First Steps With Visual FlagShip 8 for Linux

1. Requirements

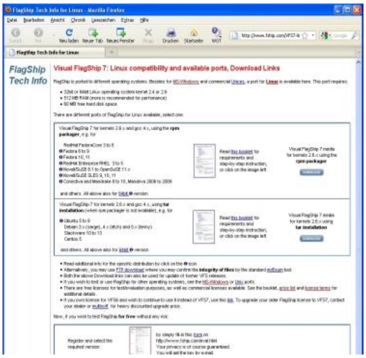
This instruction applies for FlagShip 8 port for Linux by using **tar installation**. The minimal requirements are:

- Linux kernel 3.8 or 4.x and gcc 4.x or newer
- 512 MB RAM (more is recommended for performance)
- 300 MB free hard disk space
- Installed “Development system” package (gcc compiler and tools).

For other Linux system, refer to www.fship.com/linux.html page. FlagShip for Windows is available at www.fship.com/windows.html.

FlagShip 8 (VFS8) supports both 64bit and 32bit Linux systems and creates native 64bit or 32bit executables by using the -64 or -32 compiler switch. The default is the used Linux version. See more in chapter 6.3 below.

2. Download FlagShip


FlagShip Tech Info for Linux - Mozilla Firefox

FlagShip Tech Info

Visual FlagShip 7: Linux compatibility and available ports, Download Links

FlagShip is ported to different operating systems. Besides for MS-Windows and commercial Unixes, a port for Linux is available here. This port requires:

- 32bit or 64bit Linux operating system kernel 2.4 or 2.6
- 512 MB RAM (more is recommended for performance)
- 60 MB free hard disk space

There are different ports of FlagShip for Linux available, select one:

Visual FlagShip 7 for kernels 2.6.x and gcc 4.x, using the **rpm packager**, e.g. for

RedHat FedoraCore 3 to 6
Fedora 6 to 9
Fedora 10, 11
RedHat Enterprise RHEL 3 to 5
Novell/SuSE 8.1 to OpenSuSE 11.x
Novell/SuSE SLES 9, 10, 11
Conectiva and Mandrake 8 to 10, Mandriva 2006 to 2009

and others. All above also for 64bit version.

Read this booklet for requirements and step-by-step instruction, or click on the image left

Visual FlagShip 7 media for kernels 2.6.x using the **rpm packager**

Visual FlagShip 7 for kernels 2.6.x and gcc 4.x, using **tar installation** (when rpm packager is not available), e.g. for

Ubuntu 5 to 9
Debian 3.x (sarge), 4.x (etch) and 5.x (lenny)
Slackware 10 to 13
Centos 5

and others. All above also for 64bit version.

Read this booklet for requirements and step-by-step instruction, or click on the image left

Visual FlagShip 7 media for kernels 2.6.x using **tar installation**

- Read additional info for the specific distribution by click on the icon.
- Alternatively, you may use FTP download where you may confirm the **integrity of files** by the standard md5sum tool.
- Both the above Download links can also be used for update of former VFS releases.
- If you wish to test or use FlagShip for other operating systems, see the MS-Windows or Unix ports.
- There are free licenses for test/evaluation purposes, as well as commercial licenses available. See the booklet, price list and license terms for additional details.
- If you own license for VFS6 and wish to continue to use it instead of VFS7, use this link. To upgrade your older FlagShip license to VFS7, contact your dealer or multisoft for heavy discounted upgrade price.

Now, if you wish to test FlagShip **for free** without any risk:

Register and select the required version

by simply fill-in this form on http://www.fship.com/eval.html
Your privacy is of course guaranteed.
You will get the key by e-mail.

In your preferred Web-Browser, open www.fship.com/linux.html and download the Visual FlagShip setup media for **tar installaton** (gzip-ed tar file) and save it to any folder of your choice (the Linux default is /home/yourName/Downloads directory or ~/Downloads).

Alternatively, you may download FlagShip by FTP (link on the above page), where you may cross-check the authenticity and correct download checksum by *md5sum* tool.

3. Installing FlagShip

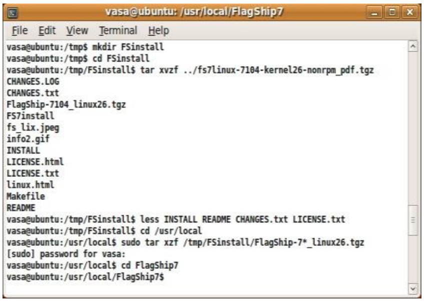
There are only few steps required to install FlagShip. In Linux, open the Terminal-Console, `cd` to your download directory and un-tar the downloaded media there, e.g.

```
cd ~/Downloads
tar xzf fs8linux*-nonrpm.tgz
```

You will get installing instruction and the **FlagShip-8*.tgz** installation file. Read the instruction and license by invoking

```
less INSTALL README \
  CHANGES.txt LICENSE.txt
```

(Note: don't type the backslash, but enter all four file names behind the ***less*** *command. Scroll the view by PgDwn and PgUp keys, select next file by* `:n` *and exit this less viewer by* `:q`*)*


vasa@ubuntu: /usr/local/FlagShip7

```
vasa@ubuntu:/tmp$ mkdir FSinstall
vasa@ubuntu:/tmp$ cd FSinstall
vasa@ubuntu:/tmp/FSinstall$ tar xvzf ../fs7linux-7104-kernel26-nonrpm_pdf.tgz
CHANGES.LOG
CHANGES.txt
FlagShip-7104_linux26.tgz
FS7install
fs_lix.jpeg
info2.gif
INSTALL
LICENSE.html
LICENSE.txt
linux.html
Makefile
README
vasa@ubuntu:/tmp/FSinstall$ less INSTALL README CHANGES.txt LICENSE.txt
vasa@ubuntu:/tmp/FSinstall$ cd /usr/local
vasa@ubuntu:/usr/local$ sudo tar xzf /tmp/FSinstall/FlagShip-7*_linux26.tgz
[sudo] password for vasa:
vasa@ubuntu:/usr/local$ cd FlagShip7
vasa@ubuntu:/usr/local/FlagShip7$
```

The following installation sequence requires log-in as super user (su) or root. An alternative (used e.g. in Ubuntu) is the restricted **sudo** command:

```
cd /usr/local
sudo tar xzf /home/<yourname>/Downloads/FlagShip-8*.tgz
```

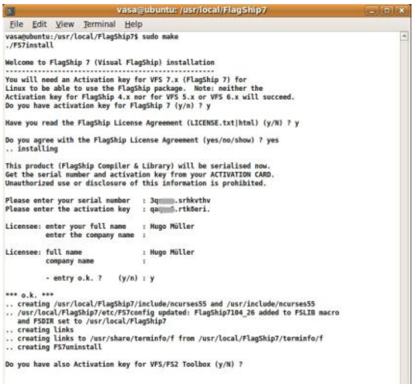
or

```
sudo -i   -or-   su
tar xzf /home/<yourname>/Downloads/FlagShip-8*.tgz
```

Check the activation key you got from multisoft or your distributor, if for Visual FlagShip 8 (abbreviated below by VFS8). If so, activate your FlagShip by

```
cd /usr/local/FlagShip8
sudo make
```

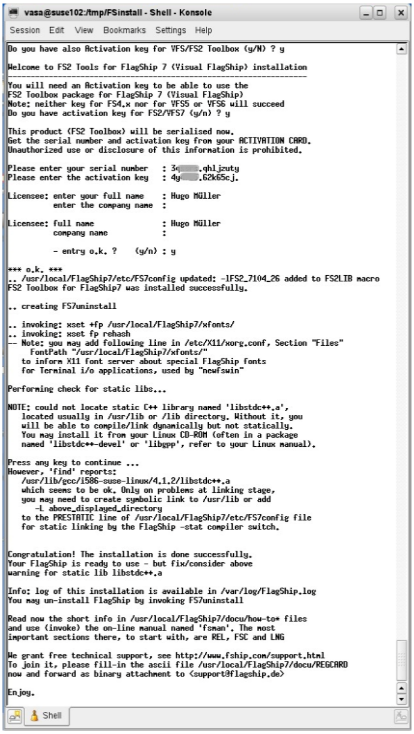
and follow the displayed instruction.


vasa@ubuntu: /usr/local/FlagShip7
File Edit View Terminal Help

```
vasa@ubuntu:/usr/local/FlagShip7$ sudo make
./FS7install

Welcome to FlagShip 7 (Visual FlagShip) installation
-----------------------------------------------------
You will need an Activation key for VFS 7.x (FlagShip 7) for
Linux to be able to use the FlagShip package.  Note: neither the
Activation key for FlagShip 4.x nor for VFS 5.x or VFS 6.x will succeed.
Do you have activation key for FlagShip 7 (y/n) ? y

Have you read the FlagShip License Agreement (LICENSE.txt|html) (y/N) ? y

Do you agree with the FlagShip License Agreement (yes/no/show) ? yes
.. installing

This product (FlagShip Compiler & Library) will be serialised now.
Get the serial number and activation key from your ACTIVATION CARD.
Unauthorized use or disclosure of this information is prohibited.

Please enter your serial number   : 3q[redacted].srhkvthv
Please enter the activation key   : qa[redacted].rtk8eri.

Licensee: enter your full name    : Hugo Müller
          enter the company name  :

Licensee: full name               : Hugo Müller
          company name            :

          - entry o.k. ?    (y/n) : y

*** o.k. ***
.. creating /usr/local/FlagShip7/include/ncurses55 and /usr/include/ncurses55
.. /usr/local/FlagShip7/etc/FS7config updated: FlagShip7104_26 added to FSLIB macro
   and FSDIR set to /usr/local/FlagShip7
.. creating links
.. creating links to /usr/share/terminfo/f from /usr/local/FlagShip7/terminfo/f
.. creating FS7uninstall

Do you have also Activation key for VFS/FS2 Toolbox (y/N) ?
```

If you are upgrading from previous FlagShip versions like FS4, VFS5, VFS6 or VFS7, you will be prompted whether you wish to un-install older FlagShip, or install VFS8 parallel to.

The installer script reports **ok** for successful serializing, otherwise check for typos (both the serial number and activation key are 15 characters long, the dot is a part of the entry; you may use lower or uppercase). The licensee name is mandatory, the company name optional.


vasa@suse102:/tmp/FSinstall - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
Do you have also Activation key for VFS/FS2 Toolbox (y/N) ? y

Welcome to FS2 Tools for FlagShip 7 (Visual FlagShip) installation
------------------------------------------------------------------
You will need an Activation key to be able to use the
FS2 Toolbox package for FlagShip 7 (Visual FlagShip)
Note: neither key for FS4.x nor for VFS5 or VFS6 will succeed
Do you have activation key for FS2/VFS7 (y/n) ? y

This product (FS2 Toolbox) will be serialised now.
Get the serial number and activation key from your ACTIVATION CARD.
Unauthorized use or disclosure of this information is prohibited.

Please enter your serial number   : 3q[redacted].qhljzuty
Please enter the activation key   : 4y[redacted].62k65cj.

Licensee: enter your full name    : Hugo Müller
          enter the company name  :

Licensee: full name               : Hugo Müller
          company name            :

          - entry o.k. ?    (y/n) : y

*** o.k. ***
.. /usr/local/FlagShip7/etc/FS7config updated: -lFS2_7104_26 added to FS2LIB macro
FS2 Toolbox for FlagShip7 was installed successfully.

.. creating FS7uninstall

.. invoking: xset +fp /usr/local/FlagShip7/xfonts/
.. invoking: xset fp rehash
-- Note: you may add following line in /etc/X11/xorg.conf, Section "Files"
     FontPath "/usr/local/FlagShip7/xfonts/"
   to inform X11 font server about special FlagShip fonts
   for Terminal i/o applications, used by "newfswin"

Performing check for static libs...

NOTE: could not locate static C++ library named 'libstdc++.a',
   located usually in /usr/lib or /lib directory. Without it, you
   will be able to compile/link dynamically but not statically.
   You may install it from your Linux CD-ROM (often in a package
   named 'libstdc++-devel' or 'libgpp', refer to your Linux manual).

Press any key to continue ...
However, 'find' reports:
   /usr/lib/gcc/i586-suse-linux/4.1.2/libstdc++.a
   which seems to be ok. Only on problems at linking stage,
   you may need to create symbolic link to /usr/lib or add
      -L above_displayed_directory
   to the PRESTATIC line of /usr/local/FlagShip7/etc/FS7config file
   for static linking by the FlagShip -stat compiler switch.


Congratulation! The installation is done successfully.
Your FlagShip is ready to use - but fix/consider above
warning for static lib libstdc++.a

Info: log of this installation is available in /var/log/FlagShip.log
You may un-install FlagShip by invoking FS7uninstall

Read now the short info in /usr/local/FlagShip7/docu/how-to* files
and use (invoke) the on-line manual named 'fsman'. The most
important sections there, to start with, are REL, FSC and LNG

We grant free technical support, see http://www.fship.com/support.html
To join it, please fill-in the ascii file /usr/local/FlagShip7/docu/REGCARD
now and forward as binary attachment to <support@flagship.de>

Enjoy.
```

If you have purchased also the additional FS2 Toolbox for your VFS8, say “y” and activate also this library now, otherwise enter “n” at the corresponding prompt.

From your entries and paths, the installer creates now un-install script named **FS8uninstall** and creates some symbolic links to /usr/bin and /usr/lib (which are removed at un-installing).

The installer also checks for requirements and displays corresponding warnings if anything is missing. In some cases, you may need to post-install the corresponding package of your Linux distribution.

Note: the displayed messages are available also in a log file named /var/log/FlagShip.log

3.a Updating FlagShip

FlagShip is permanently maintained and often extended with new features. Check frequently the *http://ww.fship.com/whatsnew.html* page for updates. If you already have VFS8 installed, and wish to update to newer sub-release, download the new *fs8inux*.tgz* file, uninstall current version (see step 4 below) and install according to step 3.

4. Uninstalling FlagShip

You may completely remove the FlagShip installation by invoking

```
   su ; FS8uninstall ; exit
```
or
```
   sudo FS8uninstall
```

5. License Types, Support

There are three different license types available, see additional details and prices on *http://www.fship.com/price.html* :

- The free **Test/Evaluation license** is fully functional, and lets you test Flag-Ship with your applications without any risk. The only limitation is the evaluation period of 30 days and the requirement of using the created executables on the same computer where FlagShip is installed. On the registration page *http://www.fship.com/eval.html* you may select FlagShip and FS2 Toolbox for your used environment. Once satisfied with testing, purchase the Personal or Pro license.

- The **Personal** FlagShip (and FS2 Tools) license is intended as a low-cost Starter Kit for personal, company internal/in-house and software developer use, as well as the presentation of applications. Its only limitation is the requirement of using the created executables on the same computer where FlagShip is installed, it allows access to the same shared database by two different users (or processes) simultaneously. You may not pass/sell the created applications (executables) to third parties.

- The FlagShip (and FS2 Tools) **Pro license** does not have any limitation. It is intended as the regular license kit for software developers who will resell their executables, and for large in-house systems, accessed simultaneously by any number of users. You also may pass/sell/distribute the by FlagShip created applications (executables) to anybody else.

There are generally no run-time fees, nor hidden cost. The development package is licensed to you/your company and may not be passed to any third party, but the executables created by the Pro license may freely be sold or passed to anybody. Refer to the License Agreement http://www.fship.com/license.html for full details.

Multisoft (or its distributor) grants **free technical support** for up to **6 months** after purchasing the package, see details on *http://ww.fship.com/support.html* The common support e-mail address is *support@multisoft.de*

6. Using FlagShip

The FlagShip development package contains full compiler, libraries, tools and examples. All are located per default in the /usr/local/FlagShip8 folder and its sub-directories. In addition to, the setup script or makefile (see 3) creates for your convenience some symbolic links to /usr/bin and /usr/lib (documented in /var/log/FlagShip.log and removed by the FS8uninstall script).

As with any compiler, the application program is based on *source file*, which is usual text file (with .prg extension) created by any editor, e.g. vi, emacs, Nedit, JEdit, gedit, kedit and similar program editors. Do not use MS-Word, OpenOffice or similar text processing software, since it often adds special formatting code into the text, which may create you headaches later.

If you are familiar with interpreters like dBase, Foxbase or FoxPro, the difference is that FlagShip compiles the .prg files into native executable instead of interpreting the source, hence you don't need any run-time modules nor need to distribute your sources.

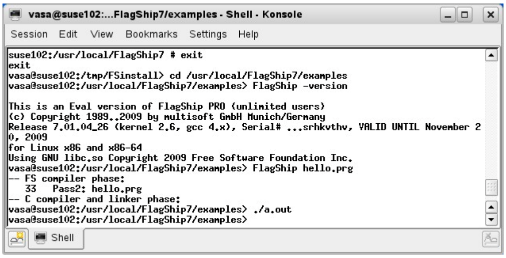
From the same source, FlagShip will create either GUI or textual oriented or basic i/o application. We will learn here in short all of them.

6.1 Invoking the FlagShip compiler

In the Terminal-Console, simply enter

```
FlagShip -version
```

which displays information about the current FlagShip version.


vasa@suse102:...FlagShip7/examples - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
suse102:/usr/local/FlagShip7 # exit
exit
vasa@suse102:/tmp/FSinstall> cd /usr/local/FlagShip7/examples
vasa@suse102:/usr/local/FlagShip7/examples> FlagShip -version

This is an Eval version of FlagShip PRO (unlimited users)
(c) Copyright 1989..2009 by multisoft GmbH Munich/Germany
Release 7.01.04_26 (kernel 2.6, gcc 4.x), Serial# ...srhkvthv, VALID UNTIL November 2
0, 2009
for Linux x86 and x86-64
Using GNU libc.so Copyright 2009 Free Software Foundation Inc.
vasa@suse102:/usr/local/FlagShip7/examples> FlagShip hello.prg
-- FS compiler phase:
   33   Pass2: hello.prg
-- C compiler and linker phase:
vasa@suse102:/usr/local/FlagShip7/examples> ./a.out
vasa@suse102:/usr/local/FlagShip7/examples>
```

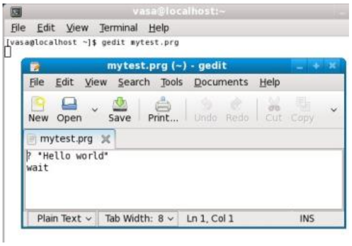
Other FlagShip options may specify the name of source file (or files) and optional compiler switches, described in detail in the on-line manual "fsman" section FSC, see chapter 7 below for details. You will get short summary of switches by **`FlagShip -h`**

6.2 Create your first program

Invoke your editor, e.g. "*gedit mytest.prg*" or "*vi mytest.prg*" etc, enter there

```
? "Hello world"
wait
```

and save it. You alternatively may use the available example


vasa@localhost:~
File Edit View Terminal Help
mytest.prg (~) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy
mytest.prg
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

```
[vasa@localhost ~]$ gedit mytest.prg
```

```
? "Hello world"
wait
```

```
cp /usr/local/FlagShip8/examples/hello.prg mytest.prg
```

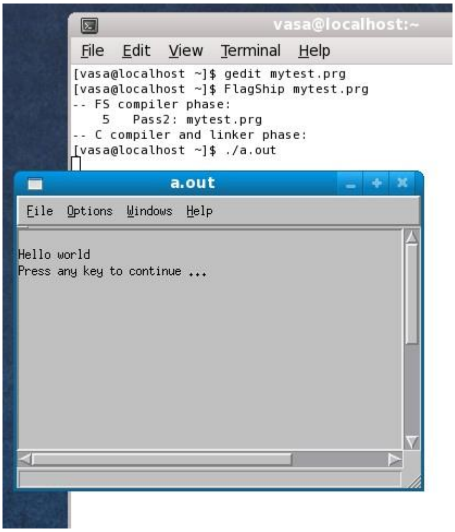
6.3 Compile your program

Enter

```
FlagShip mytest.prg
```

or

```
FlagShip -v mytest.prg
```



```
[vasa@localhost ~]$ gedit mytest.prg
[vasa@localhost ~]$ FlagShip mytest.prg
-- FS compiler phase:
   5    Pass2: mytest.prg
-- C compiler and linker phase:
[vasa@localhost ~]$ ./a.out
```

```
Hello world
Press any key to continue ...
```

This simple command will first compile the .prg file to .c, then creates object file .o and invokes the linker to link it with the library into final, native executable. In our case, the default Linux executable is named ***a.out*** - you however may specify another name by the "*–o exename*" option, as described in the manual section FSC, e.g.

```
FlagShip mytest.prg -o mytest
```

Note: if your application consists of more than one .prg file, you may compile all together by

```
FlagShip mymain.prg myadd1.prg myadd2.prg -o myapplic
```

where the first file is your main and the executable should be named "*myapplic*", or by using wildcards, e.g.

```
FlagShip my*.prg -Mmymain -o myapplic
```

where the –M... option says that the program starts in module (procedure, function, source file) named here *mymain*. You optionally may use also .c and .o files. Of course, compiling via the *make* utility is supported too. All these features are described in detail in the on-line manual “fsman” section FSC, see also chapter 7 below.

The created executable is either **64bit or 32bit** based, depending on the currently used Linux system, check by “**`file a.out`**” or “**`file myapplic`**”. If you wish to create executable for other system, use the `-32` or `-64` compiler switch, see help “**`FlagShip -h`**“ and the on-line manual (`fsman`) section FSC.1 or the corresponding /usr/local/FlagShip8/manual/pdf/FSC.pdf file.

By default, the application is linked *dynamically* using *.so libraries (64bit or 32bit based). If you wish to link *statically*, use the `-stat` compiler switch. This requires availability of static libs (*.a) for some system libraries, see also the manual section FSC.1.7 and the /usr/local/FlagShip8/etc/FS8config* file.

Note: when compiling Foxbase or FoxPro sources, use the **`–fox`** compiler switch (see FSC.1.3) and study the /usr/local/FlagShip8/system/foxpro_api.prg as well as the /usr/local/FlagShip8/include/stdfoxpro.fh files. The most of FoxPro (vers. 2.5 and 2.6) commands are supported, see additional details above and in the manual section APP.

6.4 Execute your first program

Now, invoke your application by

`./a.out`

or

`./mytest` (when `-o mytest` was specified at compiling stage)

This starts the application (executable) in GUI mode, displays the given output text, and waits for Enter key to terminate.

Instead of running it in *GUI mode*, you may force the execution in *Textual (aka Terminal) mode*, known e.g. from Clipper or dBase, by

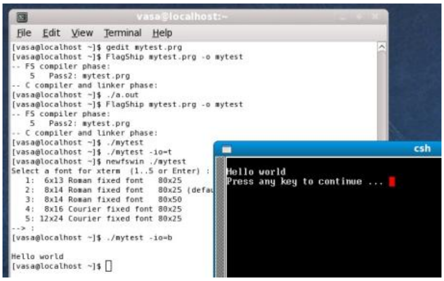
`./mytest -io=t`

or better (for correct color and key translation) by

`newfswin ./mytest`

or in the simplified *Basic i/o mode* by

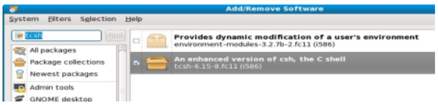
`./mytest -io=b`



```
[vasa@localhost ~]$ gedit mytest.prg
[vasa@localhost ~]$ FlagShip mytest.prg -o mytest
-- FS compiler phase:
   5   Pass2: mytest.prg
-- C compiler and linker phase:
[vasa@localhost ~]$ ./a.out
[vasa@localhost ~]$ FlagShip mytest.prg -o mytest
-- FS compiler phase:
   5   Pass2: mytest.prg
-- C compiler and linker phase:
[vasa@localhost ~]$ ./mytest
[vasa@localhost ~]$ ./mytest -io=t
[vasa@localhost ~]$ newfswin ./mytest
Select a font for xterm  (1..5 or Enter) :
   1:  6x13 Roman fixed font   80x25
   2:  8x14 Roman fixed font   80x25 (defau
   3:  8x14 Roman fixed font   80x50
   4:  8x16 Courier fixed font 80x25
   5: 12x24 Courier fixed font 80x25
--> :
[vasa@localhost ~]$ ./mytest -io=b

Hello world
[vasa@localhost ~]$
```

```
Hello world
Press any key to continue ...
```

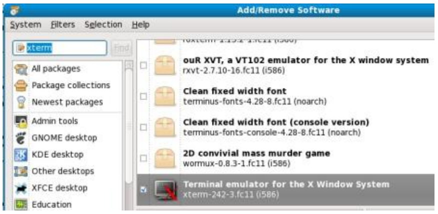
see details about different i/o modes in manual sections LNG.1.2 and LNG.9.7. These switches may be specified also directly at the compiling stage, refer to manual sect. FSC.1.3.



☐ **Provides dynamic modification of a user's environment**
environment-modules-3.2.7b-2.fc11 (i586)

☑ **An enhanced version of csh, the C shell**
tcsh-6.15-8.fc11 (i586)

Note: when the above invocation of **newfswin** script reports error (could not locate xterm or tcsh), you need to install these standard tools from your Linux distribution; they are sometimes not installed per default.



☐ **ouR XVT, a VT102 emulator for the X window system**
rxvt-2.7.10-16.fc11 (i586)

☐ **Clean fixed width font**
terminus-fonts-4.28-8.fc11 (noarch)

☐ **Clean fixed width font (console version)**
terminus-fonts-console-4.28-8.fc11 (noarch)

☐ **2D convivial mass murder game**
wormux-0.8.3-1.fc11 (i586)

☑ **Terminal emulator for the X Window System**
xterm-242-3.fc11 (i586)

Note for Linux beginners: all file names in Linux/Unix are case sensitive. As opposite to DOS/Windows, Linux (and Unix) do not search in the current directory by default (security reason). You therefore need to specify the path of your executable (`./` is the current directory), or add this search path to the PATH environment variable by **`export PATH=$PATH:.`** *which remains valid until next reboot. Alternatively, you may add this line in the* `.bashrc` *file located in your home directory, to set it at next log-in automatically.*

6.5 Compile and execute supplied examples

More interesting programs are available in the /examples directory located in the main FlagShip folder. Invoke

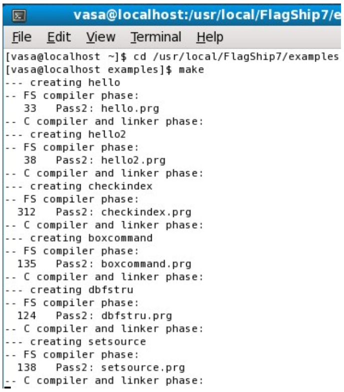
```
cd /usr/local/FlagShip8/examples
```

to select this directory, then start the standard Make utility

```
make
```

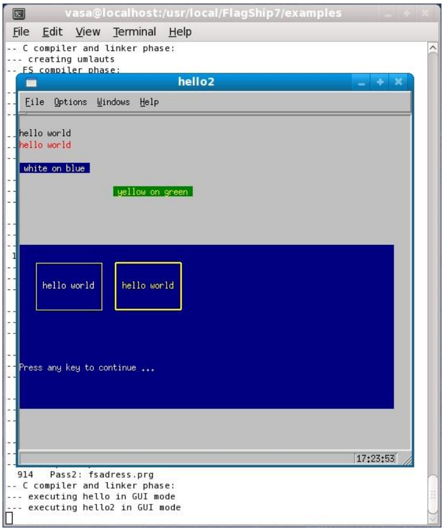
to compile and execute all files there, by using the default template named *Makefile* (described in the manual section FSC.2).

You will get e.g.



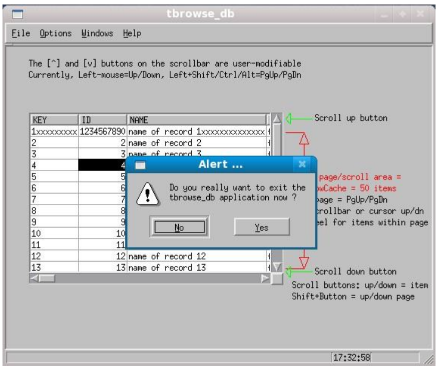
vasa@localhost:/usr/local/FlagShip7/e
File Edit View Terminal Help

```
[vasa@localhost ~]$ cd /usr/local/FlagShip7/examples
[vasa@localhost examples]$ make
--- creating hello
-- FS compiler phase:
  33    Pass2: hello.prg
-- C compiler and linker phase:
--- creating hello2
-- FS compiler phase:
  38    Pass2: hello2.prg
-- C compiler and linker phase:
--- creating checkindex
-- FS compiler phase:
 312    Pass2: checkindex.prg
-- C compiler and linker phase:
--- creating boxcommand
-- FS compiler phase:
 135    Pass2: boxcommand.prg
-- C compiler and linker phase:
--- creating dbfstru
-- FS compiler phase:
 124    Pass2: dbfstru.prg
-- C compiler and linker phase:
--- creating setsource
-- FS compiler phase:
 138    Pass2: setsource.prg
-- C compiler and linker phase:
```



vasa@localhost:/usr/local/FlagShip7/examples
File Edit View Terminal Help
hello2
File Options Windows Help

```
-- C compiler and linker phase:
--- creating umlauts
-- FS compiler phase:

hello world
hello world

white on blue

                yellow on green

  hello world    hello world

Press any key to continue ...
                                                17:23:53
  914    Pass2: fsadress.prg
-- C compiler and linker phase:
--- executing hello in GUI mode
--- executing hello2 in GUI mode
```

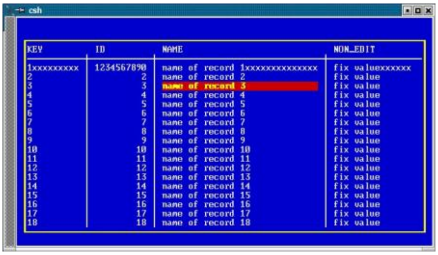


```
The [^] and [v] buttons on the scrollbar are user-modifiable
Currently, Left-mouse=Up/Down, Left+Shift/Ctrl/Alt=PgUp/PgDn

KEY        ID          NAME
1xxxxxxxxx 1234567890  name of record 1xxxxxxxxxxxxxx
2                   2  name of record 2
3                   3  name of record 3
4                   4
5                   5
6                   6
7                   7
8                   8
9                   9
10                 10
11                 11
12                 12  name of record 12
13                 13  name of record 13

Do you really want to exit the
tbrowse_db application now ?

page/scroll area =
wCache = 50 items
age = PgUp/PgDn
rollbar or cursor up/dn
el for items within page

Scroll buttons: up/down = item
Shift+Button = up/down page
                                                17:32:58
```

where the above also shows a prompt when aborting the execution by click on the [X] button at top right (or top left in Gnome).

To run the same applications in textual instead of GUI mode, invoke

```
make terminal
```



```
KEY          ID          NAME                              NON_EDIT

1xxxxxxxxx   1234567890  name of record 1xxxxxxxxxxxxxxx   fix valuexxxxxx
2                     2  name of record 2                  fix value
3                     3  name of record 3                  fix value
4                     4  name of record 4                  fix value
5                     5  name of record 5                  fix value
6                     6  name of record 6                  fix value
7                     7  name of record 7                  fix value
8                     8  name of record 8                  fix value
9                     9  name of record 9                  fix value
10                   10  name of record 10                 fix value
11                   11  name of record 11                 fix value
12                   12  name of record 12                 fix value
13                   13  name of record 13                 fix value
14                   14  name of record 14                 fix value
15                   15  name of record 15                 fix value
16                   16  name of record 16                 fix value
17                   17  name of record 17                 fix value
18                   18  name of record 18                 fix value
```

You of course may compile and/or execute each program separately, according to 6.3 and 6.4 above, the instruction is also given in the header of each source file. When you wish to modify some of these examples, best to copy the source to your working directory first.

6.6 Debugging and testing

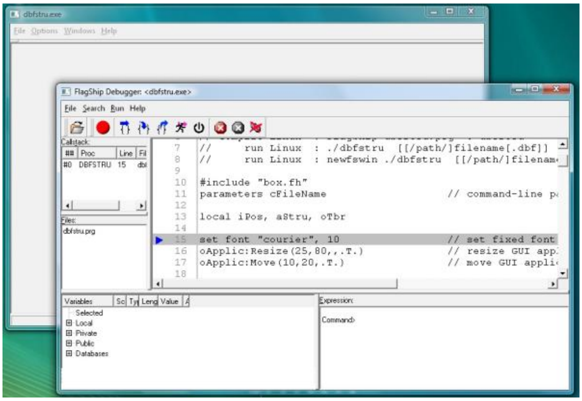
Nearly none of newly developed application is free of typos, syntax and logical errors. Fortunately, most of the problems are detected already at compile and link stage (see manual section FSC.1.8). When using prototyping of variables and the `-w` compiler switch (see manual LNG.2.6.6 and FSC.1.3), also the most misspelled variables are detected at compile-time as well. But neither the compiler nor the linker can detect errors in the program logic.

If your program does not behave as expected, you may

- either use the (common, old fashioned) output of the variable data in interest, e.g. `?# procstack(),“myvar=“,myvar` to display the content of your variable named “myvar” on separate console window incl. the program location and call stack (you may redirect this output to file by e.g. “`myapplic -io=t 2>myapplic.log`” or ”`newfswin myapplic 2\>myapplic.log`”),
- or compile and run the application with FlagShip source-code debugger (see details in manual FSC.5 and below), which gives you all the required information at any program stage.

To avoid large typing here, we will use one of the standard examples to demonstrate the use of FlagShip debugger:

a. Select your working directory
b. Copy the available source and database
```
cp /usr/local/FlagShip8/examples/dbfstru.prg .
cp /usr/local/FlagShip8/examples/tdbedit.dbf .
```
c. Compile the application with debugger
```
FlagShip -d dbfstru.prg -o dbfstru
```
d. Invoke the application
```
./dbfstru
```



```
##  Proc     Line  Fi
#0  DBFSTRU  15    dbl
```

dbfstru.prg

```
 7  //      run Linux  : ./dbfstru  [[/path/]filename[.dbf]]
 8  //      run Linux  : newfswin ./dbfstru  [[/path/]filenam
 9
10  #include "box.fh"
11  parameters cFileName                    // command-line p
12
13  local iPos, aStru, oTbr
14
15  set font "courier", 10                  // set fixed font
16  oApplic:Resize(25,80,,.T.)              // resize GUI app
17  oApplic:Move(10,20,.T.)                 // move GUI appli
18
```

```
Variables   Sc  Ty  Leng  Value
   Selected
 ⊞ Local
 ⊞ Private
 ⊞ Public
 ⊞ Databases
```

The debugger-window pop-ups together with the application, and stops at the first executable statement of your main program (here in line#15 of dbfstru.prg).

You may now set break-points (where the application should stop before executing the statement) by a click on the line number, e.g. on line#39 and line#91.

You may step thru by (or F11, F10), or continue the execution by (or F9) until next breakpoint is reached. The first step button steps over procedure or function, whilst the second steps into (but only if the procedure is available in source code).

The application prompts you now for the database name. Just hit RETURN to see what happens: the debugger stops on the breakpoint at line#39. To observe your used variables, click on the [+] sign for “Local” or “Private” or “Public” variables. You alternatively may enter `len(trim(cfilename))` in the command window, which reports 0 in this case (the string is empty). By F11 or click on the ⇒ button, you will step thru back to line#21 etc. Continue by F9 and enter now `tdbedit` for the file name. The debugger stops again at line#39; you may observe variables anew and/or step thru. Then disable this breakpoint by click on line#39 (the red mark disables), and press F9 to continue. The database structure is displayed in browser, press ESC to exit the browser.

When pressing F2 key in the entry field for database name, the debugger stops at line#91 in *function listdbf()*. You may see the call-stack in the top left debugger window (or by `procstack()` in command window), where item#0 is the current procedure or function. Item#1 shows here the internal representation of code block, double click on this item#1 shows where it was created (set key…to…). Step thru or continue by F9, you will see a list of available databases in the current directory to be displayed, select one by Up/Down and ENTER key, or by mouse click.

When setting breakpoint on line#62, you may observe in debugger also the (or all open) database structure(s) and data by a click on [+] Databases.

To save breakpoints for the next run, either select File → Save Status, and at begin of the next execution select File → Restore Status of the debugger, or specify `export  FSDEBUG_AUTO=ON` in your console (or in ~/.bashrc) to save/restore breakpoints automatically. ESC in the database entry field will terminate the program.

6.7 Other tools

There are additional tools available in the folder */usr/local/FlagShip8/tools* including short textual description and full source code. Some of them are also documented in the manual section FSC.6. Here only some, the most interesting tools:

- **fsmake** creates semi-automatically template for the Make utility for all (or selected) sources of your project(s)
- **dbu** is a database manipulation utility to create and/or manage database structure and data
- **fsi** is a simple interpreter, slightly comparable to dBase or Foxbase. It allows you quick data manipulation or syntax checking etc.
- **indexdump** may help you on problems with indices
- **webtools** provides source for the web*() functions in the standard library

Best to copy the source of interest into your working directory, and compile it there according to the description and header of the source.

In the */usr/local/FlagShip8/system* directory, there are sources and APIs to modify the default behavior of many standard functions or classes. On requirement, copy the source into your working directory, compile there according to header in the source file and link with your application. Use it with care and only if you really know what you are doing.

Many examples are also available in the reference section (CMD, FUN, OBJ) of the FlagShip manual.

7. The FlagShip Manual

This first, short overview of FlagShip cannot and will not replace the full manual. In any FlagShip distribution, there are two complete manuals available:

- the manual in .pdf format (requires e.g. Acrobat reader, or other pdf reader like *mupdf*, *evince* etc) which can also be printed (caution: more than 2.600 pages), available in the /usr/local/FlagShip8/manual/pdf directory, and
- the on-line manual (named *fsman*) is available in /usr/local/FlagShip8/bin/ as a symbolic link to fsman_32 or fsman_64 (set at installation, corresponding to your Linux version) and as symbolic link in /usr/bin/fsman

Both are equivalent in content, however the on-line manual is updated more frequently (on each sub-release), whilst the .pdf manual is updated on main release only. The on-line manual includes Release Notes for the used operating system (Linux, Windows etc), this file is available for printing in text form in the file /usr/local/FlagShip8/manual/relnotes.asc

In your terminal console, simply enter

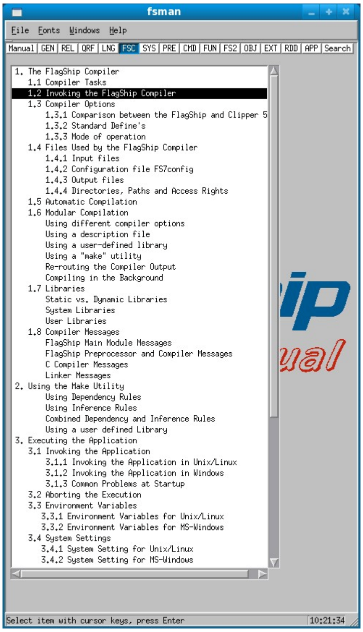
```
fsman
```

to invoke the on-line manual, here displayed in GUI mode. If you haven't X11 running, you may use the textual counterpart by

```
newfscons fsterm
```

Select the required section by click on the top menu (see Manual Contents for overview of sections), then select the chapter in the pop-up window to display the corresponding description. Browse the page(s) by PgDn or PgUp key. The most important sections (displayed at top) are LNG, REL, FSC, CMD and FUN.

Press the ESC key to close the manual description and/or the pop-up window. To exit fsman, press ESC in the main window, or select File → Exit, or click on the [X] top right button.


fsman
File Fonts Windows Help
Manual GEN REL QRF LNG FSC SYS PRE CMD FUN FS2 OBJ EXT RDD APP Search

```
1. The FlagShip Compiler
   1.1 Compiler Tasks
   1.2 Invoking the FlagShip Compiler
   1.3 Compiler Options
       1.3.1 Comparison between the FlagShip and Clipper 5
       1.3.2 Standard Define's
       1.3.3 Mode of operation
   1.4 Files Used by the FlagShip Compiler
       1.4.1 Input files
       1.4.2 Configuration file FS7config
       1.4.3 Output files
       1.4.4 Directories, Paths and Access Rights
   1.5 Automatic Compilation
   1.6 Modular Compilation
       Using different compiler options
       Using a description file
       Using a user-defined library
       Using a "make" utility
       Re-routing the Compiler Output
       Compiling in the Background
   1.7 Libraries
       Static vs. Dynamic Libraries
       System Libraries
       User Libraries
   1.8 Compiler Messages
       FlagShip Main Module Messages
       FlagShip Preprocessor and Compiler Messages
       C Compiler Messages
       Linker Messages
2. Using the Make Utility
       Using Dependency Rules
       Using Inference Rules
       Combined Dependency and Inference Rules
       Using a user defined Library
3. Executing the Application
   3.1 Invoking the Application
       3.1.1 Invoking the Application in Unix/Linux
       3.1.2 Invoking the Application in Windows
       3.1.3 Common Problems at Startup
   3.2 Aborting the Execution
   3.3 Environment Variables
      3.3.1 Environment Variables for Unix/Linux
      3.3.2 Environment Variables for MS-Windows
   3.4 System Settings
      3.4.1 System Setting for Unix/Linux
      3.4.2 System Setting for MS-Windows

Select item with cursor keys, press Enter                10:21:34
```

To search for specific text, use the "Search" section. You may enter there either single word, or word sentence, or words combined by AND or OR, optionally case sensitive. In the pop-up window, select the found page and then press "s" to skip directly to the found word (see also F1 help there).

And now, enjoy FlagShip!