS","2003-01-01-2003-12-31","2003-01-01","2003-12-31","N","2003","0000-00-00",
"987124","EUBAS97")
```

## Interface:

INPUT:

ARID, BENAMNING, ARSTART, ARSLUT, BESKATTNINGSAR, SENVERDAT,  
VERNR, KTOPLAN

OUTPUT:

```
fprintf(stderr,"OK: BARADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: BARADD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: BARADD Connection failed\n");
fprintf(stderr,"Error: BARADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# BARCHG

Funktionen anropas med parametrarna ARID, BENAMNING, ARSTART, ARSLUT, ARLAST, BESKATTNINGSAR, SENVERDAT, VERNR, KTOPLAN [databas]  
Kontrollera om bokföringsår ARID finns.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överriding villkoren A och B.

Syntax:

Path/BARCHG arid ktonr benamning arstart arslut arlast beskattningsar senverdat vernr ktoplan [databas]

Exempel:

```
$ ./BARCHG SS 2003-01-01-2003-12-31 2003-01-01 2003-12-31 N 2003 0000-00-00  
987124 EUBAS97 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BARCHG arid benamning arstart arslut arlast beskattningsar senverdat vernr  
ktoplan [databas]

Exempel:

```
$ ./STYRMAN jan BARCHG SS 2003-01-01-2003-12-31 2003-01-01 2003-12-31 N 2003 0000-00-00  
987124 EUBAS97 [olfix]
```

SQLsats som används;

```
UPDATE BOKFAR SET BENAMNING = "2003-01-01--2003-12-31",ARSTART = "2003-01-  
01",ARSLUT = "2003-12-31",ARLAST = "N",BESKATTNINGSAR = "2003",SENVERDAT = "2003-03-  
21",VERNR = "234987",KONTOPLAN = "AC" WHERE ARID = "AC"
```

## Interface:

INPUT

ARID, BENAMNING, ARSTART, ARSLUT, ARLAST, BESKATTNINGSAR,  
SENVERDAT, VERNR, KTOPLAN.

OUTPUT

```
fprintf(stderr,"OK: BARCHG Updated %lu rows\n",  
        (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr,"Error: BARCHG No update! Updated %lu rows\n",  
        (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr,"Error: BARCHG: Ange bokföringsår!\n");
```

```
fprintf(stderr, "Error: BARCHG UPDATE error: %d %s\n",  
          mysql_errno(&my_connection), mysql_error(&my_connection));  
fprintf(stderr, "Error: BARCHG Connection failed\n");  
fprintf(stderr, "Error: BARCHG Connection error %d: %s\n",  
          mysql_errno(&my_connection), mysql_error(&my_connection));
```

# BARCHK

Funktionen anropas med parametern ARID.

Kontrollera om bokföringsår ARID finns.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/BARCHK arid [databas]

Exempel:

```
$ ./BARCHK AC [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BARCHK arid [databas]

userid = userid på den som ställer frågan.

arid = det eftersökta bokföringsårets id.

Exempel:

```
$ ./STYRMAN JAPI BARCHK AC [olfix]
```

Returvärde från BARCHK = 0 om arid finns i tabellen BOKFAR annars returneras värdet -1.

Funktionen är tänkt att användas för att kontrollera om bokföringsåret finns.

SQLsats som används;

```
SELECT ARID FROM BOKFAR WHERE ARID = "AC"
```

## Interface:

INPUT

ARID [databas]

OUTPUT

```
printf("Error: BARCHK_SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: BARCHK Retrieval error: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: BARCHK Connection failed\n");
fprintf(stderr,"Error: BARCHK Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"OK: BARCHK Status = %d\n",status);
fprintf(stderr,"Error: BARCHK Bokföringsår %s finns inte!\n",arid);
```



# BARDSP

Funktionen anropas med parametern ARID.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/BARDSP arid [databas]

Exempel:

```
$ ./BARDSP AC [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 BARDSP arid [databas]

Exempel:

```
$ ./STYRMAN JAPI BARDSP AC [olfix]
```

BARDSP skriver ut data på stdout (konsolen) för angivet bokföringsår, arid.

SQLsats som används;

```
SELECT * FROM BOKFAR WHERE ARID = "AC"
```

## Interface:

INPUT:

ARID

OUTPUT:

```
fprintf(stderr, "Error: BARDSP SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "OK: Status = 0 ");
fprintf(stdout, "ARID:%s ", sqlrow[0]);
fprintf(stdout, "BENAMN:%s ", sqlrow[1]);
fprintf(stdout, "ARSTART:%s ", sqlrow[2]);
fprintf(stdout, "ARSLUT:%s ", sqlrow[3]);
fprintf(stdout, "ARLAST:%s ", sqlrow[4]);
fprintf(stdout, "SVERDAT:%s ", sqlrow[5]);
fprintf(stdout, "VERNR:%s ", sqlrow[6]);
fprintf(stdout, "KTOPLAN:%s ", sqlrow[7]);
fprintf(stdout, "BESKATT:%s ", sqlrow[8]);
fprintf(stdout, "\n");
fprintf(stderr, "Error: Status = -1 Bokföringsår saknas!\n");
```



```
fprintf(stderr, "Error: BARDSP Retrieval error: %s\n", mysql_error(&my_connection));  
fprintf(stderr, "Error: BARDSP Connection failed\n");  
fprintf(stderr, "Error: BARDSP Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# BARFND

Funktionen anropas med parametern datum.

BARFND letar reda på ett bokföringsår med ARSTART mindre än **datum** och ARSLUT större än **datum**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Villkor C övertider villkoren A och B.

Syntax:

Path/BARFND datum [databas]

Exempel:

```
$ ./BARFND 2003-06-15 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 BARFND datum [databas]

Exempel:

```
$ ./STYRMAN JAPI BARFND 2003-06-15 [olfix]
```

SQLsats som används;

```
SELECT * FROM BETVILKOR WHERE BETVILKOR = "1"
```

## Interface:

INPUT:

datum

OUTPUT:

```
fprintf(stderr,"Error: BARFND SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"1:%s ",sqlrow[0]);
fprintf(stderr,"Error: BARFND Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: BARFND Connection failed\n");
fprintf(stderr,"Error: BARFND Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr,"Error: BARFND Bokföringsår för datum %s finns inte!\n",datum);
```



# BETADD

Funktionen anropas med parametrarna betvilkor, dagar, beskrivning och som option databas. Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixtst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/BETADD betvilkor dagar beskrivning [databas]

Exempel:

```
$ ./BETADD 1 0 "kontantbetalning" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 BETADD betvilkor dagar beskrivning [databas]

Exempel:

```
$ ./STYRMAN JAPI BETADD 1 0 "kontantbetalning" [olfixtst]
```

BETADD uppdaterar databasen, tabell BETVILKOR.

SQLsats som används;

```
INSERT INTO BETVILKOR(BETVILKOR,DAGAR,BESKRIVNING) VALUES ("001","10","10 dagar netto")
```

## Interface:

INPUT:

BETVILKOR DAGAR BESKRIVNING

OUTPUT:

```
fprintf(stdout,"OK: BETADD Inserted %lu rows\n",(unsigned
long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: BETADD INSERT error: %d %s\n", mysql_errno(&my_connection),
mysql_error(&my_connection));
fprintf(stderr,"Error: BETADD Connection failed\n");
fprintf(stderr,"Error: BETADD Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```



# BETCHG

Funktionen anropas med parametrarna betvilkor, dagar, beskrivning och som option databas. BETCHG uppdaterar databasen, tabell BETVILKOR.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/BETCHG betvilkor dagar beskrivning [databas]

Exempel:

```
$ ./BETCHG 001 0 "kontantbetalning" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BETCHG betvilkor dagar beskrivning [databas]

Exempel:

```
$ ./STYRMAN JAPI BETCHG 001 0 "kontantbetalning" [olfixtst]
```

SQLsats som används;

```
UPDATE BETVILKOR SET DAGAR = "0",BESKRIVNING = "Kontantbetalning" WHERE BETVILKOR = "001"
```

## Interface:

INPUT:

BETVILKOR DAGAR BESKRIVNING

OUTPUT:

```
fprintf(stdout,"OK: BETCHG Updated %lu rows\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stdout,"Error: BETCHG Updated %lu rows\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: BETCHG UPDATE error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: BETCHG Connection failed\n");
fprintf(stderr,"Error: BETCHG Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

## BETDSP

Funktionen anropas med parametern BETVILKOR.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/BETDSP betvilkor [databas]

Exempel:

```
$ ./BETDSP betvilkor [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 BETDSP betvilkor [databas]

Exempel:

```
$ ./STYRMAN JAPI BETDSP BETVILKOR [olfix]
```

BETDSP skriver ut data på stdout (konsolen) för angivet betalningsvillkor, betvilkor.

SQLsats som används;

```
SELECT * FROM BETVILKOR WHERE BETVILKOR = "1"
```

### Interface:

INPUT:

BETVILKOR

OUTPUT:

```
fprintf(stderr,"Error: BETDSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"OK: ");
fprintf(stdout,"BETVILKOR: %s ",sqlrow[0]);
fprintf(stdout,"DAGAR: %s ",sqlrow[1]);
fprintf(stdout,"BESKRIVNING: %s ",sqlrow[2]);
fprintf(stdout,"END:");
fprintf(stdout,"\n");
fprintf(stderr,"Error: BETDSP Betalningsvillkor saknas!\n");
fprintf(stderr,"Error: BETDSP Retrieval error:%s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: BETDSP Connection failed\n");
```

```
fprintf(stderr, "Error: BETDSP Connection error %d: %s\n",  
          mysql_errno(&my_connection), mysql_error(&my_connection));
```



# BETLST

Funktionen anropas utan parametrar eller som option med databas som parameter.  
BETDSP skriver ut data på stdout (konsolen) för alla betalningsvillkor.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixtst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/BETLST [databas]

Exempel:

```
$ ./BETLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 BETLST [databas]

Exempel:

```
$ ./STYRMAN JAPI BETLST [olfix]
```

SQLsats som används;

```
SELECT * FROM BETVILKOR ORDER BY BETVILKOR
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: BETLST SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "OK: NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stdout, "\n");
fprintf(stderr, "Error: BETLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: BETLST Connection failed\n");
fprintf(stderr, "Error: BETLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# DBOKRPT

Funktion för att hämta data till dagboksrapport. Funktionen anropas med parametrarna bar (bokföringsår), fromdatum ( från och med datum) och tomdatum ( till och med datum).

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/DBOKRPT bar fromdatum tomdatum [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid DBOKRPT bar fromdatum tomdatum [databas].

SQLsats som används;

```
SELECT VERHUVUD.VERDATUM, VERRAD.VERNR, VERHUVUD.VERTEXT, VERRAD.KTONR,
      KTOPLAN.BENAMNING, VERRAD.DK, VERRAD.BELOPP
FROM VERRAD
      LEFT JOIN KTOPLAN ON KTOPLAN.KTONR = VERRAD.KTONR AND VERRAD.ARID =
KTOPLAN.ARID
      LEFT JOIN VERHUVUD ON VERRAD.ARID = VERHUVUD.ARID AND
VERRAD.VERNR=VERHUVUD.VERNR
WHERE VERRAD.ARID = "AD"
AND VERHUVUD.VERDATUM >= "2003-06-10"
AND VERHUVUD.VERDATUM <= "2003-10-16"
ORDER BY VERHUVUD.VERDATUM, VERNR, VERRAD.KTONR
```

## Interface:

### INPUT:

BAR, FROMDATUM, TOMDATUM

### OUTPUT:

```
fprintf(stderr, "Error: DBOKRPT SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: DBOKRPT Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: DBOKRPT Connection failed\n");
fprintf(stderr, "Error: DBOKRPT Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout, "END:\n");
```

# FTGADD

Funktionen för att lägga upp en ny posttyp i tabellen FTGDATA och anropas med parametern POSTTYP, POSTBESKR,FDATA

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/FTGADD posttyp postbeskr fdata [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FTGADD posttyp postbeskr fdata [databas].

Uppgifterna i fdata är vid behov separerade med mellanslag, kolon(:) och mellanslag t ex: PROGRAM AB : Box 11 : 199 99 : Progstad

Program som ska hämta data ur fältet FDATA kommer att använda sig av kolonet för att separera data.

SQLsats som används;

```
INSERT INTO FTGDATA (POSTTYP,POSTBESKR,FDATA) VALUES("FTGNR","Företagsnummer", "553411-9555")
```

## Interface:

### INPUT:

POSTTYP,POSTBESKR,FDATA

### OUTPUT:

```
fprintf(stderr,"OK: FTGADD Inserted %lu rows\n",
          (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: FTGADD UPDATE error: %d %s\n",
          mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: FTGADD Connection failed\n");
fprintf(stderr,"Error: FTGADD Connection error %d: %s\n",
          mysql_errno(&my_connection), mysql_error(&my_connection));
```



## FTGDSP

Funktionen hämtar data från tabellen FTGDATA och anropas med parametern POSTTYP och [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/FTGDSP posttyp [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FTGDSP posttyp [databas]

Uppgifterna är vid behov separerade med mellanslag, kolon(:) och mellanslag t ex: PROGRAM AB : Box 11 : 199 99 : Progstad

Program som ska hämta data ur fältet FDATA kommer att använda sig av kolonet för att separera data.

SQLsats som används;

```
SELECT POSTTYP, FDATA FROM FTGDATA WHERE POSTTYP = "ADR1"
```

Posttyp får innehålla max 5 tecken.

Posttyper:

ADR1	Postadress	(Box 9)
ADR2	Postnummer till Postadress	(199 09)
ADR3	Ort till Postadress	(STORSTAD)
ADR4	Besöksadress	(Gågatan 3)
ADR5	Postnr till Besöksadress	(199 08)
ADR6	Ort till Besöksadress	(STORSTAD)
ADR7	Godsadress	(Industrivägen 99)
ADR8	Postnr till Godsadress	(199 99)
ADR9	Ort till Godsadress	(STORSTAD)
AUTOK	Automatkontering J/N	(Grundvärde N)
BF1	Bokföringsperiod 1	(Januari 2002)
BF..	Bokföringsperiod .	(..... 2002)
BF12	Bokföringsperiod 2	(Februari 2002)
BF13	Bokföringsperiod 13	(December 2002)
EML1	E-mailadress	
FNAMN	Företagsnamn	

FTGNR	Företagsnummer/Organisationsnummer
KORNR	Senast använda kundordernummer
MOMS1	Momssats 1
MOMS2	Momssats 2
MOMS3	Momssats 3
MOMS4	Momssats 4
MOMS5	Momssats 5
MOMSI	Momskonto, ingående moms
MOMSU	Momskonto, utgående moms.
TELEX	Telexnummer
TFAX	Telefaxnummer
TFN1	Telefonnummer till vx
TFN2	Mobiltelfonnummer
TFNMB	Mobiltelefonnummer
TFNVX	Telefonnr till vx.

## Interface:

INPUT:

POSTTYP

OUTPUT:

```
fprintf(stderr, "Error: FTGDSP SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "OK: 1:%s 2:%s", sqlrow[0], sqlrow[1]);
fprintf(stderr, "Error: FTGDSP Retrieval error:
%s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: FTGDSP Connection failed\n");
fprintf(stderr, "Error: FTGDSP Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# FTGLIS

Funktionen hämtar data från tabellen FTGDATA och anropas utan parameter.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/FTGLIS [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FTGLIS [databas]

Uppgifterna är vid behov separerade med understreck och kolon(:) t ex: \_: PROGRAM AB \_: Box 11 \_: 199 99 \_: Progstad \_:

Program som ska hämta data ur fältet FDATA kommer att använda sig av understreck och kolonet för att separera data.

SQLsats som används;

```
SELECT POSTTYP,FDATA FROM FTGDATA ORDER BY POSTTYP
```

Posttyp får innehålla max 5 tecken.

Posttyper:

ADR1	Postadress	(Box 9)
ADR2	Postnummer till Postadress	(199 09)
ADR3	Ort till Postadress	(STORSTAD)
ADR4	Besöksadress	(Gågatan 3)
ADR5	Postnr till Besöksadress	(199 08)
ADR6	Ort till Besöksadress	(STORSTAD)
ADR7	Godsadress	(Industrivägen 99)
ADR8	Postnr till Godsadress	(199 99)
ADR9	Ort till Godsadress	(STORSTAD)
AUTOK	Automatkontering J/N	(Grundvärde N)
BF1	Bokföringsperiod 1	(Januari 2002)
BF..	Bokföringsperiod .	(..... 2002)
BF12	Bokföringsperiod 2	(Februari 2002)
BF13	Bokföringsperiod 13	(December 2002)
EML1	E-mailadress	



FNAMN	Företagsnamn
FTGNR	Företagsnummer/Organisationsnummer
KORNR	Senast använda kundordernummer
MOMS1	Momssats 1
MOMS2	Momssats 2
MOMS3	Momssats 3
MOMS4	Momssats 4
MOMS5	Momssats 5
MOMSI	Momskonto, ingående moms
MOMSU	Momskonto, utgående moms.
TELEX	Telexnummer
TFAX	Telefaxnummer
TFN1	Telefonnummer till vx
TFN2	Mobiltelfonnummer
TFNMB	Mobiltelefonnummer
TFNVX	Telefonnr till vx.

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: FTGLIS SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: FTGLIS Retrieval error:
%s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: FTGLIS Connection failed\n");
fprintf(stderr, "Error: FTGLIS Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# FTGLST

Funktion för att lista alla posttyper i tabellen FTGDATA. Anropas utan parametrar eller med optionen [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/FTGLST [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FTGLST [databas].

Uppgifter som hämtas är POSTTYP och POSTBESKR.

SQLsats som används;

SELECT POSTTYP,POSTBESKR FROM FTGDATA ORDER BY POSTTYP

## Interface:

### INPUT:

### OUTPUT:

```
fprintf(stderr,"Error: FTGLST SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: FTGLST Retrieval error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: FTGLST Connection failed\n");
fprintf(stderr,"Error: FTGLST Connection error %d: %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
```

## FTGUPD

Funktionen uppdaterar tabellen FTGDATA och anropas med parametrarna POSTTYP, FDATA [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Villkor C övertider villkoren A och B.

Syntax:

Path/FTGUPD posttyp "fdata" [databas]

Normalt sker uppdateringen via STYRMAN:

Path/STYRMAN userid FTGUPD posttyp "fdata" [databas].

I de fall fdata innehåller flera uppgifter ska uppgifterna separeras med mellanslag, :, mellanslag t ex:

PROGRAM AB : Box 11 : 199 99 : Progstad

Program som ska hämta data ur fältet FDATA kommer att använda sig av kolonet för att separera data.

Program som uppdaterar FDATA skall lägga in mellanslag kolon mellanslag i strängen som ska uppdatera FDATAfältet.

SQLsats som används;

UPDATE FTGDATA SET FDATA = 'txt' WHERE POSTTYP = 'ADR1'

### Interface:

INPUT:

POSTTYP, FDATA

OUTPUT:

```
fprintf(stderr, "OK: FTGUPD Inserted %lu rows\n",
           (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr, "Error: FTGUPD UPDATE error: %d %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr, "Error: FTGUPD Connection failed\n");
fprintf(stderr, "Error: FTGUPD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# HBOKRPT

Funktion för att hämta data till huvudboksrapport. Funktionen anropas med parametrarna bar (bokföringsår), fromdatum ( från och med datum) och tomdatum ( till och med datum).

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Villkor C överrider villkoren A och B.

Syntax:

Path/HBOKRPT bar fromdatum tomdatum [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid HBOKRPT bar fromdatum tomdatum [databas].

SQLsats som används;

```
SELECT
  VERRAD.KTONR, VERHUVUD.VERDATUM, VERRAD.VERNR, VERHUVUD.VERTEX,
  KTOPLAN.BENAMNING, VERRAD.DK, VERRAD.BELOPP
FROM VERRAD
LEFT JOIN KTOPLAN ON KTOPLAN.KTONR = VERRAD.KTONR AND VERRAD.ARID =
KTOPLAN.ARID
LEFT JOIN VERHUVUD ON VERRAD.ARID = VERHUVUD.ARID AND
VERRAD.VERNR=VERHUVUD.VERNR
WHERE VERRAD.ARID = "AD"
AND VERHUVUD.VERDATUM >= "2003-06-10"
AND VERHUVUD.VERDATUM <= "2003-10-16"
ORDER BY KTONR,VERNR
```

## Interface:

### INPUT:

BAR, FROMDATUM, TOMDATUM

### OUTPUT:

```
fprintf(stderr, "Error: HBOKRPT SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: HBOKRPT Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: HBOKRPT Connection failed\n");
fprintf(stderr, "Error: HBOKRPT Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout, "END:\n");
```

# INKADD

Funktionen används för att lägga upp en nya kostnadställen.

Funktionen anropas med parametern inkorderdata.

“inkorderdata” innehåller data om orderhuvud.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/INKADD inkorderdata [databas]

format på inkorderdata

```
_:_6712_:_N_:_2003-12-13_:_9999_:_Testleverantör AB_:_Delivery Street 1C_:_  
199 99_:_LEVSTAD_:_Sverige_:_SEK_:_001_:_Godsmäre_:_002_:_  
Caroline Inköpare_:_2003-12-15_:_98765_:_PROGRAM AB_:_Verktygsgatan 11_:_  
199 97_:_PROGSTAD_:_N_:_1450.50_:_
```

Fältavskiljare = \_:\_

Exempel:

```
$ ./INKADD inkorderdata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid INKADD inkorderdata [databas].

Exempel:

```
$ ./STYRMAN jan INKADD inkorderdata [databas]
```

SQLsats som används:

```
INSERT INTO INKREG (INKORDNR,BESTTYP,ORDERDATUM,LEVNR,LEVNAMN,LEVADDRESS,  
LEVPOSTNR,LEVPOSTADR,LEVLAND,LEVVALUTA,LEVBETVILLKOR,GODSMERKE,BESTTEXT,  
VARREF,LEV DATUM,KUNDNR,FTGNAMN,FTGADR,FTGPOSTNR,FTGPOSTADR,ORDERSTATUS,  
ORDERSUMMA) VALUES  
("6712","N","2003-12-13","Testleverantör AB","Delivery Street 1C",  
"199 99","LEVSTAD","Sverige","SEK","001","Godsmärke","002",  
"Caroline Inköpare","2003-12-15","98765","PROGRAM AB","Verktygsgatan 11",  
"199 97","PROGSTAD","N","1450.50")
```

## Interface:

### INPUT:

INKORDNR,BESTTYP,ORDERDATUM,LEVNR,LEVNAMN,LEVADDRESS,  
LEVPOSTNR,LEVPOSTADR,LEVLAND,LEVVALUTA,LEVBETVILLKOR,GODSMERKE,BESTTEXT,  
VARREF,LEVDATUM,KUNDNR,FTGNAMN,FTGADR,FTGPOSTNR,FTGPOSTADR,ORDERSTATUS,  
ORDERSUMMA

### OUTPUT:

```
fprintf(stderr,"Error: INKADD: Ange inköpsordernummer!\n");  
fprintf(stdout,"OK: INKADD Inserted %lu rows\n",  
        (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr,"Error: INKADD INSERT error: %d %s\n",  
        mysql_errno(&my_connection),mysql_error(&my_connection));  
fprintf(stderr,"Error: INKADD Connection failed\n");  
fprintf(stderr,"Error: INKADD Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# INKLST

Funktionen används för att visa alla inköpsorderrader, beställningsstock.  
Funktionen anropas utan parametrar.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/INKLST [databas]

Exempel:

```
$ ./INKLST [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid INKLST [databas].

Exempel:

```
$ ./STYRMAN jan INKLST [databas]
```

SQLsats som används:

```
SELECT
    INKRADREG.INKORDNR, INKORDRADNR, INKREG.LEVNR,
    INKRADREG.ARTIKELNR, ARBENEMNING1, BEKREFTKOD,
    BESTANTAL, LEVERERAT, RESTNOTERAT, INKPRIS, LEVVECKA,
    RESTNOTERAT * INKPRIS RADSUM
from
    INKRADREG, INKREG, ARTIKELREG
WHERE
    INKREG.INKORDNR = INKRADREG.INKORDNR AND
    INKRADREG.ARTIKELNR = ARTIKELREG.ARTIKELNR AND
    RESTNOTERAT > \"0\"
ORDER BY
    INKORDNR, INKORDRADNR
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: INKLST SELECT errno: %d\n",
mysql_errno(&my_connection));
fprintf(stderr,"Error: INKLST Retrieval error:  %s\n",
        mysql_error(&my_connection));
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: INKLST Connection failed\n");
fprintf(stderr,"Error: INKLST Connection error %d:  %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# INKRADD

Funktionen används för att lägga upp en nya inköpsorderrader.  
Funktionen anropas med parametern inkorderraddata.  
“inkorderraddata” innehåller data om orderrad.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/INKRADD inkorderraddata [databas]

format på inkorderraddata

\_:\_6712\_:\_010\_:\_1173-1445\_:\_ST\_:\_25.00\_:\_0\_:\_0\_:\_95.00\_:\_351\_:\_0\_:\_0\_:\_

Fältavskiljare = \_:\_

Exempel:

```
$ ./INKRADD inkorderraddata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid INKRADD inkorderraddata [databas].

Exempel:

```
$ ./STYRMAN jan INKRADD inkorderraddata [databas]
```

SQLsats som används:

```
INSERT INTO INKRADREG(INKORDNR,INKORDRADNR,ARTIKELNR,ENHET,BESTANTAL,
LEVERERAT,RESTNOTERAT,INKPRIS,LEVVECKA,TORDNR,OPNR) VALUES ( "6712", "010",
1445, "ST", "25.00", "0", "0", "95.00", "351", "0", "0" ) "1173-
```

## Interface:

### INPUT:

INKORDNR,INKORDRADNR,ARTIKELNR,ENHET,BESTANTAL,LEVERERAT,  
RESTNOTERAT,INKPRIS,LEVVECKA,TORDNR,OPNR

### OUTPUT:

```
fprintf(stderr,"Error: INKRADD: Ange inköpsordernummer!\n");  
fprintf(stdout,"OK: INKRADD Inserted %lu rows\n",  
        (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr,"Error: INKRADD INSERT error: %d %s\n",  
        mysql_errno(&my_connection),mysql_error(&my_connection));  
fprintf(stderr,"Error: INKRADD Connection failed\n");  
fprintf(stderr,"Error: INKRADD Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## INKHDSP

Funktionen används för att visa huvudet på angiven inköpsorder.  
Funktionen anropas med parametern inköpsordernr.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/INKHDSP inkordernr [databas]

Exempel:

```
$ ./INKHDSP inkordnr [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid INKHDSP inkordnr [databas].

Exempel:

```
$ ./STYRMAN jan INKHDSP inkordnr [databas]
```

SQLsats som används:

```
SELECT * FROM INKREG WHERE INKORDNR = 99999
```

## Interface:

### INPUT:

INKORDNR

### OUTPUT:

```
fprintf(stderr,"Error: Inköpsordernr saknas!\n");
fprintf(stderr,"Error: INKHDSP SELECT errno: %d\n",
        mysql_errno(&my_connection));
fprintf(stdout,"OK: ");
fprintf(stdout,"01:%s ",sqlrow[0]); /* beställningsordernr */
fprintf(stdout,"02:%s ",sqlrow[1]); /* beställningstyp */
fprintf(stdout,"03:%s ",sqlrow[2]); /* beställningsdatum */
fprintf(stdout,"04:%s ",sqlrow[3]); /* leverantörsnr */
fprintf(stdout,"05:%s ",sqlrow[4]); /* levnamn */
fprintf(stdout,"06:%s ",sqlrow[5]); /* leverantörsadress */
fprintf(stdout,"07:%s ",sqlrow[6]); /* leverantörspostnr */
fprintf(stdout,"08:%s ",sqlrow[7]); /* leverantörspostadr */
fprintf(stdout,"09:%s ",sqlrow[8]); /* leverantörsland */
fprintf(stdout,"10:%s ",sqlrow[9]); /* valuta */
fprintf(stdout,"11:%s ",sqlrow[10]); /* betalningsvillkor */
fprintf(stdout,"12:%s ",sqlrow[11]); /* leveransvillkor */
fprintf(stdout,"13:%s ",sqlrow[12]); /* leveranssätt */
fprintf(stdout,"14:%s ",sqlrow[13]); /* godsmärke */
fprintf(stdout,"15:%s ",sqlrow[14]); /* kommentar */
fprintf(stdout,"16:%s ",sqlrow[15]); /* beställningseftertext */
fprintf(stdout,"17:%s ",sqlrow[16]); /* varref */
fprintf(stdout,"18:%s ",sqlrow[17]); /* varreftn */
fprintf(stdout,"19:%s ",sqlrow[18]); /* varreffax */
fprintf(stdout,"20:%s ",sqlrow[19]); /* erref */
fprintf(stdout,"21:%s ",sqlrow[20]); /* leveransdatum */
fprintf(stdout,"22:%s ",sqlrow[21]); /* kundnr */
fprintf(stdout,"23:%s ",sqlrow[22]); /* ftgnamn */
fprintf(stdout,"24:%s ",sqlrow[23]); /* ftglevadr */
fprintf(stdout,"25:%s ",sqlrow[24]); /* ftglevpostnr */
fprintf(stdout,"26:%s ",sqlrow[25]); /* ftglevpostadr */
fprintf(stdout,"27:%s ",sqlrow[26]); /* språkkod */
fprintf(stdout,"28:%s ",sqlrow[27]); /* bekräftelsekod */
fprintf(stdout,"29:%s ",sqlrow[28]); /* orderstatus */
fprintf(stdout,"30:%s ",sqlrow[29]); /* utskriftskod */
fprintf(stdout,"31:%s ",sqlrow[30]); /* ordersumma */
fprintf(stdout,"END:");
fprintf(stdout,"\n");
fprintf(stderr,"Error: INKHDSP Data saknas: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: INKHDSP Retrieval error: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: INKHDSP Connection failed\n");
fprintf(stderr,"Error: INKHDSP Connection error %d: %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
```



# INKRLST

Funktionen används för att visa alla rader på angiven inköpsorder.  
Funktionen anropas med parametern inköpsordernr.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/INKRLST inkordnr [databas]

Exempel:

```
$ ./INKRLST inkordnr [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid INKRLST inkordnr [databas].

Exempel:

```
$ ./STYRMAN jan INKRLST inkordnr [databas]
```

SQLsats som används:

```
SELECT INKORDNR,INKORDRADNR,ARTIKELNR,ENHET,BESTANTAL,LEVERERAT,
RESTNOTERAT,INKPRIS,LEVVECKA,TORDNR,OPNR,BENEMNING
from INKRADREG
WHERE INKORDNR = 99999 ORDER BY INKORDRADNR
```

## Interface:

INPUT:

INKORDNR

OUTPUT:

```
fprintf(stderr,"Error: Inköpsordernr saknas!\n");
fprintf(stderr,"Error: INKRLST SELECT errno: %d\n",
        mysql_errno(&my_connection));
fprintf(stderr,"Error: INKRLST Retrieval error: %s\n",
        mysql_error(&my_connection));
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: INKRLST Connection failed\n");
fprintf(stderr,"Error: INKRLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KSTADD

Funktionen används för att lägga upp en nya kostnadställen.

Funktionen anropas med parametrarna ARID, KSTALLE, BENAMNING.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KSTADD arid kstalle benamning [databas]

Exempel:

```
$ ./KSTADD ad 9037 "Projekt Omega" [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KSTADD arid kstalle benamning [databas].

Exempel:

```
$ ./STYRMAN jan KSTADD ad 9037 "Projekt Omega"
```

SQLsats som används:

```
INSERT INTO KSTALLE (ARID,KSTALLE,BENAMNING VALUES ("AD","9037","Projekt Omega")
```

## Interface:

INPUT:

ARID, KSTALLE, och BENAMNING

OUTPUT:

```
fprintf(stdout,"OK: KSTADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: KSTADD INSERT error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));

fprintf(stderr,"Error: KSTADD Connection failed\n");
fprintf(stderr,"Error: KSTADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```





# KSTCHK

Funktionen används för att kontrollera om ett kostnadsställe finns.  
Funktionen anropas med parametrarna ARID, KSTALLE.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/KSTCHK arid kstalle [databas].

Exempel:

```
$ ./KSTADD AD 9037 olfix
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KSTCHK arid kstalle [databas].

Exempel:

```
$ ./STYRMAN jan KSTCHK AD 9037 olfix
```

SQLsats som används:

```
SELECT * FROM KSTALLE WHERE ARID = "AD" AND KSTALLE = 9037"
```

## Interface:

INPUT:

ARID KSTALLE

OUTPUT:

```
fprintf(stderr,"Error: KSTCHK SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: KSTCHK Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KSTCHK Connection failed\n");
fprintf(stderr,"Error: KSTCHK Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"OK: KSTCHK Status = %d\n",status);
fprintf(stderr,"Error: KSTCHK Kostnadställe %s finns inte!\n",kstalle);
```

# KSTDSP

Funktionen används för att visa information om ett kostnadställe.  
Funktionen anropas med parametrarna ARID, KSTALLE.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parameter [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixtst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/KSTDSP arid kstalle [databas].

Exempel:

```
$ ./KSTDSP ad 9037 [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KSTDSP arid kstalle [databas].

Exempel:

```
$ ./STYRMAN jan KSTDSP ad 9037 [olfixtst]
```

SQLsats som används:

```
SELECT * FROM KSTALLE WHERE (ARID = "AC" AND KSTALLE = "9037")
```

## Interface:

INPUT:

ARID, KSTALLE

OUTPUT:

```
fprintf(stderr, "Error: KSTDSP_SELECT error: %s\n", mysql_error(&my_connection));  
fprintf(stdout, "1:%s  ", sqlrow[0]);  
fprintf(stdout, "2:%s  ", sqlrow[1]);  
fprintf(stdout, "3:%s  ", sqlrow[2]);  
fprintf(stdout, "\n");  
fprintf(stderr, "Error: KSTDSP_Retrieval error: %s\n",  
mysql_error(&my_connection));  
fprintf(stderr, "Error: KSTDSP_Connection failed\n");  
fprintf(stderr, "Error: KSTDSP_Connection error %d: %s\n",  
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KSTLST

Funktionen används för att lista information om alla kostnadställen.

Funktionen anropas utan parametrar.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KSTLST [databas].

Exempel:

```
$ ./KSTLST [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KSTLST [databas].

Exempel:

```
$ ./STYRMAN jan KSTLST [olfixtst]
```

SQLsats som används:

```
SELECT * FROM KSTALLE ORDER BY ARID
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: KSTLST SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: KSTLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: KSTLST Connection failed\n");
fprintf(stderr, "Error: KSTLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



## KTOADD

Funktionen används för att lägga upp en nya konton.

Funktionen anropas med parametrarna ARID, KTONR, BENAMNING, MANUELL, MOMSKOD,SRUNR, KSTALLE, PROJEKT, SUBKTO, KTOPLAN [databas]. IB och UB sätts med automatik till "0.00".

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTOADD arid ktonr benamning manuell momskod srunr kstalle projekt subkto ktoplan [databas].

Exempel:

```
$ ./KTOADD ss 1050 Testkonto J 1 100 2000 3000 4000 EUBAS97 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTOADD arid ktonr benamning manuell momskod srunr kstalle projekt subkto ktoplan [databas].

Exempel:

```
$ ./STYRMAN jan KTOADD ss 1050 Testkonto J 1 100 2000 3000 4000 EUBAS97 [olfixtst]
```

SQLsats som används:

```
INSERT INTO KTOPLAN
(ARID,KTONR,BENAMNING,MANUELL,MOMSKOD,SRUNR,KSTALLE,PROJEKT,SUBKTO,
KTOPLAN,IB,IB) VALUES
("AC","1200","Testkonto","J","1","100","2000","3000","4000",
"EUBAS97","0.00","0.00")
```

### Interface:

INPUT:

ARID, KTONR, BENAMNING, MANUELL, MOMSKOD, SRUNR, KSTALLE, PROJEKT, SUBKTO, KTOPLAN.

OUTPUT:

```
fprintf(stdout,"OK: KTOADD Inserted %lu rows\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: KTOADD INSERT error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: KTOADD Connection failed\n");
fprintf(stderr,"Error: KTOADD Connection error %d: %s\n",
```

```
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KTOCHG

Funktionen anropas med parametrarna ARID,KTONR,BENAMNING,MANUELL,MOMSKOD SRUNR,KSTALLE,PROJEKT,SUBKTO,KTOPLAN.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTOCHG arid ktonr benamning manuell momskod srunr kstalle projekt subkto ktoplan [databas]

Exempel:

```
$ ./KTOCHK AC 1020 Postgiro N 1 100 2000 3000 4000 EUBAS99 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTOCHG arid ktonr benamning manuell momskod srunr kstalle projekt subkto ktoplan [databas]

userid = userid på den som ställer frågan.

Exempel:

```
$ ./STYRMAN JAPI KTOCHK AC 1020 Postgiro N 1 100 2000 3000 4000 EUBAS99 [olfixtst]
```

Returvärde från KTOCHG = OK:

SQLsats som används;

```
UPDATE KTOPLAN SET BENAMNING = 'Postgiro',MANUELL = 'N',MOMSKOD = '1',SRUNR = '100',KSTALLE = '2000',PROJEKT = '3000',SUBKTO = '4000',KTOPLAN = 'EUBAS99' WHERE (ARID = 'AC' AND KTONR = '1020')
```

## Interface:

INPUT:

ARID, KTONR, BENAMNING, MANUELL, MOMSKOD, SRUNR, KSTALLE, PROJEKT,SUBKTO, KTOPLAN

## OUTPUT:

```
fprintf(stdout,"OK: KTOCHG Updated %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: KTOCHG UPDATE error: %d %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: KTOCHG Connection failed\n");
fprintf(stderr,"Error: KTOCHG Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# KTOCHK

Funktionen anropas med parametern KTONR [databas]

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTOCHK ktonr [databas]

Exempel:

```
$ ./KTOCHK 1020 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTOCHK ktonr [databas]

userid = userid på den som ställer frågan.

ktonr = det eftersökta kontonummret.

Exempel:

```
$ ./STYRMAN JAPI KTOCHK 1020 [olfixtst]
```

Returvärde från KTOCHK = 0 om ktonr finns i tabellen KONTONR annars returneras värdet -1.  
Funktionen är tänkt att användas för att kontrollera om kontonummer finns.

SQLsats som används;

```
SELECT KTONR FROM KONTONR WHERE KTONR = "1020"
```

## Interface:

INPUT:

KTONR

OUTPUT:

```
fprintf(stderr,"Error: KTOCHK Selection error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KTOCHK Retrieve error  %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: KTOCHK Connection failed\n");
fprintf(stderr,"Error: KTOCHK Connection error: %d  %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stdout,"OK: KTOCHK Status = %d\n",status);
fprintf(stderr,"Error: KTOCHK Konto %s finns inte!\n",ktonr);
```

## KTODSP

Funktionen anropas med parametrarna ARID, KTONR och [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/KTODSP arid ktonr [databas]

Exempel:

```
$ ./KTOCHK AC 1020 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTODSP arid ktonr [databas]

userid = userid på den som ställer frågan.

arid = bokföringsår

ktonr = det eftersökta kontonummret.

Exempel:

```
$ ./STYRMAN JAPI KTODSP AC 1020 [olfixtst]
```

Returvärde från KTODSP = data om aktuellt ktonr eller felmeddelande.

SQLsats som används;

```
SELECT * FROM KTOPLAN WHERE (ARID = "AC" AND KTONR = "1020")
```

### Interface:

INPUT:

ARID och KTONR (Bokföringsår och kontonummer, primary key)

## OUTPUT:

```
fprintf(stderr,"Error: KTODSP SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: KTODSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: KTODSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KTODSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KTODSP Connection failed\n");
fprintf(stderr,"Error: KTODSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"1:%s ",sqlrow[0]);
fprintf(stdout,"2:%s ",sqlrow[1]);
fprintf(stdout,"3:%s ",sqlrow[2]);
fprintf(stdout,"4:%s ",sqlrow[3]);
fprintf(stdout,"5:%s ",sqlrow[4]);
fprintf(stdout,"6:%s ",sqlrow[5]);
fprintf(stdout,"7:%s ",sqlrow[6]);
fprintf(stdout,"8:%s ",sqlrow[7]);
fprintf(stdout,"9:%s ",sqlrow[8]);
fprintf(stdout,"10:%s ",sqlrow[9]);
fprintf(stdout,"11:%s ",sqlrow[10]);
fprintf(stdout,"12:%s ",sqlrow[11]);
fprintf(stdout,"\n");
```

# KTOLST

Funktionen anropas utan parameter eller med optionen [databas].

KTOLST listar alla kontonummer i tabellen KTOPLAN , sorterad per bokföringsår och kontonummer.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTOLST [databas]

Exempel:

```
$ ./KTOLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTOLST [databas]

Exempel:

```
$ ./STYRMAN olfix KTOLST [olfixtst]
```

Funktionen används för att visa information om kontona, kontonummer och kontotext.

SQLsats som används:

```
SELECT * FROM KTOPLAN ORDER BY ARID, KTONR
```

Utdata från KTOLST är en enda lång sträng med '\_' (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med 'OK: NR\_' postantal '\_' (underscore).

Exempel på hur man kan dela up fält och poster finns i konsolprogrammet REDOVs funktion kontoList.

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: KTOLST SELECT error: %s\n", mysql_error(&my_connection));  
fprintf(stdout, "OK: NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));  
fprintf(stdout, "%s:", sqlrow[field_count]);  
fprintf(stderr, "Error: KTOLST Retrieval error: %s\n", mysql_error(&my_connection));  
fprintf(stderr, "Error: KTOLST Connection failed\n");  
fprintf(stderr, "Error: KTOLST Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KTORPT

Funktionen anropas med parametern arid, fromdatum, tomdatum och [databas]

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTORPT arid fromdatum tomdatum [databas]

Exempel:

```
$ ./KTORPT ARID fromdatum tomdatum[olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTORPT arid fromdatum tomdatum [databas]

Exempel:

```
$ ./STYRMAN olfix KTORPT ARID fromdatum tomdatum[olfixst]
```

Funktionen används för att hämta KTONR, DK och BELOPP från tabellen VERRAD.

SQLsats som används:

```
SELECT VERRAD.KTONR,KTOPLAN.BENAMNING,VERRAD.DK,VERRAD.BELOPP,VERRAD.VERNR FROM
VERRAD
LEFT JOIN KTOPLAN ON KTOPLAN.KTONR = VERRAD.KTONR AND VERRAD.ARID = KTOPLAN.ARID
LEFT JOIN VERHUVUD ON VERRAD.ARID = VERHUVUD.ARID AND VERRAD.VERNR=VERHUVUD.VERNR
WHERE VERRAD.ARID = "AD"
AND VERHUVUD.VERDATUM >= "2003-08-10"
AND VERHUVUD.VERDATUM <= "2003-08-16"
ORDER BY KTONR
```

Utdata från KTORPT är en enda lång sträng med '\_' (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med 'NR\_' postantal '\_' (underscore).

## Interface:

INPUT:

ARID fromdatum tomdatum

OUTPUT:

```
fprintf(stderr, "Error: KTORPT SELECT error: %s\n", mysql_error(&my_connection));  
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));  
fprintf(stdout, "%s:", sqlrow[field_count]);  
fprintf(stderr, "Error: KTORPT Retrieval error: %s\n", mysql_error(&my_connection));  
fprintf(stderr, "Error: KTORPT Connection failed\n");  
fprintf(stderr, "Error: KTORPT Connection error %d: %s\n",  
mysql_errno(&my_connection), mysql_error(&my_connection));
```

## KTOUPD

**OBS!!! Denna funktion ska få ny inriktning. OBS!**

**Nuvarande funktionallitet gäller ej.**

Funktionen anropas med parametrarna KONTONR, DK, BELOPP.

Syntax:

Path/KTOUPD kontonummer y belopp

där y kan vara antingen D eller K.

Funktionen adderar BELOPP till ettdera av fälten KTODB och KTOKR beroende på vilket värde DK har, om det är en debetpost eller en kreditpost.

Om DK = "D" adderas beloppet till värdet i KTODB, och om DK = "K" så adderas beloppet till KTOKR.

SQLsats som används;

```
UPDATE KONTONR SET KTODB = KTODB + BELOPP
```

```
UPDATE KONTONR SET KTOKR = KTOKR + BELOPP
```

Därefter uppdaterar KTOUPD tabellen TRHD genom att anropa funktionen TRHDADD med c-funktionen **execl** och parametrarna:

TRHDADD TRHDADD KTOUPD "tid" "olfix" TRNSDATA

**tid** har formatet YYYY-MM-DD\_hh:mm:ss och hämtas med c-funktionen **strftime**.

**olfix** är defaultuser och används för kommunikation mot databasen.

TRHDDATA innehåller "KTONR ; dk ; BELOPP"

**dk** innehåller ett dera av **D** eller **K**.

# KTOVIEW

Funktionen är avsedd att visa/lista kontonummer på skärm som hjälp åt användaren att hitta rätt kontonummer eller vad ett kontonummer är avsett att användas till.

Funktionen anropas med parametern **arid** och [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KTOVIEW arid [databas]

Exempel:

```
$ ./KTOVIEW 2002 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KTOVIEW arid [databas]

Exempel:

```
$ ./STYRMAN olfix KTOVIEW 2002 [olfixtst]
```

Funktionen används för att visa information om kontona, kontonummer och kontotext.

SQLsats som används:

```
SELECT KTONR, BENAMNING FROM KTOPLAN WHERE ARID="2002" ORDER BY KTONR
```

Utdata från KTOVIEW är en enda lång sträng med '.\_:' (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med 'NR\_' postantal '.\_' (underscore).

Exempel på hur man kan dela up fält och poster finns i konsolprogrammet REDOVs funktion kontoList.

## Interface:

INPUT:

ARID            (Bokföringsår)



## OUTPUT:

```
fprintf(stderr,"Error: KTOVIEW ARID Ange årtal!\n");
fprintf(stderr,"Error: KTOVIEW SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: KTOVIEW Felaktigt årtal!\n");
fprintf(stderr,"Error: KTOVIEW Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KTOVIEW Connection failed\n");
fprintf(stderr,"Error: KTOVIEW Connection error %d: %s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stdout,"%s_",sqlrow[field_count]);
```

## KUADD

Funktionen används för att lägga upp nya kundposter i tabellen KUNDREG. Programmet känner av vem som är inloggad från envirovariabeln USER.

Funktionen anropas med parametern **kunddata** och som option, **databas**. Som fältavskiljare används **\_:\_**. Programmet spjälkar sedan upp fälten och matar in dem i sqlsatsen. Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/KUADD kunddata [databas]

Exempel:

```
$ ./KUADD _:_ "4376" _:_ "Test AB" _:_ "Provgatan 2" _:_ "199
99" _:_ "LILLEBY" _:_ "Sverige" _:_ "09-999990" _:_ "09-999999" _:_ "info@test.se" _:_ "Karl
Andersson" _:_ "09-999991" _:_ "karl.a@test.se" _:_ "Caroline
Seljare" _:_ "Kalmar" _:_ "Software" _:_ "001" _:_ "001" _:_ "001" _:_ "1" _:_ "SEK" _:_ "sv" _:_ J _:_ J _:_
J _:_ J _:_ J _:_ J _:_ 2000 _:_ J _:_ J _:_ "Fritt textfält" _:_ olfixtst
```

OBS! Mellanslag före olfixtst.

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KUADD kunddata [databas]

Exempel:

```
$ ./STYRMAN jan KUADD _:_ "4376" _:_ "Test AB" _:_ "Provgatan 2" _:_ "199
99" _:_ "LILLEBY" _:_ "Sverige" _:_ "09-999990" _:_ "09-999999" _:_ "info@test.se" _:_ "Karl
Andersson" _:_ "09-999991" _:_ "karl.a@test.se" _:_ "Caroline
Seljare" _:_ "Kalmar" _:_ "Software" _:_ "001" _:_ "001" _:_ "001" _:_ "1" _:_ "SEK" _:_ "sv" _:_ J _:_ J _:_
J _:_ J _:_ J _:_ J _:_ 2000 _:_ J _:_ J _:_ "Fritt textfält" _:_ olfixtst
```

OBS! Mellanslag före olfixtst.

SQLsats som används:

```
INSERT INTO KUNDREG
(KUNDNR,NAMN,ADRESS,POSTNR,POSTADR,LAND,TFNNR,FAXNR,EMAILADR,ERREFERENT,ERREFTFNNR,
ERREFEMAIL,SELJARE,DISTRIKT,KUNDKATEGORI,STDLEVPLATS,LEVVILLKOR,LEVSETT,BETALVILLKOR,
VALUTA,SPRAKKOD,ORDERERKENNANDE,PLOCKLISTA,FOLJESEDEL,EXPAVGIFT,FRAKTAVG,KRAVBREV,
KREDITLIMIT,DROJMALSRTA,DROJMALSFAKTURA,FRITEXT) VALUES ("4376","Test AB","Provgatan 2","199
99","LILLEBY","Sverige","09-999990","09-999999","info@test.se","Karl Andersson","09-999991","karl.a@test.se","Caroline
Seljare","Kalmar","Software","001","001","001","1","SEK","sv","J","J","J","J","J","J","2000","J","J","Fritt textfält",)
```

Utdata från KUADD är OK: KUADD Inserted %lu rows

## Interface:

INPUT:

KUNDDATA [databas]

OUTPUT:

```
fprintf(stdout,"OK: KUADD Inserted %lu rows\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: KUADD INSERT error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: KUADD Connection failed\n");
fprintf(stderr,"Error: KUADD Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KUCHG

Funktionen används för att ändra kunddata i tabellen KUNDREG.

Funktionen anropas med parametern **kunddata** och som option, **databas** .

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

**Kunddatas** format =

```
_:_4376_:_Test AB_:_Provgatan 2_:_199 99_:_LILLEBY_:_Sverige_:_  
09-999990_:_09-999999_:_info@test.se_:_Karl Andersson_:_09-999991_:_  
karl.a@test.se_:_Caroline Seljare_:_KalmarSoftware_:_001_:_001_:_001_:_1_:_  
SEK_:_sv_:_J_:_J_:_J_:_J_:_J_:_J_:_2000_:_J_:_J_:_Fritt textfält_:_  
30_:_001_:_001_:_001_:_001_:_J_:_
```

\_:\_ = fältavskiljare.

Ordningen på data är **synerligen** viktig.

Ordningen **skall** vara:

NAMN, POSTNR, POSTADR, LAND, TFNNR, FAXNR, EMAILADR, ERREFERENT,  
ERREFTFNRR, ERREFEMAIL, SELJARE, DISTRIKT, KUNDKATEGORI,  
STDLEVPLATS, LEVVILLKOR, LEVSETT, BETALVILLKOR, VALUTA, SPRAKKOD,  
ORDERERKENNANDE, PLOCKLISTA, FOLJESDEL, EXPAVGIFT, FRAKTAVG,  
KRAVBREV, KREDITLIMIT, DROJMALSRTA,DROJMALSFAKTURA, FRITEXT,  
KREDITDAGAR, KREDITKOD, EXPORTKOD, SKATTEKOD, RABATTKOD,  
SAMLINGSFAKT

Syntax:

Path/KUCHG kunddata [databas]

Exempel:

```
$ ./KUCHG kunddata [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KUCHG kunddata [databas]

Exempel:

```
$ ./STYRMAN olfix KUCHG kunddata [olfixtst]
```

## SQLsats som används:

```
UPDATE KUNDREG SET
    NAMN="TestAB",ADRESS="Provgatan 2",
    POSTNR="199 99",POSTADR="LILLEBY",LAND="Sverige",TFNNR="09-999990",
    FAXNR="09-999999",EMAILADR="info@test.se",
    ERREFERENT="Karl Andersson",ERREFTFNNR="09-999991",
    ERREFEMAIL="karl.a@test.se",SELJARE="Caroline Seljare",DISTRICKT="Kal",
    KUNDKATEGORI="Sto",STDLEVPLATS="002",LEVILLKOR="001",LEVSETT="001",
    BETALVILLKOR="001",VALUTA="SEK",SPRAKKOD="sv",
    ORDERERKENNANDE="J",PLOCKLISTA="J",FOLJESEDEL="J",EXPAVGIFT="J",
    FRAKTAVG="J",KRAVBREV="J",KREDITLIMIT="2000",DROJMALSRTA="J",
    DROJMALSFAKTURA="J",FRITEXT="Fritt textfält",KREDITDAGAR="30",
    KREDITKOD="001",EXPORTKOD="001",SKATTEKOD="001",RABATTKOD="001",
    SAMLINGSFAKT="J"
WHERE KUNDNR="kundnr"
```

## Interface:

### INPUT:

KUNDDATA [databas]

### OUTPUT:

```
fprintf(stderr,"Error: KUCHG UPDATE error: %d %s\n",mysql_error
    (&my_connection));
fprintf(stderr,"Error: KUCHG Connection failed\n");
fprintf(stderr,"Error: KUCHG Connection error %d: %s\n",
    mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"OK: KUCHG Uppdated %lu rows\n",status);
```

## KUCHK

Funktionen används för att kontrollera om angivet kundid finns i tabellen KUNDREG.  
Funktionen anropas med parametern **kundnr** och som option, **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KUCHK kundnr [databas]

Exempel:

```
$ ./KUCHK 12345678 [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KUCHK kundnr [databas].

Exempel:

```
$ ./STYRMAN olfix KUCHK 12345678 [databas]
```

SQLsats som används:

```
SELECT KUNDNR FROM KUNDREG WHERE KUNDNR = "12345678"
```

## Interface:

INPUT:

KUNDNR

OUTPUT:

```
fprintf(stderr,"Error: KUCHK_Retrieve error:  %s\n",mysql_error
        (&my_connection));
fprintf(stderr,"Error: KUCHK_Connection failed\n");
fprintf(stderr,"Error: KUCHK_Connection error %d:  %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"OK: KUCHK_Status = %d\n",status);
fprintf(stderr,"Error: KUCHK Kundnr %s finns inte!\n",kundnr);
```



## KUDSP

Funktionen används för att visa kunddata för angiven kund i tabellen KUNDREG.  
Funktionen anropas med parametern **kundnr** och som option, **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/KUDSP kundnr [databas]

Exempel:

```
$ ./KUDSP 12345678 [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KUDSP kundnr [databas].

Exempel:

```
$ ./STYRMAN olfix KUDSP 12345678 [databas]
```

SQLsats som används:

```
SELECT * FROM KUNDREG WHERE (KUNDNR = "12345678")
```

### Interface:

INPUT: KUNDNR

OUTPUT: 41 st fält,

KUNDNR, KUNDORGNR, NAMN, ADRESS, POSTNR, POSTADR, LAND, TFNNR,  
EMAILADR, FAXNR,ERREFERENT, ERREFTFNNR, ERREFEMAIL, SELJARE  
FRITEXT, VALUTA, BETALVILLKOR,LEVILLKOR, LEVSETT, DISTRIKT,  
KUNDKATEGORI, STDLEVPLATS, ORDERERKENNANDE,PLOCKLISTA,  
FOLJESEDEL, KRAVBREV, SPRAKKOD, EXPAVGIFT, FRAKTAVG, KREDITLIMIT,  
KREDITDAGAR, KREDITKOD, EXPORTKOD, SKATTEKOD, RABATTKOD,  
DROJSMALSRTA,DROJSMALSFAKTURA, SAMLINGSFAKT,  
SENASTEKRAVDATUM, SKULD, ORDERSTOCK



samt errornb och error (text)

Detta är också turordningen som data hämtas och skickas.

```
fprintf(stderr,"Error: KUDSP SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stdout,"OK: ");
fprintf(stdout,"01:%s ",sqlrow[0]);
fprintf(stdout,"02:%s ",sqlrow[1]);
fprintf(stdout,"03:%s ",sqlrow[2]);
fprintf(stdout,".....",sqlrow[....]);
fprintf(stdout,"41:%s ",sqlrow[40]);
fprintf(stdout,"\n");
fprintf(stderr,"Error: KUDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KUDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: KUDSP Connection failed\n");
fprintf(stderr,"Error: KUDSP Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

# KULST

Funktionen används för att lista kunder ur tabellen KUNDREG, kundnr och namn.  
Funktionen anropas utan paramtrar eller som option, **databas**

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/KULST [databas]

Exempel:

```
$ ./KULST [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:  
Path/STYRMAN userid KULST [databas].

Exempel:

```
$ ./STYRMAN olfix KULST [databas]
```

SQLsats som används:

```
SELECT KUNDNR,NAMN FROM KUNDREG ORDER BY NAMN
```

## Interface:

INPUT:

OUTPUT: KUNDNR, NAMN

```
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);

fprintf(stderr,"Error: KULST SELECT errno: %d\n",
            mysql_errno(&my_connection));
fprintf(stderr,"Error: KULST Retrieval error: %s\n",
            mysql_error(&my_connection));
fprintf(stderr,"Error: KULST Connection failed\n");
fprintf(stderr,"Error: KULST Connection error %d: %s\n",
            mysql_errno(&my_connection), mysql_error(&my_connection));
```

# LEVADD

Funktion för att registrera information på en leverantör.

Funktionen anropas med parametern **levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betalvilkor** och som option **databas**.

Levskuld sätts till 0.00 kr.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVADD levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto [databas]

Exempel:

```
$ ./LEVADD 123 "559999-9999" "Leverantör AB" "Postgatan 33" "199 99" "DATABY"
"SVERIGE" "09-999999" "09-999998" "99999" "kuntj@leverantor.se" "Per Josefsson"
"09-999997" "1" "0.00" "2110" "12345678" "SEK" "2" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVADD levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto [databas]

Exempel:

```
$ ./STYRMAN olfix LEVADD levnr levorgnr levnamn levadress levpostnr levpostadress
levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld
levkonto levkundnr valuta betalvilkor [olfixtst]
```

SQLsats som används:

```
INSERT INTO LEVREG
(LEVNR,LEVORGNR,LEVNAMN,LEVADRESS,LEVPOSTNR,LEVPOSTADR,LEVLAND,LEVTFNNR,LEVFXNR,
LEVTELEX,LEVEMAIL,LEVREFERENT,LEVREFTFN,LEVMOMSKOD,LEVSKULD,LEVKONTO,LEVKUNDNR,VALUTA,BETA
LVILKOR)
VALUES
("2345","559999-9999","Leverantör AB","Postgatan 33","199 99","DATABY","SVERIGE","09-
999999","09-999998","99999","kuntj@leverantor.se","Per Josefsson","09-
999997","1","0.00","2110","12345678","SEK","2")
```

## Interface:

### INPUT:

LEVNR, LEVORGNR, LEVNAMN, LEVADDRESS, LEVPOSTNR, LEVPOSTADR,  
LEVLAND, LEVTFNNR, LEVFAXNR, LEVTELEX, LEVEMAIL, LEVREFERENT,  
LEVREFTFN, LEVMOMSKOD, LEVSKULD, LEVKONTO, LEVKUNDNR, VALUTA ,  
BETALVILKOR

### OUTPUT:

```
fprintf(stdout, "OK: LEVADD Inserted %lu rows\n"  
fprintf(stderr, "Error: LEVADD INSERT error: %d %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection))  
fprintf(stderr, "Error: LEVADD Connection failed\n")  
fprintf(stderr, "Error: LEVADD Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection))
```

# LEVCHG

Funktion för att ändra information på en leverantör.

Funktionen anropas med parametern **levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betaltvilkor** och som option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

path/LEVCHG levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betaltvilkor [databas]

Exempel:

```
$ ./LEVCHG 123 "559999-9999" "Leverantör AB" "Postgatan 33"
"199 99" "DATABY" "SVERIGE" "09-999999" "09-999998" "99999"
"kundtj@leverantor.se" "Per Josefsson" "09-999997" "1" "2110"
"12345678" "SEK" "2" [olfix ]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

path/STYRMAN userid LEVCHG levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levkonto levkundnr valuta betaltvilkor [databas]

Exempel:

```
$ ./STYRMAN olfix LEVCHG 123 "559999-9999" "Leverantör AB" "Postgatan 33"
"199 99" "DATABY" "SVERIGE" "09-999999" "09-999998" "99999"
"kundtj@leverantor.se" "Per Josefsson" "09-999997" "1" "2110"
"12345678" "SEK" "2" [olfixtst]
```

SQLsats som används:

```
UPDATE LEVREG SET
LEVORGNR="559999-9999",LEVNAMN="Leverantör AB",LEVADRESS="Postgatan 33",
LEVPOSTNR="199 99",LEVPOSTADR="DATABY", LEVLAND="Sverige",
LEVTFNNR="09-999999", LEVFAXNR="09-999998", LEVTELEX="99999",
LEVEMAIL="kundtj@leverantor.se",LEVREFERENT="Per Josefsson",
LEVREFTFN="09-999997", LEVMOMSKOD="1",LEVKONTO="2110",LEVKUNDNR="12345678",
LEVVALUTA="SEK",BETALVILKOR="2"
```

WHERE LEVNR="123"

## Interface:

### INPUT:

. LEVNR, LEVORGNR, LEVNAMN, LEVADDRESS, LEVPOSTNR, LEVPOSTADR,  
LEVLAND, LEVTFNNR, LEVFAXNR, LEVTELEX, LEVEMAIL, LEVREFERENT,  
LEVREFTFN, LEVMOMSKOD, LEVKONTO, LEVKUNDNR,  
LEVVALUTA, BETALVILKOR

### OUTPUT:

```
fprintf(stdout, "OK: LEVCHG Updated %lu rows\n",  
         (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr, "Error: LEVCHG Updated %lu rows\n",  
         (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr, "Error: LEVCHG UPDATE error: %d %s\n",  
         mysql_errno(&my_connection), mysql_error(&my_connection));  
fprintf(stderr, "Error: LEVCHG Connection failed\n");  
fprintf(stderr, "Error: LEVCHG Connection error %d: %s\n",  
         mysql_errno(&my_connection), mysql_error(&my_connection));
```

# LEVDSP

Funktionen är avsedd att visa information på en leverantör.

Funktionen anropas med parametern **levnr** och som option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVDSP levnr [databas]

Exempel:

```
$ ./LEVDSP 123 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVDSP levnr [databas].

Exempel:

```
$ ./STYRMAN olfix LEVDSP levnr [olfixtst]
```

SQLsats som används:

```
SELECT * FROM LEVREG WHERE (LEVNR = "1234")
```

## Interface:

INPUT:

LEVNR

OUTPUT:

```
fprintf(stderr, "Error: LEVDSP SELECT errno: %d\n", mysql_errno(&my_connection));
fprintf(stdout, "OK:");
fprintf(stdout, "1:%s ", sqlrow[0]);
fprintf(stdout, "2:%s ", sqlrow[1]);
fprintf(stdout, "3:%s ", sqlrow[2]);
fprintf(stdout, "4:%s ", sqlrow[3]);
fprintf(stdout, "5:%s ", sqlrow[4]);
fprintf(stdout, "6:%s ", sqlrow[5]);
fprintf(stdout, "7:%s ", sqlrow[6]);
fprintf(stdout, "8:%s ", sqlrow[7]);
fprintf(stdout, "9:%s ", sqlrow[8]);
fprintf(stdout, "10:%s ", sqlrow[9]);
fprintf(stdout, "11:%s ", sqlrow[10]);
```

```

fprintf(stdout,"12:%s ",sqlrow[11]);
fprintf(stdout,"13:%s ",sqlrow[12]);
fprintf(stdout,"14:%s ",sqlrow[13]);
fprintf(stdout,"15:%s ",sqlrow[14]);
fprintf(stdout,"16:%s ",sqlrow[15]);
fprintf(stdout,"17:%s ",sqlrow[16]);
fprintf(stdout,"18:%s ",sqlrow[17]);
fprintf(stdout,"19:%s ",sqlrow[18]);
fprintf(stdout,"20:%s ",sqlrow[19]);
fprintf(stdout,"21:%s ",sqlrow[20]);
fprintf(stdout,"22:%s ",sqlrow[21]);
fprintf(stdout,"\n");
fprintf(stderr,"Error: LEVDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: LEVDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: LEVDSP Connection failed\n");
fprintf(stderr,"Error: LEVDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));

```



# LEVLST

Funktion för att lista leverantörer, leverantörsnummer och leverantörsnamn.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVLST [databas]

Exempel:

```
$ ./LEVLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVLST [databas]

Exempel:

```
$ ./STYRMAN olfix LEVLST [olfixtst]
```

SQLsats som används:

```
SELECT LEVNR,LEVNAMN FROM LEVREG ORDER BY LEVNAMN
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stdout,"\n");
fprintf(stderr,"Error: LEVLST SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVLST Retrieval error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVLST Connection failed\n");
fprintf(stderr,"Error: LEVLST Connection error %d:%s\n",mysql_errno(&my_connection),
mysql_error(&my_connection));
```



# LEVSADD

Funktion för att registrera nya **leveranssätt**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVSADD levsettnr levsetttext [databas]

Exempel:

```
$ ./LEVSADD 001 "ASG.Kundnr 999999" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVSADD levsettnr levsetttext [databas]

Exempel:

```
$ ./STYRMAN olfix LEVSADD 001 "ASG.Kundnr 999999" [olfixtst]
```

SQLsats som används:

```
INSERT INTO LEVSETT(LEVSETTNR,LEVSETTTEXT) VALUES ( "001","ASG.Kundnr 999999")
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: LEVSADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: LEVSADD INSERT error: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVSADD Connection failed\n");
fprintf(stderr,"Error: LEVSADD Connection error %d:%s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
```

# LEVSDSP

Funktion för att visa texten för önskat **leveranssätt**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVSDSP levsettnr [databas]

Exempel:

```
$ ./LEVSDSP 001 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVSDSP levsettnr [databas]

Exempel:

```
$ ./STYRMAN olfix LEVSDSP 001 [olfixtst]
```

SQLsats som används:

```
SELECT * FROM LEVSETT WHERE LEVSETTNR = "001 "
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: ");
fprintf(stdout,"01: %s ",sqlrow[0]);
fprintf(stdout,"02: %s ",sqlrow[1]);
fprintf(stdout,"END:");
fprintf(stdout,"\n");
fprintf(stderr,"Error: LEVSDSP SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVSDSP Leveranssätt saknas!\n");
fprintf(stderr,"Error: LEVSDSP Retrieval error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVSDSP Connection failed\n");
fprintf(stderr,"Error: LEVSDSP Connection error %d:%s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: Leveranssätt saknas.\n");
```



# LEVSLST

Funktion för att lista alla **leveranssätt**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVSLST [databas]

Exempel:

```
$ ./LEVSLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVSLST [databas]

Exempel:

```
$ ./STYRMAN olfix LEVSLST [olfixtst]
```

SQLsats som används:

```
SELECT * FROM LEVSETT ORDER BY LEVSETTNR
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout, "OK: ");
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: LEVSLST SELECT errno: %d\n", mysql_errno(&my_connection));
fprintf(stderr, "Error: LEVSLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: LEVSLST Connection failed\n");
fprintf(stderr, "Error: LEVSLST Connection error %d:%s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
```



# LEVVADD

Funktion för att registrera nya **leveransvillkor**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVVADD villkorsnr villkorstext [databas]

Exempel:

```
$ ./LEVVADD 001 "EXW" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVVADD villkorsnr villkorstext [databas]

Exempel:

```
$ ./STYRMAN olfix LEVVADD 001 "EXW" [olfixtst]
```

SQLsats som används:

```
INSERT INTO LEVVILLKOR(VILLKORSNR,VILLKORSTEXT) VALUES ( "001","EXW" )
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: LEVVADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: LEVVADD INSERT error: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVVADD Connection failed\n");
fprintf(stderr,"Error: LEVVADD Connection error %d:%s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
```



# LEVVDSP

Funktion för att visa texten för önskat **leveransvillkor**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVVDSP levvillkornr [databas]

Exempel:

```
$ ./LEVVDSP 001 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVLST [databas]

Exempel:

```
$ ./STYRMAN olfix LEVVDSP 001 [olfixtst]
```

SQLsats som används:

```
SELECT * FROM LEVVILLKOR WHERE VILLKORSNR = "001 "
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: ");
fprintf(stdout,"01: %s ",sqlrow[0]);
fprintf(stdout,"02: %s ",sqlrow[1]);
fprintf(stdout,"END:");
fprintf(stdout,"\n");
fprintf(stderr,"Error: LEVVDSP SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVVDSP Leveransvillkor saknas!\n");
fprintf(stderr,"Error: LEVVDSP Retrieval error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVVDSP Connection failed\n");
fprintf(stderr,"Error: LEVVDSP Connection error %d:%s\n",mysql_errno(&my_connection),
mysql_error(&my_connection));
fprintf(stderr,"Error: Leveransvillkor saknas.\n");
```



# LEVVLST

Funktion för att lista alla **leveransvillkor**.

Funktionen anropas utan parameter eller med option **databas**.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/LEVVLST [databas]

Exempel:

```
$ ./LEVVLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVVLST [databas]

Exempel:

```
$ ./STYRMAN olfix LEVSLST [olfixtst]
```

SQLsats som används:

```
SELECT * FROM LEVVILLKOR ORDER BY VILLKORSNR
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout, "OK: ");
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: LEVVLST SELECT errno: %d\n", mysql_errno(&my_connection));
fprintf(stderr, "Error: LEVVLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: LEVVLST Connection failed\n");
fprintf(stderr, "Error: LEVVLST Connection error %d:%s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
```

## LRESADD

Funktionen är avsedd att uppdatera leverantörsreskontran med en post.

Funktionen anropas med följande parametrar:

levnr, fakturanr, regdatum, faktdatum, expiredatum, fakttext, bar, momsprocent, levktonr, faktbelopp, momsktonr, momsbelopp, debetkontonr, debetbelopp, userid, valuta, valutakurs, valutabelopp, OCRnummer, verifikationsnummer och som option [databas]

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=databas existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överridet villkoren A och B.

Syntax:

Path/LRESADD levnr fakturanr regdatum faktdatum expiredatum fakttext bar momsprocent levktonr faktbelopp momsktonr momsbelopp debetkontonr debetbelopp userid valuta valutakurs valutabelopp ocrnr vernr [databas]

Exempel:

```
$ ./LRESADD "123" "1239955" "2003-06-24" "2003-06-17" "2003-07-17" "Inköp av skrivbord"
"AC" "25" "2110" "2500.00" "1470" "625.00" "1810" "1875.00" "JAN" "EUR" "9.08" "275.33"
"19966547812" "00000023" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LRESADD levnr fakturanr regdatum faktdatum expiredatum fakttext bar momsprocent levktonr faktbelopp momsktonr momsbelopp debetkontonr debetbelopp userid valuta valutakurs valutabelopp ocrnr vernr [databas]

Exempel:

```
$ ./STYRMAN olfix LRESADD "123" "1239955" "2003-06-24" "2003-06-17" "2003-07-17" "Inköp
av skrivbord" "AC" "25" "2110" "2500.00" "1470" "625.00" "1810" "1875.00" "JAN" "EUR"
"9.08" "275.33" "19966547812" "00000023" [olfixtst]
```

SQLsats som används:

```
INSERT INTO LEVRESK(LEVNR, FAKTURANR, REGDATUM, FAKTDATUM, EXPIREDATUM, FAKTTEXT, BAR,
MOMSROCENT, LEVKTONR, FAKTBELOPP, MOMSKTONR, MOMSBELOPP, DEBETKONTONR, DEBETBELOPP,
USERID, VALUTA, VALUTAKURS, VALUTABELOPP, OCRNR) VALUES ("123", "1239955", "2003-06-
24", "2003-06-17", "2003-07-17", "Inköp av
skrivbord", "AC", "25", "2110", "2500.00", "1470", "625.00", "1810", "1875.00", "JAN",
"EUR", "9.08", "275.33", "19966547812", "00000023")
```

## Interface:

### INPUT:

LEVNR, FAKTURANR, REGDATUM, FAKTDATUM, EXPIREDATUM, FAKTTEXT,  
BAR, MOMSPROCENT, LEVKTONR, FAKTBELOPP, MOMSKTONR, MOMSBELOPP,  
DEBETKONTONR, DEBETBELOPP, USERID, VALUTA, VALUTAKURS, VALUTABELOPP,  
OCRNR, VERNR

### OUTPUT:

```
fprintf(stdout, "OK: LRESADD Inserted %lu rows\n", (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr, "Error: LRESADD INSERT error: %d %s\n", mysql_errno(&my_connection),  
            mysql_error(&my_connection));  
fprintf(stderr, "Error: LRESADD Connection failed\n");  
fprintf(stderr, "Error: LRESADD Connection error %d: %s\n", mysql_errno(&my_connection),  
            mysql_error(&my_connection));
```

# LRESRPT

Funktionen är avsedd att lista obetalda leverantörsfakturor.

Funktionen anropas utan parameter eller som option [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överridet villkoren A och B.

Syntax:

path/LRESRPT [databas]

Exempel:

```
$ ./LRESRPT [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

path/STYRMAN userid LRESRPT [databas]

Exempel:

```
$ ./STYRMAN olfix LRESRPT [olfixtst]
```

Funktionen används för att hämta information om obetalda leverantörsfakturor.

SQLsats som används:

```
SELECT LEVRESK.EXPIREDATUM,LEVRESK.LEVNR,LEVREG.LEVNAMN,LEVRESK.FAKTBELOPP
FROM LEVRESK
LEFT JOIN LEVREG ON LEVREG.LEVNR = LEVRESK.LEVNR
WHERE LEVRESK.BETALD = "N"
ORDER BY LEVRESK.EXPIREDATUM;
```

Utdata från LRESRPT är en enda lång sträng med '\_' (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med 'NR\_' postantal '\_' (underscore).

## Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"NR_%lu_:",(unsigned long)mysql_num_rows(res_ptr));  
fprintf(stderr,"Error: LRESRPT SELECT error: %s\n",mysql_error(&my_connection));  
fprintf(stderr,"Error: LRESRPT Retrieval error: %s\n",mysql_error(&my_connection));  
fprintf(stderr,"Error: LRESRPT Connection failed\n");  
fprintf(stderr,"Error: LRESRPT Connection error %d: %s\n",  
        mysql_errno(&my_connection),mysql_error(&my_connection));
```

# PRGLST

Funktionen är avsedd att lista programnamn så att användaren i programmet OLFIW kan välja vilket program han/hon ska köra.

Funktionen anropas utan parameter eller som option [databas].

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parameter [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/PRGLST [databas]

Exempel:

```
$ ./PRGLST [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PRGLST

Exempel:

```
$ ./STYRMAN olfix PRGLST [olfix]
```

Funktionen används för att hämta information om huvudmeny,undermeny funktionsbeskrivning samt programnamn.

SQLsats som används:

```
SELECT * FROM PROGRAM ORDER BY MENYAVD
```

Utdata från PRGLST är en enda lång sträng med '\_' (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med 'NR\_' postantal '\_' (underscore).



## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: PRGLST SELECT error: %s\n",mysql_error(&my_connection));  
fprintf(stderr,"Error: PRGLST Retrieval error: %s\n", mysql_error(&my_connection));  
fprintf(stderr,"Error: PRGLST Connection failed\n");  
fprintf(stderr,"Error: PRGLST Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));  
fprintf(stdout,"NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));  
fprintf(stdout,"(%d)%s:",field_count,sqlrow[field_count]);
```

## PRTAPI

Function: Anropa Kugar eller Kspread med parametrar om var datafilen finns.

INPUT: csvflag printfil [prttemplate]  
(csvflag flagga "J" eller "N". csv = kommaseparerad fil).  
J för att använda Kspread och N för att använda Kugar.  
Prtfile = datafilen.  
Prttemplate = formateringsmall.

Syntax:  
Path/PRTAPI csvflag prtfile [prttemplate]

Exempel:  
\$ ./PRTAPI J /tmp/Saldolista.kud [/usr/local/olfix/report/Saldolista.kut]

# RGTADD

Funktionen anropas med parametrarna USERID och TRNSID och som option databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=databas existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/RGTADD userid trnsid [databas]

Exempel:

```
$ ./RGTADD JAPI KTOLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 RGTADD userid2 trnsid [databas]

Exempel:

```
$ ./STYRMAN olfix RGTADD JAPI KTOLST [olfixtst]
```

Funktionen används för att lägga upp ny behörighet.

SQLsats som används:

```
INSERT INTO RIGHTS(USERID,TRNSID) VALUES (USERID,TRNSID) VALUES ("KALLE","KTOADD"
```

## Interface:

INPUT:

TRNSID och USERID (fälten i tabellen RIGHTS)

OUTPUT:

```
fprintf(stdout,"OK: RGTADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: RGTADD INSERT error: %d %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: RGTADD Connection failed\n");
fprintf(stderr,"Error: RGTADD Connection error %d: %s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
```

# RGTCHK

Funktionen anropas med parametrarna USERID och TRNSID och som option databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/RGTCHK userid trnsid [databas]

Exempel:

```
$ ./RGTCHK JAPI KTOLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 RGTCHK userid2 trnsid [databas]

Exempel:

```
$ ./STYRMAN JAPI RGTCHK JAPI KTOLST [olfixtst]
```

Returvärde från RGTCHK = 0 om kombinationen userid och trnsid finns i tabellen RIGHTSt annars returneras värdet -1.

Funktionen är tänkt att användas för att kontrollera om *userid2* har rättighet att använda funktionen trnsid.

SQLsats som används;

```
SELECT USERID,TRNSID FROM RIGHTS WHERE USERID = "JAPI" AND TRNSID ="KTOLST"
```

## Interface:

INPUT:

TRNSID och USERID (fälten i tabellen RIGHTS)

OUTPUT:

```
fprintf(stderr,"Error: RGTCHK_Selection error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"RGTCHKmain:Retrieved %lu rows\n",(unsigned long)mysql_num_rows(res_ptr)
fprintf(stderr,"Error: RGTCHK_Retrieve error  %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: RGTCHK_Connection failed\n");
fprintf(stderr,"Error: RGTCHK_Connection error: %d  %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stdout,"OK: RGTCHK_Status = %d\n",status);
fprintf(stderr,"Error: RGTCHK_%s har inte behörighet till %s\n",userid,trnsid);
```

## RGTDEL

Funktionen anropas med parametrarna USERID och TRNSID och som option databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/RGTDEL userid trnsid [databas]

Exempel:

```
$ ./RGTDEL JAPI KTOLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 RGTDEL userid2 trnsid [databas]

Exempel:

```
$ ./STYRMAN JAPI RGTDEL JAPI KTOLST [olfixtst]
```

Returvärde från RGTDEL = 0 om kombinationen userid och trnsid finns i tabellen RIGHTSt annars returneras värdet -1.

Funktionen används för att radera *userid2* rättighet att använda funktionen trnsid.

SQLsats som används;

```
DELETE FROM RIGHTS WHERE USERID = "JAPI" AND TRNSID ="KTOLST"
```

### Interface:

INPUT:

TRNSID och USERID (fälten i tabellen RIGHTS)

OUTPUT:

```
fprintf(stdout,"OK: RGTDEL Deleted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: RGTDEL DELETE ERROR: Ingen post raderad\n");
fprintf(stderr,"Error: RGTDEL Connection failed\n");
fprintf(stderr,"Error: RGTDEL Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



## RGTDSP

Funktionen anropas med parametrarna **USERID** och som option **databas**.  
Funktionen returnerar **USERID** och **TRNSID** för aktuellt **userid**.

Programmet testar vilken **databas** som skall anropas genom att läsa filen **\$HOME/.olfixrc**. Om raden **DATABASE=databas** existerar så läses **databas** in och anropas.

Regler:

A. Om parametern **[databas]** är utelämnad så

1. Om **DATABASE** innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter **[databas]** innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i **[databas]**

C. Om **USER** innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/RGTDSP **userid** **[databas]**

Exempel:

```
$ ./RGTDSP JAN [olfix]
```

Normalt görs anropet via **STYRMAN** och då blir syntaxen:

Path/STYRMAN **userid1** RGTDSP **userid2** **[databas]**

Exempel:

```
$ ./STYRMAN JAPI RGTDSP JAN [olfixtst]
```

RGTDSP skriver ut data på **stdout** (konsolen), samtliga behörigheter för en användare.

SQLsats som används;

```
SELECT USERID,TRNSID FROM RIGHTS WHERE USERID = "JAN"
```

Output:

```
NR_8_:JAN_:BARADD_:JAN_:BOKF_:JAN_:KTOADD_:JAN_:KTOVIEW_:JAN_:RGTTADD_:JAN_:RGTLST_:JAN_
:USERADD_:JAN_:USERLST_:
```

**NR\_8** anger antal poster som hämtats.

**\_**: används för att särskilja data. Ingen särskilnad mellan poster.

## Interface:

INPUT:

USERID

OUTPUT:

```
fprintf(stderr,"Error: RGTDSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: RGTDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: RGTDSP Connection failed\n");
fprintf(stderr,"Error: RGTDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# RGTLST

Funktionen anropas utan parametrar eller med databas som option.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixtst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/RGTLST [databas]

Exempel:

```
$ ./RGTLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 RGTLST [databas]

Exempel:

```
$ ./STYRMAN JAPI RGTLST [olfixtst]
```

RGTLST skriver ut data på stdout (konsolen) för alla poster i tabellen RIGHTS.

Funktionen användas för lista alla behörigheter, sorterad på användare (userid).

SQLsats som används;

```
SELECT * FROM RIGHTS ORDER BY USERID
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: RGTLST SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: RGTLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: RGTLST Connection failed\n");
fprintf(stderr, "Error: RGTLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# RPTCRE

Funktionen är en rapportgenerator som skapar en CSV-fil utifrån valfri SQL-fråga. RPTCRE anropas med en SQL-sats som parameter och med databas som option.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/RPTCRE sqlsats [databas]

Exempel:

```
$ ./RPTCRE "SELECT * FROM VERRAD" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid RPTCRE sqlsats [databas]

Exempel:

```
$ ./STYRMAN JAPI RPTCRE "SELECT * FROM VERRAD" [olfixtst]
```

RPTCRE skriver ut data till filen /tmp/rptcre.txt.

Funktionen används för att själv göra rapporter med Kspread, csvformat

SQLsats som används;

Valfri.

## Interface:

INPUT:

Valfri SQL-fråga.

OUTPUT:

```
fprintf(stderr,"Error: RPTCRE SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: RPTCRE Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: RPTCRE Connection failed\n");
fprintf(stderr,"Error: RPTCRE Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

data till filen /tmp/rptcre.txt

# SLPADD

Funktionen lägger upp en ny standardleveransplats i tabellen STDLEVPLATS .

Funktionen anropas med parametrarna KUNDNR, STDLEVPLATS, ADRESS, POSTNR, POSTADR och LAND och som option databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/SLPADD kundnr stdlevplats adress postnr postadress land [databas]

Exempel:

```
$ ./SLPADD 1234 001 "Långgatan 32" "199 99" "Storstad" "Sverige" [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid SLPADD kundnr stdlevplats adress postnr postadress land [databas]

Exempel:

```
$ ./STYRMAN JAPI SLPADD 1234 001 "Långgatan 32" "199 99" "Storstad" "Sverige"
[olfixtst]
```

SQLsats som används;

```
INSERT INTO STDLEVPLATS(KUNDNR,STDLEVPLATS,ADRESS,POSTNR,POSTADR,LAND) VALUES
("1234","001","Långgatan 32","199 99","Storstad","Sverige")
```

## Interface:

INPUT:

KUNDNR STDLEVPLATS ADRESS POSTNR POSTADR LAND

OUTPUT:

```
fprintf(stdout,"OK: SLPADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: SLPADD INSERT error: %d %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: SLPADD Connection failed\n");
fprintf(stderr,"Error: SLPADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# STYRMAN

STYRMAN är det centrala styrprogrammet för OLFIX.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A.

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Villkor B övertider villkor A.

STYRMAN gör följande kontroller innan det anropar önskad funktion (program):

1. Att angiven användare(USERID) finns,
2. Att önskad funktion/program finns,
3. Att användaren har behörighet att använda funktionen.

STYRMAN anropar önskad funktion på följande sätt;

path/STYRMAN userid FUNKTION med eller utan argument.

**Exempel** på hur man lägger upp en ny användare.

Syntax:

Path/STYRMAN user1 USERADD user2 "namn" avd grupp

```
$ ./STYRMAN KALLE USERADD PELLE "Per Andersson" IT Stab
```

Ange program STYRMAN med parametrarna/argumenten:

user1 (KALLE, USERID på en användare med rättighet att lägga upp nya användare)

funk (USERADD, funktionen för att lägga upp nya användare)

user2 (PELLE, USERID på den nye användaren)

namn ("Per Andersson", NAMN på nye användaren. OBS! Skriv även in citationstecknen)

avd (IT, AVDelningen som den nye användaren tillhör)

grupp (Stab, GRUPP, Affärsmässig gruppindelning, ej att förväxla med operativsystemets grupp)

## Interface:

### INPUT:

USERID, TRNSID, trnsdata1, trnsdata2, trnsdata3 ..... trnsdata24

USERID = Userid för den användare som är inloggad.

TRNSID = Funktion som skall användas.

Trnsdata1 ... trnsdata18 = argument till anropad funktion (TRNSID).

### OUTPUT:

fprintf(stderr,"%s\n",felpek);	Önskad transaktion kunde ej utföras.
fprintf(stdout,"%s\n",&datastr);	Transaktionen lyckades.

```
fprintf(stderr,"Error: STYRMAN_main. Användare %s finns ej\n",argv[1]);
fprintf(stderr,"Error: STYRMAN_main. Function %s finns ej\n",argv[2]);
fprintf(stderr,"Error: STYRMAN_main. User %s har ej behörighet till %s\n",argv[1],argv[2]);
```

```
fprintf(stderr,"Error: STYRMAN_check-Transtyp_SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check-Transtyp_Retrieve error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check-Transtyp_Connection failed\n");
fprintf(stderr,"Error: STYRMAN_check-Transtyp_Connection error %d: %s\n",mysql_errno(&my_connection),
mysql_error(&my_connection));
```

```
fprintf(stderr,"Error: STYRMAN_check-User_SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check-User_Retrieve error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check-User_Connection failed\n");
fprintf(stderr,"Error: STYRMAN_check-User_Connection error %d: %s\n",mysql_errno(&my_connection),
mysql_error(&my_connection));
```

```
fprintf(stderr,"Error: STYRMAN_check_Rights_SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check_Rights_Retrieve error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: STYRMAN_check_Rights_Connection failed\n");
fprintf(stderr,"Error: STYRMAN_checkRights_Connection error %d: %s\n",mysql_errno(&my_connection),
mysql_error(&my_connection));
```

# TRHDADD

Funktionen anropas med parametrarna TRNSID, TID, USERID och TRNSDATA och som option databas. TRHDADD anropas alltid från någon annan funktion, aldrig från GUI eller konsol.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/TRHDADD trnsid tid userid trnsdata [databas]

trnsid = 8 tecken

trnstid = 21 tecken (Format YYYY-MM-DD\_tt:mm:ss)

userid = 8 tecken

trnsdata = 60 tecken ?

SQLsats;

```
INSERT INTO TRHD(TRNSID, TID, USERID, TRNSDATA) VALUES
("trnsid","trnstid","userid","trnsdata")
```

## Interface:

INPUT:

TRNSID, TID, USERID, TRNSDATA

OUTPUT:

```
fprintf(stdout,"OK: TRHDADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: TRHDADD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: TRHDADD Connection failed\n");
fprintf(stderr,"Error: TRHDADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# TRNSADD

Funktionen anropas med parametrarna TRNSID , TRNSTXT och som option databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/TRNADD trnsid trnstxt [databas].

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid TRNSADD trnsid trnstxt [databas]

SQLsats:

```
INSERT INTO TRANSID(TRNSID,TRNSTXT) VALUES "VALDSP","Visa en valuta")
```

## Interface:

INPUT:

TRNSID, TRNSTXT

OUTPUT:

```
fprintf(stdout,"OK: Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: TRNSADD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: TRNSADD Connection failed\n");
fprintf(stderr,"Error: TRNSADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# TRNSLST

Funktionen anropas utan parametrar eller ,som option, med databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/TRNSLST [databas].

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid TRNSLST [databas]

SQLsats;

```
SELECT * FROM TRANSID ORDER BY TRNSID
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: TRNSLST SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: TRNSLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: TRNSLST Connection failed\n");
fprintf(stderr,"Error: TRNSLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"NR_%lu:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
```

# TXTADD

Funktionen anropas med parametrarna `textnr`, `txt` och `,som option, databas`.

Programmet testar vilken databas som skall anropas genom att läsa filen `$HOME/.olfixrc`. Om raden `DATABASE=databas` existerar så läses databas in och anropas.

Regler:

A. Om parametern `[databas]` är utelämnad så

1. Om `DATABASE` innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter `[databas]` innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i `[databas]`

C. Om `USER` innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

`Path/TXTADD textnr txt [databas]`.

Normalt görs anropet via `STYRMAN` och då blir syntaxen:

`Path/STYRMAN userid TXTADD textnr txt [databas]`.

SQLsats;

```
INSERT INTO TEXTREG(TEXTNR,TXT) VALUES VALUES ("001","Löpande text med information")
```

## Interface:

INPUT:

`TEXTNR, TXT`

OUTPUT:

```
fprintf(stdout,"OK: TXTADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: TXTADD INSERT error: %d %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: TXTADD Connection failed\n");
fprintf(stderr,"Error: TXTADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# TXTDEL

Funktionen anropas med parametern textnr och ,som option, med databas.

Funktionen plockar bort en post (textnr) ur tabellen TEXTREG.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider villkoren A och B.

Syntax:

Path/TXTDEL textnr [databas].

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid TXTDEL textnr [databas].

SQLsats;

```
DELETE FROM TEXTREG WHERE TEXTNR = "001"
```

## Interface:

INPUT:

TEXTNR

OUTPUT:

```
printf("OK: TXTDEL Deleted %lu rows\n",
      (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: TXTDEL DELETE error: %d %s\n", mysql_errno(&my_connection),
      mysql_error(&my_connection));
fprintf(stderr,"Error: TXTDEL Connection failed\n");
fprintf(stderr,"Error: TXTDEL Connection error %d: %s\n",
      mysql_errno(&my_connection), mysql_error(&my_connection));
```

# TXTDSP

Funktionen anropas med parametern textnr och ,som option, med databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/TXTDSP textnr [databas].

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid TXTDSP textnr [databas].

SQLsats;

```
SELECT TEXTNR, TXT FROM TEXTREG WHERE TEXTNR = "001"
```

## Interface:

INPUT:

TEXTNR

OUTPUT:

```
fprintf(stderr, "Error: TXTDSP SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "OK: ");
fprintf(stdout, "01:%s ", sqlrow[0]);
fprintf(stdout, "02:%s ", sqlrow[1]);
fprintf(stdout, "\n");
fprintf(stderr, "Error: TXTDSP: Data saknas! %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: TXTDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: TXTDSP Connection failed\n");
fprintf(stderr, "Error: TXTDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# USERADD

Funktionen anropas med parametrarna USERID, NAMN, AVD, GRUPP och, som option, databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=databas existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.

Vilkor C övertider villkoren A och B.

Syntax:

Path/USERADD userid "namn" avd grupp [databas]

Exempel:

```
$ ./USERADD KALLE "Karl Andersson" Ekonomi Stab [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 USERADD userid2 namn avd grupp

Exempel:

```
$ ./STYRMAN olfix USERADD KALLE "Karl Andersson" Ekonomi Stab [olfixst]
```

Funktionen används för att lägga upp en ny användare av OLFIX.

SQLsats som används:

```
INSERT INTO USR(USERID,NAMN,AVD,GRUPP) VALUES ("KALLE","Karl Andersson","Ekonomi","Stab")
```

## Interface:

INPUT:

USERID, NAMN, AVD och GRUPP

OUTPUT:

```
fprintf(stdout,"OK: USERADD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: USERADD INSERT error: %d %s\n", mysql_errno(&my_connection),
        mysql_error(&my_connection));
fprintf(stderr,"Error: USERADD Connection failed\n");
fprintf(stderr,"Error: USERADD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# USERCHG

Funktionen anropas med parametern USERID, NAMN,AVD,GRUPP och, som option, databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/USERCHG userid namn avd grupp [databas]

Exempel:

```
$ ./USERCHG KALLE "Karl Andersson" Personal Stab [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 USERCHG userid2 namn avd grupp [databas]

Exempel:

```
$ ./STYRMAN olfix USERCHG KALLE "Karl Andersson" Personal Stab [olfixst]
```

Funktionen används för att ändra information på en användare.

SQLsats som används:

```
UPDATE USR SET NAMN = "Karl Andersson",AVD = "Personal",GRUPP = "Stab" WHERE USERID = "KALLE"
```

## Interface:

INPUT:

USERID, NAMN, AVD, GRUPP

OUTPUT:

```
fprintf(stdout,"OK: USERCHG Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: USERCHG INSERT error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: USERCHG Connection failed\n");
fprintf(stderr,"Error: USERCHG Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# USERDEL

Funktionen anropas med parametern USERID och, som option, databas.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Villkor C övertider villkoren A och B.

Syntax:

Path/USERDEL userid [databas]

Exempel:

```
$ ./USERDEL JAN [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 USERDEL userid2 [databas]

Exempel:

```
$ ./STYRMAN olfix USERDEL JAN [olfixtst]
```

Funktionen används för att visa all information på en användare.

SQLsats som används:

```
DELETE FROM USR WHERE USERID = "JAN"
```

## Interface:

INPUT:

USERID

OUTPUT:

```
printf("OK: USERDEL Deleted %lu rows\n",
      (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr, "Error: USERDEL INSERT error: %d %s\n", mysql_errno(&my_connection),
      mysql_error(&my_connection));
fprintf(stderr, "Error: USERDEL Connection failed\n");
fprintf(stderr, "Error: USERDEL Connection error %d: %s\n",
      mysql_errno(&my_connection), mysql_error(&my_connection));
```

## USERDSP

Funktionen anropas med parametern USERID och, som option, databas.  
Funktionen används för att visa all information på en användare.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    - 1. Om DATABASE innehåller ett värde används detta
    - 2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    - 1. Om värdet är **99** så används **olfixtst**, testföretag
    - 2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/USERDSP userid [databas]

Exempel:

```
$ ./USERDSP JAN [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 USERDSP userid2 [databas]

Exempel:

```
$ ./STYRMAN olfix USERDSP JAN [olfixtst]
```

SQLsats som används:

```
SELECT USERID,NAMN,AVD,GRUPP FROM USR WHERE USERID = "JAN"
```

Output:

```
1:JAN 2:Jan Pihlgren 3:Ekonomi 4:Stab
```

### Interface:

INPUT:

USERID

OUTPUT:

```
fprintf(stderr,"Error: USERDSP: Ange userid!\n");
fprintf(stderr,"Error: USERDSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"1:%s ",sqlrow[0]);
fprintf(stdout,"2:%s ",sqlrow[1]);
fprintf(stdout,"3:%s ",sqlrow[2]);
fprintf(stdout,"4:%s ",sqlrow[3]);
fprintf(stdout,"\n");
fprintf(stderr,"Error: USERDSP. User %s finns ej!\n",userid);
fprintf(stderr,"Error: USERDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: USERDSP Connection failed\n");
fprintf(stderr,"Error: USERDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```





# USERLST

Funktionen anropas utan parameter eller med, som option, *databas*.  
Funktionen används för att visa all information på en användare.

Programmet testar vilken *databas* som skall anropas genom att läsa filen `$HOME/.olfixrc`. Om raden `DATABASE=databas` existerar så läses *databas* in och anropas.

Regler:

- A. Om parametern [*databas*] är utelämnad så
    1. Om `DATABASE` innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [*databas*] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [*databas*]
  - C. Om `USER` innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

`Path/USERLST [databas]`

Exempel:

```
$ ./USERLST [olfix]
```

Normalt görs anropet via `STYRMAN` och då blir syntaxen:

`Path/STYRMAN userid USERLST [databas]`

Exempel:

```
$ ./STYRMAN olfix USERLST [olfixtst]
```

SQLsats som används:

```
SELECT * FROM USR ORDER BY USERID
```

Utdata från `USERLST` är en enda lång sträng med `'_'` (underscore och semikolon) som fältåtskiljare och poståtskiljare.

Strängen inleds med `'NR_'` postantal `'_'` (underscore).

Exempel på hur man kan dela up fält och poster finns i konsolprogrammet `ADMINs` funktion `UserList`.

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: USERLST SELECT error: %s\n", mysql_error(&my_connection));  
fprintf(stderr, "Error: USERLST Retrieval error: %s\n", mysql_error(&my_connection));  
fprintf(stderr, "Error: USERLST Connection failed\n");  
fprintf(stderr, "Error: USERLST Connection error %d: %s\n",  
        mysql_errno(&my_connection), mysql_error(&my_connection));  
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
```

```
fprintf(stdout,"%s:",sqlrow[field_count]);
```

# VALADD

Funktionen anropas med parametrarna VALUTAID, LAND, SALJ, KOP, BETECKNING och som option, databas.

Funktionen används för att lägga upp en ny valuta.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C överrider villkoren A och B.

Syntax:

Path/VALADD valutaaid land salj kop beteckning [databas]

Exempel:

```
$ ./VALADD SEK Sverige 1.01 1.00 Kronor [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALADD valutaaid land salj kop beteckning [databas]

Exempel:

```
$ ./STYRMAN olfix VALADD SEK Sverige 1.01 1.00 Kronor [olfixtst]
```

SQLsats som används:

```
INSERT INTO VALUTA(VALUTAID, LAND, SALJ, KOP, BETECKNING) VALUES ("SEK", "Sverige", "1.01", "1.00", "Kronor")
```

## Interface:

INPUT:

VALUTAID, LAND, SALJ, KOP, BETECKNING

OUTPUT:

```
fprintf(stdout, "OK: VALADD Inserted %lu rows\n",
         (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr, "Error: VALADD INSERT error: %d %s\n",
         mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr, "Error: VALADD Connection failed\n");
fprintf(stderr, "Error: VALADD Connection error %d: %s\n",
         mysql_errno(&my_connection), mysql_error(&my_connection));
```

# VALCHG

Funktionen anropas med parametern VALUTAID samt , som option, databas.  
Funktionen används för att ändra information på en valuta.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/VALCHG valutaid land salj kop beteckning [databas]

Exempel:

```
$ ./VALCHG SEK Sverige 1.01 1.00 Kronor [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALCHG valutaid land salj kop beteckning [databas]

Exempel:

```
$ ./STYRMAN olfix VALCHG SEK Sverige 1.01 1.00 Kronor [olfixtst]
```

SQLsats som används:

```
UPDATE VALUTA SET LAND = "Sverige",SALJ = "1.01",KOP = "1.02",BETECKNING = "Kronor" WHERE VALUTAID = "SEK"
```

## Interface:

INPUT:

VALUTAID

OUTPUT:

```
fprintf(stdout,"OK: VALCHG Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: VALCHG INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: VALCHG Connection failed\n");
fprintf(stderr,"Error: VALCHG Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```



# VALDEL

Funktionen anropas med parametern VALUTAID samt, som option, databas.  
Funktionen används för att ta bort en valuta.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/VALDEL valutaaid [databas]

Exempel:

```
$ ./VALDEL SEK [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALDEL valutaaid [databas]

Exempel:

```
$ ./STYRMAN olfix VALDEL SEK [olfixtst]
```

SQLsats som används:

```
DELETE FROM VALUTA WHERE VALUTAID = "SEK"
```

## Interface:

INPUT:

VALUTAID

OUTPUT:

```
fprintf(stdout, "OK: VALDEL Deleted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr, "Error: VALDEL INSERT error: %d %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr, "Error: VALDEL Connection failed\n");
fprintf(stderr, "Error: VALDEL Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# VALDSP

Funktionen anropas med parametern VALUTAID samt, som option, databas.  
Funktionen används för att visa information om en valuta.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixtst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixtst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/VALDSP valutaaid [databas]

Exempel:

```
$ ./VALDSP SEK [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALDSP valutaaid [databas]

Exempel:

```
$ ./STYRMAN olfix VALDSP SEK [olfixtst]
```

SQLsats som används:

```
SELECT VALUTAID, LAND, BETECKNING, KOP, SALJ FROM VALUTA WHERE VALUTAID = "SEK"
```

## Interface:

INPUT:

VALUTAID

OUTPUT:

```
fprintf(stderr, "Error: VALDSP SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "1:%s  ", sqlrow[0]);
fprintf(stdout, "2:%s  ", sqlrow[1]);
fprintf(stdout, "3:%s  ", sqlrow[2]);
fprintf(stdout, "4:%s  ", sqlrow[3]);
fprintf(stdout, "5:%s  ", sqlrow[4]);
fprintf(stdout, "\n");
fprintf(stderr, "Error: VALDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: VALDSP Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: VALDSP Connection failed\n");
fprintf(stderr, "Error: VALDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```





# VALLST

Funktionen anropas utan parametrar eller, som option, med databas.  
Funktionen används för att visa information om alla valutor.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

- A. Om parametern [databas] är utelämnad så
    1. Om DATABASE innehåller ett värde används detta
    2. Annars används **olfixstst**, testföretag
  - B. Om parameter [databas] innehåller ett värde så
    1. Om värdet är **99** så används **olfixstst**, testföretag
    2. Annars används värdet i [databas]
  - C. Om USER innehåller strängen **test** eller **prov** så används **olfixstst**.
- Vilkor C övertider villkoren A och B.

Syntax:

Path/VALLST [databas]

Exempel:

```
$ ./VALLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALLST [databas]

Exempel:

```
$ ./STYRMAN olfix VALLST [olfixstst]
```

SQLsats som används:

```
SELECT * FROM VALUTA ORDER BY VALUTAID
```

## Interface:

INPUT:

OUTPUT:

```
fprintf(stderr, "Error: VALLST SELECT error: %s\n", mysql_error(&my_connection));
fprintf(stdout, "NR_%lu:", (unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout, "%s:", sqlrow[field_count]);
fprintf(stderr, "Error: VALLST Retrieval error: %s\n", mysql_error(&my_connection));
fprintf(stderr, "Error: VALLST Connection failed\n");
fprintf(stderr, "Error: VALLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## VERUPD

BOKFORSW är det program där bokföringen sker.

Initialt kontrolleras att den user som vill registrera en verifikation har behörighet till detta.

Posterna lagras i temporärfilen vernr.txt. Sökvägen till vernr.txt hämtas från \$HOME/.olfixrc.

BOKFORSW kontrollerar fortlöpande saldot på verifikationen vilket registrerats på verifikationsrad nr 1.

För varje därpå följande verifikationsrad så minskas saldot med det belopp som konteras på raden. Först när saldot är 0 (noll) så kan verifikationen godkännas som färdigbehandlad.

När verifikationen är färdigbehandlad anropas funktionen VERUPD via STYRMAN.

Funktionen VERUPD uppdaterar databasen från filen vernr.txt. När alla poster i vernr.txt är behandlade så raderas filen.

Programmet testar vilken databas som skall anropas genom att läsa filen \$HOME/.olfixrc. Om raden DATABASE=*databas* existerar så läses databas in och anropas.

Regler:

A. Om parametern [databas] är utelämnad så

1. Om DATABASE innehåller ett värde används detta
2. Annars används **olfixtst**, testföretag

B. Om parameter [databas] innehåller ett värde så

1. Om värdet är **99** så används **olfixtst**, testföretag
2. Annars används värdet i [databas]

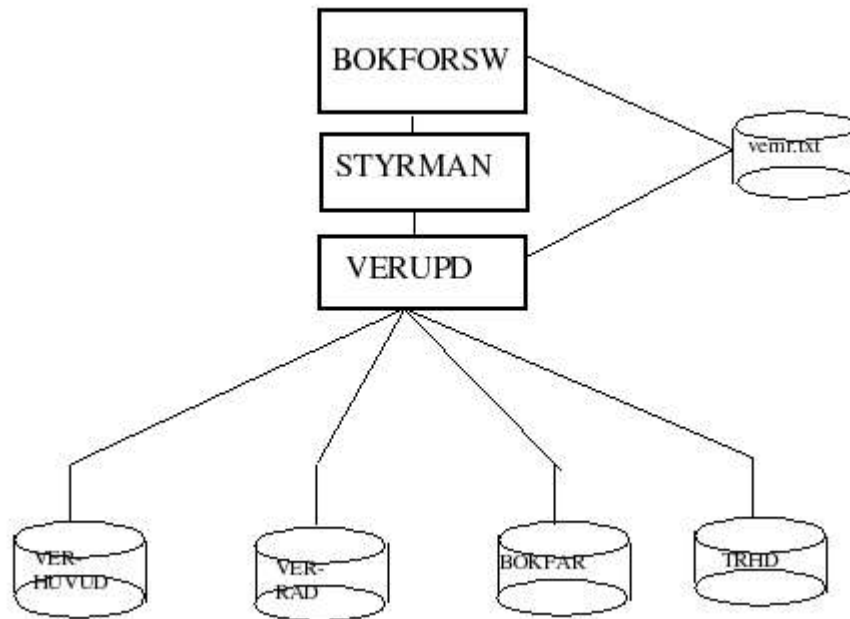
C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C övertider villkoren A och B.

VERUPD läser temporärfilen och uppdatera databasen med dess uppgifter.

För varje post som uppdateras i tabellerna VERHUVUD, VERRAD och TRHD

TRHD är en logg på alla ekonomiska transaktioner och lagrar informationen för varje transaktion.



## Interface:

INPUT:

verifikationsnummer

OUTPUT:

```

fprintf(stderr,"Error: VERUPD. Argumentet med filnamn saknas!\n");
fprintf(stderr,"Error: VERUPD. Filen \"%s\" finns inte!\n",vrnrfil);
fprintf(stdout,"OK: Databasen uppdaterad!\n");

VERHUVUD
fprintf(stdout,"OK: VERUPD VERHUVUD_Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: VERUPD VERHUVUD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: VERUPD VERHUVUD Connection failed\n");
fprintf(stderr,"Error: VERUPD VERHUVUD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));

VERRAD
fprintf(stdout,"OK: VERUPD VERRAD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: VERUPD VERRAD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: VERUPD VERRAD Connection failed\n");
fprintf(stderr,"Error: VERUPD VERRAD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));

TRHD
fprintf(stdout,"OK: VERUPD TRHD Inserted %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: VERUPD TRHD INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: VERUPD TRHD Connection failed\n");
fprintf(stderr,"Error: VERUPD TRHD Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));

VERN
fprintf(stdout,"OK: VERUPD VERNR updated %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: VERUPD VERNR INSERT error: %d %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: VERUPD VERNR Connection failed\n");
fprintf(stderr,"Error: VERUPD VERNR Connection error %d: %s\n",

```

```
mysql_errno(&my_connection), mysql_error(&my_connection));
```

## WRREC ..... vernr.txt

Skapa filen /tmp/vernr.txt.

Sökvägen till vernr.txt hämtas från \$HOME/.olfixrc

OBS. Vernr skall vara en siffra motsvarande verifikationsnumret.

Filen är en temporärfil för att lagra data vid registrering av en verifikation.

När verifikationen har registrerats färdig användsfilen för inlagring i databasen och när inlagringen är klar så raderas filen.

Poststorlek = 178 tecken

Exempel:

/tmp/1234001.txt

H AC 00000012 001 2110 D	4000	2003-03-06 jan	test nr 12
D AC 00000012 002 1810 K	3000		
D AC 00000012 003 2480 K	1000		

Fältbeskrivning huvudpost:

Pos		H	Posttyp
1			
2		blank	
3 - 4			Bokföringsår
5		blank	
6 - 13			Verifikationsnummer
14		blank	
15 - 17			Radnr
18		blank	
19 - 22			Kontonr
23		blank	
24			D/K
25		blank	
26 - 37			Belopp
38		blank	
39 - 42			Kostnadsställe
43		blank	
44 - 47			Subkonto
48		blank	
49 - 58			Datum
59		blank	
60 - 67			Userid
68		blank	
69 - 178			Verifikationstext

Fältbeskrivning detaljpost:

Pos		D	Posttyp
1			
2		blank	
3 - 4			Bokföringsår
5		blank	
6 - 13			Verifikationsnummer

14	blank	
15 - 17		Radnr
18	blank	
19 - 22		Kontonr
23	blank	
24		D/K
25	blank	
26 - 37		Belopp
38	blank	
39 - 42		Kostnadsställe
43	blank	
44 - 47		Subkonto
48 -178	blank	

## Tabeller i OLFIXs databas

### **ARTIKELREG**

Tabell för artikeldata.

### **BOKFAR**

Tabell för att administrera bokföringen.

### **BETVILKOR**

Tabell för betalningsvillkor

### **FTGDATA**

Tabell för företagsdata

### **INKREG**

Tabell för inköpsorder, orderhuvud

### **INKRADREG**

Tabell för inköpsorder, orderrader

### **KSTALLE**

Tabell för kostnadställe/Resultatenhet.

### **KTOPLAN**

Tabell för kontonummer.

### **KUNDREG**

Tabell för kunder.

### **LAGERSTELLEREG**

Tabell för kompletterande artikeldata.

### **LEVREG**

Tabell för leverantörer

### **LEVRESK**

Tabell för leverantörsreskontra.

### **LEVSETT**

Tabell för leveranssätt.

### **LEVVILLKOR**

Tabell för leveransvillkor.

### **PROGRAM**



Tabell för program i OLFIX.

**RIGHTS**

Tabell för rättigheter

**STDLEVPLATS**

Tabell för standardleveransplatser för kunder.

**TEXTREG**

Tabell för att lagra diverse texter.

**TRANSID**

Tabell för funktioner

**TRHD**

Tabell, logg av ekonomiska transaktioner (transaktionshistorik).

**USR**

Tabell för användare

**VALUTA**

Tabell för valutor.

**VERHUVUD**

Tabell för verifikationernas huvudposter.

**VERRAD**

Tabell för verifikationernas huvudposter.

## ARTIKELREG

ARTIKELNR	VARCHAR(30) not null PRIMARY KEY	Artikelnummer/ArtikelID
ARBENEMNING1	VARCHAR(30) not null	Benämning, rad 1
ARBENEMNING2	VARCHAR(30)	Benämning, rad 2
ARENHET	VARCHAR(5)	Lagerenhet (ST,LITER,PAR,DUSS) osv
ARFPRIS	DECIMAL(10,2)	Försäljningspris
ARLEDTID	VARCHAR(3)	Ledtid, dagar
ARPRODKLASS	VARCHAR(5)	Produktklass
ARPRODKTO	VARCHAR(5)	Produktkonto
ARLEVNR1	VARCHAR(10)	Leverantörsnr leverantör nr 1, LEVREG
ARLEVNR2	VARCHAR(10)	Leverantörsnr leverantör nr 2, LEVREG
ARLEVNR3	VARCHAR(10)	Leverantörsnr leverantör nr 3, LEVREG
ARNETTOVIKT	DECIMAL(10,2)	Nettovikt
ARARTTYP	ENUM ('0','1','2','3')DEFAULT '0'	Artikeltyp 0=tillverkning lager 1=tillverkning ej lager 2=köp lager 3=köp ej lager
ARSTRUKT	ENUM (' ','B','T','T','F')DEFAULT ' '	Strukturkod för artikeln Blank = ingår ej i struktur B = Bottenartikel I = Ingår "mitt" i artikel T = Toppartikel F = Fantom, ingår "mitt" i artikel
ARURBENEMNING	VARCHAR(30)	Ursprungsbenämning
ARURLAND	VARCHAR(30)	Ursprungsland
ARURARTNR	VARCHAR(30)	Leverantörens artikelnr
ARTULLTAX	VARCHAR(10)	Tulltaxekod
ARVOLYM	DECIMAL(2,3)	Volym i kubikmeter
AROMRFAKTOR	INT(11)	Omräkningsfaktor

## BOKFAR

ARID	VARCHAR(2) Primary Key	Årtalsid. Anges med två bokstäver (AA-ZZ) För att skilja kontoplan och verifikat från olika
	år.	
BENAMNING	VARCHAR(25)	Förklara vad bokföringsåret kallas/omfattar.
ARSTART	DATE	Startdatum för bokföringsår.
ARSLUT	DATE	Slutdatum för bokföringsår
ARLAST	ENUM (J/N)	Flagga för om bokföringsåret är låst för
	registrering.	
BESKATTNINGSAR	VARCHAR(4)	Beskattningsår.
SENVERDAT	DATE	Senaste verifikationsdatum.
VERNR	INT(11)	Nästa verifikationsnummer
KTOPLAN	VARCHAR(15)	Kontoplan typ "BAS2000"/"EUBAS97"

Att använda bokstäverna AA-ZZ för att ange bokföringsår medger 676 bokföringsår.  
 Det blir lättare att ange brutet räkenskapsår, att ändra vilken tidsperiod bokföringsårets omfattningar mm.  
 I fältet BENAMNING kan man i klartext ange bokföringsårets sträckning, t ex 2002-08-01-2003-07-31.

Skapa tabellen med sqlscriptet "CreateTableBOKFAR.sql"

## BETVILKOR

BETVILKOR	VARCHAR(3)	Betalningsvillkor,
DAGAR	VARCHAR(3)	Antal dagar för betalningsvillkoret.
BESKRIVNING	VARCHAR(40)	Beskriver betalningsvillkoret.

Tabellen BETVILKOR beskriver olika betalningsvillkor som används inom företaget.

Skapa tabellen med sqlscriptet "CreateTableBETVILKOR.sql".

Skriptet laddar även tabellen med grunddata.

Grunddata finns i filen BETVILKORdata.txt

## FTGDATA

POSTTYP	VARCHAR(5)	Förkortning för innehållet i posten
POSTBESKR	VARCHAR(60)	Beskriver vad posten används till.
FDATA	TEXT	Postdata

Tabellen FTGDATA innehåller grunddata för företaget såsom postadress, besöksadress, leveransadress, företagsnr, momssatser, bokföringsperioder, nästa inköpsordernummer, nästa kundnummer, mm.

Skapa tabellen med sqlscriptet "CreateTableFTGDATA.sql".

Skriptet laddar även tabellen med grunddata, posttyper och postbeskrivningar men i övrigt är tabellen tom. Allt eftersom kan tabellen fyllas på med flera posttyper. Användaren måste dock fylla på med data i fältet FDATA.

Grunddata finns i filen FTGDATAdata.txt

Innehållet i FTGDATAdata.txt:

"ADR1","Postadress"  
"ADR2","Besöksadress"  
"ADR3","Godsadress"  
"TFN1","Telefonnummer till vx"  
"TFN2","Mobiltelefonnummer"  
"TFAX","Telefaxnummer"  
"TELEX","Telexnummer"  
"EML1","E-mailadress"  
"FTGNR","Företagsnummer"  
"MOMS1","Momssats 1"  
"MOMS2","Momssats 2"  
"MOMS3","Momssats 3"  
"MOMS4","Momssats 4"  
"MOMS5","Momssats 5"  
"BF1","Bokföringsperiod 1"  
"BF2","Bokföringsperiod 2"  
"BF3","Bokföringsperiod 3"  
"BF4","Bokföringsperiod 4"  
"BF5","Bokföringsperiod 5"  
"BF6","Bokföringsperiod 6"  
"BF7","Bokföringsperiod 7"  
"BF8","Bokföringsperiod 8"  
"BF9","Bokföringsperiod 9"  
"BF10","Bokföringsperiod 10"  
"BF11","Bokföringsperiod 11"  
"BF12","Bokföringsperiod 12"  
"BF13","Bokföringsperiod 13"  
"BVLK1","Betalningsvilkor 1"

“BVLK2”, ”Betaltvillkor 2”

“BVLK3”, ”Betaltvillkor 3”

## INKREG

INKORDNR	VARCHAR(10) not null,	Primary key
BESTTYP	ENUM('N','D','T','L','A'),	N=Normalbeställning, D=Direktbeställning, I=Inleveransbest, L=Legobest. A=Avrop.
ORDERDATUM	DATE,	
LEVNR	VARCHAR(10),	
LEVNAMN	VARCHAR (30),	
LEVADDRESS	VARCHAR(30),	
LEVPOSTNR	VARCHAR(6),	
LEVPOSTADR	VARCHAR(30),	
LEVLAND	VARCHAR(30),	
LEVVALUTA	VARCHAR(3),	
LEVBETVILLKOR	VARCHAR(50),	
LEVVILLKOR	VARCHAR(150),	
LEVSETT	VARCHAR(150),	
GODSMERKE	VARCHAR(30),	
KOMMENTAR	VARCHAR(250),	
BESTTEXT	VARCHAR(3),	
VARREF	VARCHAR (30),	
VARREFTFN	VARCHAR(15),	
VARREFFAX	VARCHAR(15),	
ERREEF	VARCHAR(20),	
LEVDATUM	DATE,	
KUNDNR	VARCHAR(30),	
FTGNAMN	VARCHAR(30),	Leveransadress
FTGADR	VARCHAR(30),	Leveransadress
FTGPOSTNR	VARCHAR(6),	Leveransadress
FTGPOSTADR	VARCHAR(30),	Leveransadress
SPRAKKOD	VARCHAR(3) DEFAULT 'sv',	
BEKREFTKOD	ENUM('H','D','E')DEFAULT 'E',	H=Hela best bekräftad, D=Delvis bekräftad, E=Ej bekräftad
ORDERSTATUS	ENUM('N','F','B','M')DEFAULT 'N',	N=Normal status, F=Fakturaavprickad best., B=Slutbehandlad, ska plockas bort, M=Makulerad
UTSKRIFTSKOD	ENUM('J','N')DEFAULT 'J',	
ORDERSUMMA	DECIMAL(10,2)	

## INKRADREG

INKORDNR	VARCHAR(10) not null, Primary key	
INKORDRADNR	INT(4)not null, Primary key	
ARTIKELNR	VARCHAR(30),	
BENEMNING	VARCHAR(30),	
LEVARTIKELNR	VARCHAR(30),	
LEVBENEMNING	VARCHAR(30),	
ENHET	VARCHAR(5),	
BESTANTAL	DECIMAL(10,2),	
LEVERERAT	DECIMAL(10,2),	
RESTNOTERAT	DECIMAL(10,2),	
INKPRIS	DECIMAL(10,2),	
LEVVECKA	VARCHAR(5),	Format ÅVV eller ÅÅVV
TORDNR	INT(6),	Tillverkningsordernr vid legobeställning
OPNR	INT(6)	Operationsnummer vid legobeställning



## LEVREG

Tabellen är än så länge preliminär.

LEVNR	VARCHAR(10) not null, Primary Key	
LEVORGNR	VARCHAR(12),	
LEVNAMN	VARCHAR(30) not null,	
LEVADDRESS	VARCHAR(30),	
LEVPOSTNR	VARCHAR(6),	
LEVPOSTADR	VARCHAR(30),	
LEVLAND	VARCHAR(30),	
LEVTFNNR	VARCHAR(15),	
LEVFXNR	VARCHAR(15),	
LEVTELEX	VARCHAR(10),	
LEVEMAIL	VARCHAR(30),	
LEVPOSTGIRONR	VARCHAR(10),	
LEVBANKGIRONR	VARCHAR(10),	
LEVREFERENT	VARCHAR(20),	
LEVREFTFN	VARCHAR(15),	
LEVMOMSKOD	VARCHAR(1) DEFAULT '1',	
LEVSKULD	DECIMAL(10,2),	
LEVKONTO	VARCHAR(4)	
LEVKUNDNR	VARCHAR(30),	// Kundnr hos leverantören
LEVVALUTA	CHAR(3),	
BETALVILKOR	VARCHAR(3)	

Skapa tabellen med CreateTableLEVREG.sql

## LEVRESK

Tabellen är än så länge preliminär.

LEVNR	VARCHAR(10) not null,	PRIMARY KEY
FAKTURANR	VARCHAR(20) not null,	PRIMARY KEY
REGDATUM	DATE,	
FAKTDATUM	DATE,	
EXPIREDATUM	DATE,	
FAKTTEXT	VARCHAR(100),	
BAR	VARCHAR(2),	
MOMSPROCENT	DECIMAL(2,2),	
VALUTA	CHAR(3),	
VALUTAKURS	DECIMAL(3,2),	
VALUTABELOPP	DECIMAL(10,2),	
LEVKTONR	VARCHAR(4),	
FAKTBELOPP	DECIMAL(10,2),	
MOMSKTONR	VARCHAR(4),	
MOMSBELOPP	DECIMAL(10,2),	
DEBETKONTONR	VARCHAR(4),	
DEBETBELOPP	DECIMAL(10,2),	
USERID	VARCHAR(8),	
VERNR	INT,	
BETALD	ENUM('J','N')DEFAULT 'N',	
BETALDDATUM	DATE	
OCRNR	VARCHAR(20)	

Skapa tabellen med CreateTableLEVRESK.sql

## KSTALLE

ARID	VARCHAR(2) Primary Key	Årtalsid. (AA – ZZ) För att skilja kontoplan och verifikat från olika år.
KSTALLE	VARCHAR(4)	Kostnadställe/Resultatenhet.
BENAMNING	VARCHAR(100)	Namn på kostnadsställe.

Skapa tabellen med sqlscriptet "CreateTableKSTALLE.sql"

## KTOPLAN

ARID	VARCHAR(2)	Primary Key	Årtalsid (AA-ZZ)
KTONR	VARCHAR(4)	Primary Key	Kontonummer.
BENAMNING	VARCHAR(100)		Beskrivning av kontot.
MANUELL	ENUM("J","N")	DEFAULT "J"	Flagga som bestämmer om man får boka manuellt på detta konto.
MOMSKOD	VARCHAR(4)		Berättar om hur kontot ska behandlas av momsrapporten.
SRUNR	INT(3)		En kod, fn 100-999, för export till deklarationsprogram, RSV.
KSTALLE	VARCHAR(4)		Om kontot skall användas för Kostnadställe/Resultatenhet.
PROJEKT	VARCHAR(4)		Om kontot skall användas för Projekt/Objekt hantering.
SUBKTO	VARCHAR(4)		Om konto kan ha SUBKONTO
KTOPLAN	VARCHAR(15)		Kontoplantyp "BAS2000"/"EUBAS97"
IB	DECIMAL(10,2)		Ingående balans.
UB	DECIMAL(10,2)		Utgående balans.

Skapa tabellen med sqlscriptet "CreateTableKTOPLAN.sql"

Ladda tabellen med sqlscriptet LoadKTOPLAN.sql. Data till laddningen finns i KTOPLANdata.txt.

## KUNDREG

KUNDNR	VARCHAR(10) not null,	PRIMARY KEY
KUNDORGNR	VARCHAR(12),	Ska eventuellt bort
NAMN	VARCHAR(60) not null,	Kundnamn
ADRESS	VARCHAR(30),	Kundadress
POSTNR	VARCHAR(6),	
POSTADR	VARCHAR(30),	
LAND	VARCHAR(30),	
TFNNR	VARCHAR(15),	Telefonnummer
EMAILADR	VARCHAR(30),	E-mailadress
FAXNR	VARCHAR(15),	
ERREFERENT	VARCHAR(30),	Kundens referent
ERREFTFNNR	VARCHAR(15),	Kundens referent's telefonnummer
ERREFEMAIL	VARCHAR(60),	Kundens referent's emailadress
SELJARE	VARCHAR(20),	Vår säljare
FRITEXT	VARCHAR(100),	Valfri text
VALUTA	VARCHAR(3),	
BETALVILLKOR	VARCHAR(3),	
LEVVILLKOR	VARCHAR(3),	Leveransvillkor
LEVSETT	VARCHAR(3),	Leveranssätt
DISTRIKT	VARCHAR(3),	
KUNDKATEGORI	VARCHAR(3),	
STDLEVPLATS	VARCHAR(3) DEFAULT '001',	Leveransadress
ORDERERKENNANDE	ENUM('J','N')DEFAULT 'J',	
PLOCKLISTA	ENUM('J','N')DEFAULT 'J',	
FOLJESEDEL	ENUM('J','N')DEFAULT 'J',	
KRAVBREV	ENUM('J','N')DEFAULT 'J',	
SPRAKKOD	VARCHAR(3),	
EXPAVGIFT	ENUM('J','N')DEFAULT 'J',	
FRAKTAVG	ENUM('J','N')DEFAULT 'J',	
KREDITLIMIT	DECIMAL(10,2),	
KREDITDAGAR	INT,	
KREDITKOD	VARCHAR(3),	
EXPORTKOD	VARCHAR(3),	
SKATTEKOD	VARCHAR(3),	
RABATTKOD	VARCHAR(3),	
DROJMALSRTA	ENUM('J','N')DEFAULT 'J',	
DROJMALSFAKTURA	ENUM('J','N')DEFAULT 'J',	
SAMLINGSFAKT	ENUM('J','N')DEFAULT 'J',	
SENASTEKRAVDATUM	DATE,	
SKULD	DECIMAL(10,2),	
ORDERSTOCK	DECIMAL(10,2)	

## LAGERSTELLEREG

Skapa med sqlscriptet "CreateTableLAGERSTELLEREG.sql"

ARLAGST	VARCHAR(1) not null PRIMARY KEY	Lagerställe
ARTIKELNR	VARCHAR(30) not null PRIMARY KEY	Artikelnummer/ArtikelID
		ARTIKELREG
ARLAGHYLLA	VARCHAR(10)	Lagerhylla
ARLAGSALDO	DECIMAL(10,2)	Lagersaldo för lagerställe
ARINVGRP	VARCHAR(3)	Inventeringsgrupp
ARABC	VARCHAR(2)	ABCkod
ARVALUTA	VARCHAR(3)	Valuta, VALUTA-registret
ARIPRIS	DECIMAL(10,2)	Inköpspris, senaste inköp
ARIKVANT0	DECIMAL(10,2)	Inköpskvantitet, senaste inköp
ARIKVANT1	DECIMAL(10,2)	Inköpskvantitet, näst senaste inköp
ARIKVANT2	DECIMAL(10,2)	Inköpskvantitet, näst,näst senaste inköp
ARKALKPRIS	DECIMAL(10,2)	Kalkylpris
ARBESTKVANT	DECIMAL(10,2)	Beställd kvantitet
ARBESTPUNKT	DECIMAL(10,2)	Beställningspunkt
AROMKOST	DECIMAL(10,2)	Omkostnader

# LEVILLKOR

Skapa med sqlscriptet "CreateTableLEVILLKOR.sql"

VILLKORSNR	VARCHAR(3) not null PRIMARY KEY	Idnummer för leveransvillkor
VILLKORSTEXT	VARCHAR(150)	Text för leveransvillkor

001	EXW	(Ex works?)
-----	-----	-------------

## LEVSETT

Skapa med sqlscriptet "CreateTableLEVSETT.sql"

LEVSETTNR	VARCHAR(3) not null PRIMARY KEY	Idnummer för leveranssätt
LEVSETTEXT	VARCHAR(150)	Text för leveranssätt



# PROGRAM

PRGNR	VARCHAR(3)	PRIMARY KEY	Löpnummer.
MENYAVD	VARCHAR(20)		Huvudmeny.
MENYGRP	VARCHAR(30)		Undermeny.
MENYTXT	VARCHAR(30)		Funktionsbeskrivning.
PROGRAM	VARCHAR(8)		Programnamn.

Tabellen PROGRAM innehåller de program som finns i OLFIX och används av OLFIXW.

Skapa och ladda tabellen med sqlscriptet "CreateTablePROGRAM.sql"

Grunddata finns i filen PROGRAMdata.txt

Innehållet i PROGRAMdata.txt:

```
"001","Administration","Användaradministration","Ny användare","ADDUSRW"
"002","Administration","Användaradministration","Ändra användarinfo","CHGUSRW"
"003","Administration","Användaradministration","Ta bort användare","DELUSRW"
"004","Administration","Användaradministration","Visa en användare","DSPUSRW"
"005","Administration","Användaradministration","Lista användare","LSTUSRW"
"006","Administration","Behörighetsadministration","Ny behörighet","ADDRGTW"
"007","Administration","Behörighetsadministration","Ändra behörighet",""
"008","Administration","Behörighetsadministration","Ta bort behörighet","DELRGTW"
"009","Administration","Behörighetsadministration","Visa behörighet",""
"010","Administration","Behörighetsadministration","Lista behörigheter","LSTRGTW"
"011","Administration","Funktionsadministration","Ny funktion","ADDFNCW"
"012","Administration","Funktionsadministration","Lista funktioner","LSTFNCW"
"013","Ekonomi","Bokföring","Registrera verifikation",""
"014","Ekonomi","Bokföring","Registrera ver. standard","BOKFORSW"
"015","Ekonomi","Kontoadministration","Nytt konto","ADDKTOW"
"016","Ekonomi","Kontoadministration","Ändra konto","CHGKTOW"
"017","Ekonomi","Kontoadministration","Ta bort konto",""
"018","Ekonomi","Kontoadministration","Visa konto","DSPKTOW"
"019","Ekonomi","Kontoadministration","Lista konton","LSTKTOW"
"020","Ekonomi","Kostnadställeadministration","Nytt kostnadställe","ADDKSTW"
"021","Ekonomi","Kostnadställeadministration","Ändra kostnadställe",""
"022","Ekonomi","Kostnadställeadministration","Ta bort kostnadställe",""
"023","Ekonomi","Kostnadställeadministration","Visa ett kostnadsställe","DSPKSTW"
"024","Ekonomi","Kostnadställeadministration","Lista kostnadsställen","LSTKSTW"
"025","Ekonomi","Valutaadministration","Ny valuta","ADDVALW"
"026","Ekonomi","Valutaadministration","Ändra valuta","CHGVALW"
"027","Ekonomi","Valutaadministration","Ta bort valuta","DELVALW"
"028","Ekonomi","Valutaadministration","Visa valuta","DSPVALW"
"029","Ekonomi","Valutaadministration","Lista valutor","LSTVALW"
"030","Ekonomi","Rapporter","Kontorapport","RPTKTOW"
"031","Ekonomi","Rapporter","Rapportgenerator","RPTGENW"
```

"032", "Ekonomi", "Räkenskapsår", "Nytt bokföringsår", "ADDBARW"  
"033", "Ekonomi", "Räkenskapsår", "Ändra bokföringsårsdata", "CHGBARW"  
"034", "Administration", "Företagsdata", "Ny post", "ADDFTGW"  
"035", "Administration", "Företagsdata", "Ändra post", "CHGFTGW"  
"036", "Administration", "Företagsdata", "Visa företagsdata", "DSPFTGW"  
"037", "Försäljning", "Kunddata", "Ny kund", "ADDKUW"  
"038", "Försäljning", "Kunddata", "Ny leveransadress för kund", "ADDLEVPW"  
"039", "Administration", "Leverantörsdata", "Ny leverantör", "ADDLEVW"  
"040", "Administration", "Leverantörsdata", "Visa en leverantör", "DSPLEVW"  
"041", "Administration", "Leverantörsdata", "Ändra leverantörsdata", "CHGLEVW"  
"042", "Ekonomi", "Bokföring", "Reg. leverantörsfaktura", "LEVFAKTW"  
"043", "Ekonomi", "Rapporter", "Leverantörsreskontra", "LEVRESKW"  
"044", "Ekonomi", "Rapporter", "Förfallna levfakturer", "ATTBETW"  
"045", "Ekonomi", "Rapporter", "Saldolista", "SDOLISW"  
"046", "Försäljning", "Kunddata", "Visa kunddata", "DSPKUW"  
"047", "Försäljning", "Kunddata", "Ändra kunddata", "CHGKUW"  
"048", "Försäljning", "Kunddata", "Lista kunder", "LSTKUW"  
"049", "Administration", "Företagsdata", "Byta företag", "BYTFTGW"  
"050", "Materialhantering", "Artikeldata", "Ny artikel", "ADDARW"  
"051", "Materialhantering", "Artikeldata", "Visa grunddata för en artikel", "DSPARW"  
"052", "Materialhantering", "Artikeldata", "Visa en artikels ekonomidata", "DSPAREW"

## RIGHTS

USERID	VARCHAR(8)	Användarid
TRNSID	VARCHAR(8)	Funktionsnamn.

Tabellen RIGHTS innehåller vilken användare som har vilken behörighet.

Skapa tabellen med sqlskriptet "CreateTableRIGHTS.sql"

Med sqlskriptet "LoadRIGHTS.sql" laddas tabellen med grunddata för att inledningsvis kunna använda OLFIX.

Grunddata finns i filen RIGHTSdata.txt

Innehållet i RIGHTSdata.txt:

```
"OLFIX","KTOCHK"  
"OLFIX","RGTADD"  
"OLFIX","RGTCHK"  
"OLFIX","RGTDEL "  
"OLFIX","TRHDADD"  
"OLFIX","TRNSADD"  
"OLFIX","TRNSDEL "  
"OLFIX","USERADD"  
"OLFIX","VERUPD"
```

## STDLEVPLATS

STDLEVPLATS	VARCHAR(3)	PRIMARY KEY
KUNDNR	VARCHAR(10)	PRIMARY KEY
ADRESS	VARCHAR(30)	
POSTNR	VARCHAR(6)	
POSTADR	VARCHAR(30)	
LAND	VARCHAR(30)	

STDLEVPLATS (Standardleveransplats) är avsedd för leveransadresser.

**STDLEVPLATS 001** är kundens förstahandsadress, den adress där kunden befinner sig eller uppger som sin förstahandsadress. STDLEVPLATS 002 kan registreras i samband med nyupplägg av kund.

STDLEVPLATS 002 och högre kan även registreras vid ett senare tillfälle.

Tabellen skapas med scriptet CreateTableSTDLEVPLATS.sql

## TEXTREG

TEXTNR	VARCHAR(3)	Primary Key	Funktionsnamn.
TXT	TEXT		Data, löpande text.

Tabellen TEXTREG lagrar diverse texter som används av olika program.

Skapa tabellen med sqlscriptet "CreateTableTEXTREG.sql"

## TRANSID

TRNSID	VARCHAR(8)	Primary Key	Funktionsnamn.
TRNSTXT	VARCHAR(60)		Beskrivning av funktionen.

Tabellen TRANSID tillsammans med tabellen USR ger möjlighet att någorlunda enkelt skapa behörighetsregler till OLFIX.

Skapa tabellen med sqlscriptet "CreateTableTRANSID.sql"

Ladda tabellen med sqlskriptet "LoadTRANSID.sql"

## TRHD

TRNSNR	INT	Primary Key	Autoincrement
TRNSID	VARCHAR(8)		
TID	VARCHAR(20)		Format YYYY-MM-DD_hh:mm:ss
USERID	VARCHAR(8)		Användare som gjort transaktionen
TRNSDATA	VARCHAR(255)		Transaktionsdata

TRNSDATAAs delar avskiljs med ; (semikolon).

Skapa tabellen med sqlscriptet "CreateTableTRHD.sql"

## USR

USERID	VARCHAR(8)	Primary Key	Användarid
NAMN	VARCHAR(30)		Namnet på användaren
AVD	VARCHAR(30)		Den avdelning användaren tillhör.
GRUPP	VARCHAR(10)		Affärsgrupp användaren tillhör.

GRUPP ska icke förväxlas med vad operativsystemet menar med grupp.  
Tabellen USR innehåller uppgifter på användare i OLFIX.

Skapa tabellen med sqlskriptet "CreateTableUSR.sql"

Med sqlskriptet "LoadUSR.sql" laddas tabellen med grunddata för att man inledningsvis kunna använda OLFIX.

Grunddata finns i filen USRdata.txt

Innehållet i USRdata.txt:

"OLFIX","Olfix Superuser","IT","Stab"



## VALUTA

VALUTAID	VARCHAR(3) Primary Key	Valuta.
LAND	VARCHAR(15)	Valutaland
SALJ	DECIMAL(3,2)	Säljkurs i kronor.
KOP	DECIMAL(3,2)	Köpkurs i kronor.
BETECKNING	VARCHAR(15)	Valutans namn.

Skapa tabellen med sqlscriptet "CreateTableVALUTA.sql"  
Initialt laddas följande data:

"DKK","Danmark","1.22","1.22","Kronor"  
"NOK","Norge","1.23","1.23","Kronor"  
"NYZ","Nya Zeeland","4.45","4.45","Dollar"  
"SAR","Saudiarabien","2.40","2.40","Riyal"  
"HKD","Honkong","0.0","0.0","Dollar"  
"MYR","Malaysia","2.36","2.36","Ringgit"  
"SGD","Singapore","5.08","5.08","Dollar"  
"CAD","Kanada","5.66","5.66","Dollar"  
"AUD","Australien","5.03","5.03","Dollar"  
"USD","USA","8.97","8.97","Dollar"  
"JPY","Japan","7.38","7.38","Yen"  
"GBP","Storbritanien","14.26","14.26","Pund"  
"EUR","Europa","9.08","9.08","Euro"  
"CHF","Schweiz","0.00","0.00","France"

## VERHUVUD

VERNR	INT(11)	Primary Key	Verifikationsnummer.
ARID	VARCHAR(2)	Primary Key	Årtalsid. (AA-ZZ)
VERDATUM	DATE		Verifikationsdatum.
REGDAT	DATE		Datum för när man registrerade
		verifikatet.	
DEBET	DECIMAL(10,2)		Summa debetposter på verifikatet.
KREDIT	DECIMAL(10,2)		Summa kreditposter på verifikatet.
VERTEXT	VARCHAR(60)		Verifikationstext.
KORRIGERAR	INT(11)		Innehåller vernr. Om detta verifikat
			korrigerar ett annat felaktigt verifikat
			så att man kan se att felet är rättat.
KORRIGERAD	INT(11)		Innehåller vernr. Om detta verifikat är
			korrigerat av ett annat verifikat.
USERID	VARCHAR(8)		Användarid på den som bokfört
		verifikatet.	

Skapa tabellen med sqlscriptet "CreateTableVERHUVUD.sql"

## VERRAD

VERNR	INT(11)	Primary Key	Verifikationsnummer.
RADNR	SMALLINT(6)	Primary Key	Radnummer
ARID	VARCHAR(2)	Primary Key	Årtalsid. (AA-ZZ)
KTONR	VARCHAR(4)		Kontonummer.
BELOPP	DECIMAL(10,2)		Radbelopp.
KSTALLE	VARCHAR(4)		Belastat kostnadsställe.
PROJEKT	VARCHAR(4)		Belastat projekt.
SUBKTO	VARCHAR(4)		Underkonto.
DEFINITIV	ENUM('J','N')DEFAULT 'N'		Flagga som berättar om att raden
		är sparad.	
STRUKEN	ENUM('J','N')DEFAULT 'N'		Flaggasom berättar att raden är
		struken.	

Skapa tabellen med sqlscriptet "CreateTableVERRAD.sql"

## Credits

### Deltagare i projekt OLFIX

Jens Odsvall	<a href="mailto:jens.odsvall@bonetmail.com">jens.odsvall@bonetmail.com</a>
Joakim Gustavsson	<a href="mailto:joakim@nibor.se">joakim@nibor.se</a>
Jonas Björk	<a href="mailto:jonas@mbs.nu">jonas@mbs.nu</a>
Anders Forsgren	spoky_@hotmail.com
Martin Johansson	<a href="mailto:martin.jo@home.se">martin.jo@home.se</a>
Jan Pihlgren	<a href="mailto:jan@pihlgren.se">jan@pihlgren.se</a>
Yann Vernier	<a href="mailto:yann@algonet.se">yann@algonet.se</a>
Kurt Svensson	<a href="mailto:Kurt.Svensson@ksam.se">Kurt.Svensson@ksam.se</a>
Sune Gustavsson	<a href="mailto:sunegustafsson@telia.com">sunegustafsson@telia.com</a>
Larry Well	<a href="mailto:be_well@linuxmail.org">be_well@linuxmail.org</a>
Andreas Westerlund	<a href="mailto:andreas.westerlund@multi.fi">andreas.westerlund@multi.fi</a>
Leif Larsson	<a href="mailto:budgetdata@comhem.se">budgetdata@comhem.se</a>

## Appendix A

### MySQL databasens fysiska läge.

Databasens fysiska läge:	Utrymme	Ant filer	Ant tabeller
/var/lib/mysql/ mysql/tabell.MYD mysql/tabell.MYI mysql/tabell.frm	62.2 Kb	18 filer	7 tabeller
olfix/tabell.MYD olfix/tabell.MYI olfix/tabell.frm	151.3 Kb	36 filer	12 tabeller (2003-03-07)
test			

### Installation av MySQL

The basic commands you must execute to install and use a MySQL binary distribution are:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> cd /usr/local
shell> gunzip < /path/to/mysql.VERSION-OS.tar.gz | tar xvf -
shell> ln -s mysql.VERSION-OS mysql
shell> cd mysql
shell> scripts/mysql_install_db
shell> chown -R root /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/data
shell> chgrp -R mysql /usr/local/mysql
shell> chown -R root /usr/local/mysql/bin
shell> bin/safe_mysqld -user=mysql &
```

Se till att mysqld startas automatiskt vid start av datorn.

För mera information se MySQLReferenceManual, <http://www.mysql.com/documentation/>

### MySQL Navigator

Följande 2 filer ska finnas i biblioteket /usr/local/mysqlnavigator-version.binary

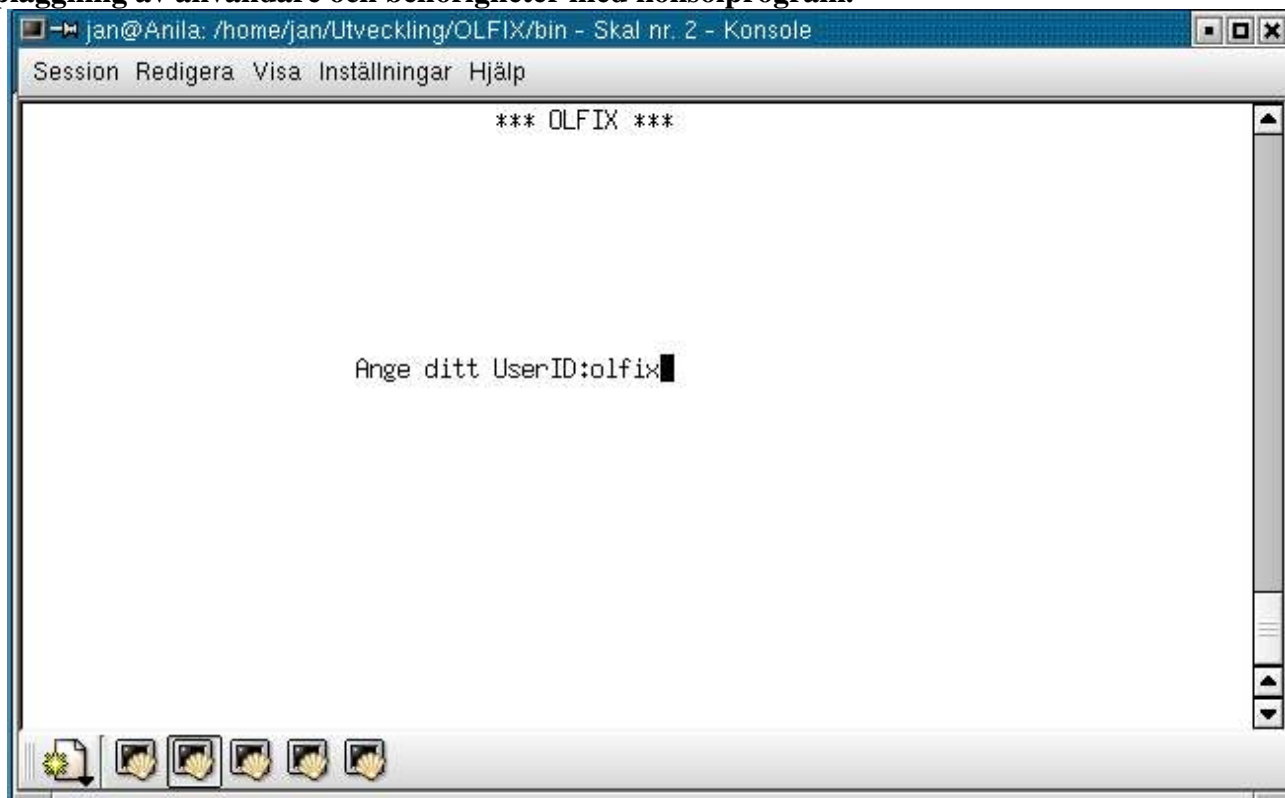
```
libstdc++libc6.2-2.so.3
mysqlnavigator-static
```

MySQL Navigator startas som root med commando;

```
shell> /usr/local/mysqlnavigator-static &
```

## Appendix B

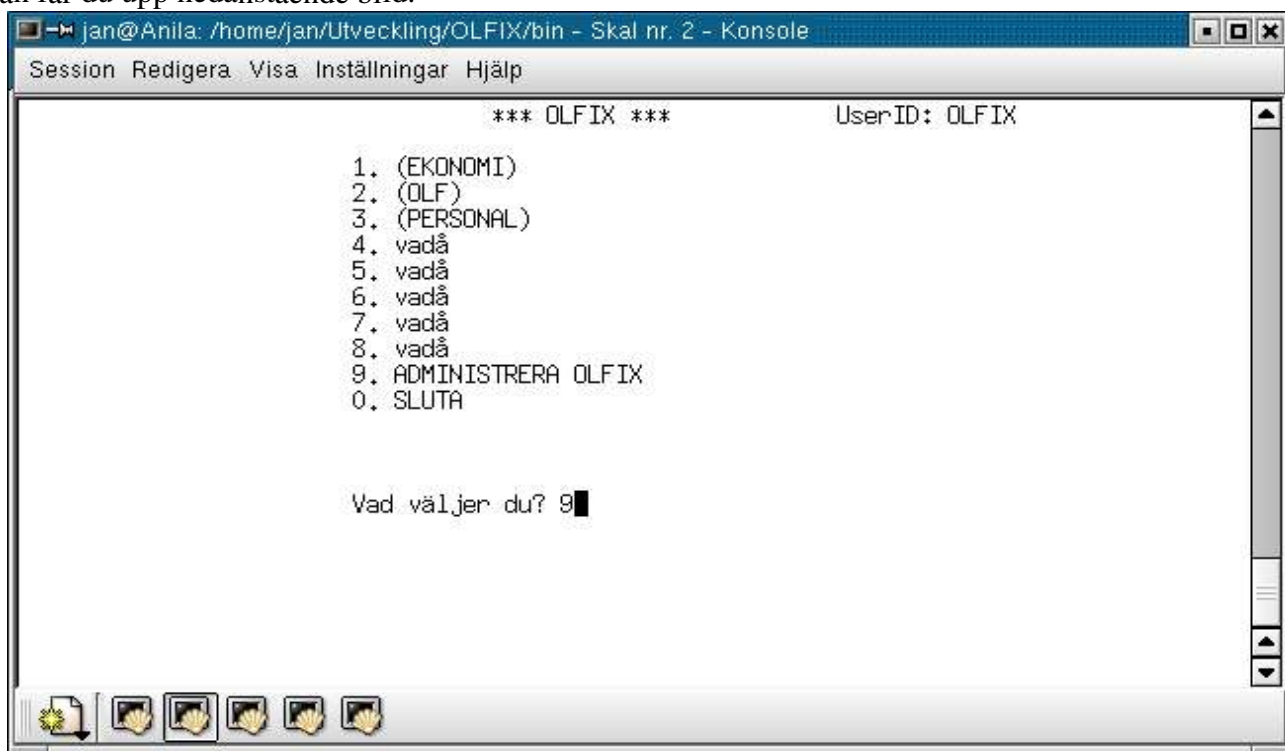
### Uppläggning av användare och behörigheter med konsolprogram.



Starta programmet OLFIX och du får upp denna bild.

Som userID **ska** du ange **olfix**. Då får du behörighet att lägga upp användare och behörigheter.

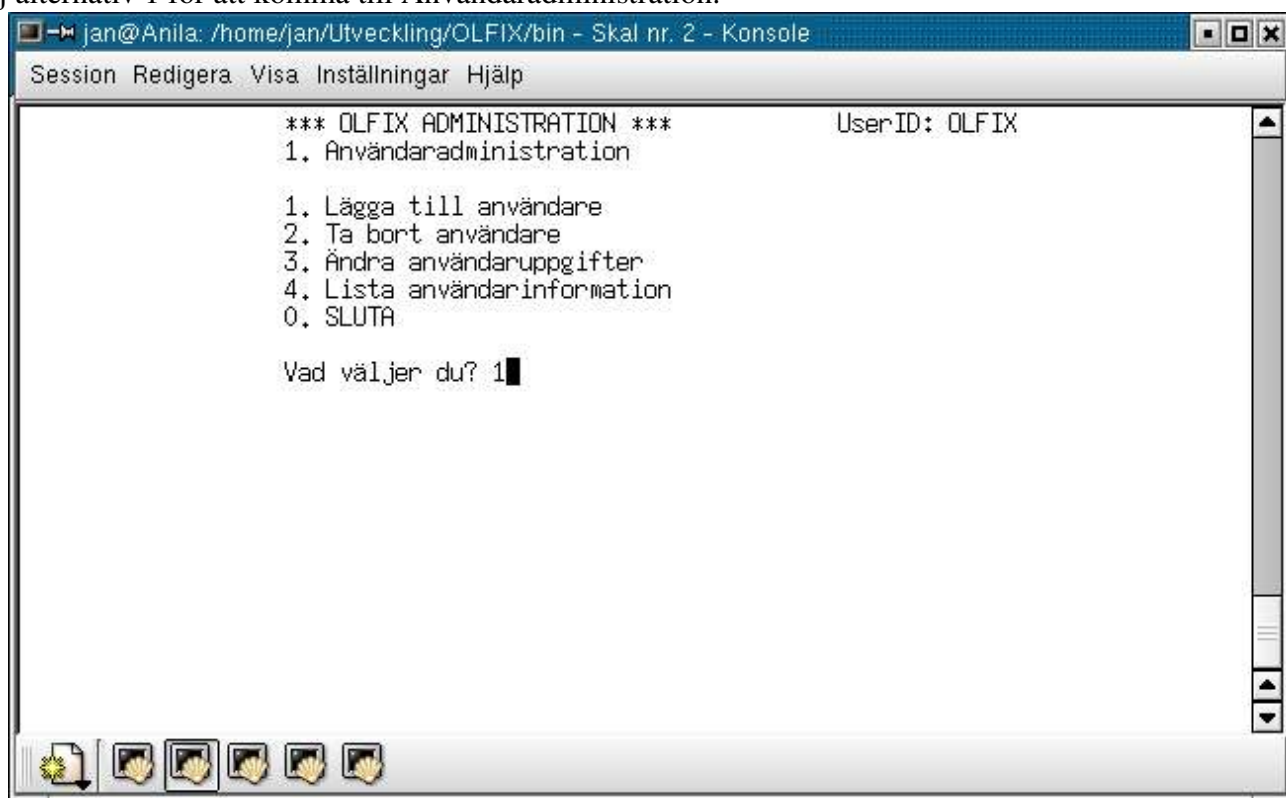
Sedan får du upp nedanstående bild.



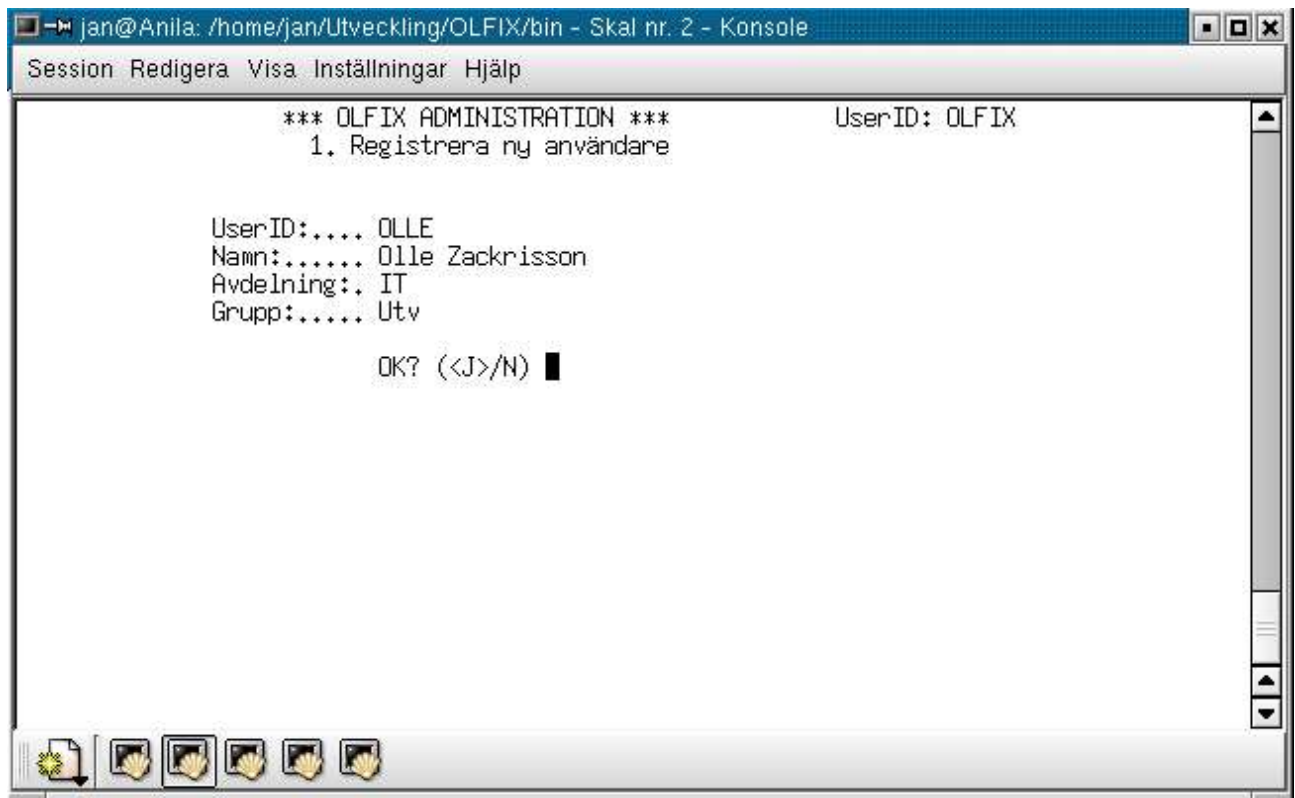
Välj funktion 9 och du kommer till nedanstående bild.



Välj alternativ 1 för att komma till Användaradministration.



Välj alternativ 1 för att lägga upp en ny användare.



Fyll i de uppgifterna.

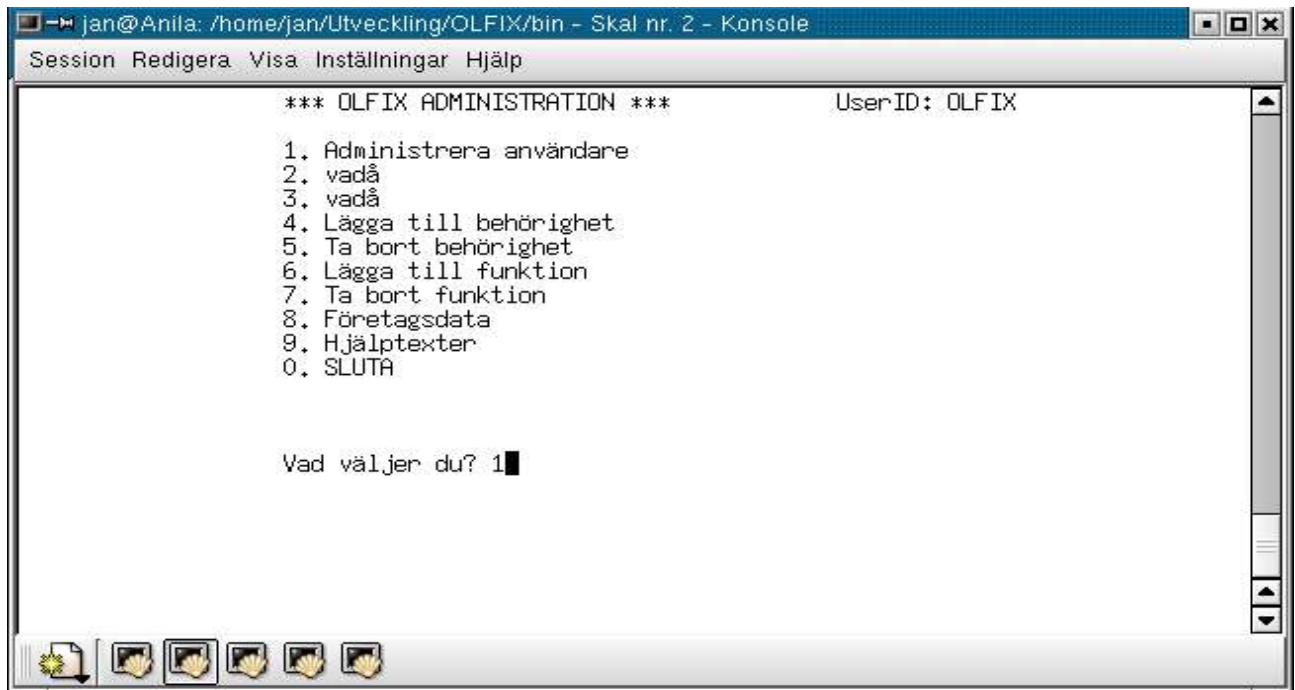
I fältet UserID ska du fylla i det userid som används för att logga in på datorn.

OLFIX kommer nämligen att hämta upp userid på den som loggat in på datorn och jämföra detta mot behörighetslistan. (tabellen RIGHTS i databasen.)

I fältet Namn skriver du in användarens namn.

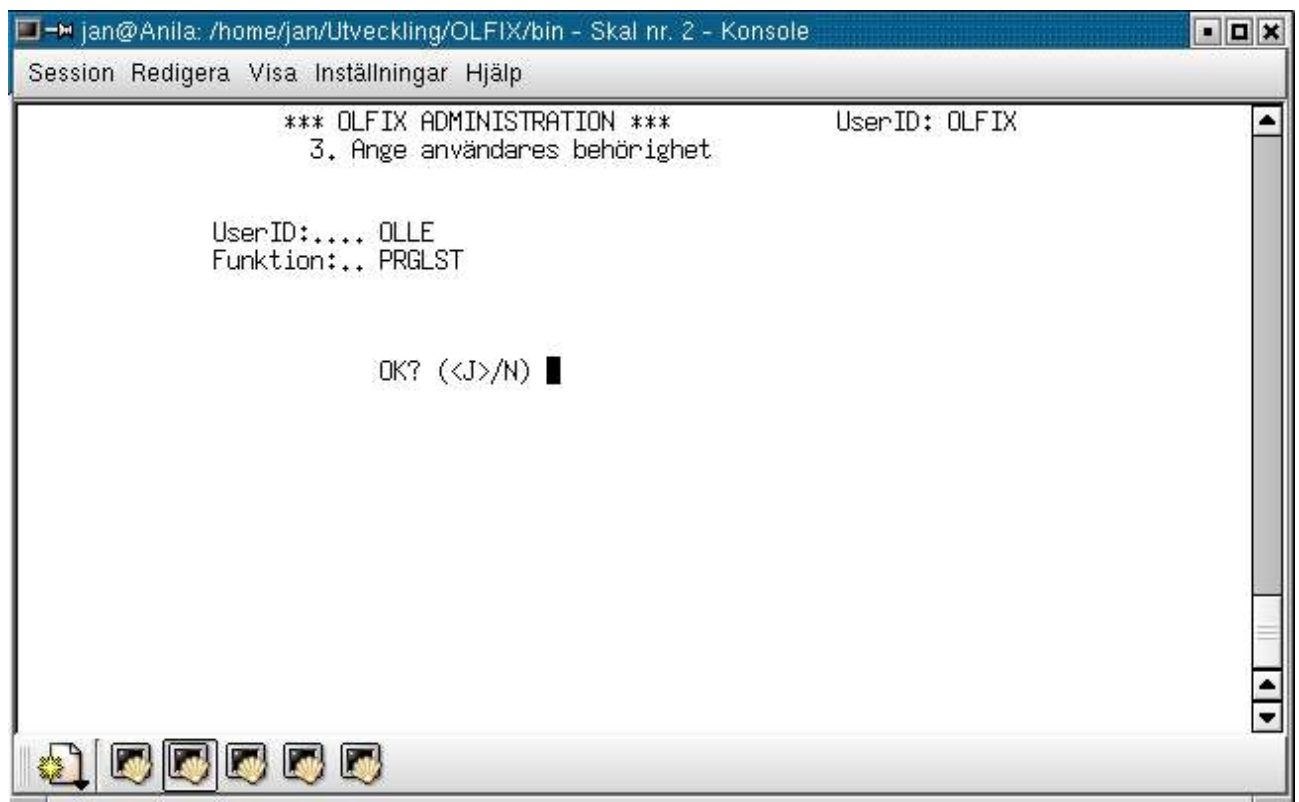
I fälten Avdelning och Grupp kan du fylla i vad som gäller i din organisation..

Backa sedan till baka till nedanstående bild.



Välj alternativ 4 för att registrera behörigheter.





Här måste du upprepa registreringen för varje program/funktion som användaren ska ha tillgång till.

## Appendix C

### Felmeddelanden



“Connection error 1045: Access denied for user '[olfix@localhost](#)' (Using password YES)”

Detta fel beror på felaktigt password för “olfix”.

De övriga felmeddelandena är en följd av detta.

Detta fel ska normalt inte uppträda, men kan uppstå i samband med installation av OLFIX .

En åtgärd som bör vidtagas, efter rättning av password, är att starta om maskinen.

```
[jan@localhost jan]$ mysql -u jan -p;  
Enter password:  
ERROR 1045: Access denied for user: 'jan@localhost' (Using password: YES)
```

Felaktigt password för “jan”.

En annan orsak kan vara att “jan” inte har behörighet att arbeta med mysql.

```
mysql> use olfix;  
ERROR 1044: Access denied for user: '@localhost' to database 'olfix'
```

```
[jan@localhost jan]$ mysql  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 1 to server version: 3.23.52
```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql> select * from PROGRAM;  
ERROR 1046: No Database Selected  
mysql> use olfix;  
ERROR 1044: Access denied for user: '@localhost' to database 'olfix'  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| mysql    |  
| olfix    |  
| test     |  
+-----+
```

3 rows in set (0.10 sec)

mysql> exit  
Bye

[jan@localhost jan]\$ mysql -u jan -p;  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 5 to server version: 3.23.52

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use olfix;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A

Database changed  
mysql>

## Appendix D

### Manuell skapande av databasen olfix.

Efter att ha editerat filerna enligt Installationsanvisningarna ska du köra ett antal sql-scriptenligt följande:

- [root@localhost sql]#mysql<add\_user\_dittnamn.sql>mysql.out
- [root@localhost sql]#mysql<add\_user\_olfix.sql>mysql.out
- [root@localhost sql]#mysql<CreateAll.sql>mysql.out
- [root@localhost sql]#mysql<CreateAllolfixtst.sql>mysql.out
- [root@localhost sql]#mysql<LoadAll.sql>mysql.out
- [root@localhost sql]#mysql<LoadAll\_olfixtst.sql>mysql.out
- [root@localhost sql]#mysql<LoadMinimumRIGHTSdata.sql>mysql.out