

Copyright Jan Pihlgren 2002

# Order/Lager/Fakturering för Linuxsystem

## Teknisk manual

Version 0.61

2006-03-22

\*\*\*\*\*

\* \* \* \* \*

\* 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.  
\* \* \*

\*\*\*\*\*

\* Copyright (c) 2003, 2004, 2005 Jan Pihlgren. \* \*

\* Permission is granted to copy, distribute and/or modify this document \*  
\* under the terms of the GNU Free Documentation License, Version 1.2 \*  
\* or any later version published by the Free Software Foundation; \*  
\* with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.\*  
\* A copy of the license is included in the section entitled "GNU \*  
\* Free Documentation License". \*  
\*\*\*\*\*

# Innehåll

Målgrupp.....	12
Filosofin bakom konstruktionen .....	13
OLFIX grundkonstruktion.....	13
Strukturen för det grafiska gränsnittet.....	14
Programnamnsval.....	14
Kataloger.....	14
Säkerhet.....	15
Dokumentation.....	15
Förutsättningar för OLFIX.....	16
Installationsanvisningar.....	17
Bibliotek.....	17
Installation.....	17
Skapa databasen olfix.....	18
Script.....	19
Behörighet.....	19
Resursfilen .olfixrc.....	20
Administration.....	21
Upplägg av nya användare.....	21
Tilldela behörighet till användare.....	21
Tillägg av nya program.....	21
Tillägg av nya funktioner.....	21
GUI-program i OLFIX .....	22
OLFIXW.....Grafiskt menyprogram.....	24
Behörighetskrav:.....	26
ADDBARW.....Nytt bokföringsår.....	27
Behörighetskrav:.....	29
ADDBETVW.....Nytt betalningsvillkor.....	30
Behörighetskrav:.....	32
ADDNCW.....Ny funktion .....	33
Behörighetskrav:.....	35
ADDFORW.....Nya företagsdatabaser .....	36
Funktionsbeskrivning.....	38
Behörighetskrav:.....	38
ADDFTGW.....Nya företagsdata .....	39
Behörighetskrav:.....	42
ADDINKW.....Registrera inköpsorder.....	43
Behörighetskrav:.....	48
ADDKSTW.....Nytt kostnadsställe.....	49
Behörighetskrav:.....	51
ADDKTOW.....Nytt Konto .....	52
Behörighetskrav:.....	54
ADDKUW ..... Ny kund.....	55
Behörighetskrav:.....	58
ADDLEVW ..... Ny leverantör.....	59
Behörighetskrav:.....	61
ADDLEVPW ..... Ny standardleveransplats.....	63
Behörighetskrav:.....	65
ADDLEVSW ..... Nytt leveranssätt.....	66

Behörighetskrav:.....	68
ADDLEVW ..... Nytt leveransvillkor.....	69
Behörighetskrav:.....	71
ADDORDW ..... Ny kundorder .....	73
Behörighetskrav:.....	76
ADDPKDW ..... Ny produktgrupp.....	80
Behörighetskrav:.....	83
ADDRGTW.....Ny behörighet .....	84
Behörighetskrav:.....	86
ADDTXTW.....Ny text till TEXTREG .....	88
Behörighetskrav:.....	90
ADDUSRW.....Ny användare .....	91
Behörighetskrav:.....	93
ADDVALW.....Ny valuta .....	94
Behörighetskrav:.....	97
ATTBETW.....Lista obetalda leverantörsfakturor .....	98
Behörighetskrav:.....	102
BALRPTW.....Balansrapport.....	103
Behörighetskrav:.....	105
BOKFORSW.....Bokföring standard.....	106
Behörighetskrav:.....	110
BYTFTGW ..... Byta företag.....	112
Flera företag.....	112
CHGBARW.....Ändra bokföringsårsdata.....	115
Behörighetskrav:.....	118
CHGBETVW.....Ändra betalningsvillkor.....	120
Behörighetskrav:.....	123
CHGFTGW.....Ändra företagsdata.....	124
Behörighetskrav:.....	126
CHGLEVW.....Ändra leverantörsdata.....	127
Behörighetskrav:.....	130
CHGORDW.....Ändra kundorder.....	131
Behörighetskrav:.....	134
CHGPRISW.....Uppdatera priser.....	135
Behörighetskrav:.....	137
CHGUSRW.....Ändra användardata.....	138
Behörighetskrav:.....	140
CHGKTOW.....Ändra kontodata.....	141
Behörighetskrav:.....	144
CHGKUW.....Ändra kunddata.....	146
Behörighetskrav:.....	149
CHGVALW.....Ändra valutadata.....	150
Behörighetskrav:.....	152
DAGBOKW.....Dagbok .....	154
Behörighetskrav:.....	156
Exempel.....	158
DELRGTW.....Radera behörighet .....	159
Behörighetskrav:.....	161
DELTXTW.....Radera text .....	162
Behörighetskrav:.....	164

DELUSRW.....Radera användare .....	165
Behörighetskrav:.....	168
DELVALW.....Radera valuta .....	169
Behörighetskrav:.....	172
DSPARW.....Visa artikel, grunddata .....	174
Behörighetskrav:.....	177
DSPAREW.....Visa artikel, ekonomidata .....	178
Behörighetskrav:.....	181
DSPBARW.....Visa bokföringsår .....	182
Behörighetskrav:.....	184
DSPFTGW.....Visa en företagsdata.....	185
Behörighetskrav:.....	187
DSPINKW.....Visa en inköpsorder.....	188
Behörighetskrav:.....	190
DSPKSTW.....Visa kostnadsställdata.....	191
Behörighetskrav:.....	193
DSPKUW.....Visa kunddata.....	195
Behörighetskrav:.....	197
DSPLEVW.....Visa leverantörsdata.....	198
Behörighetskrav:.....	201
DSPORDW.....Visa en kunder.....	202
Behörighetskrav:.....	205
DSPTXTW.....Visa en post i TEXTREG.....	207
Behörighetskrav:.....	209
DSPUSRW.....Visa användardata.....	210
Behörighetskrav:.....	212
DSPVALW.....Visa valutainformation.....	213
Behörighetskrav:.....	216
HUVBOKW.....Huvudbok.....	217
Behörighetskrav:.....	220
KUFAKTW..... Fakturering .....	221
Behörighetskrav:.....	228
KUFAKTBW..... Registrera kundfakturor.....	229
KURESKW..... Kundreskontra .....	232
Behörighetskrav:.....	234
LEVFAKTW.....Registrera fakturor .....	235
Behörighetskrav:.....	239
LSTBARW.....Lista räkenskapsår .....	241
Behörighetskrav:.....	243
LSTFNCW.....Lista funktioner .....	244
Behörighetskrav:.....	246
LSTFTGW.....Lista företagsdata .....	247
Behörighetskrav:.....	249
LSTINKW.....Beställningsstock .....	250
Behörighetskrav:.....	252
LSTKSTW.....Lista kostnadsställen.....	253
Behörighetskrav:.....	255
LSTKTOW.....Lista konton .....	256
Behörighetskrav:.....	258
LSTKUW.....Lista kunder.....	259

Behörighetskrav:.....	261
LSTLEVPW....Lista kunders leveransplatser.....	262
Behörighetskrav:.....	264
LSTLEVSW....Lista leveranssätt.....	265
Behörighetskrav:.....	267
LSTLEVW.....Lista leveransvillkor.....	268
Behörighetskrav:.....	270
LSTORDW....Lista kundorder.....	271
Behörighetskrav:.....	273
LSTPGMW....Lista program.....	274
Behörighetskrav:.....	276
LSTPKDW....Lista produktgrupper.....	277
Behörighetskrav:.....	279
LSTRGTW....Lista behörigheter.....	280
Behörighetskrav:.....	282
LSTTXTW....Lista textregistrets poster .....	283
Behörighetskrav:.....	285
LSTUSRW....Lista alla användare .....	286
Behörighetskrav:.....	288
LSTVALW....Lista alla valutor.....	289
Behörighetskrav:.....	291
OLFIXHLP....Online hjälp.....	292
PLORDW....Skriv ut plocklista.....	294
Behörighetskrav:.....	296
PLCHGW..... Pricka av plocklista.....	298
Behörighetskrav:.....	300
PRTINKW....Skriv ut beställning.....	301
Behörighetskrav:.....	304
RPTGENW....Rapportgenerator.....	305
Behörighetskrav:.....	307
RPTSIEW....Skapa SIE-fil.....	308
Flödesschema.....	309
Behörighetskrav:.....	311
SDOLISW....Saldolista.....	312
Behörighetskrav:.....	319
Flödesschema.....	320
SRCHARW.... Söka artiklar.....	322
Behörighetskrav:.....	324
SRCHKUW....Söka kunder.....	325
Behörighetskrav:.....	327
Konsolprogram i OLFIX.....	328
ADMIN.....	329
BOKF .....	336
FTGADM.....	337
OLFIX (konsolprogram) .....	340
REDOV.....	343
Standardrapporter i OLFIX.....	350
Leverantörsfakturor förfallna till betalning.....	350
Kontorrapport, endast på skärm.....	350
Leverantörsreskontrarapport, endast skärm.....	350

Huvudbok.....	350
Saldolista.....	350
Balansräkning.....	350
Funktionerna i OLFIX.....	351
ARADD.....	361
Interface:.....	362
ARCHK.....	363
Interface:.....	363
ARLST.....	365
Interface:.....	365
ARLSPK.....	367
Interface:.....	367
ARLSTL.....	369
Interface:.....	369
ARSRCH.....	371
Interface:.....	372
AR2UPD.....	373
Interface:.....	373
ARICHK.....	375
ATTBET.....	376
Interface:.....	376
BARADD.....	378
Interface:.....	378
BARCHG.....	380
Interface:.....	380
BARCHK.....	382
Interface:.....	382
BARDSP.....	384
Interface:.....	384
BARFND.....	386
Interface:.....	386
BARLST.....	388
Interface:.....	388
BETADD.....	389
Interface:.....	389
BETCHG.....	390
Interface:.....	390
BETDSP.....	392
Interface:.....	392
BETLST.....	394
Interface:.....	394
DBCHK.....	396
Interface:.....	396
DBOKRPT.....	397
Interface:.....	397
FORADD.....	398
Interface:.....	398
FORCHK.....	399
Interface:.....	399
FORDSP.....	400

Interface:	400
FORLST.....	401
Interface:	401
FTGADD.....	402
Interface:	402
FTGDSP.....	403
Interface:	404
FTGLIS.....	405
Interface:	406
FTGLST.....	407
Interface:	407
FTGUPD.....	408
Interface:	408
HBOKRPT.....	410
Interface:	410
INKADD.....	411
Interface:	412
INKLST.....	413
Interface:	414
INKRADD.....	415
Interface:	416
INKHDSP.....	417
Interface:	418
INKRLST.....	419
Interface:	419
KRESADD.....	421
Interface:	421
KRESLST.....	423
Interface:	423
KSTADD.....	425
Interface:	425
KSTCHK.....	427
Interface:	427
KSTDSP.....	428
Interface:	428
KSTLST.....	429
Interface:	429
KTOADD.....	430
Interface:	430
KTOCHG.....	432
Interface:	432
KTOCHK.....	434
Interface:	434
KTODSP.....	436
Interface:	436
KTOLST.....	438
Interface:	438
KTORPT.....	440
Interface:	441
KTOUPD.....	442

KTOVIEW.....	443
Interface:.....	444
KUADD.....	445
Interface:.....	446
KUCHG.....	447
Interface:.....	449
KUCHK.....	450
Interface:.....	450
KUDSP.....	452
Interface:.....	452
KULST.....	454
Interface:.....	454
KUSRCH.....	456
Interface:.....	457
LEVADD.....	458
Interface:.....	460
LEVCHG.....	461
Interface:.....	462
LEVDSP.....	463
Interface:.....	463
LEVLST.....	465
Interface:.....	465
LEVPDSP.....	467
Interface:.....	467
LEVPLST.....	469
Interface:.....	469
LEVSADD.....	471
Interface:.....	471
LEVSDSP.....	472
Interface:.....	472
LEVSLST.....	474
Interface:.....	474
LEVVADD.....	475
Interface:.....	475
LEVVDSP.....	476
Interface:.....	476
LEVVLST.....	478
Interface:.....	478
LRESADD.....	479
Interface:.....	480
LRESRPT.....	481
Interface:.....	481
ORDADD.....	483
Interface:.....	483
ORDCHG.....	485
Interface:.....	486
ORDRADD.....	487
Interface:.....	487
ORDRCHG.....	489
Interface:.....	490

ORDCHK.....	491
Interface:.....	491
ORDDSP.....	493
Interface:.....	493
ORDRDSP.....	495
Interface:.....	495
ORDLST.....	497
Interface:.....	497
ORDLST2.....	499
Interface:.....	499
ORDUPD.....	501
Interface:.....	501
ORADUPD.....	503
Interface:.....	503
PICKADD.....	505
Interface:.....	505
PICKDSP.....	507
Interface:.....	507
PICKLST.....	509
Interface:.....	510
PKDADD.....	511
Interface:.....	511
PKDDSP.....	513
Interface:.....	513
PKDLST.....	515
Interface:.....	515
PRGLST.....	517
Interface:.....	518
PRISDSP.....	519
Interface:.....	520
PRISUPD.....	521
Interface:.....	522
PRTAPI.....	523
RGTADD.....	524
Interface:.....	524
RGTCHK.....	526
Interface:.....	526
RGTDEL.....	528
Interface:.....	528
RGTDSP.....	530
Interface:.....	531
RGLST.....	532
Interface:.....	532
RPTCRE.....	534
Interface:.....	534
SIEEXPK.....	535
Interface:.....	535
SIEEXPR.....	537
Interface:.....	537
SIEEXPV.....	539

Interface:.....	540
SLPADD.....	541
Interface:.....	541
STYRMAN.....	543
Interface:.....	544
TRHDADD.....	545
Interface:.....	545
TRNSADD.....	546
Interface:.....	546
TRNSLST.....	547
Interface:.....	547
TXTADD.....	548
Interface:.....	548
TXTDEL.....	549
Interface:.....	549
TXTDSP.....	550
Interface:.....	550
TXTLST.....	551
Interface:.....	551
USERADD.....	552
Interface:.....	553
USERCHG.....	554
Interface:.....	554
USERDEL.....	556
Interface:.....	556
USERDSP.....	558
Interface:.....	558
USERLST.....	560
Interface:.....	561
VALADD.....	562
Interface:.....	563
VALCHG.....	564
Interface:.....	565
VALDEL.....	566
Interface:.....	566
VALDSP.....	567
Interface:.....	568
VALLST.....	569
Interface:.....	569
VERUPD.....	570
Interface:.....	571
WKUDSP .... Visa webbkundsdata.....	572
Interface:.....	573
WRREC .... vernr.txt.....	574
Tabeller i OLFIXs databas .....	576
ARTIKELREG.....	579
BOKFAR.....	580
BETVILKOR.....	581
DATABASES.....	582
FTGDATA.....	583

INKREG.....	585
INKRADREG.....	587
LEVREG.....	588
LEVRESK.....	589
KSTALLE.....	590
KTOPLAN.....	591
KUNDREG.....	592
KUNDKATEGORI.....	594
KURESK.....	595
LAGERSTELLEREG.....	596
LEVPRISER.....	597
LEVVILLKOR.....	598
LEVSETT.....	599
ORDERREG.....	601
ORDERRADREG.....	603
PASSW utgår .....	604
PLOCKLISTEREG.....	605
PRISLISTA.....	606
PRODUKTGRUPP.....	607
PROGRAM.....	608
RIGHTS.....	610
STDLEVPLATS.....	611
TEXTREG.....	612
TRANSID.....	613
TRHD.....	614
USR.....	615
VALUTA.....	616
VERHUVUD.....	617
VERRAD.....	618
Credits.....	619
Appendix A.....	620
MySQL databasens fysiska läge.....	620
Installation av MySQL .....	620
MySQL Navigator.....	620
Appendix B.....	621
Uppläggning av användare och behörigheter med konsolprogram.....	621
Appendix C.....	626
Felmeddelanden.....	626
Appendix D.....	629
Manuell skapande av databasen olfix.....	629
Appendix E. GNU Free Documentation License.....	630

## **Målgrupp**

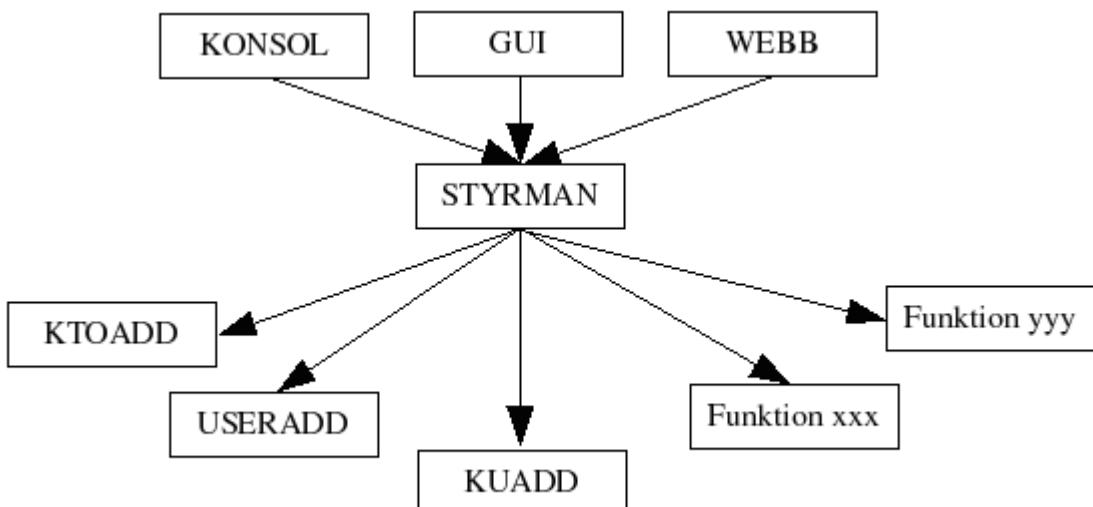
Målgruppen för OLFIX är färmansföretag och små företag med en eller flera samtidiga användare.

## Filosofin bakom konstruktionen

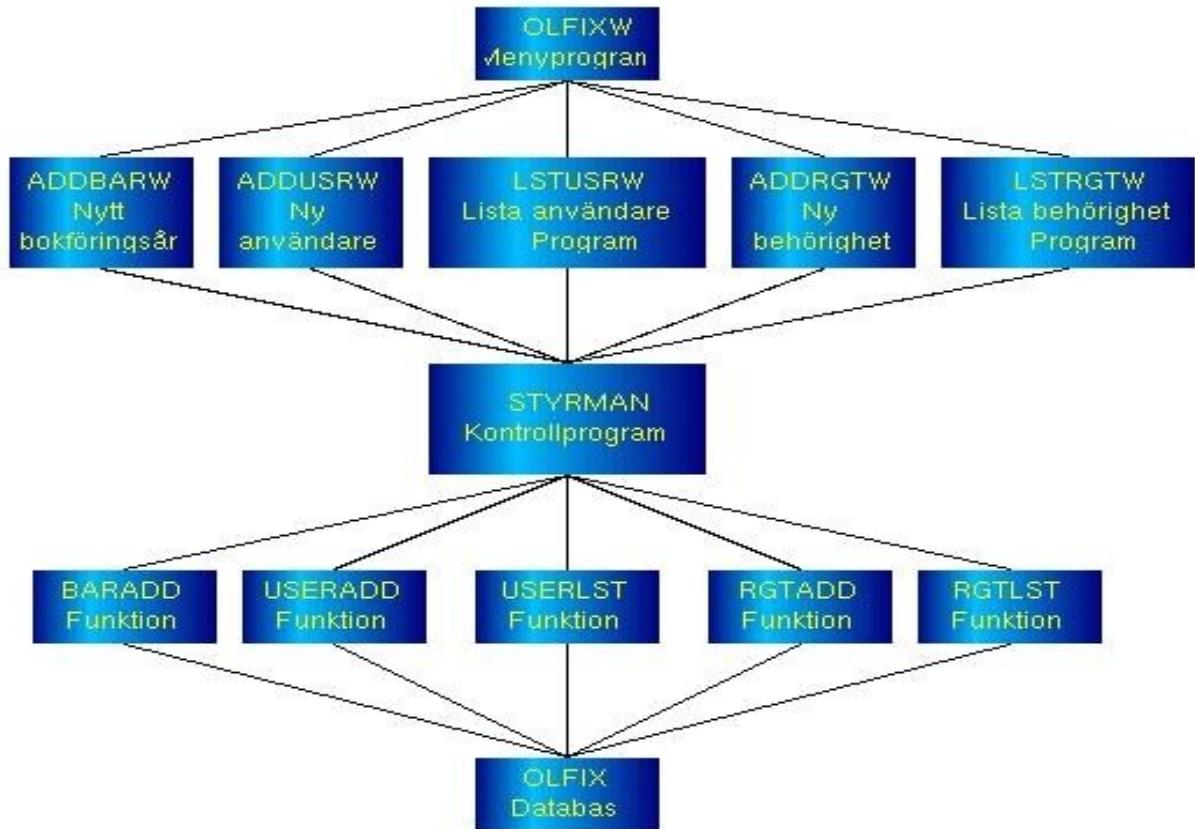
Det finns flera skäl till valet att använda små programmoduler (här kallade funktioner) .

- Det förenklar för användaren att begränsa användningen av funktioner.  
T ex kan en användare tillåtas se information men inte ändra eller lägga till ny information.  
Användare kan få begränsad möjlighet att se viss information.
- Det förenklar införandet av nya funktioner.
- Det ger möjlighet till att utveckla program och funktioner i andra programspråk.
- Det underlättar fejlrättning.
- 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 gränsnittet.



## 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 /opt/olfix/. Detta görs med programpaketet Acrobat med flera.

Det är ett oskick att sprida filer som tillhör programpaketet över hela systemet, därför hålls hella 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

Online hjälp

## Förutsättningar för OLFIX

För att kunna använda OLFIX fodras 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

Kugar och Kspread är delar av Koffice-paketet.

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:

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

I Mandrakelinux blir biblioteksstrukturen följande:

/usr/bin	(Här ligger alla binärfiler)
/usr/share/olfix/data	
/usr/share/olfix/report	
/usr/share/olfix/script	
/usr/share/olfix/sql	
/usr/share/doc/olfix-0.1.1.15a	(Här finns dokumentationen)

Följande behörigheter bör du ha:

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

## Installation.

Se till att du är ”root”.

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

För installation med flera användare. Detta är inte nödvändigt.

```
shell> cd /opt
shell> chown -R olfix olfix
shell> chgrp -R olfix olfix
shell> cd /opt/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 /opt/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 (max 8 tecken), 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 /opt/olfix/data
6. Editera filen **USRdata.txt**

```
"OLFIX", "Olfix Superuser", "IT", "Stab"  
"USERID", "Ditt Namn", "Avd", "Sektion"
```

USERID får vara max 8 tecken. Det skall dessutom vara det samma som inloggningsID till datorsystemet.

## **Script.**

7. Kopiera filen **/opt/olfix/util/.olixrc** till \$HOME.

.**olixrc** 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 returntecken. Måste ligga sist i filen.

## **Behörighet.**

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

8. Behörigheterna på binärfilerna ska göras enligt följande:

Här måste du ha rootbehörighet.

Sätt uid=olfix och gid=olfix

shell> cd /opt/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

Filten 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.

Filten .olfixrc

```
PATH=/opt/olfix/bin/  
HOST=localhost  
DATABASE=olfixtst  
HELPFILE=/doc/helpfiles/usermanual/UserManual.html  
REPORT=/opt/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..

Filten **.olfixrc** levereras i katalogen /opt/olfix/script

Innan man börjar använda OLFIX ska man vid behov 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 kopieras **/opt/olfix/script/.olfixrc** till **\$HOME/.olfixrc** 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 därfor finns det inga GUI-program.

## Upplägg av nya användare.

En ny användare registreras med hjälp av programmet ADDUSRW. AnvändarID får vara max 8 tecken och skall vara detsamma som inloggningsID.

## 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 innehåra 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 redigera filen /opt/olfix/data/PROGRAMdata.txt.

Kolumn 1 är ett lopnummer och tillika primary key.

Kolumn 2 anger huvudmeny.

Kolumn 3 anger submenu.

Kolumn 4 är ledtext för programmet.

Kolumn 5 är programnamn.

Därefter exekveras filen /opt/olfix/sql/LoadPROGRAM.sql:

```
]$ mysql -u root < /opt/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 redigera filen /opt/olfix/data/TRANSIDdata.txt.

Kolumn 1 är funktionsid/transaktionsid.

Kolumn 2 är beskrivning av transaktionen.

Därefter exekveras filen /opt/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>ADDNCW</b>	Lägga upp ny funktion.
<b>ADDFORW</b>	Lägga upp nya databashänvisningar. Olika företag.
<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>ADDLEVW</b>	Lägga upp ett nytt leveransvillkor.
<b>ADDLEVW</b>	Lägga upp en ny leverantör.
<b>ADDLPLW</b>	Lägga kompletterande artikeldata per lagerplats.
<b>ADDORDW</b>	Lägga upp en ny kundorder.
<b>ADDPKDW</b>	Lägga till ny produktgrupp/produktklass/produktkod
<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 leverantörsfakturor förfallna till betalning.
<b>BALRPTW</b>	Balansräkning.
<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 kundata.
<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>DELTXTW</b>	Radera texter ur textregistret.

<b>DELUSRW</b>	Ta bort användare samt dennes behörigheter
<b>DELVALW</b>	Ta bort en valuta.
<b>DSPBARW</b>	Visa räkenskapsår.
<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>KUFAKTW</b>	Fakturera kundorder.
<b>KURESKW</b>	Kundreskontra. Lista ej betalda kundfakturor.
<b>LEVFAKTW</b>	Registrera leverantörsfakturor.
<b>LEVRESKW</b>	Visa obetalda leverantörsfakturor, leverantörsreskontra.
<b>LSTBARW</b>	Lista bokföringsår/räkenskapsår.
<b>LSTFNCW</b>	Lista funktioner (transaktionstyper)
<b>LSTFTGW</b>	Lista företagsdata
<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>LSTORDW</b>	Lista kundorder på skärm.
<b>LSTPKDW</b>	Lista produktgrupper/produktklasser
<b>LSTRGTW</b>	Lista behörigheter.
<b>LSTTXTW</b>	Lista TEXTREGs texter.
<b>LSTUSRW</b>	Lista användare.
<b>LSTVALW</b>	Lista alla valutor.
<b>OLFIXW</b>	Grafiskt menyprogram för OLFIX.
<b>PLORDW</b>	Skapa och skriv ut en plocklista för en kundorder.
<b>PRTINKW</b>	Utskrift av beställning. Använder sig av Kugar.
<b>RPTGENW</b>	Generell rapportgenerator.
<b>RPTKTOW</b>	Kontorrapport.
<b>RPTSIEW</b>	Skapa en SIE-fil.
<b>SDOLISW</b>	Saldolista.
<b>SRCHARW</b>	Söka artiklar.
<b>SRCHKUW</b>	Söka kunder.

## OLFIXW.....Grafiskt menyprogram

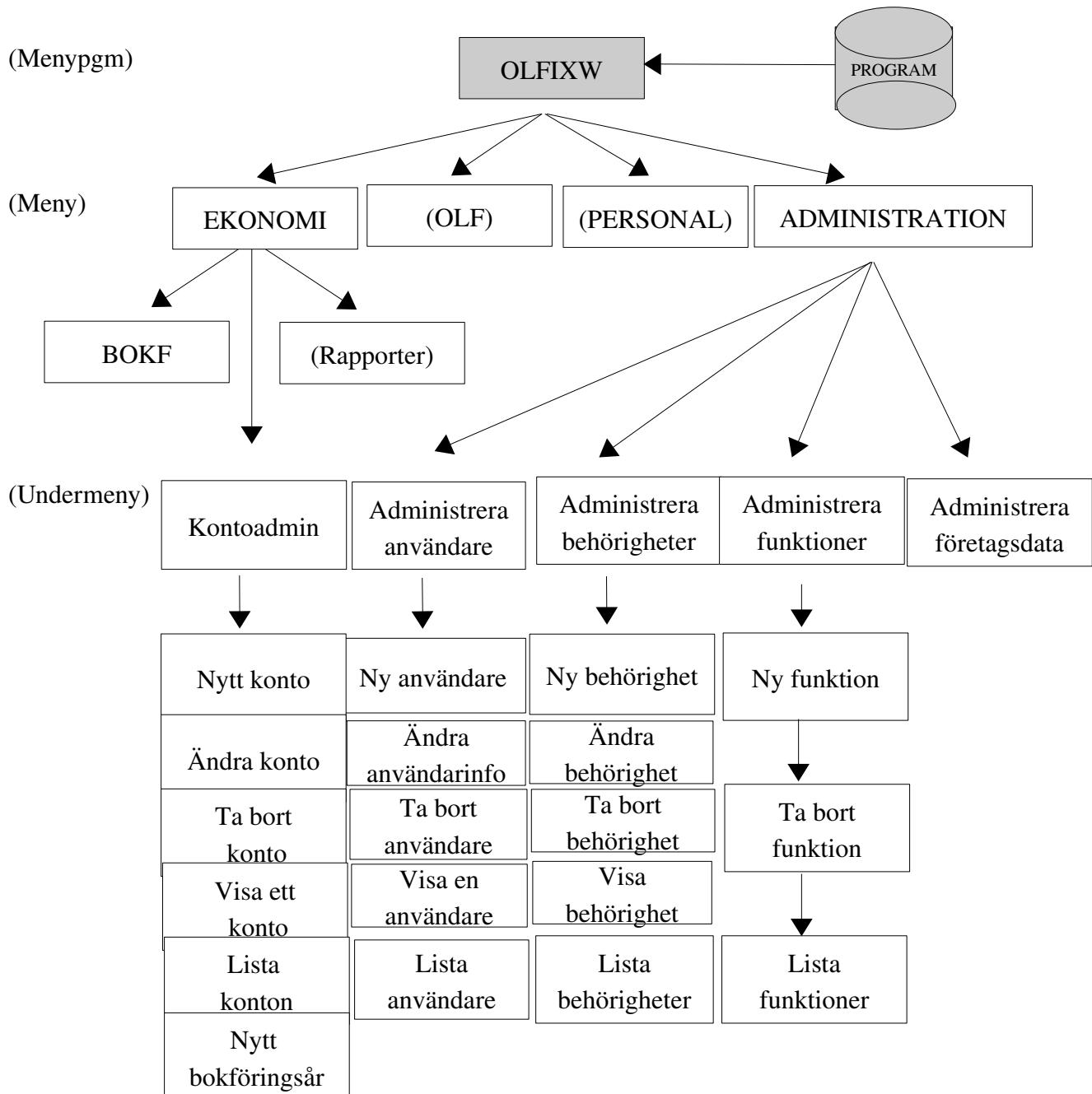
OLFIXW är ett grafiskt menyprogram.

Programmet plockar upp userid från environment.

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

Nya program läggs in genom registrering i tabellen PROGRAM.

OLFIXW har hårdkodat en användare, olfix, för att kunna börja använda OLFIX. Detta innebär att **användaren OLFIX absolut inte får tagas bort i tabellerna USR och RIGHTS**.



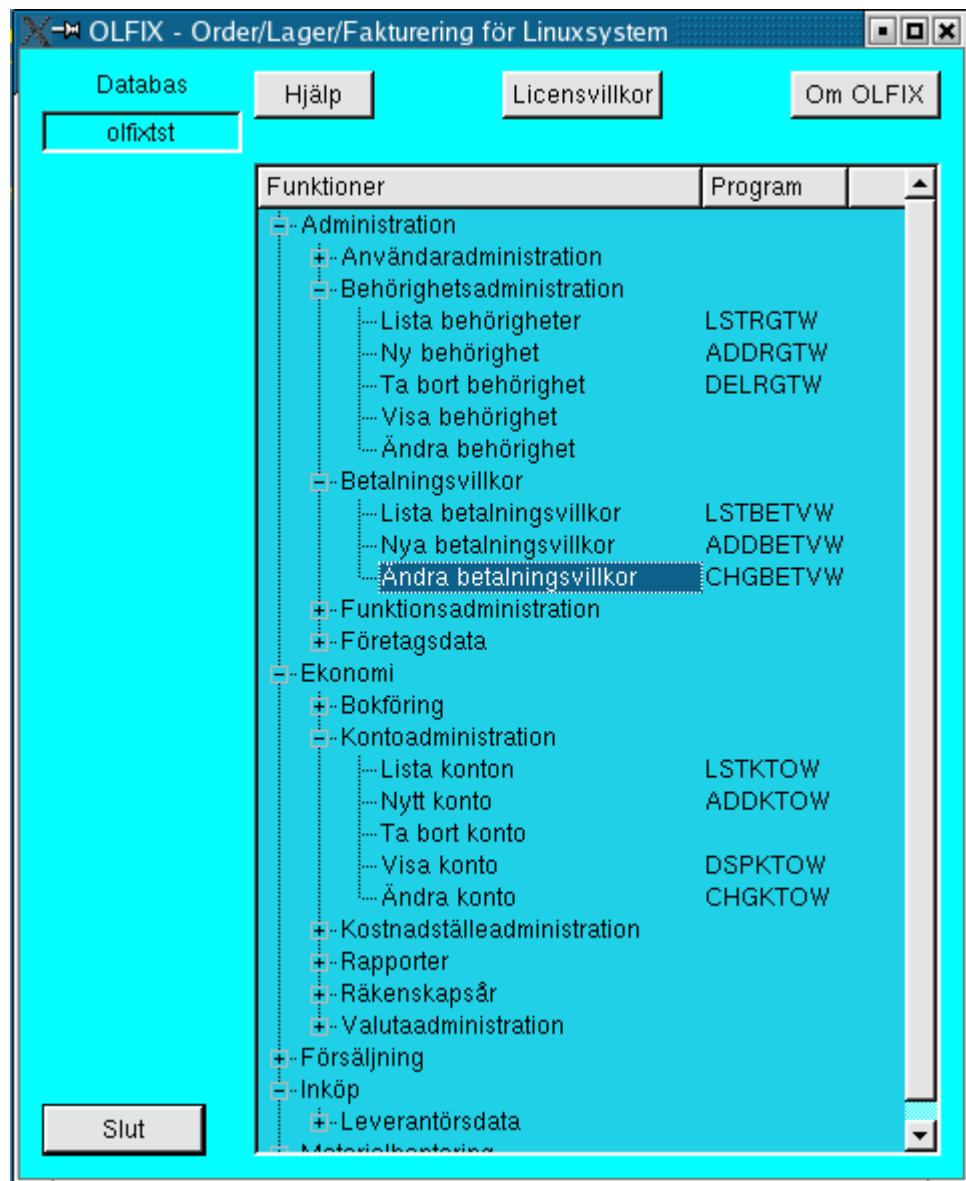


Bild 1.

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

Om filen **.olfixrc** saknas så kopieras den från /opt/olfix/script/.olfixrc varpå OLFIXW läser in data från filen.

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:

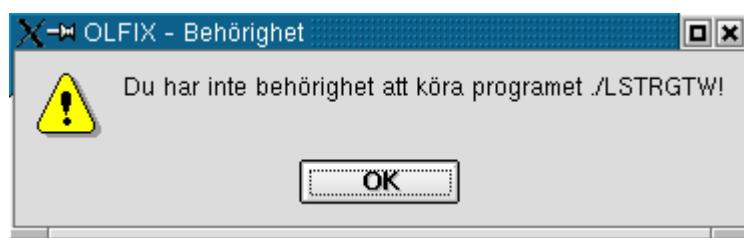


Bild 2.

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

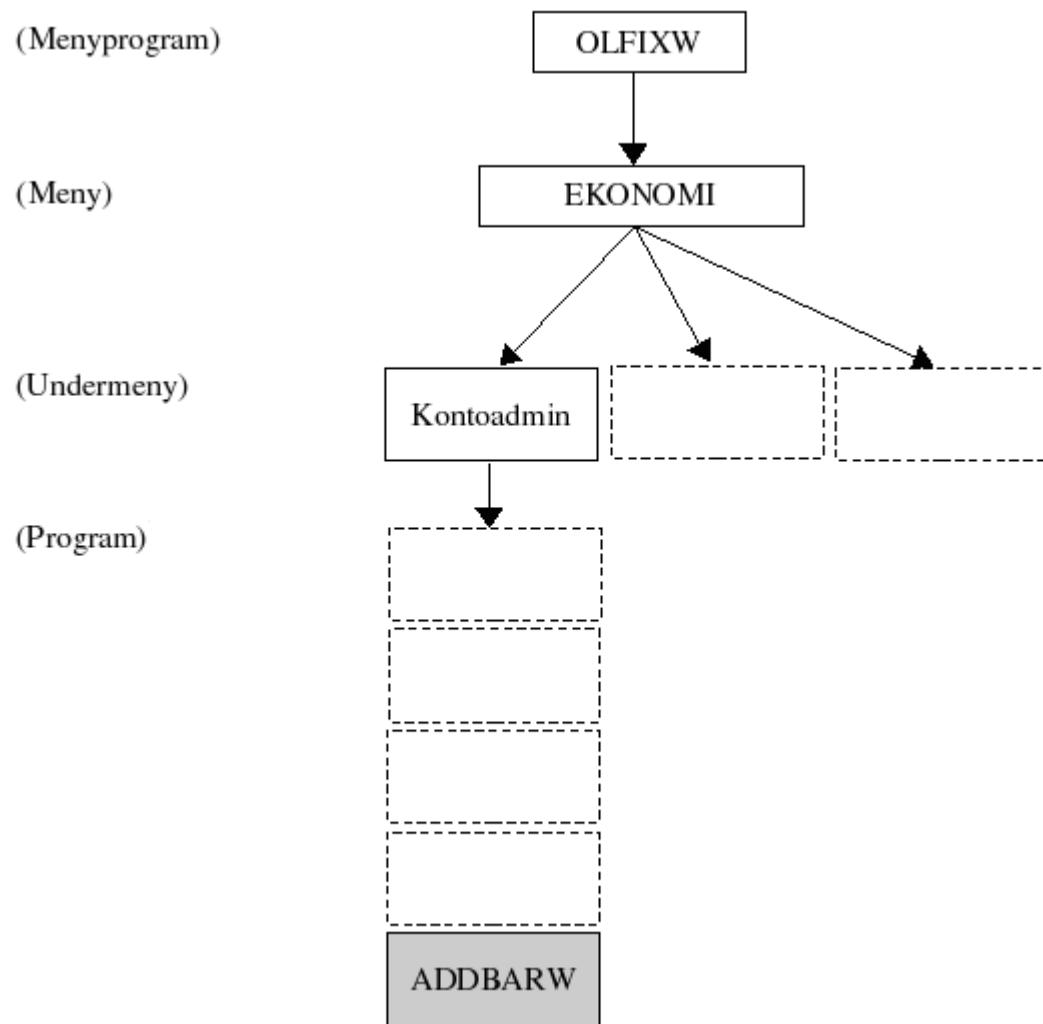
Test av behörigheten görs mot både **programmet** och **funktionen**.

#### **Behörighetskrav:**

För att kunna köra OLFIXW behövs behörighet till PRGLST.

## **ADDBARW.....Nytt bokföringsår**

ADDBARW, ett grafiskt program för att registrera nytt bokföringsår.



X-ADDBARW Lägga upp nytt bokföringsår

Bokföringsår: (arid, 2 teck.)	<input type="text"/>	Obligatoriskt.
Benämning:	<input type="text"/>	
Startdatum:	<input type="text"/>	Obligatoriskt.
Slutdatum:	<input type="text"/>	Obligatoriskt.
Beskattningsår	<input type="text"/>	Obligatoriskt.
Senaste ver.datum:	<input type="text"/>	
Nästa ver.nummer:	<input type="text"/> 1	
Kontoplan:	<input type="text"/>	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(userid);                // userid
process->addArgument("BARADD");               // OLFIX funktion
process->addArgument(arid);
process->addArgument(benamn);
process->addArgument(arstart);
process->addArgument(arslut);
process->addArgument(arlast);
process->addArgument(beskattar);
process->addArgument(senverdat);
process->addArgument(vernr);
process->addArgument(ktoplan);
```

Detta blir:

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

ARLAST sätts till "N"

Programmet hämtar sökvägen till hjälpparten från \$HOME/.olficrc.

Sedan kan tryck på Hjälp-knappen hitta OLFIXHLP och rätt plats i hjälpparten för att visa hjälpinformation om ADDBARW.

```
frmAddBar::readResursFil();                  // Hämta path till hjälpparten
hjelppfil=hjelppfil+"#NYTTBAR";             // Lägg till position

process = new QProcess();
process->addArgument("OLFIXHLP");            // OLFIX program
process->addArgument(hjelppfil);
```

Nyckeln till OLFIXHLP är #NYTTBAR. OLFIXHLP anropas med värdet i \$HOME/.olficrc HELPFILES kompletterat med #NYTTBAR t ex;

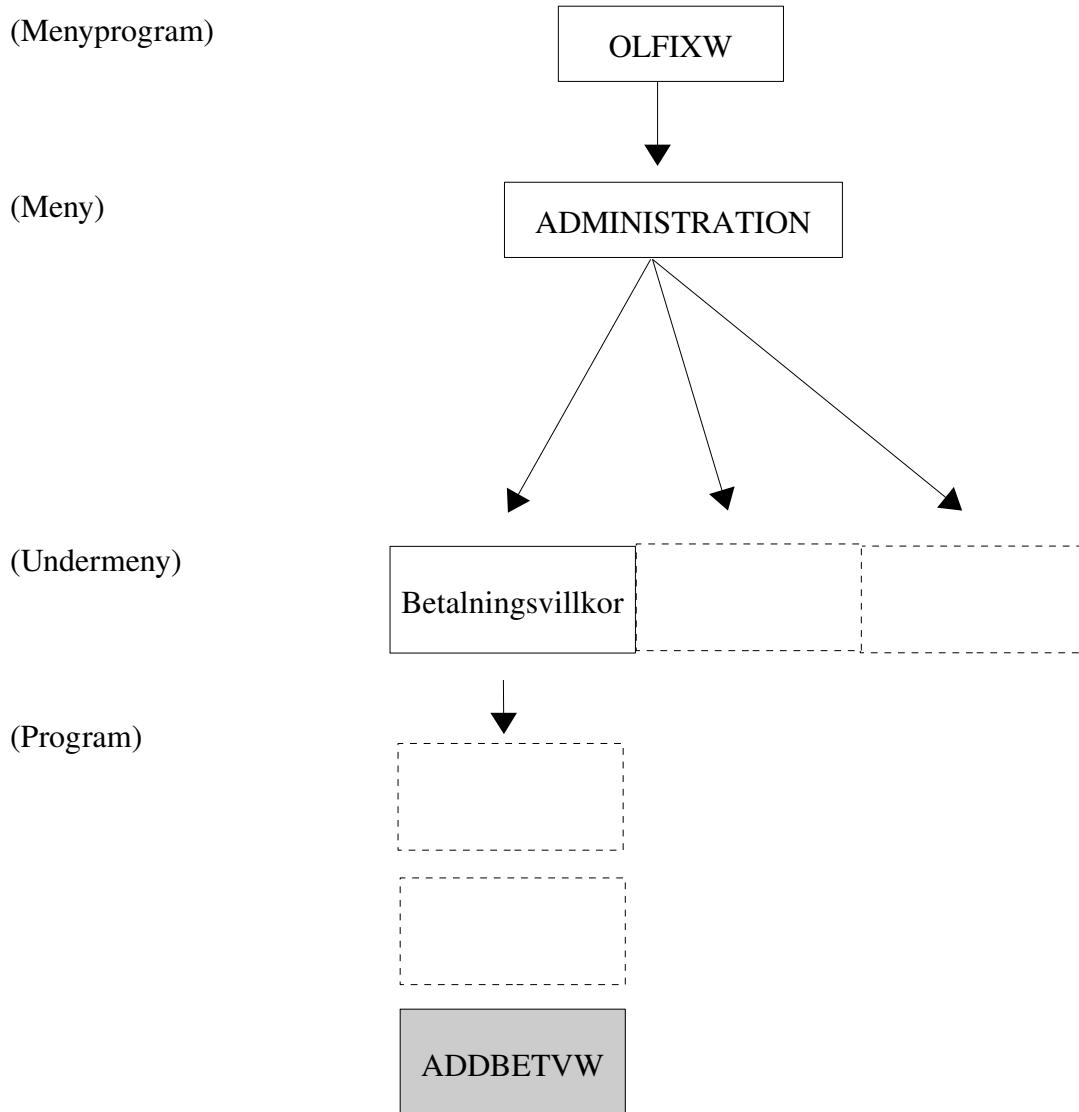
```
./OLFIXHLP /doc/helpfiles/usermanual/UserManual.html#NYTTBAR
```

### Behörighetskrav:

För att kunna köra ADDBARW behövs behörighet till  
PRGLST  
BARADD

## **ADDBETVW.....Nytt betalningsvillkor**

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





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(userid);                 // userid
process->addArgument("BETADD");                // OLFIX funktion
process->addArgument(betvillk);
process->addArgument(dagar);
process->addArgument(beskrivning);
```

Detta blir:

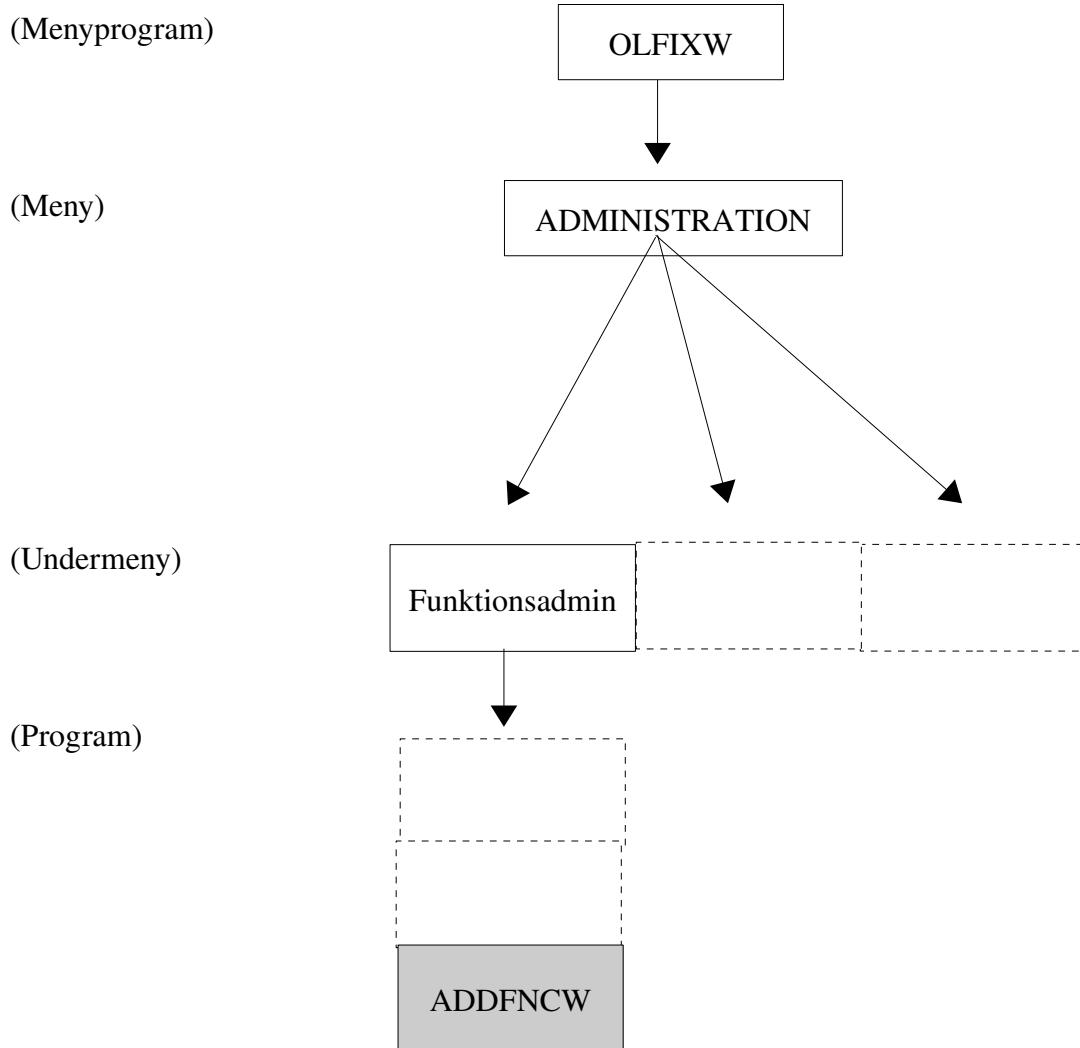
```
./STYRMAN userid BETADD betvillk dagar beskrivning.
```

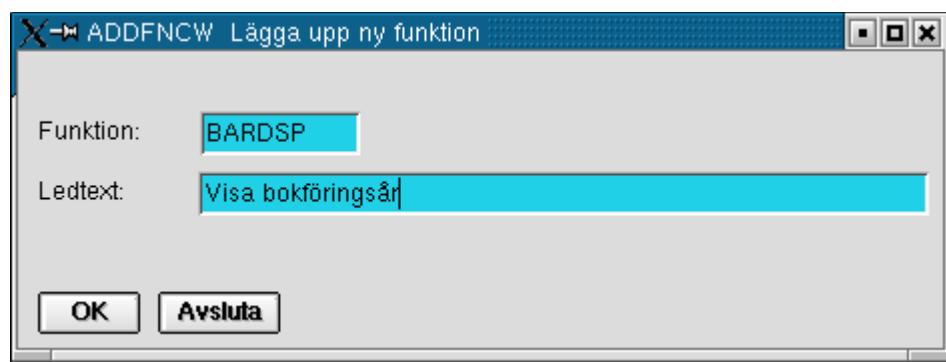
#### **Behörighetskrav:**

För att kunna köra ADDBETVW behövs behörighet till  
PRGLST  
BETADD

## **ADDFNCW....Ny funktion**

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





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

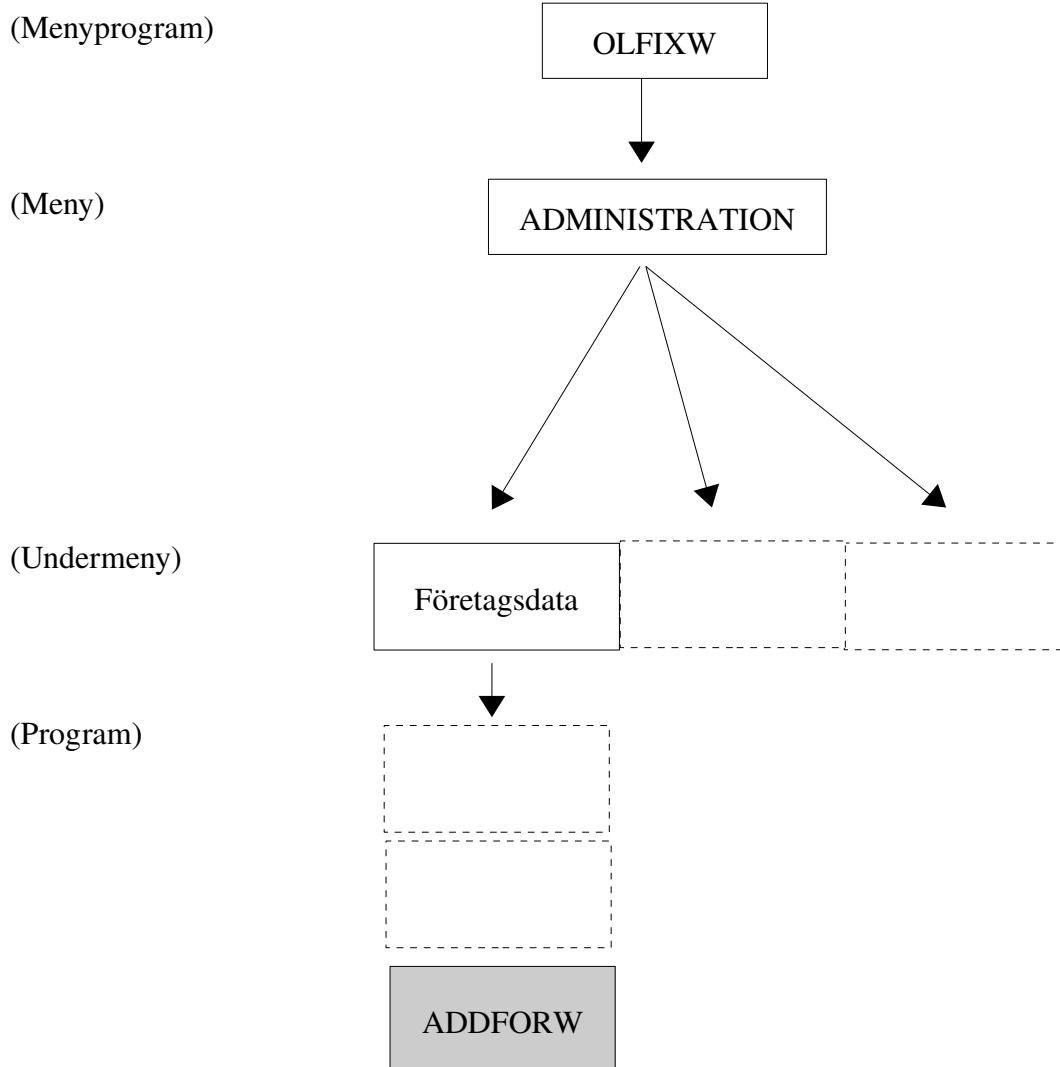
## ADDFORW....Nya företagsdatabaser

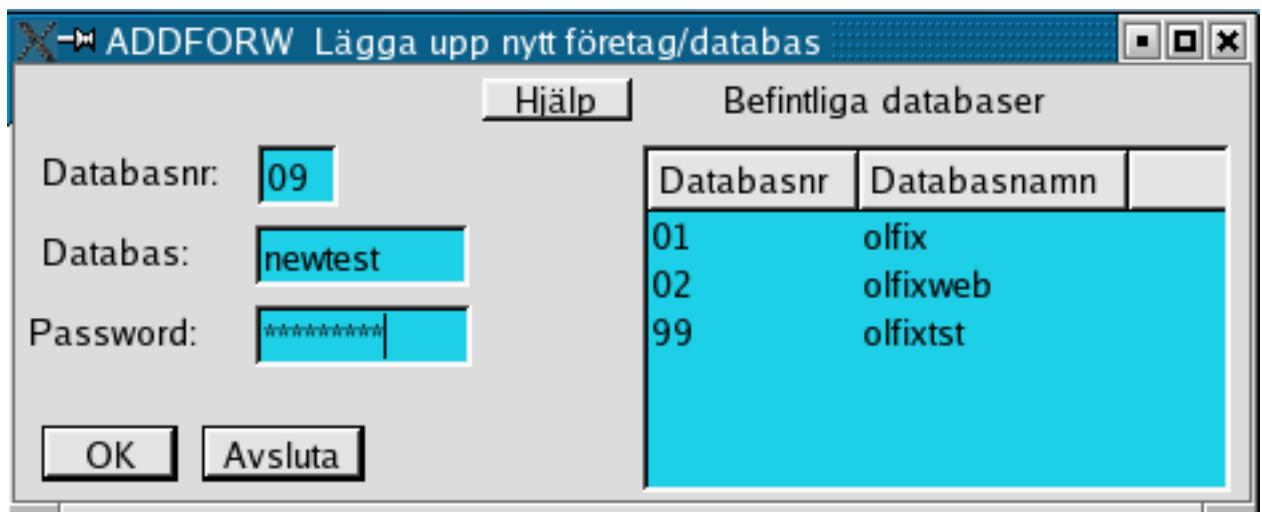
ADDFORW, ett grafiskt program för att skapa nya databaser samt lägga upp data i tabellen DATABAS. Programmet skapar en tom databas (grunddatabas).

För att kunna lägga upp nya databaser i MySQL krävs att vederbörande har nödvändiga **adminrättigheter**. Vidare bör denne ha goda kunskaper i MySQL och i frågespråket SQL.

DATABAS håller reda på vilka databaser, företag, som finns upplagda i MySQL.

DATABAS medger att en användare kan arbeta med flera företag.





Ange vilket nr databasen ska ha samt vilket namn databasen erhöll av DBAn.

Password ska vara det password som används för att administrera MySQLs databaser. Användaren måste också ha behörighet att skapa nya databaser.

## **Funktionsbeskrivning.**

ADDFORW är avsedd att skapa grundförutsättningarna för att möjliggöra för användare att arbeta med flera företag. ADDFORW lägger också upp en ny **fysisk** databas i MySQL.

ADDFORW lägger även upp information i tabellen DATABAS, fälten DATABASNR och DATABASTEXT. Dock läggs informationen bara upp i den nya databasen sant i den databas användaren råkar vara i när funktionen används.

## **Behörighetskrav:**

För att kunna köra ADDFORW behövs behörighet till

PRGLST

FORADD

FORCHK

FORDSP

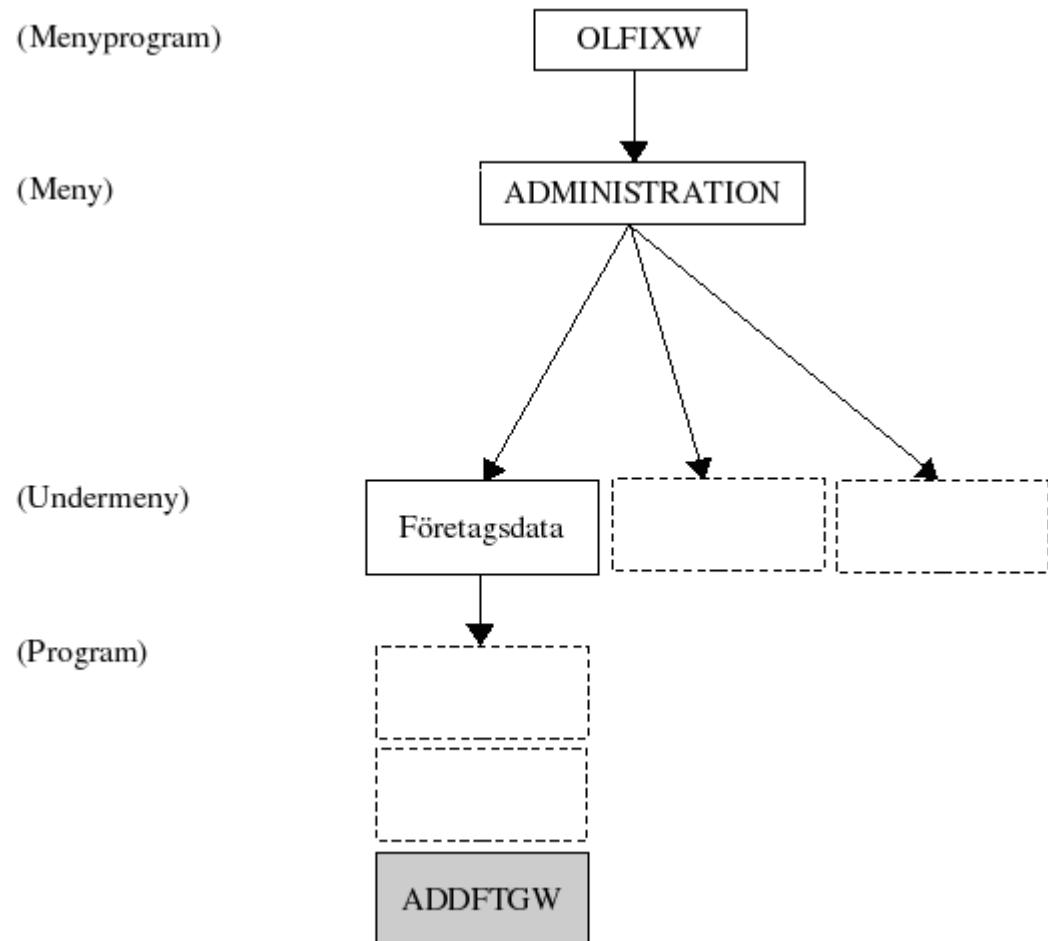
FORLST

OLFIXHLP

DBCHK (Hämtar vilka databaser som finns upplagda i databasen mysql, tabell db.)  
Behörighet att som administratör skapa(create) nya databaser och tabeller i MySQL.

## **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 skrivas 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 sparar i tabellen när man klickar på **Spara**.

Uppdateringen sker post för post, se FTGUPD.

X-» 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	Mobiltelefon:	070-991199	Telefax:	09-109919				
e-mailadress:	info@program.se								
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 :	<input type="checkbox"/> J								
<input type="button" value="Spara"/>	<input type="button" value="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 ADDFTGW.

ADDFTGW anropar TRNSADD via STYRMAN med parametrar.

```
process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(userid);                // 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",besoksaldr);
slotUpdateFtgdata("ADR5",postnr2);
slotUpdateFtgdata("ADR6",besoksort);
slotUpdateFtgdata("ADR7",godsaldr);
slotUpdateFtgdata("ADR8",postnr3);
slotUpdateFtgdata("ADR9",godsort);

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

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

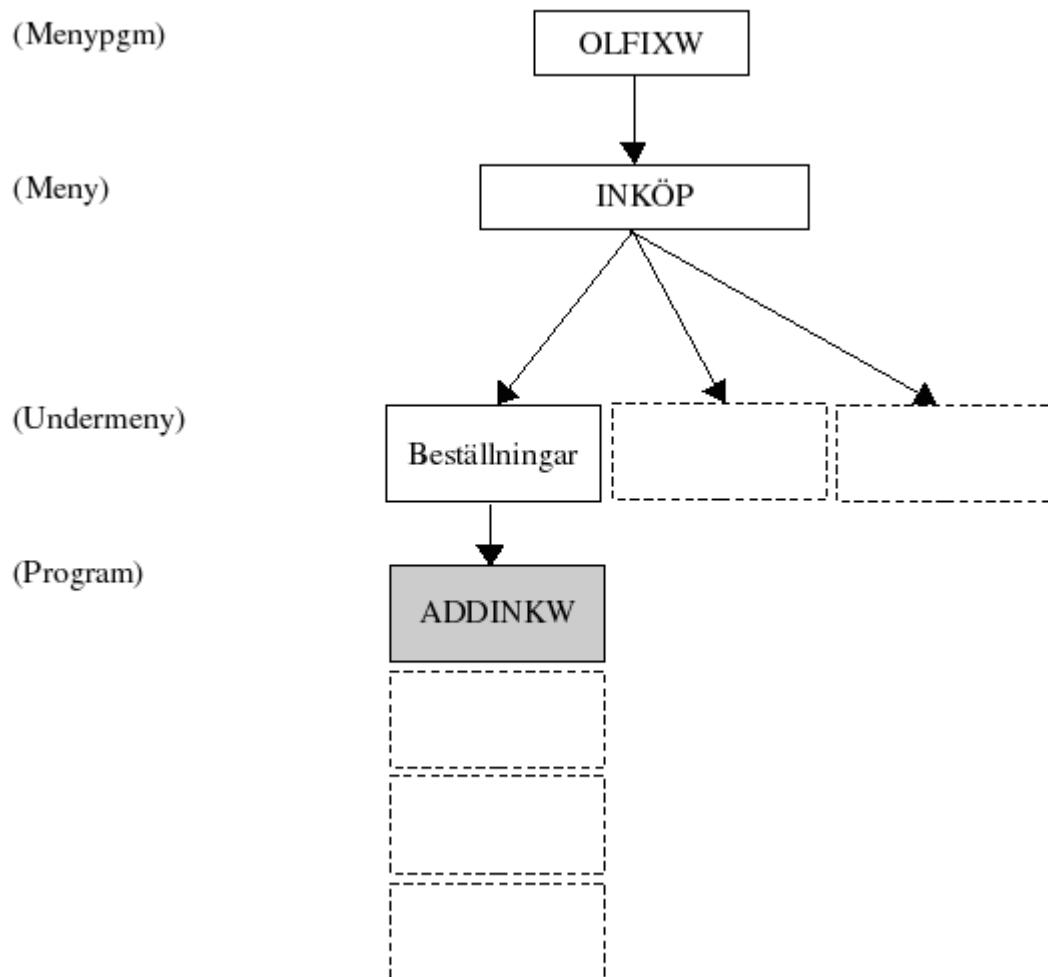
slotUpdateFtgdata("MOMS1",moms1);
slotUpdateFtgdata("MOMS2",moms2);
slotUpdateFtgdata("MOMS3",moms3);
slotUpdateFtgdata("MOMS4",moms4);
slotUpdateFtgdata("MOMS5",moms5);
slotUpdateFtgdata("MOMSI",momsin);           // momkonto ingående moms
slotUpdateFtgdata("MOMSU",momsut);           // momskonto utgående 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.



**X ADDINKW - Registrera inköpsorder**

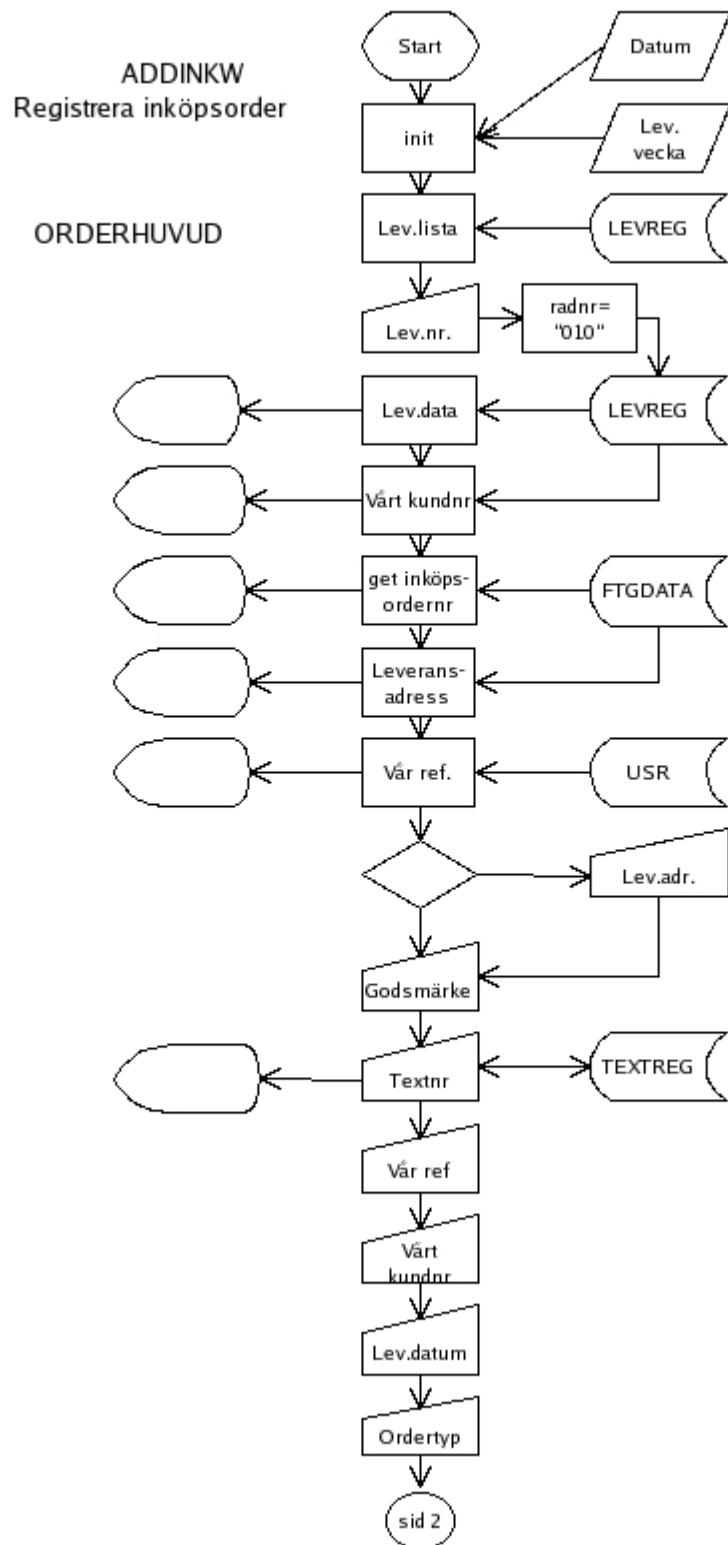
Leverantörsnr: .... \* 126 Beställningsnr: ..... 27 Beställningsdatum: ... 2004-02-05 \* = Obligatoriskt.

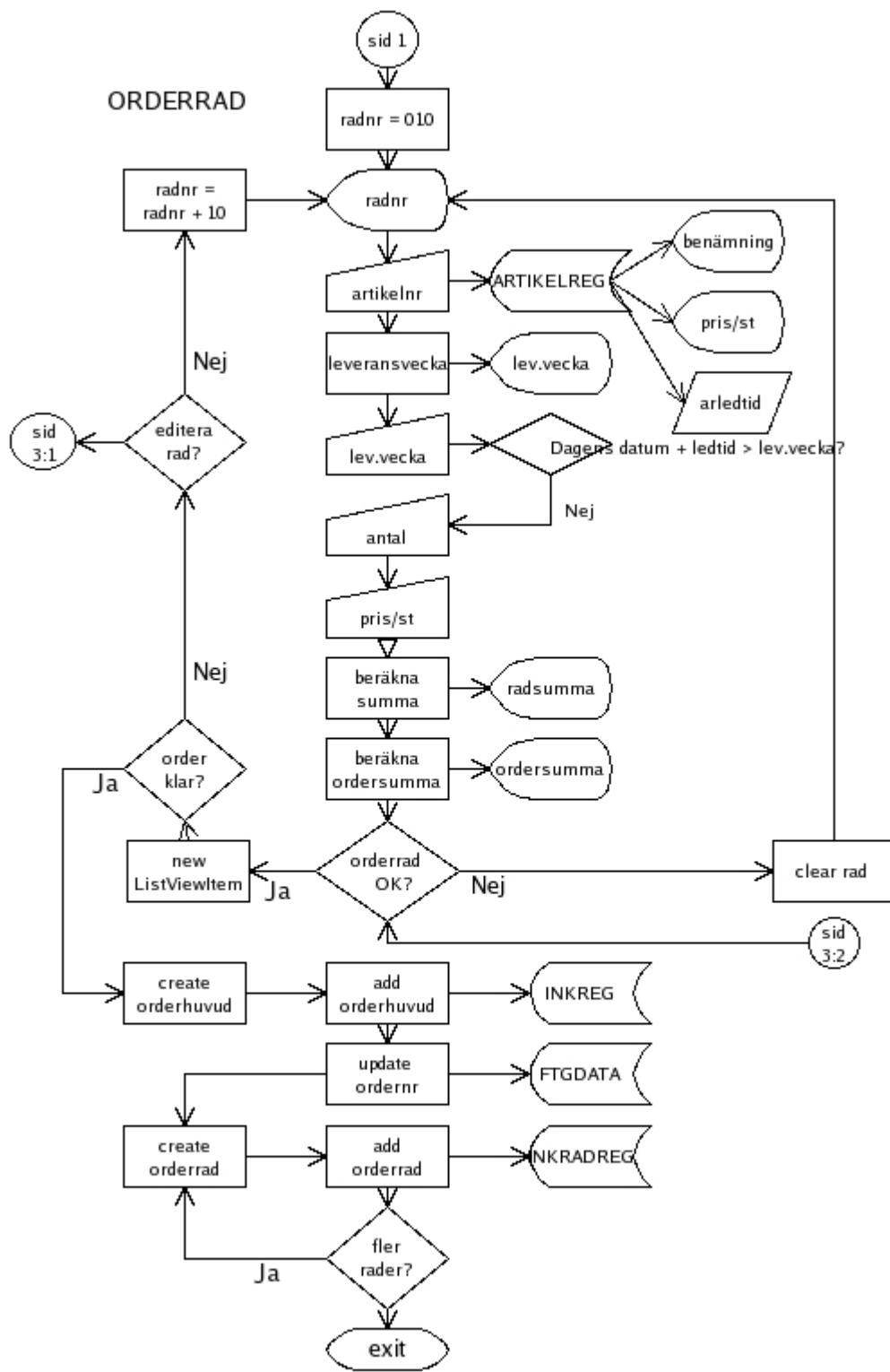
Leverantörens postadress	Orderhuvud	Mottagarens Leveransadress/Godsadress
Namn: ..... Dataspecialisten AB		PROGRAM AB
Adress: ..... Storgatan 1		Verkygsgatan 11
Postnummer: ..... 199 11		199 97
Postadress: ..... STORSTAD		PROGSTAD
Land: ..... Sverige	Valuta: SEK	
Godsmärke: ..... PELLE		
Vår referent: ..... Jan Pihlgren	Er referens: ..	Ola Norman
Direktfn: ..... 09-112233	Direktfax: ... 09112239	Best.typ: N Vårt kundnr: ... 567891
Leveransdatum: .... 2004-02-25	Leveransvillkor: .... 001	Leveranssätt: 001 Betalningsvillkor: 1
Kommentar: ..... Direkt		Eftertext: ..... 001

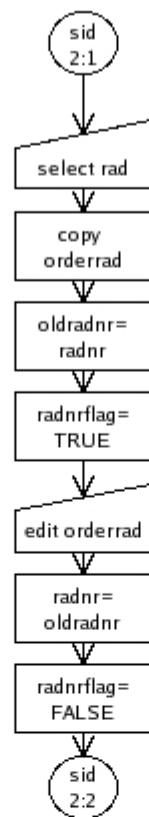
**Detaljrader**

Radnr	Artikelnr *	Benämning	Lev.vecka	Antal *	Pris/st	Summa	Godkänna rad
030	1173-1447	Timerkrets	4093	0.00	5.45		<input type="checkbox"/> Ja <input type="checkbox"/> Nej
Radnr	Artikelnr	Benämning	Lev.vecka	Antal	Pris	Summa	
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  OK  Avbryt 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

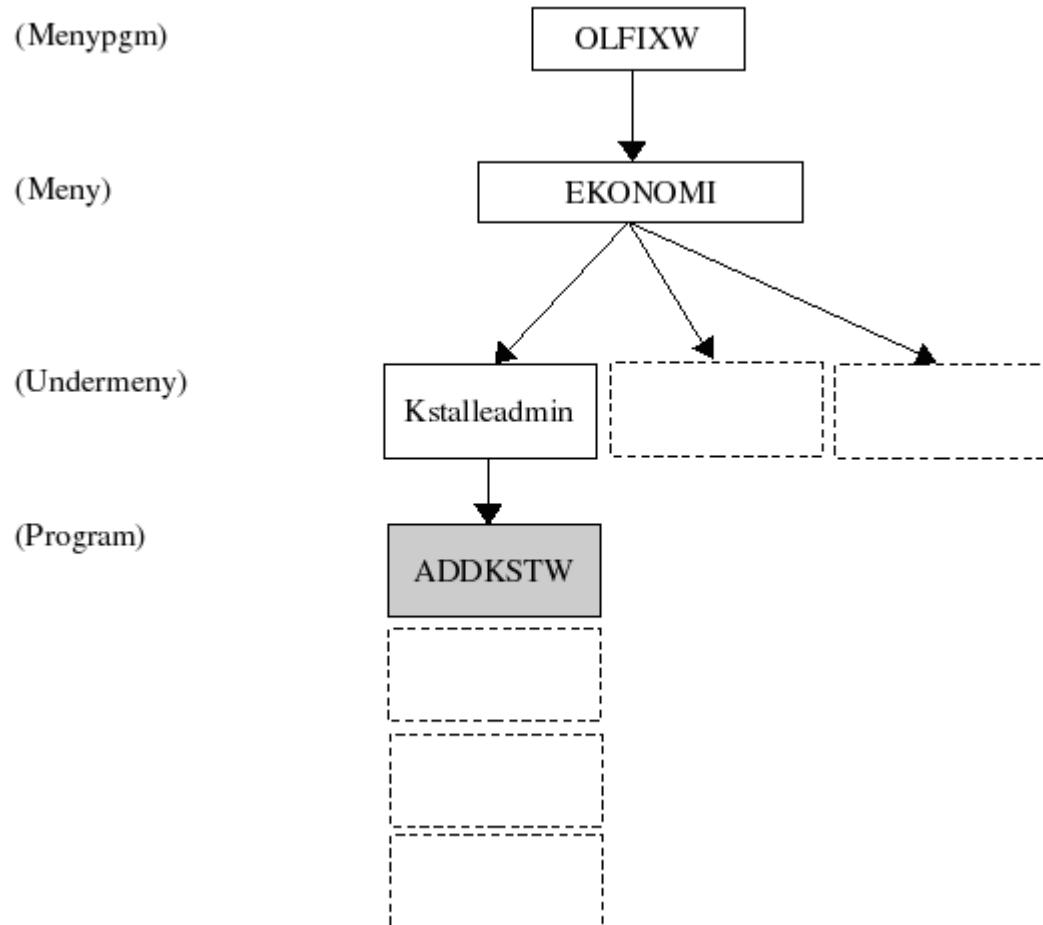
### Behörighetskrav:

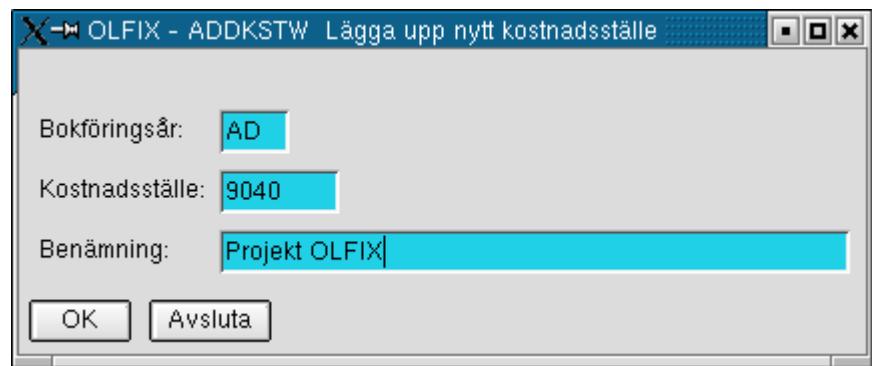
För att kunna köra ADDINKW behövs behörighet till

ARDSP  
ARDSPL  
FTGDSP  
FTGUPD  
INKADD  
INKRADD  
LEV\_DSP  
LEVLST  
LEV\_VDSP  
LEV\_SDSP  
PRGLST  
TXTDSP  
USERDSP

## **ADDKSTW.....Nytt kostnadsställe**

ADDKSTW, ett grafiskt program för att registrera nya kostnadstället. Man måste ange bokföringsår (arid). 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 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:

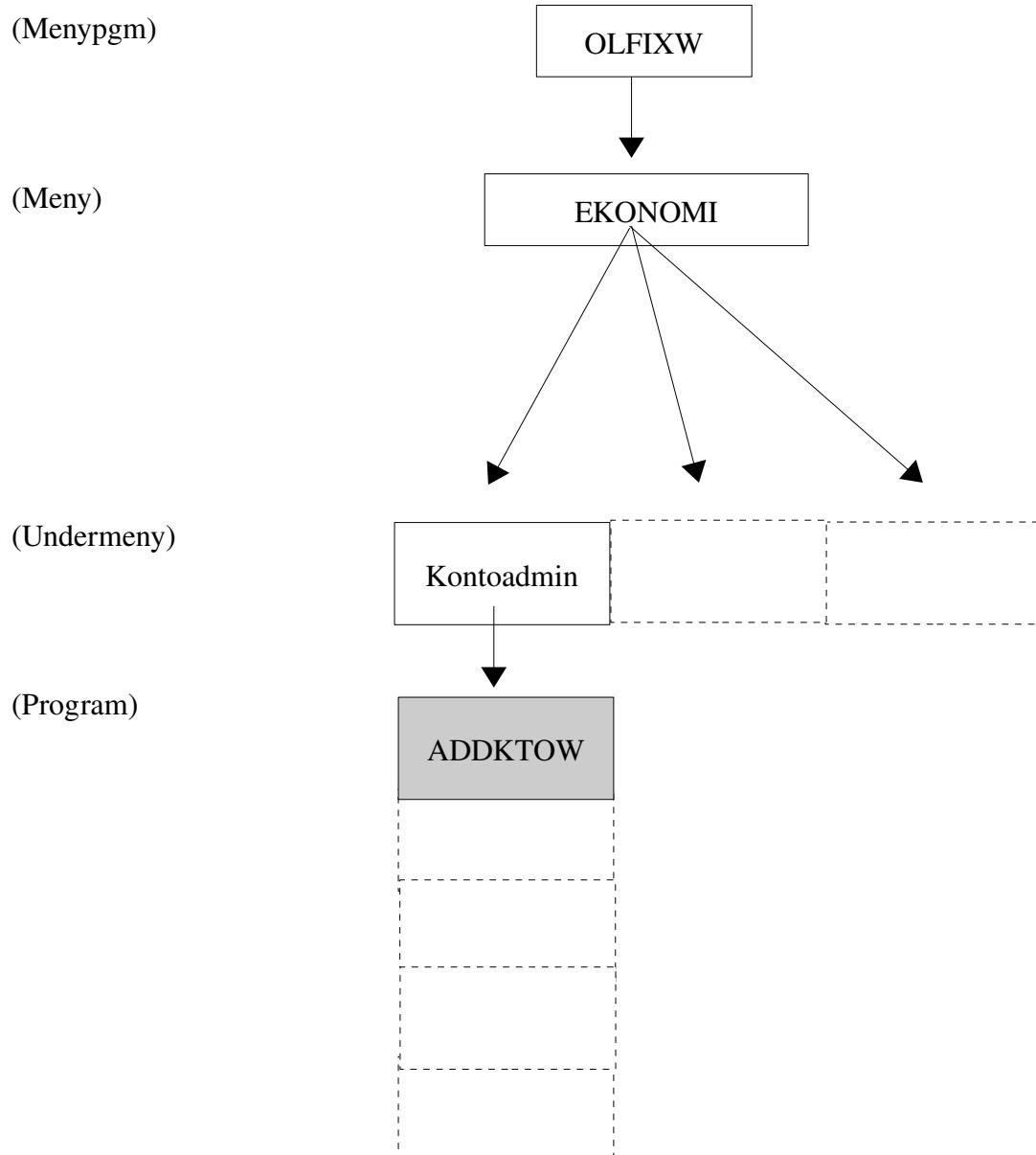
```
./STYRMAN userid KSTADD arid kstalle benamn
```

#### **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.



**ADDTOW Lägga upp nytt konto**

\* = Obligatorisk

Bokföringsår: \* ZZ  
(arid, 2 teck.)

Kontonummer \* 2440

Benämning: Leverantörsskulder

Manuell (J/N): \* J

Momskod: \* 1

SRUnr: \* 0

Kostnadssättle:

Projekt:

Subkonto:

Kontoplan: \* EUBAS97

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 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:

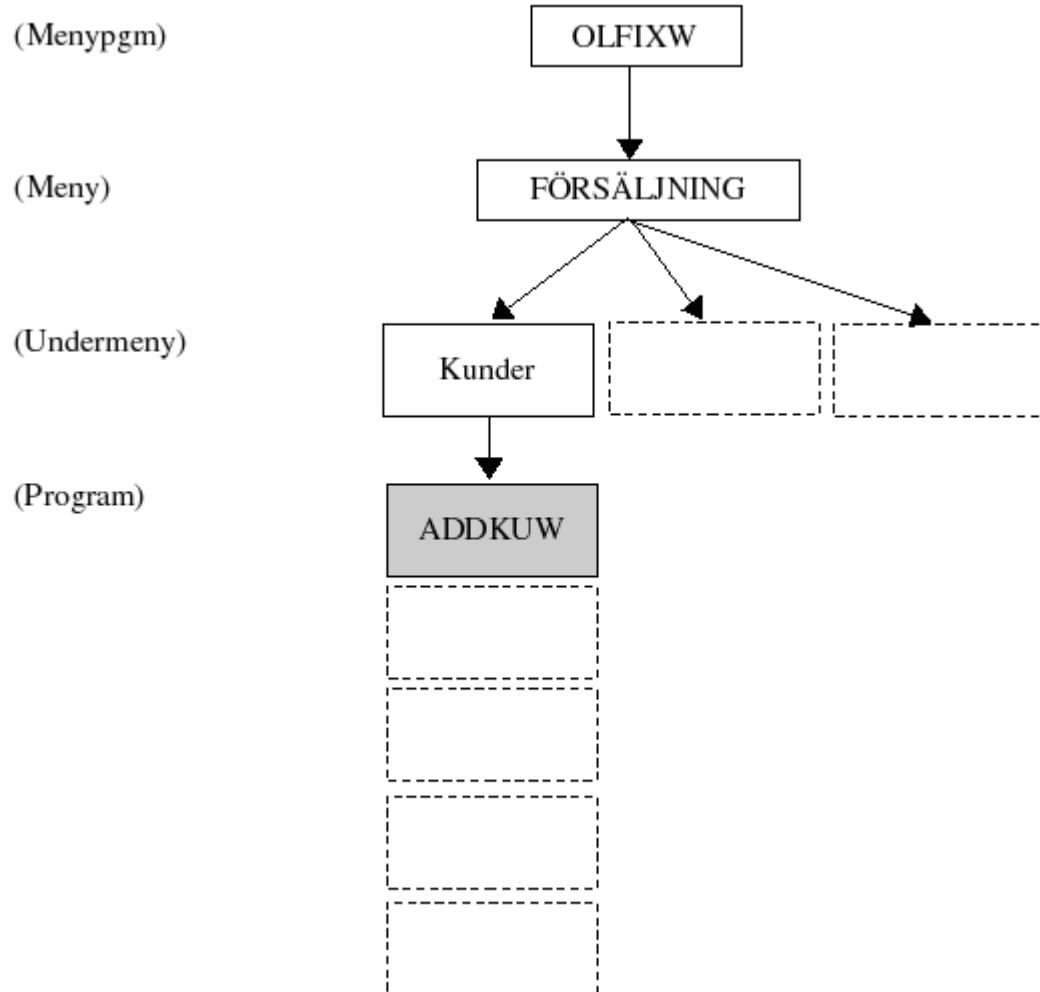
```
./STYRMAN usr KTOADD arid ktonr benamn manuell momskod srunr kst
                    projekt subkonto ktoplan
```

### 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.



X ADDKUW - Lägga upp en ny kund.

\* = Obligatorisk information

KundID: .....	*	4381	Max 10 tecken		
Kundnamn: .....	*	Testbolaget AB	Org.nr: 559999-9999		
Kundadress: .....	Storgatan 33				
Postnummer: .....	199 11				
Postadress: .....	LILLEBY				
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: ..	l.andersson@testbolaget.se				
Vår säljare: .....	Caroline Seljare				
Distrikt: .....	TST	Kundkategori: .....	ST	Prislista:	0
Leveransplats: ....	001	Leveransvillkor: ....	001	Leveranssätt:	.... 001
Betalningsvillkor:	1				
Valuta: .....	SEK	Språkkod: .....	sv		
Ordererkänndande: ..	J	Plocklista :	J	Följesedel:	..... J
Expeditionsavgift: ..	J	Fraktavgift: .....	J	Kravbrev:	..... J
Kreditlimit: .....	100000	Kreditkod:	JN		
Dröjsmålsränta: ....	J	Dröjsmålsfaktura: ..	J		
Fri text (100 tkn): ..	Bästa kunden				

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 *kundata* enligt följande:

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

```
kunddata.append(drojmalsrenta);
kunddata.append(skilj);
kunddata.append(drofmalsfakt);
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" );
process->addArgument(usr); // OLFIX funktion
process->addArgument( "KUADD" );
process->addArgument(kunddata);
process->addArgument(momskod);
```

Detta blir:

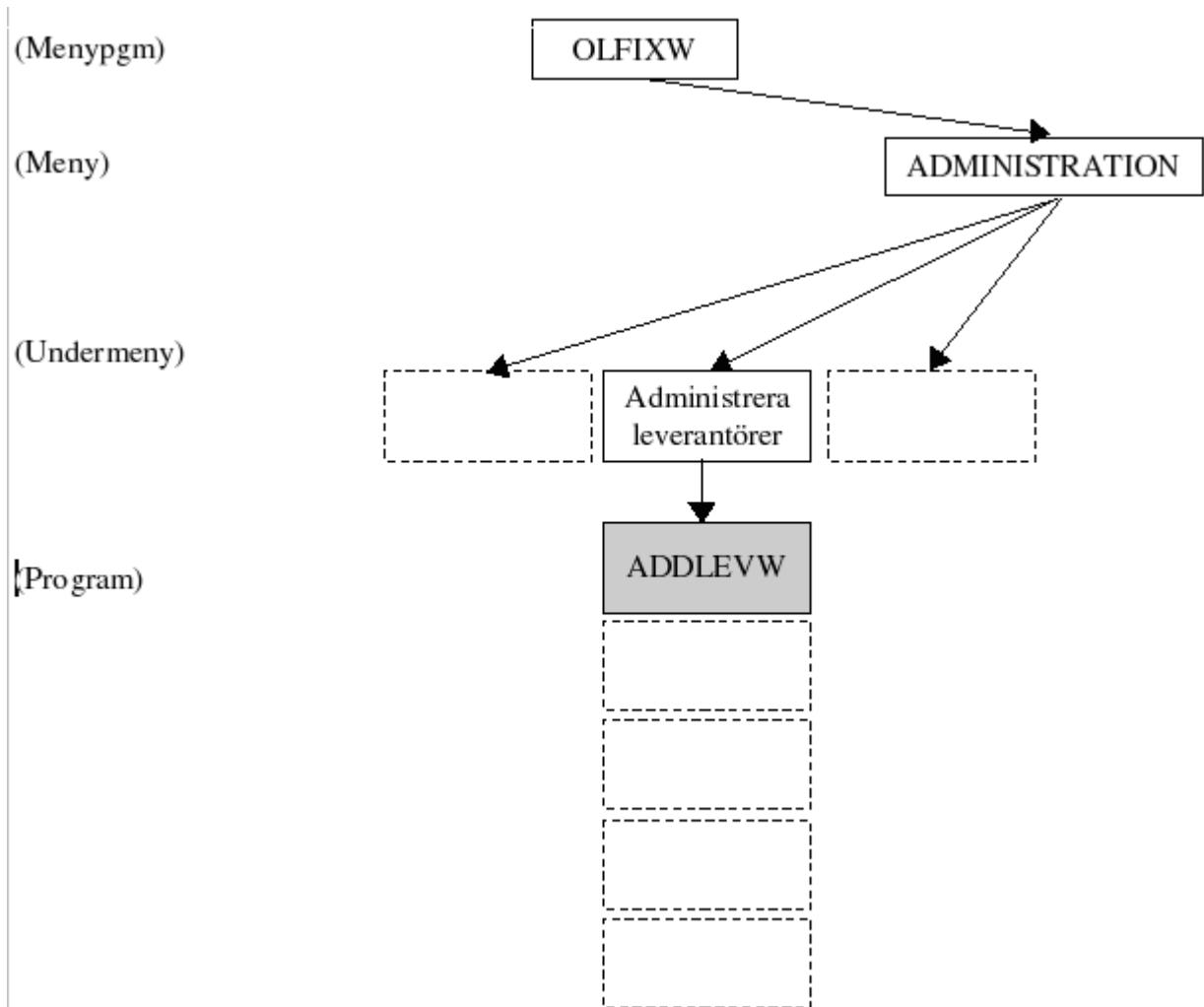
```
./STYRMAN usr KUADD kunddata
```

### **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.



X ADDLEVW - Lägga upp en ny leverantör.

Leverantörsnummer:	Max 10 tecken.	Obligatoriskt.
Organisationsnr:		
Leverantörsnamn:		Obligatoriskt.
Leverantörsadress:		
Postnummer: .....		
Postadress: .....		
Land: .....		
Telefonnummer: .....		
Faxnummer: .....		
Telex: .....		
E-mail: .....		
Referent: .....		
Ref's telefonnr: .....		
Momskod: .....	1	Obligatoriskt.
Kontonummer: .....		
Postgironummer: ....		
Bankgironummer: ...		
Kundnr: .....		

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(levtelelexnr);
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 levtelelexnr 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

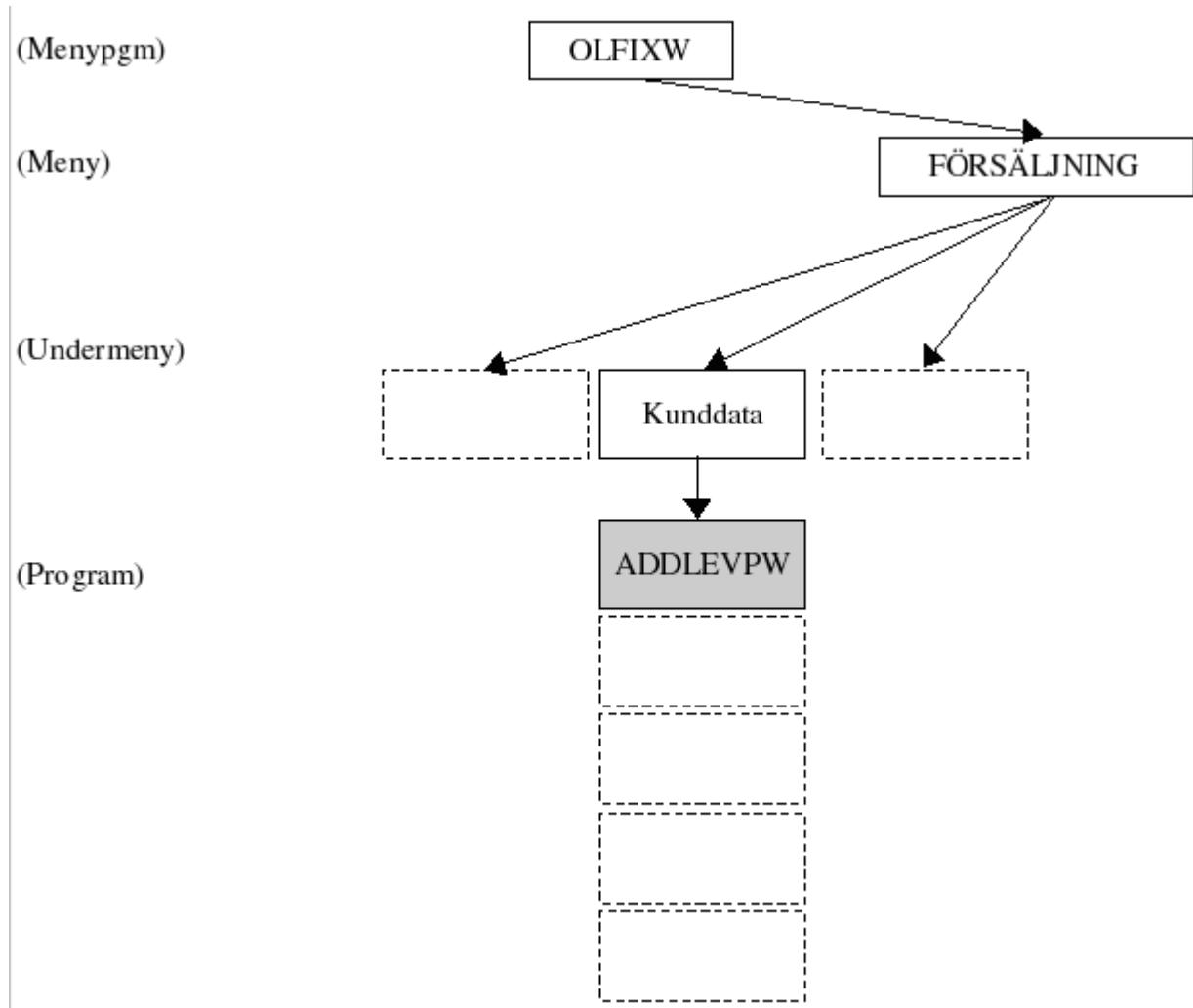
PRGLST

LEVADD



## ADDLEVPW ..... Ny standardleveransplats

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





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:

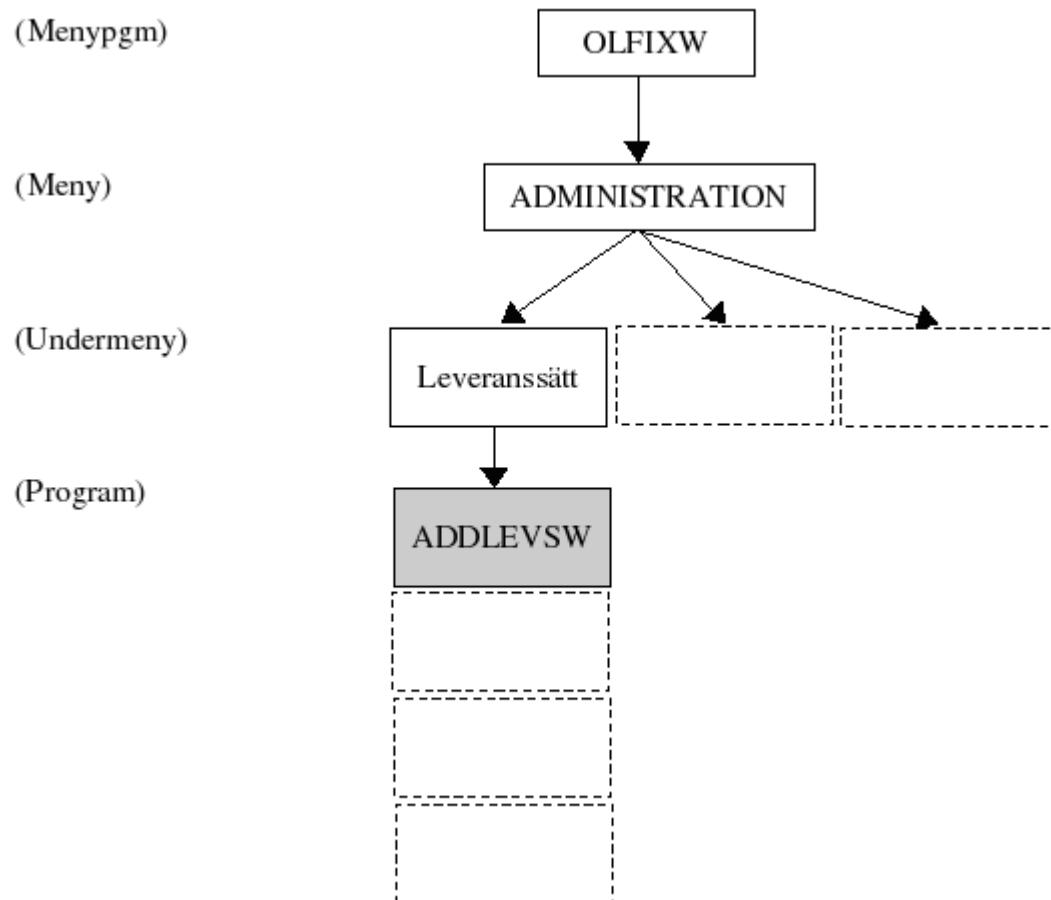
./STYRMAN usr SLPADD kundid levplatsnr levadress postnr postadr land  
Turordningen på argumenten är viktig, SLPADD bearbetar dem i denna ordning.

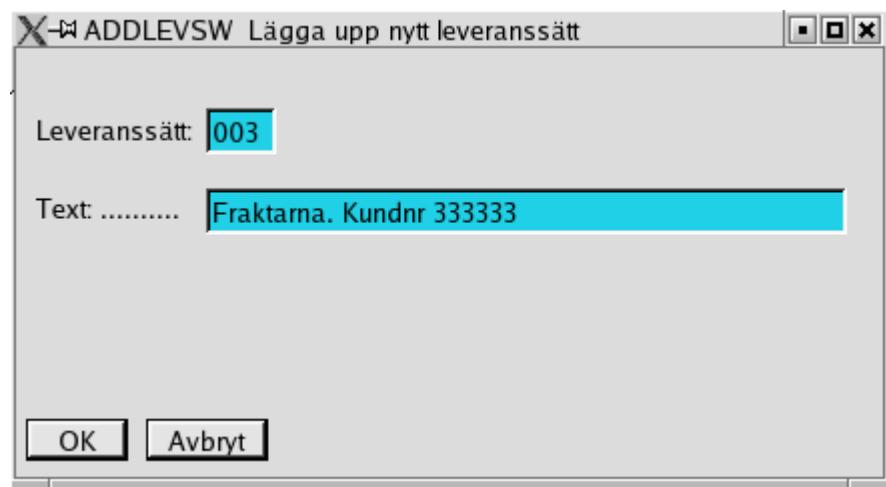
### Behörighetskrav:

För att kunna köra ADDLEVV 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

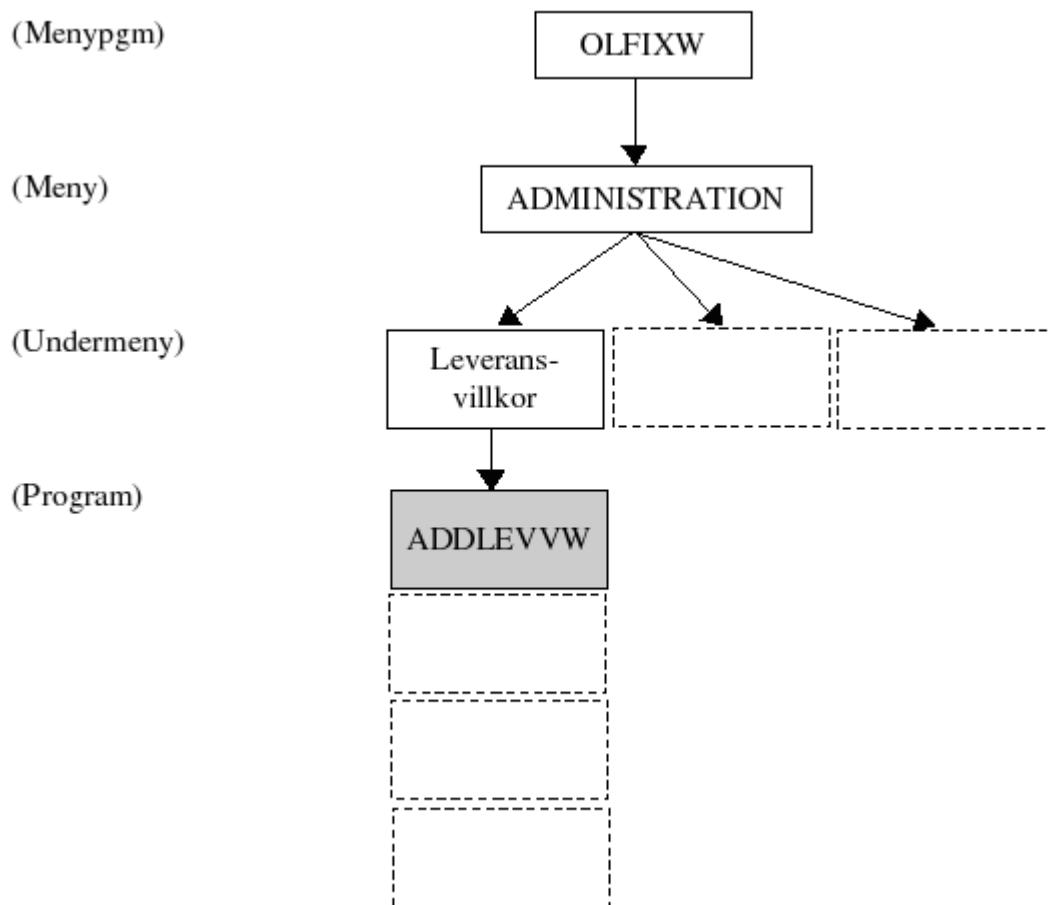
Turordningen på argumenten är viktig, LEVSADD bearbetar dem i denna ordning.

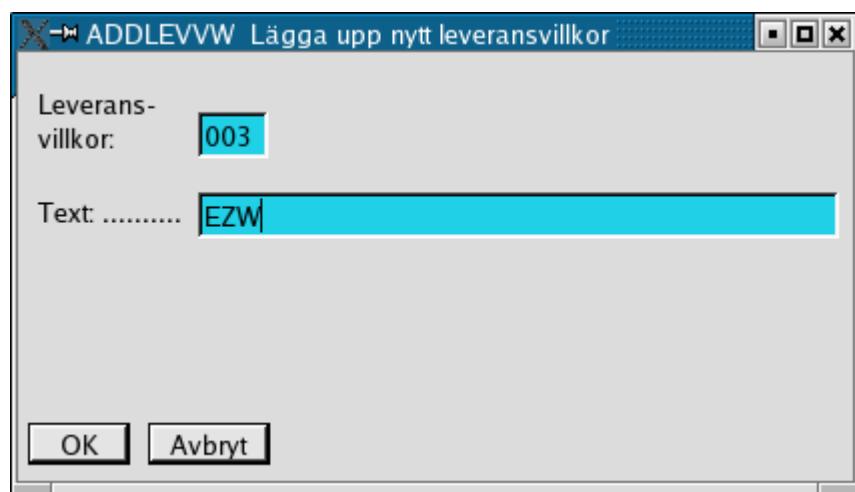
#### **Behörighetskrav:**

För att kunna köra ADDLEVSW behövs behörighet till  
PRGLST  
LEVSADD

## ADDLEVW ..... Nytt leveransvillkor

ADDLEVW 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 ADDLEVWW.

ADDLEVWW 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.

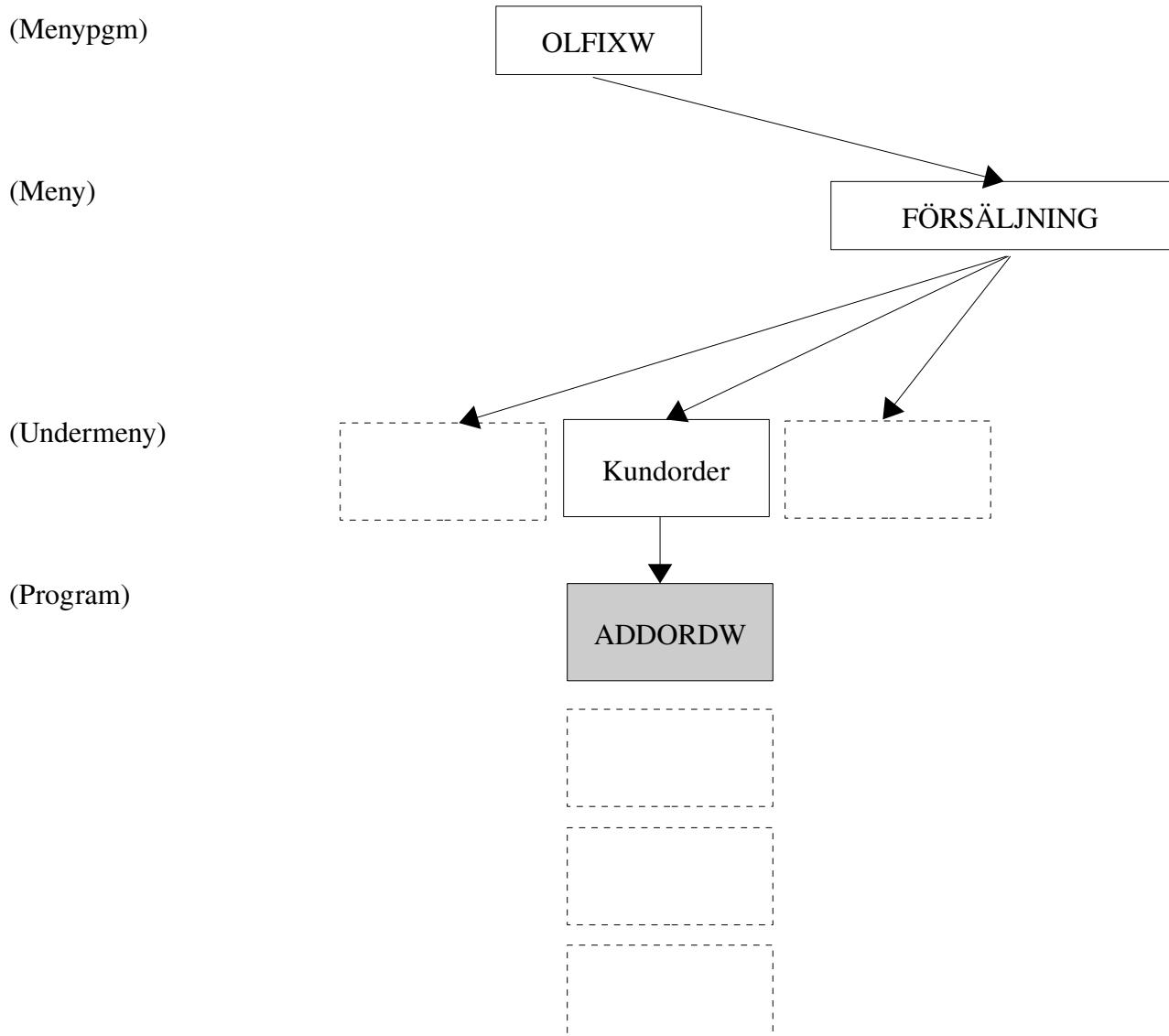
#### **Behörighetskrav:**

För att kunna köra ADDLEVWW behövs behörighet till  
PRGLST  
LEVVADD



## ADDORDW ..... Ny kundorder

ADDORDW är ett grafiskt program för att lägga upp ny kundorder.  
Programmet plockar upp userid från environment.



**ADDORDW - Registrera kundorder**

Kundnr: ..... *	4383	Datum	* = Obligatoriskt.	Hjälp					
Ordernummer: .....	24	2006-03-03							
<b>Postadress, Fakturaadress</b>		<b>Leveransaddress</b>							
Namn: .....	Government Inc	Ref:	Tony Blair	Leveransplats .....					
Adress: .....	Downing Street 10		Downing Street 10	Leveransvillkor .....					
Postnummer: .....	G23KB1		123 45	Betalningsvillkor: ....					
Postadress: .....	LONDON		LONDON	Valuta: .....					
Land: .....	Great Brittain		England	Moms: ..... %					
Säljare .....	Jan Pihlgren	Önskad leveranstid	2006-03-21	Prislista: .....					
Godsmärke: .....	Godsmärke			5					
Radnr	Artikelnr *	Artikelsök.	Benämning	Leveransvecka	Antal *	Pris/st	Summa	Moms	Godkänn rad
020	1000-1003		Linux På egen hand	6122		117.00			6 <input type="checkbox"/> Ja <input type="checkbox"/> Nej
Radnr	Artikelnr		Benämning	Leveransvecka	Antal	Pris/st	Summa	Moms	
010	1000-1006		The C Programming language	6095	1	117.00	124.02	7.02	

Summa      Frakt      Fraktmoms      Summa moms      Order Total      Inga fler orderrader.      Registrera ordern!

124.02      0.00      7.02      0.00      Order klar!      Avbryt      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 ADDORDW.

ADDORDW anropar följande funktioner via STYRMAN med parametrar:

ARDSP artikelnr	
AR2UPD val lagerstelle artikelnr	(uppdatera reserverad kvantitet)
FTGDSP "KORNR"	(hämta senast använda kundordernr)
FTGDSP "SKUNR"	(hämta senast använda kundnr)
FTGUPD "KORNR" ordernr	(uppdatera senast använda kundordernr)
KUADD kunddata	(lägga upp en ny kund)
KUDSP kundnr	
KULST	(inget argument)
LEVPDSP kundnr platsnr	(hämta adressuppgifter från STDLEVPLATS)
ORDADD orderhuvuddata	(spara data för orderhuvud)
ORDRADD orderraddata	(spara data för orderrad)(används engång för varje orderrad)
PKDDSP produktklass	(hämta MOMSKOD)
PRISDSP artikelnr	(Hämta eventuellt rabattpris)
PRGLST	(används av OLFIXW)

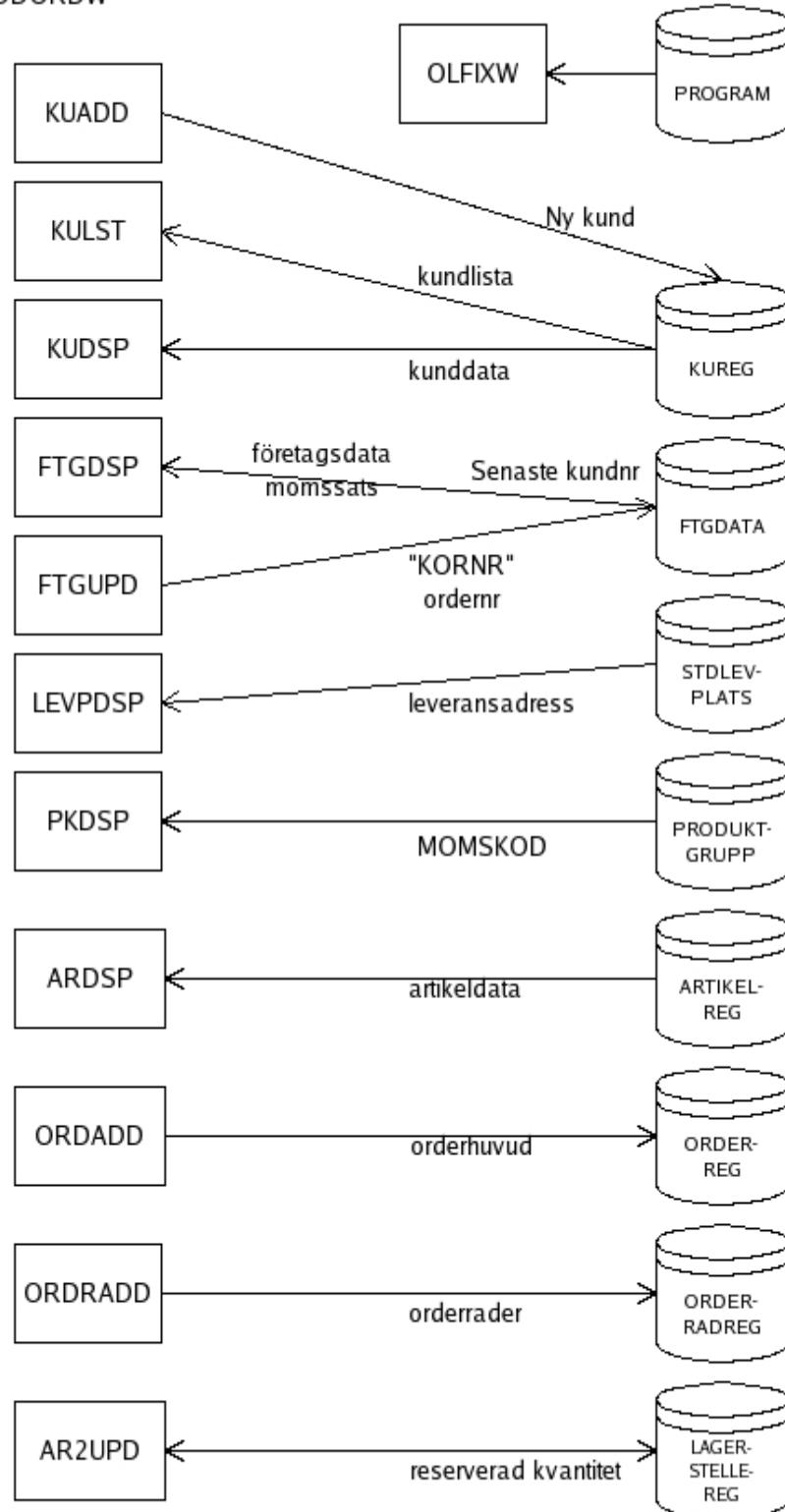
Detta blir t ex:

./STYRMAN usr ARDSP artikelnr

**Behörighetskrav:**

För att kunna köra ADDORDW behövs behörighet till  
AR2UPD  
ARDSP  
FTGDSP  
FTGUPD  
KUADD  
KUDSP  
KULST  
LEVPDSP  
ORDADD  
ORDRADD  
PKDSP  
PRGLST  
PRISDSP

## ADDORDW



### Registrering av orderrad. ADDORDW

```
void lineEditArtikelNr_returnedPressed()
    getArtikeldata();

slotArdataEndOfProcess()
    fpris=inradArtikel.mid(i5+3,m-4);
    prodklass=inradArtikel.mid(i7+3,m-4);
    getRadMoms();

slotProdkodeEndOfProcess()
    (från PRODUKTGRUPP)
    getMoms(momskod,"R");

slotgetMomsEndOfProcess()
    radmoms = inrad.mid(i2+2,m-4);
    getPrislista();
    return;

slotPrisdataEndOfProcess()
    orderradpris=fpris;
    choosePris();

orderradpris=prislista1-5 || fpris;
lineEditAPris->setText(orderradpris);

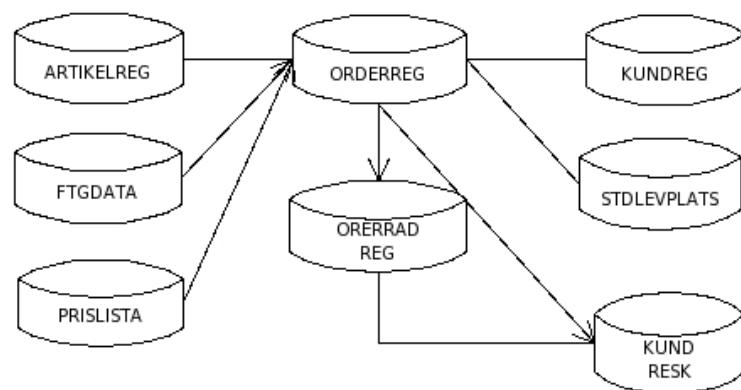
void lineEditBenamn_returnPressed()

void lineEditLeveransvecka_returnPressed()

void lineEditAntal_returnPressed()

lineEditAPris_returnPressed()
    summa=aPris x antal
    momsbelopp=summa x radmoms /100
    summa=summa+momsbelopp
```

Relationer för ORDERREG

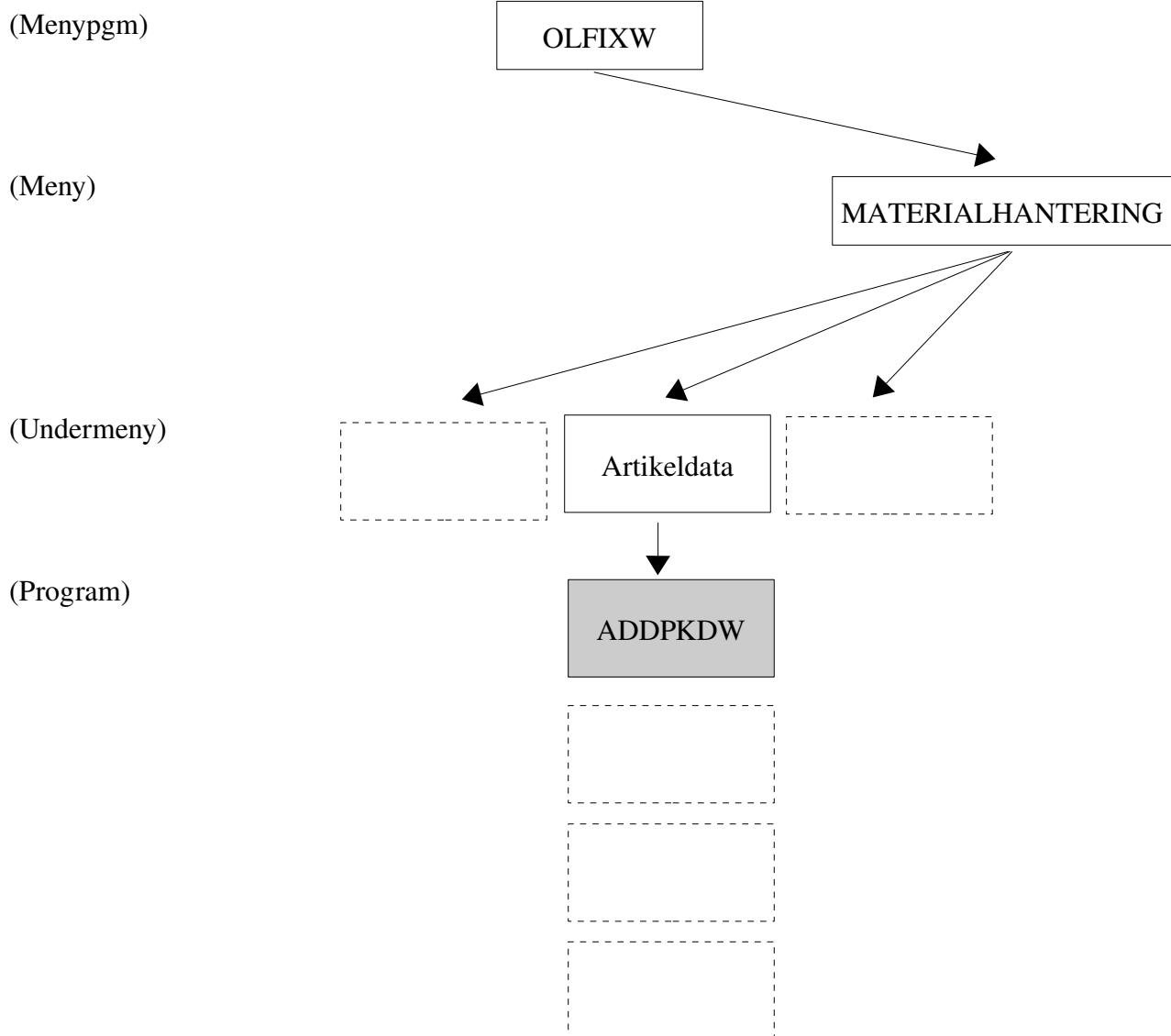


## ADDPKDW ..... Ny produktgrupp

ADDPKDW är ett grafiskt program för att lägga upp nya **produktgrupper/produktklasser**.

Se DSPARW – produktklass.

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 ADDPKDW behövs behörighet till  
PRGLST  
PKDADD

## ADDRGTW.....Ny behörighet

ADDRGTW är ett grafiskt program för att lägga upp nya behörigheter

Programmet plockar upp userid från environment.

(Menypgm)

OLFIXW

(Meny)

ADMINISTRATION

(Undermeny)

Administrera  
behörigheter

(Program)

ADDRGTW



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.

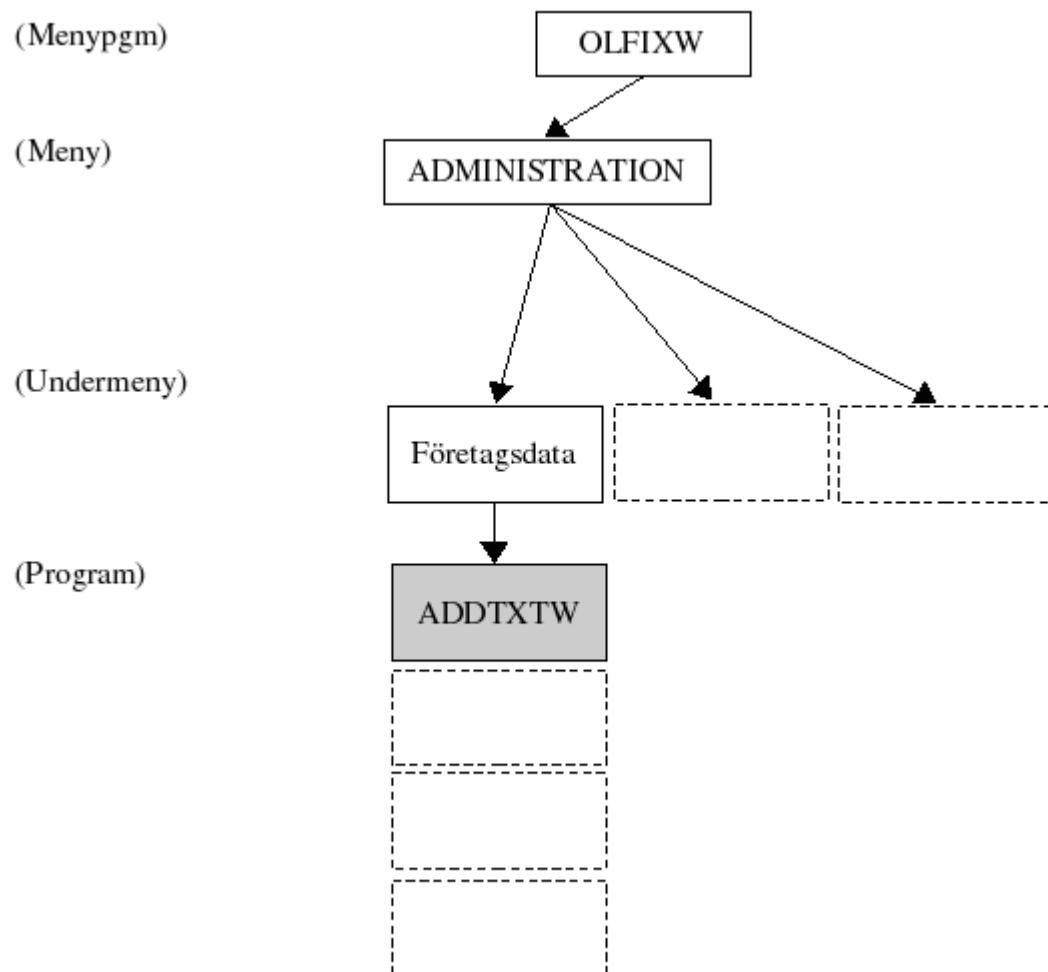
**Behörighetskrav:**

För att kunna köra ADDRGTW behövs behörighet till  
PRGLST  
RGTADD



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

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





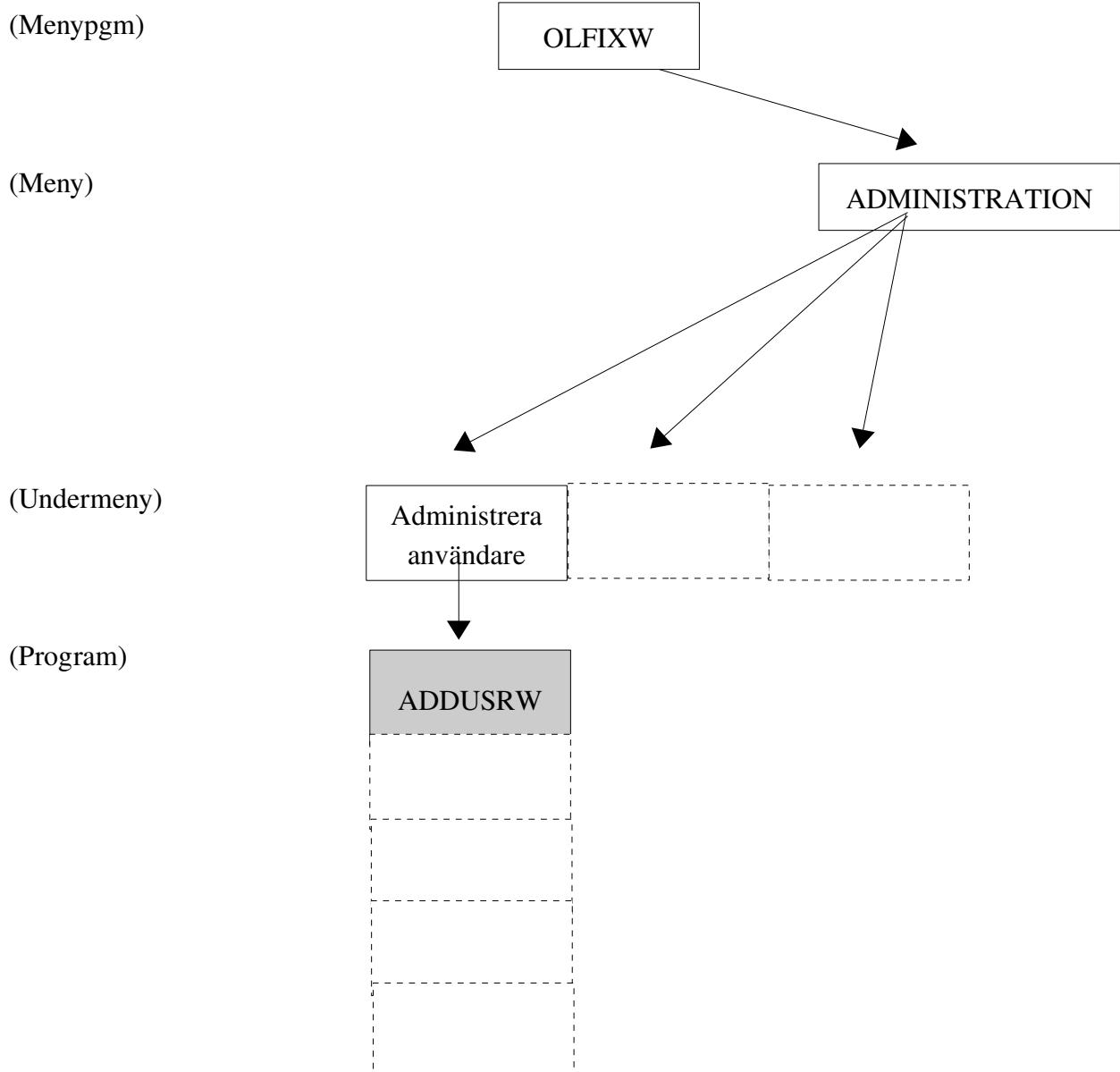
För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.  
Detta görs i main.cpp.

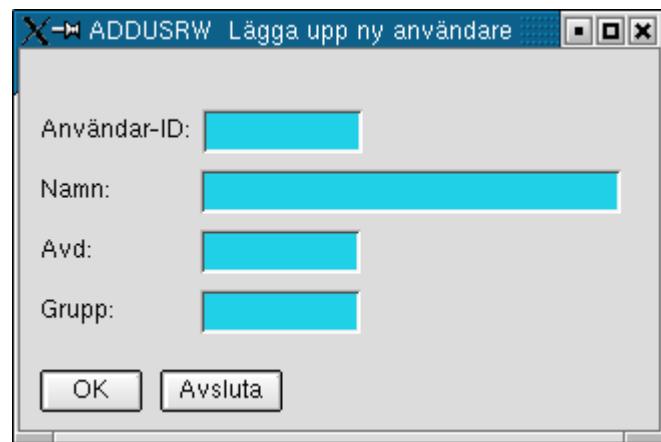
**Behörighetskrav:**

För att kunna köra ADDTWT behövs behörighet till  
PRGLST  
TXTADD

## ADDUSRW.....Ny användare

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





Användar-ID får vara max 8 tecken och skall vara detsamma som inloggningsID.

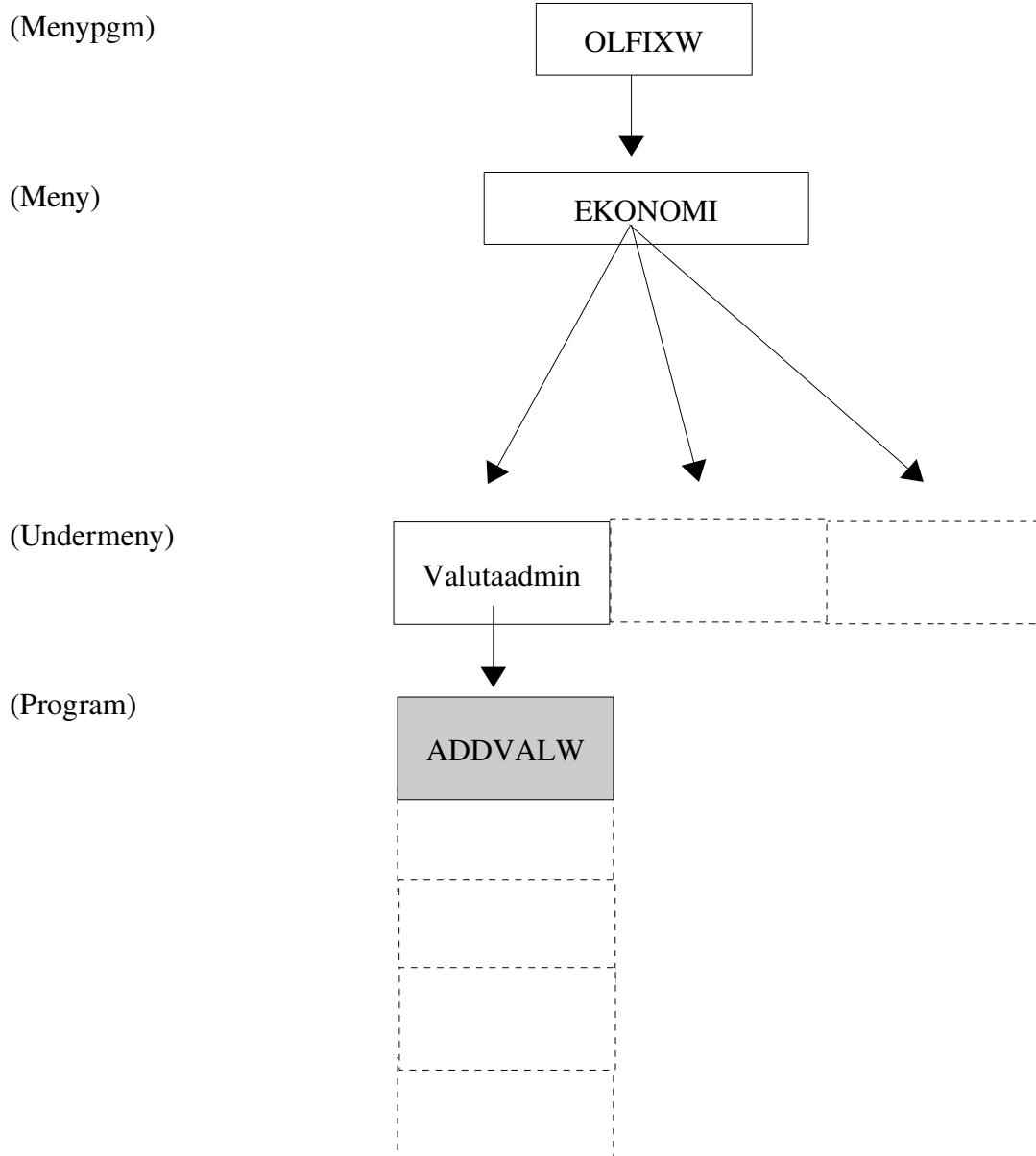
För att säkerställa att programmen körs i ett och samma bibliotek hämtas biblioteket från filen .olfixrc som ska finnas i \$HOME.  
Detta görs i main.cpp.

**Behörighetskrav:**

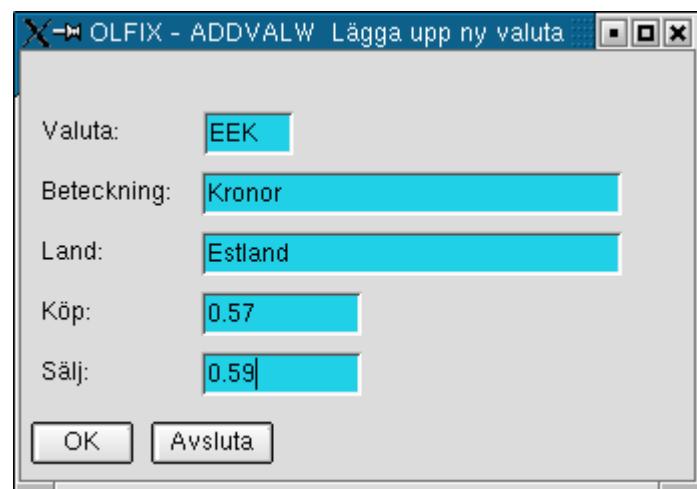
För att kunna köra ADDUSRW behövs behörighet till  
PRGLST  
USERADD

## ADDVALW.....Ny valuta

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







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

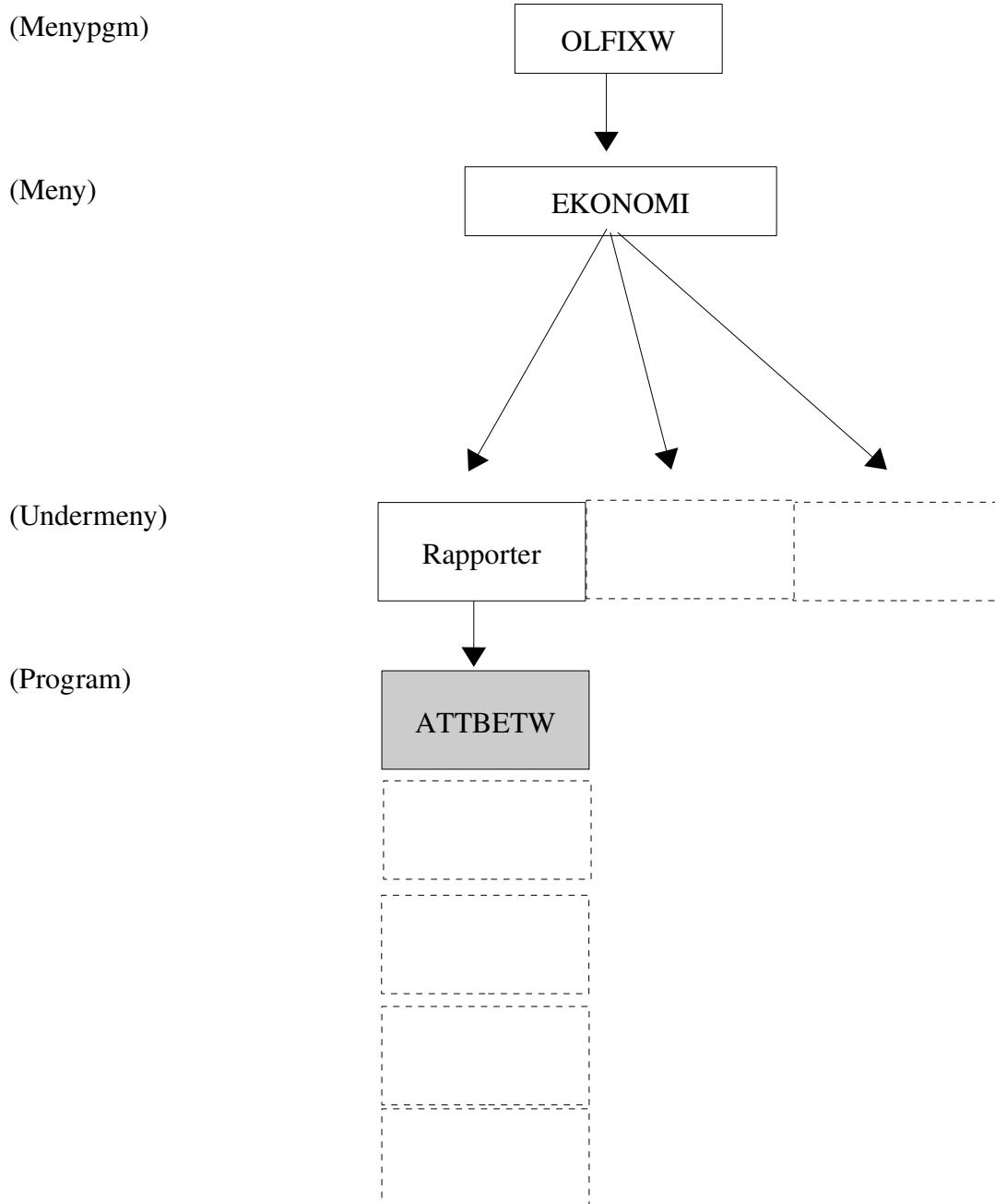
PRGLST

VALADD

## ATTBETW.....Lista obetalda leverantörsfakturor

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

Programmet plockar upp userid från environment.



## Funktionsbeskrivning.

Programmet ATTBETW skapar en datafil för rapportgeneratorn 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 /opt/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.



Kugar

Arkiv Kör Inställningar Hjälp

PROGRAM AB

Leverantörsfakturor förfallna till betalning

Förfallodatum	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

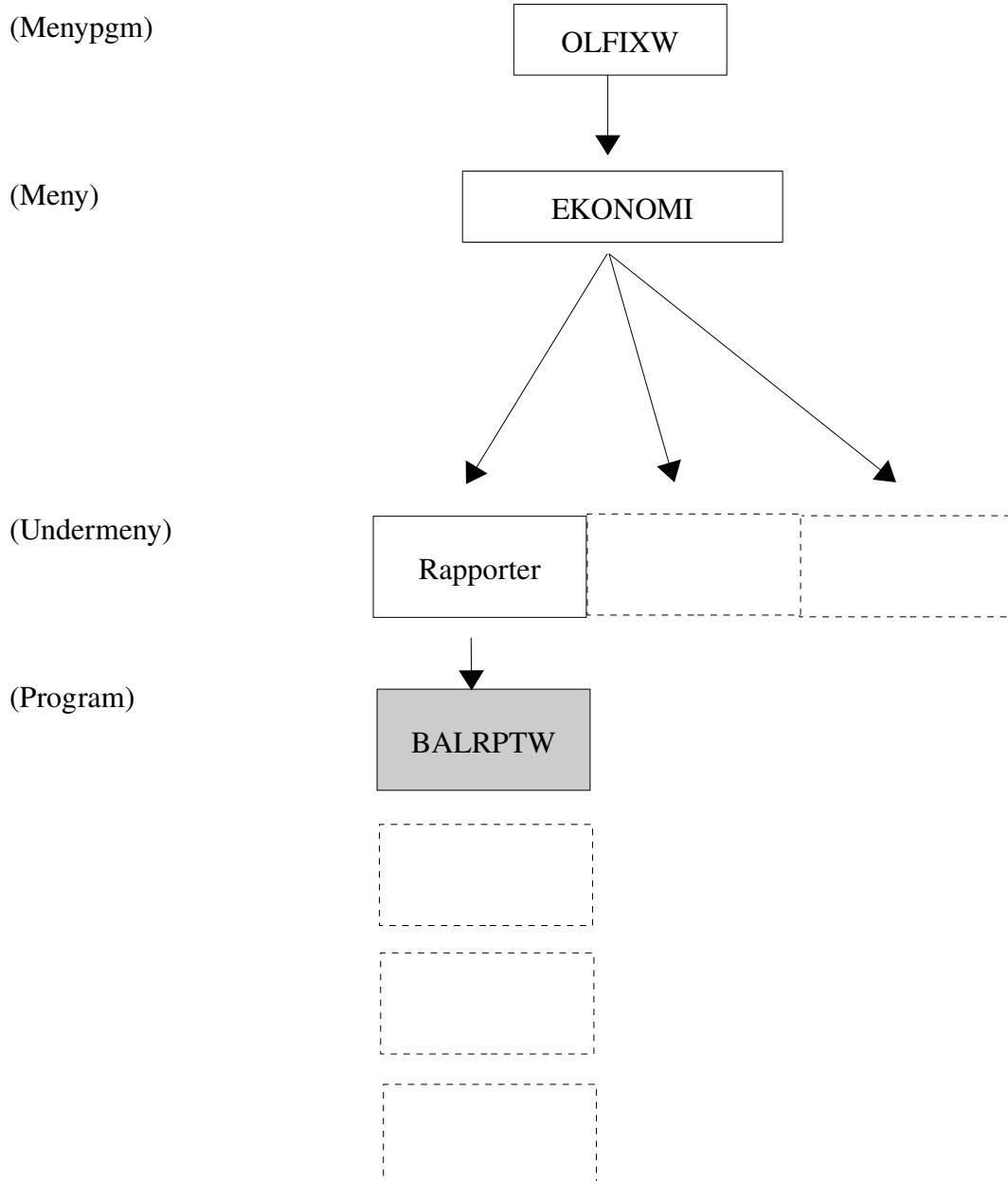
ATTBET

Programmet Kugar

Läsrättigheter i biblioteket /opt/olfix/data/

## BALRPTW.....Balansrapport

BALRPTW, ett grafiskt program för att skriva ut balansrapporter (bokföring). Programmet plockar upp userid från environment.





## PROGRAM AB

2005-01-16

Balansräkning		För perioden   2003-07-30 -- 2003-09-11		
Konto	Kontonamn	Ing Balans	Denna period	Utg Saldo
<b>1 TILLGÅNGAR</b>				
12	MASKINER OCH INVENTARIER			
1220	Inventarier	18750.00	0.00	18,750.00
<b>Delsumma</b>		<b>18750.00</b>	<b>0.00</b>	<b>18750.00</b>
<b>2 EGET OCH FRÄMMANDE KAPITAL</b>				
20	EGET KAPITAL			
2081	Aktiekapital	0.00	300000.00	-300,000.00
23	LÄNGFRISTIGA SKULDER			
2330	Checkräkningskredit	799450.00	60000.00	739,450.00
2350	Banklån	0.00	500000.00	-500,000.00
24	KORTFISTIGA SKULDER, KUNDER, LEVERANTÖRE			
2440	Leverantörsskulder	0.00	190680.00	-190,680.00
26	MOMS			
2611	Utgående moms	10215.00	0.00	10,215.00
2641	Ingående moms	49455.00	0.00	49,455.00
<b>Delsumma</b>		<b>859120.00</b>	<b>1050680.00</b>	<b>-191560.00</b>
<b>Total</b>		<b>877870.00</b>	<b>1050680.00</b>	<b>-172810.00</b>

Sida: 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.

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

BALRPTW anropar KTORPT via STYRMAN. Se KTORPT.

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

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

Detta blir:

```
./STYRMAN usr KTORPT bar fromdate todate
```

### Behörighetskrav:

För att kunna köra BALRPTW behövs behörighet till

PRGLST

FTGDSP

VERHDSP

KTORPT

PRTAPI

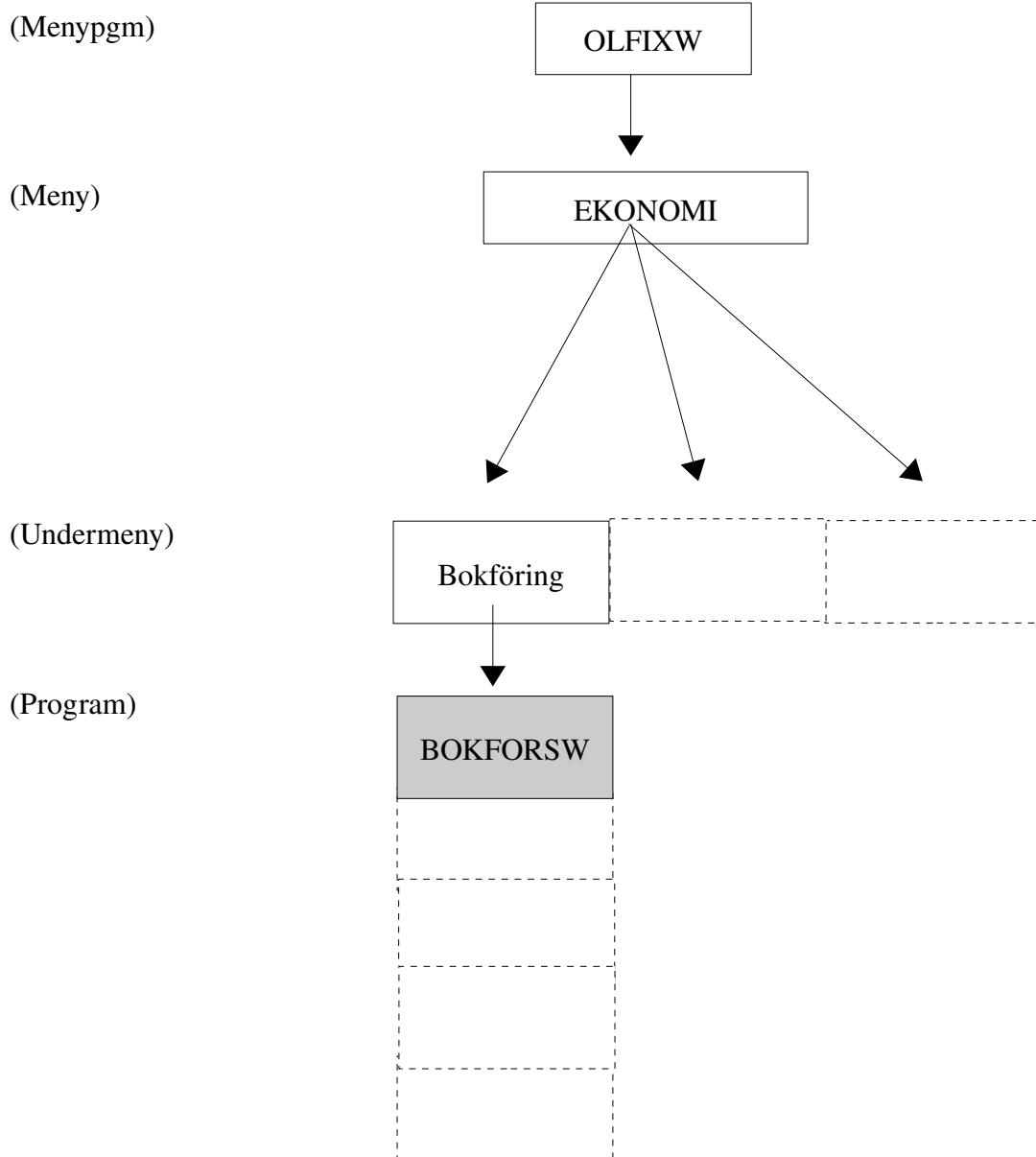
Programmet Kugar

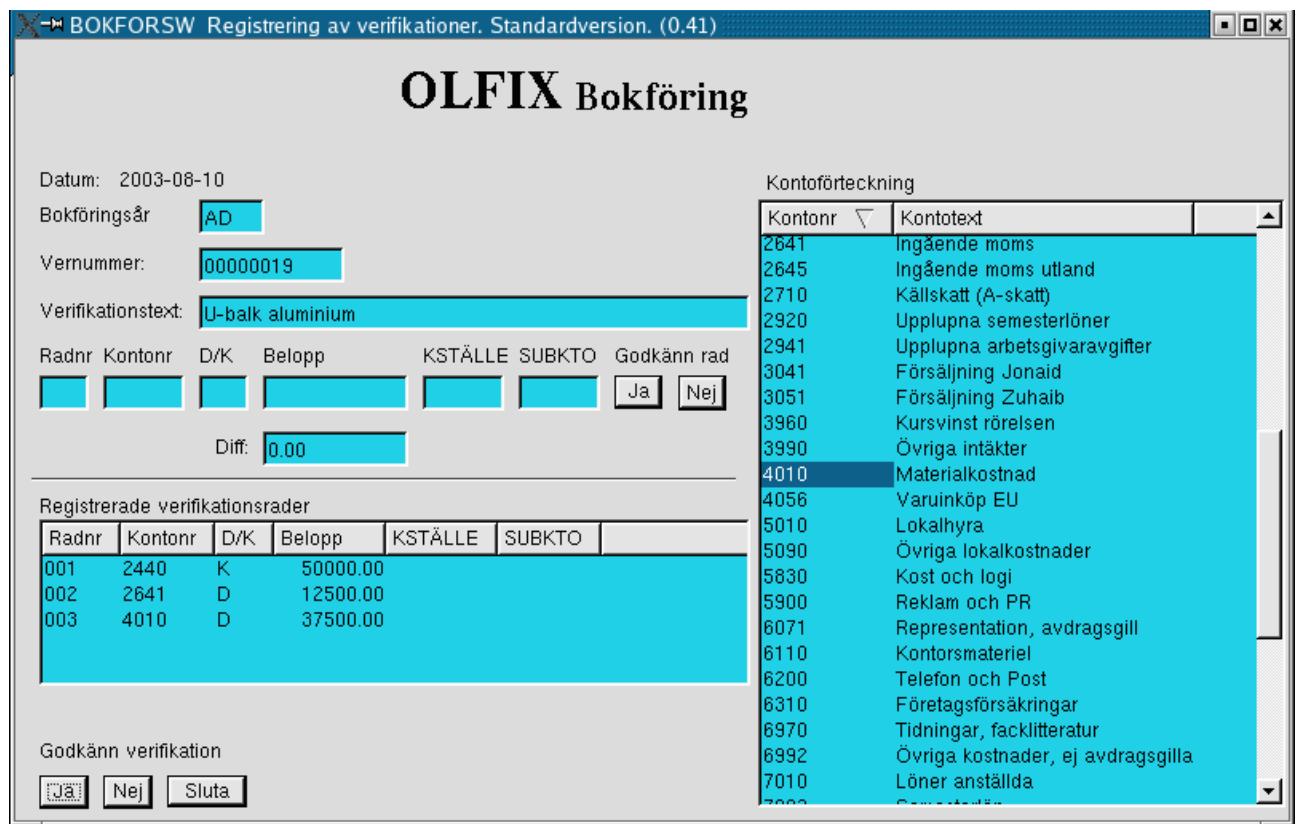
Programmet Kspread

Läsrättigheter i biblioteket /opt/olfix/data/

## BOKFORSW.....Bokföring standard

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





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

**Vernummer** sätts automatiskt.

**Verifikationstext** skrivas 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 godkänner 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**

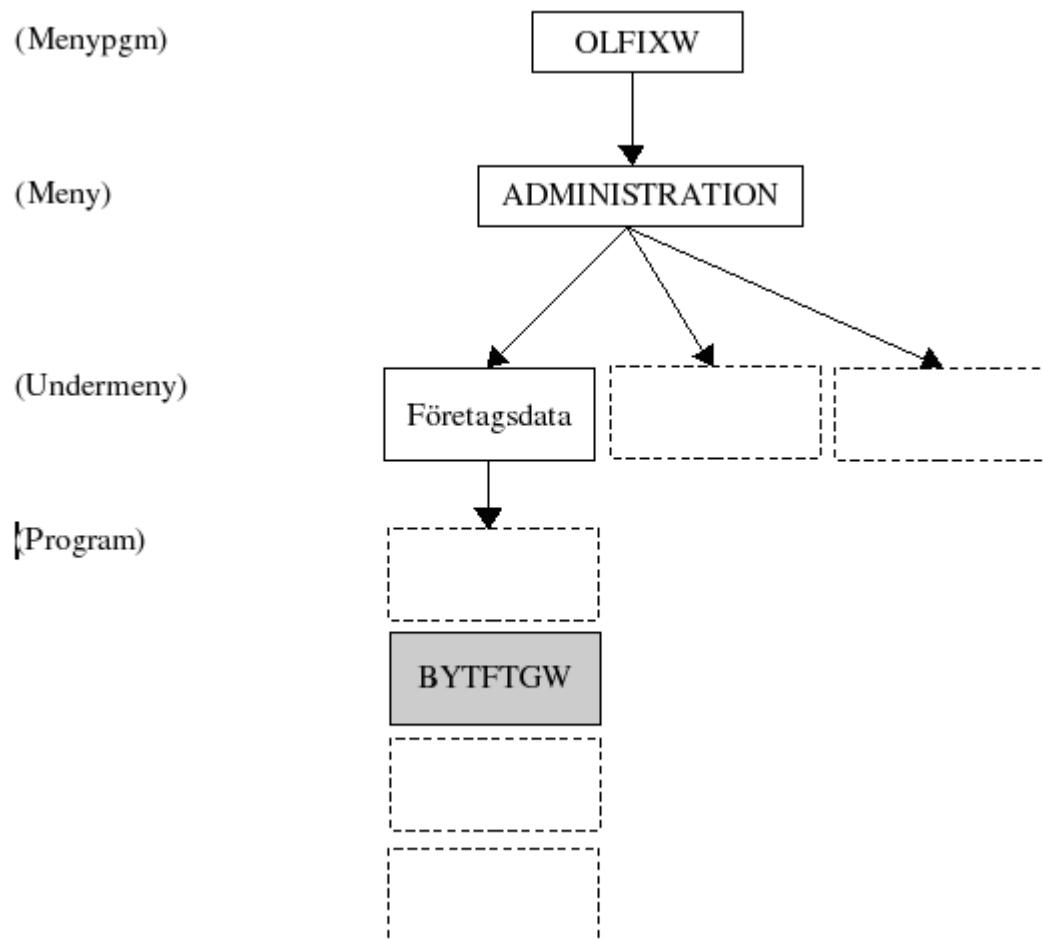
Funktionen WRREC används också.

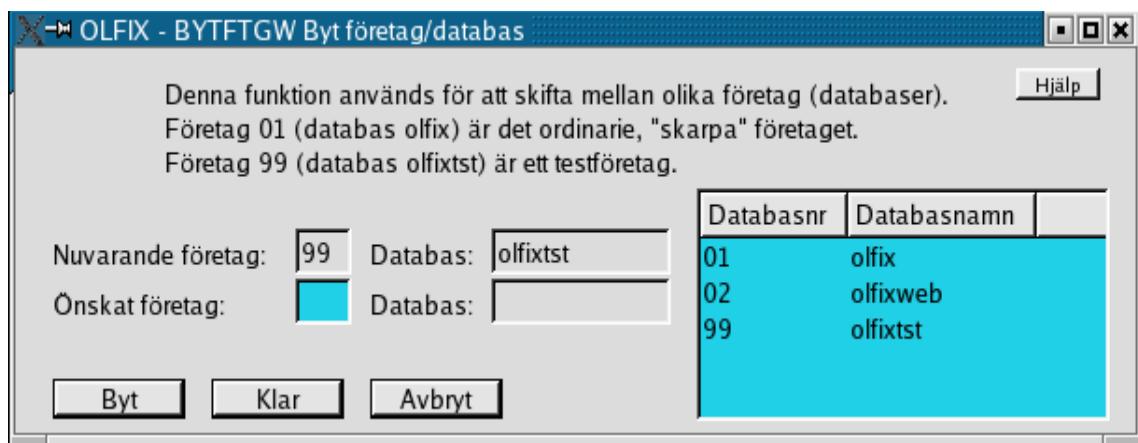
## **BYTFTGW ..... Byta företag.**

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.

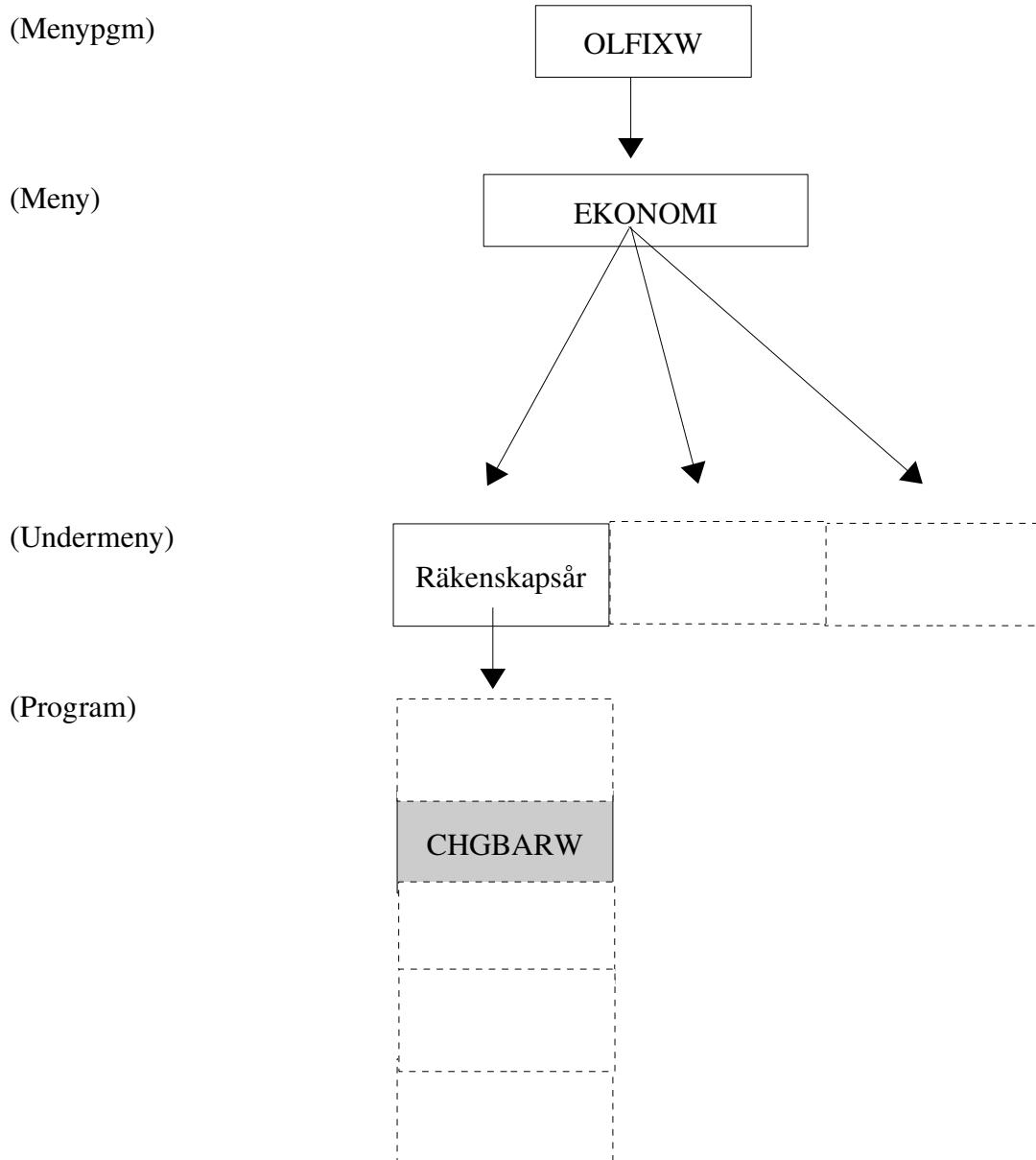
```
PATH=/home/jan/Utveckling/OLFIX/bin/  
HOST=localhost  
DATABASE=olfixtst  
VTMP=/tmp/
```

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

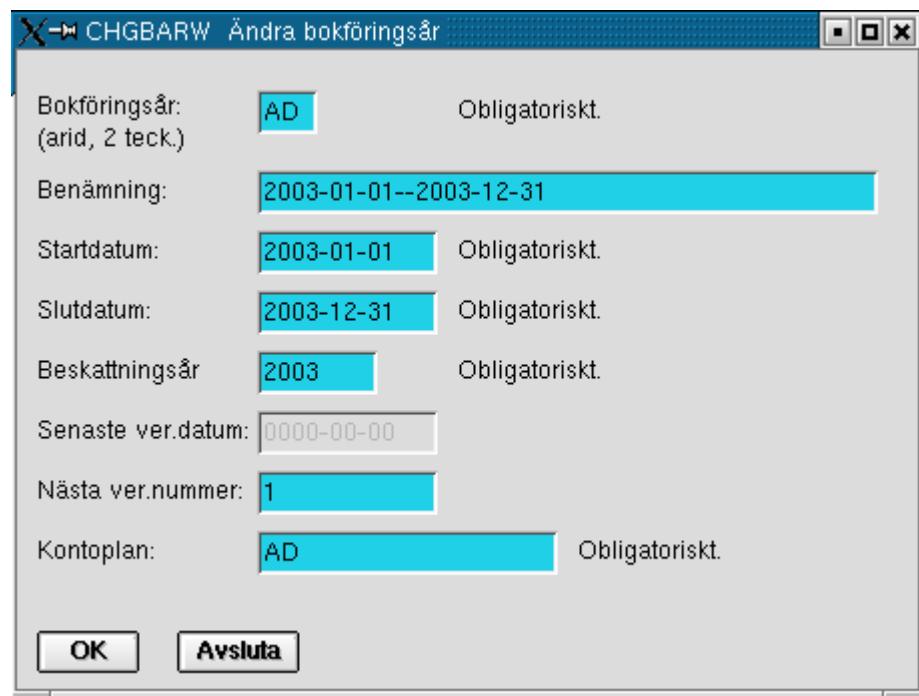
## CHGBARW....Ändra bokföringsårsdata

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

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 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(arslut);
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 arslut arlast beskattningsar  
senverdat vernr ktoplansar

usr är den inloggades userid.

### Behörighetskrav:

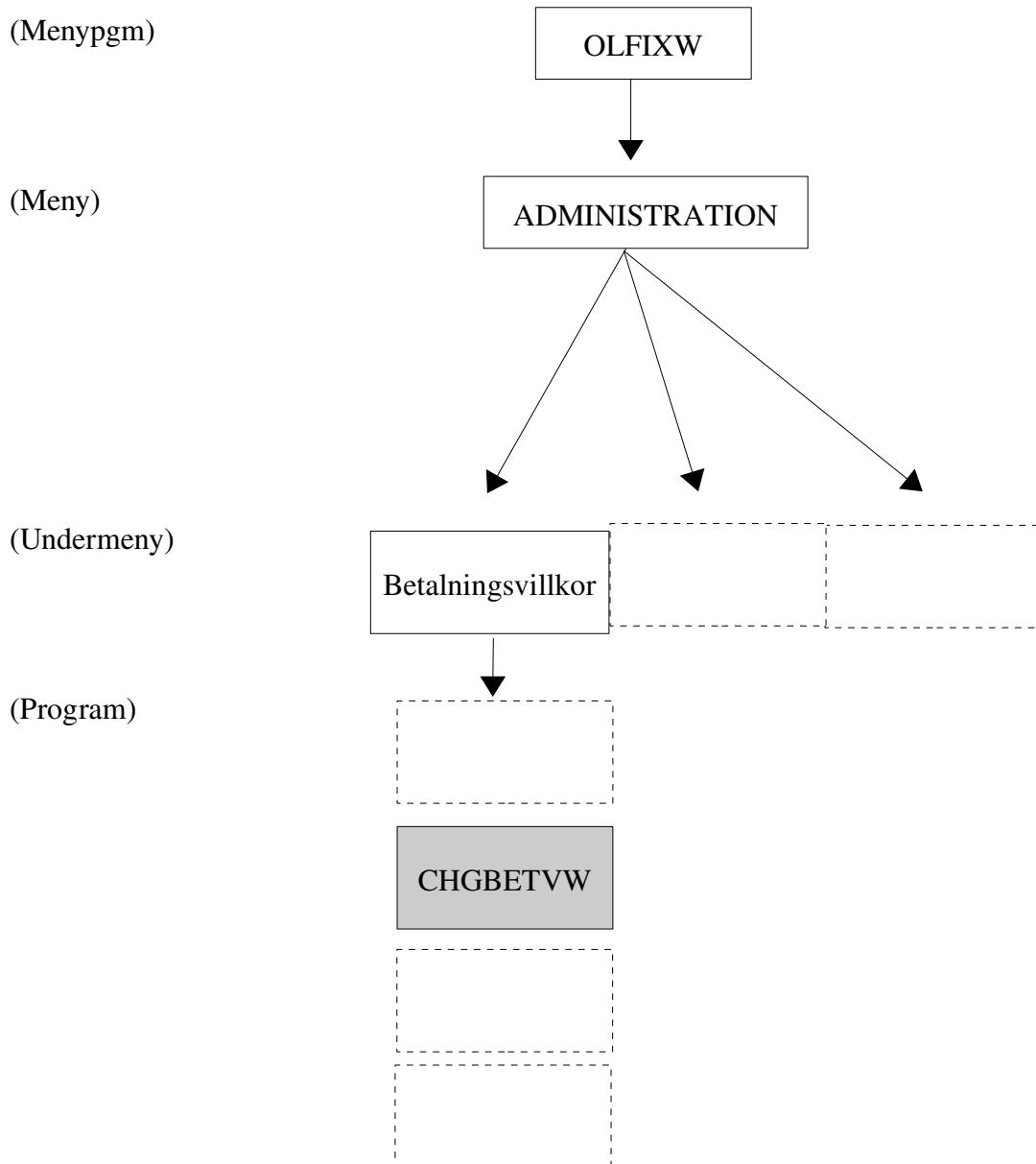
För att kunna köra CHGBARW behövs behörighet till  
PRGLST  
BARCHG  
BARDSP

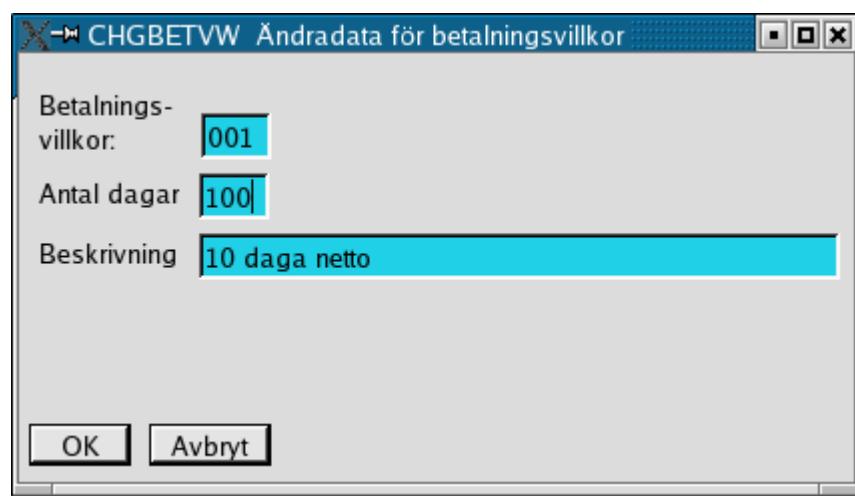


## **CHGBETVW.....Ändra betalningsvillkor**

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.







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

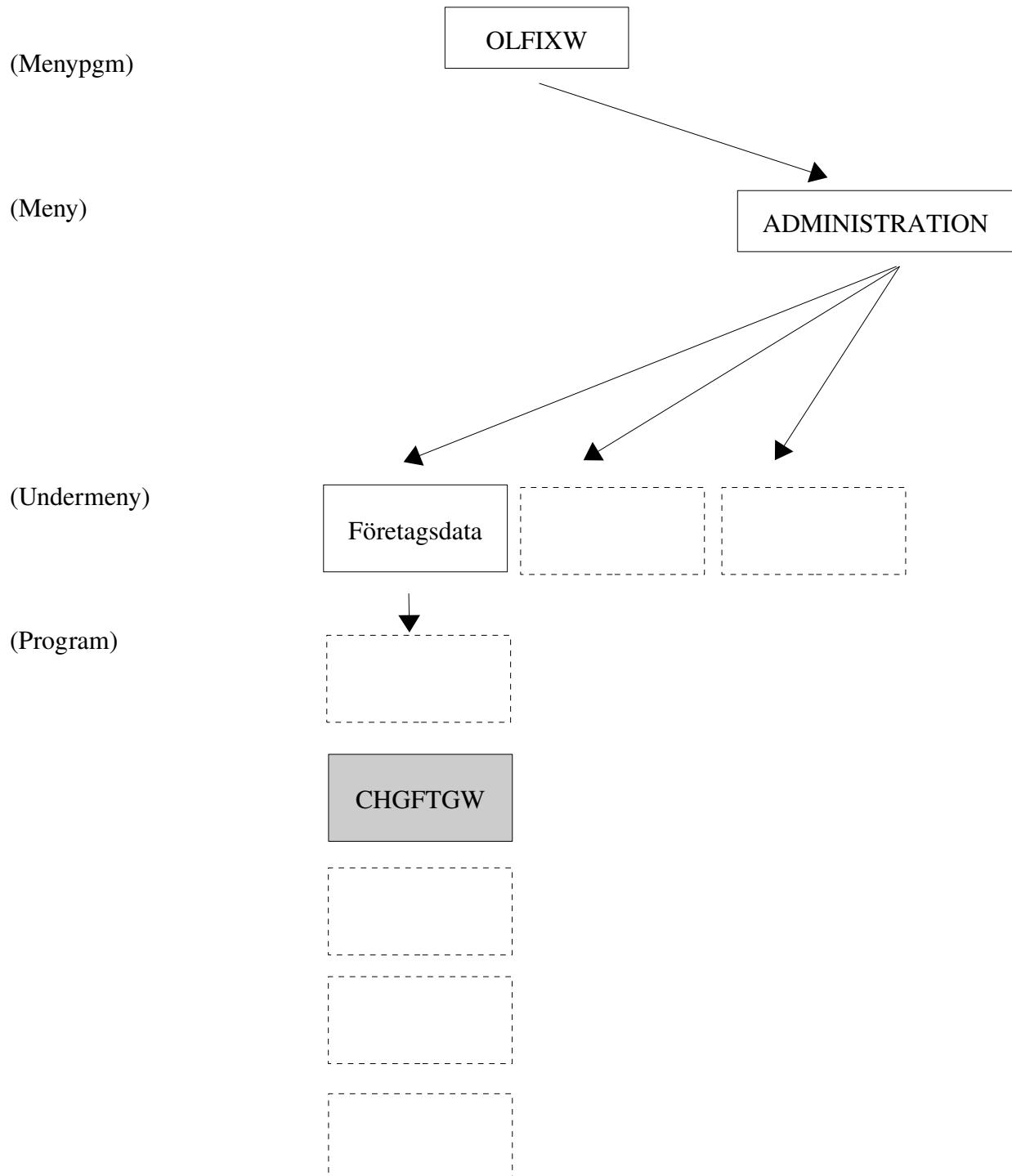
PRGLST

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.



**X-CHGFTGW - Ändra företagsdata**

Företagsnamn:	OLFIX AB	Organisationsnr:	991199-1991						
		Branschkod:	82301						
Postadress:	Box 70	Postnummer:	199 98						
Besöksadress:	Syntaxvägen 99	Postnummer:	199 98						
Godsadress:	Verktygsgatan 11	Postnummer:	199 97						
Telefonnummer:	09-199300	Mobiltelefon:	070-98765411						
e-mailadress:	jan@pihlgren.se	Telefax:	09-199397						
Telex:	11225								
Momssats 1:	25	Momssats 2:	12	Momssats 3:	6	Momssats 4:		Momssats 5:	
Momskonto, ingående moms:	.....	2641	Momskonto, utgående moms:	.....	2611				
<hr/>									
Automatkontering J/N :	N								
Kontonr kundfodringar:	1920	Kontonr inbetalning:	1511						
Antal fakturmärskerier:	2	Senaste fakturarnr:	1341	Senaste fakturarnr: 100070 Serie 2					
		Senaste inköpsordernr:	28						
Senaste kundnr:	4383	Senaste kundordernr:	63						
<hr/>									
<input type="button" value="Avhryt..."/>	<input type="button" value="Uppdatera"/>								

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="";

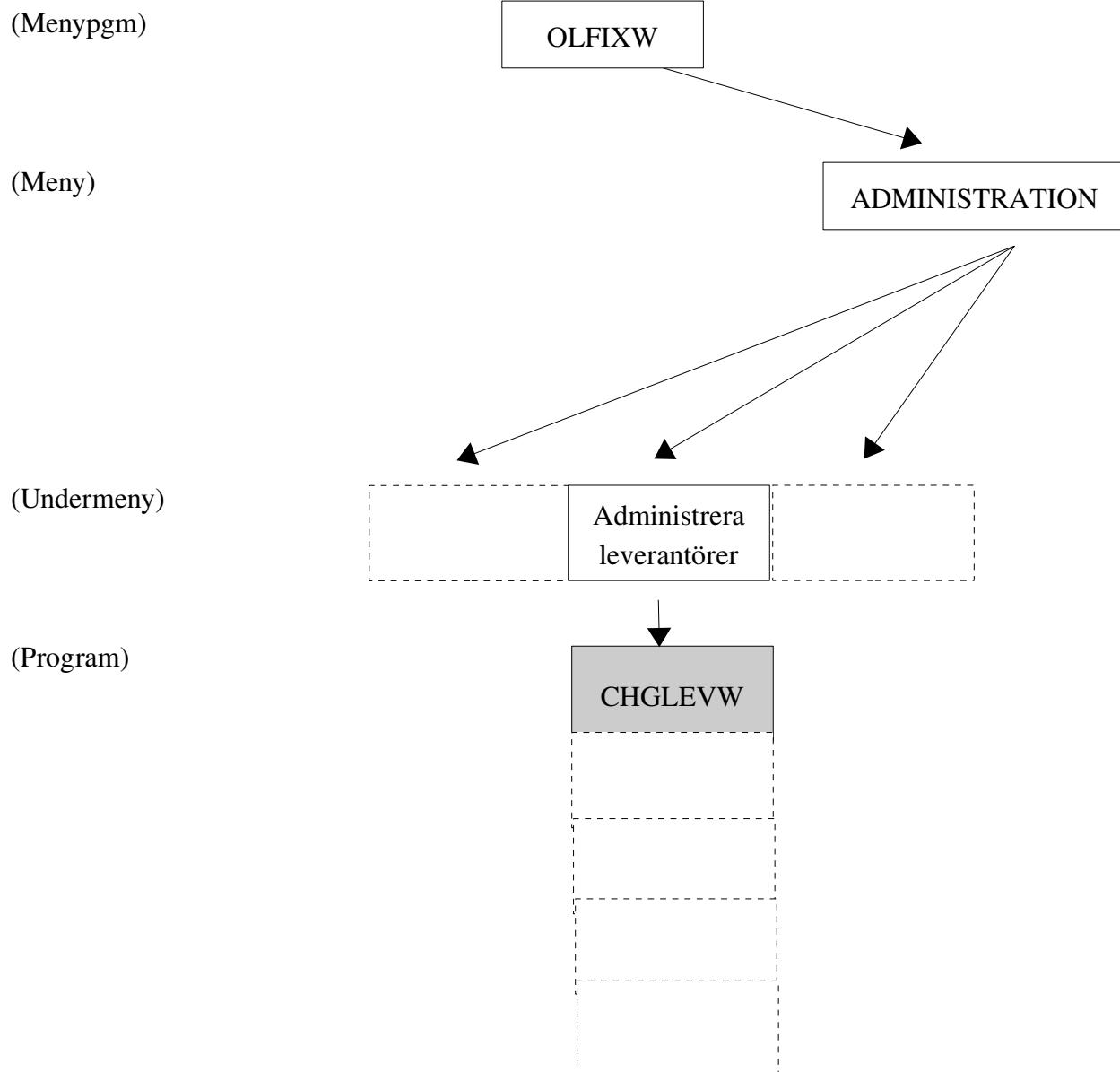
process = new QProcess();
process->addArgument("./STYRMAN");      // OLFIX styrprogram
process->addArgument(usr);              // userid
process->addArgument("FTGUPD");         // OLFIX funktion
process->addArgument(posttyp);
process->addArgument(ftgdata);
```

### 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.



X-> 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(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 LEVCHG levnr levorgnr levnamn levadress levpostnr levpostadr
levland levtfnnr levfaxnr levtelexnr levemail levpgnr levbgnr levref
levreftfnnr levmomskod levkontonr levkundnr
```

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

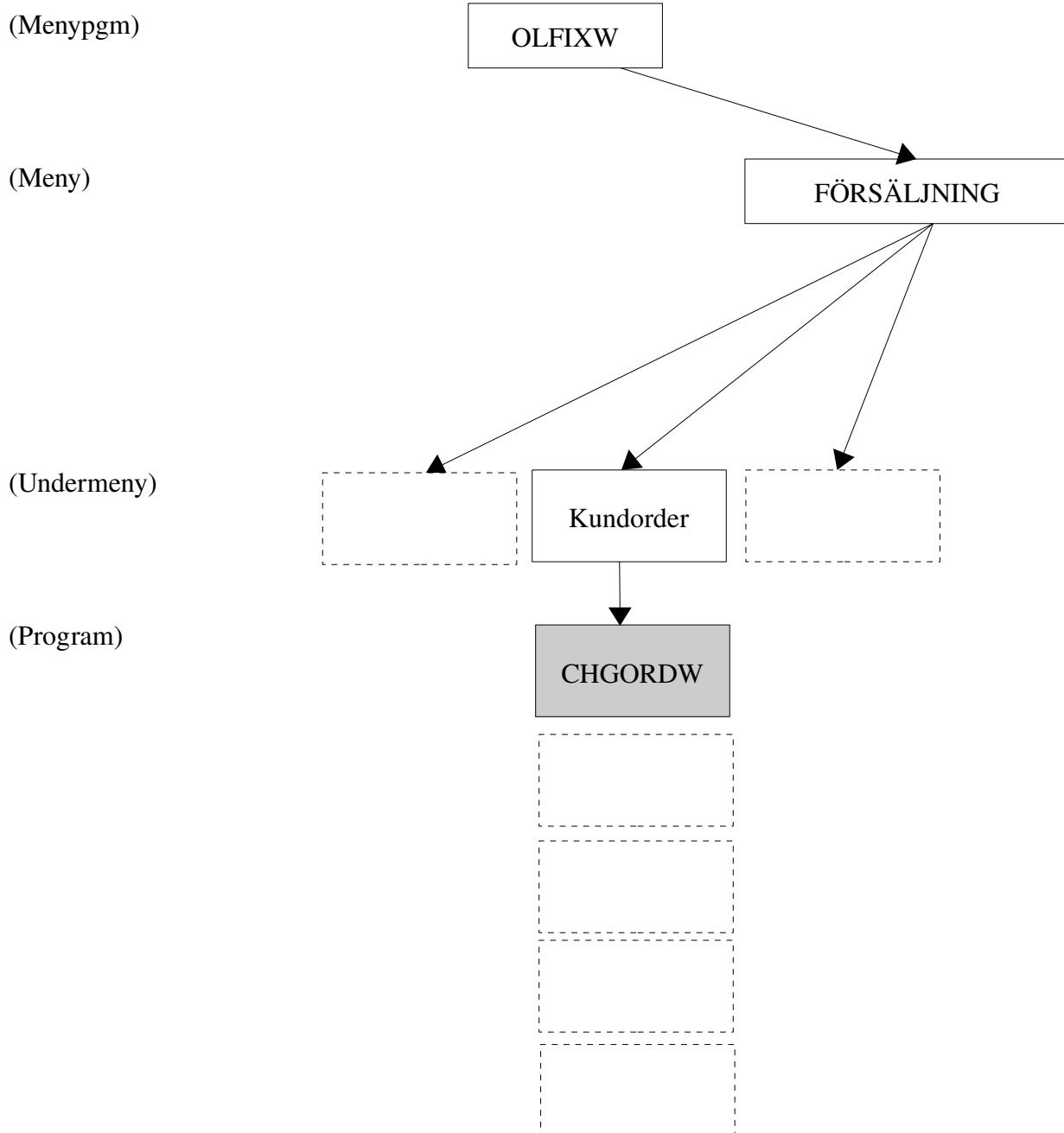
**Behörighetskrav:**

För att kunna köra CHGLEVW behövs behörighet till  
PRGLST  
LEVDSP  
LEVCHG

## CHGORDW....Ändra kundorder.

CHGORDW är ett grafiskt program för att ändra kundorder.

Programmet plockar upp userid från environment. Programmet skapar en temporärfil i katalogen \$HOME/tmp.



**CHGORDW - Ändra kundorder**

Ordernummer: ..... *	22	Datum	* = Obligatoriskt.	Hjälp				
Kundnummer: .....	4383	2006-02-17						
<b>Postadress, Fakturaadress</b>		<b>Leveransaddress</b>						
Namn: .....	Government Inc	Ref:	Tony Blair	Leveransplats .....				
Adress: .....	Downing Street 10		Downing Street 10	Leveransvillkor .....				
Postnummer: .....	G23KB1		123 45	Betainingsvillkor: ...				
Postadress: .....	LONDON		LONDON	Valuta: .....				
Land: .....	Great Brittain			Moms: ..... %				
Säljare .....	Jan Pihlgren 2	Önskad leveranstid	2006-02-17	Prislista: .....				
Godsmärke: .....				0				
Orderstatus: .....	A							
Radnr	Artikelnr *	Benämning	Leveransvecka	Antal *	Pris/st	Summa	Moms	Godkänn rad
030						0.00	0.00	Ja Nej
Radnr	Artikelnr	Benämning	Leveransvecka	Antal	Pris/st	Summa	Moms	
10	1000-1001	Att använda GNU/LINUX	6087	6.00	120.00	763.20	43.20	
20	1000-1002	Linux MAGNUM	6087	10.00	117.00	1240.20	70.20	

Summa      Frakt      Fraktmoms      Summa moms      Order Total

Bara order som inte fakturerats visas i ordertabellen till höger.

Klicka på det ordernummer som ska ändras (i listan till höger). Man kan också skriva in ordernumret (även fakturerade ordrar) manuellt. Tryck därför på TAB för att hämta orderdata.

Därefter kan man ändra önskade fält i "orderhuvudet". Vill man ändra uppgifter på en orderrad ska man klicka på den orderraden varpå orderraden flyttas upp editeringsfälten och ändringar kan utföras.

I fall man behöver lägga till en orderrad så placeras markören i det tomta fältet Artikelnr, skriver in önskat artikelnummer och trycker på TAB.

När alla ändringar är avslutade klicka på knappen Order klar och därefter på knappen Spara ordern, OK. Sedan kan man fortsätta att ändra en ny order. I annat fall avslutar man med att klicka på knappen 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 programmet CHGORDW.

Uppdatering sker med hjälp av SQL-kommando REPLACE när det gäller orderrader för att möjliggöra tillägg av nya orderrader. Se ORDRCHG.

Uppdatering av orderhuvudet sker med SQL-kommandot UPDATE. Till orderhuvudet kan inga tillägg göras. Se ORDCHG.

Utdata för orderhuvud, indata till ORDCHG är:

Ordernr, kundnamn, kundadress, postnr, postadress, land, kundens ref, leveransadress, postnr, postadress, land, säljare, orderdatum, moms (%), valuta, betallningsvillkor, leveransvillkor, leveranssätt, godsmärke, ordertotal exkl moms, frakt exkl moms, fraktmoms, total moms, ordertotal inkl moms.

Exempel:

```
_:_23:_Lilla Kunden Eftr AB _:_Bakgatan 1C _:_199 09 _:_SMÅSTAD _:_Sverige  
_:_Lillemor Andrén _:_Bakgatan 1D _:_199 09 _:_SMÅSTAD _:_:_Jan Pihlgren _:_2006-02-  
17 _:_25.00 _:_SEK _:_1 _:_001 _:_002 _:_Godsmärke jan  
_:_1755.00 _:_90.00 _:_22.50 _:_127.80 _:_1995.30 _:_END
```

Utdata för varje orderrad, indata till ORDRCHG är:

Ordernr, radnr, kundnr, artikelnr, artikelbenämning, leveransvecka(ÅVVD), antal, styckepris, radtotal inkl moms, momsbelopp, levererat antal, restnoterat antal, radrabatt, kalkylpris, leveransdatum, enhet, fakturerat antal, radordertyp.

Exempel:

```
_:_23:_10:_4375 _:_1000-1011:_Professional Linux Programming  
_:_6095:_10.00:_117.00 _:_1240.20:_70.20:_0.00:_0.00:_0.0:_0.00:_0000-00-  
00:_ST:_0.00:_N:_END
```

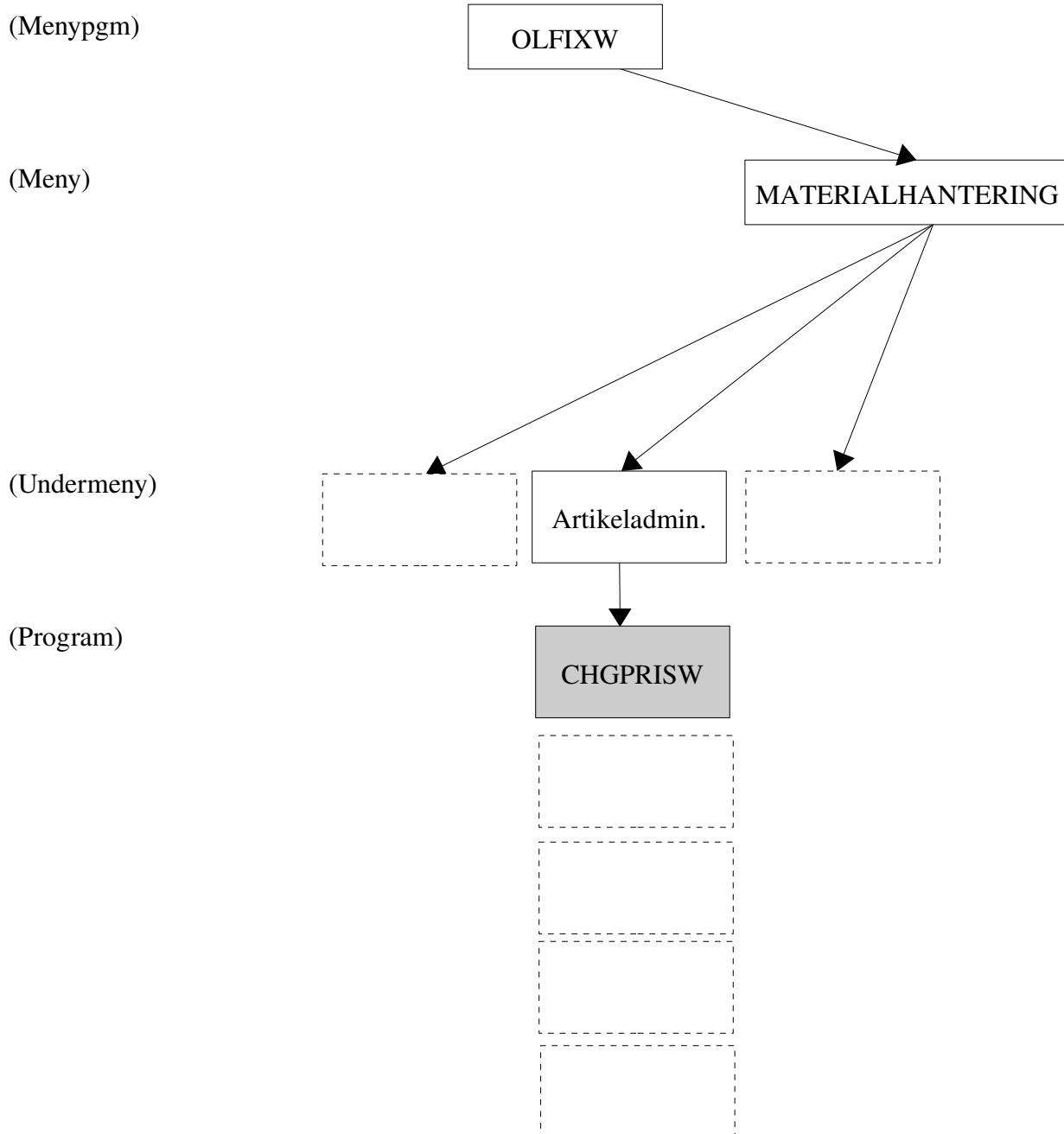
**Behörighetskrav:**

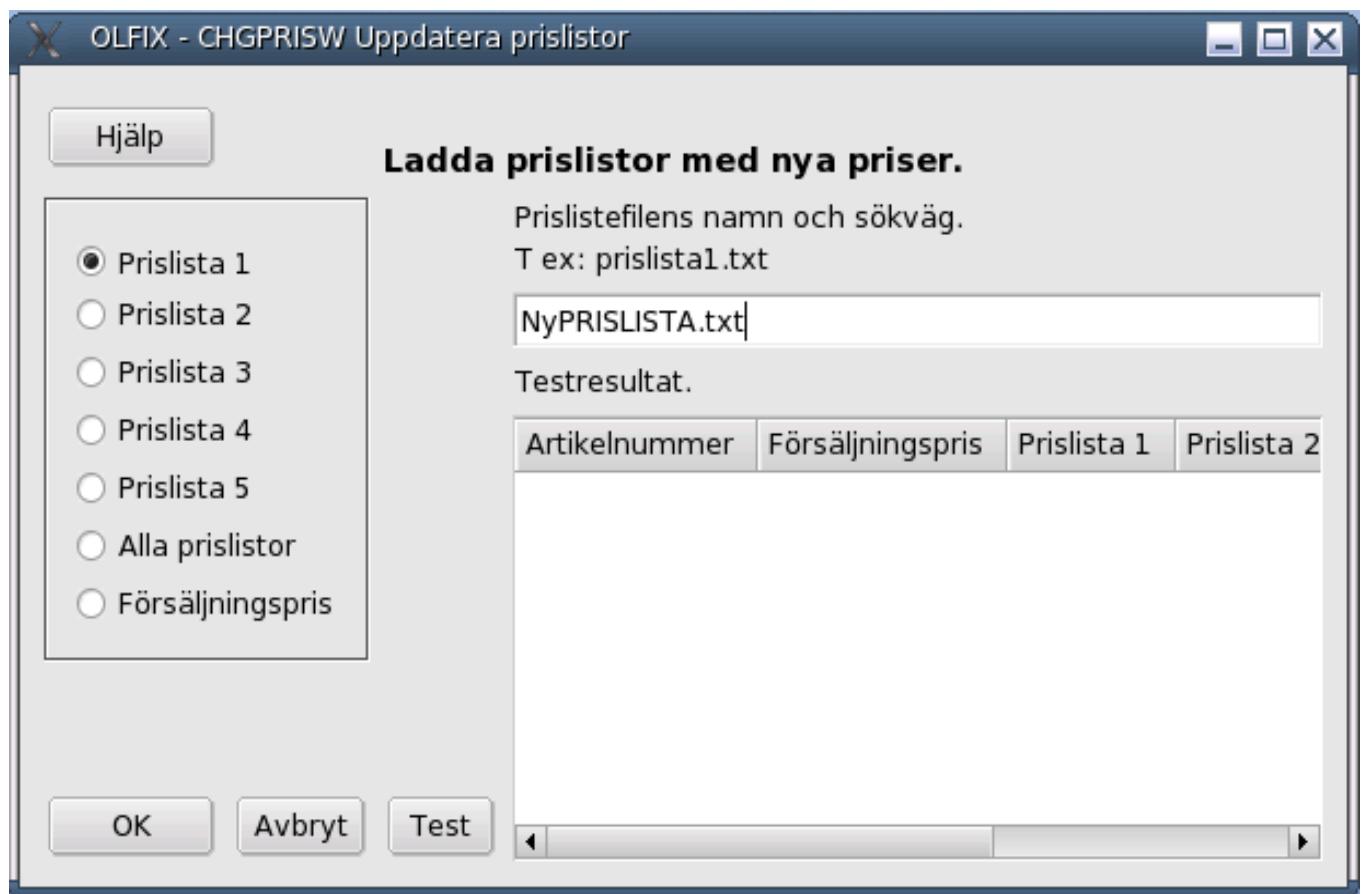
För att kunna köra CHGORDW behövs behörighet till

PRGLST  
ORDCHG  
ORDRCHG  
ORDDSP  
ORDRDSP  
ORDLST2  
FTGDSP  
USERDSP  
ARDSP  
AR2UPD  
PRISDSP  
PKDDSP

## CHGPRISW.....Uppdatera priser

CHGPRISW är ett grafiskt program för att uppdatera prislistor/priser (försäljningspriser). Programmet plockar upp userid från environment. Programmet skapar en temporär fil i katalogen \$HOME/tmp.





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 PRISUPD.

CHGPRISW anropar PRISUPD via STYRMAN.

```
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("PRISUPD"); // OLFIX funktion
process->addArgument(file);
process->addArgument(price);

file= temporärfilensnamn=$HOME/tmp/pricefile.tmp
price= prislista, ettdera av 1,2,3,4,5,F,A (1-5 är prislista1-5, F=f örsäljningspris i ARTIKELREG , A = alla priserna 1-5)
usr är den inloggades userid.
UserId är userid på den användare som man önskar data om.
```

INDATA: En kommaseparerad textfil som ska finnas i \$HOME/tmp.

Artiklarna måste finnas registrerade tidigare (ingen nyregistrering sker).

Prislistefilen har följande utseende:

```
"1000-1001", "132.00", "120.00", "110.00", "105.00", "0", "0"
"1000-1002", "352.00", "320.00", "300.00", "0", "0", "0"
"1000-1003", "385.00", "305.00", "295.00", "0", "0", "0"
"1000-1004", "355.50", "335.00", "320.00", "0", "0", "0"
"1000-1005", "275.00", "250.00", "220.00", "200.00", "0", "0"
"1000-1006", "214.50", "195.00", "0", "0", "0", "0"
"1000-1007", "93.50", "85.00", "0", "0", "0", "0"
"1000-1008", "225.50", "205.00", "0", "0", "0", "0"
"1000-1009", "415.00", "0", "0", "0", "0", "0"
"1000-1010", "395.00", "0", "0", "0", "0", "0"
"1000-1011", "365.50", "0", "0", "0", "0", "0"
"1000-1012", "478.50", "435.00", "415.00", "0", "0", "0"
"1000-1014", "423.50", "385.00", "375.00", "0", "0", "0"
```

Artikelnr, försäljningspris, prislista1, prislista2, prislista3, prislista4, prislista5.  
Fälten ska omges av situationstecken ( " ). Fälten åtskiljs av kommatecken ( , ).

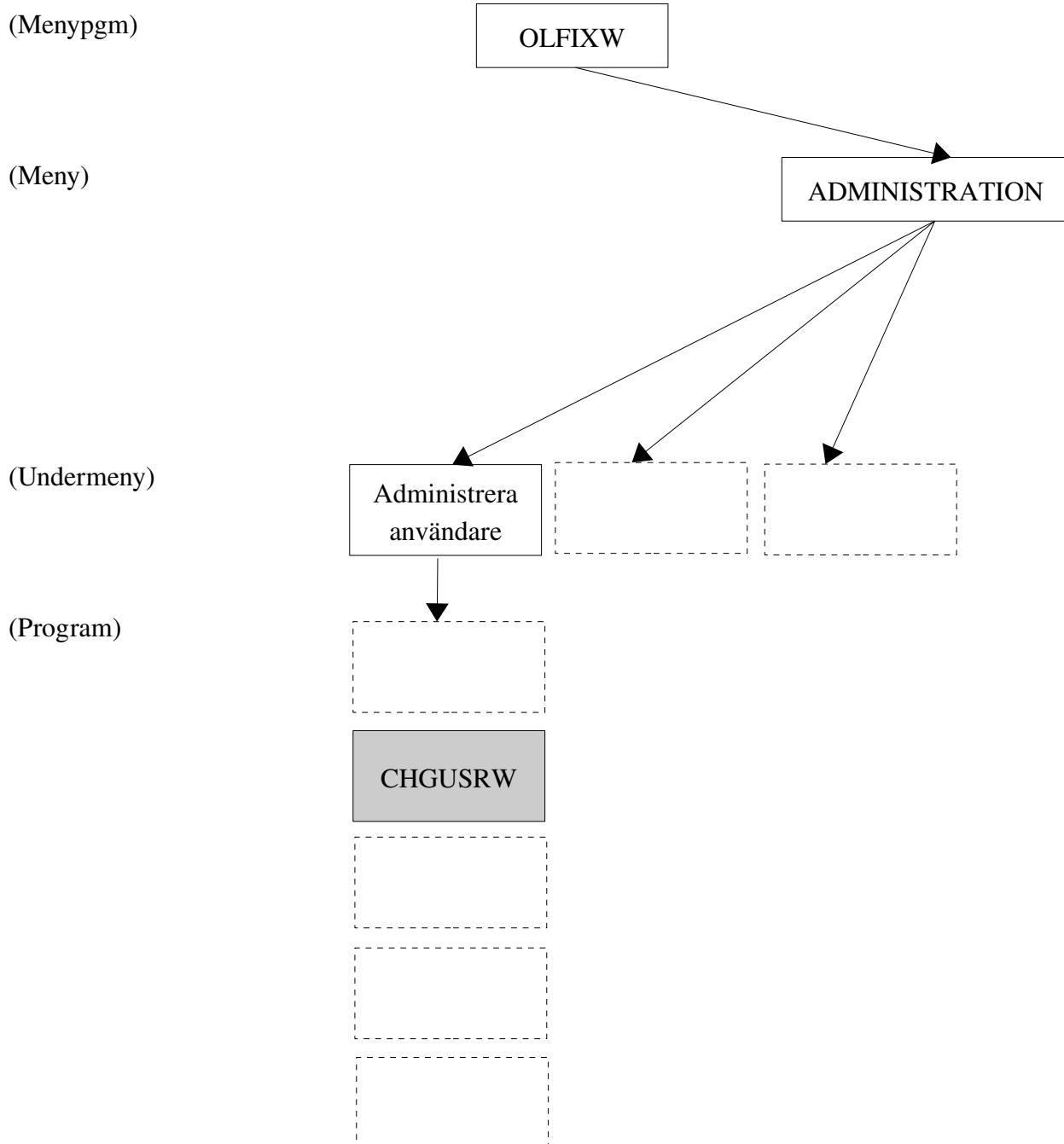
#### Behörighetskrav:

För att kunna köra CHGPRISW behövs behörighet till  
PRISUPD  
PRGLST

## CHGUSRW.....Ändra användardata

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

För att använda programmet krävs behörighet till funktionerna USERDSP, USERCHG och USERLST. Programmet plockar upp userid från environment.



**CHGUSRW Ändra användare**

Användar-ID	JAN
Namn:	Jan Pihlgren 2
Avd:	IT
Grupp:	Stab
<b>Uppdatera</b> <b>Avsluta</b>	

Användarlista

Namn	Använda
Administratör av OLFIX	ADMINA
Caroline Inköpare	CARRO
Guest Apache	GUEST
Jan Pihlgren 2	JAN
Olfix Superuser	OLFIX
Testare Test	TESTARE

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, USERCHG och USERLST via STYRMAN.

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

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.latin1()); // user OLFIX
process->addArgument("USERDSP"); // OLFIX program
process->addArgument( Userid.latin1() );
```

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.

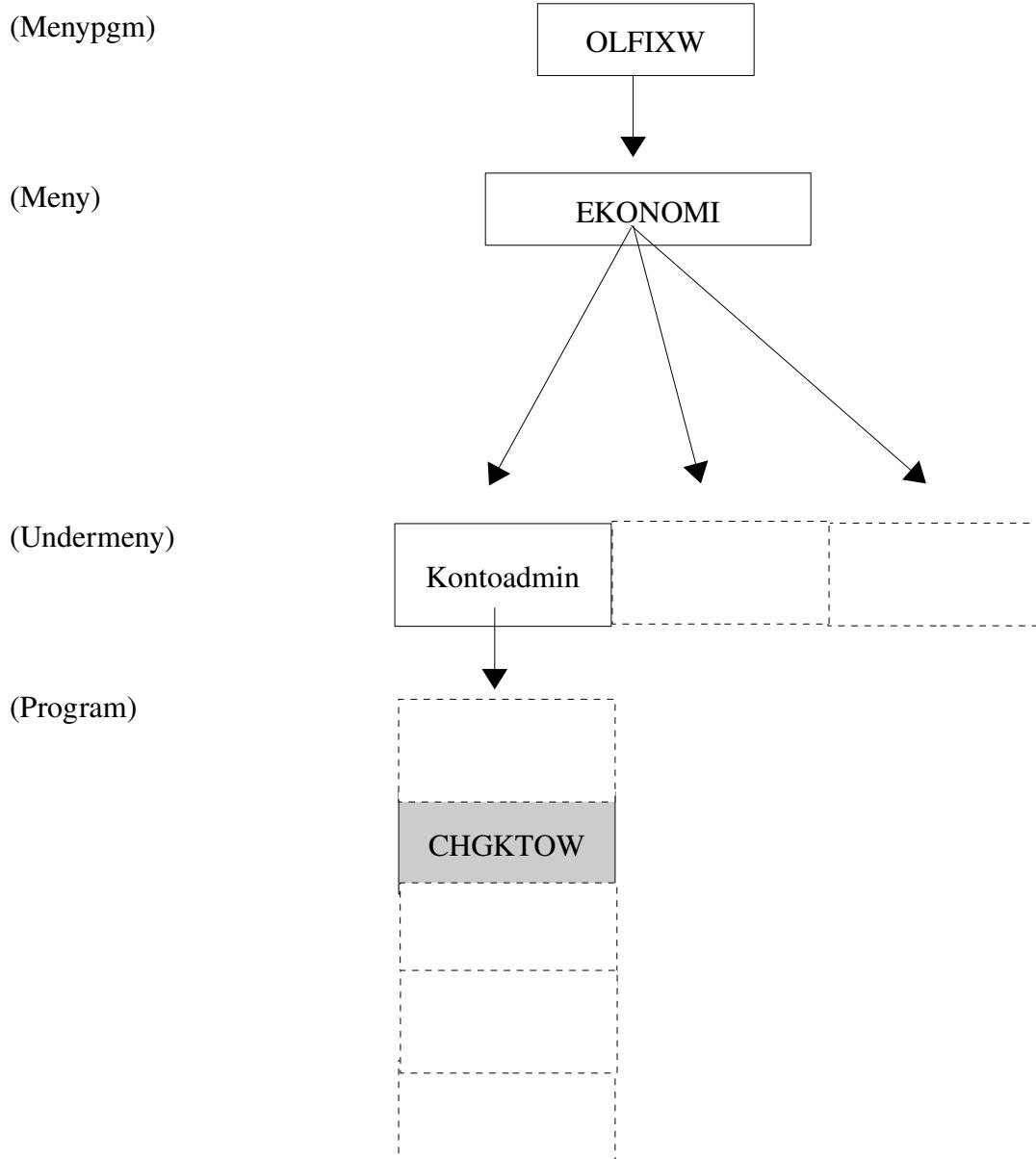
### Behörighetskrav:

För att kunna köra CHGUSRW behövs behörighet till  
PRGLST  
USERDSP  
USERDSP  
USERLST

## 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.





X-CHGKTOW Ändra konto

Bokföringsår: (arid, 2 teck.)	AC	Obligatoriskt.
Kontonummer:	1020	Obligatoriskt. <input type="button" value="Hämta"/>
Benämning:	Postgiro	
Manuell (J/N):	N	Obligatoriskt.
Momskod:	1	Obligatoriskt.
SRUnr:	100	Obligatoriskt.
Kostnadsställe:		
Projekt:		
Subkonto:		
Kontoplan:	BAS90 Obligatoriskt.	
IB:	0.00	
UB:	0.00	
<input type="button" value="OK"/> <input type="button" value="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 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.

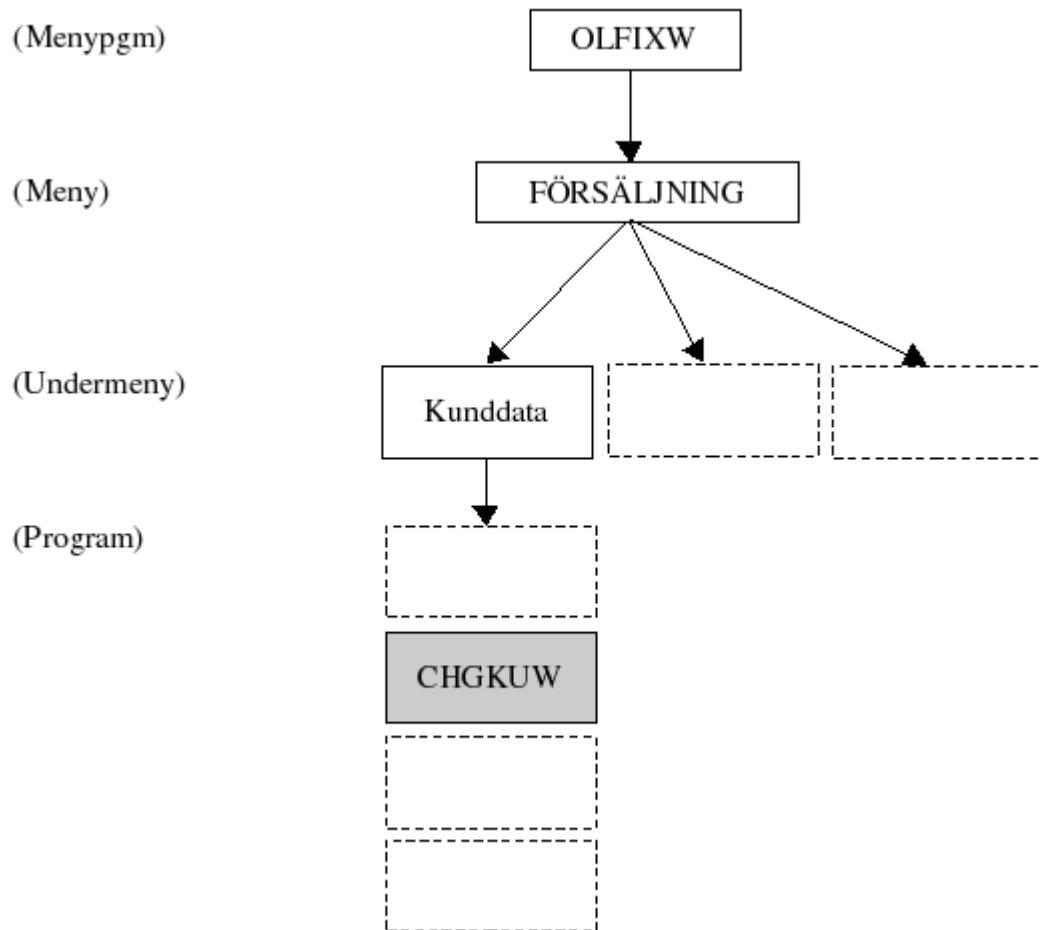
### Behörighetskrav:

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 (kandid). Programmet plockar upp userid från environment.



CHGKUW - Ändra kunddata.

KundID: .....	4378	Kundlista.	Hjälp
Organisationsnr: ...	[NULL]		
Kundnamn: .....	Nya Storkund AB		
Kundadress: .....	Fina gatan 2		
Postnummer: .....	100 01		
Postadress: .....	LYXBY		
Land: .....	Sverige		
Telefonnummer: ...	09-109990		
Faxnummer: .....	09-109999		
E-mail: .....	info@storkund.se		
Er Referent: .....	Carl von Petersen		
Er Ref's telefonnr: ..	09-109991		
Er Ref's e-mailadr: .	c.p@storkund.se		
Vår säljare: .....	Carolina Seljare		
Distrikt: .....	LYX	Kundkategori: .....	001
Leveransplats: ....	001	Leveransvillkor: ....	001
Betalningsvillkor:	SEK		
Valuta: .....	sv	Språkkod: .....	J
Ordererkännande: .	J	Plocklista :	J
Expeditionsavgift: .	J	Fraktavgift: .....	J
Kreditlimit: .....	0.00	Kreditdagar	45
Exportkod: .....		Skattekod: .....	
Dröjsmålsränta: ....	J	Dröjsmålsfaktura: ..	J
Senaste kravdatum	0000-00-00	Skuld: .....	0.00
Orderstock: .....	0.00		
Fri text (100 tkn): ..	Detta är en stor kund		

**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 och KULST 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);
```

och

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

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

Detta blir:

./STYRMAN usr KUDSP kundnr

och

./STYRMAN usr KUCHG kunddata

och

./STYRMAN usr KULST

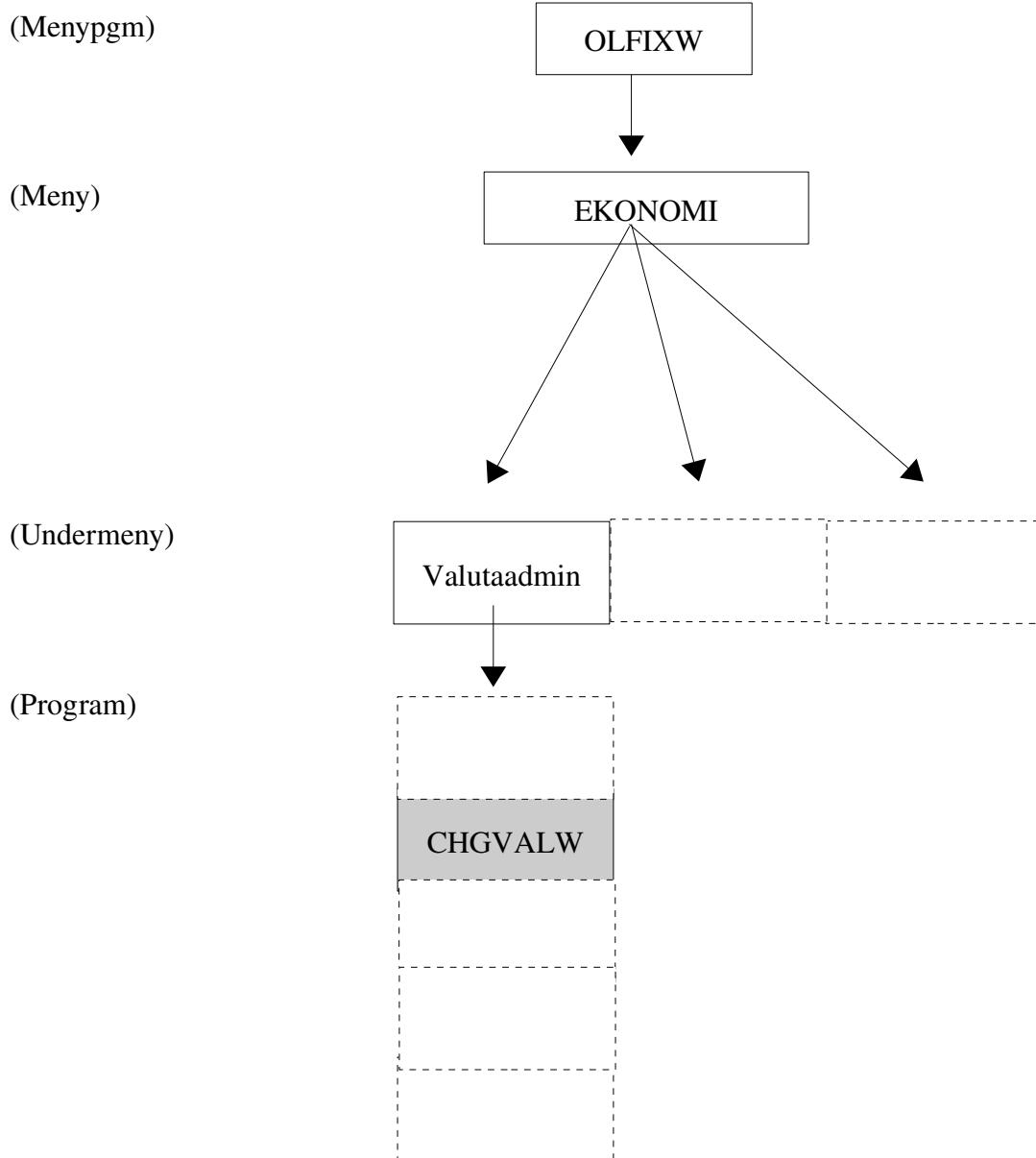
usr är den inloggades userid.

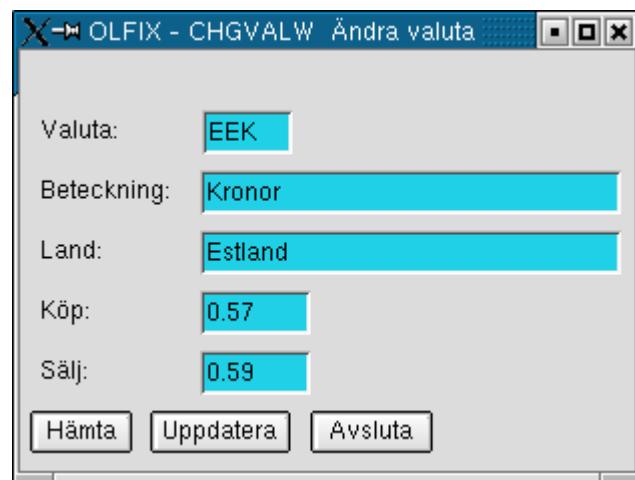
**Behörighetskrav:**

För att kunna köra CHGKUW behövs behörighet till  
PRGLST  
KUCHG  
KUDSP  
KULST  
OLFIXHLP

## **CHGVALW.....Ändra valutadata**

CHGVALW, ett grafiskt program för att ändra info för ett konto.  
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 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.

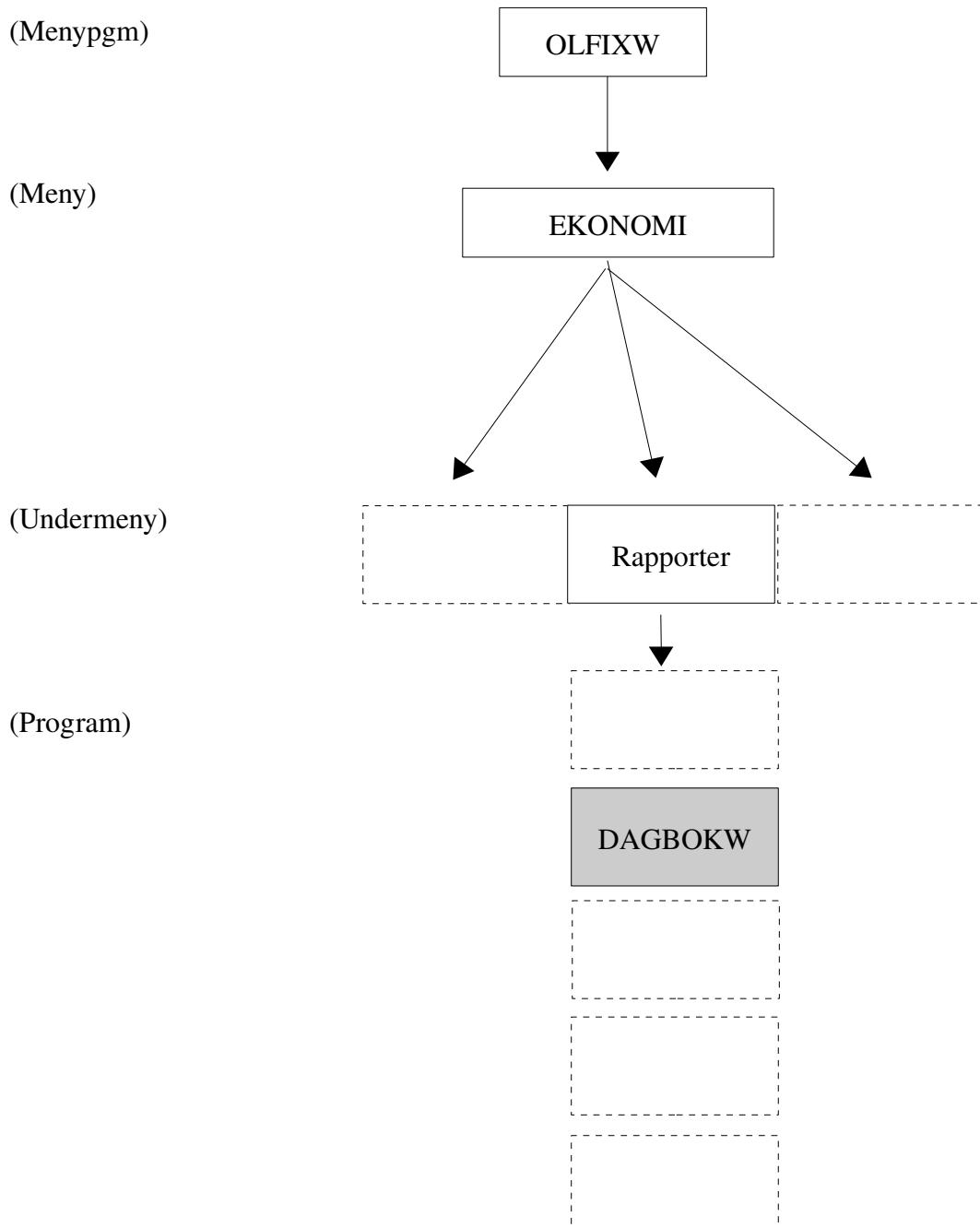
### Behörighetskrav:

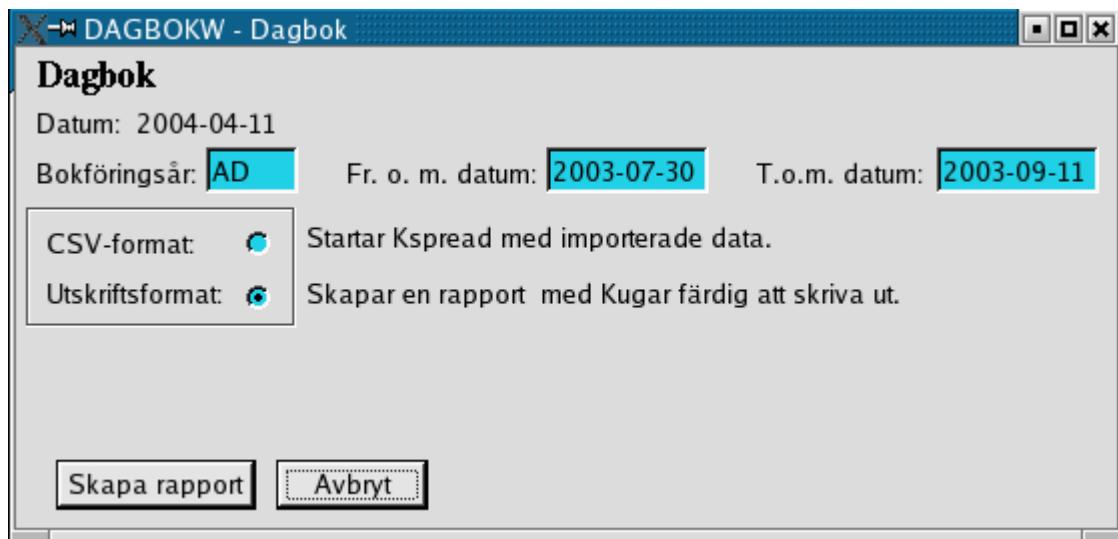
För att kunna köra CHGVALW behövs behörighet till  
PRGLST  
VALCHG  
VALDSP



## DAGBOKW.....Dagbok

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





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



Exempel

**Kugar**

Arkiv Kör Inställningar Hjälp

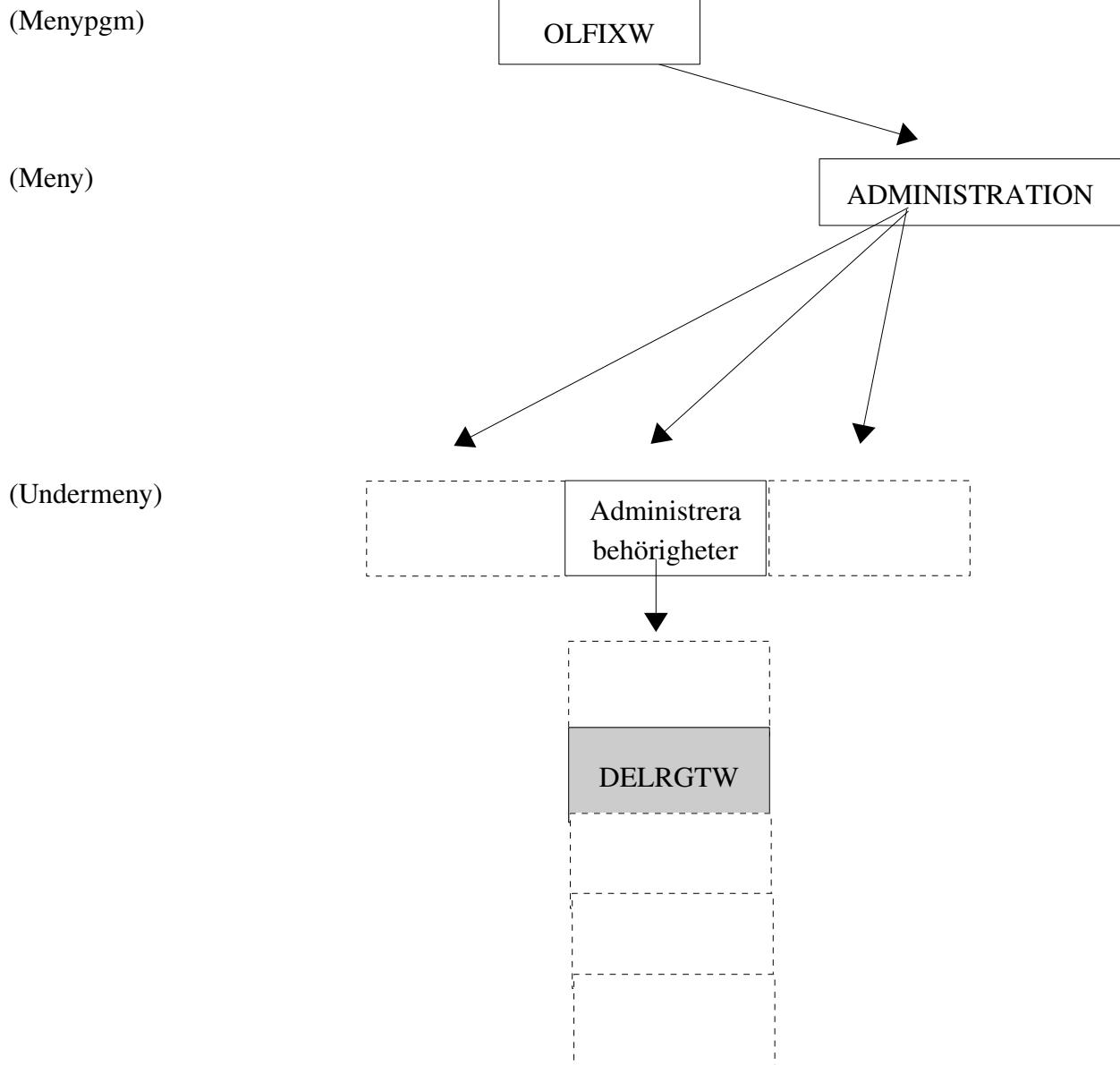
PROGRAM AB 2004-04-11

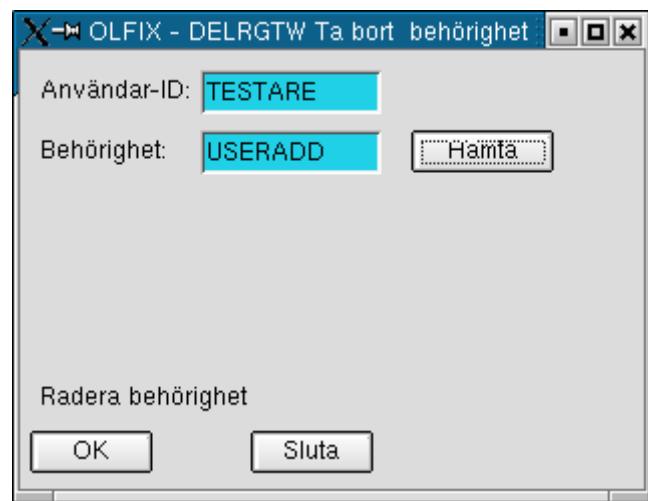
Dagbok För perioden 2003-07-30 -- 2003-09-11

Konto Vernr	Verifikationstext	Debet	Kredit
<b>Ver: 1 2003-07-30 Lån eget kapital</b>			
2081 Aktiekapital	0.00	300000.00	
2330 Checkräkningskredit	300000.00	0.00	
<b>Ver: 2 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: 3 2003-07-30 Lokalhyra 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: 7 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: 8 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: 9 2003-08-03 Järnplåt. 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: 10 2003-08-04 Järnplåt. 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: 11 2003-08-05 Järnplåt. 0,3 mm. 3 m3.</b>			

## **DELRGTW.....Radera behörighet**

DELRGTW är ett grafiskt program för att ta bort en behörighet för en användare.  
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 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

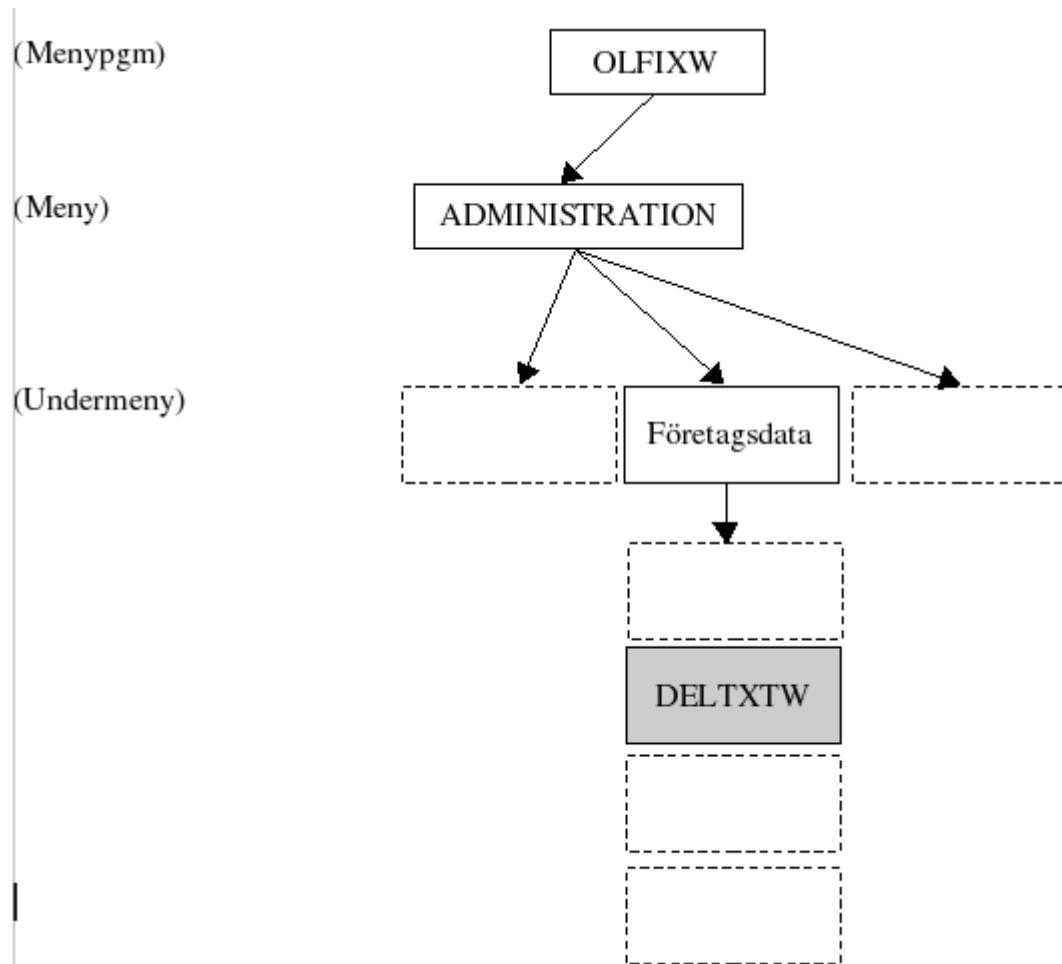
usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DELRGTW behövs behörighet till  
PRGLST  
RGTCHK  
RGTDEL

## **DELTXTW.....Radera text.**

DELTXTW 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 och TXTDEL

DELTXTW 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:

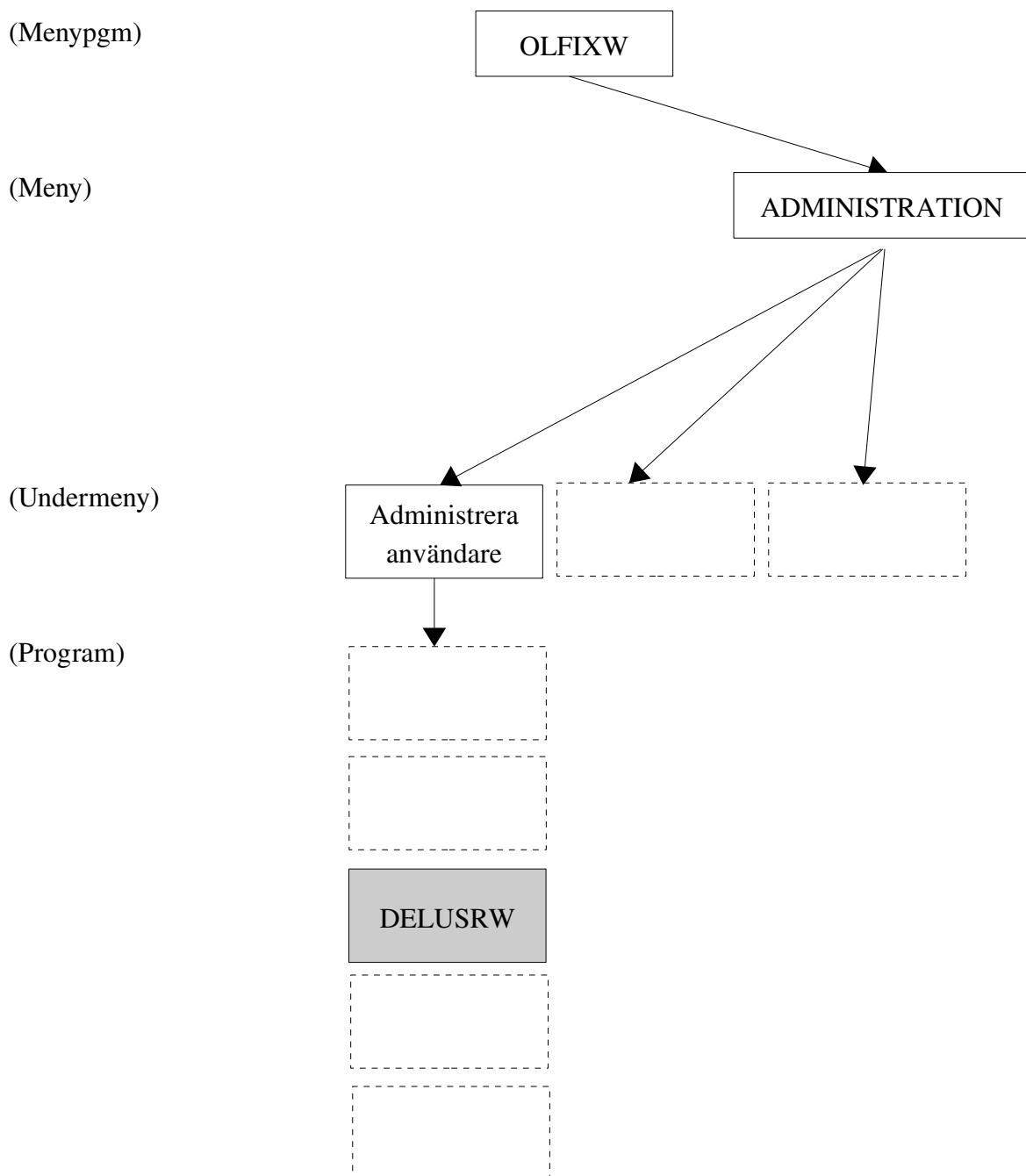
För att kunna köra DELXTW behövs behörighet till  
PRGLST  
TXTDEL  
TXTDSP

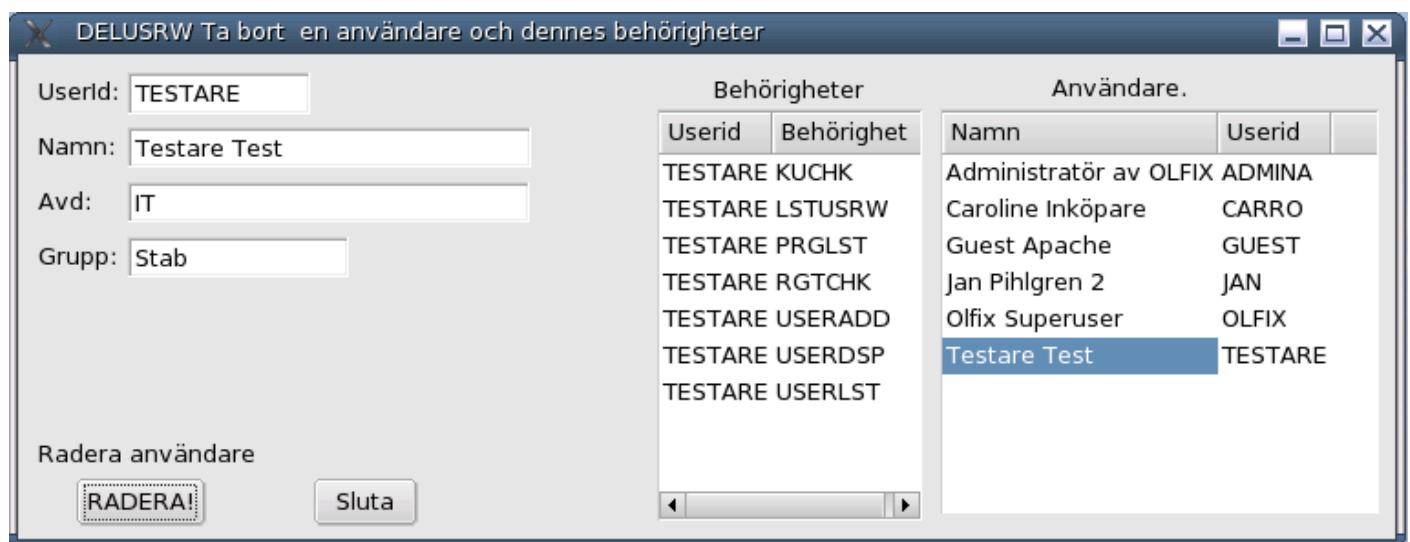
## **DELUSRW.....Radera användare**

DELUSRW, ett grafiskt program för att ta bort en användare och vederbörandes behörigheter från OLFIX. För att använda programmet krävs behörighet till funktionerna RGTDSP, USERDSP, USERLST, RGTDEL och USERDEL.

Programmet plockar upp userid från environment.

**VARNING!** Tag inte bort användaren OLFIX, då slutar programmet OLFIXW att fungera.





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.latin1());           // user OLFIX
process->addArgument( "USERDSP" );           // OLFIX program
process->addArgument( Userid.latin1() );
```

och

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

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.latin1());
process->addArgument( "RGTDSP" );           // OLFIX
program
process->addArgument( Userid.latin1() );
```

och

```
const char *userp = getenv("USER");
QString usr(userp);
process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.latin1());
process->addArgument( "RGTDEL" );           // OLFIX program
process->addArgument( anVID.latin1() );
process->addArgument( fncID.latin1() );
```

och

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

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.latin1());
process->addArgument( "USERDEL" );           // OLFIX program
process->addArgument( anVID.latin1() );
```

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

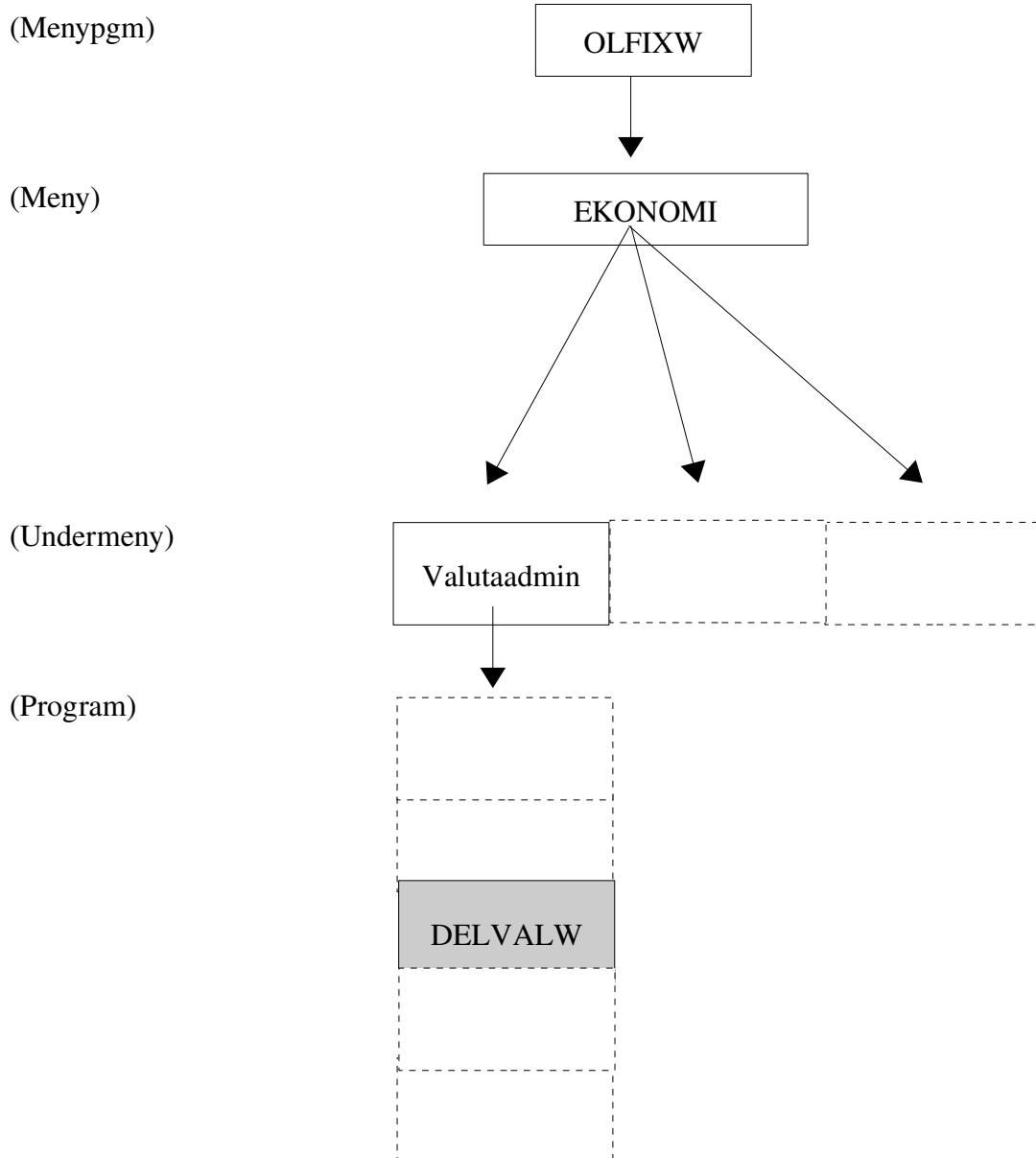
USERDEL

USERDSP

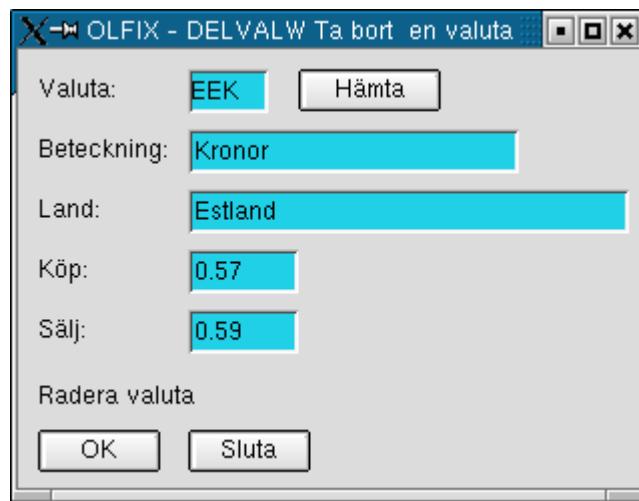
USERLST

## **DELVALW.....Radera valuta**

CHGVALW, ett grafiskt program för att ändra info för en valuta.  
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 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.

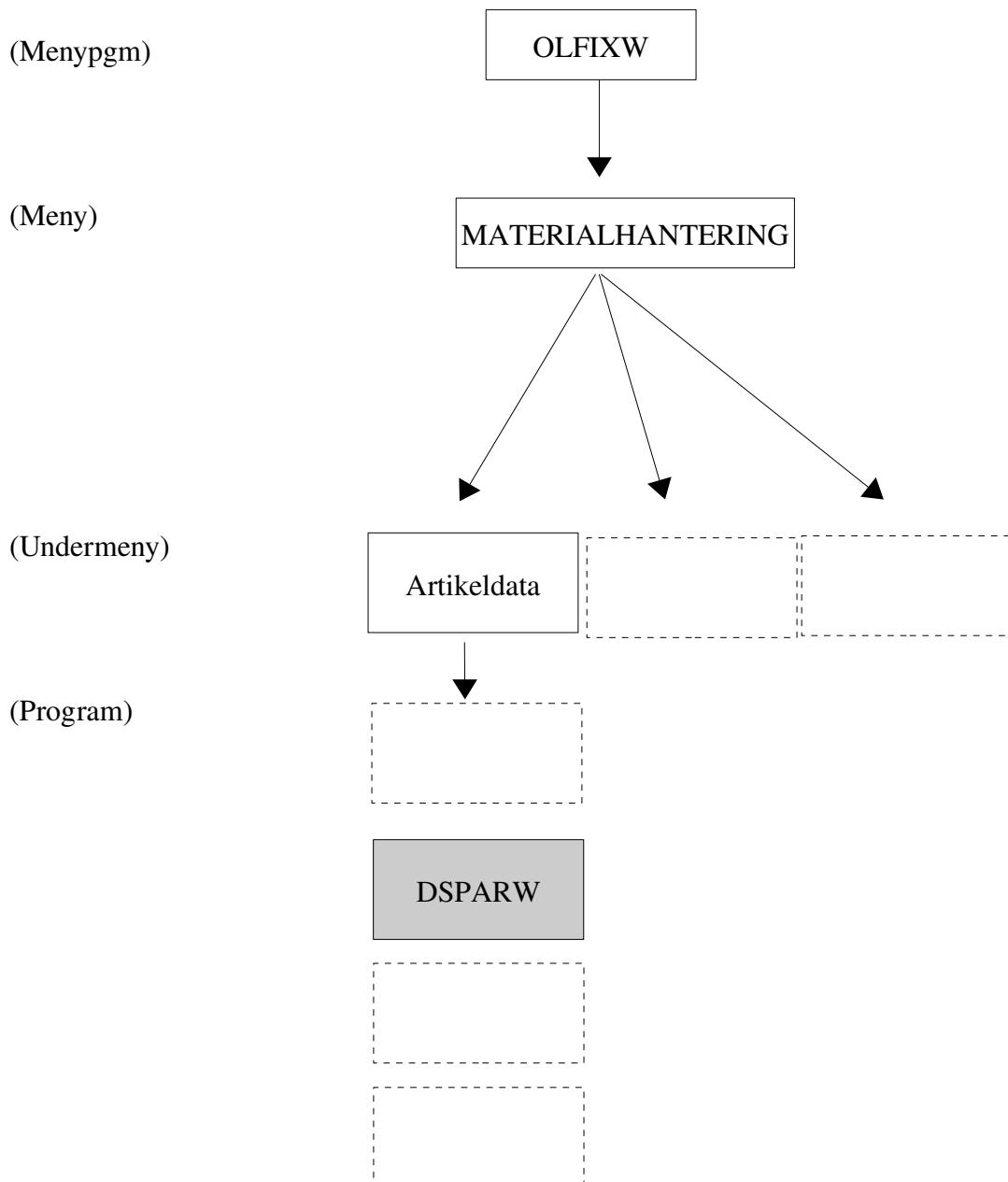
### Behörighetskrav:

För att kunna köra DELVALW behövs behörighet till  
PRGLST  
VALDSP  
VALDEL



## DSPARW.....Visa artikel, grunddata

DSPARW, ett grafiskt program för att visa grunddata för en artikel.  
Programmet plockar upp userid från environment.





X-DSPARW - Visa grunddata för en artikel

Artikelnummer: .....	1000-1001		
Benämning 1: .....	Att använda GNU/LINUX		
Benämning 2: .....	Linus Walleij		
Enhet: .....	ST	Omräkningsfaktor: 0	
Ledtid: .....	7	Nettovikt: ..... 0.00 kg	Volym: 0.000 m <sup>3</sup>
Artikeltyp: .....	2	Produktklass: 1000	Produktkonto: .....
Leverantör 1: .....	100	Struktur: .....	Tulltaxenr: .....
Leverantör 2: .....	200		
Leverantör 3: .....	300		
Ursprungsbenämning: .....	Att använda GNU/LINUX		
Ursprungsland: .....	Sverige		
Lev. artikelnummer: ...	ISBN 91-44-03400-8		
Lagerplatsdata			
Lagerplats: .....	1		
Lagerhylla: .....	ABC002		
Lagersaldo: .....	103.00		
Inventeringsgrupp: 001	ABC-kod: ..... A		
Senaste inköpskvantitet: .....			
Näst senaste inköpskvant: .....			
Näst,näst senaste ink.kvant: .....			
Beställd kvantitet: .....	62.00	Beställningspunkt: 125.00	
Reserverad kvantitet: .....	10.00		

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 ARDSP och ARDSPL.

DSPARW anropar ARDSP och ARDSPL via STYRMAN.

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

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

och

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

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

Detta blir:

./STYRMAN usr ARDSP artikelnr

och

./STYRMAN usr ARDSPL lagerstalle artikelnr

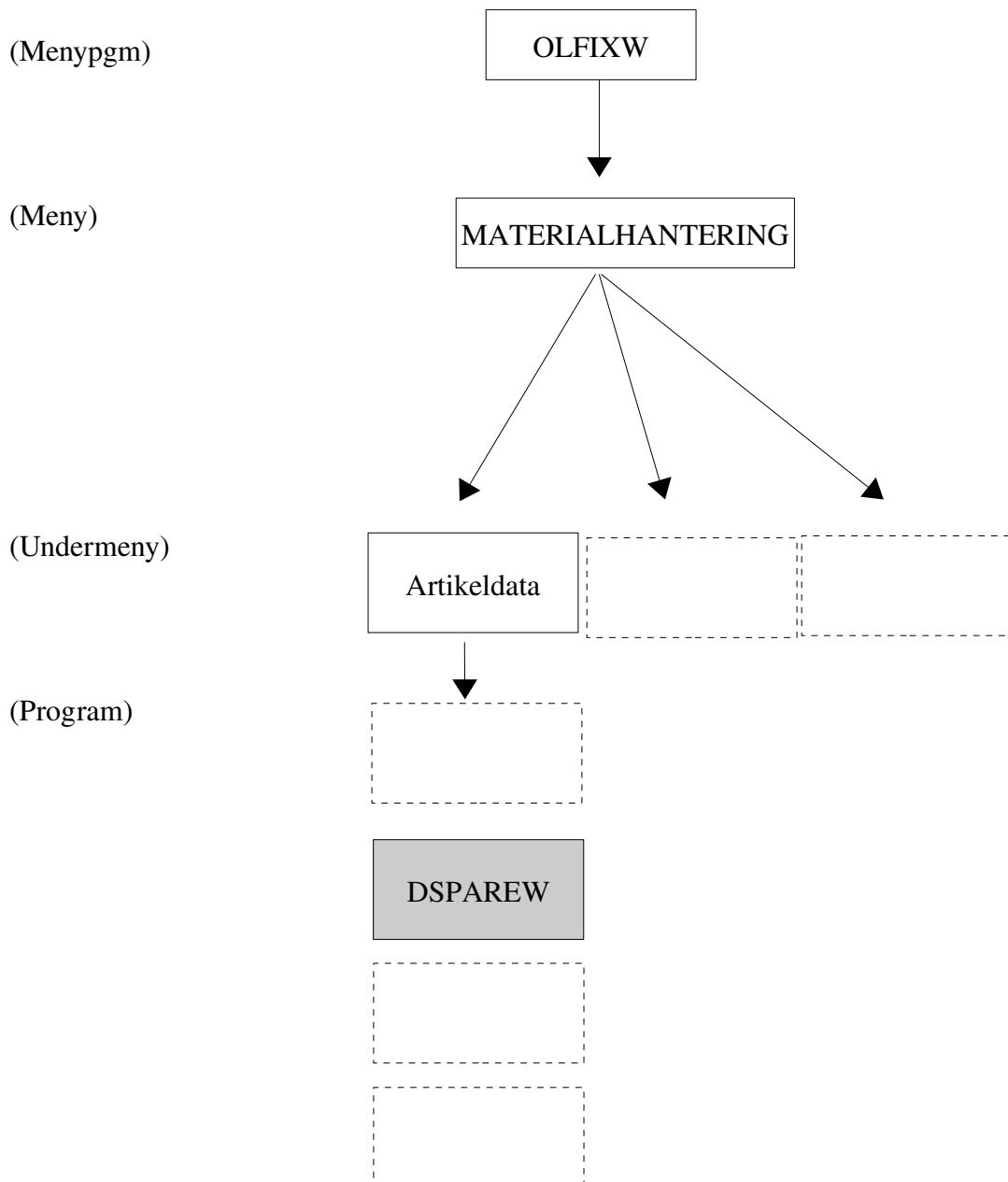
usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra DSPARW behövs behörighet till  
PRGLST  
ARDSP  
ARDSPL

## DSPAREW.....Visa artikel, ekonomidata

DSPARW, ett grafiskt program för att visa ekonomidata för en artikel.  
Programmet plockar upp userid från environment.





X-> DSPAREW - Visa ekonomiska data för en artikel

Artikelnummer: .....	1000-1001	
Benämning 1: .....	Att använda GNU/LINUX	
Benämning 2: .....	Linus Walleij	
Enhet: .....	ST	Omräkningsfaktor: 1
Ledtid: .....	7	
Artikeltyp: .....	2	
Leverantör 1: .....	100	
Leverantör 2: .....	200	
Leverantör 3: .....	300	
Ursprungsbenämning:	Att använda GNU/LINUX	
Ursprungsland:	Sverige	
Lev. artikelnummer:	ISBN 91-44-03400-8	

— Ekonomidata —

Lagerplats: .....	1	Försäljningspris: .....	117.00
Lagersaldo: .....	103.00	Gällande kalkylpris: ...	160.00
Omkostnader: .....	34.50	Planerat kalkylpris: ....	202.50
Senaste inköpspris: .....	125.50	Fryst kalkylpris: .....	160.00
Senaste inköpskvantitet: .....	60.00	Beställningspunkt: .....	125.00
Näst senaste inköpskvant: ...	75.00		
Näst,näst senaste ink.kvant: .	50.00		
Beställd kvantitet: .....	62.00		
Reserverad kvantitet: .....	10.00		

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 ARDSP och ARDSPL.

DSPAREW anropar ARDSP och ARDSPL via STYRMAN.

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

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

och

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

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

Detta blir:

./STYRMAN usr ARDSP artikelnr

och

./STYRMAN usr ARDSPL lagerställe artikelnr

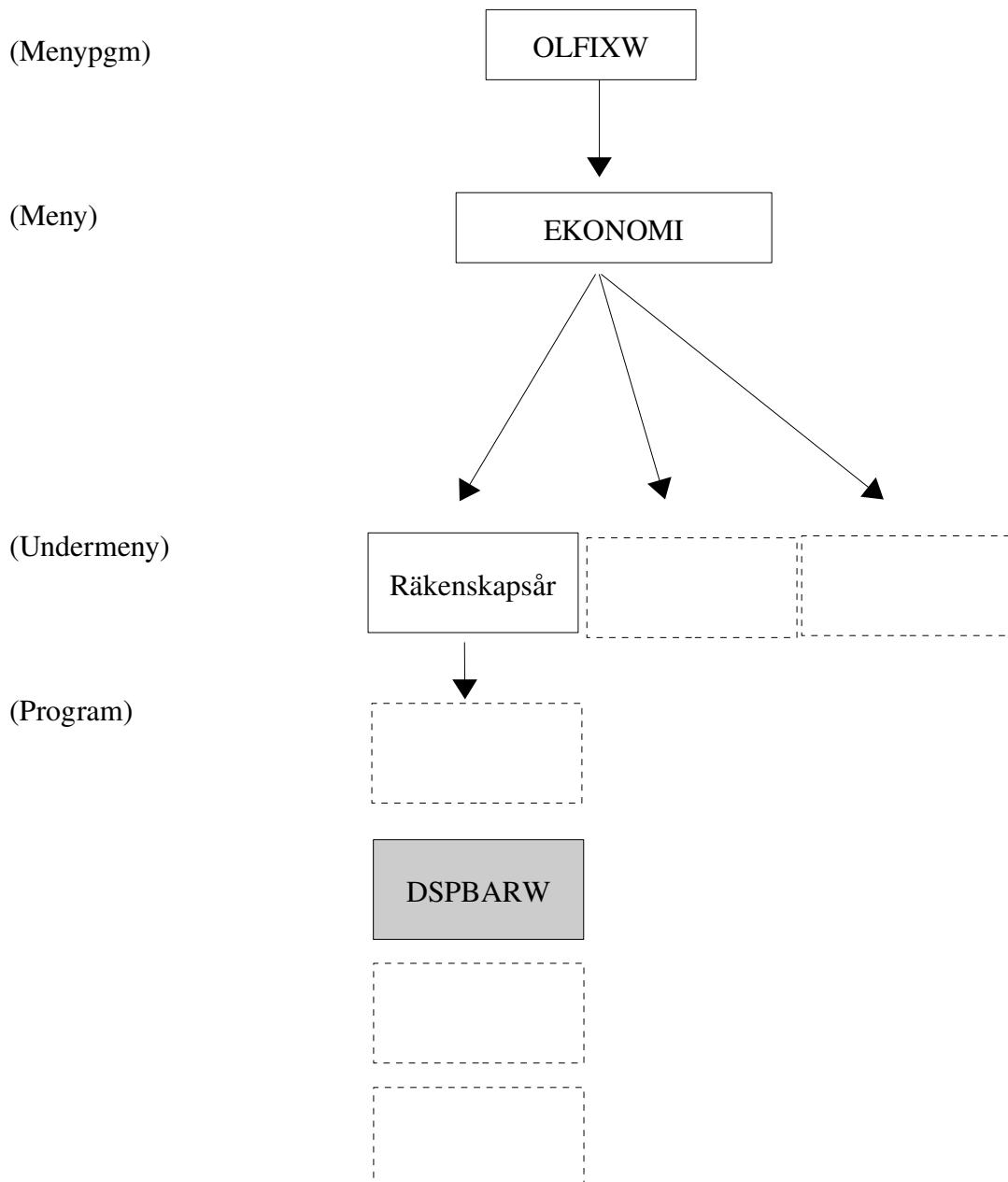
usr är den inloggades userid.

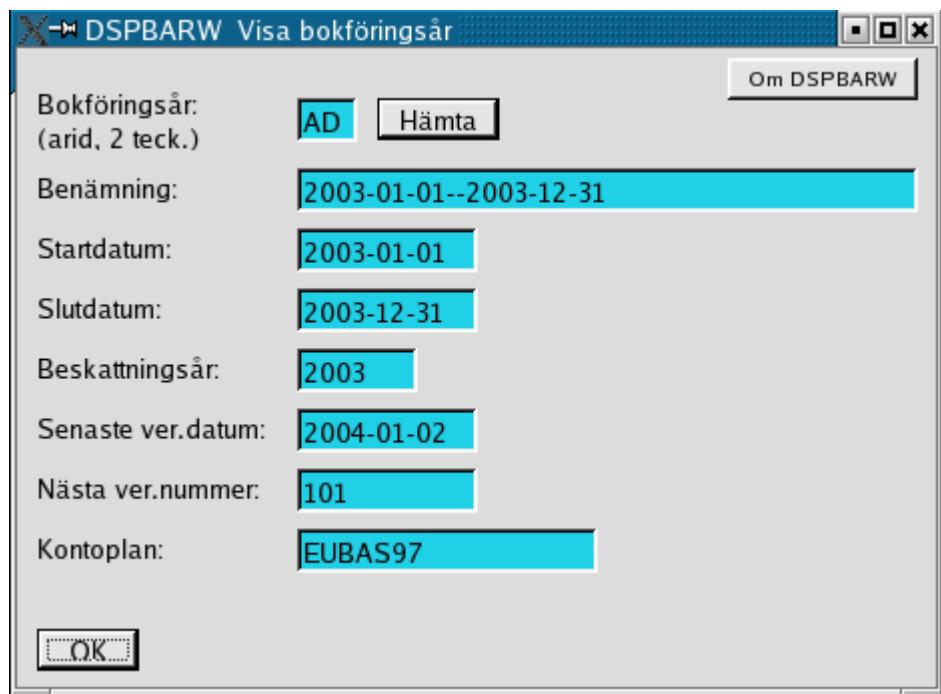
### Behörighetskrav:

För att kunna köra DSPAREW behövs behörighet till  
PRGLST  
ARDSP  
ARDSPL

## DSPBARW.....Visa bokföringsår

DSPBARW, ett grafiskt program för att visa informationen för ett räkenskapsår.  
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 BARDSP.

DELBARW anropar BARDSP via STYRMAN.

```
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);                 // bokföringsårID
```

Detta blir:

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

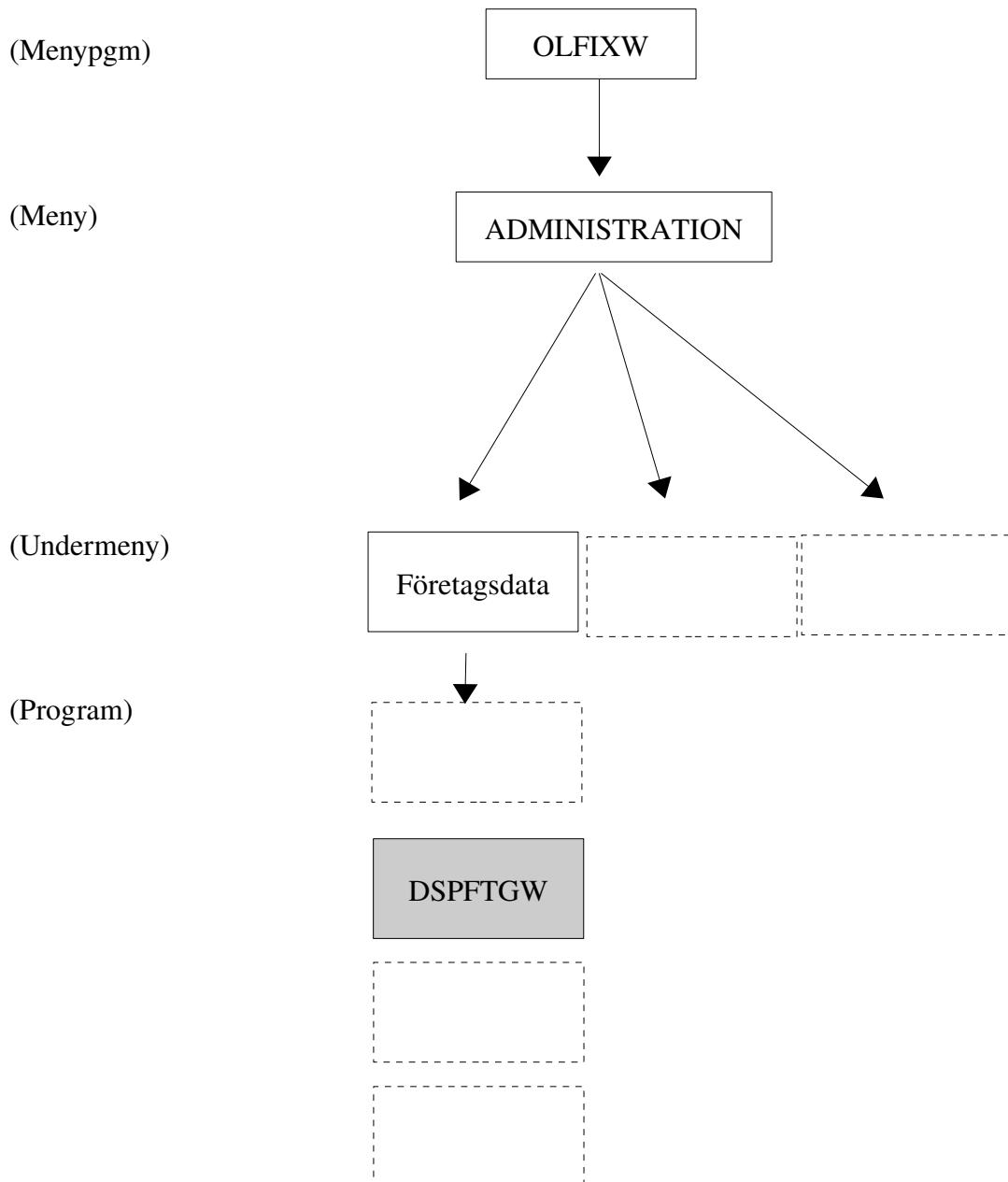
usr är den inloggades userid.

#### **Behörighetskrav:**

För att kunna köra DSPBARW behövs behörighet till  
PRGLST  
BARDSP

## DSPFTGW....Visa en företagsdata

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



X DSPFTGW - Visa företagsdata

Företagsnamn:	OLFIX AB	Organisationsnr:	991199-1991						
		Branschkod:	82301						
Postadress:	Box 70	Postnummer:	199 98						
Besöksadress:	Syntaxvägen 99	Postnummer:	199 98						
Godsadress:	Verktygsgatan 11	Postnummer:	199 97						
Telefonnummer:	09-199300	Mobiltelefon:	070-98765411	Telefax:	09-199397				
e-mailadress:	jan@pihlgren.se			Telex:	11225				
Momssats 1:	25	Momssats 2:	12	Momssats 3:	6	Momssats 4:		Momssats 5:	
Momskonto, ingående moms:	.....	2641	Momskonto, utgående moms:	.....	2611				
Automatkontering J/N : N									
Kontonr kundfordringar:	1920	Kontonr inbetalning:	1511						
Antal fakturnrserier:	2	Senaste fakturarnr:	1341	Senaste fakturarnr:	100070				
		Senaste inköpsordernr:	28	Serie 2					
Senaste kundnr:	4383	Senaste kundordernr:	63						

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 FTGDSP.

DSPINKW anropar FTGDSP 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( "FTGDSP" );               // OLFIX funktion
process->addArgument( posttyp );
```

Detta blir:

```
./STYRMAN usr FTGDSP posttyp
```

usr är den inloggades userid.

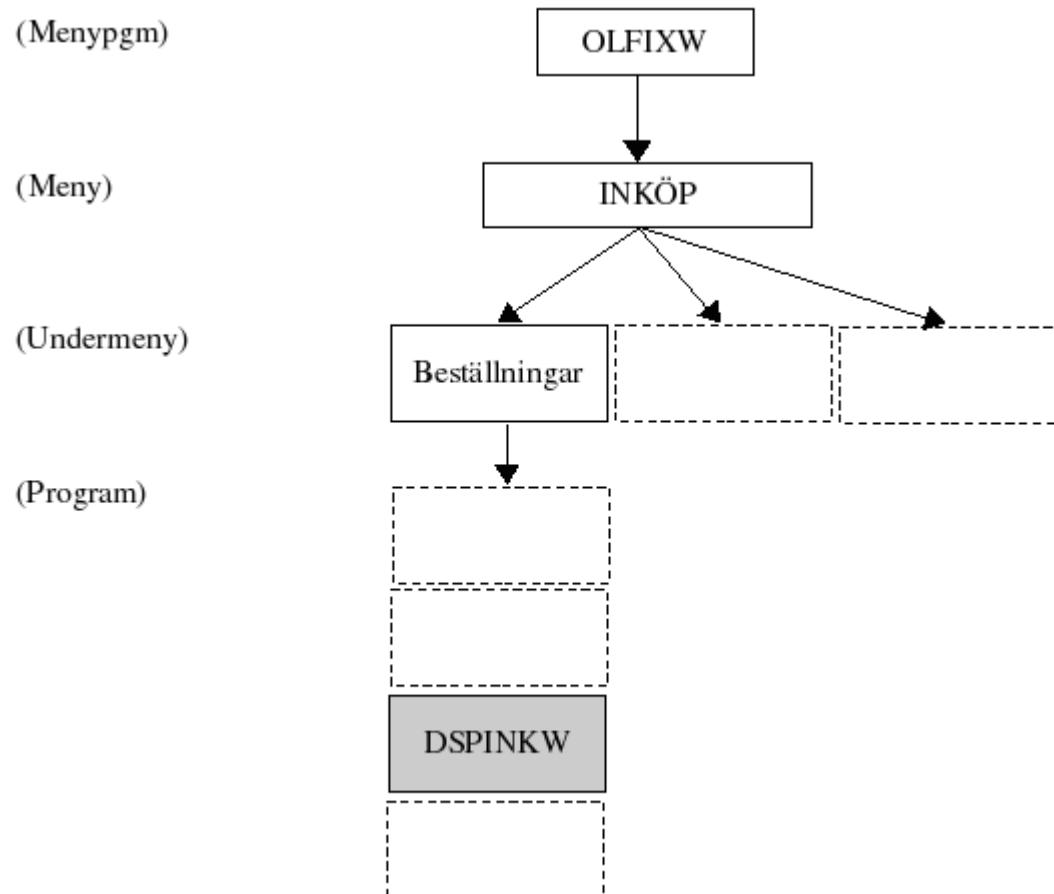
Programmet FTGDSP anropas engång för varje posttyp,  
ADR1, ADR2, ADR3, ADR4, ADR5, ADR6, ADR7, ADR8, ADR9,  
AUTOK, EML1, FAKNR, FKNR2, FKNRS, FNAMN, FTGNR,  
INKNR, INKTO, KORNR, KFKTO  
MOMS1, MOMS2, MOMS3, MOMS4, MOMS5, MOMSI, MOMSU,  
SKUNR, SNIKD, TELEX, TFAX, TFN1, TFNMB.

#### Behörighetskrav:

För att kunna köra DSPFTGW behövs behörighet till  
PRGLST  
FTGDSP

## DSPINKW....Visa en inköpsorder

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



**DSPINKW - Visa en inköpsorder**

Leverantörsnr: ..... 9999	* = Obligatoriskt.																																																																
Beställningsnr: ... * 18	Beställningsdatum: ... 2003-12-24																																																																
<table border="1"> <tr> <td>Leverantörens postadress Namn: ..... Testleverantör AB</td> <td>Orderhuvud Adress: ..... Delivery Street 1C</td> <td>Mottagarens Leveransadress/Godsadress PROGRAM AB Verktygsgatan 11 Postnummer: ..... 199 99 Postadress: ..... LEVSTAD Land: ..... Sverige Godsmärke: ..... P-order Vår referent: ... Jan Pihlgren Leveransdatum: .... 2004-01-10 Best.typ: <input checked="" type="checkbox"/> N Värt kundnr: ... 98765 Kommentar: ..... Kommentar</td> <td>Betalningsvillkor. 30 dagar netto Leveransvillkor. EXW Leveranssätt. ASG kundnr:99901111 Eftertext. Ordererkänndande önskas inom 3 arbetsdagar (om ej redan bekräftats) Ange alltid vårt artikelnummer på följesedel och faktura.</td> </tr> </table>		Leverantörens postadress Namn: ..... Testleverantör AB	Orderhuvud Adress: ..... Delivery Street 1C	Mottagarens Leveransadress/Godsadress PROGRAM AB Verktygsgatan 11 Postnummer: ..... 199 99 Postadress: ..... LEVSTAD Land: ..... Sverige Godsmärke: ..... P-order Vår referent: ... Jan Pihlgren Leveransdatum: .... 2004-01-10 Best.typ: <input checked="" type="checkbox"/> N Värt kundnr: ... 98765 Kommentar: ..... Kommentar	Betalningsvillkor. 30 dagar netto Leveransvillkor. EXW Leveranssätt. ASG kundnr:99901111 Eftertext. Ordererkänndande önskas inom 3 arbetsdagar (om ej redan bekräftats) Ange alltid vårt artikelnummer på följesedel och faktura.																																																												
Leverantörens postadress Namn: ..... Testleverantör AB	Orderhuvud Adress: ..... Delivery Street 1C	Mottagarens Leveransadress/Godsadress PROGRAM AB Verktygsgatan 11 Postnummer: ..... 199 99 Postadress: ..... LEVSTAD Land: ..... Sverige Godsmärke: ..... P-order Vår referent: ... Jan Pihlgren Leveransdatum: .... 2004-01-10 Best.typ: <input checked="" type="checkbox"/> N Värt kundnr: ... 98765 Kommentar: ..... Kommentar	Betalningsvillkor. 30 dagar netto Leveransvillkor. EXW Leveranssätt. ASG kundnr:99901111 Eftertext. Ordererkänndande önskas inom 3 arbetsdagar (om ej redan bekräftats) Ange alltid vårt artikelnummer på följesedel och faktura.																																																														
<p style="text-align: center;"><b>Detaljrader</b></p> <table border="1"> <thead> <tr> <th>Radnr</th> <th>Artikelnr</th> <th>Benämning</th> <th>Lev.vecka</th> <th>Beställt Antal</th> <th>Lev.Antal</th> <th>Resterande ant.</th> <th>Pris</th> <th>Summa</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>1173-0911</td> <td>Spänningssregulator positiv</td> <td>402</td> <td>10.00</td> <td>0.00</td> <td>10.00</td> <td>1.50</td> <td>15.00</td> </tr> <tr> <td>20</td> <td>1173-0963</td> <td>Spänningssregulator negativ</td> <td>402</td> <td>10.00</td> <td>10.00</td> <td>0.00</td> <td>5.50</td> <td>0.00</td> </tr> <tr> <td>30</td> <td>1173-1175</td> <td>Spänningssregulator positiv</td> <td>402</td> <td>10.00</td> <td>0.00</td> <td>10.00</td> <td>30.00</td> <td>300.00</td> </tr> <tr> <td>40</td> <td>1173-1445</td> <td>D/A Omvandlare 12-bit</td> <td>402</td> <td>10.00</td> <td>0.00</td> <td>10.00</td> <td>95.00</td> <td>950.00</td> </tr> <tr> <td>50</td> <td>1173-1447</td> <td>Timerkrets</td> <td>402</td> <td>10.00</td> <td>0.00</td> <td>10.00</td> <td>5.45</td> <td>54.50</td> </tr> <tr> <td>60</td> <td>1173-6910</td> <td>Quadruple 2input NAND gate</td> <td>402</td> <td>10.00</td> <td>0.00</td> <td>10.00</td> <td>1.45</td> <td>14.50</td> </tr> </tbody> </table>			Radnr	Artikelnr	Benämning	Lev.vecka	Beställt Antal	Lev.Antal	Resterande ant.	Pris	Summa	10	1173-0911	Spänningssregulator positiv	402	10.00	0.00	10.00	1.50	15.00	20	1173-0963	Spänningssregulator negativ	402	10.00	10.00	0.00	5.50	0.00	30	1173-1175	Spänningssregulator 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
Radnr	Artikelnr	Benämning	Lev.vecka	Beställt Antal	Lev.Antal	Resterande ant.	Pris	Summa																																																									
10	1173-0911	Spänningssregulator positiv	402	10.00	0.00	10.00	1.50	15.00																																																									
20	1173-0963	Spänningssregulator negativ	402	10.00	10.00	0.00	5.50	0.00																																																									
30	1173-1175	Spänningssregulator 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																																																									
<input type="button" value="Stäng"/> Orderstatus <input checked="" type="checkbox"/> N Order bekräftad <input checked="" type="checkbox"/> E		Ursprunglig ordertotal <input type="text" value="1389.00"/>	Beställning Total <input type="text" value="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 INKHDSR inkordernr

och

./STYRMAN usr INKRLST inkordernr

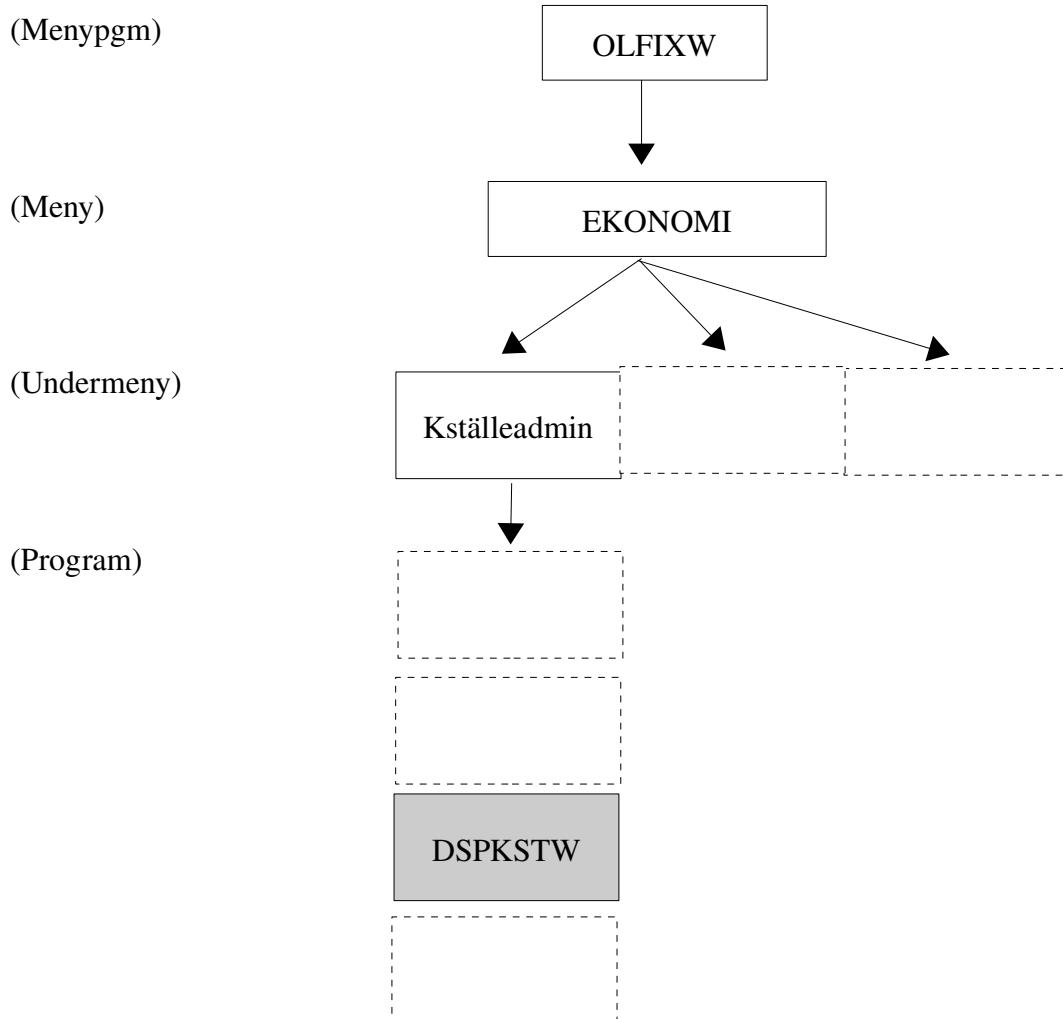
usr är den inloggades userid.

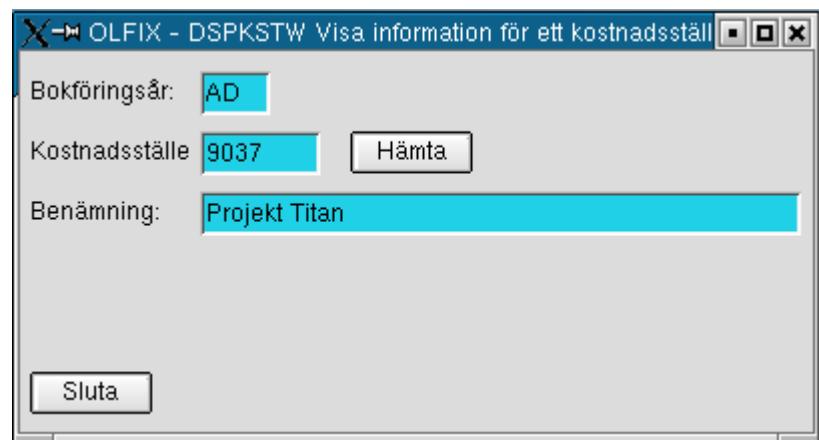
### Behörighetskrav:

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.





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.

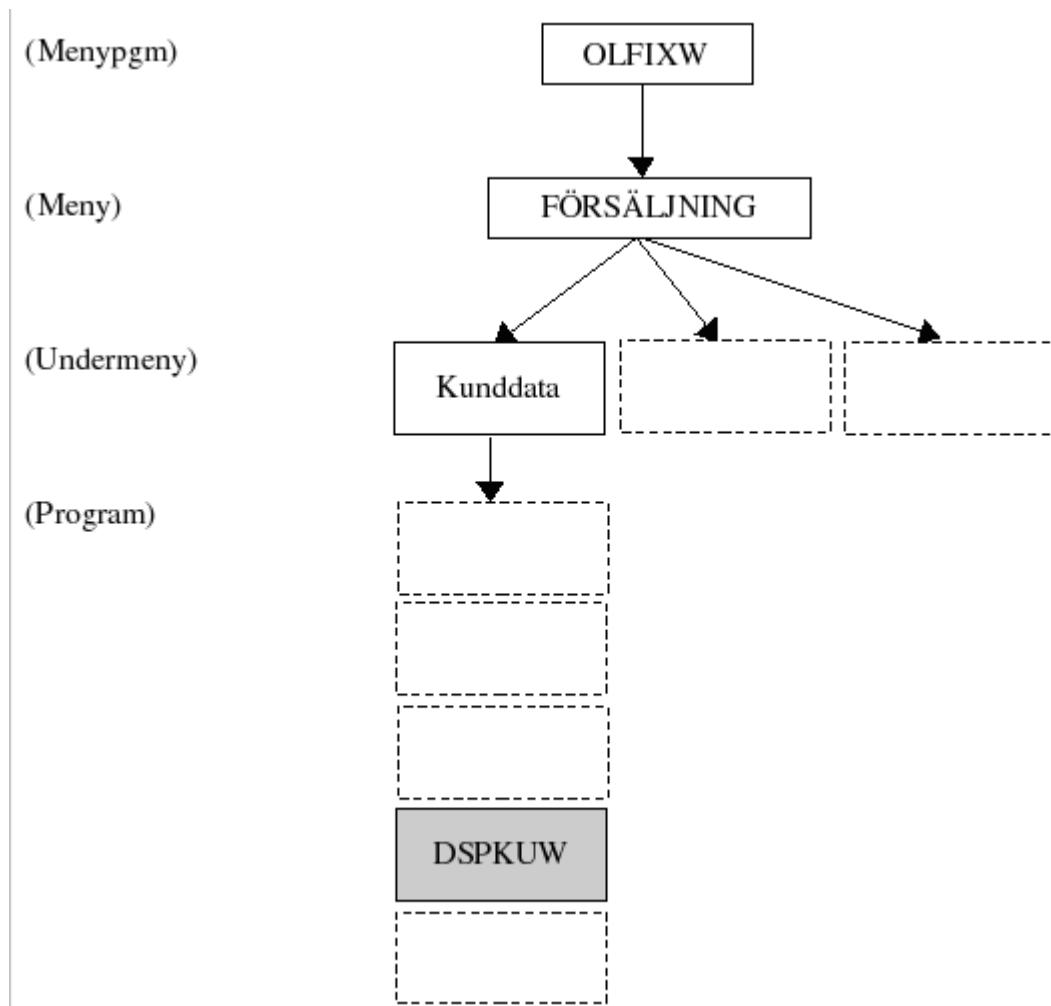
### **Behörighetskrav:**

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.



X DSPKUW - Visa kunddata.

Hjälp

KundID: .....	4380	Organisationsnr: .....	559999-9999		
Kundnamn: .....	Testkund AB				
Kundadress: .....	Testvägen 34				
Postnummer: .....	199 89	Postadress: .....	LYXSTAD		
Land: .....	Sverige				
Telefonnummer: ...	09-999980				
Faxnummer: .....	09-999989				
E-mail: .....	info@testkund.se				
Er Referent: .....	Tessie Testdottir				
Er Ref's telefonnr: ..	09-999971				
Er Ref's e-mailadr: ..	tessi.t@testkund.se				
Vår säljare: .....	Caroline Seljare				
Distrikt: .....	21	Kundkategori: .....	10	Prislista: .....	1
Leveransplats: ....	001	Leveransvillkor: ....	002	Leveranssätt: ....	003
Betalningsvillkor:	004				
Valuta: .....	SEK	Språkkod: .....	sv		
Ordererkännande: .	J	Plocklista :	J	Följesedel: .....	J
Expeditionsavgift: .	J	Fraktavgift: .....	J	Kravbrev: .....	J
Kreditlimit: .....	25000.00	Kreditdagar	(null)	Kreditkod: .....	JN
Exportkod: .....	(null)	Skattekod: .....	(null)	Rabattkod: .....	(null)
Dröjsmålsränta: ....	J	Dröjsmålsfaktura: ..	J	Samlingsfaktura: J	
Senaste kravdatum	(null)	Skuld: .....	(null)		
Orderstock: .....	(null)				
Fri text (100 tkn): ..	Ingen text				

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
```

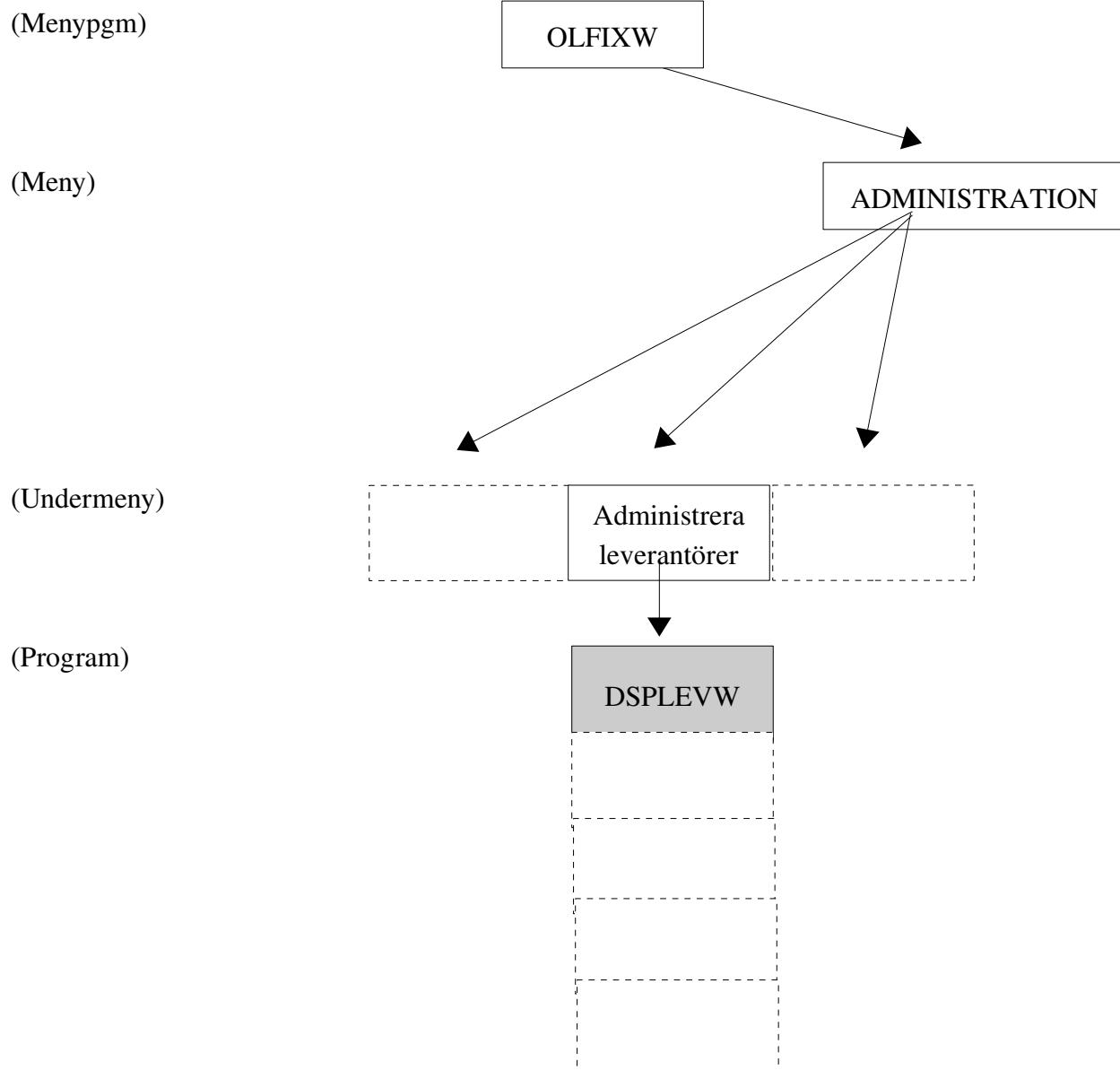
usr är den inloggades userid.

#### **Behörighetskrav:**

För att kunna köra DSPKUW behövs behörighet till  
PRGLST  
KUDSP  
OLFIXHLP

## DSPLEVW.....Visa leverantörsdata

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





X DSPEVW - 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:

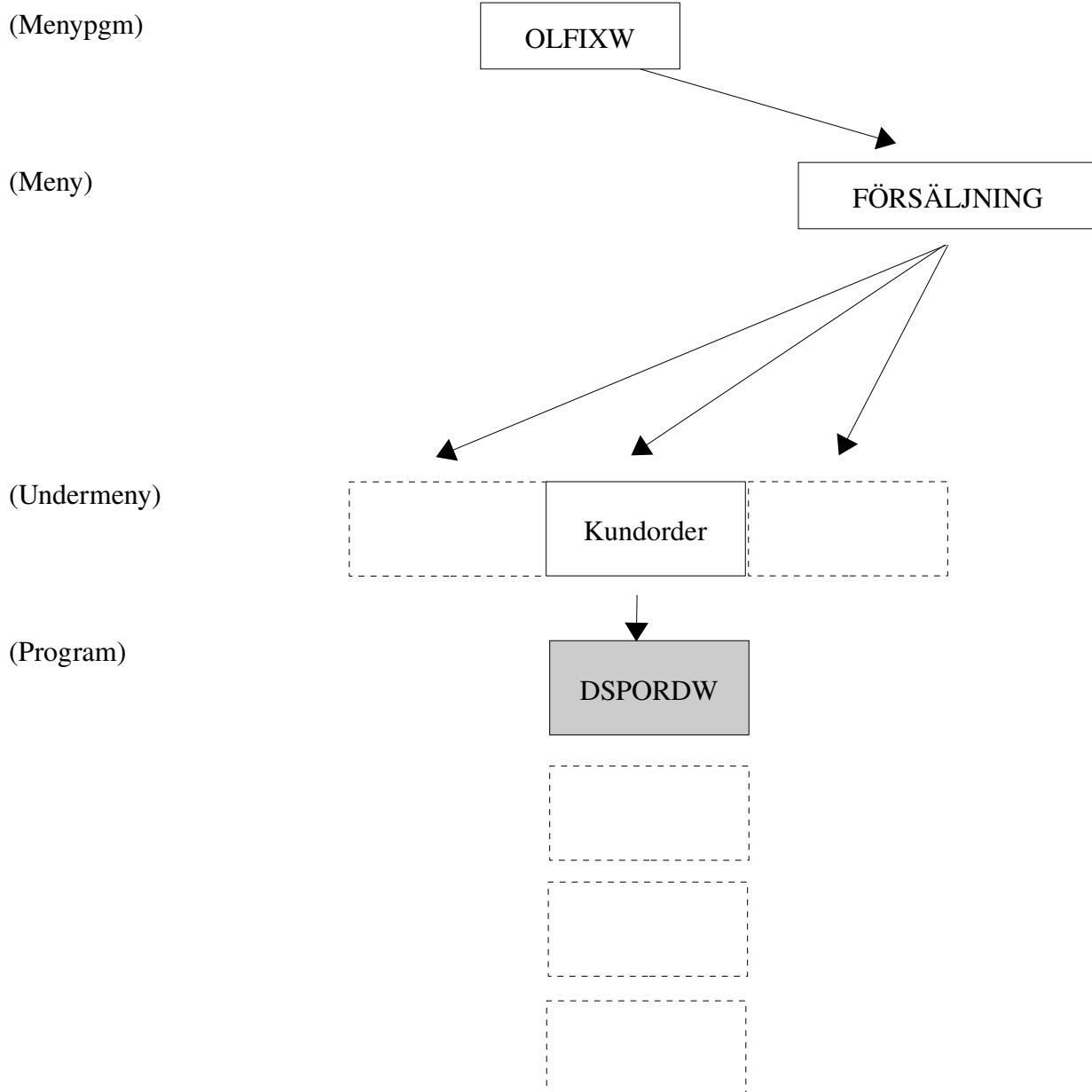
För att kunna köra DSPLEVW behövs behörighet till

PRGLST

LEVDSP

## DSPORDW.....Visa en kundorder

DSPORDW är ett grafiskt program för att visa data på en kundorder.  
Programmet plockar upp userid från environment.





**DSPORDW - Visa kundorder**

Ordernummer: .....	33	Datum	Hjälp					
Kundnummer: .....	4377	2005-03-14						
<b>Postadress, Fakturaadress</b>		<b>Leveransadress</b>						
Namn: .....	Nya Kund AB	Ref: Per Karlsson	Leveransplats ..... 002					
Adress: .....	Provgatan 23	Testgatan 21	Leveransvillkor ..... 001					
Postnummer: .....	199 97	199 91	Betalningsvillkor: .... 1					
Postadress: .....	LILLEBY	LILLEBY	Valuta: ..... SEK					
Land: .....	Sverige	Sverige	Moms: .....% 25.00					
Säljare .....	Josef Seljare	Önskad leveranstid	2005-03-14					
Godsmärke: .....	godsmärke							
Orderstatus: .....	A							
Radnr	Artikelnr	Benämning	Leveransvecka	Antal	Pris/st	Summa	Moms	
10	1000-1015	Seagate Baracuda 72008250GB	5111	4.00	1250.00	5000.00	1250.00	
20	1000-1016	Seagate Baracuda 7200+7 160GB	5111	4.00	650.00	2600.00	650.00	
Summa		Frakt	Fraktmoms	Summa moms	Order Total			
7600.00		90.00	22.50	1922.50	9635.00	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 ORDDSP och ORDRDSP.

DSPORDW anropar ORDDSP och ORDRDSP via STYRMAN.

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

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

samt

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

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

Detta blir:

./STYRMAN usr ORDDSP ordernr

samt

./STYRMAN usr ORDRDSP ordernr

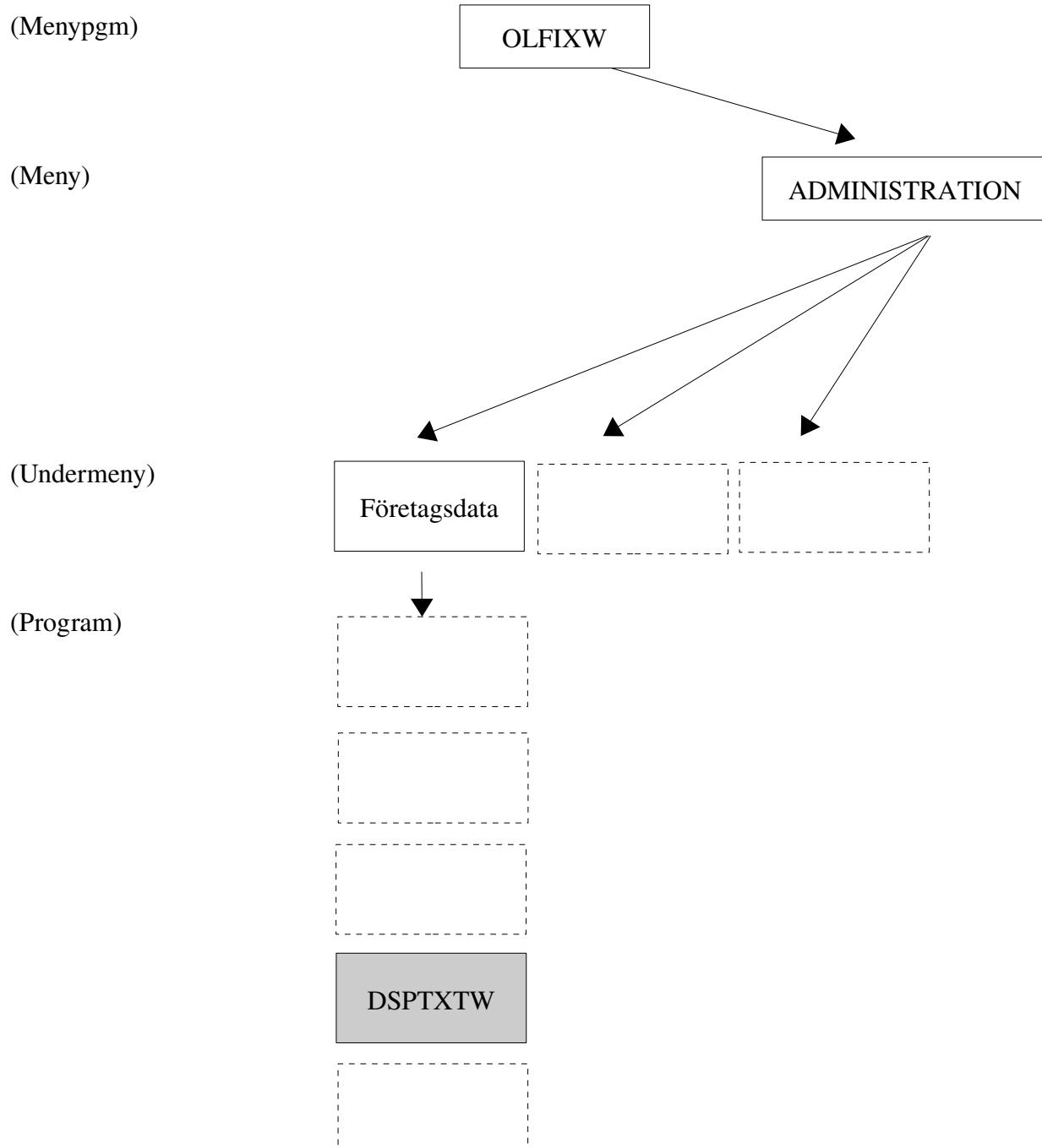
### Behörighetskrav:

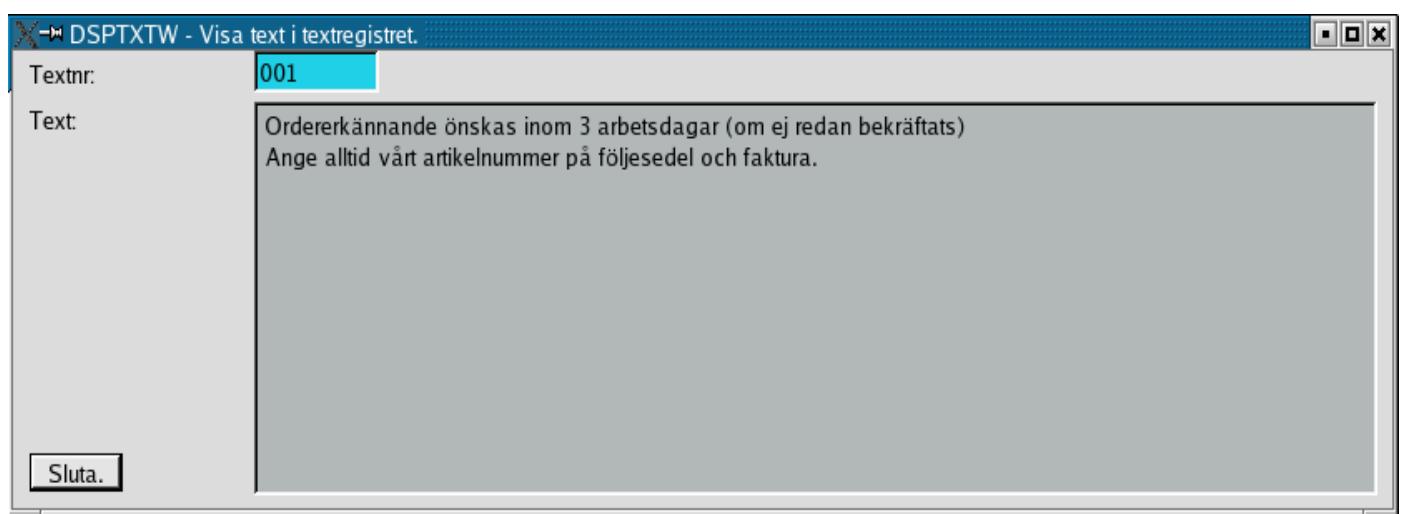
För att kunna köra DSPLEVV behövs behörighet till  
PRGLST  
ORDDSP  
ORDRDSP



## DSPTXTW.....Visa en post i TEXTREG

DSPTXTW, ett grafiskt program för att visa information av en post 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.

Programmet anropas från OLFIXW. OLFIXW kontrollerar att användaren har behörighet att använda funktionen TXTDSP. Programmet kan också anropas från LSTXTTW.

DSPTXTW anropar TXTDSP via STYRMAN.

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

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

Detta blir:

./STYRMAN usr TXTDSP txtnr

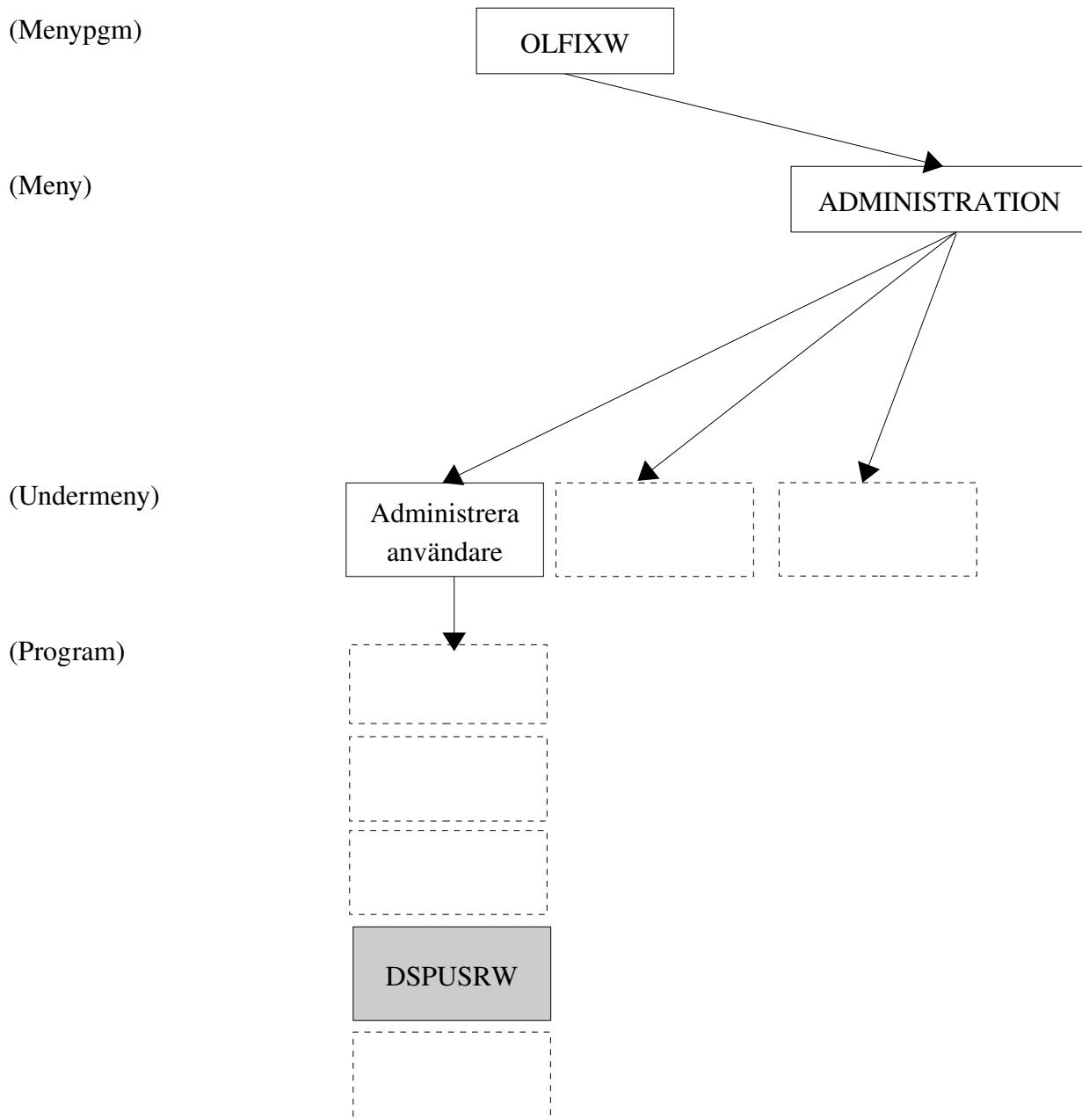
### Behörighetskrav:

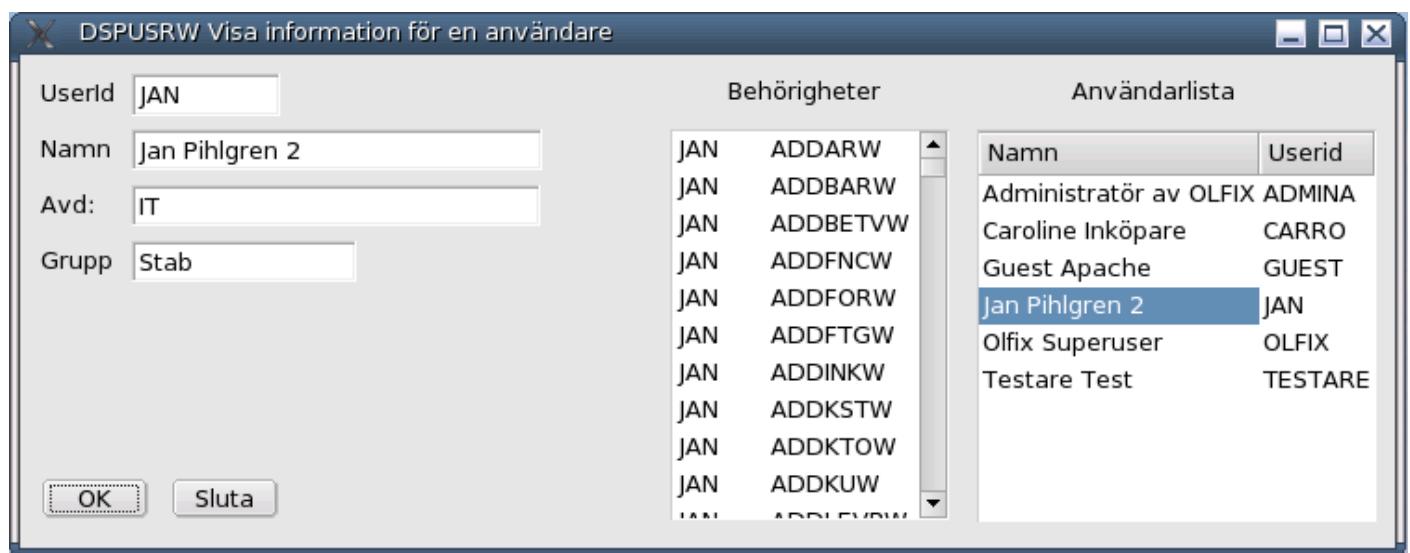
För att kunna köra DSPUSRW behövs behörighet till  
PRGLST  
TXTDSP

## DSPUSRW.....Visa användardata

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

För att använda programmet krävs behörighet till funktionerna RGTDSP, USERDSP LSTUSR.  
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 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.latin1());
process->addArgument( "USERDSP"); // user OLFIX
process->addArgument( Userid.latin1() ); // OLFIX program
```

och

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

process = new QProcess();
process->addArgument("./STYRMAN");
process->addArgument(usr.latin1());
process->addArgument( "RGTDSP"); // OLFIX program
process->addArgument( Userid.latin1() );
```

Detta blir:

./STYRMAN userid1 RGTDSP userid2

och

./STYRMAN userid1 USERDSP userid2

userid1 är den inloggades userid.

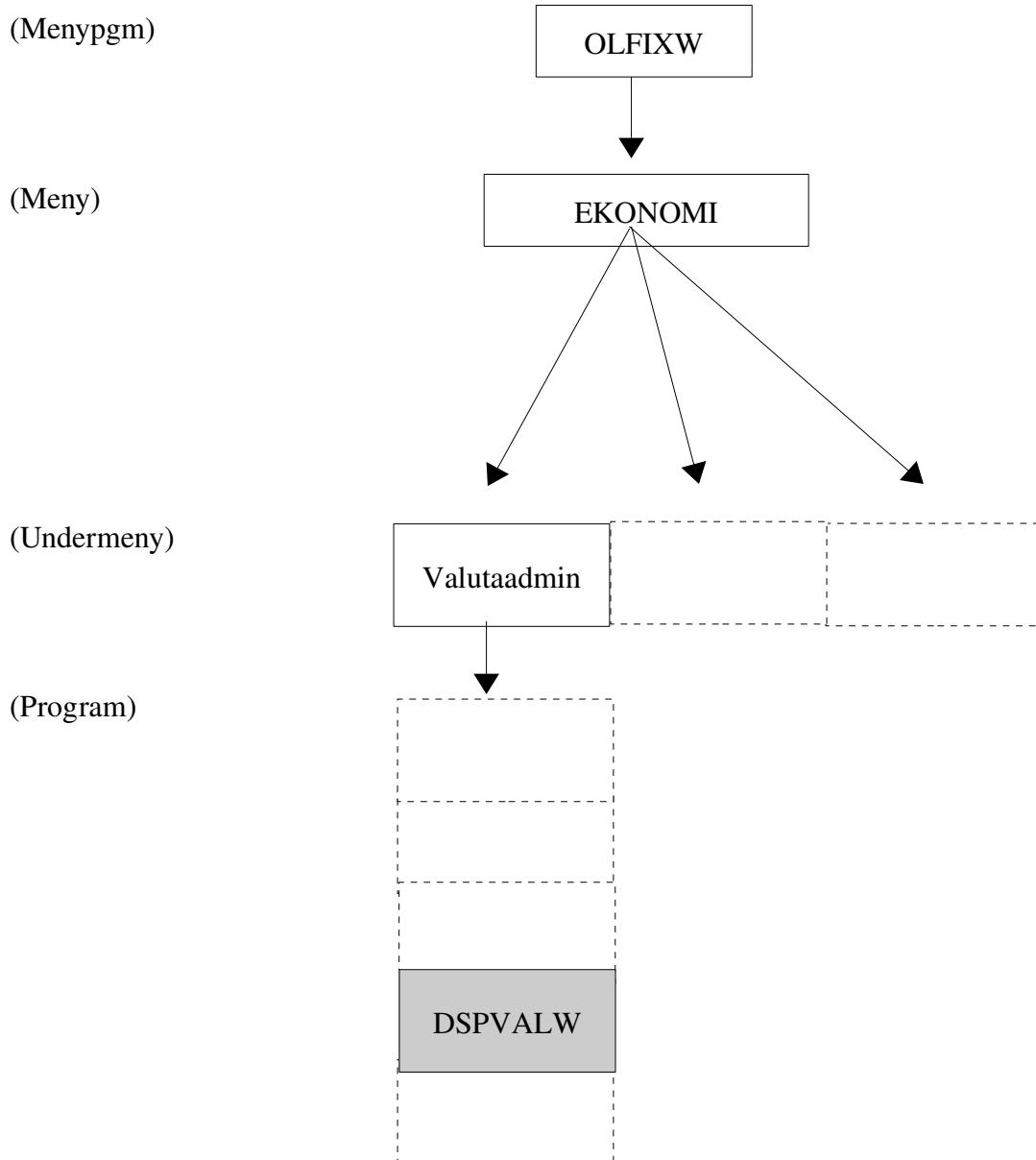
userid2 är userid på den användare som man önskar data om.

### Behörighetskrav:

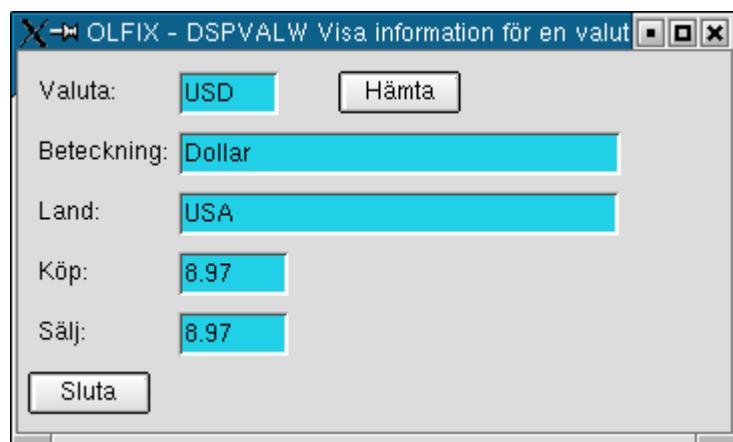
För att kunna köra DSPUSRW behövs behörighet till  
PRGLST  
USERDSP  
RGTDSP  
LSTUSR

## DSPVALW.....Visa valutainformation

DSPVALW, ett grafiskt program för att visa information om ett kostnadställe.  
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 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
```

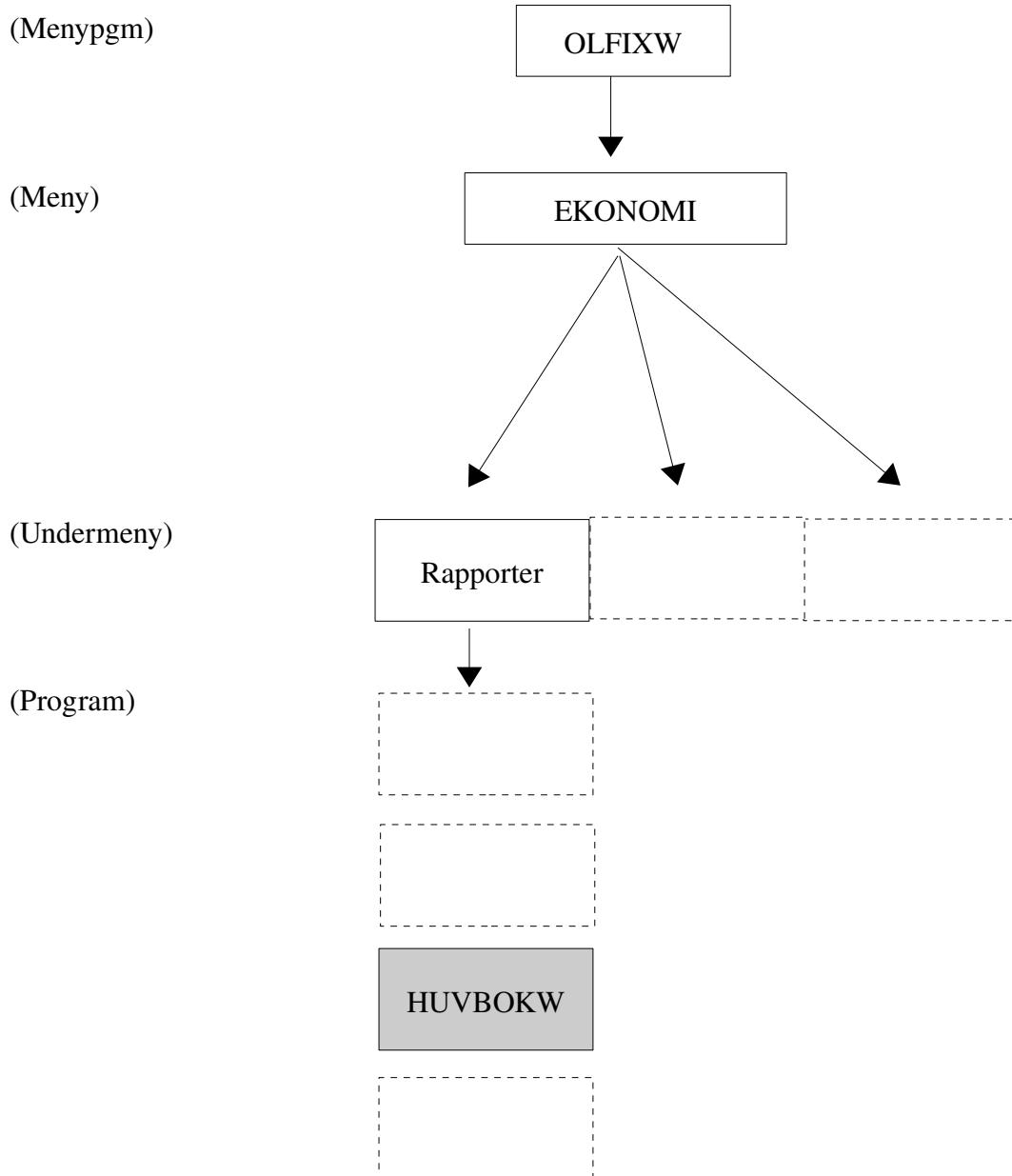
usr är den inloggades userid.

### **Behörighetskrav:**

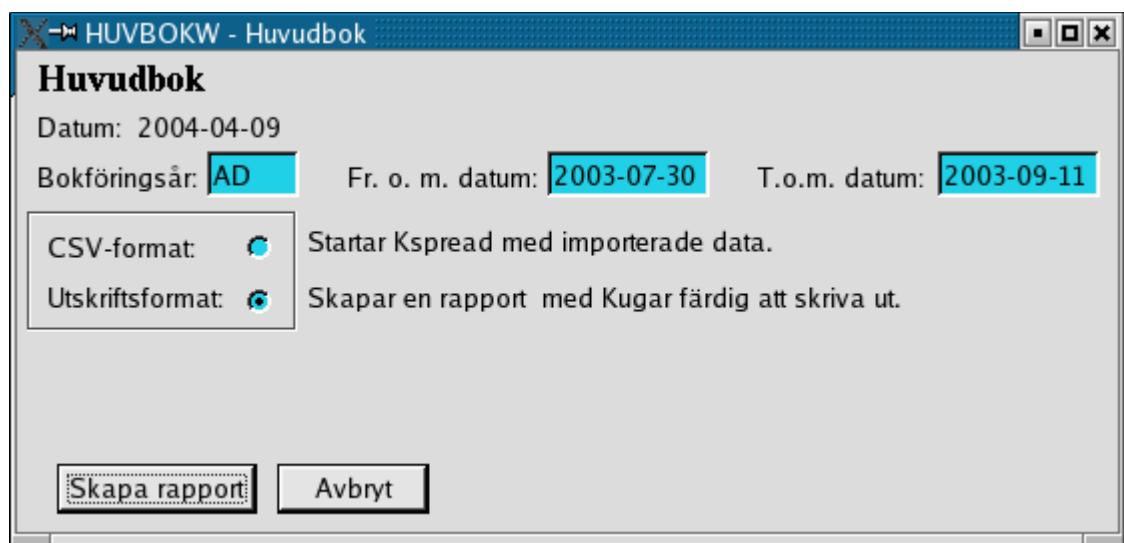
För att kunna köra DSPVALW behövs behörighet till  
PRGLST  
VALDSP

# HUVBOKW.....Huvudbok

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







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);
process->addArgument( "HBOKRPT" );
process->addArgument(bar);
process->addArgument(fromdatum);
process->addArgument(tomdatum);
```

// user OLFIX  
// OLFIX funktion  
// Bokföringsår  
// Från och med datum  
// 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");
process->addArgument(usr);
process->addArgument( "VERHDSP" );
process->addArgument(bar);
```

// OLFIX styrprogram  
// userid  
// OLFIX funktion  
// Bokföringsår

Detta blir:

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

Samt FTGDSP

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

// OLFIX styrprogram  
// userid  
// OLFIX funktion  
// Företagsnamn

Detta blir:

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

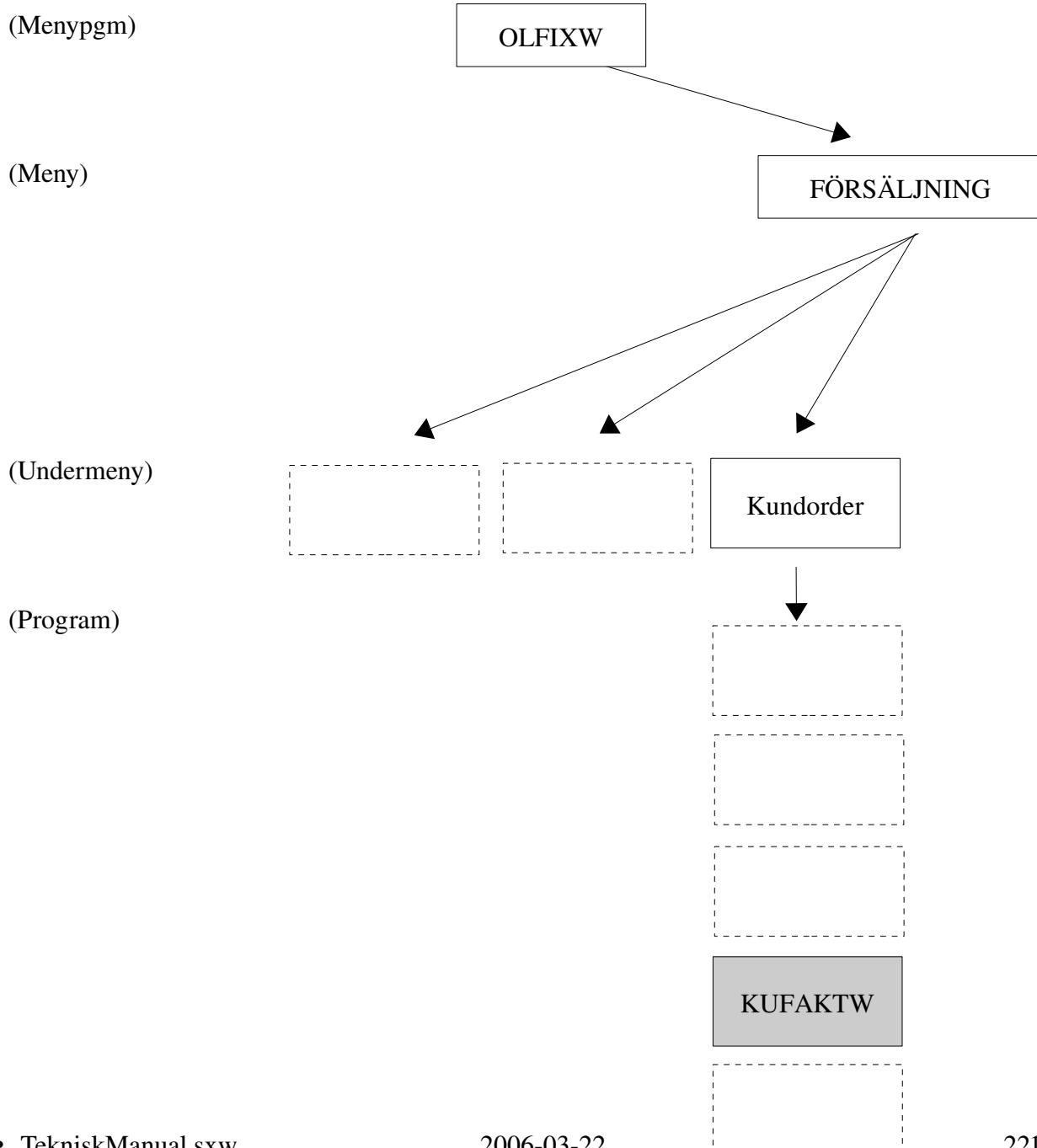
### Behörighetskrav:

För att kunna köra HUVBOKW behövs behörighet till  
PRGLST  
FTGDSP  
VERHDSP  
HBOKRPT

## KUFAKTW.... Fakturering

KUFAKTW, ett grafiskt program för att skapa en kundfaktura. För formatering och utskrift av fakturan används Kugar.

Programmet plockar upp userid från environment.





KUFAKTW - Fakturera

Ordernr 62	Orderstatus A	Hjälp	Datum 2005-11-13	Fakturarnr 100015	<input checked="" type="radio"/> Fakturanrserie 1 <input checked="" type="radio"/> Fakturanrserie 2						
<b>Fakturaadress</b>			<b>Kundorder.</b>								
Kundnr: ..... Namn: ..... Adress: ..... Postnummer: ..... Postadress: ..... Land: ..... Referens: .....	4375 Lilla Kunden Efr AB Bakgatan 1C 199 09 SMÄSTAD Sverige Lillemor Andrén	Förfallodatum: 2005-12-13	Ordnr 35 36 37 38 39 40 62 63	Kundnr 4383 4376 4379 4379 4377 4377 4375 4378							
Radnr 000	Artikelnummer 010 1000-1001 020 1000-1002	Benämning Att använda GNU/LINUX Linux MAGNUM	Beställt antal 0	Levererat antal 0	Pris/st 0	Moms% 25.00	Moms Kr 87.75	Radtotal inkl moms 438.75	Godkänn rad Ja Nej		
Radnr 010 020	Artikelnr 1000-1001 1000-1002	Benämning Att använda GNU/LINUX Linux MAGNUM	Ant 10 20	Lev.ant 3 3	Pris/st 117.00 117.00	Moms% 25.00	Moms Kr 87.75	Radtotal 438.75			
Skapa faktura/skriv ut			Räkna om fakturans summor.			Nettobelopp: 702.00	Frakt: 90.00	Fraktmoms % 25.00	Summa moms: 198.00	Avrundning: -0.00	Total: 990.00
OK	Nästa	Avbryt	Kalkylera								

**OLFIX AB**

**Faktura**

Fakturadatum

2005-11-19

FakturNr

100070

Ordernr

35

Sida

1

Kundnummer

4383

Leveransadress

Ulla Persdotter  
Storgatan 1  
199 99 STORSTAD  
Sverige

Fakturaadress

Ulla Persdotter  
Storgatan 1  
199 99 STORSTAD  
Sverige

Betalningsvillkor

Kontantköp

Leveransvillkor

Valuta

Leveranssätt

E referens  
U Persdotter

Vår referens  
Jan Pihlgren

**Godsmärke**  
Godsmärke

Artikelnr Pos Benämning	Beställt antal	Levererat antal	Pris per st	Radens summa
010 1000-1003 Linux På egen hand	1	1	117.00	146.25
020 1000-1007 The C Programming lang.Facit	1	1	117.00	146.25
030 1000-1011 Professional Linux Programming	1	1	117.00	146.25

Förfallodatum: **2005-11-19**

Nettobelopp	Frakt	Moms	Avrundning	Totalt
351.00	90.00	110.25	-0.25	551.00

OLFIX AB  
Box 70

199 98  
PROGSTAD

Telefon: 09-199300

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 KUFAKTW.

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

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

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

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument( "ORDCHK" );            // OLFIX funktion
process->addArgument("2");                   // check orderstatus
process->addArgument(kundordernr);

procfaktura = new QProcess();
procfaktura->addArgument("./STYRMAN");       // OLFIX styrprogram
procfaktura->addArgument(usr);               // userid
procfaktura->addArgument( "ORDRUPD" );        // OLFIX funktion
procfaktura->addArgument(ordernum);
procfaktura->addArgument(radnum);
procfaktura->addArgument(levantal);
procfaktura->addArgument(restantal);

proctrhd = new QProcess();
proctrhd->addArgument("./STYRMAN");          // OLFIX styrprogram
proctrhd->addArgument(usr);                  // userid
proctrhd->addArgument( "TRHDADD" );          // OLFIX funktion
proctrhd->addArgument(trhdData);             // TRNSID
proctrhd->addArgument(tidpunkt);
proctrhd->addArgument(usr);
proctrhd->addArgument(data);

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

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument( "ORDLST2" );           // OLFIX funktion

process = new QProcess();
process->addArgument("./STYRMAN");           // OLFIX styrprogram
process->addArgument(usr);                   // userid
process->addArgument( "FTGUPD" );            // OLFIX funktion
if (fakturanrflag=="2"){
    process->addArgument( "FKNR2" );
} else{
    process->addArgument( "FAKNR" );
}
```

```

process->addArgument(fakturanr);

prockresk = new QProcess();
prockresk->addArgument("./STYRMAN"); // OLFIX styrprogram
prockresk->addArgument(usr); // userid
prockresk->addArgument( "KRESADD"); // OLFIX funktion
prockresk->addArgument(kureskpost);

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

procorder = new QProcess(this);
procorder->addArgument("./STYRMAN"); // OLFIX styrprogram
procorder->addArgument(usr); // userid
procorder->addArgument( "ORDUPD"); // OLFIX funktion
procorder->addArgument("1"); // 1 = uppdatera fältet ORDERSTATUS
procorder->addArgument(ordernr); // för ordernr
procorder->addArgument("F"); // med värdet F (fakturerat)

procorad = new QProcess(this);
procorad->addArgument("./STYRMAN"); // OLFIX styrprogram
procorad->addArgument(usr); // userid
procorad->addArgument( "ORADUPD"); // OLFIX funktion
procorad->addArgument(temp); // antal orderrader
procorad->addArgument(tmp); // data

```

## Funktioner:

(ej linEdit, EndOf..)

```

frmKundFaktura::init()
frmKundFaktura::radioButton1_clicked()
frmKundFaktura::radioButton2_clicked()
frmKundFaktura::CalculateRadtotal()
frmKundFaktura::pushButtonOK_clicked()
frmKundFaktura::calculateFaktura()
frmKundFaktura::createFaktura()
    frmKundFaktura::CreateReportHeader()
    frmKundFaktura::createFakturaRad()
    frmKundFaktura::CreateReportFot()
    frmKundFaktura::KugarVersion()
    frmKundFaktura::CallKugar()
frmKundFaktura::GetOrderHeader()
frmKundFaktura::GetOrderRow()
frmKundFaktura::GetReportDir()
frmKundFaktura::checkStatus()
frmKundFaktura::listViewRader_format()
frmKundFaktura::listViewRader_clicked( QListWidgetItem * )
frmKundFaktura::pushBtnOKRad_clicked()
frmKundFaktura::pushBtnHelp_clicked()
frmKundFaktura::readResursFil()
frmKundFaktura::TRHDregAdd(QString trhdData, QString data )
frmKundFaktura::CreateReportHeader()
frmKundFaktura::createFakturaRad()
frmKundFaktura::CreateReportFot()
frmKundFaktura::KugarVersion()
frmKundFaktura::CallKugar()
frmKundFaktura::FileRemove(QString filnamn)
frmKundFaktura::getForetagsdata(QString posttyp)
frmKundFaktura::listKundorder()
frmKundFaktura::listKundordrar()
frmKundFaktura::slotPickupOrdernr( QListWidgetItem * item)
frmKundFaktura::updateFakturanr()
frmKundFaktura::createKURESKpost()
frmKundFaktura::getBetalvillkor(QString villkor)
frmKundFaktura::nextFaktura()

```

```
frmKundFaktura::updateOrderreg()
frmKundFaktura::updateOrderradreg()
```

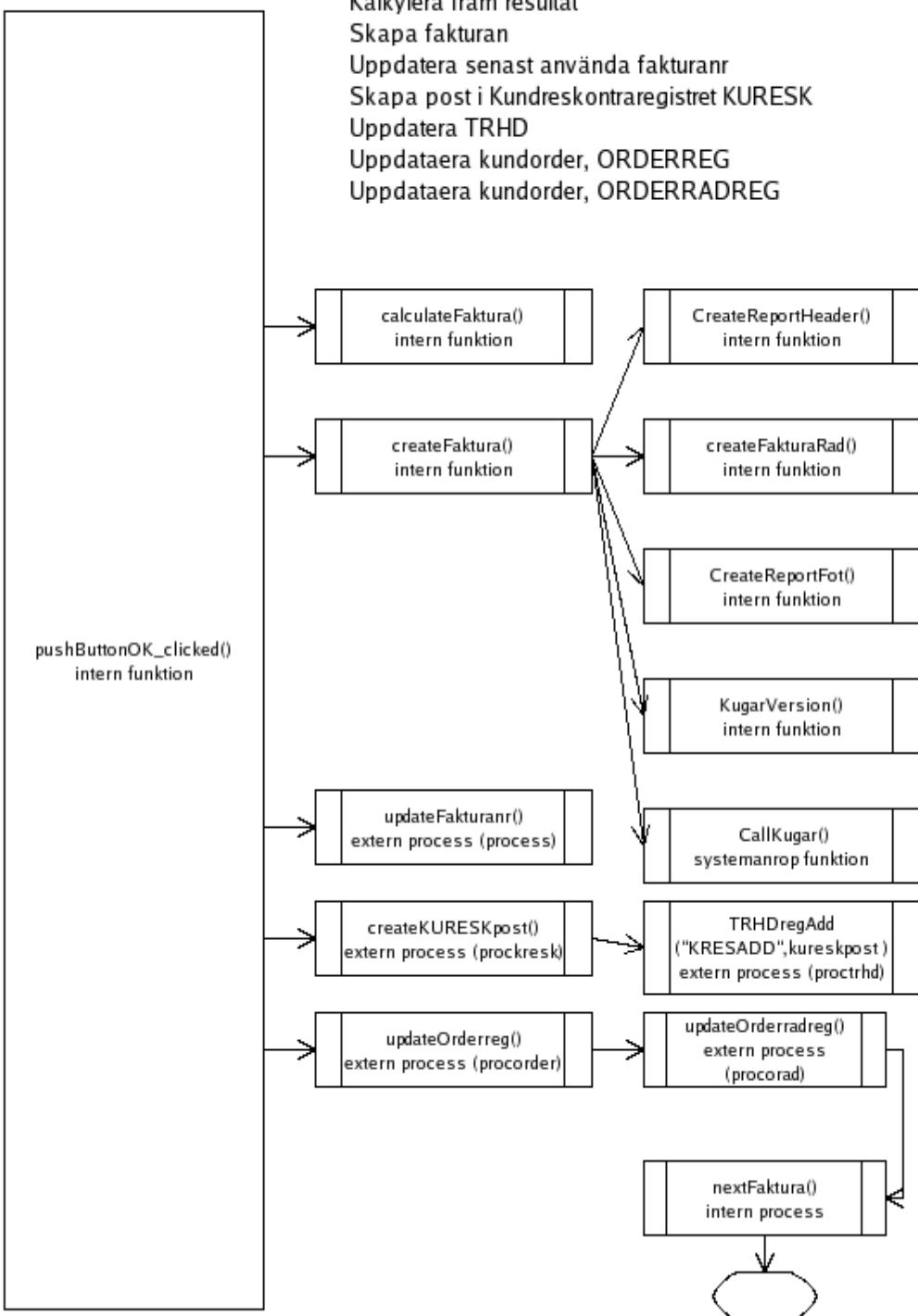
**Behörighetskrav:**

För att kunna köra KUFAKTW behövs behörighet till

PRGLST  
ORDDSP  
ORDRDSP  
ORDCHK  
ORDRUPD  
TRHDADD  
FTGDSP  
ORDLST2  
FTGUPD  
KRESADD  
BETDSP  
ORDUPD  
ORADUPD

## Kundfaktura

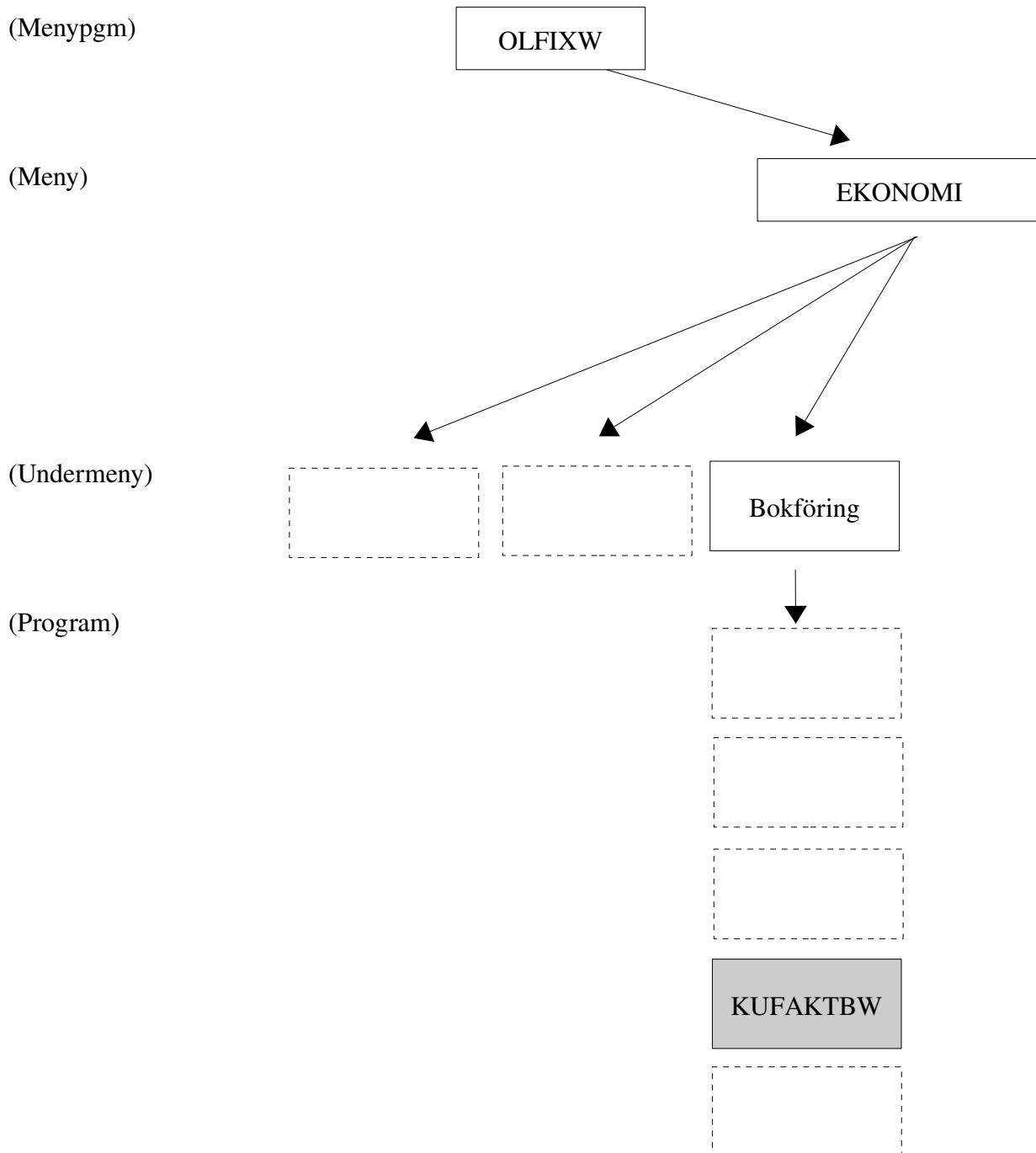
Godkänn fakturan  
 Kalkylera fram resultat  
 Skapa fakturan  
 Uppdatera senast använda fakturanr  
 Skapa post i Kundreskontraregistret KURESK  
 Uppdatera TRHD  
 Uppdataera kundorder, ORDERREG  
 Uppdataera kundorder, ORDERRADREG



## KUFAKTBW..... Registrera kundfakturor

KUFAKTBW, ett grafiskt program för att registrera inbetalda kundfakturor. Uppdatering sker av

kundreskontra, (bokföringen, kommer) samt historikregistret TRHD.  
Programmet plockar upp userid från environment.



KUFAKTBW - Registrera betalda kundfakturor

Fakturanr:	100013	Dagens datum:	2005-12-18	Hjälp
Ordernr:	63			
		Belopp		
		4000.00		
<input checked="" type="checkbox"/> Bokföring J/N				
Bokföringsår:	AF			
Ver.nr:	2			
Debetkonto:	1920	4000.00		
Kreditkonto:	1511	4000.00		
Godkänn		Avbryt		

**Obetalda kundfakturor**

Fakturanr	Fakturadatum	Kundnr	
000000	2005-11-17	4383	↑
100007	2005-11-12	4375	
100010	2005-11-13	4375	
100011	2005-11-13	4378	
100012	2005-11-13	4378	
100018	2005-11-14	4379	
100019	2005-11-14	4379	
100020	2005-11-14	4375	
100021	2005-11-14	4379	
100022	2005-11-14	4379	
100023	2005-11-14	4379	
100024	2005-11-14	4379	
100025	2005-11-14	4377	

**Betalda kundfakturor**

Fakturanr	Belopp	Kundnr	Fakt
100009	877.50	4375	2005
100013	4000.00	4378	2005

**Dagens datum** kommer att användas som betalningsdatum för fakturan.

Genom att klicka på ett fakturanr i tabellen/listan **Obetalda kundfakturor** kopieras fakturanumret till fältet **Fakturanr**. Eller så kan man skriva fakturanumret direkt i fältet **Fakturanr**.

Avsluta med Enter .

Information om fakturan hämtas från kundreskontran och presenteras i fälten **Ordernr, Summa** samt i fälten för belopp för **Debetkonto** och **Kreditkonto**.

Kontonumret för **Debetkonto** och **Kreditkonto** hämtas från företagsregistret (se DSPFTGW).

Bägge fälten kan givetvis redigeras (ändras) i samband med registreringen.

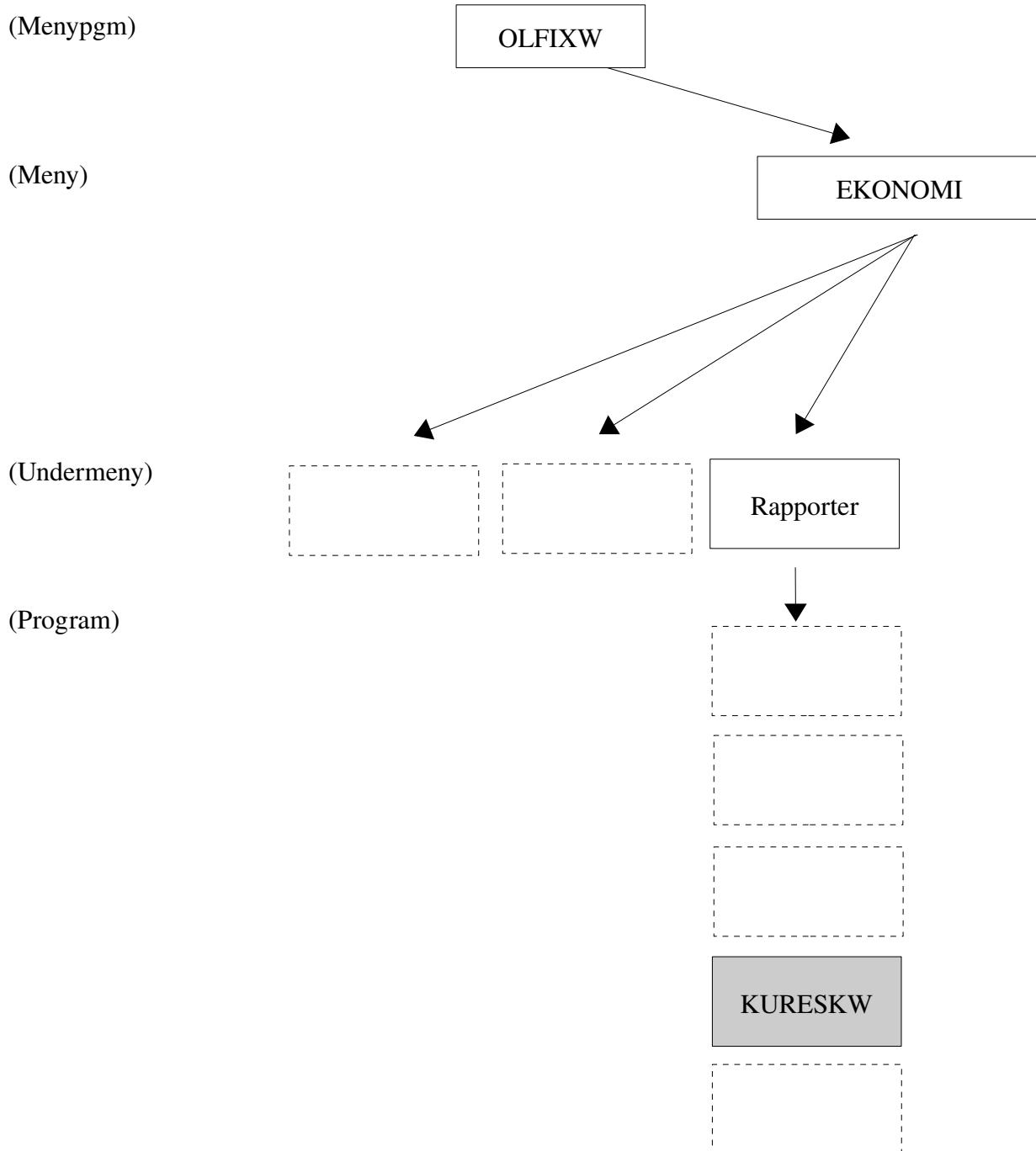
Genom att markera en faktura i listan av Obetalda fakturor och sedan klickar på "högerpil" flyttas fakturaposten till listan **Betalda fakturor**.

När man sedan klickar på **Godkänn** så startas bearbetningen och kundreskontran uppdateras.

I fall man bockat för **Bokföring J/N** så bokförs dessutom fakturorna i redovisningen.

## KURESKW..... Kundreskontra

KURESKW, ett grafiskt program för att lista kundreskontra på skärm.  
Programmet plockar upp userid från environment.



X KURESKW - Kundreskontra

Datum: 2005-11-15

Ordernr	Fakturanr	Kundnr	Fakturabelopp	Förfallodatum	Status
37	100016	4379	585.00	2005-12-14	N
37	100017	4379	585.00	2005-12-14	N
37	100018	4379	585.00	2005-12-14	N
37	100019	4379	585.00	2005-12-14	N
37	100021	4379	585.00	2005-12-14	N
37	100022	4379	585.00	2005-12-14	N
37	100023	4379	585.00	2005-12-14	N
37	100024	4379	585.00	2005-12-14	N
39	100025	4377	292.00	2005-12-14	N
39	100026	4377	585.00	2005-12-14	N
39	100027	4377	585.00	2005-12-14	N
39	100028	4377	585.00	2005-12-14	N

Sluta    Kundfordringar totalt: 40798.00

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 programmet KURESKW.

KURESKW anropar KRESLST 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("KRESLST");             // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr KRESLST
```

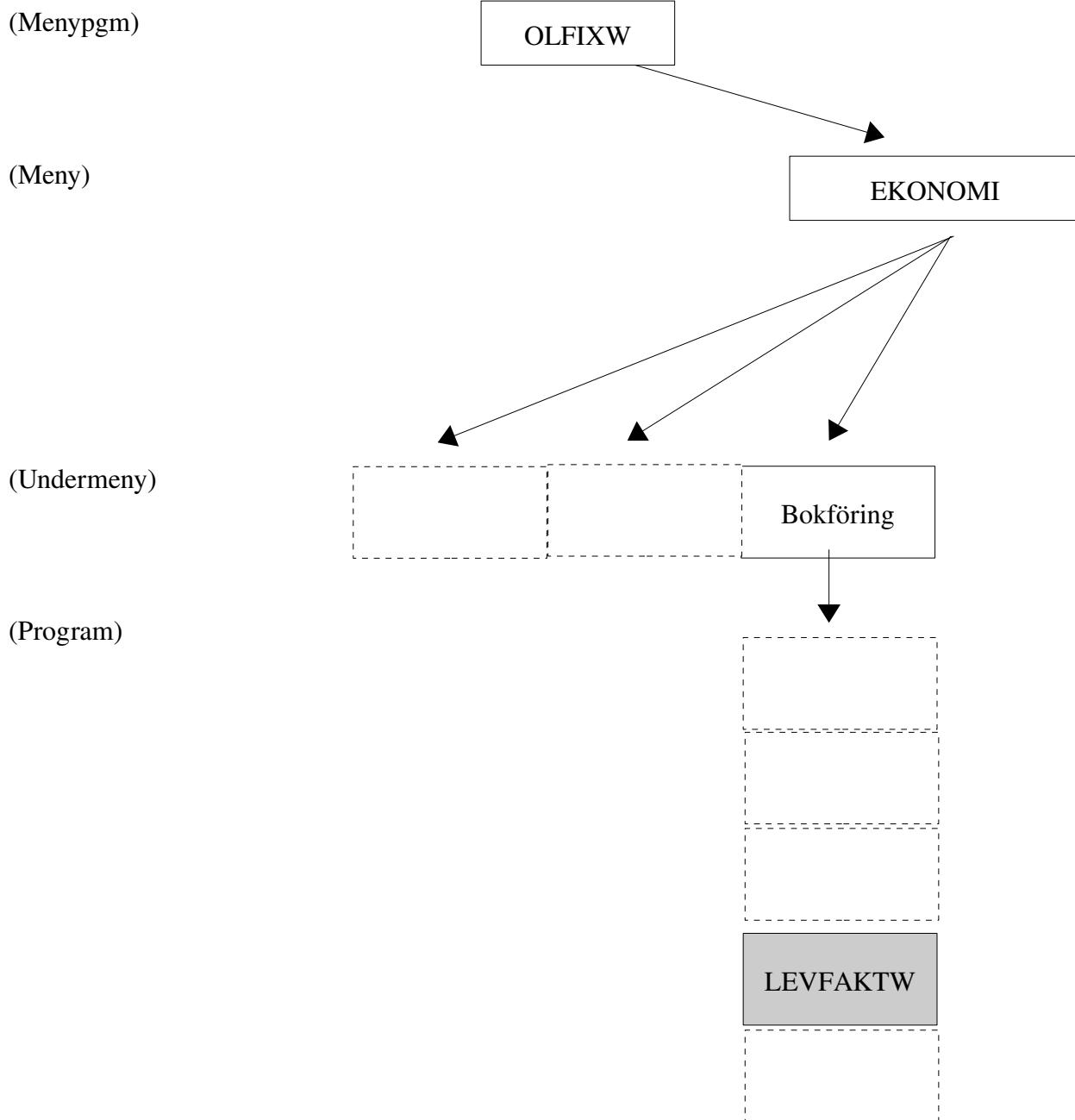
usr är den inloggades userid.

#### **Behörighetskrav:**

För att kunna köra KURESKW behövs behörighet till  
PRGLST  
KRESLST

## **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.



## 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 %**.

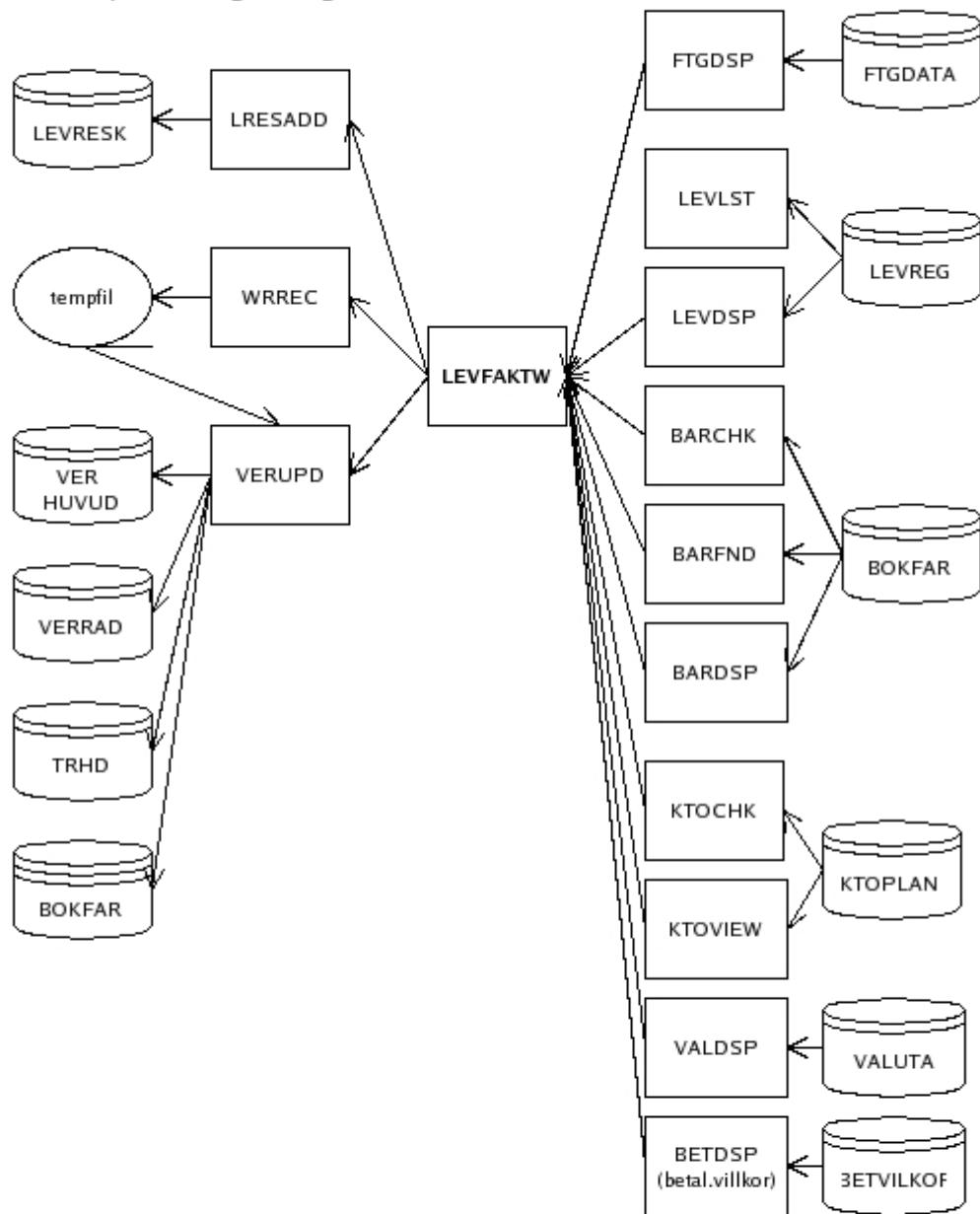
Ifall man ändrar bokföringsår så hämtas kontoplanen för angivet bokföringsår och presenteras i listan.

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 leverantörsfakturna.

Leverantörsnummer:	<input type="text"/>	Leverantörens organisationsnummer:	<input type="text"/>		
Leverantörsnamn:	<input type="text"/>				
Adress:	<input type="text"/>		Postnr:	<input type="text"/>	
Postadress:	<input type="text"/>		Land:	<input type="text"/>	
Kundnummer:		Registreringsdatum: <input type="text" value="2003-08-13"/>			
Fakturanummer:		<input type="text"/>			
Fakturadatum:		<input type="text"/>			
Betalningsvillkor:		dagar netto			
Förfallodatum:		<input type="text"/> OCR nummer: <input type="text"/>			
Fakturatext:		<input type="text"/>			
Bokföringsår:	<input type="text" value="AD"/>	Verifikationsnr:	<input type="text"/>	Momskod:	<input type="text"/>
Valuta:	<input type="text"/>	Valutakurs:	<input type="text"/>	Moms %	<input type="text"/>
Fakturabelopp:	<input type="text"/> i fakturans valuta.				
Kontonr: (Kredit)	<input type="text"/>	Bokfört belopp (SEK):	<input type="text"/>		
Momskontonr:	<input type="text"/>	Momsbelopp (SEK):	<input type="text"/>		
Kontonr: (Debet)	<input type="text"/>	Debetbelopp (SEK):	<input type="text"/>		

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 inventarie

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");
process->addArgument(usr);
process->addArgument( "LRESADD" );
process->addArgument(levnr);
process->addArgument(fakturanr);
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 momspcent
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

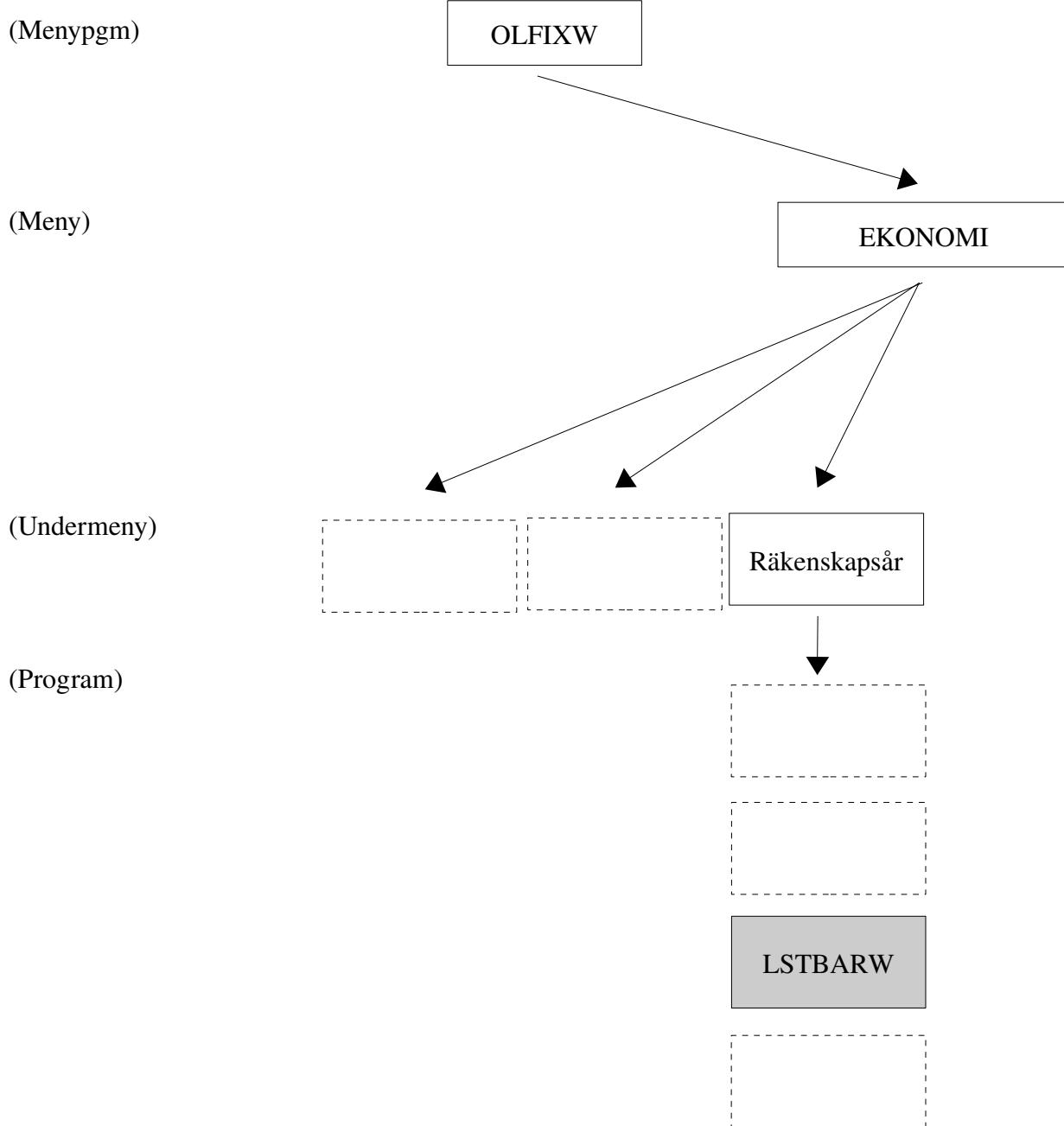
KTOCHK  
KTOVIEW  
WRREC  
VERUPD

## LSTBARW....Lista räkenskapsår

LSTBARW, ett grafiskt program för att lista befintliga räkenskapsår (bokföringsår). Anropet sker utan parametrar.

För att använda programmet krävs behörighet till funktionerna BARLST.

Programmet plockar upp userid från environment.



X-> LSTBARW Lista bokföringsår

Bokföringsår	Benämning	Kontoplan	
AD	2003-01-01 -- 2003-12-31 EUBAS97		
AF	2005-01-01 -- 2005-12-31 EUBAS97		

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 LSTBARW.

LSTBARW anropar BARLST 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("BARLST"); // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr BARLST
```

### Behörighetskrav:

För att kunna köra LSTBARW behövs behörighet till

PRGLST

BARLST

## 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.

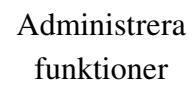
(Menypgm)



(Meny)



(Popdown)



(Popdown)



X-> OLFIX - LSTFNCW Lista funktioner (transaktionstyper)

Transid	Trans.beskrivning
ARICHK	Kontrollera om visst bokföringsår finns
BARCHK	Kontrollera om visst bokföringsår finns
BARDSP	Hämta data för angivet bokföringsår
BOKF	Bokföringsprogram
FORDSP	Visa företagsdata
FTGUPD	Uppdatera företagsdata
KSTADD	Nyupplägg av kostnadsställe
KSTCHK	Kontrollera om visst kostnadställe finns
KSTDSP	Visa info för ett kostnadställe
KSTLST	Lista kostnadsställen på skärm
KTOADD	Lägga till en ny post i konto
KTOCHG	Ändra info för ett kontonr
KTOCHK	Kontrollera om visst konto finns
KTODEL	Ta bort post i konto (endast utan transaktioner)
KTODSP	Visa info nätt konto

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.

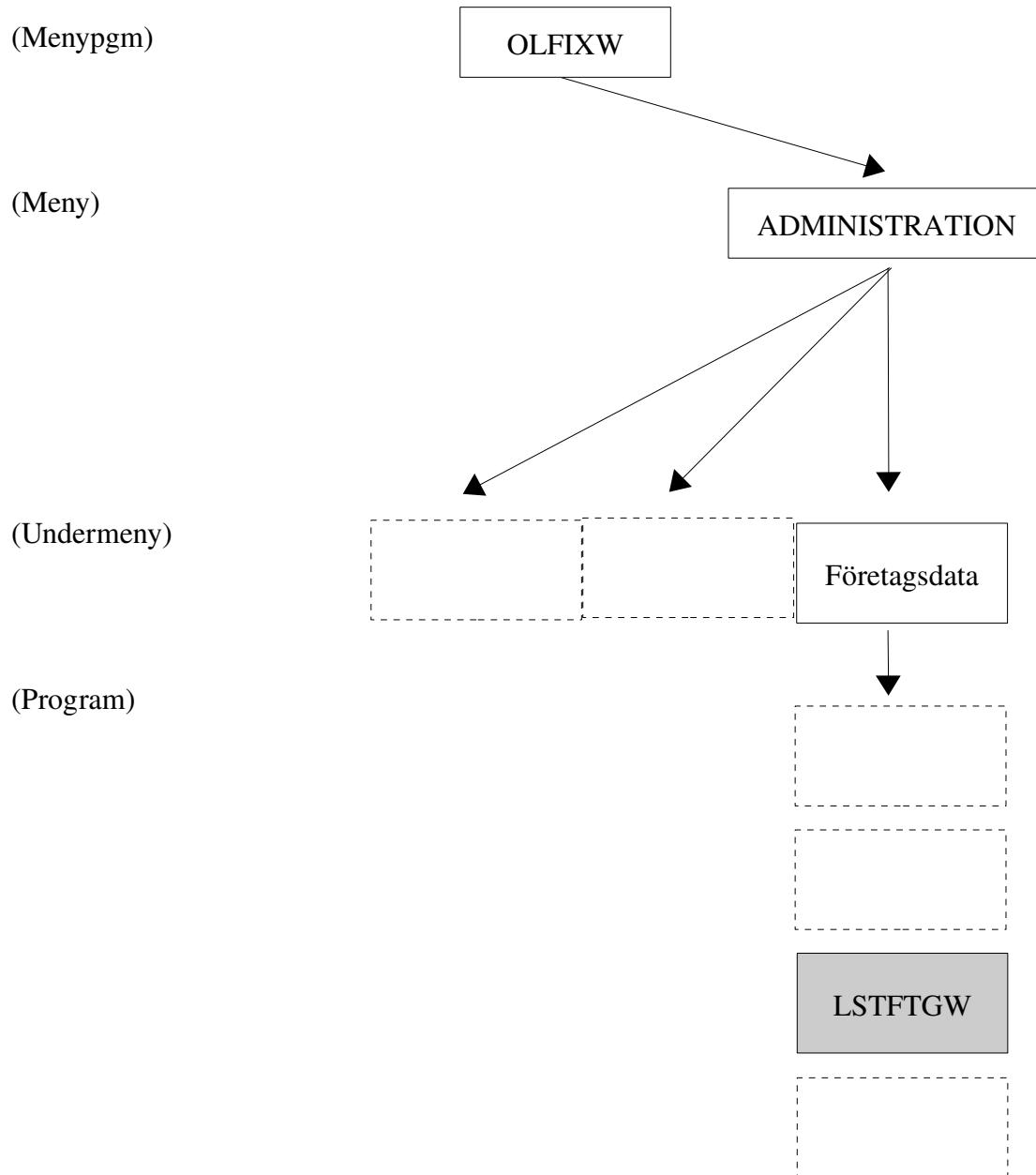
Programmet använder funktionen TRNSLST för att hämta data från databasen. Anropet av TRNSLST sker via STYRMAN.

**Behörighetskrav:**

För att kunna köra LSTFNCW behövs behörighet till  
PRGLST  
TRNSLST

## LSTFTGW.....Lista företagsdata

LSTFTGW, ett grafiskt program för att lista samtliga posttyper med data för företaget.  
Programmet plockar upp userid från environment.



LSTFTGW Lista företagsdata

Posttyp	Beskrivning	Data
ADR1	Postadress	Box 70
ADR2	Postnummer till Postadress	199 98
ADR3	Ort till Postadress	PROGSTAD
ADR4	Besöksadress	Syntaxvägen 99
ADR5	Postnr till Besöksadress	199 98
ADR6	Ort till Besöksadress	PROGSTAD
ADR7	Godsadress	Verktygsgatan 11
ADR8	Postnr till Godsadress	199 97
ADR9	Ort till Godsadress	PROGSTAD
AUTOK	Automatkontering J/N	N
BF1	Bokföringsperiod 1	Januari
BF10	Bokföringsperiod 10	Oktober
BF11	Bokföringsperiod 11	Novemper
BF12	Bokföringsperiod 12	December
BF13	Bokföringsperiod 13	Januari

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 FTGLIS och FTGLST.

LSTFTGW anropar FTGLST 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( "FTGLST" );               // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr FTGLST
```

samt

FTGLIS 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( "FTGLIS" );               // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr FTGLIS
```

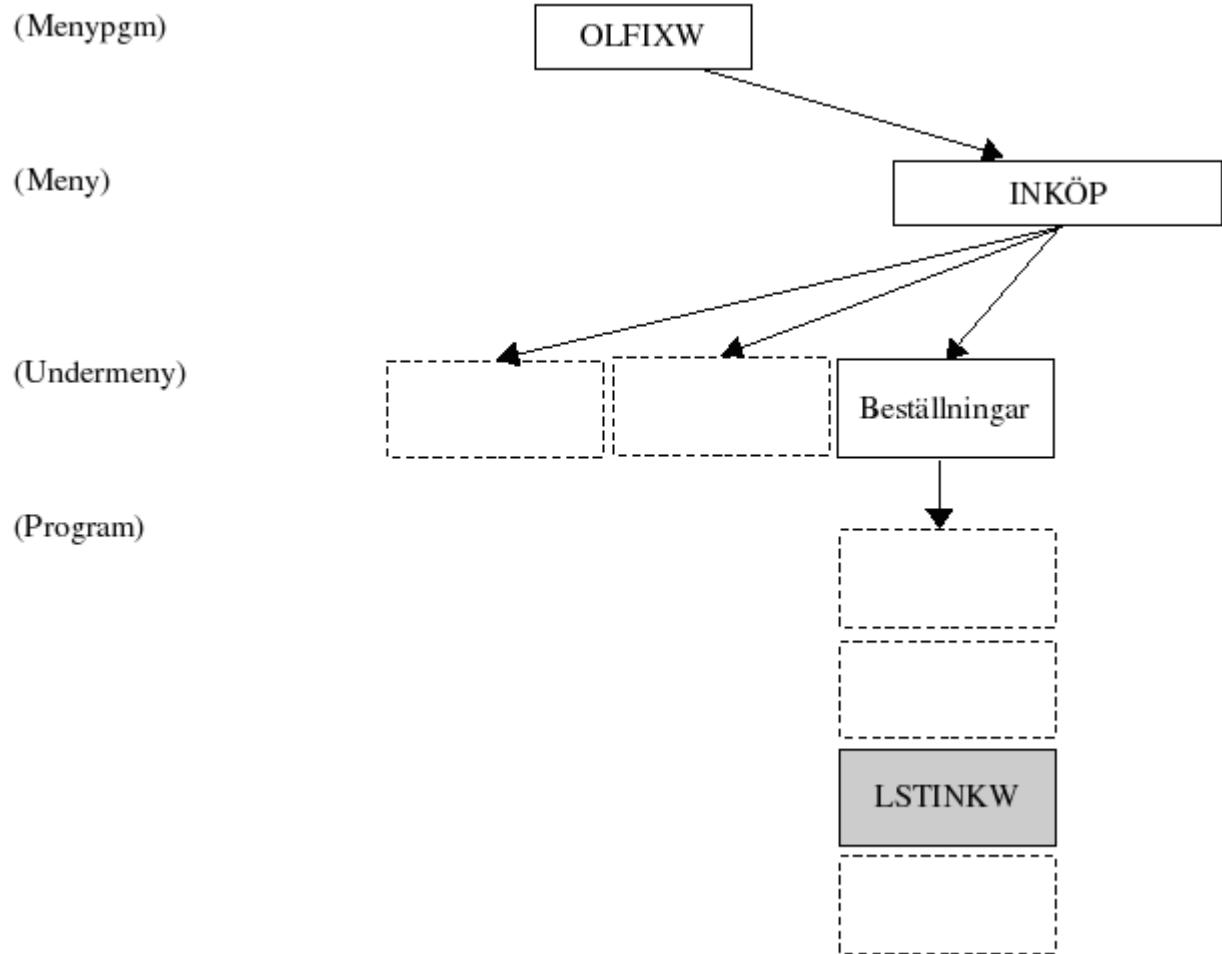
usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra LSTFTGW behövs behörighet till  
PRGLST  
FTGLIS  
FTGLST

## 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	4008496 326 389	Batteri LR03/AAA	403	E	100.00	25.00	75.00	34.90	2617.50
20	20	1000	4008496 326426	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

[Stand...]

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
```

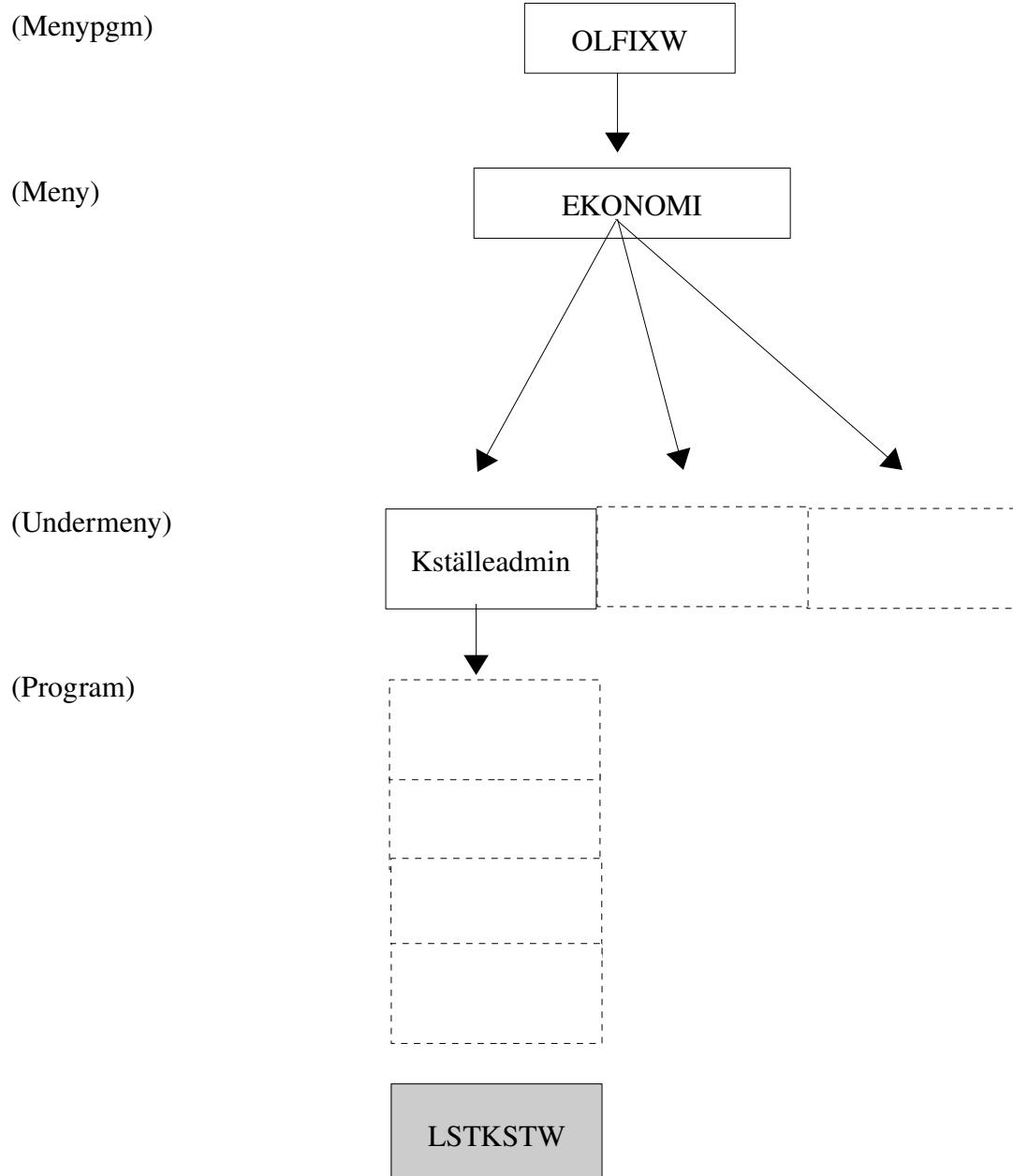
usr är den inloggades userid.

#### **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.



X-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
```

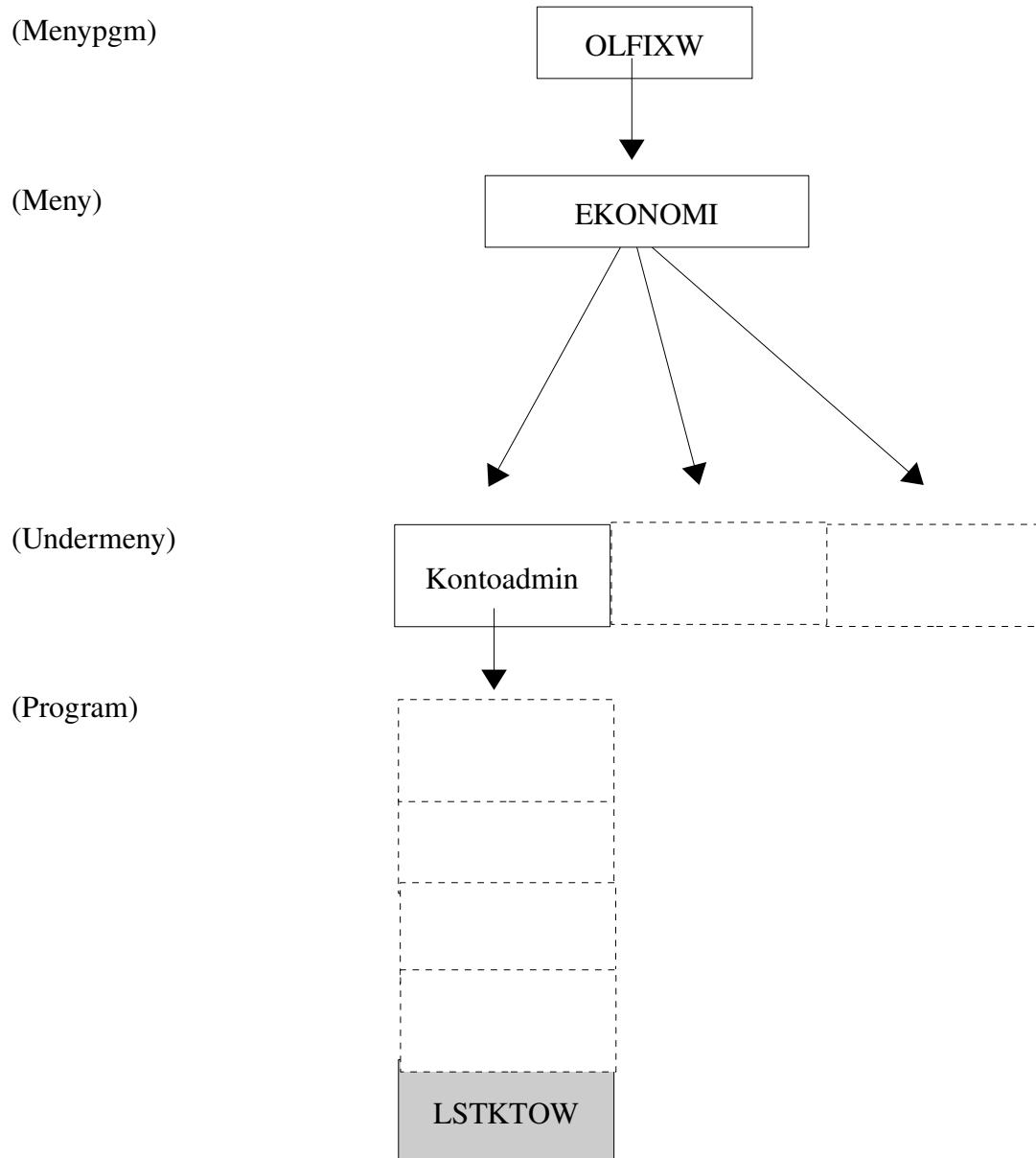
usr är den inloggades userid.

### **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.





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 parametrarna userid och arid

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

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

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

Detta blir:

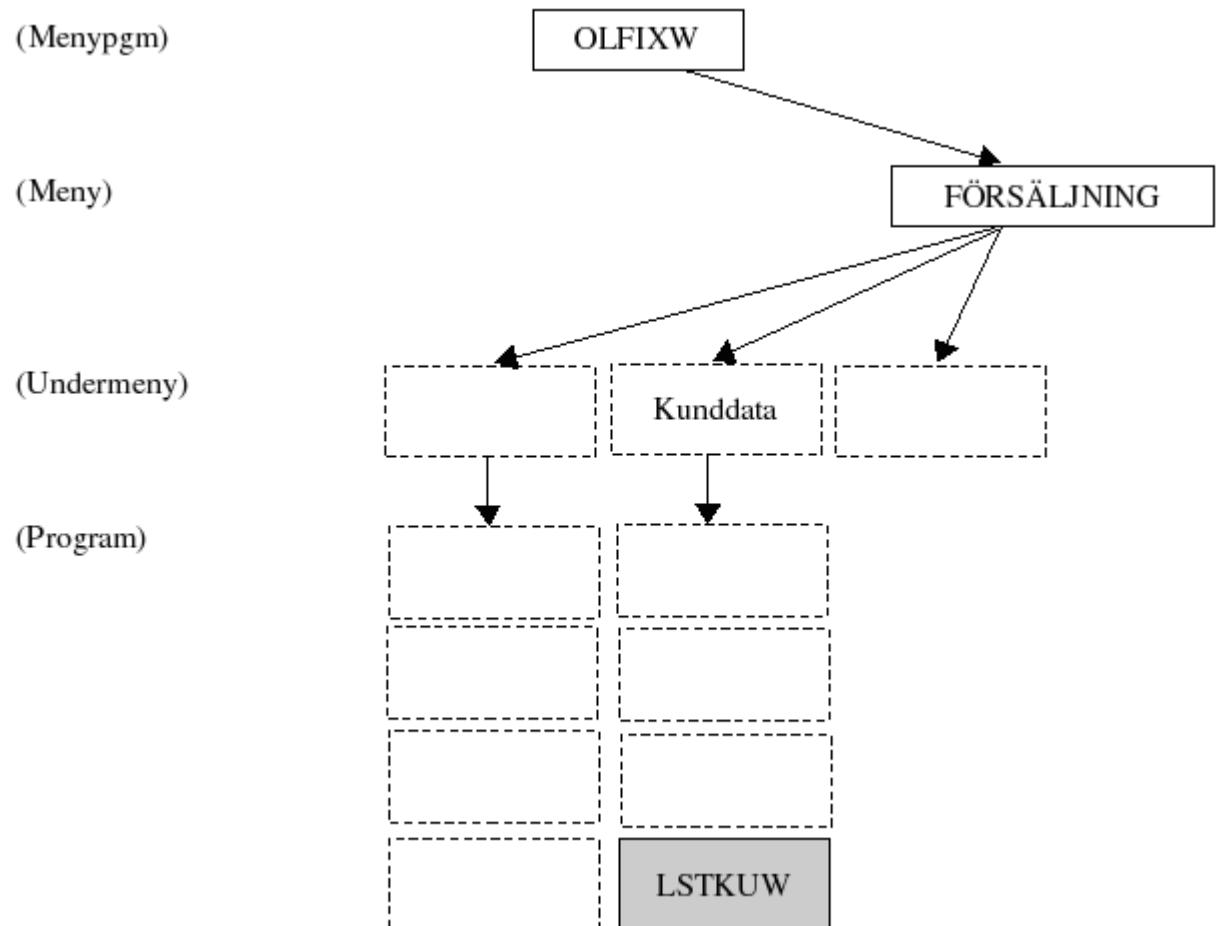
```
./STYRMAN usr KTOVIEW arid
```

### 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.



X-> OLFIX - LSTKUW Lista kunder

Kundnr	Namn
4377	Kund AB
4379	Småkund AB
4378	Storkund AB
4376	Test AB

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 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:

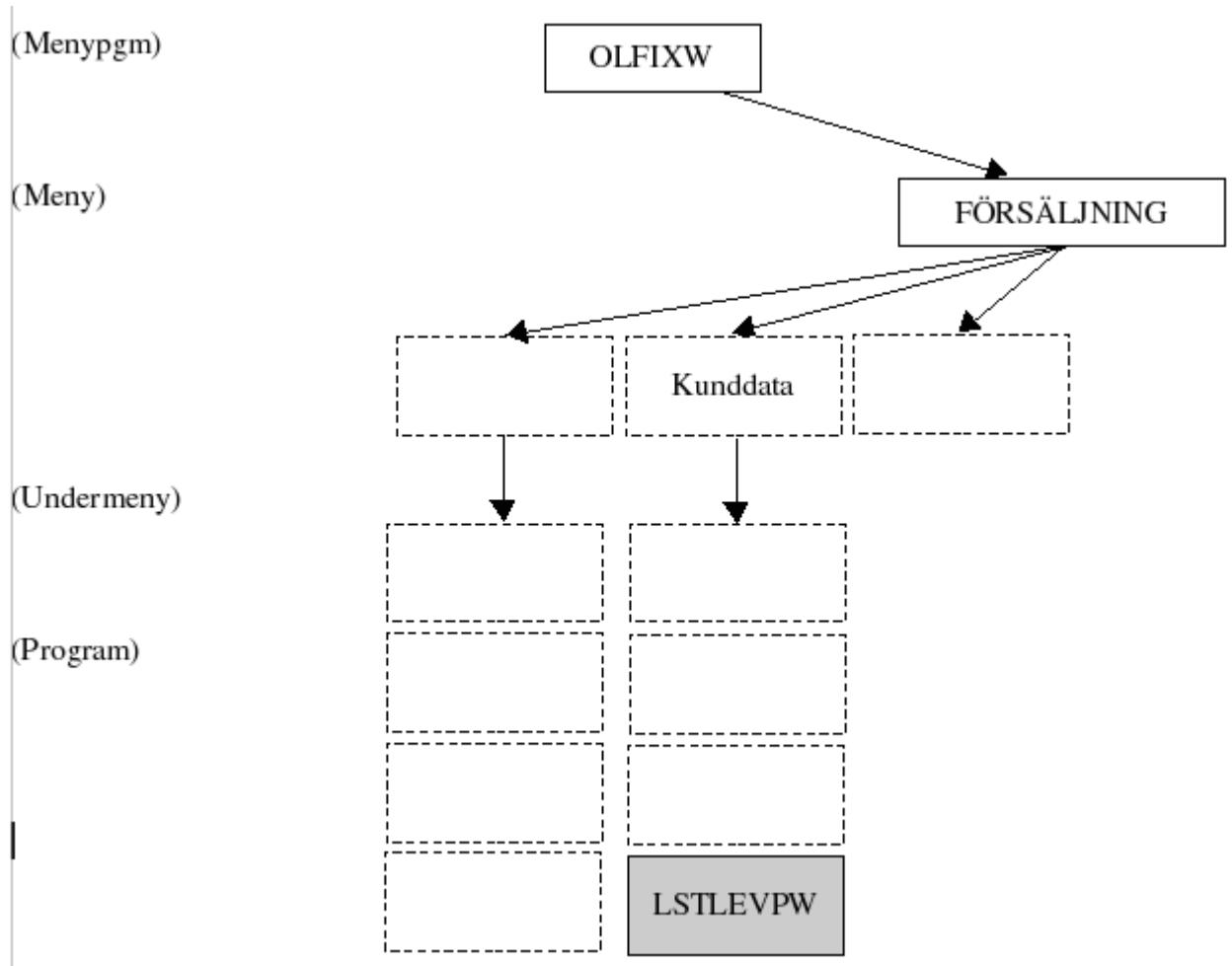
För att kunna köra LSTKUW behövs behörighet till

PRGLST

KULST

## LSTLEVPW.....Lista kunders leveransplatser

LSTLEVPW, ett grafiskt program för att lista **leveransplatser** på skärmen. Med leveransplatser menas här kundernas olika leveransadresser.



X OLFIX - LSTLEVPW Lista leveransplatser

Levplatsnr	Kundnr	Adress	Postnr	Postadress
001	4375	Bakgatan 1C	199 09	SMÅSTAD
002	4375	Bakgatan 1D	199 09	SMÅSTAD
555	4375	Testgatan 3	199 02	PROVSTAD

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 LSTLEVPW.

LSTLEVPW anropar LEVPLST 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("LEVPLST"); // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr LEVPLST
```

### Behörighetskrav:

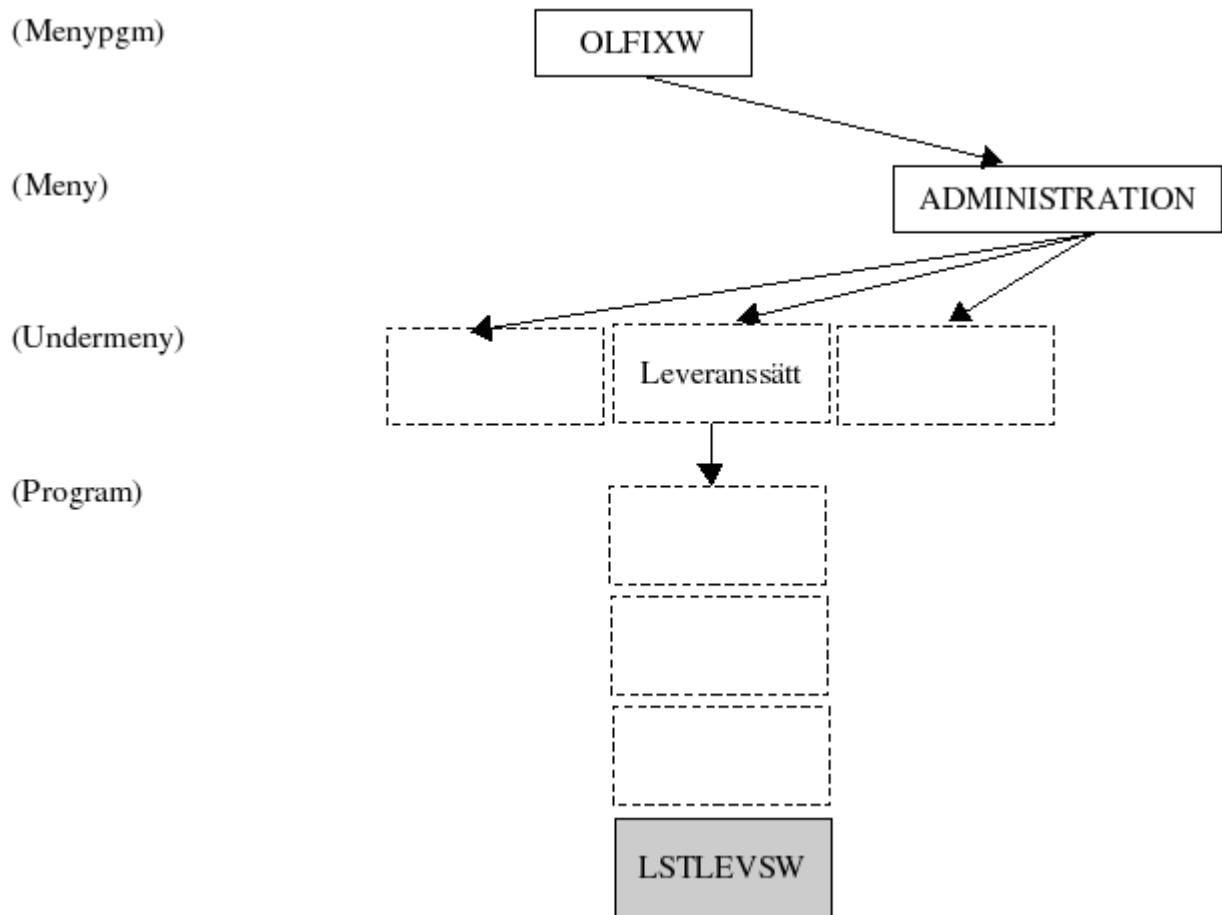
För att kunna köra LSTLEVPW behövs behörighet till

PRGLST

LEVPLST

## LSTLEVSW.....Lista leveranssätt

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



X-W LSTLEVSW - Lista leveranssätt

Leveranssätt	Beskrivning
001	Schenker kundnr:11105232
002	ASG. Kundnr 111111

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 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:

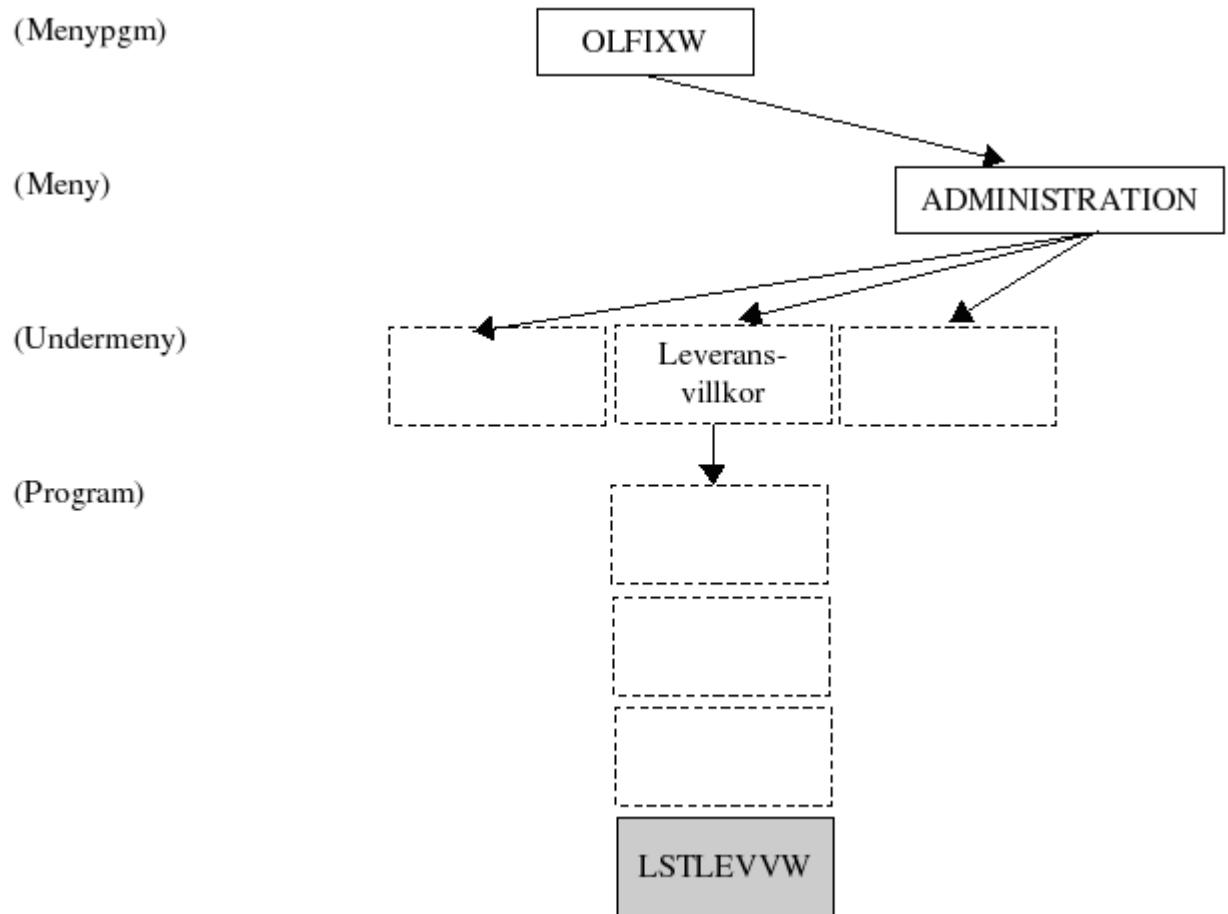
För att kunna köra LSTLEVSW behövs behörighet till

PRGLST

LEVSLST

## LSTLEVW....Lista leveransvillkor

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



X-LSTLEVW - Lista leveransvillkor

Lev.villkor	Beskrivning
001	EXW
002	EYW

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 LSTLEVVW.

LSTLEVVW anropar 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 LSTLEVVW behövs behörighet till

PRGLST

LEVVLST

## LSTORDW.....Lista kundorder

LSTORDW, ett grafiskt program för att lista kundorder på skärmen.

(Menypgm)

OLFIXW

(Meny)

FÖRSÄLJNING

(Undermeny)

Kundorder

(Program)

LSTORDW

X-OLFIX - LSTORDW - Lista kundorder

Ordernr	Kundnr	Leveransdatum	Orderstatus	Ordersumma
31	4377	2005-03-14	A	3260.00
32	4378	2005-03-14	A	0.00
33	4377	2005-03-14	A	9635.00

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 LSTORDW.

LSTORDW anropar ORDLST2 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("ORDLST2");     // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr ORDLST2
```

### Behörighetskrav:

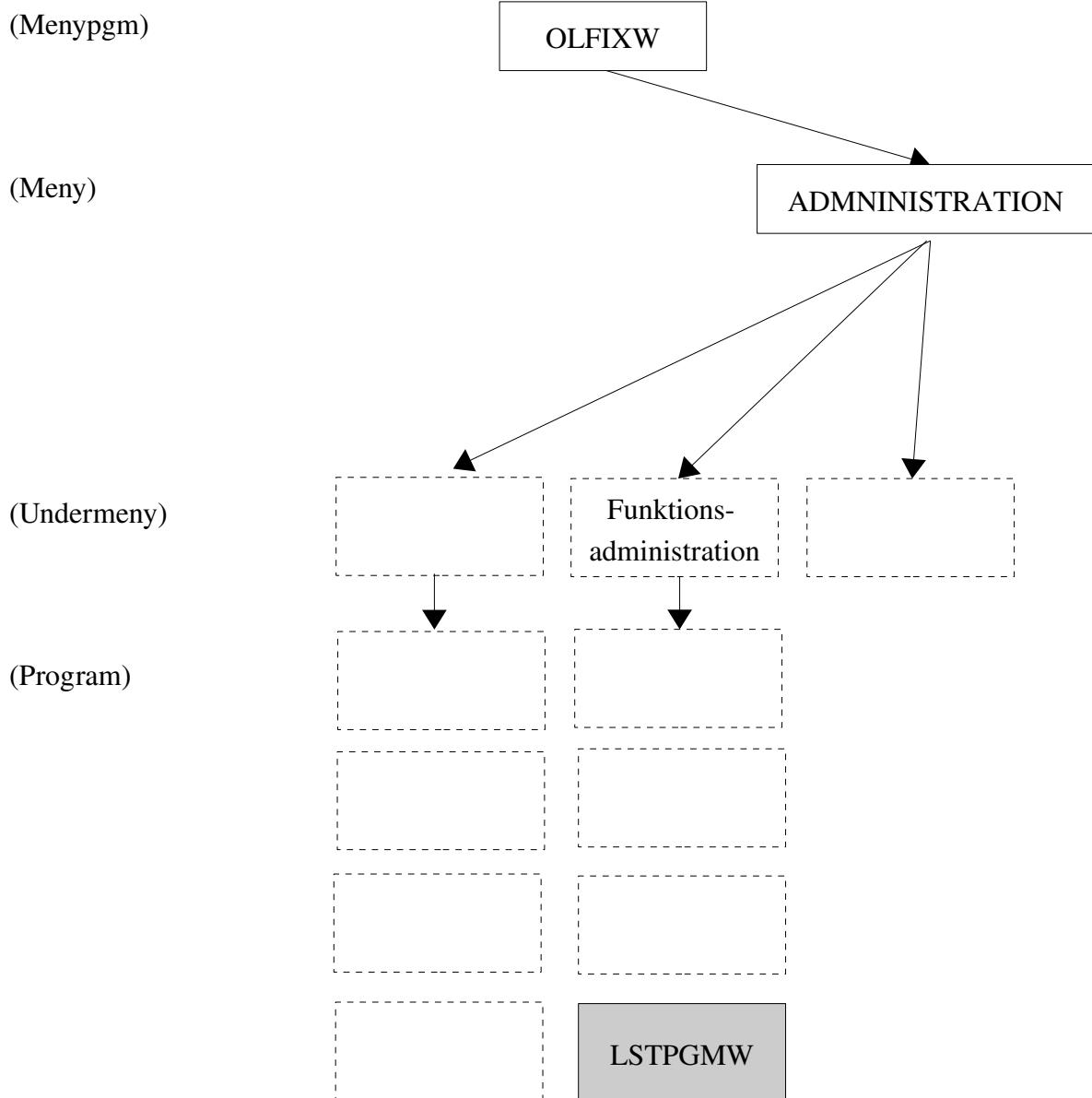
För att kunna köra LSTORDW behövs behörighet till

PRGLST

ORDLST2

## LSTPGMW.....Lista program

LSTPGMW, ett grafiskt program för att lista program/moduler på skärmen.



Program	Programbeskrivning
	Ta bort konto
	Ta bort kostnadställe
	Visa behörighet
	Ändra behörighet
	Ändra kostnadställe
ADDARW	Ny artikel
ADDBARW	Nytt bokföringsår
ADDBETVW	Nya betalningsvillkor
ADDFNCW	Ny funktion
ADDFORW	Ny databasregistrering
ADDFTGW	Ny post
ADDINKW	Registrera inköpsorder

Blankt i tabellen Program innebär att program saknas.

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 LSTPGMW.

LSTPGMW anropar PRGLST via STYRMAN utan parametrar.

Informationen hämtas från tabellen PROGRAM.

```
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("PRGLST"); // OLFIX funktion
```

Detta blir:

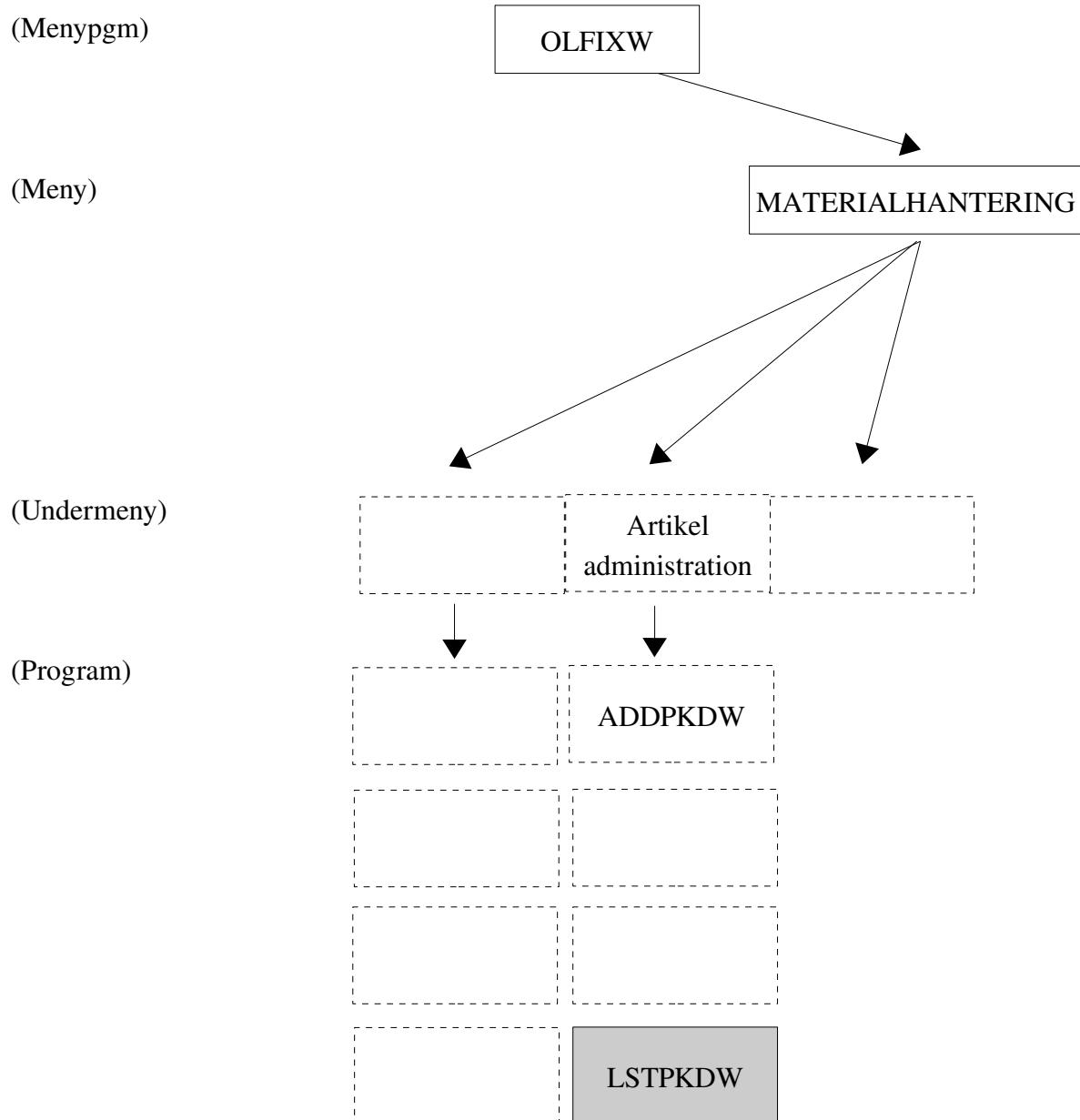
./STYRMAN usr PRGLST

### Behörighetskrav:

För att kunna köra LSTPGMW behövs behörighet till  
PRGLST

## LSTPKDW.....Lista produktgrupper

LSTPKDW, ett grafiskt program för att lista produktgrupper/produktklasser/ på skärmen.



X-> LSTPKDW - Lista produktklasser/produktgrupper/produktkoder

Produktklass	Beskrivning
2100	Switchar
2200	Hårddiskar
2300	Tangentbord
2400	Routers
2500	Bildskärmar

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 LSTPKDW.

LSTPKDW anropar PKDLST 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("PKDLST");      // OLFIX funktion
```

Detta blir:

./STYRMAN usr PKDLST

### Behörighetskrav:

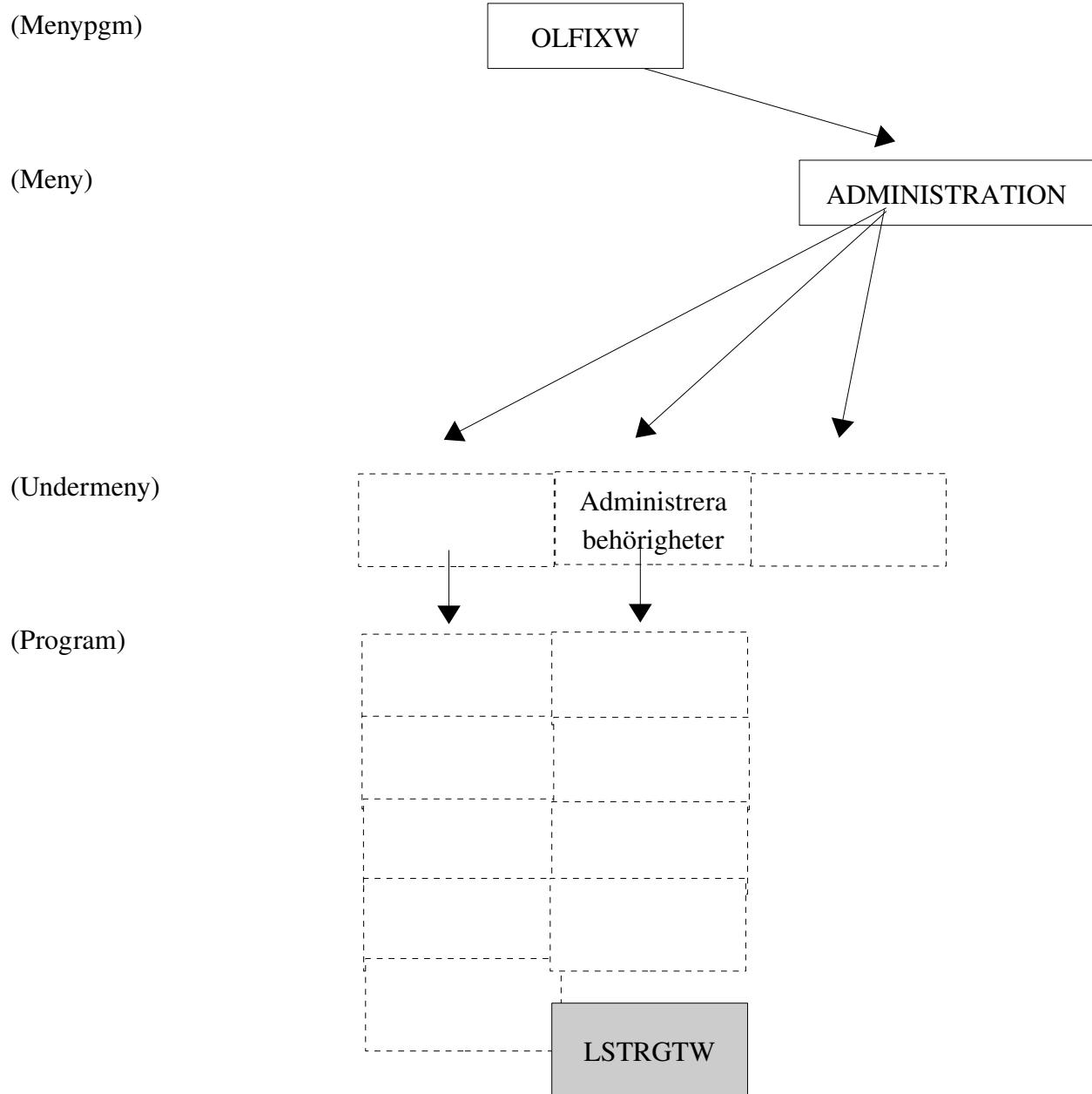
För att kunna köra LSTPKDW behövs behörighet till

PRGLST

PKDLST

## LSTRGTW.....Lista behörigheter

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



X-> OLFIX Lista behörigheter

Userid	Behörighet
JAN	ADDKSTW
JAN	ADDRGTW
JAN	ADDUSRW
JAN	ADDVALW
JAN	BARADD
JAN	BARCHK
JAN	BARDSP
JAN	BOKF
JAN	BOKFORW
JAN	CHGUSRW
JAN	CHGVALW
JAN	DELRGTW
JAN	DELVALW
JAN	DSPKSTW
JAN	DSPUSRW

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.

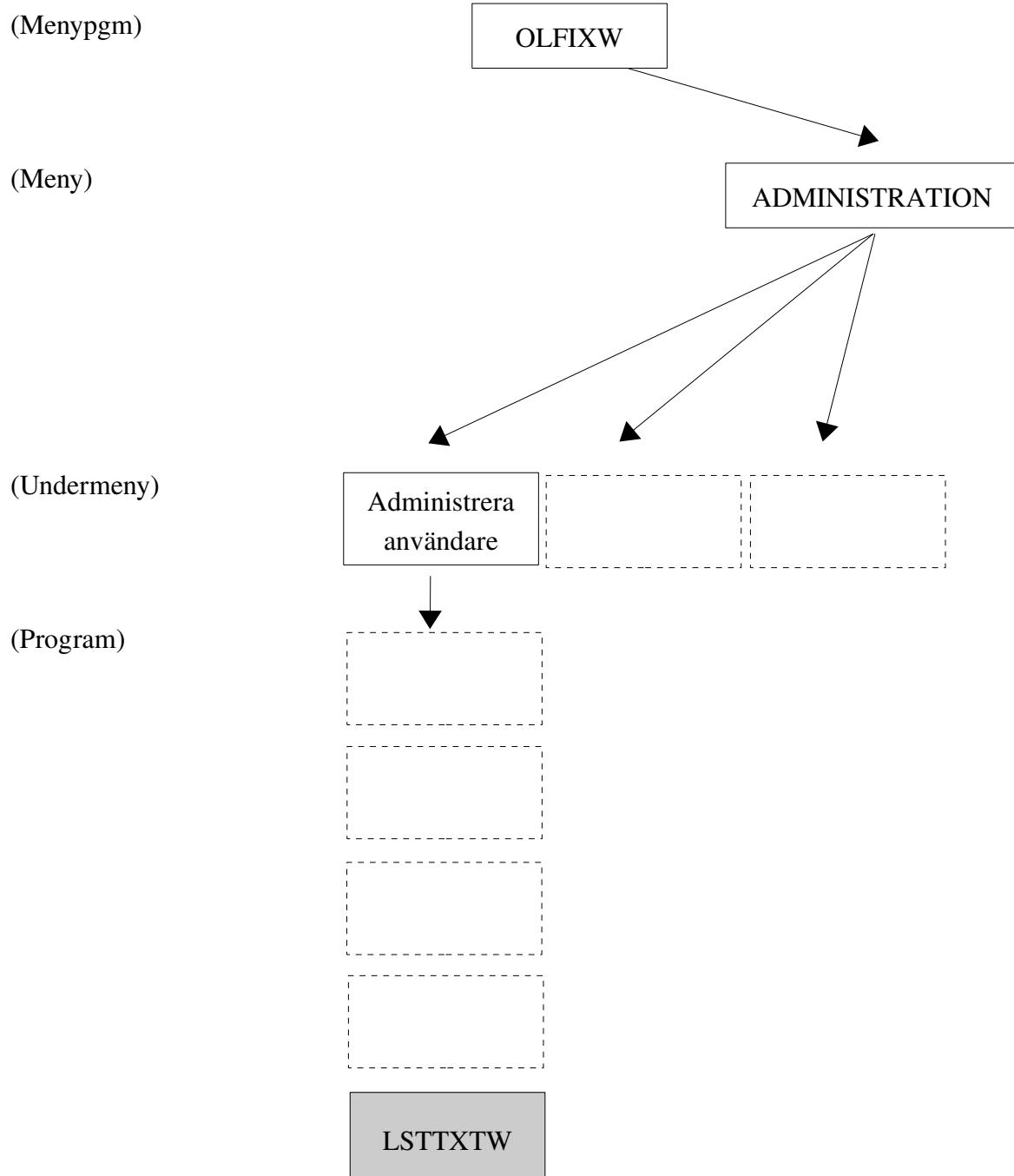
Programmet använder funktionen RGTLST för att hämta data från databasen. Anropet av RGTLST sker via STYRMAN.

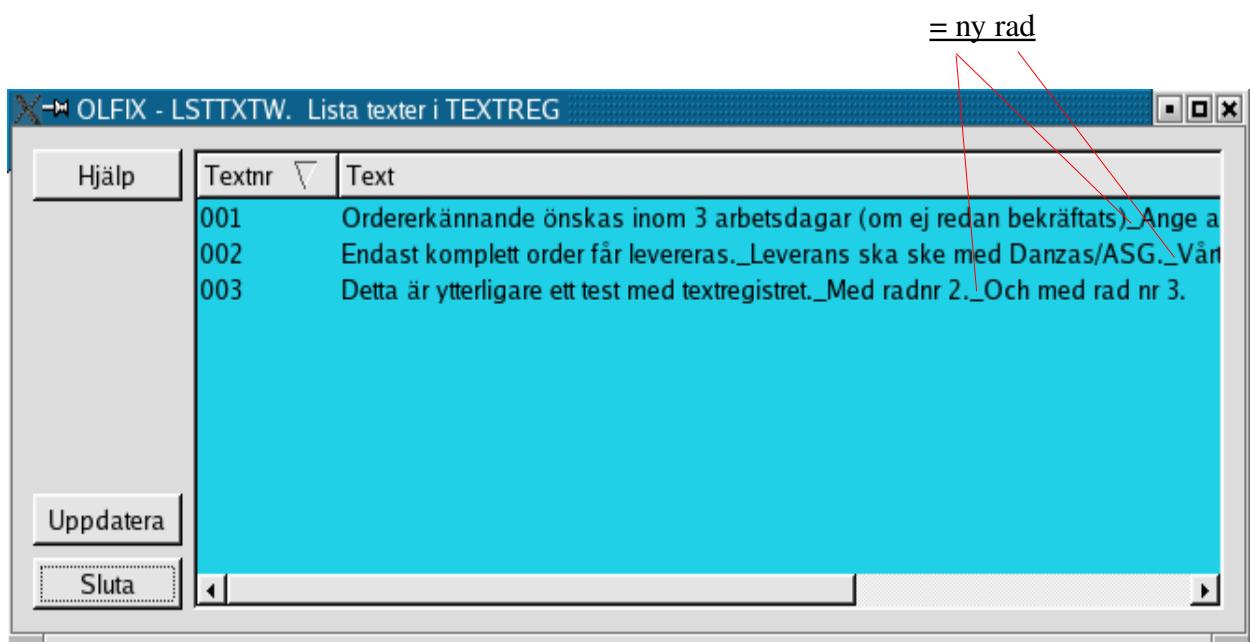
**Behörighetskrav:**

För att kunna köra LSTRGTW behövs behörighet till  
PRGLST  
RGTLST

## LSTTXTW.....Lista textregistrets poster

LSTTXTW, ett grafiskt program för att lista innehållet i textregistret på bildskärm.





\_ (underscore) används för att markera ny rad.

För att se texten formaterad och hur den ser ut vid utskrift titta på DSPTXTW.

Genom att klicka på en rad så öppnas DSPTXTW med **Textnr**.

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 TXTLST för att hämta data från databasen. Anropet av TXTLST sker via STYRMAN.

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

frmListTextReg::connect( process, SIGNAL(readyReadStdout()),this, SLOT(slotDataOnStdout()) );
frmListTextReg::connect( process, SIGNAL(readyReadStderr()),this, SLOT(slotDataOnStderr()) );
frmListTextReg::connect( process, SIGNAL(processExited()),this, SLOT(slotEndOfProcess()) );
```

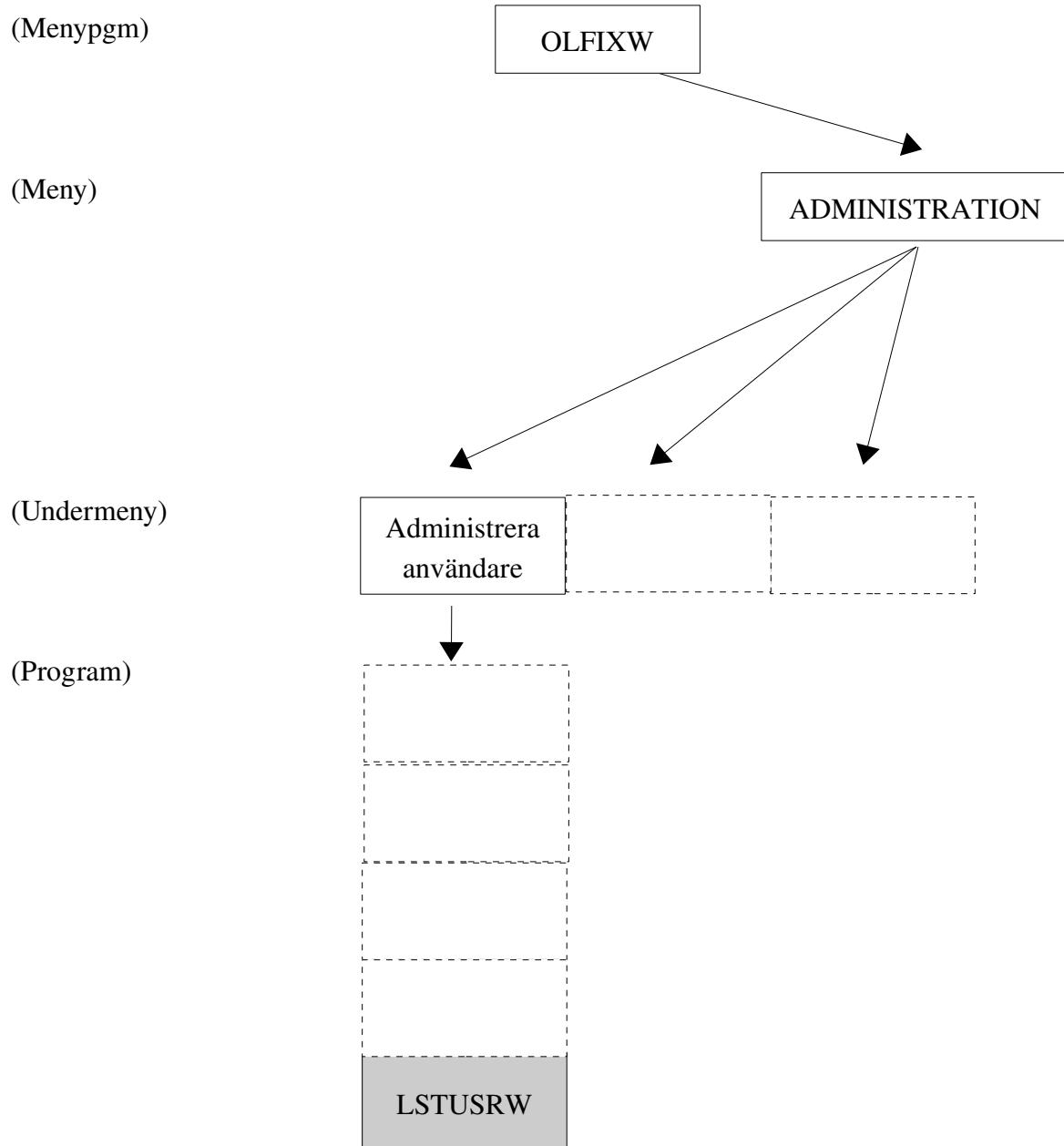
usr = den användare som kör programmet.

### **Behörighetskrav:**

För att kunna köra LSTUSRW behövs behörighet till  
PRGLST  
TXTLST

## LSTUSRW....Lista alla användare

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



X LSTUSRW Lista användare

Userid	Namn	Avd	Grupp
ADMIN	Admin Administratör	IT	Stab
JAN	Jan Olfixsson	Ekonomi	Stab
OLFIX	Olfix Superuser	IT	Stab
TESTARE	Testare Test	Prov	Utv

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.

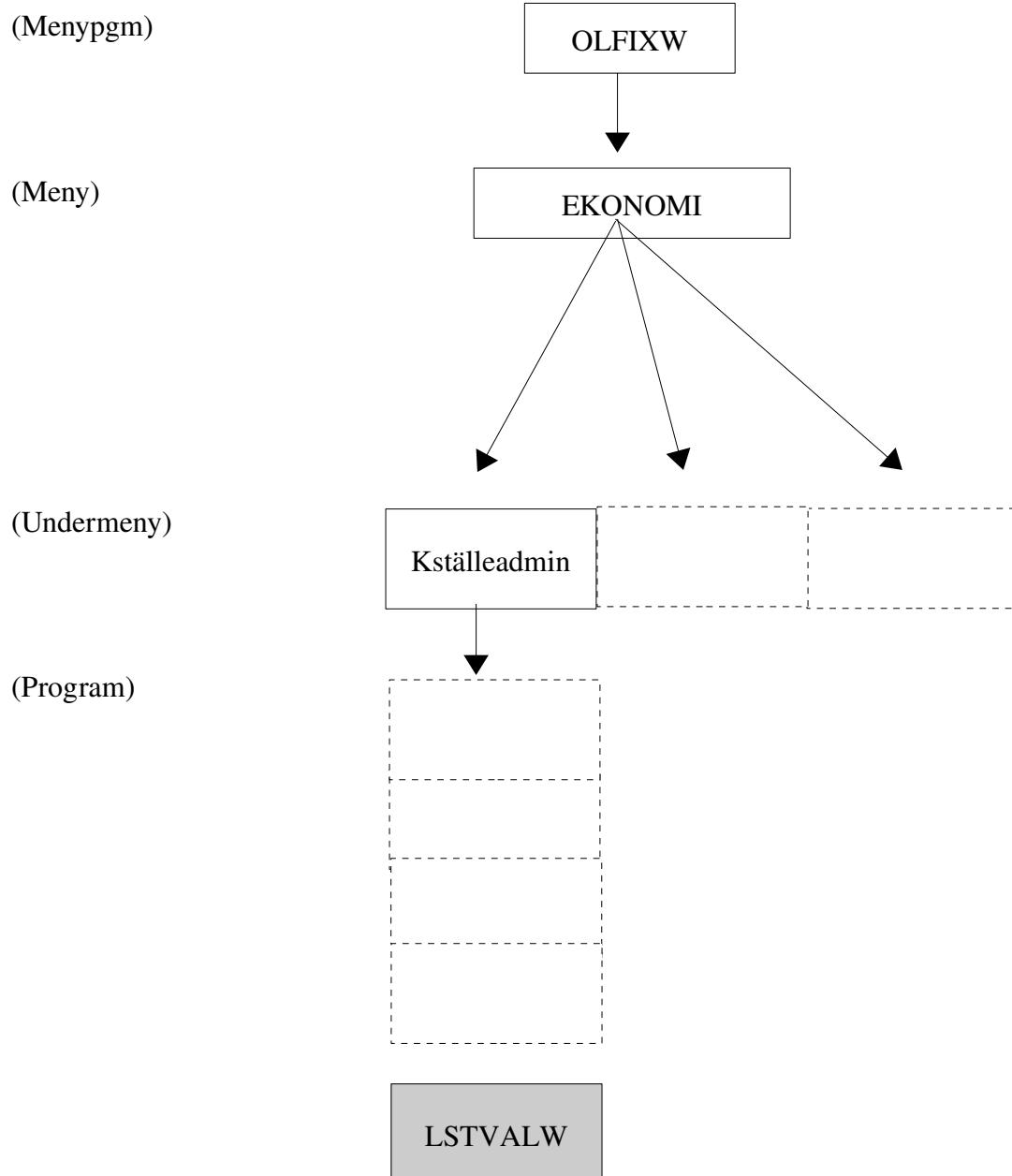
Programmet använder funktionen USERLST för att hämta data från databasen. Anropet av USERLST sker via STYRMAN.

**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.



X-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	Eestland	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	Singanore	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.

OLFIXHLPs dokument finns i katalogen /opt/olfix/doc/helpfiles/usermanual. Dokumenten är i htmlformat. Huvuddokument är UserManual.html som är innehållsförteckning. Därifrån kan man sedan nå de andra dokumenten. Varje dokument är ett eget kapitel som beskriver programmens funktion ur användarens perspektiv.

Programmet anropas från OLFIXW.

OLFIXHLP kan anropas på två sätt;

1. Med endast filnamnet och man hamnar i toppen.

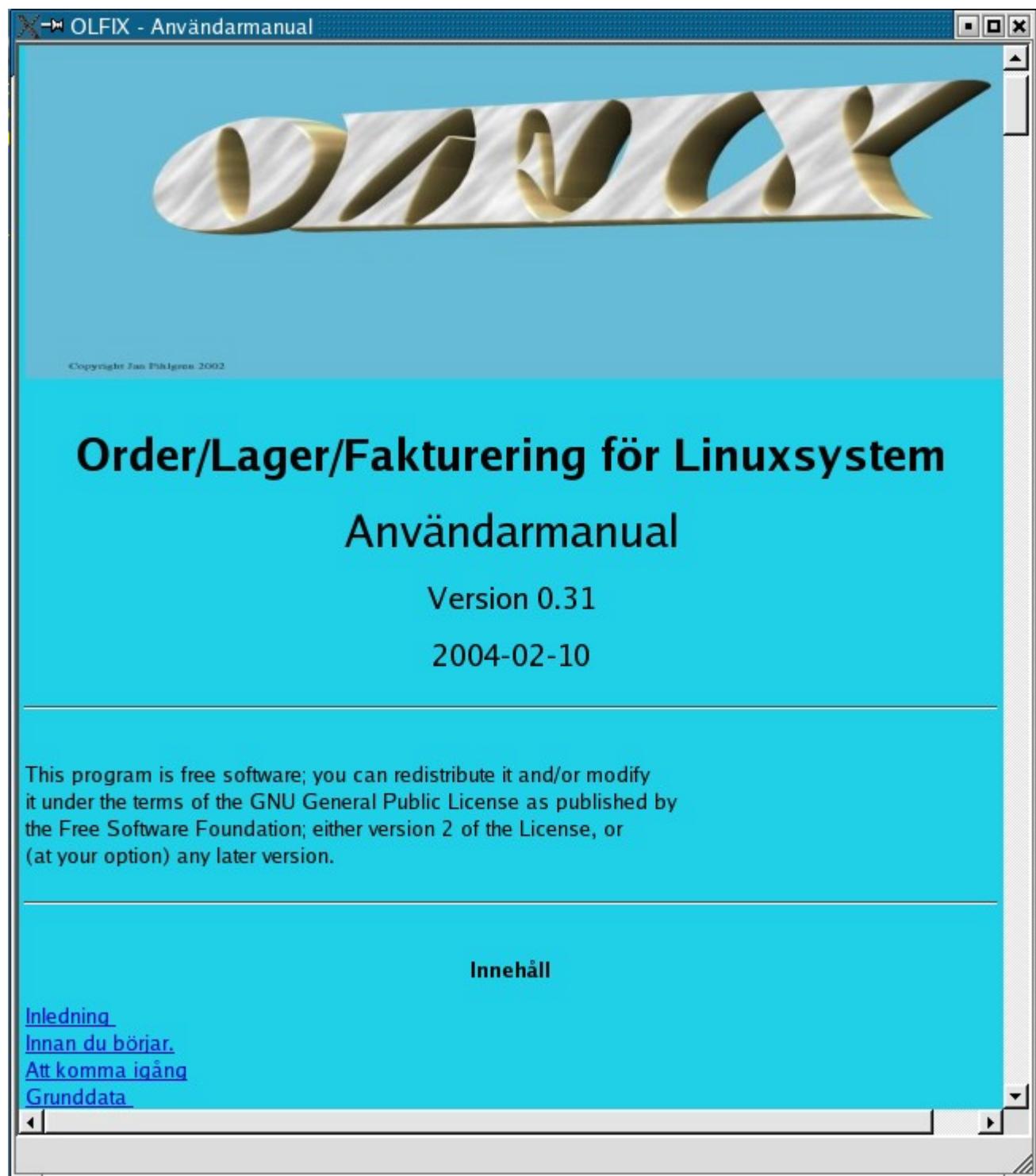
2. Med en parameter, t ex /doc/helpfiles/usermanual/UserManual.html#NYTTBAR.

*/doc/helpfiles/usermanual/UserManual.html* hämtas från resursfilen \$HOME/.olfixrc och värdet från HELPFILE.

I detta fall kommer man direkt till den punkt i hjälppolen som adresseras av #NYTTBAR.

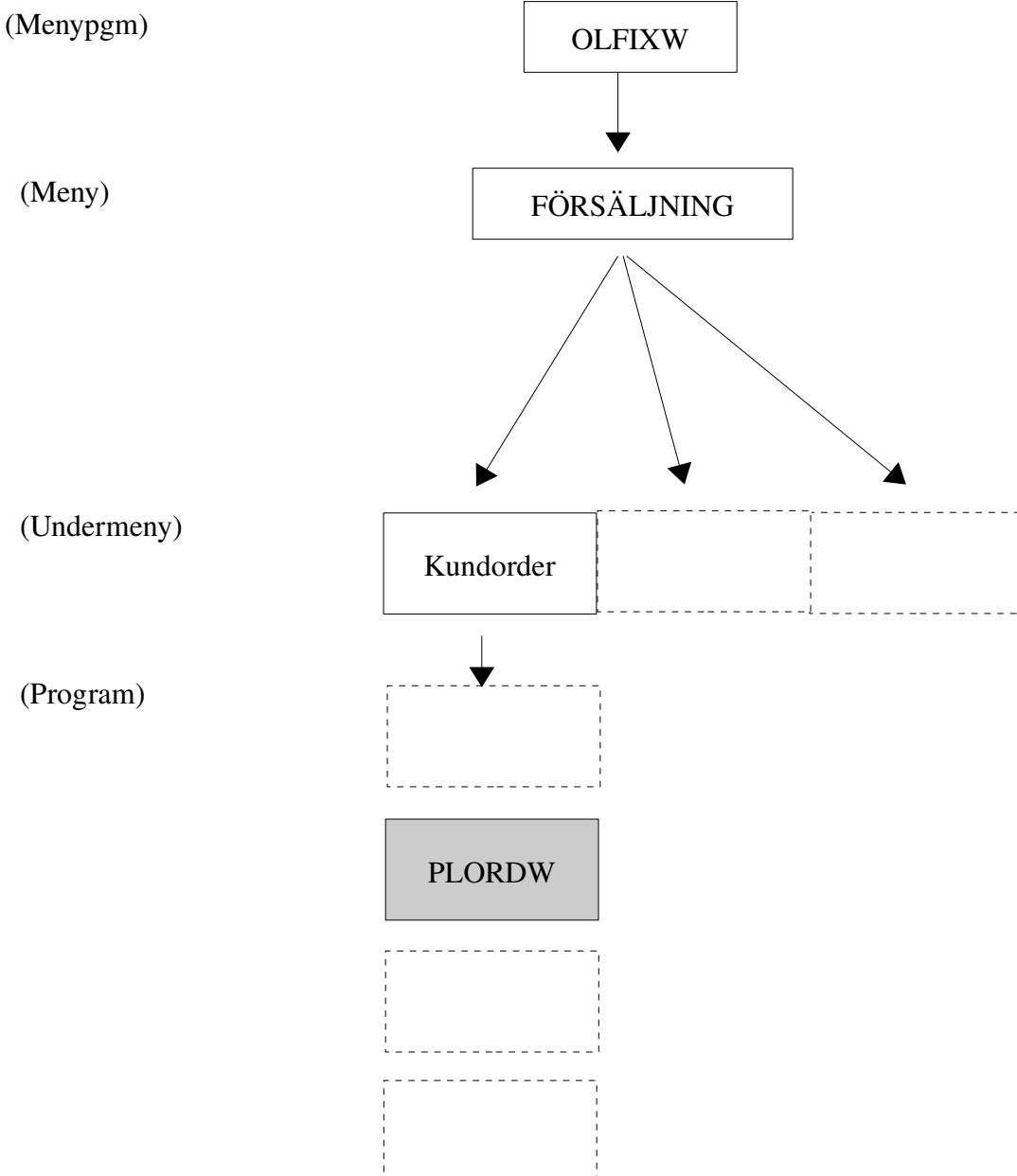
Eftersom OLFIXs program skalar av sökvägen och kompletterar sedan med sökväg och namn till rätt dokument bör man inte ändra filnamnet (UserManual.html). Man kan dock skriva sina egna dokument och ersätta befintliga dokument. Eller man kan också ändra och göra tillägg i dokumenten. Och man kan göra nya dokument och lägga till i UserManul.html.

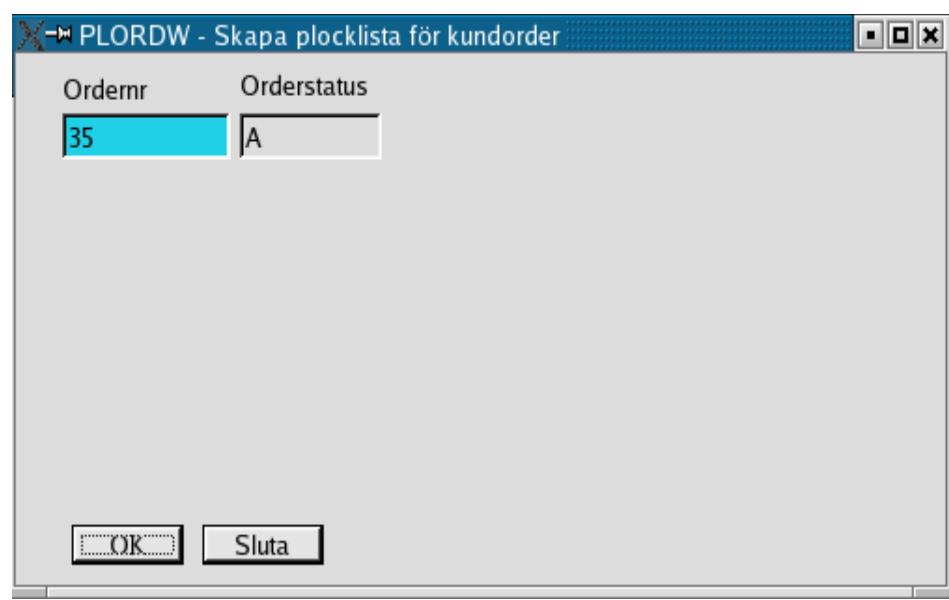
Texterna är en html-fil, **UserManual.html** som ligger i biblioteket /opt/olfix/doc/helpfiles/usermanual.



## PLORDW.....Skriv ut plocklista

PLORDW, ett grafiskt program för att med hjälp av programmet Kugar skriva ut plocklistor för kundordrar. 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 PLORDW.

PLORDW anropar ORDDSP, ORDRDSP, ORDCHK och FTGDSP 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( "ORDDSP" );               // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr ORDDSP kundordernr
./STYRMAN usr ORDRDSP kundordernr
./STYRMAN usr ORDCHK val kundordernr
./STYRMAN usr FTGDSP posttyp
```

val kan vara 1 eller 2

1 = check ordernr

2 = check orderstatus (A=I arbete,B=Ordern är avslutad, ska plockas bort)

usr är den inloggades userid.

### Behörighetskrav:

För att kunna köra PLORDW behövs behörighet till

PRGLST

ORDDSP

ORDRDSP

ORDCHK

FTGDSP

Exempel på plocklista.

The screenshot shows a window titled "Kugar" with a menu bar: Arkiv, Kör, Inställningar, Hjälp. Below the menu is a toolbar with icons for file operations. The main area displays a picking list for "OLFIX AB".

**Plocklista**

Datum	Ordernummer	Sida
2005-03-29	35	1

Leveransadress Fakturaadress

**OLFIX AB**

Storgatan 1  
199 99 STORSTAD  
Sverige

Betalningsvillkor

Leveransvillkor

Valuta Leveranssätt

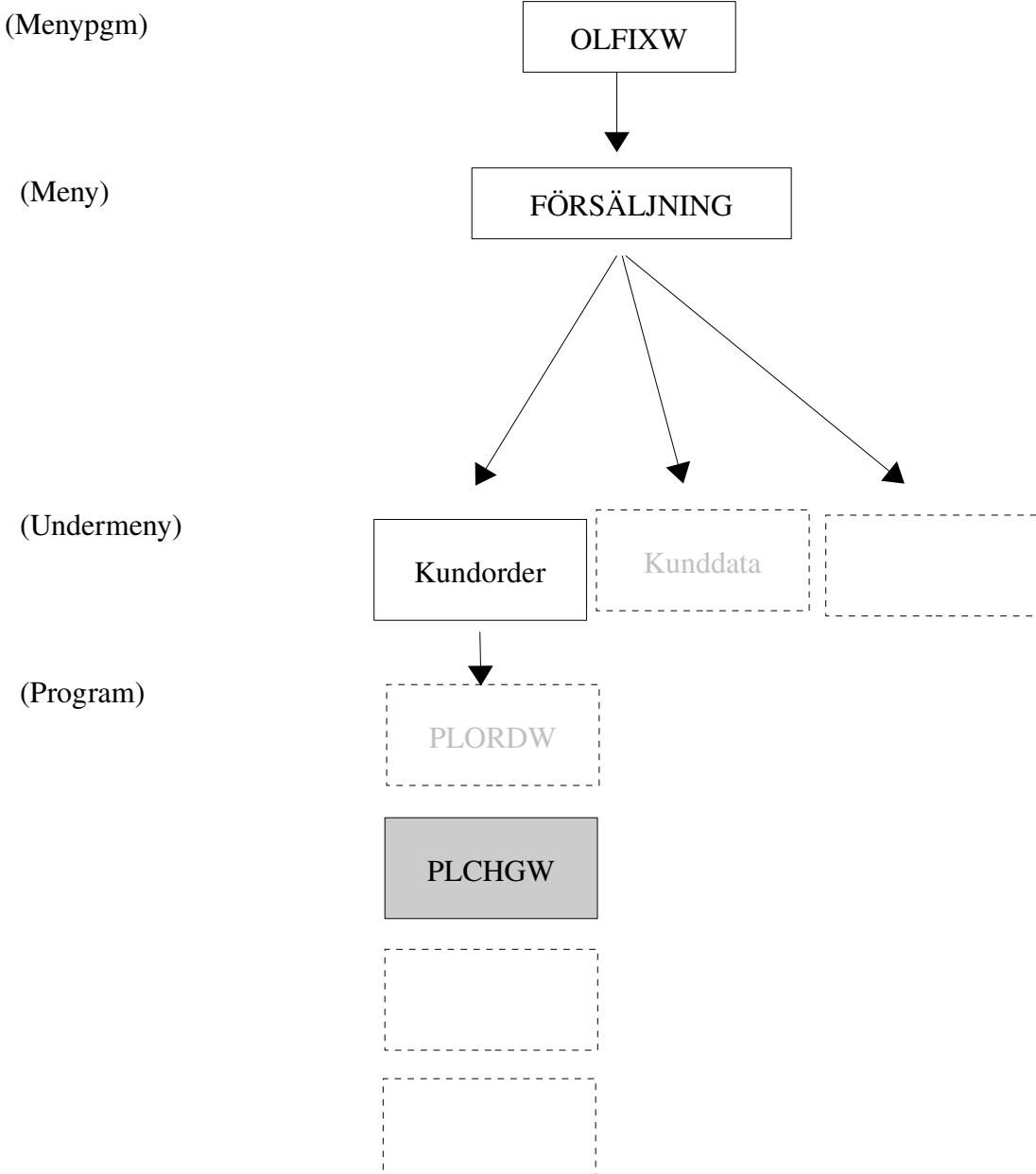
E referens Vår referens  
U Persdotter Jan Pihlgren

**Godsmärke**  
Godsmärke

Artikelnr Pos	Benämning	Antal	Sort	Leverans vecka
10	1000-1003 Linux På egen hand	1		5134
20	1000-1007 The C Programming lang.Facit	1		5134
30	1000-1011 Professional Linux Programming	1		5134

## PLCHGW..... Pricka av plocklista

PLCHGW, ett grafiskt program för att pricka av plocklistor för kundorder.  
Programmet plockar upp userid från environment.



PLCHGW - Pricka av plocklista för kundorder

Ordernr	Orderstatus	Hjälp
000	status	

Radnr	Artikelnummer	Benämning	Leveransvecka	Beställt antal	Antal att leverera	Plockat antal	Godkänn rad
000						0	<input type="button" value="Ja"/> <input type="button" value="Nej"/>

Radnr	Artikelnr	Benämning	Leveransvecka	Beställt antal	Att leverera	Plockat antal

Plockningen klar

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 PLCHGW.

PLCHGW anropar ORDDSP, ORDRDSP, ORDCHK, ORDRUPD, AR2UPD, PICKADD och TRHDADD 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( "ORDDSP" );              // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr ORDDSP kundordernr
./STYRMAN usr ORDRDSP kundordernr
./STYRMAN usr ORDCHK val kundordernr
./STYRMAN usr ORDRUPD ordernum radnum levantal restantal
./STYRMAN usr AR2UPD val artikelnr data
./STYRMAN usr TRHDADD tidpunkt usr trnsdata
./STYRMAN usr PICKADD plockdata
```

usr är den inloggades userid.

Tabeller som uppdateras:

ORDERRADREG

ARTIKELREG

PLOCKLISTEREG

TRHD

### Behörighetskrav:

För att kunna köra PLCHGW behövs behörighet till

PRGLST

ORDDSP

ORDRDSP

ORDCHK

ORDRUPD

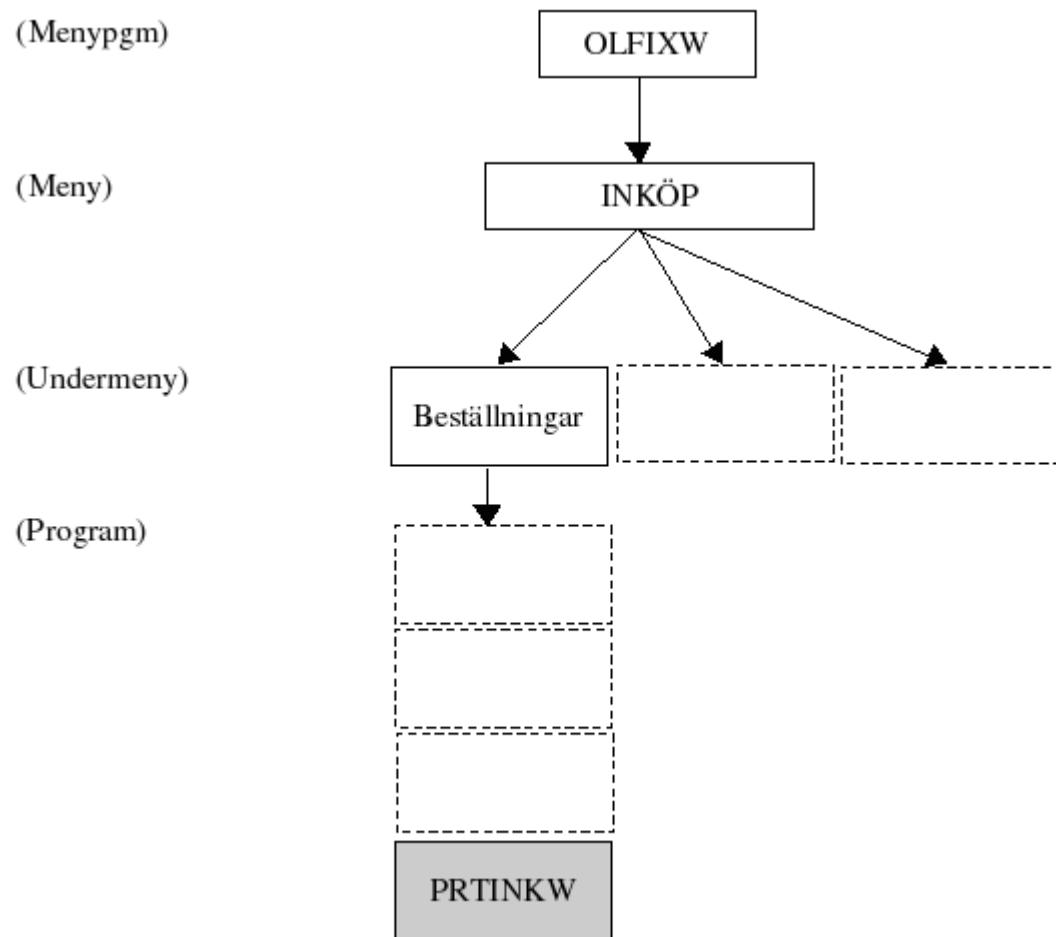
AR2UPD

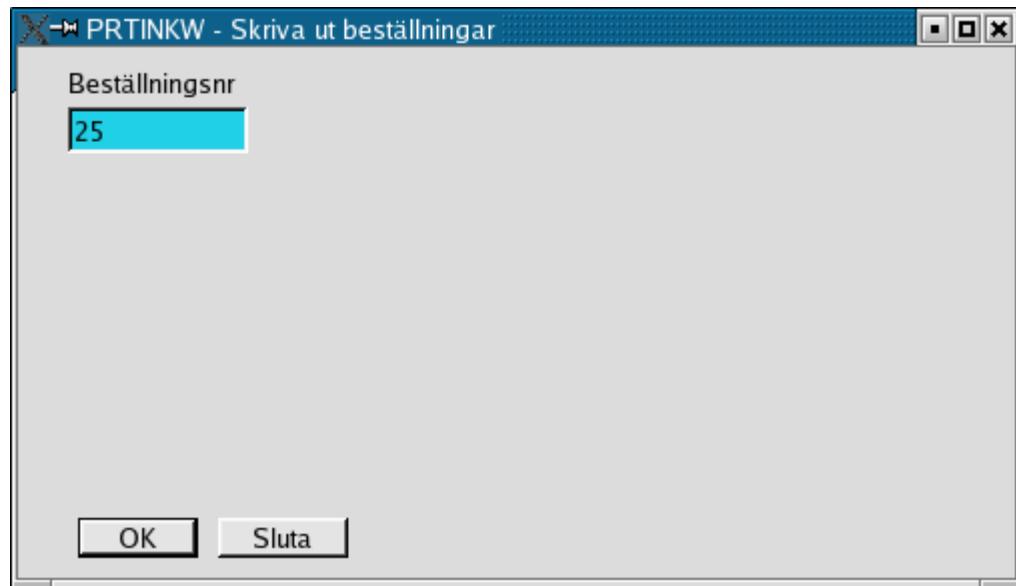
TRHDADD

PICKADD

## 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	Beställningsnr	Sida
2004-02-05	26	1

Leverantörsnr  
123

Leveransadress  
**PROGRAM AB**  
Verktygsgatan 11  
199 97 PROGSTAD

Leverantr AB  
Postgatan 33  
199 99 DATABY  
SVERIGE

Kommentar  
Detta är en kommentar

Betalningsvillkor  
20 dagar netto

Telefon: 08-59112449

Leveransvillkor  
EXW

Fax: 09-112233

Valuta  
SEK

Leveranssätt  
Schenker kundnr:11105232

Ereferens  
Per Josefsson

Vår referens  
Jan Pihlgren

Godsmärke  
KALLE PALL 26

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

Ordererkanande räknas inom 3 arbetsdagar (om ej redan  
bekräts)

Ange alltid vårt artikelnummer på fljesedel 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 Bestellning.kud i XML-format.

PRTINKW anropar sedan Kugar.

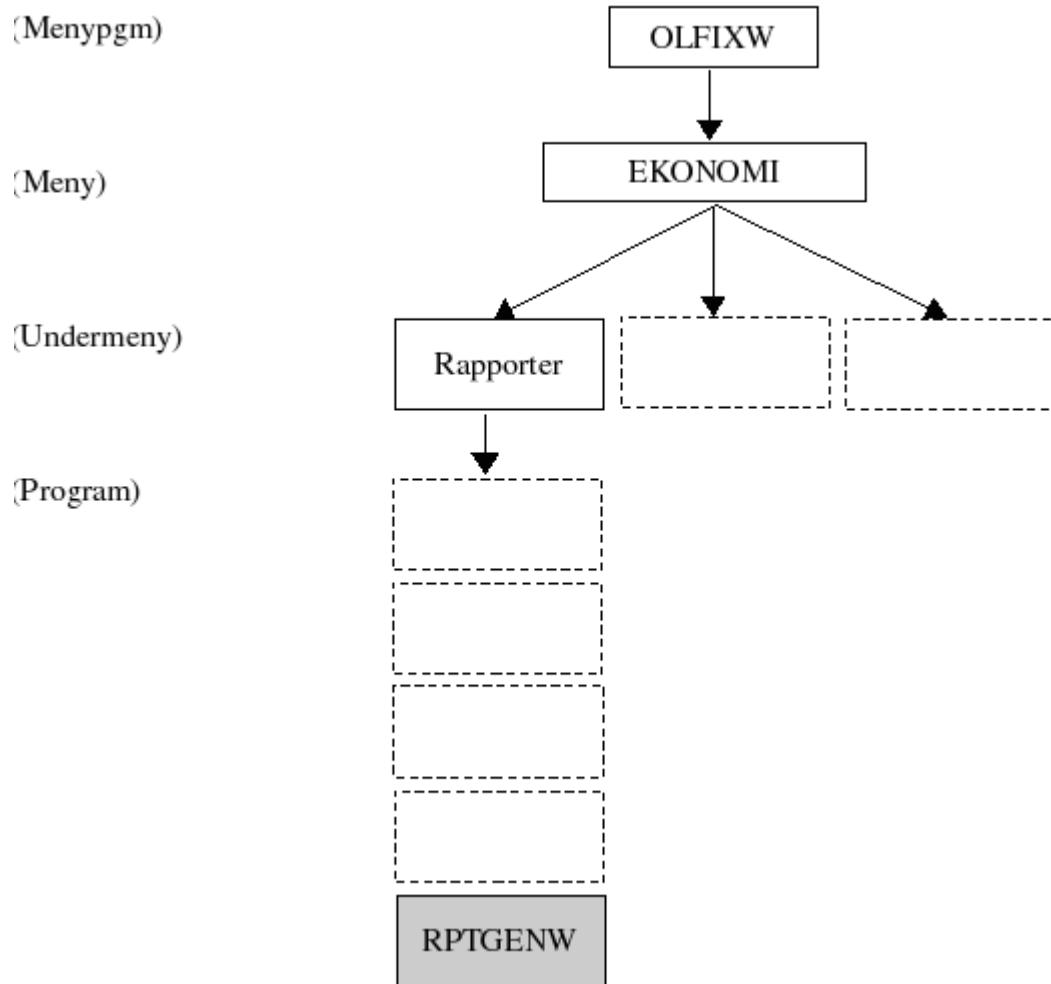
```
if (kugarversion<"1.2.92"){
    kommando="kugar -d /tmp/Bestellning.kud -r "+reportpath+"Bestellning.kut";
    system(kommando);
} else{
    system("kugar /tmp/Bestellning.kud");
}
```

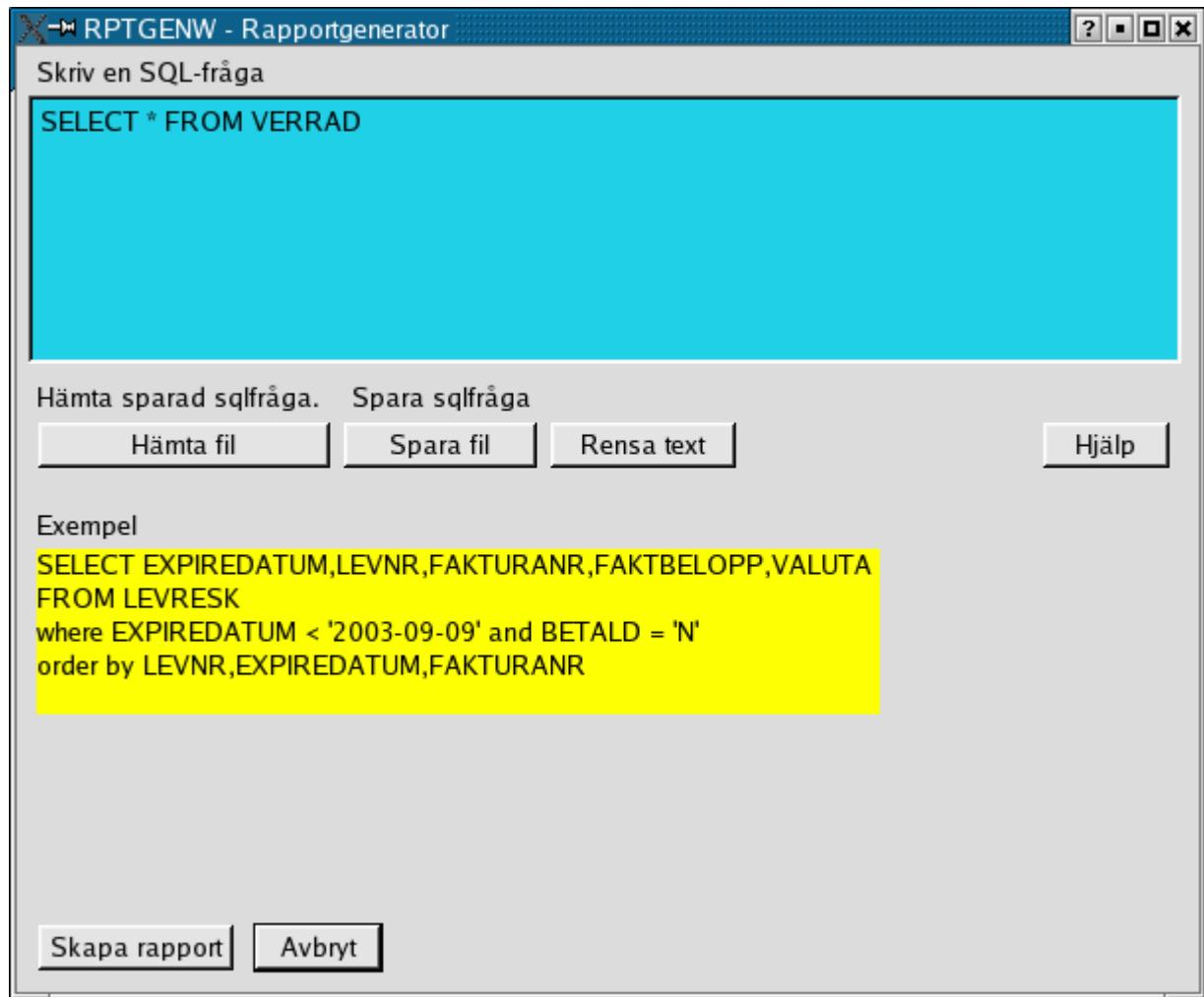
### Behörighetskrav:

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

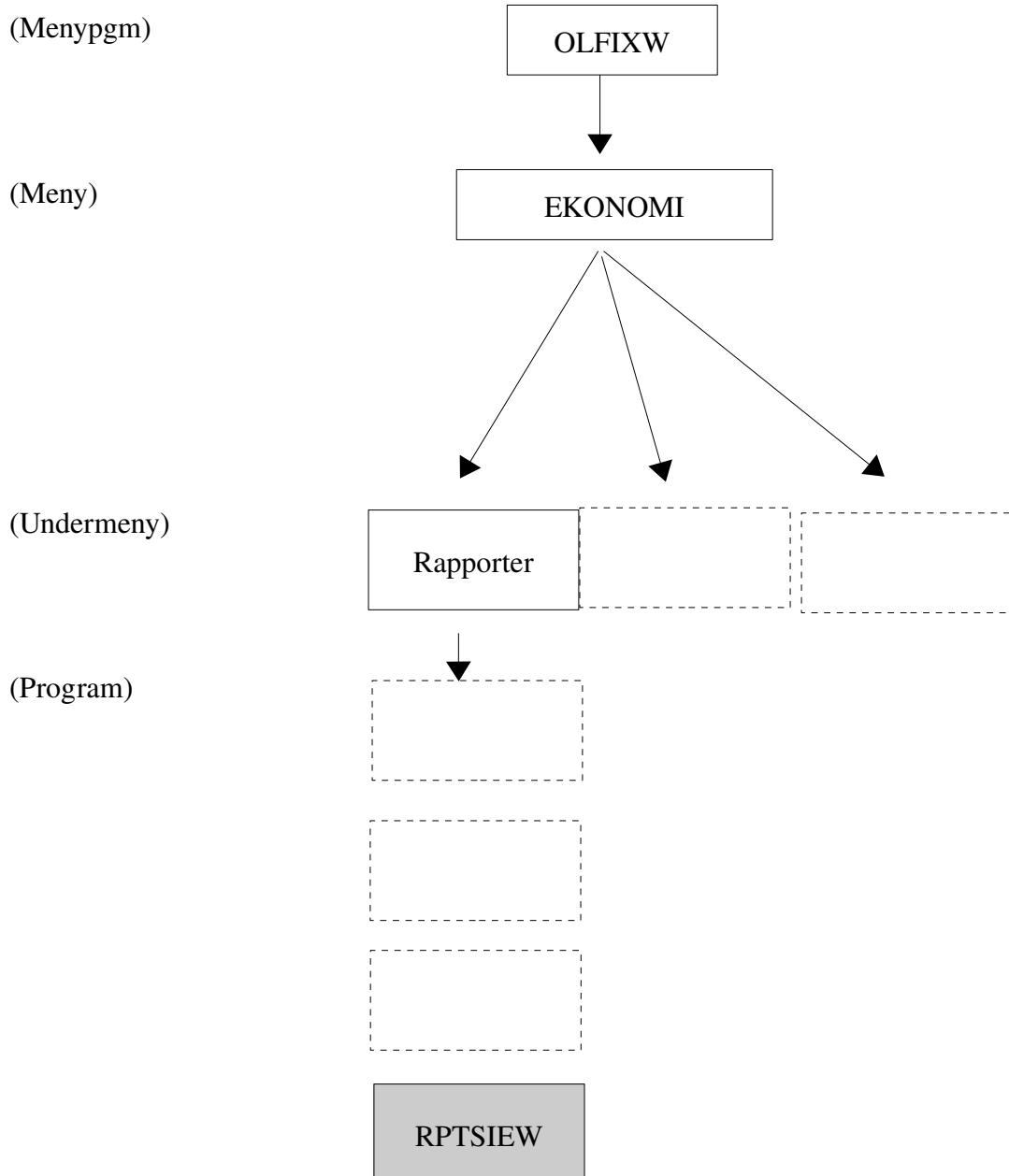
PRGLST

RPTCRE

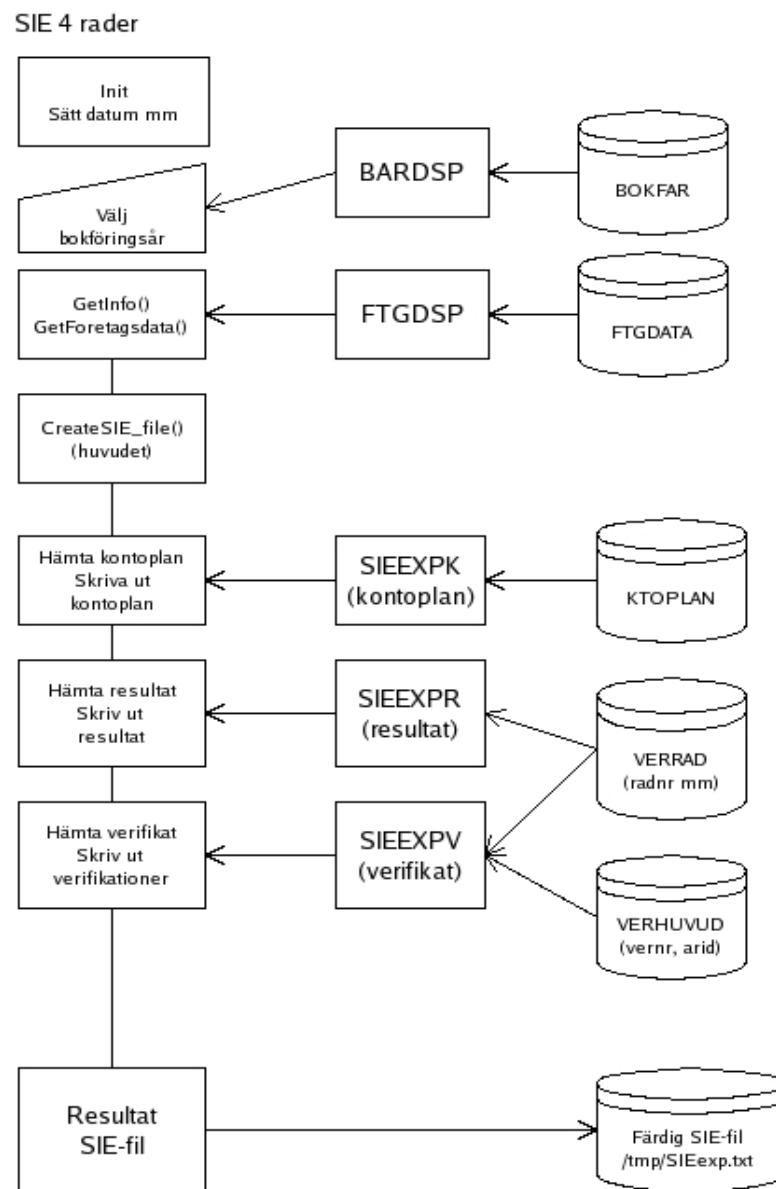
## RPTSIEW.....Skapa SIE-fil

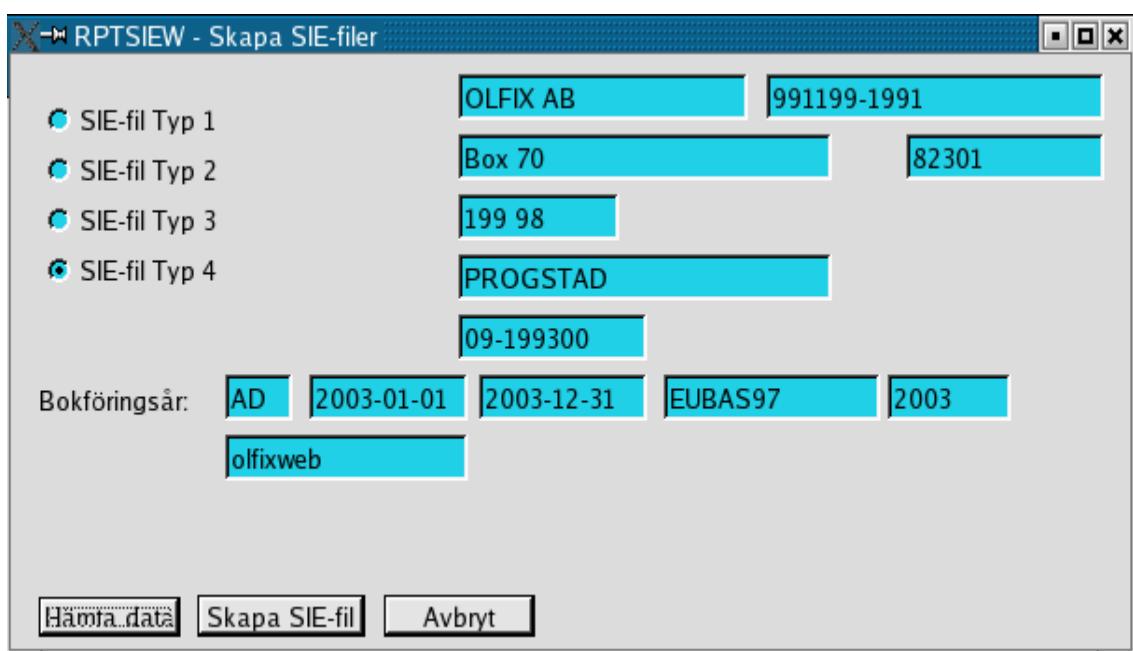
RPTSIEW, ett grafiskt program för att skapa en s k SIE-fil. SIE-filen kan sedan hämtas i /tmp mappen, namn = SIEtyp4.txt.

Programmet plockar upp userid från environment.



## Flödesschema





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 RPTSIEW.

RPTSIEW anropar via STYRMAN funktionerna FTGDSP, BARDSP, SIEEXPK, SIEEXPR, SIEEXPV.

```
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( "SIEEXPK" );              // OLFIX funktion
process->addArgument(arid);

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

process = new QProcess();
process->addArgument( "./STYRMAN" );
process->addArgument(usr);                      // user OLFIX
process->addArgument( "SIEEXPV" );              // OLFIX funktion
process->addArgument(arid);
process->addArgument("0");                      // serie
```

Detta blir:

```
./STYRMAN usr SIEEXPK AD
./STYRMAN usr SIEEXPR AD
samt
./STYRMAN usr SIEEXPV AD 0
```

usr är den inloggades userid.

RPTSIEW skapar filen /tmp/SIETyp4.txt.

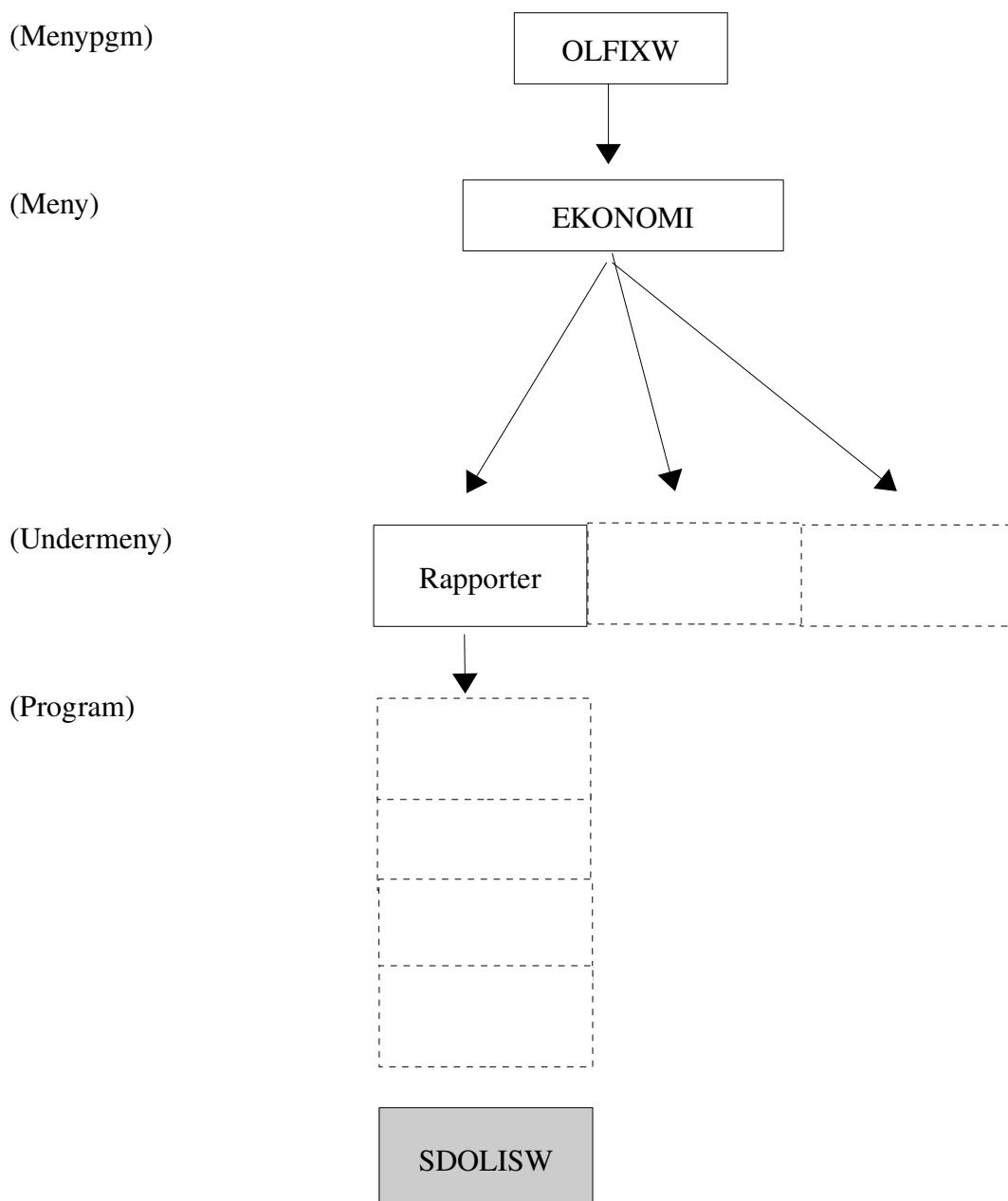
### Behörighetskrav:

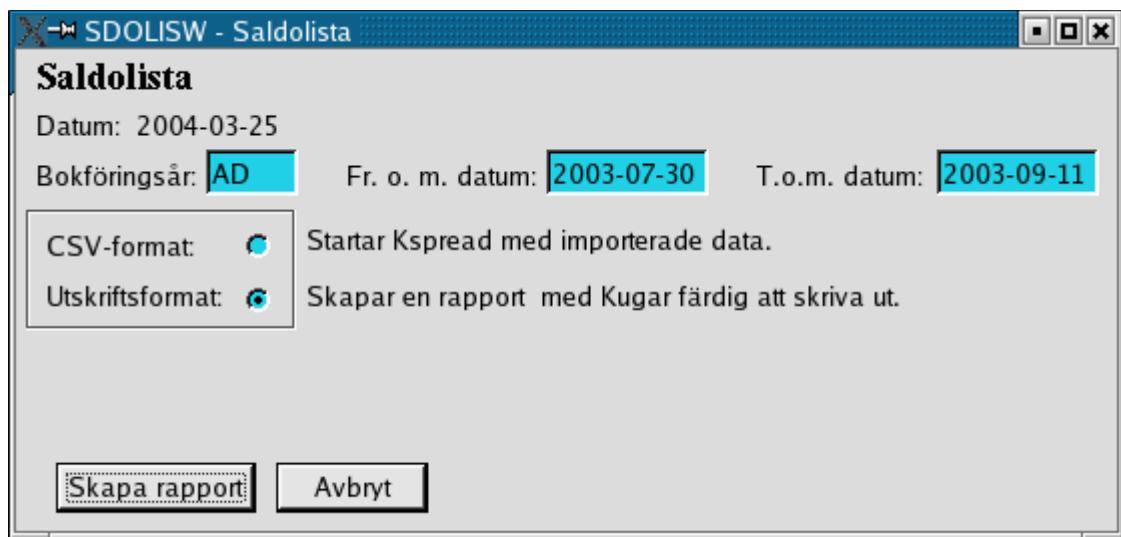
För att kunna köra RPTSIEW behövs behörighet till  
PRGLST  
BARDSP  
FTGDSP  
SIEEXPK  
SIEEXPR  
SIEEXPV

## 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.





The screenshot shows a Kspread spreadsheet window with the title bar "file:/home/jan/Utveckling/OLFIX/doc/SaldolistaKspreadExempel.ksp - Kspread". The menu bar includes Arkiv, Redigera, Visa, Infoga, Format, Data, Verktyg, Inställningar, and Hjälp. The toolbar has various icons for file operations and data manipulation.

The spreadsheet displays a balance sheet with columns A, B, C, D, and E. The rows show financial accounts and their corresponding values. The formula bar shows  $\Sigma f(x)$  and SUM. The font is set to Sans and size 10. The current cell is 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					

At the bottom left is the label "Arbetsblad 1/" and at the bottom right is "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 */opt/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
totaldebet 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" delsumdebet="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" delsumdebet="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" delsumdebet="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" delsumdebet="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" delsumdebet="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" delsumdebet="550.00" delsumkredit="0.00" delsumutgsaldo="550.00"/>
<Blankrad level="3" blank=" "/>
<Totalsumma level="4" totaldebet="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

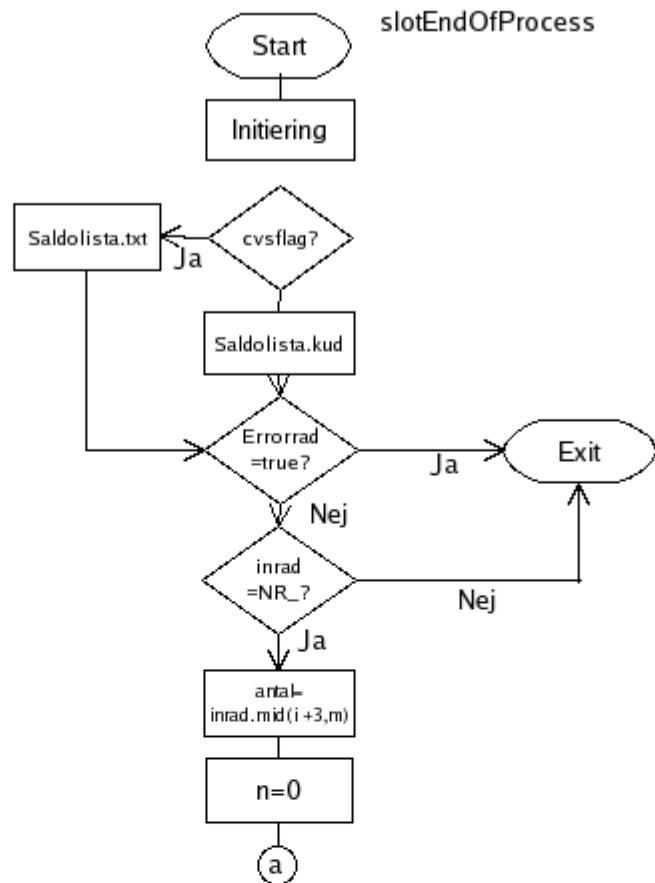
FTGDSP

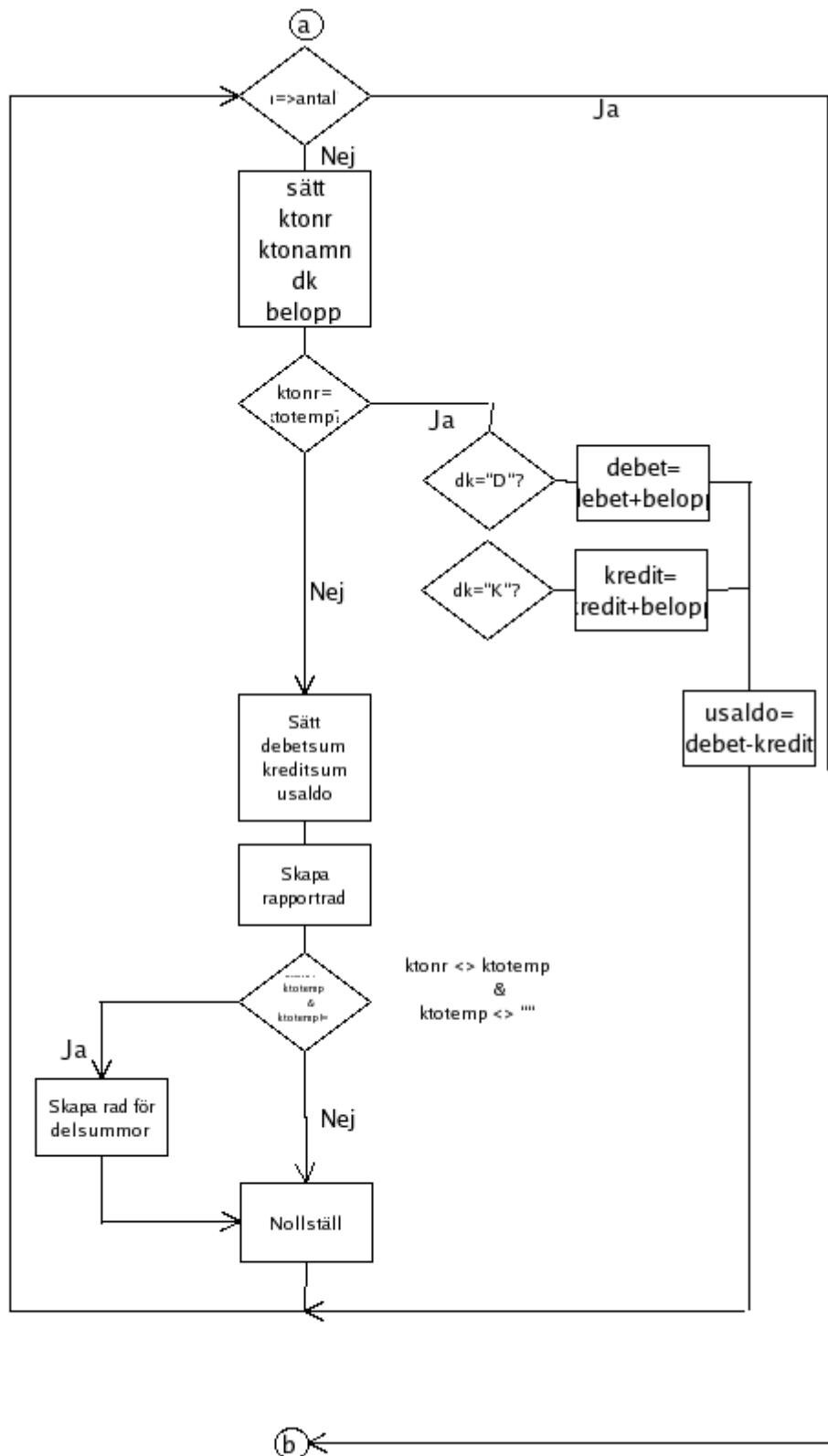
KTORPT

VERHDSP

PRTAPI

## Flödesschema



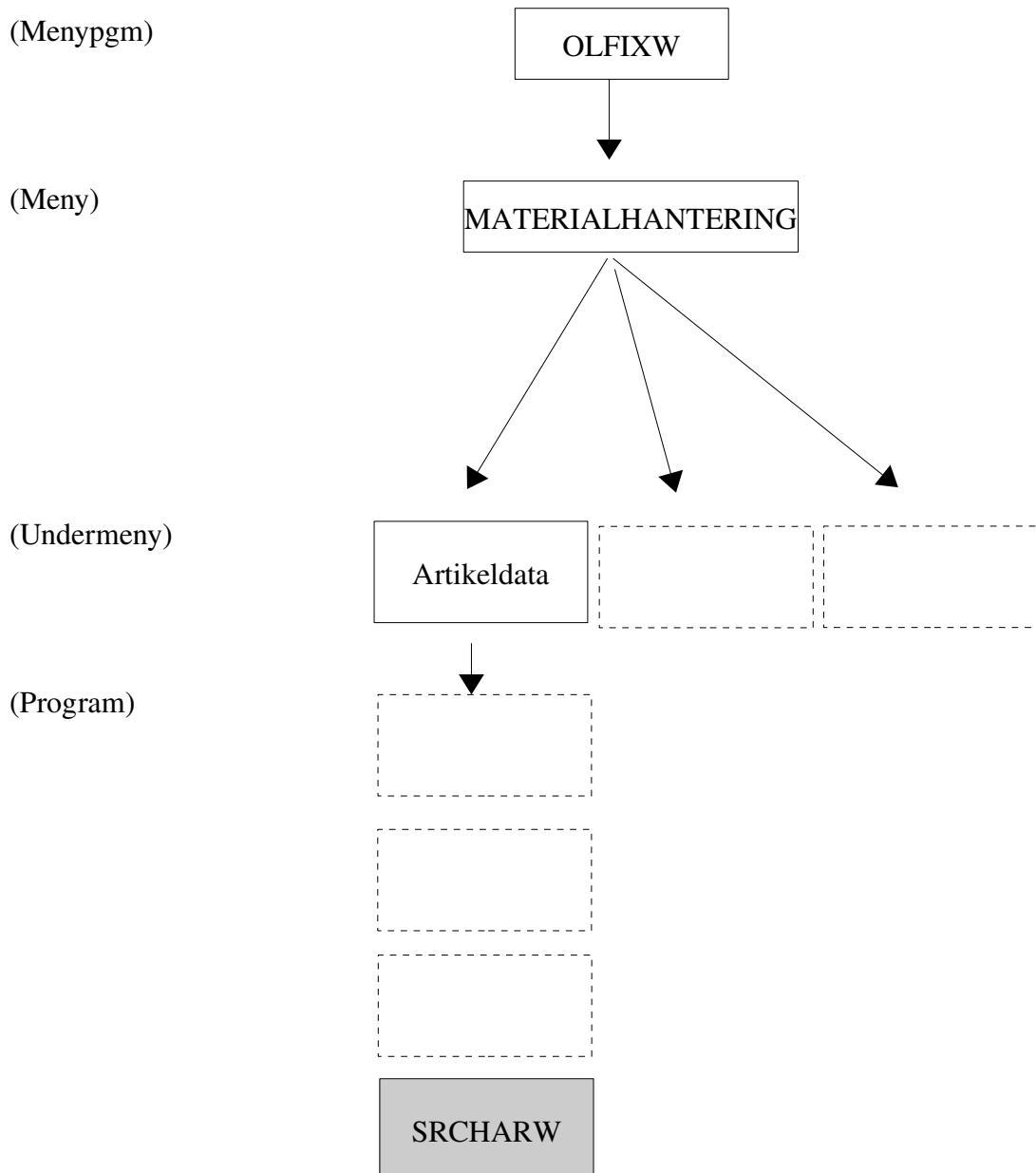


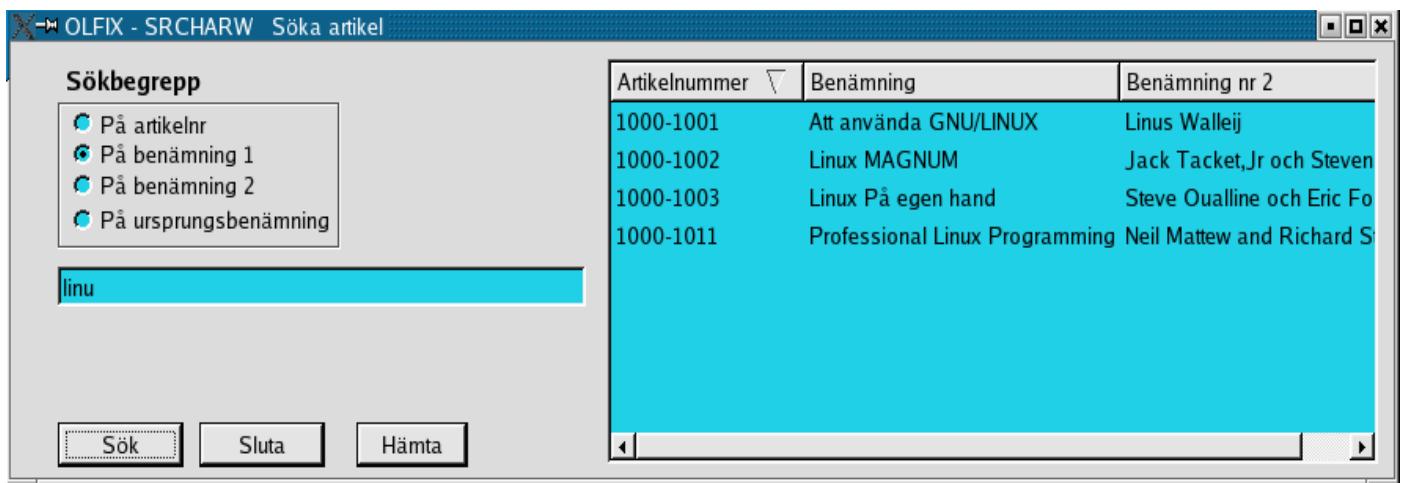
## SRCHARW..... Söka artiklar

SRCHKUW, ett grafiskt program för att söka artiklar. Sökning kan göras på artikelnummer, benämning 1, benämning 2 och ursprungsbenämning.

När man hittat önskat artikelnummer kan man direkt klicka fram artikeldata (DSPARW).

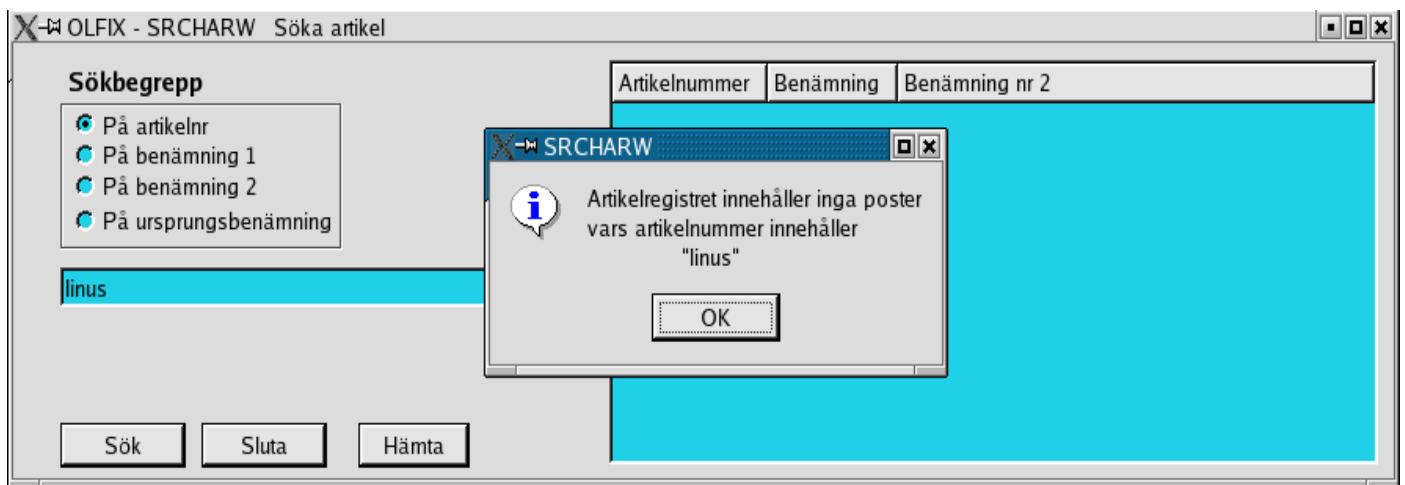
Programmet plockar upp userid från environment.





Genom att klicka på aktuellt artikelnummer (rad för aktuellt artikelnummer) kopieras artikelnumret till sökfältet och sökbegreppet sätts till "På artikelnummer".

Därefter kan man klicka på "Hämta" och programmet DSPARW kommer upp med artikelnumret ifyllt. När programmet startas från ADDORDW kan man genom att dubbelklicka på ett Artikelnummer överföra artikelnumret till ADDORDWs Artikelnummerfält.



I fall sökresultatet blir noll så presenteras en resultatruta som meddelar att det inte finns någon artikel med detta begrepp.

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 SRCHARW.

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

SRCHARW anropar ARSRCH via STYRMAN med parametrarna soekbegrepp och soekord.

Sökbegreppen är

1 för artikelnummer

2 för benämning 1

3 för benämning 2

4 för ursprungsbenämning

```
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( "ARSRCH" );   // OLFIX funktion
process->addArgument(begrepp);
process->addArgument(soekord);
```

Detta blir:

```
./STYRMAN usr ARSRCH begrepp soekord
```

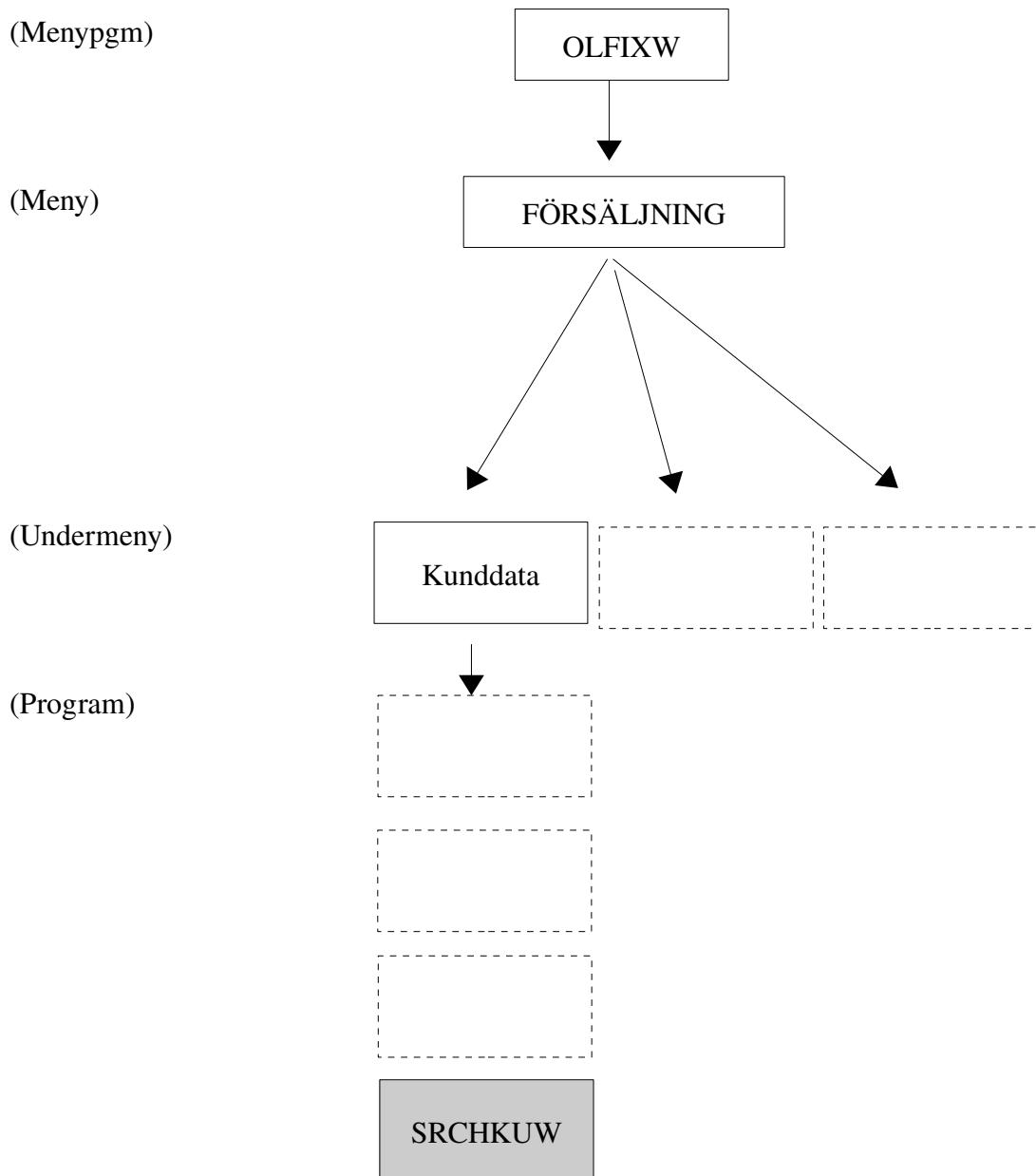
### Behörighetskrav:

För att kunna köra SRCHARW behövs behörighet till  
PRGLST  
ARSRCH  
DSPARW

## SRCHKUW.....Söka kunder

SRCHKUW, ett grafiskt program för att söka kunder. Sökning kan göras på namn, postnr, telefonnr och postadress(ort).

Programmet plockar upp userid från environment.



## SRCHKUW - Söka efter kund

- Söka på namn
- Söka på postnummer
- Söka på telefonnummer
- Söka på postadress (ort)

Kundnr	Namn	Postnr	Ort (Postadress)	Tfnr
4375	Lilla Kunden Eftr AB	199 09	SMÅSTAD	09-390000
4377	Nya Kund AB	199 97	LILLEBY	09-999190
4379	Nya Småkund AB	199 02	SMÅSTAD	09-129990
4378	Nya Storkund AB	100 01	LYXBY	09-109990

Sökord:

KUND

Sök

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 SRCHKUW.

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

SRCHKUW anropar KUSRCH via STYRMAN med parametrarna soekbegrepp och soekord.

```
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("KUSRCH");    // OLFIX funktion
```

Detta blir:

```
./STYRMAN usr KUSRCH soekbegrepp soekord
```

#### **Behörighetskrav:**

För att kunna köra SRCHKUW behövs behörighet till  
PRGLST  
KUSRCH

## **Konsolprogram i OLFIX**

Konsolprogrammen är icke kompletta och innehåller ingen felhantering.

ADMIN Konsolprogram.

BOKF Konsolprogram för bokföring.

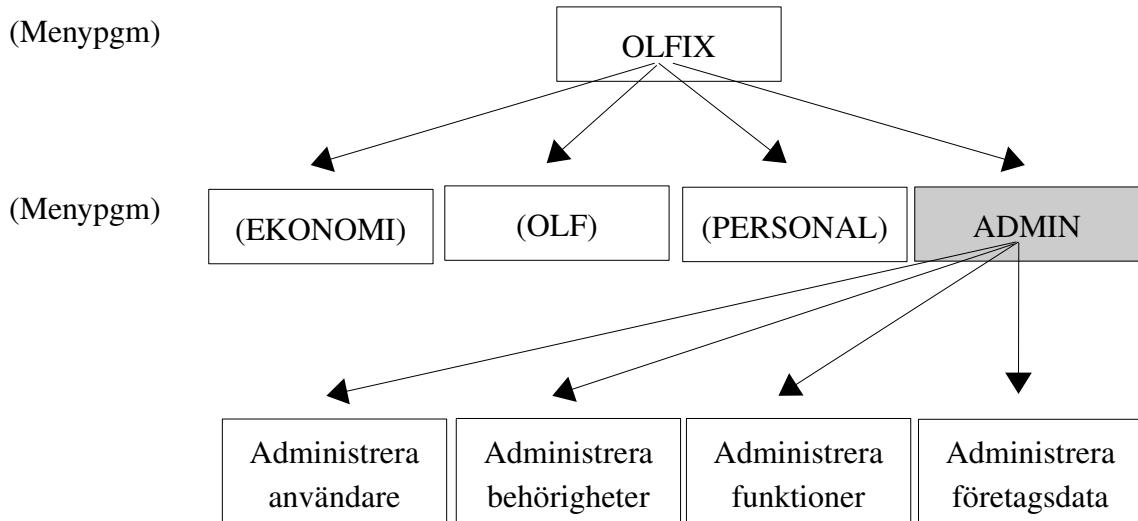
FORADM Konsolprogram.

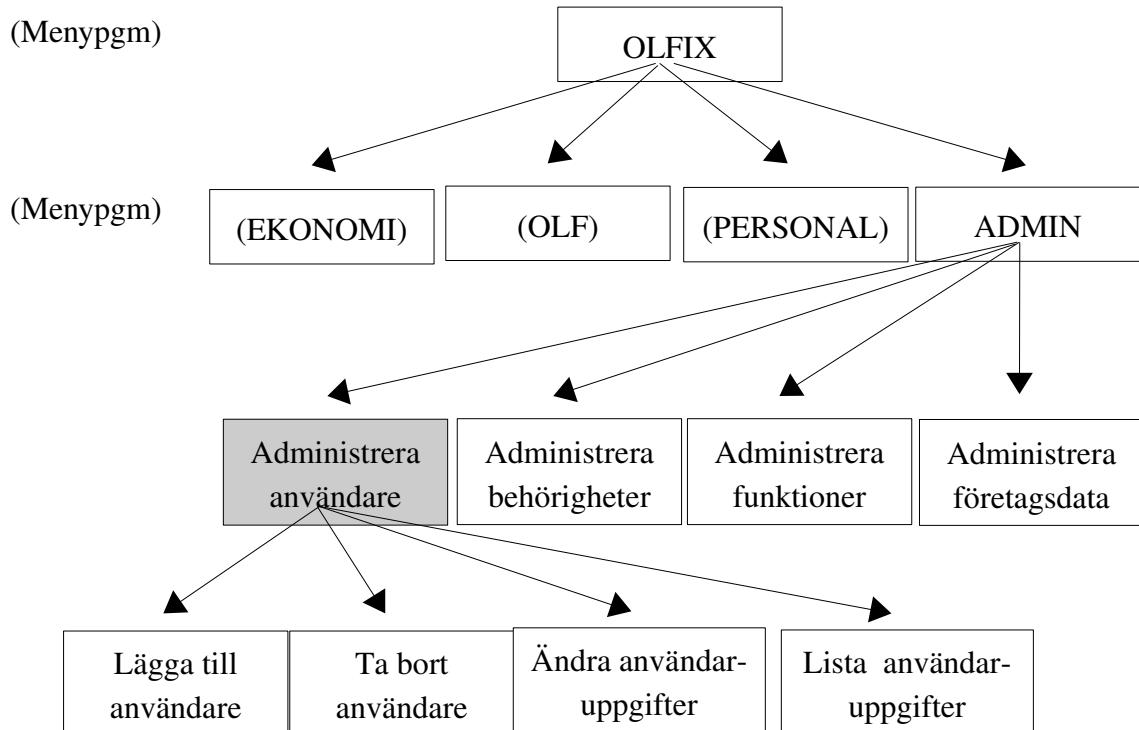
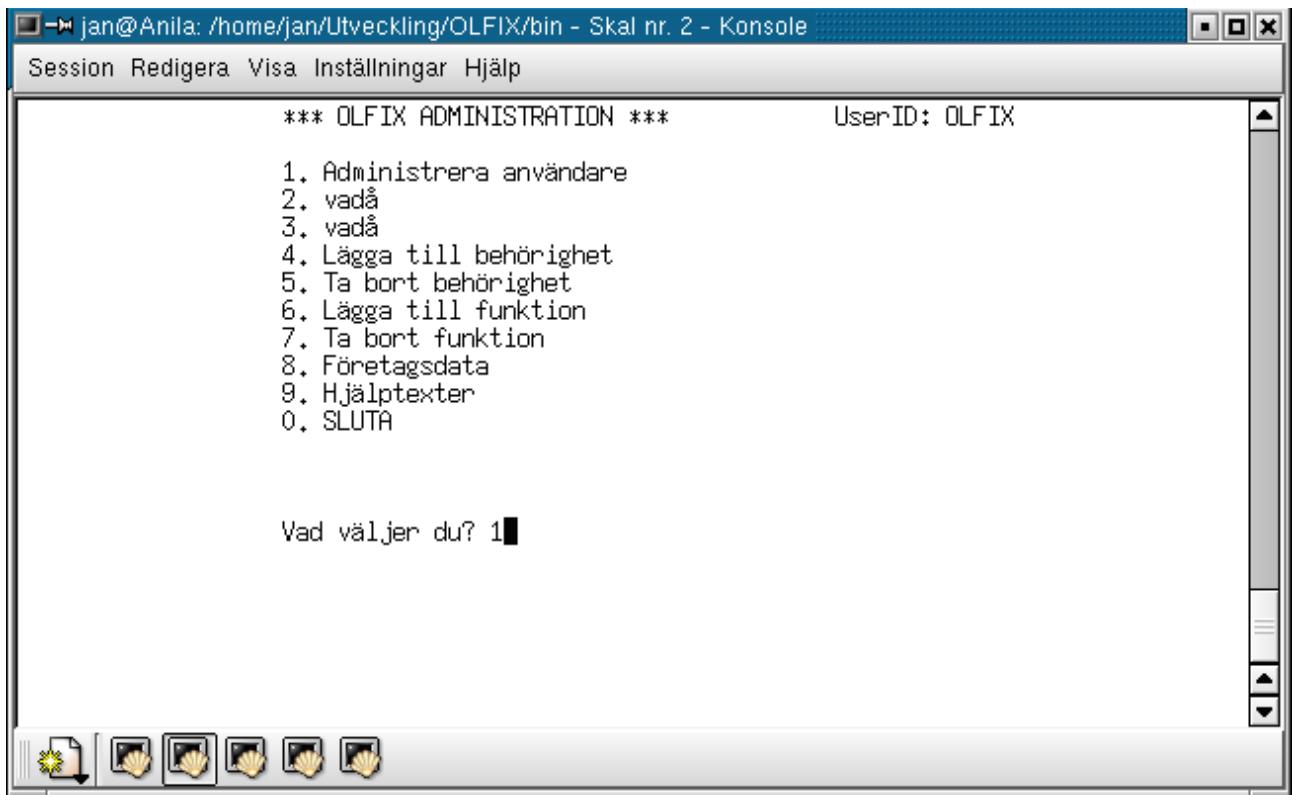
OLFIX Huvudprogram, konsolprogram.

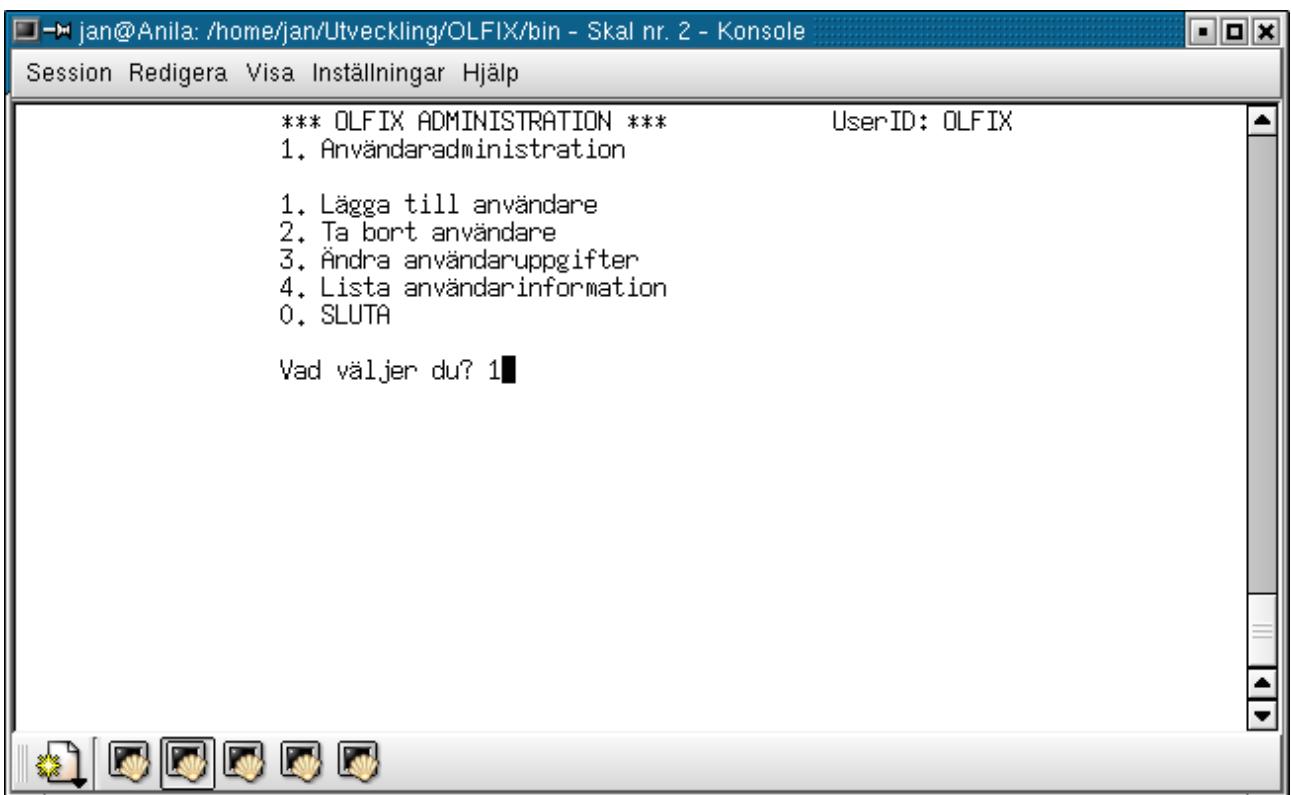
REDOV Konsolprogram.

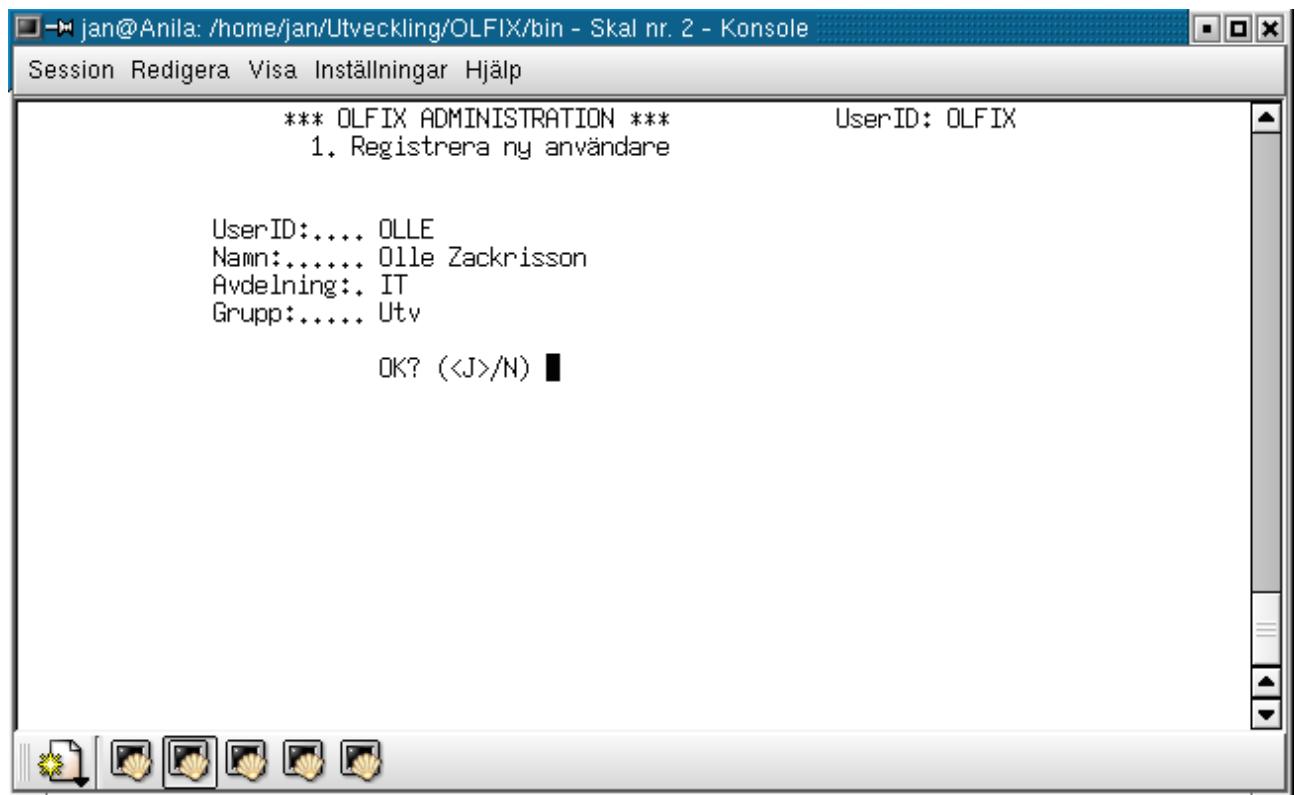
## ADMIN(program)

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









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

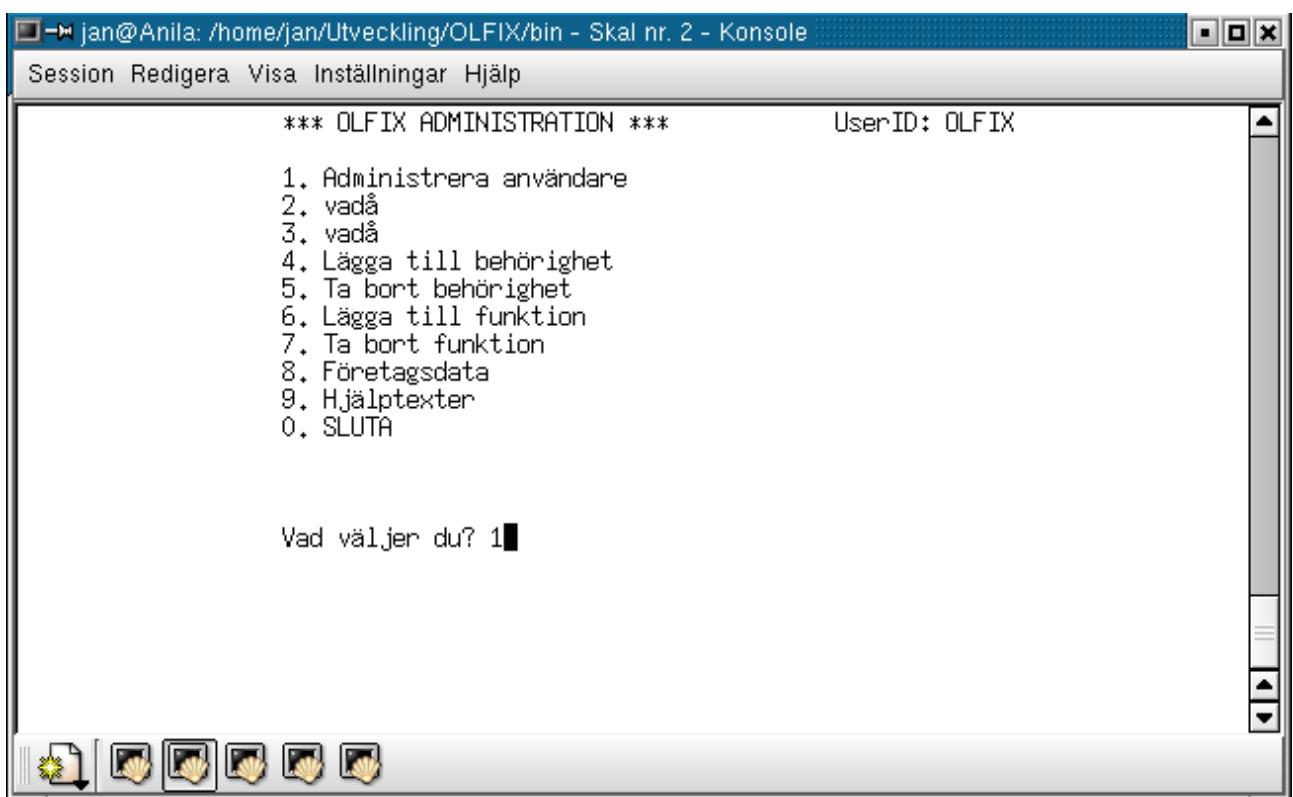
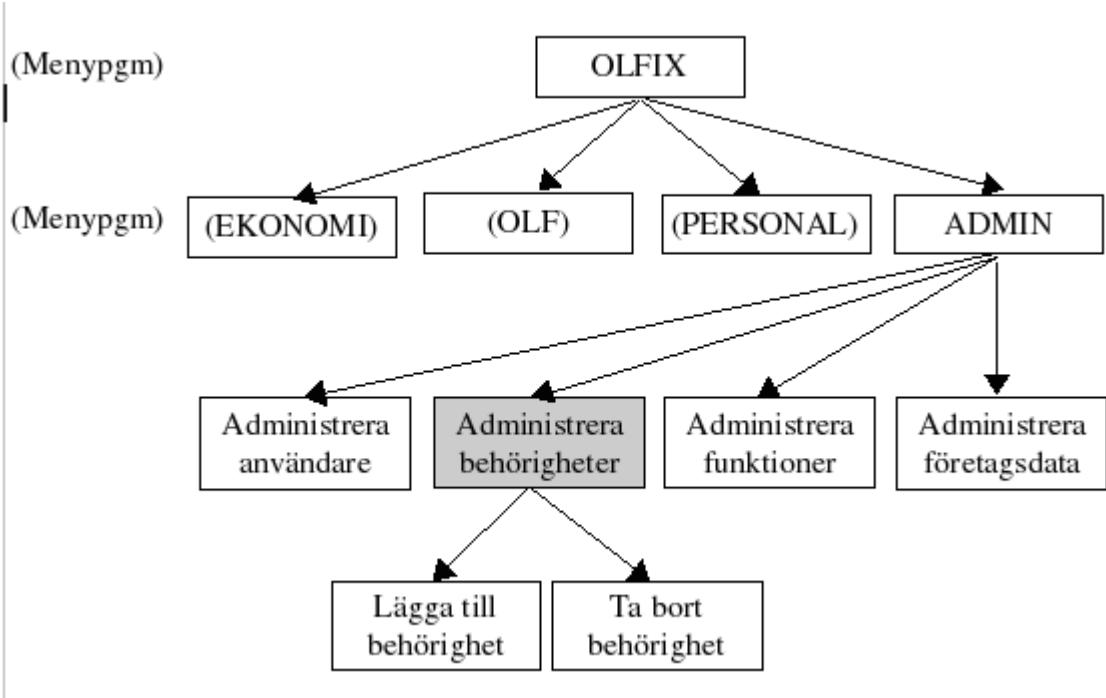
UserID: JAPI

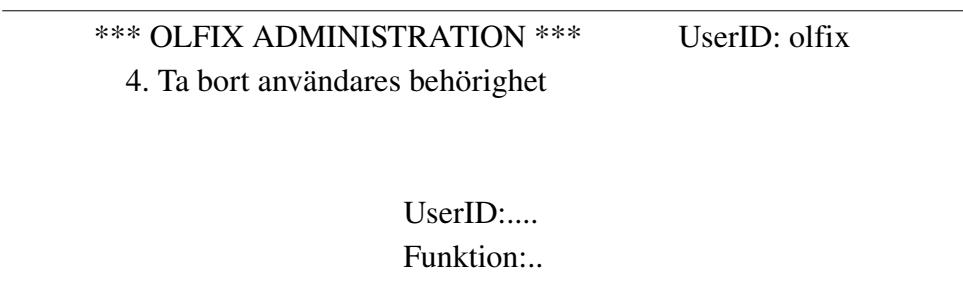
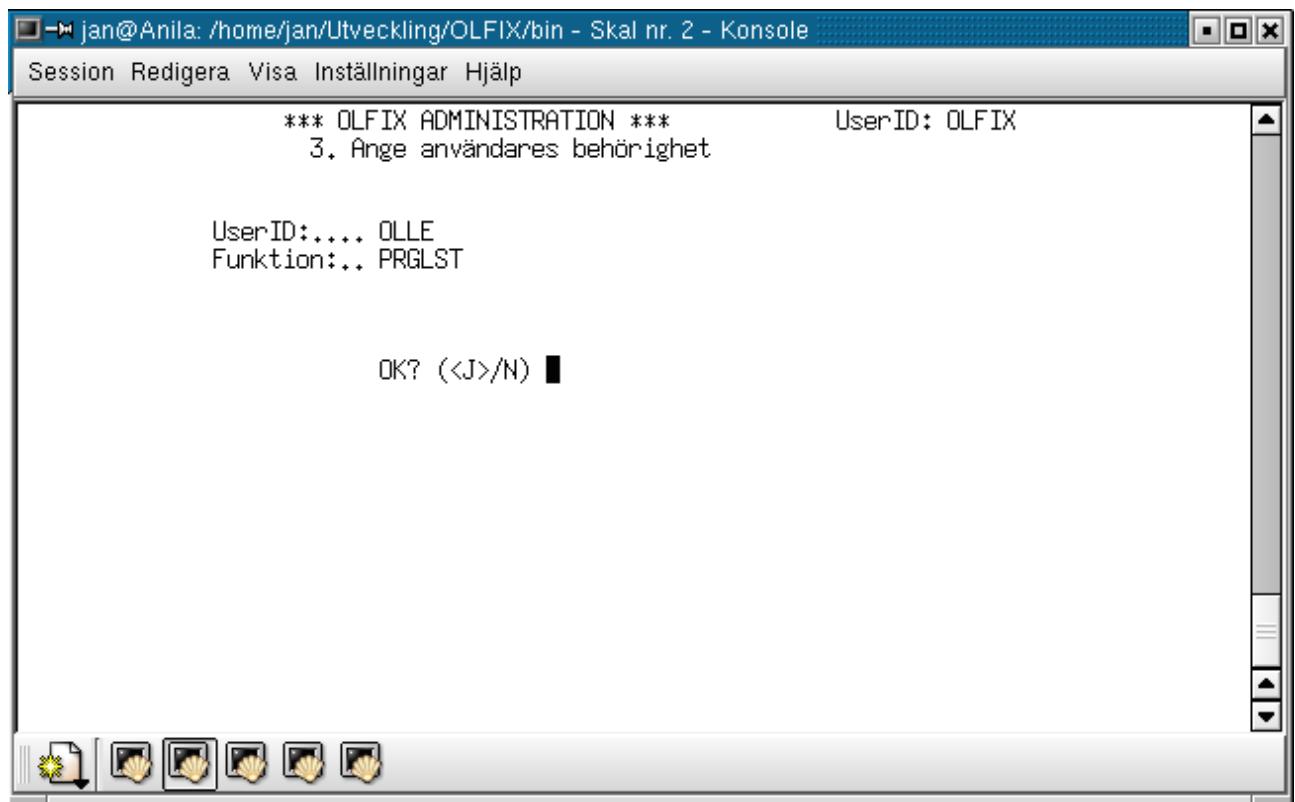
4. Lista användare

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)

---





## BOKF

OBS! Fungerar ej (2003-03-08)

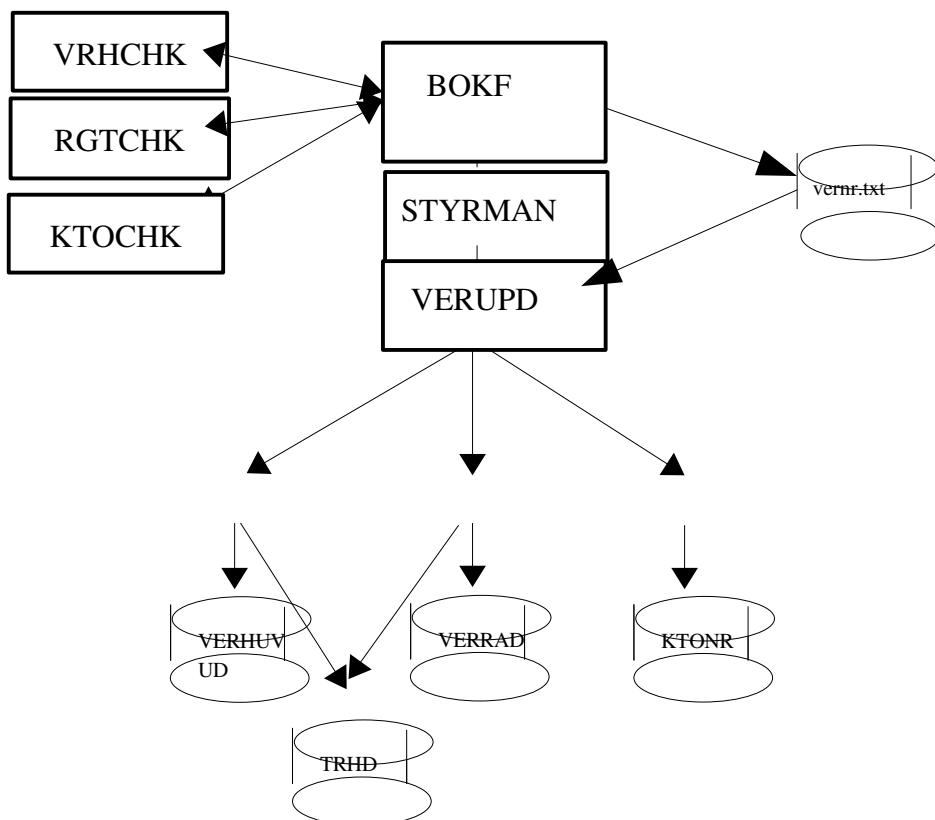
BOKF är ett konsolprogram där bokföringen sker.

Initialt kontrolleras att user har behörighet att använda programmet.

Posterna lagras i temporärfilen vernr.txt

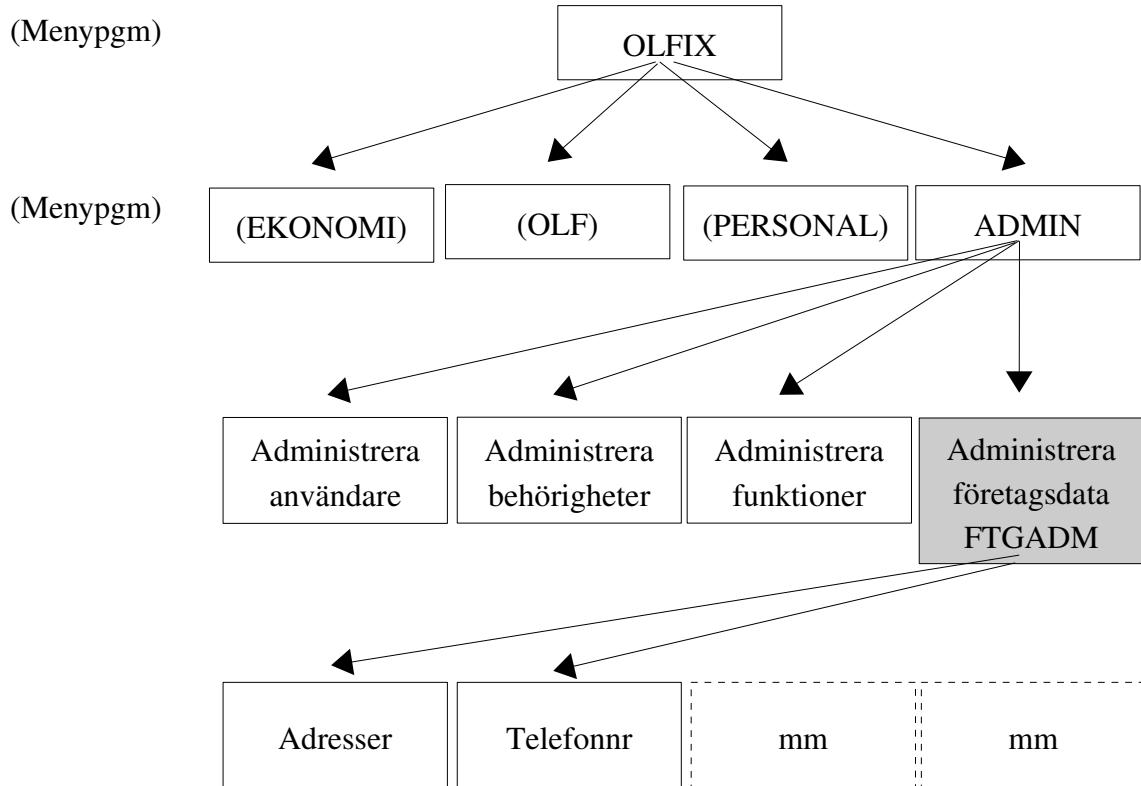
BOKF kontrollerar fortlöpande saldot på verifikationen vilket registreras på verifikationsrad nr 1. För varje därpå följande verifikationsrad så minskas saldot med det belopp som konteras på raden. Först när saldot är 0 (noll) så godkänns verifikationen som färdigbehandlad.

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.



\*\*\* 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.

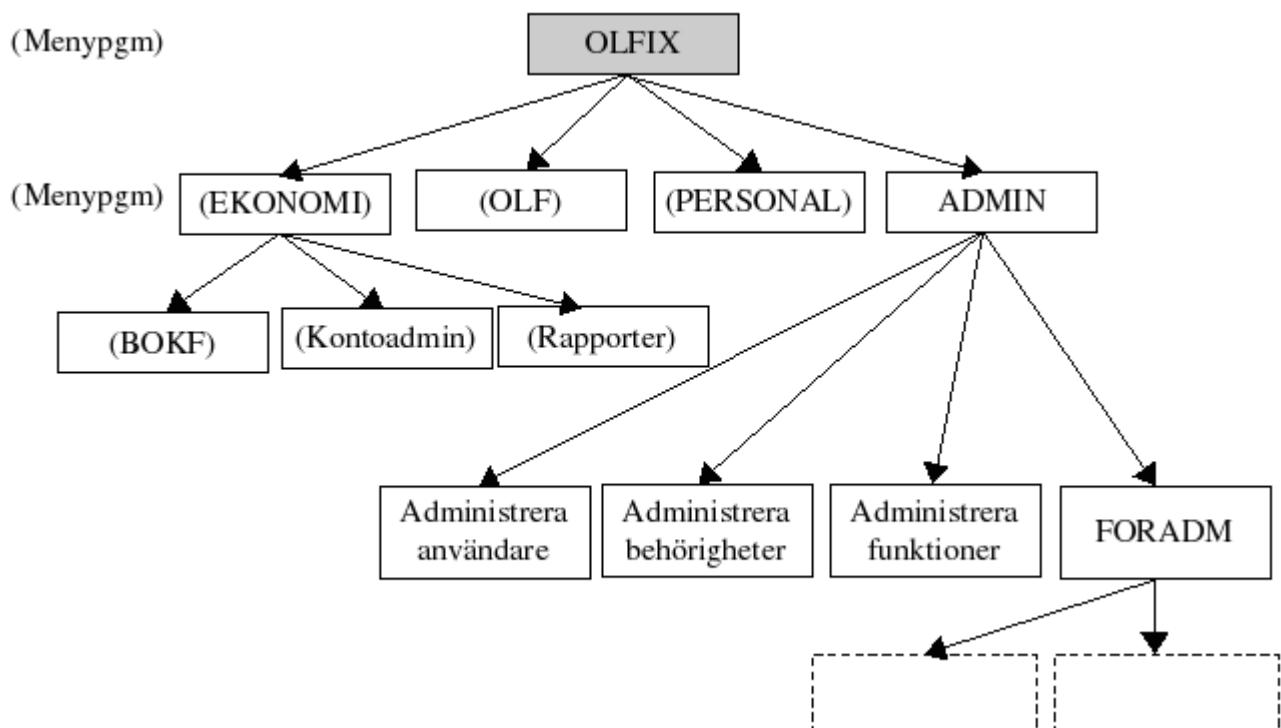
Regler:

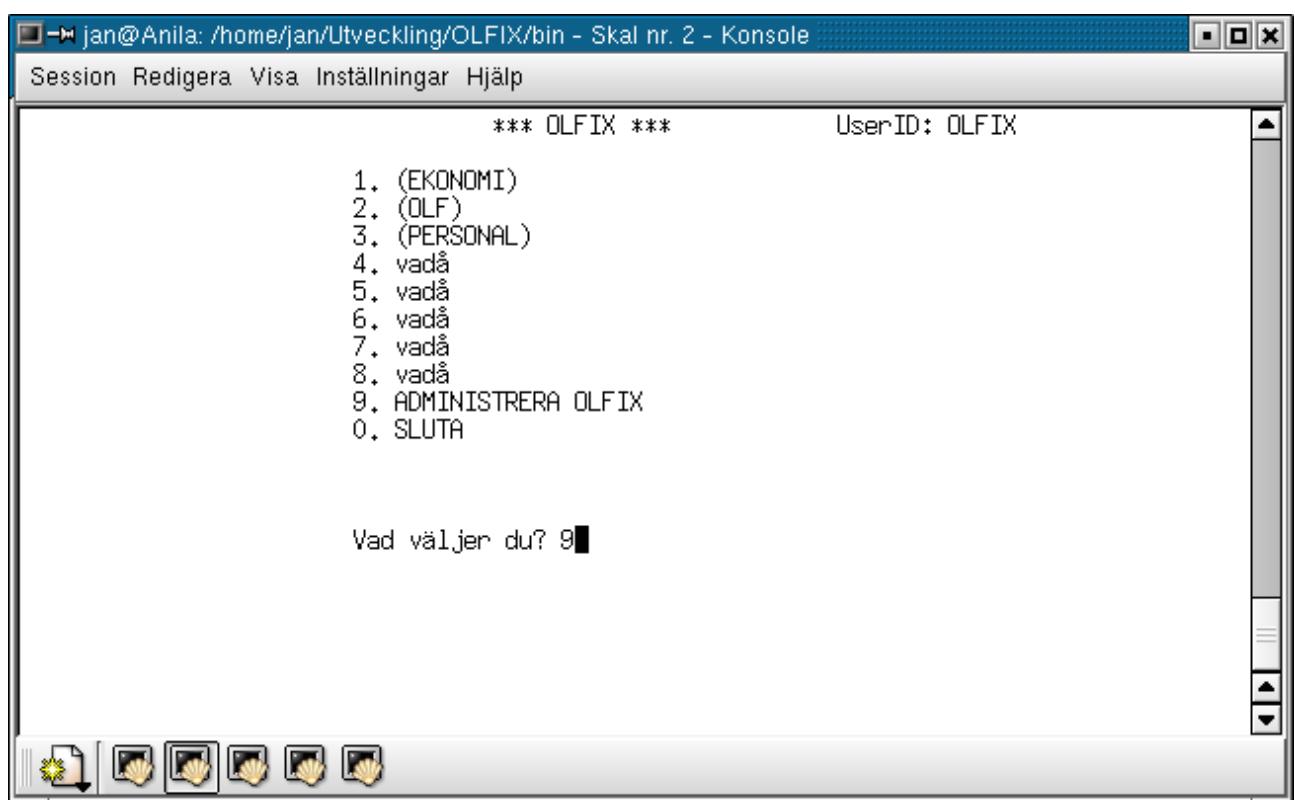
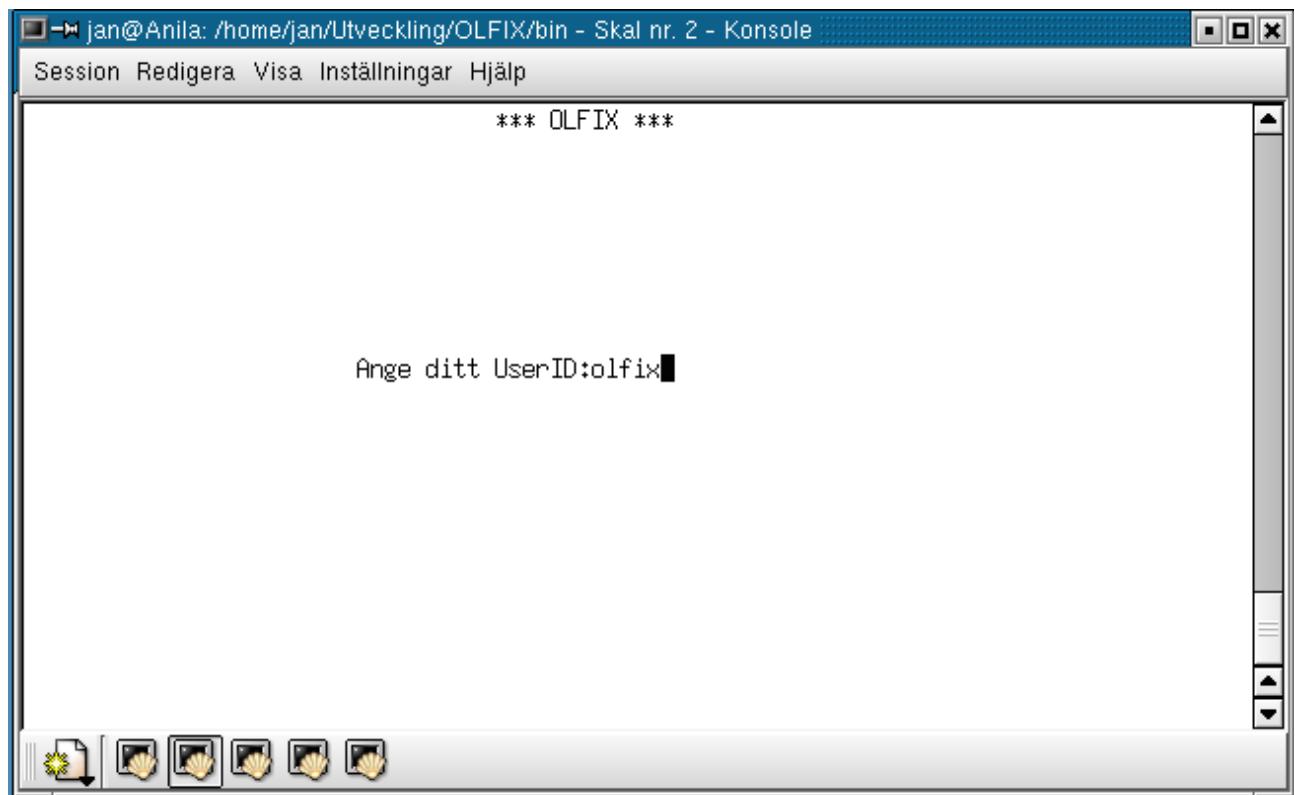
- A. Om parametern [databas] är utelämnad så
  1. Om DATABASE innehåller ett värde används detta
  2. Annars används **olfixst**, testföretag

- 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.





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

UserID: JAPI

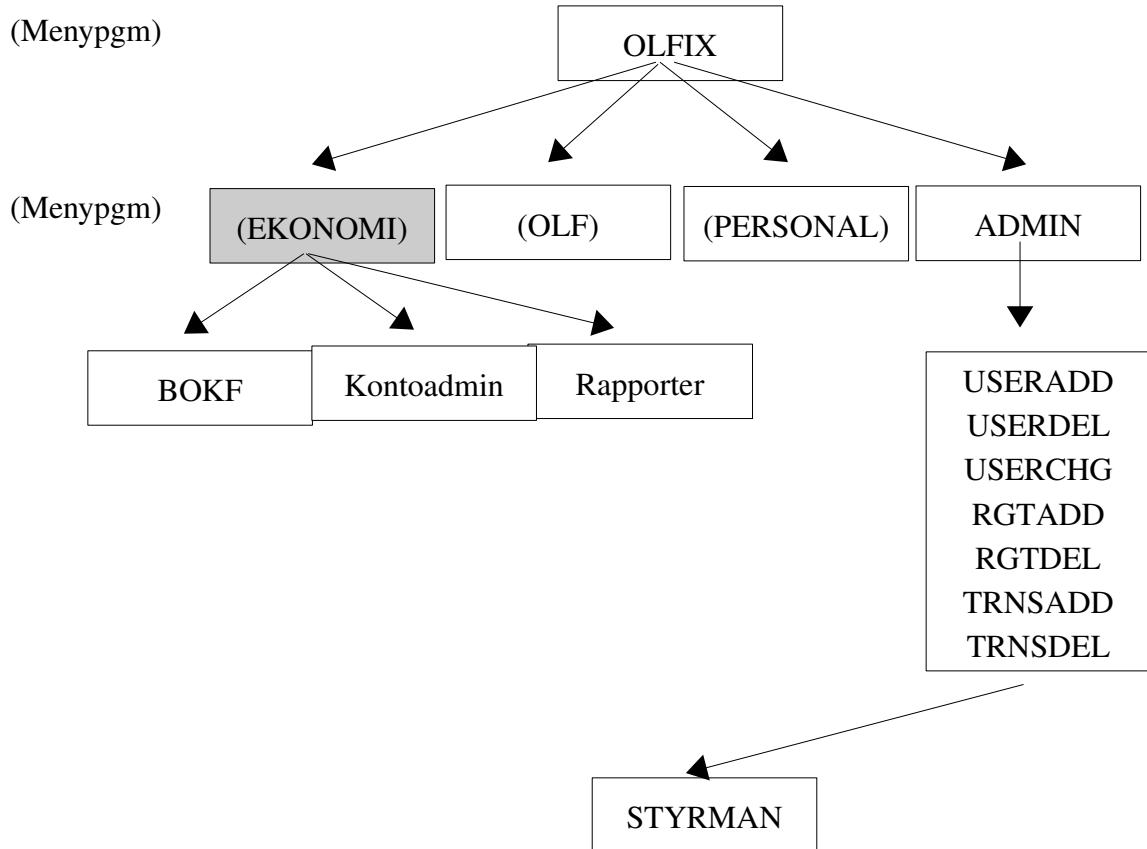
1. BOKFÖRING, registrera verifikationer
2. (Dagboksrapport)
3. (Huvudboksrapport)
4. Lägga till konton
5. Ta bort konton
6. Ändra bokföringsperioder
7. Ändra momssatser
8. Automatkontering (J/N)
9. vadå
0. SLUTA

Vad väljer du?

---

## REDOV(konsolprogram)

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



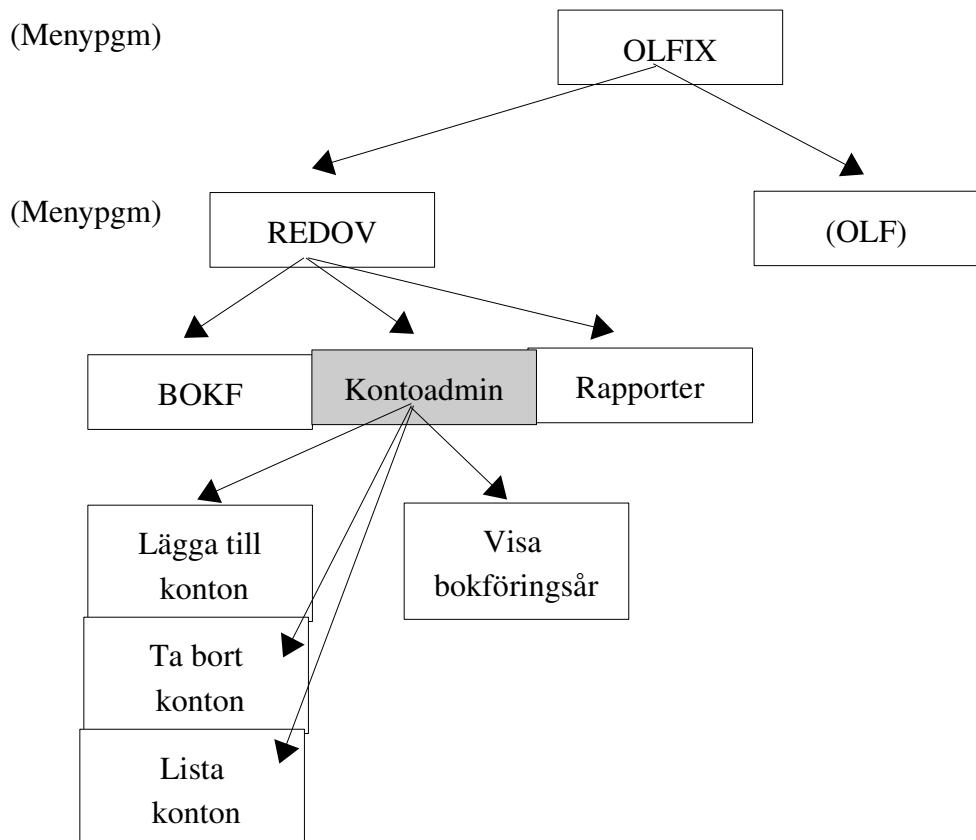
\*\*\* OLFIX REDOVISNING \*\*\*

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?

---



\*\*\* 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

---

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

UserID: JAN

Bokför.år: 2002

3. Lista konton

Kontonr Kontobeskrivning

---

1010	Kassa
1020	Postgiro
1040	Checkkonto
1050	Bank
1120	Aktier och andelar
1210	Kundfordringar
1230	Belånade kundfordringar (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>	Kontorrapport, endast på skärm.
<b>LEVRESKW</b>	Leverantörsreskontrarapport, endast skärm.
<b>HUVBOKW</b>	Huvudbok
<b>SDOLISW</b>	Saldolista
<b>BALRPTW</b>	Balansräkning

## 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 **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.

**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]*

**ARLSPK** Lista artikelregistret. Artikelnummer, benämning 1, benämning 2 och ursprungsbenämning.

Path/STYRMAN userid *ARLSPK produktkod [databas]*

**ARLSTL** Lista artikelsaldo i artikelregistret, lagerstellereg. Artikelnummer, benämning 1, benämning 2 och saldo från lagerställeregistret.

Path/STYRMAN userid *ARLSTL [databas]*

**AR2UPD** Uppdatera LAGERSTELLEREG

Path/STYRMAN userid *AR2UPD val artikelnr data*

- ARSRCH** Söka artiklar i ARTIKELREG  
Path/STYRMAN userid *ARSRCH* sökbegrepp sökord
- 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)
- BARLST** Lista bokföringsår ur tabellen BOKFAR.  
Path/STYRMAN userid *BARLST*
- 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]
- DBCCHK** Lista befintliga databaser i mysql.

Path/STYRMAN userid DBCHK

**DBOKRPT** Hämta data till dagboksrapport.

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

**FORADD** Lägga till ny databas i tabellen DATABASES

Path/STYRMAN userid *FORLST databasnr databasnamn*

**FORCHK** Kontrollera om databas finns i tabellen DATABASES

Path/STYRMAN userid *FORLST databasnr*

**FORDSP** Visa information för angiven databas från tabellen DATABASES

Path/STYRMAN userid FORDSP databasnr [databas]

**FORLST** Lista vilka databaser som finns i tabellen DATABASES

Path/STYRMAN userid *FORLST*

**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*

**KRESADD** Lägga upp poster i kundreskontran.  
Path/STYRMAN userid *KRESADD fakturadata*

**KRESLST** Lista poster i kundreskontran på bildskärm  
Path/STYRMAN userid *KRESLST*

**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 srunr kstalle projekt subkto ktoplan*

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

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

**KTORPT** Läsa ut data ur VERRAD.

Path/STYRMAN userid *KTORPT arid*

**KTOUPD OBS! Använd ej! Ska få ny funktionalitet!**

Uppdatera kontonummer 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 kunddata

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 leicemail levreferent levreftfnnr levmomskod levskuld levkonto*
- LEVCHG** Ändra data på en leverantör.  
path/STYRMAN userid *LEVCHG levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnnr levfaxnr levtelex leicemail levreferent levreftfnnr levmomskod levkonto*
- LEVDSP** Visa information om en leverantör.  
path/STYRMAN userid *LEVDSP levnr*
- LEVLST** Lista leverantörer, leverantörsnummer och leverantörsnamn.  
path/STYRMAN userid *LEVLST*
- LEVPDSP** Visa data för en av en kunds standardleveransplatser.  
Path/STYRMAN userid *LEVPDSP kundnr leveransplatsnr*
- LEVPLST** Lista kunders leveransadresser.  
path/STYRMAN userid *LEVPLST*
- LEVSADD** Lägga till nya leveranssätt.  
path/STYRMAN userid *LEVSADD levsettnr levsettext*
- 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 momsprocent levktonr faktbelopp momsktonr momsbelopp kreditkontonr kreditbelopp userid*

**LRESRPT** Lista obetalda leverantörsfakturor på skärm.

Path/STYRMAN userid *LRESRPT*.

**ORDADD** Lägga upp en ny huvudpost på en kundorder

Path/STYRMAN userid *ORDADD orderhuvuddata*

**ORDDSP** Visa huvudposten på en kundorder.

Path/STYRMAN userid *ORDDSP ordernr*

**ORDRDSP** Visa samtliga orderrader för en kundorder.

Path/STYRMAN userid *ORDRDSP ordernr*

**ORDLST** Lista kundorder, samtliga fält.

Path/STYRMAN userid *ORDLST*

**ORDLST2** Lista kundorder, begränsat antal fält.

Path/STYRMAN userid *ORDLST2*

**ORDUPD** Uppdatera kundorder, begränsat antal fält.

Path/STYRMAN userid *ORDUPD val ordernr data*

**ORADUPD** Uppdatera kundorderrader, begränsat antal fält.

Path/STYRMAN userid *ORDUPD val ordernr data*

**PICKADD** Skapa en plockpost i PLOCKLISTEREG

Path/STYRMAN userid *PICKADD plockdata*

**PICKDSP** Visa en poster på en kundorder

Path/STYRMAN userid *PICKDSP ordernr*

**PICKLST** Lista plockposter som ej är utskrivna eller klara

Path/STYRMAN userid *PICKLST [data]*

**PKDADD** Lägga upp en ny produktgrupp/produktklass.

Path/STYRMAN userid *PKDADD produktkodsnummer beskrivning momskod.*

<b>PKDDSP</b>	Visa data från en produktgrupp. path/STYRMAN userid <i>PKDDSP produktkodsnummer</i>
<b>PKDLST</b>	Lista produktgrupper/produktklasser path/STYRMAN userid <i>PKDLST</i>
<b>PRGLST</b>	Lista programnamn för programmet OLFIXW. path/STYRMAN userid <i>PRGLST</i>
<b>PRTAPI</b>	Interface till utskriftsprogram för rapporter. path/ <i>PRTAPI csvflag prtfile [prttemplate]</i>
<b>RGTADD</b>	Lägga till nya rättigheter för användare path/STYRMAN userid <i>RGTADD userid function</i>
<b>RGTCHK</b>	Kontrollera om användare har viss behörighet. path/STYRMAN userid <i>RGTCHK userid funktion</i>
<b>RGTDEL</b>	Ta bort rättigheter för användare path/STYRMAN userid <i>RGTDEL userid function</i>
<b>RGTDSP</b>	Visa rättigheter för en användare path/STYRMAN userid <i>RGTDSP userid funktion</i>
<b>RGTLST</b>	Visa rättigheter för alla användare path/STYRMAN userid <i>RGTLST</i>
<b>RPTCRE</b>	Rapportgenerator som skapar en CSV-fil utifrån valfri SQL-fråga path/STYRMAN userid <i>RPTCRE sqlquery</i>
<b>SIEEXPK</b>	Skapa/hämta kontoplan för en SIE-fil path/STYRMAN userid <i>SIEEXPK arid (bokföringsår)</i>
<b>SIEEXPR</b>	Skapa/hämta resultat för en SIE-fil path/STYRMAN userid <i>SIEEXPR arid (bokföringsår)</i>
<b>SIEEXPV</b>	Skapa/hämta verifikat för en SIE-fil path/STYRMAN userid <i>SIEEXPV arid (bokföringsår)</i>

**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*

**TXTLST** Lista poster i TEXTREG.

path/STYRMAN userid *TXTLST*

**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**.

Villkor C överrider 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, ARFPRISE, ARLEDDTID,  
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**.

Villkor C överrider 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 Retriev 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**.

Villkor C överrider 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 Retriev 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));
```

# ARLSPK

Funktionen anropas med parametern ARPRODKLASS och, som option, databas.

Funktionen används för att lista artikelnummer,benämning 1 och benämning 2 från ARTIKELREG och lagersaldo från LAGERSTELLEREG.

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/ARLSPK [databas]

Exempel:

```
$ ./ARLSPK 1003 [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARLSPK produktklass [databas]

userid = userid på den som ställer frågan.

Exempel:

```
$ ./STYRMAN JAPI ARLSPK 1003 [olfix]
```

SQLsats som används;

```
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2,ARURARTNR AS ISBNNR FROM ARTIKELREG  
WHERE ARPRODKLASS = "1003" 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: ARLSPK SELECT errno: %d\n",mysql_errno(&my_connection));
```

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

## ARLSTL

Funktionen anropas utan parameter eller, som option, databas.

Funktionen används för att lista artikelnummer,benämning 1 och benämning 2 från ARTIKELREG och lagersaldo från LAGERSTELLEREG.

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/ARLSTL [databas]

Exempel:

```
$ ./ARLSTL [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARLSTL [databas]

userid = userid på den som ställer frågan.

Exempel:

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

SQLsats som används;

```
SELECT ARTIKELREG.ARTIKELNR , ARBENEMNING1 , ARBENEMNING2 ,  
LAGERSTELLEREG.ARLAGSALDO FROM ARTIKELREG  
JOIN LAGERSTELLEREG  
WHERE ARTIKELREG.ARTIKELNR = LAGERSTELLEREG.ARTIKELNR  
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: ARLSTL SELECT error: %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: ARLSTL Retriev error:  %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: ARLSTL Connection failed\n");
fprintf(stderr,"Error: ARLSTL Connection error %d:  %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

# ARSRCH

Funktionen anropas med parametrarna sökbegrepp och sökord samt, som option, databas. Sökbegrepp är endera av siffrorna 1 – 4.

- 1 = artikelnummer
- 2 = benämning 1
- 3 = benämning 2
- 4 = ursprungsbenämning

Sökord är den alfanumeriskecken sträng man söker.

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/ARSRCH begrepp ord [databas]

Exempel:

\$ ./ARSRCH 2 "linu" "10" [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ARSRCH begrepp ord [databas]

userid = userid på den som ställer frågan.

Exempel:

\$ ./STYRMAN JAPI ARSRCH 1 "101" [olfix]

SQLsatser som används;

```
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2 FROM ARTIKELREG WHERE ARTIKELNR  
LIKE "%101%" ORDER BY ARTIKELNR ;  
  
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2 FROM ARTIKELREG WHERE ARBENEMNING1  
LIKE "%linux%" ORDER BY ARBENEMNING1 ;  
  
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2 FROM ARTIKELREG WHERE ARBENEMNING2  
LIKE "%linux%" ORDER BY ARBENEMNING2 ;
```

```
SELECT ARTIKELNR,ARBENEMNING1,ARBENEMNING2 FROM ARTIKELREG WHERE ARURBENEMNING  
LIKE "%linux%" ORDER BY ARURBENEMNING";
```

## Interface:

### INPUT:

sökbegrepp sökord

### OUTPUT:

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

## AR2UPD

Funktionen anropas med parametrarna val, artikelnummer, data samt, som option, databas. Funktionen används för att uppdatera fältet RESERVERAT i tabellen LAGERSTELLEREG.

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/AR2UPD val artikelnr data [databas]

Exempel:

```
$ ./AR2UPD 1 "1000-1001" "10" [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid AR2UPD val artikelnr data [databas]

userid = userid på den som ställer frågan.

Exempel:

```
$ ./STYRMAN JAPI AR2UPD 1 "1000-1001" "10" [olfix]
```

SQLsats som används;

```
UPDATE LAGERSTELLEREG SET RESERVERAT = RESERVERAT + "10";  
WHERE ARTIKELNR = "1000-1001"
```

**Interface:**

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: Måste sätta val större än 0!\n");  
fprintf(stderr,"OK: AR2UPD Inserted %lu rows\n",  
       (unsigned long)mysql_affected_rows(&my_connection));  
fprintf(stderr,"Error: AR2UPD UPDATE error: %d %s\n",  
       mysql_errno(&my_connection),mysql_error(&my_connection));
```

```
fprintf(stderr,"Error: AR2UPD Connection failed\n");
fprintf(stderr,"Error: AR2UPD 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**.

Villkor 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 leverantörsfakturor 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 **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 överrider 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,FAKTBEOLOPP,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 Retriev 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**.

Villkor 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,  
VERNAR,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**.

Villkor C överrider 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,

## SENDERDAT, 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 **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 överrider 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 Retriev 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 **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/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 Retriev 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 överrider 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 Retriev 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);
```

## BARLST

BARLST anropar tabellen BOKFAR utan parametrar, eventuellt 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 överrider villkoren A och B.

Syntax:

Path/BARLST [databas]

Exempel:

```
$ ./BARLST [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BARLST [databas]

Exempel:

```
$ ./STYRMAN JAPI BARLST [olfixtst]
```

SQLsats som används;

```
SELECT ARID,BENAMNING,KONTOPLAN FROM BOKFAR ORDER BY ARID
```

### Interface:

INPUT:

OUTPUT:

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

## 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**.

Villkor C överrider 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/BETCHG betvillkor dagar beskrivning [databas]

Exempel:

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

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid BETCHG betvillkor 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 **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/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 Retriev 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 **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/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 Retriev 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));
```

## DBCHK

Funktionen anropas utan parametrar.

Programmet anropar databasen **mysql**.

DBCHK skriver ut data på stdout (konsolen) för alla registrerade databaser i systemet.

Programmet är avsett att användas för att kontrollera vilka databaser som existerar.

Syntax:

Path/DBCHK [databas]

Exempel:

```
$ ./DBCHK [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid1 DBCHK [databas]

Exempel:

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

SQLsats som används;

```
SELECT db FROM db
```

### Interface:

INPUT:

OUTPUT:

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

Exempel på utdata:

```
NR_7_:test_:test\_%_:bbswiki_:mysql_:olfix_:olfixtst_:test_:
```

# 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 **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 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, 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 Retriev 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");
```

## FORADD

Funktionen för att lägga upp en ny posttyp i tabellen DATABAS och anropas med parametern DATABASNR, DATABASNAMN

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 överrider villkoren A och B.

Syntax:

Path/FORADD databasnr databasnamn [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FORADD databasnr databasnamn [databas].

SQLsats som används;

INSERT INTO DATABAS(DATABASNR,DATABASTEXT) VALUES ("55", "foretag")

Med foretag menas här namnet på databasen i MySQL.

### Interface:

**INPUT:** DATABASNR DATABASTEXT

**OUTPUT:**

```
fprintf(stderr,"Error: FORADD Databasen %s finns inte!\n",dbnr);
fprintf(stderr,"OK: Inserted %lu rows\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stdout,"OK: FORADD Status = %d\n",status);
fprintf(stderr,"Error: FORADD INSERT error: %d %s\n",
mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: FORADD Connection failed\n");
fprintf(stderr,"Error: FORADD Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

## FORCHK

Funktion för att kontrollera förekomsten av databasnummer i tabellen DATABAS och anropas med parametern DATABASNR.

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/FORCHK databasnr [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FORCHK databasnr [databas].

SQLsats som används;

SELECT DATABASNR FROM DATABAS WHERE DATABASNRNR = "55"

### Interface:

**INPUT:** DATABASNR

**OUTPUT:**

```
fprintf(stderr,"Error: FORCHK SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: FORCHK Retriev error:  %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: FORCHK Connection failed\n");
fprintf(stderr,"Error: FORCHK Connection error %d:  %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stdout,"OK: FORCHK Status = %d\n",status);
fprintf(stderr,"Error: FORCHK Databasen %s finns inte!\n",dbnr);
```

## FORDSP

Funktion för att kontrollera förekomsten av databasnummer i tabellen DATABAS och anropas med parametern DATABASNR.

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/FORDSP databasnr [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FORDSP databasnr [databas].

SQLsats som används;

SELECT DATABASTEXT FROM DATABAS WHERE DATABASNR="55"

### Interface:

**INPUT:** DATABASNR

**OUTPUT:**

```
fprintf(stderr,"Error: FORDSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: FORDSP Retriev error:  %s\n",
        mysql_error(&my_connection));
fprintf(stderr,"Error: FORDSP Connection failed\n");
fprintf(stderr,"Error: FORDSP 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]);
ex: NR_1_:olfix_:
fprintf(stderr,"Error: FORDSP Databasen %s finns inte!\n",dbnr);
```

## FORLST

Funktion för att lista innehållet i tabellen DATABASES. Funktionen 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/FORLST [databas]

Normalt sker anrop via STYRMAN:

Path/STYRMAN userid FORLST [databas].

SQLsats som används;

SELECT \* FROM DATABASES ORDER BY DATABASESNR

### Interface:

#### INPUT:

#### OUTPUT:

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

## 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**.

Villkor C överrider 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**.

Villkor C överrider 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	Mobiltelefonnummer	
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 Retriev 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**.

Villkor C överrider 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	Mobiltelefonnummer	
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 Retriev 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**.

Villkor C överrider 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 Retriev 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 **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/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));
```

# HBORPT

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.VERTEXT,
    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 Retriev 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 kostnadstället.

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**.

Villkor C överrider 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:_:_Godsmärke:_:_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, LEVADRESS,  
LEVPOSTNR, LEVPOSTADR, LEVLAND, LEVVALUTA, LEVBETVILLKOR, GODSMERKE, BESTTEXT,  
VARREF, LEVDATUM, 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,LEVADRESS,  
LEVPOSTNR,LEVPOSTADR,LEVLAND,LEVVALUTA,LEVBEVILLKOR,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**.

Villkor 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 , INKPRAIS , LEVVECKA ,
      RESTNOTERAT * INKPRAIS 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 Retriev 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 **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 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**.

Villkor 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]);      /* leverantörsländ */
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]);     /* varreftfn */
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 Retriev 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/INKRLST inkordernr [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, INKPRAIS, 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 Retriev 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));
```

## KRESADD

Funktionen används för att lägga upp nya poster i kundreskontran KURESK.  
Funktionen anropas med parametrarna fakturadata 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/KRESADD arid kstalle benamning [databas]

Exempel:

```
$ ./KRESADD fakturadata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KRESADD fakturadata [databas].

Exempel:

```
$ ./STYRMAN jan KRESADD fakturadata
```

SQLsats som används:

```
INSERT INTO KURESK  
(ORDERNR,FAKTURANR,KUNDNR,FAKTURADATUM,EXPIREDATUM,NETTOBELOPP,MOMSBELOPP,  
FAKTURABELOPP,BETALD,BETALDDATUM,USERID,VALUTA,VALUTAKURS,VALUTABELOPP,BAR,  
VERNR,MOMSKTONR,KTONR,DEBETBELOPP)  
VALUES  
( "1093" , "1351" , "4376" , "2005-11-11" , "2005-12-11" ,  
"1523.50" , "380.90" , "1904.40" , "N" ,  
"0000-00-00" , "JAPI" , "SEK" , "1.00" , "1904.40" , "AE" , "25341" , "2410" ,  
"1210" , 1904.40 )
```

### Interface:

INPUT:

fakturadata

ex \_:\_1093\_:\_1351\_:\_4376\_:\_2005-11-11\_:\_2005-12-11 \_:\_1523.50\_:\_380.90\_:\_  
1904.40\_:\_N\_:\_0000-00-00\_:\_JAPI\_:\_SEK\_:\_1.00\_:\_1904.40\_:\_AE\_:\_25341\_:\_  
2410\_:\_1210\_:\_1904.40\_:\_END

## OUTPUT:

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

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

## KRESLST

Funktionen används för att lista poster i kundreskontran, KURESK.

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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/KRESLST [databas]

Exempel:

\$ ./KRESLST [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KRESLST [databas].

Exempel:

\$ ./STYRMAN jan KRESLST

SQLsats som används:

```
SELECT ORDERNR ,FAKTURANR ,KUNDNR ,FAKTURABELOPP ,EXPIREDATUM ,BETALD  
FROM KURESK  
ORDER BY ORDERNR ;
```

### Interface:

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: KRESLST SELECT error:%s\n",  
        mysql_error(&my_connection));  
fprintf(stdout,"OK: NR_%lu:_",(unsigned long)mysql_num_rows(res_ptr));  
fprintf(stdout,"END");  
fprintf(stderr,"Error: KRESLST Retriev error: %s\n",
```

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

# KSTADD

Funktionen används för att lägga upp en nya kostnadstället.

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**.

Villkor C överrider 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**.  
Villkor 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 Retriev 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 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 överrider 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_Retrieve 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**.

Villkor C överrider 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 Retriev 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**.

Villkor C överrider 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**.

Villkor C överrider 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**.

Villkor C överrider 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 faild\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**.

Villkor 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 Retriev 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**.

Villkor C överrider 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 upp 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 Retriev 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 **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 överrider 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[olfixtst]
```

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\_' postantalt '\_' (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 Retriev 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 funktionalitet 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 **exec** 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 ettdera 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**.

Villkor C överrider 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 upp 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 Retriev 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 environtvariabeln 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**.

Villkor C överrider 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 _:_ 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 _:_ 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,EXPAGIFT,FRAKTAVG,KRAVBREV,
KREDITLIMIT,DROJMALSRTA,DROJMALSAKTURA,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","SEK","sv","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**.

Villkor C överrider 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:_:2000:_:J:_:J:_:Fritt textfält:_:  
30:_:001:_:001:_:001:_:J:_:  
_:_ = fältavskiljare.
```

Ordningen på data är **synnerligen** viktig.

Ordningen **skall** vara:

NAMN, 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,DROJMALSAFTURA, FRITEMPT,  
KREDITDAGAR, KREDITKOD, EXPORTKOD, SKATTEKOD, RABATTKOD,  
SAMLINGSAFTURA

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", DISTRIKT="Kal",
    KUNDKATEGORI="Sto", STDLEVPLATS="002", LEVVILLKOR="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 Updated %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 **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 överrider 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**.

Villkor C överrider 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, LEVVILLKOR, LEVSETT, DISTRIKT,

KUNDKATEGORI, STDLEVPLATS, ORDERERKENNANDE,PLOCKLISTA,  
FOLJESEDEL, KRAVBREV, SPRAKKOD, EXPAVGIFT, FRAKTAVG, KREDITLIMIT,  
KREDITDAGAR, KREDITKOD, EXPORTKOD, SKATTEKOD, RABATTKOD,  
DROJSMALSRTA,DROJSMALSAFTURA, 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("..... ",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 Retriev 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 parametrar 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**.

Villkor C överrider 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 Retriev 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));
```

# KUSRCH

Funktionen används för att söka efter kunder i tabellen KUNDREG. Sökning kan göras på namn, postnr, telefonnr och postadress(ort).

Funktionen anropas med parametrarna **soekbegrepp** och **soekord** samt, som option, **databas**. Soekbegrepp är 1=namn, 2=postnr, 3=telefonnr, 4=postadress.

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 överrider villkoren A och B.

Syntax:

Path/KUSRCH soekbegrepp soekord [databas]

Exempel:

```
$ ./KUSRCH soekbegrepp soekord [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid KUSRCH soekbegrepp soekord [databas].

Exempel:

```
$ ./STYRMAN olfix KUSRCH soekbegrepp soekord [databas]
```

SQLsats som används:

```
SELECT KUNDNR,NAMN,POSTNR,POSTADR,TFNNR FROM KUNDREG WHERE NAMN LIKE "%soekord%" ORDER BY NAMN
SELECT KUNDNR,NAMN,POSTNR,POSTADR,TFNNR FROM KUNDREG WHERE POSTNR LIKE "%soekord%" ORDER BY POSTNR
SELECT KUNDNR,NAMN,POSTNR,POSTADR,TFNNR FROM KUNDREG WHERE TFNNR LIKE "%soekord%" ORDER BY TFNNR
SELECT KUNDNR,NAMN,POSTNR,POSTADR,TFNNR FROM KUNDREG WHERE POSTADR LIKE "%soekord%" ORDER BY POSTADR
```

## **Interface:**

### **INPUT:**

KUSRCH 4 "små"

### **OUTPUT:**

OK: NR\_2\_:4379\_:Nya Småkund AB\_:199 02\_:SMÅSTAD\_:09-129990\_:4375

### **Beskrivning:**

_:	Fältavskiljare
NR_2	Antal poster
4379	Kundnummer
Nya Småkund AB	Kundnamn
199 02	Postnr
SMÅSTAD	Postadress (ort)
09-129990	Telefonnr
4375	Nästa kund
osv ...	

## LEVADD

Funktion för att registrera information på en leverantör.

Funktionen anropas med parametern **levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betalvilkor** och som option **databas**.

Levkuld 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/LEVADD levnr levorgnr levnamn levadress levpostnr levpostadress levland levtfnr levfaxnr levtelex levreferent levreftfn levmomskod levskuld levkonto [databas]

Exempel:

```
$ ./LEVADD 123 "559999-9999" "Leverantör AB" "Postgatan 33" "199 99" "DATABY"  
"SVERIGE" "09-999999" "09-999998" "99999" "kundtj@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 levtfnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betalvilkor [olfixtst]

Exempel:

```
$ ./STYRMAN olfix LEVADD levnr levorgnr levnamn levadress levpostnr levpostadress  
levland levtfnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld  
levkonto levkundnr valuta betalvilkor [olfixtst]
```

SQLsats som används:

```
INSERT INTO LEVREG  
(LEVNR,LEVORGNR,LEVNAME,LEVADDRESS,LEVPOSTNR,LEVPOSTADR,LEVLAND,LEVTNNR,LEVFAFN,  
LEVTELEX,LEVEMAIL,LEVREFERENT,LEVREFTFN,LEVOMSKOD,LEVSKULD,LEVKONTO,LEVKUNDNR,VALUTA,BET  
ALVILKOR)
```

```
VALUES
( "2345" , "559999-9999" , "Leverantör AB" , "Postgatan 33" , "199 99" , "DATABY" , "SVERIGE" , "09-
999999" , "09-999998" , "99999" , "kundtj@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 lepostnr lepostadress levland levtnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betalvilkor** 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 överrider villkoren A och B.

Syntax:

path/LEVCHG levnr levorgnr levnamn levadress lepostnr lepostadress levland levtnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levskuld levkonto levkundnr valuta betalvilkor [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 lepostnr lepostadress levland levtnr levfaxnr levtelex levemail levreferent levreftfn levmomskod levkonto levkundnr valuta betalvilkor [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="kundt\_j@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**.

Villkor C överrider 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:\n");  
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 Retriev 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**.

Villkor C överrider 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 Retriev 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));
```

## LEVPDSP

Funktion för att visa en **leveransplats**. Med leveransplats menas här kundens olika leveransadresser. Funktionen anropas med parametrarna kundnr och leveransplatsnr samt, 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 **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 överrider villkoren A och B.

Syntax:

Path/LEVPDSP kundnr leveransplatsnr [databas]

Exempel:

```
$ ./LEVPDSP kundnr leveransplatsnr [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

```
Path/STYRMAN userid LEVPDSP kundnr leveransplatsnr [databas]
```

Exempel:

```
$ ./STYRMAN olfix LEVPDSP kundnr leveransplatsnr [olfixtst]
```

SQLsats som används:

```
SELECT * FROM STDLEVPLATS WHERE (kundnr = "12334" AND STDLEVPLATS = "002")
```

### Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: ");
fprintf(stdout,"01:%s ",sqlrow[0]);
fprintf(stdout,"02:%s ",sqlrow[1]);
fprintf(stdout,"03:%s ",sqlrow[2]);
fprintf(stdout,"04:%s ",sqlrow[3]);
```

```
fprintf(stdout,"05:%s ",sqlrow[4]);
fprintf(stdout,"06:%s ",sqlrow[5]);
fprintf(stdout,"%s_:",sqlrow[field_count]);
fprintf(stderr,"Error: LEVPDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: LEVPDSP Retriev error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVPDSP Connection failed\n");
fprintf(stderr,"Error: LEVPDSP Connection error %d:%s\n",mysql_errno(&my_connection),
        mysql_error(&my_connection));
```

## LEVPLST

Funktion för att lista alla **leveransplatser**. Med leveransplats menas här kundens olika leveransadresser. 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/LEVPLST [databas]

Exempel:

\$ ./LEVPLST [olfix]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid LEVPLST [databas]

Exempel:

\$ ./STYRMAN olfix LEVPLST [olfixtst]

SQLsats som används:

```
SELECT * FROM STDLEVPLATS ORDER BY STDLEVPLATS
```

### Interface:

INPUT:

OUTPUT:

```
fprintf(stdout,"OK: NR_%lu_:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: LEVPLST SELECT errno: %d\n",mysql_errno(&my_connection));
```

```
fprintf(stderr,"Error: LEVPLST Retriev error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVPLST Connection failed\n");
fprintf(stderr,"Error: LEVPLST 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**.

Villkor C överrider 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**.

Villkor C överrider 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 Retriev 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**.

Villkor C överrider 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 Retriev 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**.

Villkor C överrider 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**.

Villkor C överrider 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 LEVVDS 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: LEVVDS SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: LEVVDS Leveransvillkor saknas!\n");
fprintf(stderr,"Error: LEVVDS Retriev error: %s\n",mysql_error(&my_connection));
fprintf(stderr,"Error: LEVVDS Connection failed\n");
fprintf(stderr,"Error: LEVVDS 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**.

Villkor C överrider 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 Retriev 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, fakt datum, 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**.

Villkor C överrider villkoren A och B.

Syntax:

Path/LRESADD *levnr fakturanr regdatum fakt datum 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 fakt datum 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,
MOMSPROCENT, 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**.

Villkor C överrider 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.LEVNRL, LEVREG.LEVNAMN, LEVRESK.FAKTBELOPP
      FROM LEVRESK
     LEFT JOIN LEVREG ON LEVREG.LEVNRL = LEVRESK.LEVNRL
    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 Retriev 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));
```

## ORDADD

Funktionen uppdaterar databasen, tabell ORDERREG.

Funktionen anropas med parametern orderhuvuddata 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 överrider villkoren A och B.

Syntax:

Path/ORDADD orderhuvuddata [databas]

Exempel:

```
$ ./ORDADD orderhuvuddata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDADD orderhuvuddata

Exempel:

```
$ ./STYRMAN userid ORDADD orderhuvuddata [databas]
```

SQLsats som används:

```
INSERT INTO ORDERREG  
(ORDERNR, ORDERDATUM, KUNDNR, KUNDNAMN, KUNDADDRESS, KUNDPOSTNR, KUNDPOSTADR, KUNDLAND,  
ERREF, LEVADRESS, LEVPOSTNR, LEVPOSTADR, LEVLAND,  
SELJARE, LEVDATUM, MOMS, VALUTA, BETVILLKOR, LEVVILLKOR, LEVSETT, GODSMERKE,  
ORDERSUMMA, FRAKTSUMMA, FRAKTMOMSKR, ORDERMOMS, ORDERTOTAL) VALUES  
(1, "2005-03-12", "4376", "Nya Test AB", "Provgatan 2", "199 99", "LILLEBY",  
"Sverige", "Karl Andersson", "Bokhållargatan 3", "199 19", "LILLEBY", "Sverige",  
"Lars Seller", "2005-03-15", "25", "SEK", "1", "001", "002", "Godsmärke",  
"5117.00", "90.00", "22.50", "1279.52", "6509.02")
```

### Interface:

INPUT:

orderhuvuddata ser ut på detta vis:

\_:\_1:\_:2005-03-12:\_:4376:\_:Nya Test AB \_:\_Provgatan 2 \_:\_199 99 \_:\_LILLEBY \_:\_Sverige \_:\_  
Karl Andersson \_:\_Bokhållargatan 3 \_:\_199 19 \_:\_LILLEBY \_:\_Sverige \_:\_Lars Seller\_:\_2005-03-15\_:\_25\_:\_SEK\_:\_  
1 \_:\_001 \_:\_002 \_:\_Godsmärke:\_:5117.00\_:\_90.00:\_:22.50\_:\_1279.52\_:\_6509.02\_:\_END

## OUTPUT:

Utdata från ORDADD är:

```
OK: ORDADD Inserted 1 rows
fprintf(stderr,"Error: ORDADD: Ange kundordernummer!\n");
"Error: ORDADD INSERT error: %d %s\n", mysql_errno(&my_connection),
      mysql_error(&my_connection));
fprintf(stderr,"Error: ORDADD Connection failed\n");
fprintf(stderr,"Error: ORDADD Connection error %d: %s\n",
      mysql_errno(&my_connection), mysql_error(&my_connection));
```

# ORDCHG

Funktionen uppdaterar databasen, tabell ORDERREG.

Funktionen anropas med parametern orderhuvuddata 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 överrider villkoren A och B.

Syntax:

Path/ORDCHG orderhuvuddata [databas]

Exempel:

```
$ ./ORDCHG orderhuvuddata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDCHG orderhuvuddata

Exempel:

```
$ ./STYRMAN userid ORDCHG orderhuvuddata [databas]
```

SQL-sats som används:

```
UPDATE ORDERREG SET KUNDNAMN="Nya Test AB ",  
    KUNDADRESS="Provgatan 2",  
    KUNDPOSTNR="199 99",  
    KUNDPOSTADR="LILLEBY",  
    KUNDLAND="Sverige",  
    ERREF="Karl Andersson",  
    LEVADRESS="Bokhållargatan 3",  
    LEVPOSTNR="199 19",  
    LEVPOSTADR="LILLEBY",  
    LEVLAND="Sverige",  
    SELJARE="Lars Seller",  
    LEVDATUM="2006-02-15",  
    MOMS="25",  
    VALUTA="SEK",
```

```

BETVILLKOR="1",
LEVVILLKOR="001",
LEVSETT="002",
GODSMERKE="Godsmärke",
ORDERSUMMA="5117.00",
FRAKTSUMMA="90.00",
FRAKTMOMSKR="22.50",
ORDERMOMS="1279.52",
ORDERTOTAL="6509.02"
WHERE ORDERNR="1"

```

## Interface:

INPUT: orderhuvuddata databas

där orderhuvuddata =

```

" _:_1:_:_Nya Test AB:_:_Provgatan 2:_:_199 99:_:_LILLEBY:_:_Sverige _:__
Karl Andersson:_:_Bokhållargatan 3:_:_199 19:_:_LILLEBY:_:_Sverige _:__
Lars Seller:_:_2005-03-15:_:_25:_:_SEK:_:_1 _:_001 _:_002 _:_Godsmärke:_:__
5117.00:_:_90.00:_:_22.50:_:_1279.52:_:_6509.02:_:_END"

```

\_:\_ = skiljetecken för fält.

## OUTPUT:

```

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

```

## ORDRADD

Funktionen uppdaterar databasen, tabell ORDERRADREG.

Funktionen anropas med parametern kundorderraddata 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 överrider villkoren A och B.

Syntax:

Path/ORDRADD kundorderraddata [databas]

Exempel:

```
$ ./ORDRADD kundorderraddata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDRADD kundorderraddata

Exempel:

```
$ ./STYRMAN userid ORDRADD kundorderraddata [databas]
```

SQLsats som används:

```
INSERT INTO ORDERRADREG
(ORDERNR,ORDERRAD,KUNDNR,ARTIKELNR,BENEMNING,LEVERANSVECKA,BESTELLT,APRIS,
SUMMA,MOMSKR) VALUES("33","020","4377","1000-1016","Seagate Baracuda 7200+7 160GB",
"5111","4","650.00","2600.00","650.00")
```

### Interface:

INPUT:

kundorderraddata ser ut på detta vis:

```
_:_33_:_020_:_4377_:_1000-1016_:_Seagate Baracuda 7200+7 160GB _:_5111_:_4_:_650.00
_:_2600.00_:_650.00_:_END
```

## OUTPUT:

Utdata från ORDRADD är:

```
fprintf(stderr,"Error: ORDRADD: Ange inköpsordernummer!\n");
OK: ORDRADD Inserted 1 rows
"Error: ORDRADD INSERT error: %d  %s\n", mysql_errno(&my_connection),
      mysql_error(&my_connection));
fprintf(stderr,"Error: ORRDADD Connection failed\n");
fprintf(stderr,"Error: ORDRADD Connection error %d:  %s\n",
      mysql_errno(&my_connection), mysql_error(&my_connection));
```

## ORDRCHG

Funktionen uppdaterar databasen, tabell ORDERRADREG.

Funktionen anropas med parametern kundorderraddata 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 överrider villkoren A och B.

Syntax:

Path/ORDRCHG kundorderraddata [databas]

Exempel:

```
$ ./ORDRCHG kundorderraddata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDRCHG kundorderraddata

Exempel:

```
$ ./STYRMAN userid ORDRCHG kundorderraddata [databas]
```

SQLsats som används:

```
REPLACE INTO ORDERRADREG
(ORDERNR,ORDERRAD,KUNDNR,ARTIKELNR,BENEMNING,LEVERANSVECKA,BESTELLT,APRIS,SUMMA,MOMSKR,
LEVERERAT,RESTNOTERAT,RADRABATT,KALKYLPRISE,LEVDATE,ENHET,FAKTURERAT,RADORDERTYP)
VALUES(_:_33:_:020:_:4377:_:1000-1016:_:Seagate Baracuda 7200+7 160GB:_:
5111:_:4:_:650.00:_:2600.00:_:650.00:_:2.00:_:13.00:_:14.0:_:15.00:_:2006-02-20:_:
ST:_:18.00:_:_END)
```

REPLACE är en utökning av MySQL.

REPLACE medger ersättning av data i en post. Om posten saknas görs en INSERT i stället.

OBS! Alla fälten i tabellen ORDERRADREG byts ut.

## Interface:

### INPUT:

kundorderraddata ser ut på detta vis:

```
_:_1:_10:_4376 _:_1000-1016:_Seagate Baracuda 7200+7 160GB  
_:_6113:_4.00:_650.00:_2600.00:_650.00:_2.00:_13.00:_14.0:_15.00:_  
2006-02-20:_ST:_1:_D:_END"
```

### OUTPUT:

Utdata från ORDRCHG är:

```
fprintf(stderr,"Error: ORDRCHG: Ange ordernummer!\n");  
OK: ORDRCHG Inserted 1 rows  
"Error: ORDRCHG INSERT error: %d %s\n", mysql_errno(&my_connection),  
      mysql_error(&my_connection));  
fprintf(stderr,"Error: ORDRCHG Connection failed\n");  
fprintf(stderr,"Error: ORDRCHG Connection error %d: %s\n",  
      mysql_errno(&my_connection), mysql_error(&my_connection));
```

## ORDCHK

Funktionen kontrollerar diverse uppgifter på en kundorder, tabell ORDERREG.

Funktionen anropas med parametrarna val och ordernr samt, 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 överrider villkoren A och B.

Syntax:

Path/ORDCHK val ordernr [databas]

Exempel:

```
$ ./ORDCHK val ordernr [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDCHK val ordernr

Exempel:

```
$ ./STYRMAN userid ORDCHK val ordernr [databas]
```

SQLsats som används:

```
SELECT ORDERNR FROM ORDERREG WHERE ORDERNR = "1022";  
SELECT ORDERSTATUS FROM ORDERREG WHERE ORDERNR = "1022"
```

### Interface:

INPUT:

```
ORDCHK 1 62  
ORDCHK 2 62
```

OUTPUT:

```
OK: NR:_1:_62:_  
OK: NR:_1:_A:_
```

\_:\_ = Skiljetecken mellan fält  
1 = Antal poster  
62 = Ordernummer  
A = Orderstatus A (order under arbete)

## ORDDSP

Funktionen visar huvudposten på en kundorder, tabell ORDERREG.

Funktionen anropas med parametern ordernr 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 överrider villkoren A och B.

Syntax:

Path/ORDDSP ordernr [databas]

Exempel:

```
$ ./ORDDSP ordernr [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDDSP ordernr

Exempel:

```
$ ./STYRMAN userid ORDDSP ordernr [databas]
```

SQLsats som används:

```
SELECT * FROM ORDERREG WHERE (ORDERNR = "3421")
```

## Interface:

INPUT:

OUTPUT:

```
OK: 01:32 02:4378 03:N 04:A 05:2005-03-14 06:2005-03-14 07:Nya Storkund AB
08:Fina gatan 2 09:100 01 10:LYXBY 11:Sverige 12:Carl von Petersen
13:Storagatan 4B 14:100 02 15:LYXBY 16:Sverige 17:(null) 18:Pelle Chef
19:godsmärke 20:F 21:1 22:001 23:002 24:J 25:J 26:J 27:001 28:25.00
29:SEK 30:001 31:sv 32:2600.00 33:0.00 34:0.00 35:0.00 36:0.00 END
```

OK:	= start på posten
:	= skiljetecken mellan fält
01 – 36	= löpnr för fälten
END	= slut på posten
01	= Ordernr
02	= Kundnr
03	= Ordertyp, (N=Normalorder, D=Direktorder, E=Efterfaktura, F=Förfaktura)
04	= Orderstatus, (A=Under arbete,N=Normal status,F= Frisläppt,B=Slutbehandlad,ska plockas bort)
05	= Orderdatum
06	= Leveransdatum, huvuddatum för ordern
07	= Kundens namn
08	= Adress, postadress,besöksadress
09	= Postnummer
10	= Postadress, ort
11	= Land
12	= Kundens referent
13	= Leveransadress
14	= Postnummer för leveransadressen
15	= Postadress för leveransadressen, ort
16	= Land för leveransadressen
17	= Vår referent
18	= Säljare
19	= Godsmärke
20	= Betalningsvillkorstyp (F=Faktura(Reskontra),P=Postförskott,K=Kontant)
21	= Betalningsvillkor
22	= Leveransvillkor
23	= Leveranssätt
24	= Plocklista (J/N)
25	= Följesedel (J/N)
26	= Fraktavgift (J/N)
27	= Skattekod
28	= Moms i procent (grundvärde för ordern)
29	= Valuta
30	= Exportkod
31	= Språkkod (från kundregistret eller grundvärde om inget angivits)
32	= Ordersumma exklusive moms
33	= Fraktsumma exklusive moms
34	= Moms i kronor (valuta) på fraktkostnaden
35	= Moms totalt på order
36	= Ordersumma inklusive moms

## ORDRDSP

Funktionen visar enkundorders orderrader, tabell ORDERRADREG.

Funktionen anropas med parametern ordernr 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/ORDRDSP ordernr [databas]

Exempel:

\$ ./ORDRDSP ordernr [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDRDSP ordernr

Exempel:

\$ ./STYRMAN userid ORDRDSP ordernr [databas]

SQLsats som används:

```
SELECT * FROM ORDERRADREG WHERE (ORDERNR = "3421")
```

### Interface:

INPUT:

OUTPUT:

```
OK: 1:_:_ 02:_:_ 10:_:_ 03:_:_ 4377:_:_ 04:_:_ N:_:_ 05:_:_ 1000-1001:_:_  
06:_:_ Att använda GNU/LINUX:_:_ 07:_:_ 5116:_:_ 08:_:_ 1.00:_:_ 09:_:_ 117.00:_:_  
10:_:_ 117.00:_:_ 11:_:_ 7.02:_:_ 12:_:_ (null)_:_ 13:_:_ (null)_:_ 14:_:_ (null)_:_  
15:_:_ (null)_:_ 16:_:_ (null)_:_ 17:_:_ ST:_:_:_ NEXT:_:_  
END
```

OK: 1_:_	= start på ordern samt antal orderrader
_:_	= skiljetecken mellan fält
01 – 17	= löpnr för fälten
NEXT	= Skiljemarkör mellan orderrader
END	= slut på ordern
01	= (Skulle vara ordernummer)
02	= Radnummer
03	= Kundnummer
04	= Radtyp (N=Normalorder, D=Direktorder, på denna orderrad)
05	= Artikelnummer
06	= Benämning
07	= Leveransvecka för orderrad
08	= Beställt antal
09	= A'pris
10	= Summa, exklusive moms, för orderrad
11	= Moms i kronor
12	= Levererat antal
13	= Restnoterat antal
14	= Radrabatt
15	= Kalkylpris
16	= Leveransdatum, datum då varan levereras
17	= Enhet

## ORDLST

Funktionen listar kundordrar,samtliga fält, från tabell ORDERREG.

Se också ORDLST2.

Funktionen anropas utan parametrar 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/ORDLST [databas]

Exempel:

```
$ ./ORDLST [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDLST

Exempel:

```
$ ./STYRMAN userid ORDLST [databas]
```

SQLsats som används:

```
SELECT * FROM ORDERREG ORDER BY ORDERNR
```

### Interface:

INPUT:

OUTPUT:

```
OK: NR_3_:31_:4377_:N_:A_:2005-03-14_:2005-03-15_:Nya Kund AB_:Provatan 23_:
199 97_:LILLEBY_:Sverige_:Per Karlsson_:Testgatan 21_:199 91_:LILLEBY_:Sverige_:
(null)_:Josef Seljare_:_:F_:1_:001_:002_:J_:J_:J_:001_:25.00_:SEK_:001_:sv_:
2500.00_:90.00_:22.50_:647.50_:3260.00_:32_
```

Beskrivning:

_:	Fältavskiljare
OK: NR_3	Antal poster som hämtats
31	Ordernr
4377	Kundnr
N	Ordertyp (N=Normalorder)
A	Orderstatus (A=Order under arbete)
2005-03-14	Orderdatum
2005-03-15	Leveransdatum
Nya Kund AB	Kundnamn
199 97	Provgatan 23
LILLEBY	Adress
Sverige	Postnr
Per Karlsson	Postadress (Ort)
Testgatan 21	Land
199 91	Er referent
LILLEBY	Leveransadress (Kan vara lika med Adress)
Sverige	Postnr för leveransadress
(null)	Postadress för leveransadress
Josef Seljare	Land för leveransadress
Godsmärke	Vår referent (tomt)
F	Säljare
1	Plats för godsmärke (blankt)
001	Betalningsvillkorstyp
002	Betalningsvillkor
J	Leveransvillkor
J	Leveranssätt
J	Plocklista J/N
001	Följesedel J/N
25.00	Fraktavgift J/N
SEK	Skattekod
001	Moms %
sv	Valuta
2500.00	Exportkod
90.00	Språkkod
22.50	Ordersumma, exklusive moms
647.50	Fraktavkift i kr
3260.00	Moms på frakten
32	Summa moms på ordern
osv ....	Ordersumma inklusive moms
	Ordernr nästa order

## ORDLST2

Funktionen listar kundordrar,fälten ORDERNR,KUNDNR,LEVDATUM,ORDERSTATUS,ORDERTOTAL , från tabell ORDERREG.

Funktionen anropas utan parametrar 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/ORDLST2 [databas]

Exempel:

\$ ./ORDLST2 [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDLST2

Exempel:

\$ ./STYRMAN userid ORDLST2 [databas]

SQLsats som används:

```
SELECT ORDERNR , KUNDNR , LEVDATUM , ORDERSTATUS , ORDERTOTAL  
FROM ORDERREG ORDER BY ORDERNR
```

### Interface:

INPUT:

OUTPUT:

OK: NR\_3\_:31\_:4377\_:2005-03-14\_:A\_:3260.00\_:32

Beskrivning:

_:	Fältavskiljare
OK: NR_3	Antal poster som hämtats

31	Ordernr
4377	Kundnr
2005-03-14	Leveransdatum
A	Orderstatus (A=order under arbete)
3260.00	Ordersumm inklusive moms
32	Nästa ordernr

## ORDUPD

Funktionen uppdaterar angivet val i tabellen ORDERREG.

(val kan vara 1, uppdatera/sätta ORDERSTATUS, 2005-11-18. Kommer att byggas ut.)

Funktionen anropas med parametrarna **val**, **ordernr**, **data** och, som option, [databas].

ORDERSTATUS kan vara **A**, **N**, **F**, **B**.

**A**=Under arbete(default)

**N**= Normal

**F**=Fakturerad

**B**=Klar, kan/ska plockas bort.

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/ORDUPD val ordernr data [databas]

Exempel:

\$ ./ORDUPD 1 52391 F [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORDUPD

Exempel:

\$ ./STYRMAN userid ORDUPD 1 52391 F [databas]

SQLsats som används:

```
UPDATE ORDERREG SET ORDERSTATUS = "F" WHERE ORDERNR = "52391"
```

**Interface:**

INPUT:

1 52391 F

## OUTPUT:

```
fprintf(stderr,"Error: Måste sätta val större än 0!\n");
fprintf(stdout,"OK: ORDUPD Updated %lu rows\n",
        (unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: ORDUPD UPDATE error: %d  %s\n",
        mysql_errno(&my_connection),mysql_error(&my_connection));
fprintf(stderr,"Error: ORDUPD Connection failed\n");
fprintf(stderr,"Error: ORDUPD Connection error %d:  %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## ORADUPD

Uppdatera fältet FAKTURERAT i tabellen ORDERRADREG

Funktionen anropas med parametrarna **ordernr**, **antal poster**, **data** och, som option, [databas].

Funktionen tar **antal poster** och skannar igenom **data** och extraherar ordernr, radnr och antal (antal fakturerat). Rad för rad uppdateras aktuellt ordernr-radnr med fakturerat antal.

Antal levererat är tillsvidare satt till 0(noll).

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/ORADUPD ordernr antal data [databas]

Exempel:

```
$ ./ORADUPD 3 "35|010|1|35|020|2|35|030|3" [databas]
```

| = fältavskiljare.

```
Antal orderrader |ordernr|radnr|antal|ordernr|radnr|antal|ordernr|radnr|antal|
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid ORADUPD

Exempel:

```
$ ./STYRMAN userid ORADUPD 3 "35|010|1|35|020|2|35|030|3" [databas]
```

SQLsats som används:

```
UPDATE ORDERRADREG SET FAKTURERAT = FAKTURERAT + "1" , LEVERERAT =
LEVERERAT + "0"
WHERE ORDERNR = "35" AND ORDERRAD = "010"
```

### Interface:

INPUT:

```
|35|010|1|35|020|2|35|030|3|
```

## OUTPUT:

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

## PICKADD

Funktionen lägger till en post i tabellen PLOCKLISTEREG.

Funktionen anropas med parametern plockdata 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/PICKADD plockdata [databas]

Exempel:

```
$ ./PICKADD plockdata [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PICKADD plockdata

Exempel:

```
$ ./STYRMAN userid PICKADD plockdata [databas]
```

SQLsats som används:

```
INSERT INTO PLOCKLISTEREG
(ORDERNR,ORDERRAD,KUNDNR,ARTIKELNR,BENEMNING,LEVERANSVECKA,BESTELLT,
ATTLEVERERA, LEVERERAT,PLOCKAT,RESTNOTERAT,LEVDATUM,ENHET,PLOCKSTATUS,
PLOCKDATUM) VALUES
("1093","10","4376","1000-1001","Att använda GNU/LINUX","5523","20","20",
"0","7","13","2005-09-30","ST","P","2005-10-20")
```

### Interface:

INPUT:

```
_:_1093_:_10_:_4376_:_1000-1001_:_ Att använda GNU/LINUX_:_5523_:_ 20_:_20_:_0_:_7_:_13_:_ 2005-09-30 _:_ST_:_P_:_2005-10-20 _:_END
```

Input-ordning:

Ordernr, orderradnr, kundnr, artikelnr, benämning, leveransvecka,beställt antal, antal att leverera, levererat antal, plockat antal, restnoterat antal, leveransdatum, plockstatus, plockdatum.

OUTPUT:

OK: PICKADD Inserted 1 rows

**Beskrivning:**

\_ : \_ Fältavskiljare  
END Postavslutare

## PICKDSP

Funktionen visar alla poster som inte har plockstatus = "B" för en order i tabellen PLOCKLISTEREG. Funktionen anropas med parametrarna ordernr (och vid önskemål plockdatum plockstatus ) samt, som option, [databas].

Det går alltså att anropa funktionen med 1, 2, 3 eller 4 parametrar. Ifall parametern plockdatum ej anges gäller dagens datum. Ifall plockstatus inte anges gäller plockstatus B.

Dock måste alltid ordernr anges.

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/PICKDSP ordernr (plockdatum plockstatus ) [databas]

Exempel:

```
$ ./PICKDSP ordernr [plockdatum] [plockstatus] [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PICKDSP ordernr (plockdatum plockstatus )

Exempel:

```
$ ./STYRMAN userid PICKDSP ordernr [plockdatum] [plockstatus] [databas]
```

SQLsats som används:

```
SELECT * FROM PLOCKLISTEREG
WHERE ORDERNR = "1234" AND PLOCKSTATUS <> "P" AND PLOCKDATUM = "2005-10-23"
ORDER BY PLOCKDATUM, ORDERRAD
```

### Interface:

INPUT:

```
63 "2005-10-23" B
```

Input-ordning:

Ordernr, , plockdatum plockstatus.

## OUTPUT:

```
OK: 2:_  
01:15 02:_ 63:_ 03:_ 10:_ 04:_ 4378:_ 05:_ 1000-1016:_ 06:_  
Seagate Baracuda 7200+7 160GB:_ 07:_ 5134:_ 08:_ 3.00:_ 09:_ 1.00:_ 10:_  
-0.00:_ 11:_ 1.00:_ 12:_ 0.00:_ 13:_ 2005-03-31:_ 14:_ ST:_ 15:_ P:_ 16:_  
2005-10-23:_ NEXT:_  
01:20 02:_ 63:_ 03:_ 20:_ 04:_ 4378:_ 05:_ 1000-1017:_ 06:_  
Deskstar 7K400_250:_ 07:_ 5134:_ 08:_ 3.00:_ 09:_ 1.00:_ 10:_  
-0.00:_ 11:_ 1.00:_ 12:_ 0.00:_ 13:_ 2005-03-31:_ 14:_ ST:_ 15:_ P:_ 16:_  
2005-10-23:_ NEXT:_  
END
```

## Beskrivning:

OK: 2	Antal poster som lästs
_:_ 01	Fält nr
_:_	Fältavskiljare
NEXT:_	Postavskiljare
END	Transaktionsavslutare

## PICKLST

Funktionen listar fälten PLOCKNR, ORDERNR, RADNR, PLOCKSTATUS för poster i tabellen PLOCKLISTEREG.

Urvalet gäller poster som ej har plockstatus B eller U (Bortplockas, Utskrivna).

Funktionen anropas med parametern (plockstatus)?, 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/PICKLST plockstatus [databas]

Exempel:

```
$ ./PICKLST plockstatus [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PICKLST ordernr

Exempel:

```
$ ./STYRMAN userid PICKLST plockstatus [databas]
```

SQLsats som används:

```
SELECT PLOCKNR, ORDERNR, ORDERRAD, PLOCKSTATUS FROM PLOCKLISTEREG
WHERE PLOCKSTATUS <> "B" and PLOCKSTATUS <> "U"
```

## **Interface:**

INPUT:

OUTPUT:

Fälten PLOCKNR, ORDERNR, RADNR, PLOCKSTATUS.

OK: NR\_9\_:17\_:63\_:10\_:P\_:16\_:63\_:20\_:P\_:15\_:63\_:10\_:P  
\_:14\_:62\_:10\_:P\_:12\_:62\_:20\_:P\_:18\_:63\_:20\_:P\_:20\_:63\_:20\_:P  
\_:21\_:62\_:10\_:P\_:22\_:62\_:20\_:P\_:

Beskrivning:

\_: Fältavskiljare

## PKDADD

Funktionen uppdaterar databasen, tabell PRODUKTGRUPP, med PRODKLASS, BESKRIVNING och MOMSKOD.

Funktionen anropas med parameterarna prodklass och 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 överrider villkoren A och B.

Syntax:

Path/PKDADD produktklass beskrivning momskod[databas]

Exempel:

```
$ ./PKDADD produktklass beskrivning momskod [databas]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PKADD produktklass beskrivning momskod

Exempel:

```
$ ./STYRMAN userid PKPADD produktklass beskrivning momskod[databas]
```

SQLsats som används:

```
INSERT INTO PRODUKTGRUPP(PRODKLASS,BESKRIVNING,MOMSKOD) VALUES  
("2200","Hårddiskar","MOMS1")
```

## Interface:

INPUT:

OUTPUT:

Utdata från PKDADD är:

```
OK: PKDADD Inserted 1 rows
```

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

Exempel:

```
Error: PKDADD INSERT error: 1062  Duplicate entry '2200' for key 1
```

## PKDDSP

Funktionen visar en produktgrupps information från tabell PRODUKTGRUPP, med PRODKLASS , BESKRIVNING och MOMSKOD

Funktionen anropas med parametern produktsklass, 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/PKDDSP produktsklass [databas]

Exempel:

\$ ./PKPDSP produktsklass [databas]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PKDDSP produktsklass

Exempel:

\$ ./STYRMAN userid PKDDSP produktsklass [databas]

SQLsats som används:

```
SELECT * FROM PRODUKTGRUPP WHERE (PRODKLASS = "12334")
```

## Interface:

INPUT:

produktsklass

OUTPUT:

Utdata från PKDDSP är:

```
OK: PKDDSP Inserted 1 rows
fprintf(stderr,"Error: PKDDSP SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: PKDDSP Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: PKDDSP Connection failed\n");
fprintf(stderr,"Error: PKDDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr,"Error: PKDDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stdout,"OK: ");
```

```
fprintf(stdout,"01:%s  ",sqlrow[0]);
fprintf(stdout,"02:%s  ",sqlrow[1]);
fprintf(stdout,"03:%s  ",sqlrow[2]);
```

Exempel:

Error: Nr för produktklass saknas.

## PKDLST

Funktionen är avsedd att lista produktkoder/produktgrupper/produktklasser.  
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 överrider villkoren A och B.

Syntax:

Path/PKDLST [databas]

Exempel:

\$ ./PKDLST [olfixtst]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PKDLST

Exempel:

\$ ./STYRMAN olfix PKDLST [olfix]

SQLsats som används:

```
SELECT * FROM PRODUKTGRUPP ORDER BY PRODKLASS
```

Utdata från PDKLST 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: PKDLST SELECT errno: %d\n",mysql_errno(&my_connection));
```

```
fprintf(stdout,"OK: NR_%lu_:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stdout,"%s:",sqlrow[field_count]);
fprintf(stderr,"Error: PKDLST Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: PKDLST Connection failed\n"
```

## 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 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/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 Retriev 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]);
```

## PRISDSP

Funktionen är avsedd att hämta/visa priser i PRISLISTA för angiven artikel.  
Funktionen anropas med parametern artikelnr 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/PRISDSP artikelnr [databas]

Exempel:

```
$ ./PRISDSP "1000-1001" [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid PRISDSP artikelnr [databas]

Exempel:

```
$ ./STYRMAN olfix PRISDSP "1000-1001" [olfix]
```

SQLsats som används:

```
SELECT * FROM PRISLISTA WHERE (ARTIKELNR ="1000-1001" )
```

## Interface:

INPUT:

```
"1000-1001"
```

OUTPUT:

```
fprintf(stdout,"OK: ");
fprintf(stdout,"00:%s ",sqlrow[0]);
fprintf(stdout,"01:%s ",sqlrow[1]);
fprintf(stdout,"02:%s ",sqlrow[2]);
fprintf(stdout,"03:%s ",sqlrow[3]);
fprintf(stdout,"04:%s ",sqlrow[4]);
fprintf(stdout,"05:%s ",sqlrow[5]);
fprintf(stdout,"END:");
fprintf(stdout,"\n");

fprintf(stderr,"Warning: PRISDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: PRISDSP SELECT errno: %d %s\n", mysql_errno(&my_connection));
fprintf(stderr,"Error: PRISDSP Connection failed\n");
fprintf(stderr,"Error: PRISDSP Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## PRISUPD

Funktionen är avsedd att uppdatera antingen tabellen PRISLISTA eller tabellen ARTIKELREG, fältnamn ARFPRI (försäljningspris) så att användaren kan uppdatera prislistor eller försäljningspris. Funktionen anropas med parametrarna filnamn och prislista, och som option [databas]. Filnamnet är **pricefile.tmp**. Filen skapas i vederbörandes hemmakatalog (\$HOME/tmp/ av programmet CHGPRISW. Filen raderas efter användning.

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/PRISUPD filnamn prislistenr [databas]

**prislistenr** kan vara ett av 1, 2, 3, 4, 5, A, F (A= alla 1-5, F=försäljningspriset i ARTIKELREG)

Exempel:

```
$ ./PRISUPD /home/jan/tmp/pricefile.tmp 1 [olfixtst]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid filnamn prislistenr [databas]

Exempel:

```
$ ./STYRMAN olfix PRISUPD /home/jan/tmp/pricefile.tmp 1 [olfix]
```

SQLsatser som används:

```
UPDATE IGNORE ARTIKELREG SET ARFPRI = 235.00 WHERE ARTIKELNR = "1000-1001"
```

```
UPDATE IGNORE PRISLISTA SET PRIS1 = 235.00, PRIS2 = 225.00, PRIS3 = 205.00, PRIS4 = 200.00, PRIS5 = 195.50, WHERE ARTIKELNR = "1000-1001"
```

## Interface:

INPUT:

```
/home/jan/tmp/pricefile.tmp 1
```

OUTPUT:

För varje post i pricefile.tmp.

```
fprintf(stderr,"OK: 1:PRISUPD %lu poster uppdaterade. \n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"OK: 2:PRISUPD %lu poster uppdaterade.\n",
(unsigned long)mysql_affected_rows(&my_connection));
fprintf(stderr,"Error: PRISUPD UPDATE error: %d  %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
fprintf(stderr,"Error: PRISUPD Connection failed\n");
fprintf(stderr,"Error: PRISUPD Connection error %d:  %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
```

## **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 [/opt/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 överrider 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 överrider 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 faild\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 överrider 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 överrider 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_:RGTADD_: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 Retriev 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 överrider 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(stderr,"NR_%lu_:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stderr,"%s_:",sqlrow[field_count]);
fprintf(stderr,"Error: RGTLST Retriev 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 överrider 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 Retriev 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
```

## SIEEXPK

Funktionen används för att skapa/hämta kontoplanen för angivet bokföringsår för att skapa en SIE-fil. .

Funktionen anropas med parametern arid (bokföringsår) 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/SIEEXPK arid [databas]

Exempel:

\$ ./SIEEXPK AD [olfix]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid SIEEXPK arid [databas]

Exempel:

\$ ./STYRMAN JAPI SIEEXPK AD [olfixtst]

SQLsats som används;

```
SELECT KTONR, BENAMNING,SRUNR FROM KTOPLAN  
WHERE ARID = "AD" ORDER BY KTONR
```

### Interface:

INPUT:

ARID

OUTPUT:

```
OK: ANTAL_55_  
#KONTO 1210 "Maskiner"  
#SRU 1210 0  
#KONTO 1219 "Ack avskrivningar maskiner"
```

```
#SRU 1219 0
osv .....
fprintf(stdout,"OK: ANTAL_%lu_\n", (unsigned long)mysql_num_rows(res_ptr))
fprintf(stderr,"Error: SIEEXPK Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPK SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: SIEEXPK Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPK Connection failed\n");
fprintf(stderr,"Error: SIEEXPK Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## SIEEXPR

Funktionen används för att skapa/hämta resultat för angivet bokföringsår för att skapa en SIE-fil. .

Funktionen anropas med parametern arid (bokföringsår) 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/SIEEXPR arid [databas]

Exempel:

```
$ ./SIEEXPR AD [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid SIEEXPR arid [databas]

Exempel:

```
$ ./STYRMAN JAPI SIEEXPR AD [olfixtst]
```

SQLsats som används;

```
SELECT KTONR, SUM(BELOPP),DK FROM VERRAD  
WHERE ARID = "AD"  
GROUP BY KTONR,DK";
```

**Interface:**

INPUT:

ARID

OUTPUT:

```
OK: ANTAL_12_  
#UB 0 1220 18750.00  
#UB 0 2081 -300000.00  
....  
#RES 0 4010 117510.00  
#RES 0 5010 48000.00
```

```
osv .....
fprintf(stdout,"OK: ANTAL_%lu_\n", (unsigned long)mysql_num_rows(res_ptr))
fprintf(stderr,"Error: SIEEXPR Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPR SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: SIEEXPR Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPR Connection failed\n");
fprintf(stderr,"Error: SIEEXPR Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## SIEEXPV

Funktionen används för att skapa/hämta verifikat för angivet bokföringsår för att skapa en SIE-fil. .

Funktionen anropas med parametrarna arid (bokföringsår) och serie (innevarande/föregående bokföringsår) samt 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/SIEEXPR arid serie [databas]

Exempel:

```
$ ./SIEEXPV AD 0 [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid SIEEXPR arid serie[databas]

Exempel:

```
$ ./STYRMAN JAPI SIEEXPR AD 0 [olfixtst]
```

SQLsats som används;

```
SELECT VERRAD.VERNR, VERRAD.RADNR, VERHUVUD.VERDATUM,
VERHUVUD.VERTEXT,VERHUVUD.REGDAT,VERRAD.KTONR,VERRAD.BELOPP,
VERRAD.DK
FROM VERHUVUD,VERRAD
WHERE VERHUVUD.ARID = " AD " AND VERRAD.ARID=VERHUVUD.ARID
AND VERHUVUD.VERNR=VERRAD.VERNR
ORDER BY VERRAD.VERNR,VERRAD.RADNR
```

## Interface:

INPUT:

ARID serie

OUTPUT:

```
OK: ANTAL_65_
#VER "0" "1" 2003-07-30" "Lån eget kapital" "2003-07-30"
{
#TRANS 2330  {} 300000.00
#TRANS 2081  {} -300000.00
}
#VER "0" "2" 2003-07-30" "Insättning checkräkningskredit" "2003-07-30"
osv .....

fprintf(stdout,"OK: ANTAL_%lu_\n", (unsigned long)mysql_num_rows(res_ptr))
fprintf(stderr,"Error: SIEEXPV Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPV SELECT errno: %d\n",mysql_errno(&my_connection));
fprintf(stderr,"Error: SIEEXPV Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: SIEEXPV Connection failed\n");
fprintf(stderr,"Error: SIEEXPV Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));
```

## 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));
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 **olfixtst**, testföretag

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

Vilkor B överrider 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  
operativsystems 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\_Retrieval 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\_Retrieval 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\_Retrieval 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 **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/TRHDADD trnsid tid userid trnsdata [databas]

trnsid = 8 tecken

trnstd = 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","trnstd","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 **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/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 **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/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 Retriev 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 **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/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 **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/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 Retriev 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));
```

## TXTLST

Funktionen anropas utan parameter ,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 **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/TXTLST [databas].

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid TXTDSP [databas].

SQLsats;

```
SELECT * FROM TEXTREG ORDER BY TEXTNR
```

### Interface:

INPUT:

OUTPUT:

```
fprintf(stderr,"Error: TXTLST SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"OK: NR_%lu_:",(unsigned long)mysql_num_rows(res_ptr));
fprintf(stderr,"Error: TXTLST: Data saknas! %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: TXTLST Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: TXTLST Connection failed\n");
fprintf(stderr,"Error: TXTLST Connection error %d: %s\n",
        mysql_errno(&my_connection), mysql_error(&my_connection));

fprintf(stdout,"END\n");
```

\_\_ använder som fältavskiljare.

END = slut på data.

## 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 **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/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 [olfixtst]
```

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.

**VARNING!** Tag inte bort användaren OLFIX, då slutar programmet OLFIXW att fungera.

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/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 överrider 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 Retriev 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 överrider 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 Retriev 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 valutaid 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 valutaid 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 överrider 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 överrider villkoren A och B.

Syntax:

Path/VALDEL valutaid [databas]

Exempel:

```
$ ./VALDEL SEK [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALDEL valutaid [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 överrider villkoren A och B.

Syntax:

Path/VALDSP valutaid [databas]

Exempel:

```
$ ./VALDSP SEK [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALDSP valutaid [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 Retriev 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 **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/VALLST [databas]

Exempel:

\$ ./VALLST [olfix]

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid VALLST [databas]

Exempel:

\$ ./STYRMAN olfix VALLST [olfixtst]

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 Retriev 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 godkänns 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]

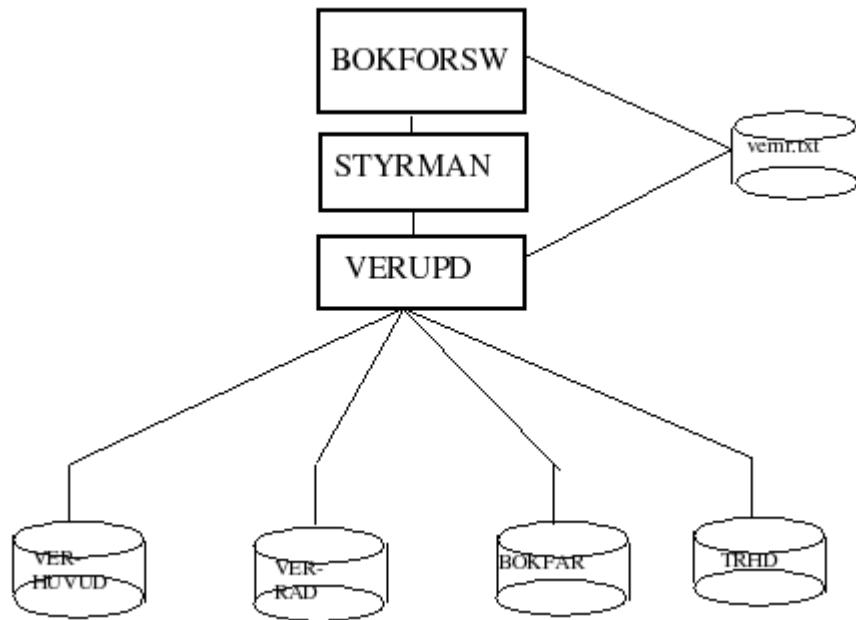
- C. Om USER innehåller strängen **test** eller **prov** så används **olfixtst**.

Vilkor C överrider 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",vrnrfile);
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));

VERNR
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));

```

## **WKUDSP ..... Visa webbkundsdata**

Funktionen anropas med parametrarna KUNDNR och PASSW, som option, med databas.  
Funktionen används för att visa information om en webbkund.  
PASSW är okrypterat.

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/WKUDSP kundnr password [databas]

Exempel:

```
$ ./WKUDSP 4376 webbman [olfix]
```

Normalt görs anropet via STYRMAN och då blir syntaxen:

Path/STYRMAN userid WKUDSP kundnr password [databas]

Exempel:

```
$ ./STYRMAN olfix WKUDSP 4376 webbman [olfixtst]
```

SQLsats som används:

```
SELECT KUNDREG.KUNDNR,KUNDREG.NAMN,KUNDREG.ADRESS,KUNDREG.POSTNR,
KUNDREG.POSTADR,KUNDREG.LAND,KUNDREG.EMAILADR,STDLEVPLATS.ADRESS,
STDLEVPLATS.POSTNR,STDLEVPLATS.POSTADR,STDLEVPLATS.LAND,PASSW.PASSW
FROM KUNDREG
join PASSW,STDLEVPLATS
where KUNDREG.KUNDNR = "4376" and KUNDREG.KUNDNR = PASSW.KUNDNR and
STDLEVPLATS.KUNDNR = KUNDREG.KUNDNR AND STDLEVPLATS.STDLEVPLATS="001" and
PASSW.PASSW LIKE "webbman"
```

## Interface:

INPUT:

KUNDNR PASSW

OUTPUT:

```
fprintf(stderr,"Error: Lösenord måste anges!\n");
fprintf(stderr,"Error: Kundnr saknas.\n");

fprintf(stderr,"Error: WKUDSP Data saknas: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: WKUDSP SELECT error: %s\n",mysql_error(&my_connection));
fprintf(stdout,"OK: ");
fprintf(stdout,"_:%s ",sqlrow[0]);
fprintf(stdout,"_:%s ",sqlrow[1]);
fprintf(stdout,"_:%s ",sqlrow[2]);
fprintf(stdout,"_:%s ",sqlrow[3]);
fprintf(stdout,"_:%s ",sqlrow[4]);
fprintf(stdout,"_:%s ",sqlrow[5]);
fprintf(stdout,"_:%s ",sqlrow[6]);
fprintf(stdout,"_:%s ",sqlrow[7]);
fprintf(stdout,"_:%s ",sqlrow[8]);
fprintf(stdout,"_:%s ",sqlrow[9]);
fprintf(stdout,"_:%s ",sqlrow[10]);
fprintf(stdout,"_:%s ",sqlrow[11]);
fprintf(stdout,"_:%s ",sqlrow[12]);
fprintf(stdout,"\n");
fprintf(stdout,"%s_:",sqlrow[field_count]);
fprintf(stderr,"Error: WKUDSP Retriev error: %s\n", mysql_error(&my_connection));
fprintf(stderr,"Error: WKUDSP Connection failed\n");
fprintf(stderr,"Error: WKUDSP 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 inläggning i databasen och när inläggningen 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	1	H	Posttyp
	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	1	D	Posttyp
	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.

## **BETVILKOR**

Tabell för betalningsvillkor

## **BOKFAR**

Tabell för att administrera bokföringen.

## **DATABAS**

Tabell för olika företag/databaser

## **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.

## **KUNDKATEGORI**

Tabell för beskrivning av kundkategorier

## **KURESK**

Tabell för kundreskontran.

## **LAGERSTELLEREG**

Tabell för kompleterande artikeldata.

## **LEVREG**

Tabell för leverantörer

## **LEVRESK**

Tabell för leverantörsreskontra.

## **LEVSETT**

Tabell för leveranssätt.

## **LEVVILLKOR**

Tabell för leveransvillkor.

## **ORDERREG**

Tabell för kundorder

## **ORDERRADREG**

Tabell för kundorderrader

## **PASSW**

Tabell för password till webbshop

## **PLOCKLISTEREG**

Tabell för att mellanlägra information av plock för kundordrar.

## **PRODUKTGRUPP**

Tabell för produktgrupper/produktklasser

## **PROGRAM**

Tabell för program i OLFIX.

## **PRISLISTA**

Tabell för prislistor.

## **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(60) not null	Benämning, rad 1
ARBENEMNING2	VARCHAR(60)	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','4')DEFAULT '0'	Artikeltyp  0=tillverkning lager 1=tillverkning ej lager 2=köp lager 3=köp ej lager 4=tjänst (för konsulter mm)
ARSTRUKTENUM (' ','B','I','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(60)	Ursprungsbemäning
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

## **DATABAS**

<b>DATABASNR</b>	<b>VARCHAR(3)</b>	Databasnummer, 01 - 99
<b>DATABASTEXT</b>	<b>VARCHAR(15)</b>	Databasens namn enligt MySQL.

Tabellen DATABAS används för att från OLFIXW kunna skifta mellan olika företag.

Skapa tabellen med sqlskriptet “CreateTableDATABAS.sql”.

Ladda tabellen med grunddata med sqlskriptet LoadTableDATABAS.

Grunddata finns i filen TableDATABASdata.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 tomt. 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","Postnr 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”  
“AUTOK”,”Automatkontering J/N”  
"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"  
"EML1","E-mailadress"  
"FAKNR","Senaste fakturanr på kundorder"  
"FKNR2","Senaste fakturanr på kundorder,serie 2"  
"FNAMN","Företagsnamn"  
"FTGNR","Företagsnummer"  
"INKNR","Senast använda inköpsordernummer"  
"INKTO","Konto för inbetalningar, standard"  
"KFKTO","Konto för kundfordringar"  
"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"  
"SKUNR","Senast använda kundnr"  
"SNIKD","Branshtillhörighet"  
"TELEX","Telexnummer"  
"TFAX","Telefaxnummer"  
"TFN1","Telefonnummer till vx"  
"TFN2","Mobiltelefonnummer"  
"TFNMB","Mobiltelefonnummer"  
"TFNVX","Telefonnummer till vx"  
"WKUNR","Senast använda webbkundnummer"

## 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),	
LEVADRESS	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),
LEV BENEMNING	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,
LEVADRESS	VARCHAR(30),
LEVPOSTNR	VARCHAR(6),
LEVPOSTADR	VARCHAR(30),
LEVLAND	VARCHAR(30),
LEVTFNNR	VARCHAR(15),
LEVFAXNR	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),
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),	
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),	
EXPAGIFT	ENUM('J','N')DEFAULT 'J',	
FRAKTAVG	ENUM('J','N')DEFAULT 'J',	
KREDITLEMIT	DECIMAL(10,2),	
KREDITDAGAR	INT,	
KREDITKOD	VARCHAR(3),	
EXPORTKOD	VARCHAR(3),	
SKATTEKOD	VARCHAR(3),	
RABATTKOD	VARCHAR(3),	
DROJMALSRTA	ENUM('J','N')DEFAULT 'J',	
DROJMALSAKTURA	ENUM('J','N')DEFAULT 'J',	
SAMLINGSFAKT	ENUM('J','N')DEFAULT 'J',	
SENASTEKRAVDATUM	DATE,	

SKULD	DECIMAL(10,2),
ORDERSTOCK	DECIMAL(10,2),
PRISLISTA	INT(11)

## KUNDKATEGORI

Skapa med sqlscriptet "CreateTableKUNDKATEGORI.sql"

KATEGORINR	VARCHAR(3) not null PRIMARY KEY	Löpnr
BESKRIVNING	VARCHAR(40) not null	Beskrivning av kundkategori

## KURESK

Tabell för kundreskontran.

Skapa med sqlscriptet "CreateTableKURESK.sql"

ORDERNR	VARCHAR(10) not null	PRIMARY KEY	Ordernr
FAKTURANR	VARCHAR(20) not null,	PRIMARY KEY	Fakturanummer
KUNDNR	VARCHAR(10) not null,		Kundnr
FAKTURADATUM	DATE,		Fakturadatum
EXPIREDATUM	DATE,		Fakturans förfallodatum
NETTOBELOPP	DECIMAL(10,2),		Nettobelopp exkl moms
MOMSBELOPP	DECIMAL(10,2),		Momsbelopp i kronor
FAKTURABELOPP	DECIMAL(10,2),		Fakturans totalsumma inkl moms
BETALD	enum('J','N') DEFAULT 'N' not null,		Fakturan betald J/N
BETALDATUM	date,		Datum då fakturan betalats
USERID	VARCHAR(8),		Userid på den som skapat fakturan
VALUTA	char(3) DEFAULT 'SEK',	Valuta	
VALUTAKURS	DECIMAL(10,2) DEFAULT '1' not null,	Aktuell valutakurs	
VALUTABELOPP	DECIMAL(10,2) DEFAULT '0,00',	Fakturans belopp i angiven valuta	
BAR	VARCHAR(2),		Bokföringsår.
VERNR	INT,		Verifikationsnummer i bokföringen
MOMSKTONR	VARCHAR(4),		Konto där momsen bokförs
KTONR	VARCHAR(4),		Konto där fakturans nettobelopp bokförs
DEBETBELOPP	DECIMAL(10,2)		Belopp som bokförs på KTONR
BETALTBELOPP	DECIMAL(10,2)		Inbetalt belopp

## LAGERSTELLEREG

Skapa med sqlscriptet "CreateTableLAGERSTELLEREG.sql"

ARLAGST	VARCHAR(1) not null PRIMARY KEY	Lagerställe
ARTIKELNR	VARCHAR(30) not null PRIMARY KEY	Artikelnummer/ArtikelID
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)	Inköpsspris, senaste inköp
ARIKVANT0	DECIMAL(10,2)	Inköpskvantitet, senaste inköp
ARIKVANT1	DECIMAL(10,2)	Inköpskavntitet, näst senaste inköp
ARIKVANT2	DECIMAL(10,2)	Inköpskvantitet, näst,näst senaste inköp
ARKALKPRIS	DECIMAL(10,2)	Kalkyl pris
ARBESTKVANT	DECIMAL(10,2)	Beställd kvantitet
ARBESTPUNKT	DECIMAL(10,2)	Beställningspunkt
AROMKOST	DECIMAL(10,2)	Omkostnader
RESERVERAT	DECIMAL(10,2)	Reserverat antal

## **LEVPRISER**

Tabellen LEVPRISER skapas med scriptet CreateTableLEVPRISER.sql.

KUNDNR	VARCHAR (10) NOT NULL default '' PRIMARY KEY	Kundnr och
ARTIKELNR	VARCHAR (30) NOT NULL default '' PRIMARY KEY	Artikelnr utgör tillsammans nyckel för tabellen. Artikelnummer = leverntörens artikelnummer.
BENEMNING	VARCHAR(60) default '0'	Leverantörens benämning på artikeln.
PRIS	DECIMAL(10,2) default '0'	Leverantörens pris.
MATERIALKLASS	VARCHAR(5) default '0'	Leverantörens produktkod/produktklass mm.
ENHET	VARCHAR(5) default '0'	St, liter, kg, m mm
XREF	VARCHAR(30) default '0'	Korsreferens till artikelnummer i ARTIKELREG.

## **LEVVILLKOR**

Skapa med sqlscriptet "CreateTableLEVVILLKOR.sql"

VILLKORSNR	VARCHAR(3) not null PRIMARY KEY	
VILLKORSTEXT	VARCHAR(150)	

		Idnummer för leveransvillkor
		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



## ORDERREG

Skapa med sqlscriptet "CreateTableORDERREG.sql"

Huvudpost för kundorder. Dessutom skapas en post per orderrad i ORDERRADREG.

ORDERNR	VARCHAR(10) not null PRIMARY KEY	Ordernr
KUNDNR	VARCHAR(10) not null	Kundsnummer
ORDERTYP	ENUM('N','D','E','F')DEFAULT 'N' N=Normalorder,D=Direktorder,E=Efterfaktura,F=Förfaktura	
ORDERSTATUS	ENUM('A','N','F','B')DEFAULT 'A' A=Under arbete,N=Normal status, F=Frisläppt,B=Slutbehandlad, ska plockas bort	
ORDERDATUM	DATE	
LEVDATUM	DATE	Orderns huvudleveransdatum
KUNDNAMN	VARCHAR(60)	Kunnamn
KUNDADRESS	VARCHAR(30)	Ordinarie adress
KUNDPOSTNR	VARCHAR(6)	Ordinarie adress
KUNDPOSTADR	VARCHAR(30)	Ordinarie adress
KUNDLAND	VARCHAR(30)	Ordinarie adress
ERREF	VARCHAR(20)	Kundens referens (KUREG)
LEVADRESS	VARCHAR(30)	Leveransadress
LEVPOSTNR	VARCHAR(6)	Leveransadress
LEVPOSTADR	VARCHAR(30)	Leveransadress
LEVLAND	VARCHAR(30)	Leveransadress
VARREF	VARCHAR(20)	Vår referens (KUREG)
SELJARE	VARCHAR(20)	(KUREG)
GODSMERKE	VARCHAR(100)	Valfri text
BETVILLKTYP	ENUM('F','P','K') not null F=Faktura(Reskontra),P=Postförskott,K=Kontant	
BETVILLKOR	VARCHAR(3) DEFAULT '001'	=001,001,003,... (KUREG)
LEVVILLKOR	VARCHAR(3) DEFAULT '001'	=001,001,003,... (KUREG)
LEVSETT	VARCHAR(3) DEFAULT '001'	=001,001,003,... (KUREG)
PLOCKLISTA	ENUM('J','N') DEFAULT 'J'	
FOLJESEDEL	ENUM('J','N') DEFAULT 'J'	
FRAKTAVG	ENUM('J','N') DEFAULT 'J'	
SKATTEKOD	VARCHAR(3) DEFAULT '001' not null	=001,001,003,... (KUREG)
MOMS	DECIMAL(2,2)	I procent
VALUTA	VARCHAR(3) DEFAULT 'SEK, Valutakod,"SEK","USD","EUR" osv (KUREG)	
EXPORTKOD	VARCHAR(3) DEFAULT '001'	=001,001,003,... (KUREG)
SPRAKKOD	VARCHAR(3)DEFAULT 'sv'	Språkkod,"sv","us","en" osv

		(KUREG)
ORDERSUMMA	DECIMAL(10,2)	Exklusive moms
FRAKTSUMMA	DECIMAL(10,2)	Exklusive moms
FRAKTMOMSKR	DECIMAL(10,2)	Moms i kronor
ORDERMOMS	DECIMAL(10,2)	Moms i kronor,order total
ORDERTOTAL	DECIMAL(10,2)	Inklusive moms

## ORDERRADREG

Skapa med sqlscriptet "CreateTableORDERRADREG.sql"

Radpost för kundorder. Det skapas en post för varje orderrad i en kundorder. Dessutom skapas en huvudpost per order i ORDERREG.

ORDERNR	VARCHAR(10) not null PRIMARY KEY	Ordernr
ORDERRAD	INT(4)not null	
KUNDNR	VARCHAR(10) not null	(KUREG)
RADORDERTYP	ENUM('N','D')DEFAULT 'N'	
ARTIKELNR	VARCHAR(30) not null	(ARREG)
BENEMNING	VARCHAR(60)	(ARREG)
LEVERANSVECKA	VARCHAR(5)	
BESTELLT	DECIMAL(10,2) not null	
APRIS	DECIMAL(10,2) not null	(ARREG)
SUMMA	DECIMAL(10,2)not null	
MOMSKR	DECIMAL(10,2) not null	
LEVERERAT	DECIMAL(10,2)	
RESTNOTERAT	DECIMAL(10,2)	
RADRABATT	DECIMAL(2,1)	
KALKYLPRIS	DECIMAL(10,2)	
LEVDATUM	DATE	
ENHET	VARCHAR(4)DEFAULT 'ST'	

## **PASSW** utgår

Utgår från och med 2006-03-25

Skapa med sqlscriptet "CreateTablePASSW.sql"

KUNDNR	VARCHAR(10) not null PRIMARY KEY	Kundsnummer
PASSW	VARCHAR(16)	Lösenord, ej crypterat

Tabellen är tänkt att användas i samband med en webbshop. Kundnumret ska vara samma som finns i KUNDREG.

## PLOCKLISTEREG

Skapa med sqlscriptet "CreateTablePLOCKLISTEREG.sql"

PLOCKNR	INT(11) auto increment not null PRIMARY KEY	Plocknummer
ORDERNR	VARCHAR(10)	Ordernr
ORDERRAD	INT(4)	Radnummer
KUNDNR	VARCHAR(10) not null	Kundsnummer
ARTIKELNR	VARCHAR(30)	Artikelnr
BENEMNING	VARCHAR(60)	Artikelbenämning
LEVERANSVECKA	VARCHAR(5)	Leveransvecka
BESTELLT	DECIMAL(10,2)	Beställt antal
ATTLEVERERA	DECIMAL(10,2)	Antal att leverera
LEVERERAT	DECIMAL(10,2)	Tidigare levererat
PLOCKAT	DECIMAL(10,2)	Plockat antal
RESTNOTERAT	DECIMAL(10,2)	Resterande antal
LEVDATUM	DATE	Lovad leveransdag
ENHET	VARCHAR(4)	Enhet
PLOCKSTATUS	ENUM('P','U','B') default 'P'	Plockstatus (P=plockat, U=Utskrivet, B=Ska Bortrensas)

Tabellen används i samband med plockning av kundorder.

Ordernumret ska vara samma som finns i ORDERREG.

Kundnummer ska vara samma som finns i KUNDREG.

Artikelnummer ska vara samma som finns i ARTIKELREG.

## **PRISLISTA**

Skapa tabellen PRISLISTA med sqlscriptet “CreateTablePRISLISTAsql”

ARTIKELNR	VARCHAR(30) not null PRIMARY KEY	Artikelnr
PRIS1	DECIMAL(10,2) default 0	Prislista nr 1
PRIS2	DECIMAL(10,2) default 0	Prislista nr 2
PRIS3	DECIMAL(10,2) default 0	Prislista nr 3
PRIS4	DECIMAL(10,2) default 0	Prislista nr 4
PRIS5	DECIMAL(10,2) default 0	Prislista nr 5

Innehållet i fälten PRIS1 – PRIS5 skall vara i lämplig valuta. För svenska kunder förslagsvis i SEK.

PRISLISTA är kopplat till både KUNDREG och ARTIKELREG.

Vid uppläggning av en kundorder kontrolleras vilken prislista kunden har. Default = 0, vilket innebär att försäljningspriset (ARFPRI) i ARTIKELREG används.

Ifall en prislista finns angiven tas försäljningspriset från PRISLISTA och angiven prislista. T ex 5.

I fall värdet i prislista nr 5 är tomt eller 0 tas värdet från prislista nr 4. Om värdet i prislista nr 4 också är 0 tas värdet från prislista nr 3, osv.

Om värdet i prislista nr 1 är 0 tas priset från fältet ARFPRI i ARTIKELREG.

## **PRODUKTGRUPP**

Skapa med sqlscriptet "CreateTablePRODUKTGRUPP.sql"

PRODKLASS	VARCHAR(5) not null PRIMARY KEY	Produktgruppsnummer
BESKRIVNING	VARCHAR(30)	Beskrivande text.
MOMSKOD	VARCHAR(5)	Momskod enligt FTGDATA

Tabellen är tänkt att användas för att kunna beskriva vad olika produktgrupper/produktklasser/produktkoder betyder. Använtbart som samlingsnamn/rubrik för produkter av likartad karraktär samt för produktgruppen gemensamma data.

# 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 OLFIW och används av OLFIWX.

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","DELRGWTW"
"009","Administration","Behörighetsadministration","Visa behörighet","", ""
"010","Administration","Behörighetsadministration","Lista behörigheter","LSTRGWTW"
"011","Administration","Funktionsadministration","Ny funktion","ADDFNCFW"
"012","Administration","Funktionsadministration","Lista funktioner","LSTFNCFW"
"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","Kontorrapport","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","ADDLEVW"  
"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 levfakturor","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 sqlskriptet “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

TRNSDATAs 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 korrigeras 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 strukan.

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	martin.jo@home.se
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	be_well@linuxmail.org
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>
Oden Eriksson	<a href="mailto:oeriksson@mandrakesoft.com">oeriksson@mandrakesoft.com</a>
Robert Wollner	robert@reportingtools.net
Jari Kanerva	jari.kan@comhem.se

## 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 MySQLReferenseManual, <http://www.mysql.com/documentation/>

### MySQL Navigator

Följande 2 filer ska finnas i biblioteket /usr/local/mysqlnavigator-version.binary

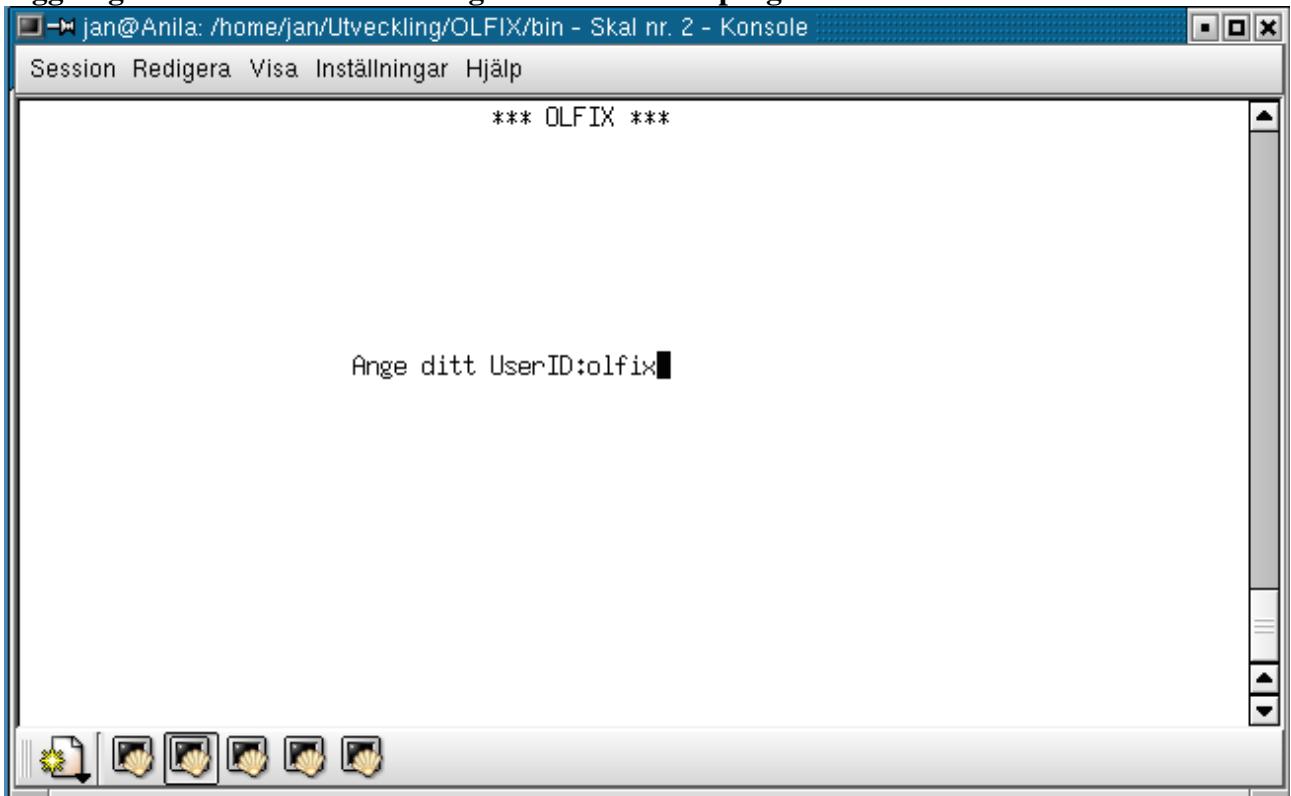
```
libstdc++libc6.2-2.so.3
mysqlnavigator-static
```

MySQL Navigator startas som root med commando;

```
shell> /usr/local/mysqlnavigator-static &
```

## Appendix B

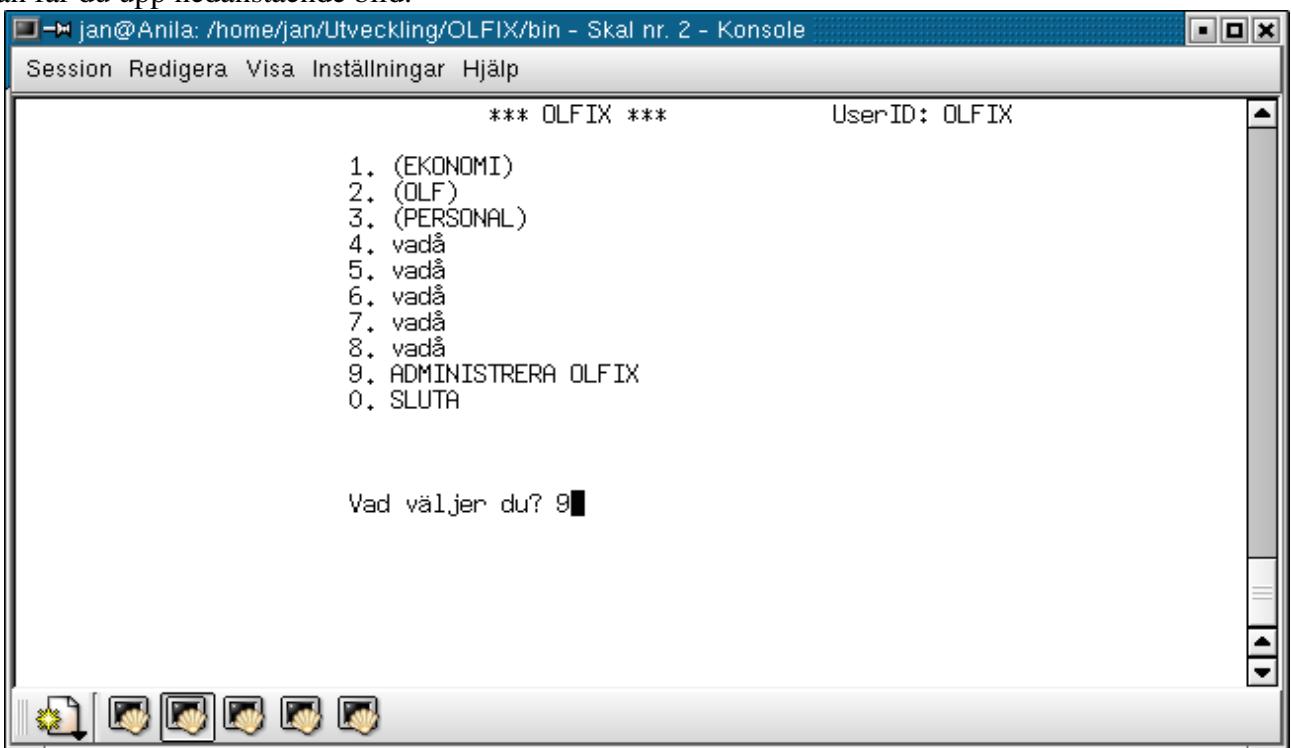
### Uppläggning av användare och behörigheter med konsolprogram.



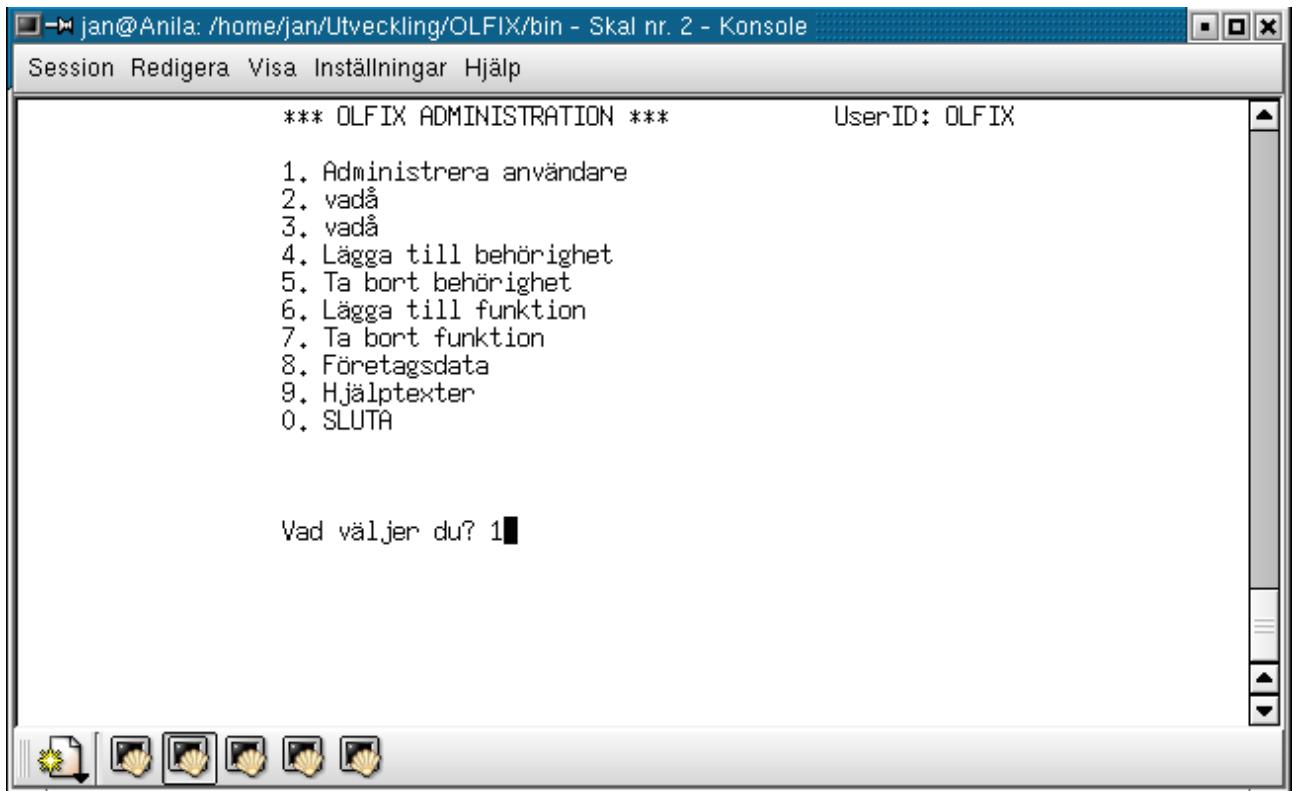
Starta programmet OLFIX och du får upp denna bild.

Som userID **ska** du ange **olfix**. Då får du behörighet att lägga upp användare och behörigheter.

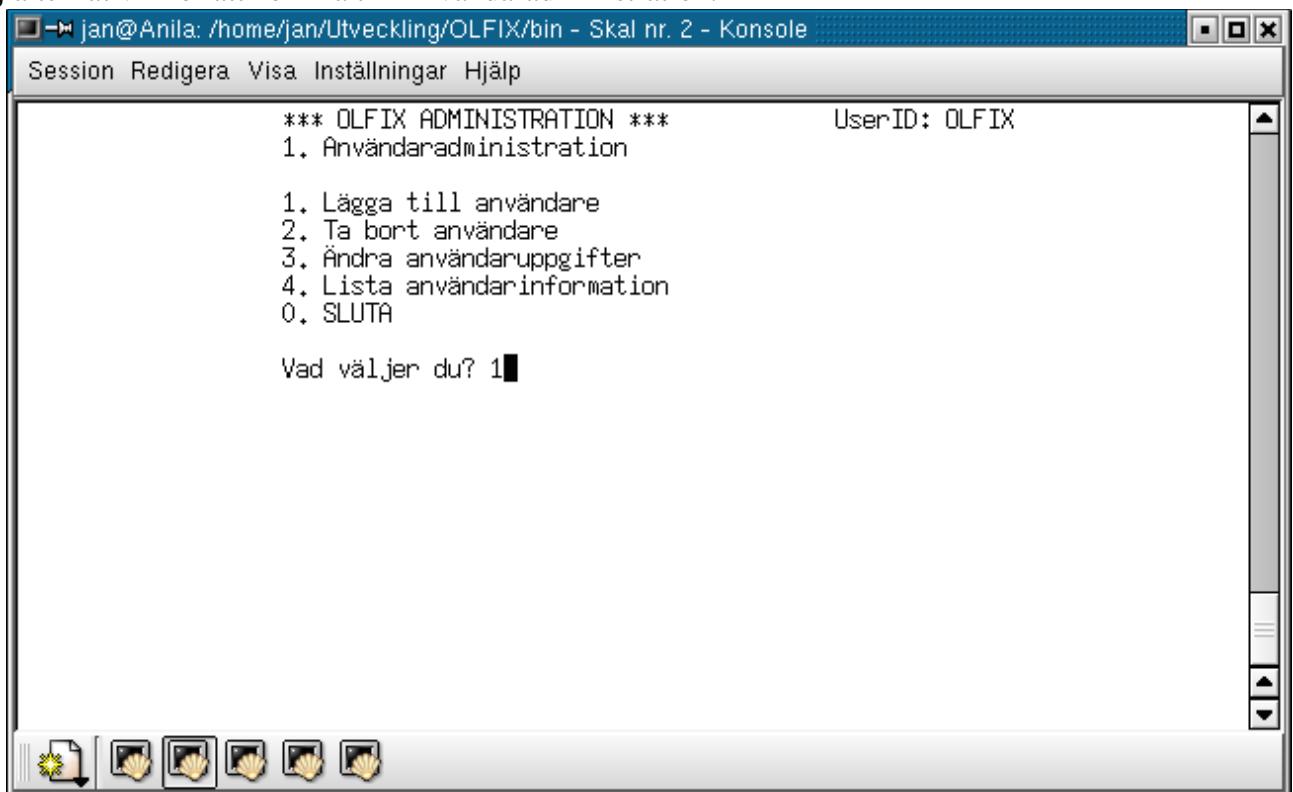
Sedan får du upp nedanstående bild.



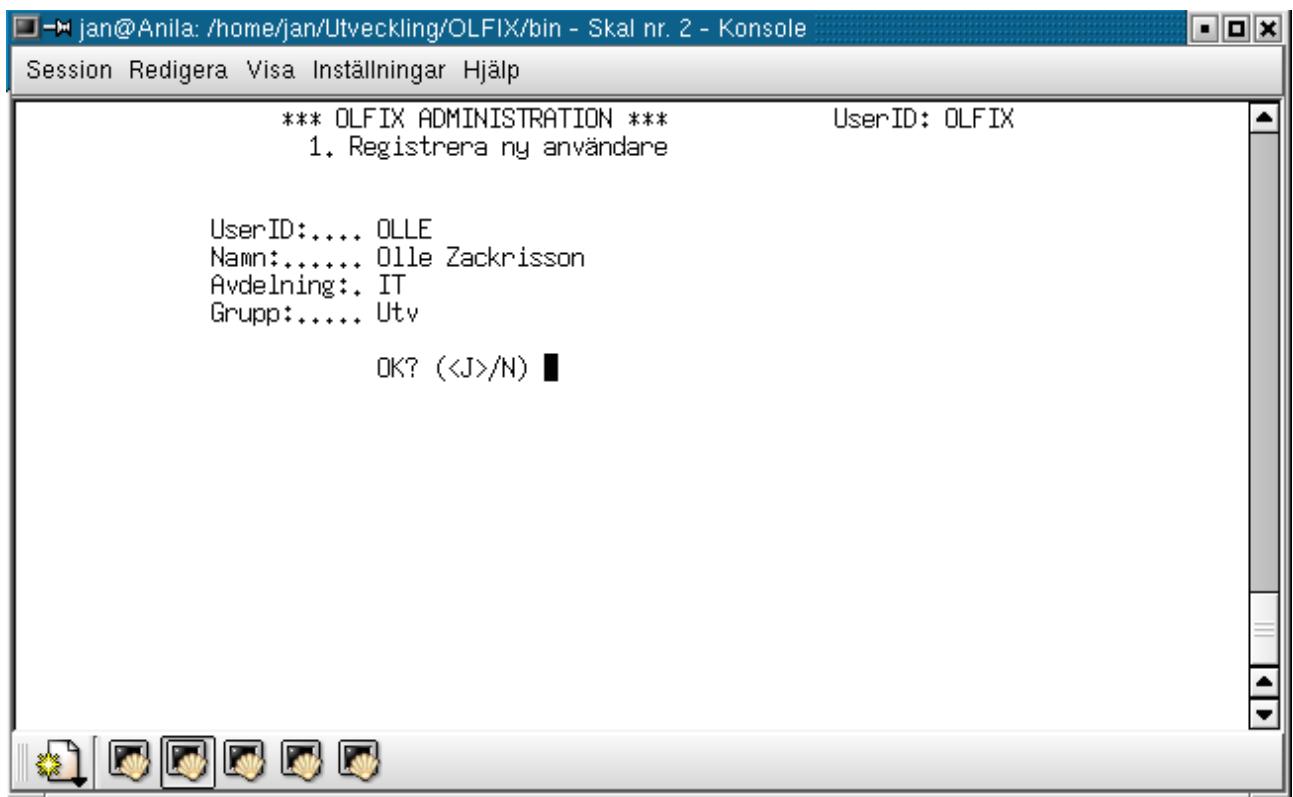
Välj funktion 9 och du kommer till nedanstående bild.



Välj alternativ 1 för att komma till Användaradministration.



Välj alternativ 1 för att lägga upp en ny användare.



Fyll i de uppgifterna.

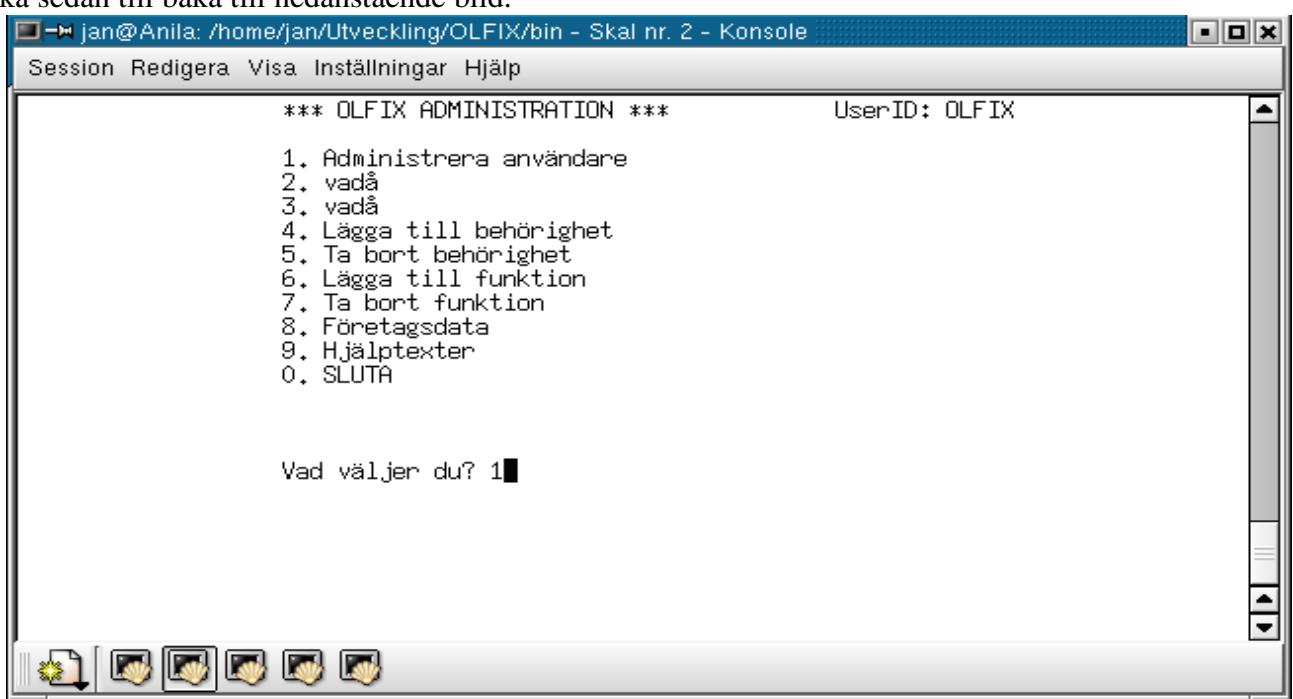
I fältet UserID ska du fylla i det userid som används för att logga in på datorn.

OLFIX kommer nämligen att hämta upp userid på den som loggat in på datorn och jämföra detta mot behörighetslistan. (tabellen RIGHTS i databasen.)

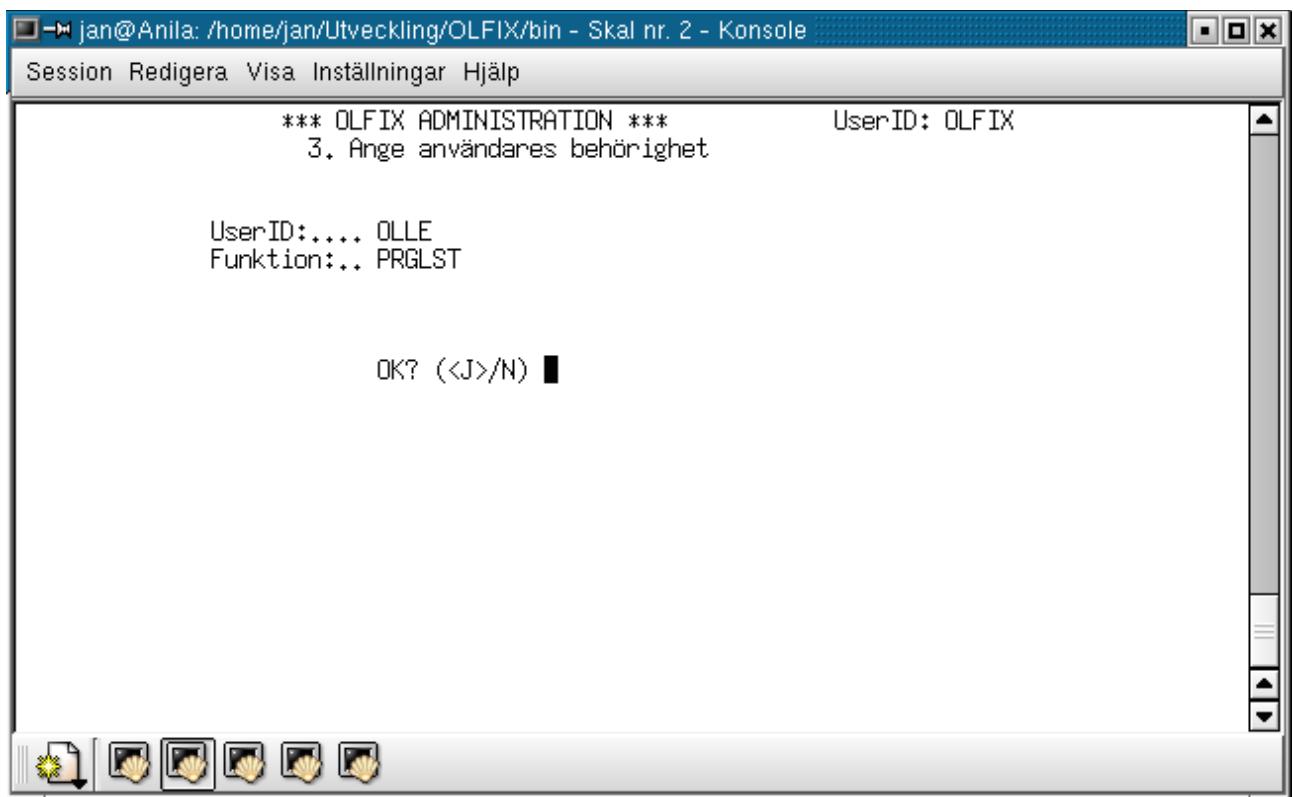
I fältet Namn skriver du in användarens namn.

I fälten Avdelning och Grupp kan du fylla i vad som gäller i din organisation..

Backa sedan till baka till nedanstående bild.



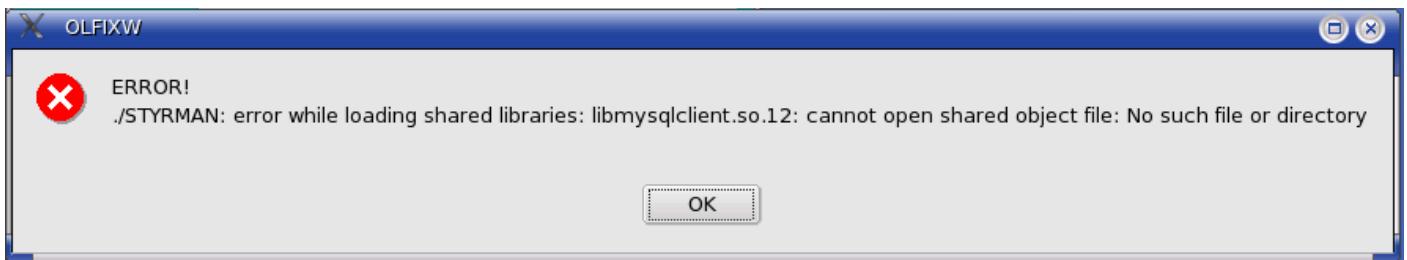
Välj alternativ 4 för att registrera behörigheter.



Här måste du upprepa registreringen för varje program/funktion som användaren ska ha tillgång till.

## Appendix C

### Felmeddelanden



Ovanstående felmeddelande kan man få när man har en senare version av MySQL installerad på sin dator.  
T ex:

Mandrake 9.2 har libmysqlclient.so.12  
Mandriva 2006.0 har libmysqlclient.so.14

När man får felmeddelande behöver man skapa en länk till nuvarande libmysqlclient.so.xx.

Om den nuvarande ”libben” heter libmysqlclient.so.14 kan felet rättas till genom att skapa en länk.  
Kommando:

```
ln -s libmysqlclient.so.14 libmysqlclient.so.12
```

Mer kan läsas i **man ln**



“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 antall 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

## Appendix E. GNU Free Documentation License Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any

mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty

Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

## 0. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this

License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such

parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# INDEX

ADDBARW	27
ADDBETVW	30
ADDFNCW	33
ADDFORW	36
ADDFTGW	39
ADDINKW	43
ADDKTOW	52
ADDKUW	55
ADDLEV PW	63
ADDLEV SW	66
ADDLEV W	59
ADDLEV V W	69
ADDORDW	73
ADDPKDW	80
ADDRGTW	84
ADDTXTW	88
ADDUSRW	91
ADDVALW	94
ADMIN	329
AR2UPD	76, 134, 300, 373
ARADD	361
ARCHK	363
ARDSP	48, 76, 134, 177, 181
ARDSPL	48, 177, 181
ARICHK	375
ARLSPK	367
ARLST	365
ARLSTL	369
ARSRCH	324, 371
ARTIKELREG	576, 579
ATTBET	102, 376
ATTBETW	98
BALRPTW	103
BARADD	29, 378
BARCHG	118, 380
BARCHK	110, 239, 382
BARDSP	110, 118, 184, 311, 384
BARFND	239, 386
BARLST	243, 388
BETADD	32, 389
BETCHG	123, 390
BETDSP	123, 228, 392
BETLST	394
BETVILKOR	576, 581
BOKF	336
BOKFAR	107, 576, 580
BYTFTGW	112
CHGBARW	115
CHGBETVW	120
CHGFTGW	124
CHGKTOW	141
CHGKUW	146
CHGLEV W	127
CHGORDW	131
CHGPRI SW	135
CHGUSRW	138
CHGVALW	150
DAGBOKW	154, 156
DATABAS	576, 582

DBCHK	38, 396
DBOKRPT	156, 397
DELRGTTW	159
DELTXTW	162
DELUSRW	165
DELVALW	169
DSPAREW	178
DSPARW	174, 324
DSPBARW	182
DSPFTGW	185
DSPINKW	188
DSPKSTW	191
DSPKUW	195
DSPLEVW	198
DSPORDW	202
DSPTXTW	207
DSPUSRW	210
DSPVALW	213
FORADD	38, 398
FORCHK	38, 399
FORDSP	38, 400
FORLST	38, 401
FTGADD	402
FTGADM	337
FTGDATA	576, 583
FTGDSP	48, 76, 105, 134, 156, 187, 220, 228, 239, 296, 311, 319, 403
FTGLIS	42, 249, 405
FTGLST	126, 249, 407
FTGUPD	42, 48, 76, 126, 228, 408
HBOKRPT	220, 410
HUVBOKW	217
INKADD	48, 411
INKDSP	190
INKHDSP	304, 417
INKLST	252, 413
INKRADDD	48, 415
INKRADREG	576
INKRADREG	587
INKREG	576
INKREG	585
INKRLST	190, 304, 419
KRESADD	228, 421
KRESLST	234, 423
KSTADD	51, 425
KSTALLE	108, 576, 590
KSTCHK	110, 427
KSTDSP	193, 428
KSTLST	255, 429
KTOADD	54, 430
KTOCHG	144, 432
KTOCHK	110, 240, 434
KTODSP	144, 436
KTOLST	438
KTOPLAN	107, 576, 591
KTORPT	105, 319, 440
KTOUPD	442
KTOVIEW	110, 240, 258, 443
KUADD	58, 76, 445
KUCHG	149, 447
KUCHK	58, 450
KUDSP	76, 149, 197, 452

KUFAKTBW 229  
KUFAKTW 221  
KULST 76, 149, 261, 454  
KUNDKATEGORI 576, 594  
KUNDREG 576, 592  
KURESK 576, 595  
KURESKW 232  
KUSRCH 327, 456  
LAGERSTELLEREG 576, 596  
LEVADD 61, 458  
LEVCHG 130, 461  
LEVDSP 48, 130, 201, 239, 463  
LEVFAKTW 235  
LEVLST 48, 239, 465  
LEVPDSP 76, 467  
LEVPLST 264, 469  
LEVPRISER 597  
LEVREG 577  
LEVREG 588  
LEVRESK 577, 589  
LEVSADD 68, 471  
LEVSDSP 48, 472  
LEVSETT 577, 599  
LEVSLST 267, 474  
LEVVADD 71, 475  
LEVVDSP 48, 476  
LEVVILLKOR 577, 598  
LEVVLST 270, 478  
libmysqlclient.so.12 626  
LRESADD 239, 479  
LRESRPT 481  
LSTBARW 241  
LSTFNCW 244  
LSTFTGW 247  
LSTINKW 250  
LSTKSTW 253  
LSTKTOW 256  
LSTKUW 259  
LSTLEVPW 262  
LSTLEVSW 265  
LSTLEVWW 268  
LSTORDW 271  
LSTPGMW 274  
LSTPKDW 277  
LSTRGTW 280  
LSTTXTW 283  
LSTUSR 212  
LSTUSRW 286  
LSTVALW 289  
OLFIXHLP 38, 149, 197, 292  
OLFIXW 24  
ORADUPD 228, 503  
ORDADD 76, 483  
ORDCHG 134, 485  
ORDCHK 228, 296, 300, 491  
ORDDSP 134, 205, 228, 296, 300, 493  
ORDERRADREG 577, 603  
ORDERREG 577, 601  
ORDLST 497  
ORDLST2 134, 228, 273, 499  
ORDRADD 76, 487

ORDRCHG 134, 489  
ORDRDSP 134, 205, 228, 296, 300, 495  
ORDRUPD 228, 300  
ORDUPD 228, 501  
PASSW 577, 604  
PASSW utgår 604  
PICKADD 300, 505  
PICKDSP 507  
PICKLST 509  
PKDADD 83, 511  
PKDDSP 134, 513  
PKDLST 279, 515  
PKDSP 76  
PLCHGW 298  
PLOCKLISTEREG 577, 605  
PLORDW 294  
PRGLST 26, 29, 32, 35, 38, 42, 48, 51, 54, 58, 61, 65, 68, 71, 76, 83, 86, 90, 93, 97, 102, 105, 110, 118, 123, 130, 134, 140, 144, 149, 152, 156, 161, 164, 168, 172, 177, 181, 184, 187, 190, 193, 197, 201, 205, 209, 212, 216, 220, 228, 234, 239, 243, 246, 249, 252, 255, 258, 261, 264, 267, 270, 273, 276, 279, 282, 285, 288, 291, 296, 300, 304, 307, 311, 319, 324, 327, 358, 517f.  
PRISDSP 76, 134, 519  
PRISLISTA 577, 606  
PRISUPD 137, 521  
PRODUKTGRUPP 577, 607  
PROGRAM 577, 608  
PROGRAM 608  
PRTAPI 105, 319, 523  
PRTINKW 301  
REDOV 343  
RGTADD 86, 524  
RGTCHK 161, 526  
RGTDE 161  
RGTDEL 168, 528  
RGTDSP 168, 212, 530  
RGTLST 282, 532  
RIGHTS 577, 610  
RPTCRE 307, 534  
RPTGENW 305  
RPTSIEW 308  
SDOLISW 312  
SIEEXPK 311, 535  
SIEEXPR 311, 537  
SIEEXPV 311, 539  
SLPADD 58, 65, 541  
SRCHARW 322  
SRCHKUW 325  
STDLEVPLATS 577, 611  
STYRMAN 543  
TEXTREG 578, 612  
TRANSID 578, 613  
TRHD 108, 578, 614  
TRHDADD 228, 300, 545  
TRNSADD 35, 546  
TRNSLST 246, 547  
TXTADD 90, 548  
TXTDEL 164, 549  
TXTDSP 48, 164, 209, 550  
TXTLST 285, 551  
USERADD 93, 552  
USERCHG 554  
USERDEL 168, 556  
USERDSP 48, 134, 140, 168, 212, 558

USERLST	140, 168, 288, 560
USR	578, 615
VALADD	97, 562
VALCHG	152, 564
VALDEL	172, 566
VALDSP	152, 172, 216, 567
VALLST	291, 569
VALUTA	239, 578, 616
VERHDSP	105, 156, 220, 319
VERHUVUD	108, 578, 617
VERRAD	108, 578, 618
VERUPD	110, 240
VERUPD	570
WKUDSP	572
WRREC	111, 240, 574 577f., 607