First Steps With Visual FlagShip 7 for Linux 2.6

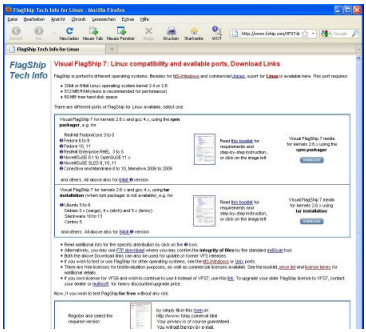
1. Requirements

This instruction applies for FlagShip port for Linux by using **rpm installer**. The common distributions are Novell/SuSE, RedHat, Fedora, Mandriva etc. The minimal requirements are:

- Linux kernel 2.6.x and gcc 4.x
- 512MB RAM (more is recommended for performance)
- 60 MB free hard disk space
- Installed “Development system” package (gcc 4.x 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.

2. Download FlagShip



In your preferred Web-Browser, open www.fship.com/linux.html and download the Visual FlagShip setup media for the **rpm packager** (gzip-ped tar file) and save it to any folder of your choice.

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, create temporary directory and un-tar the downloaded media there, e.g.

```
cd /tmp
mkdir FSinstall
cd FSinstall
tar xvzf /your/download/path/fs7linux*-rpm.tgz
```

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`)

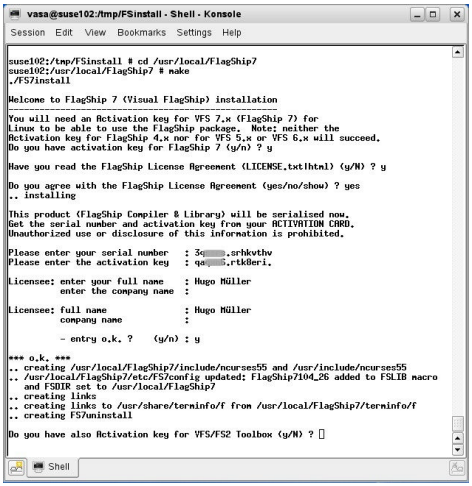
Log-in as superuser (or root) and install the FlagShip media by

```
su
rpm -ih FlagShip-7*.rpm
```

Check the activation key you got from multisoft or your distributor, if for Visual FlagShip 7 (abbreviated below by VFS7). If so, activate your FlagShip (yet as su or root user) by

```
cd /usr/local/FlagShip7
make
```

and follow the displayed instruction.



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

If you have purchased also the additional FS2 Toolbox for your VFS7, 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 `FS7uninstall` 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 FlagShip.log located in the /var/log directory.

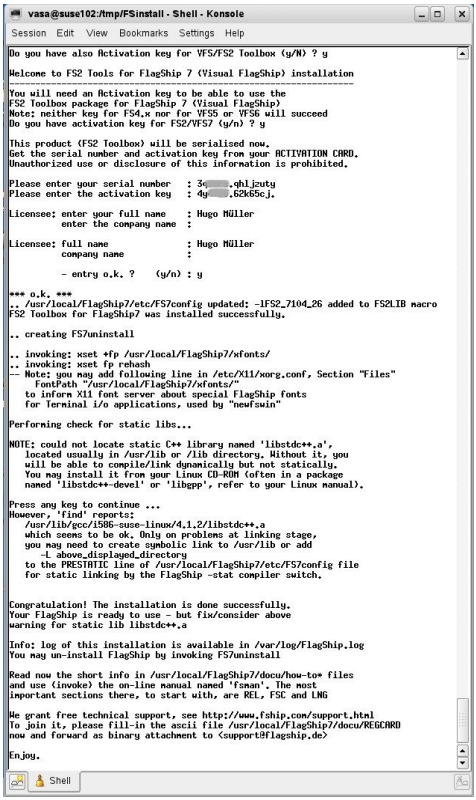
Once the installation finishes, you may remove the temporary directory

```
rm -rf /tmp/FSinstall
```

to save disk space, log-out from the su or root mode by

```
exit
```

and continue (see chapter 6 below) as standard user in your preferred directory.



4. Uninstalling FlagShip

You may completely remove the FlagShip installation by invoking

```
su
FS7uninstall
```

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 FlagShip 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 www.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/FlagShip7 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 FS7uninstall 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.

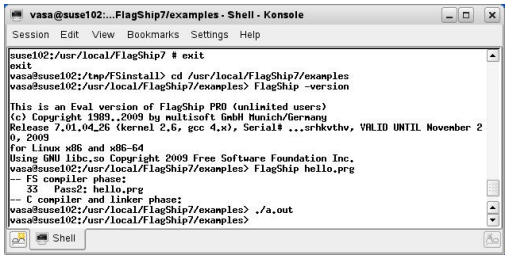
From the same source, FlagShip will create either GUI or textual oriented or basic i/o application. We will learn here in short all of them.

6.1 Invoking the FlagShip compiler

In the Terminal-Console, simply enter

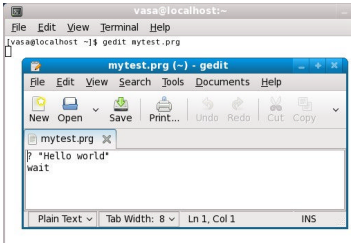
```
FlagShip -version
```

which displays information about the current FlagShip version.



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



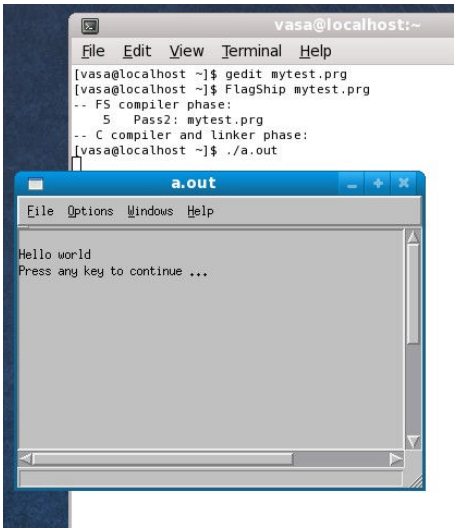
Invoke your editor, e.g. "*gedit mytest.prg*" or *"vi mytest.prg*" etc, enter there

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

and save it. You alternatively may use the available example

```
cp /usr/local/FlagShip7/examples/hello.prg mytest.prg
```

6.3 Compile your program



Enter

```
FlagShip mytest.prg
```

or

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

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.

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 (UDF, procedure) *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 chapter 7 below.

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 mode* (known e.g. from Clipper or dBase) by

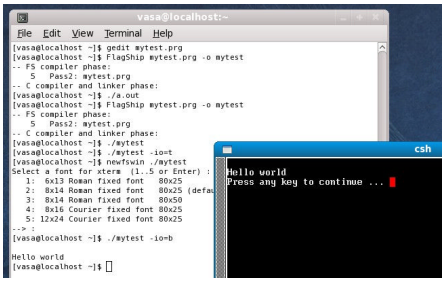
```
./mytest -io=t
```

or better (for correct color and key translation) by

```
newfswin ./mytest
```

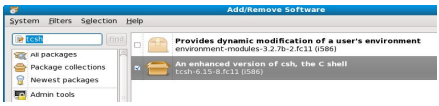
or in the simplified *Basic i/o mode* by

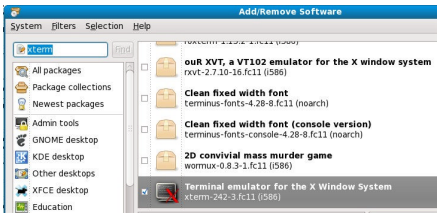
```
./mytest -io=b
```



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.

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.





Note for Linux beginners: all file names in 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

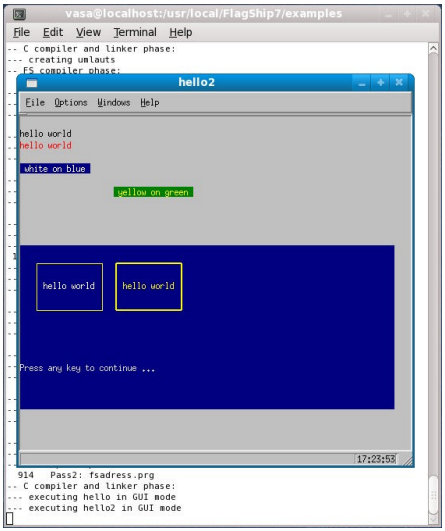
```
cd /usr/local/FlagShip7/examples
```

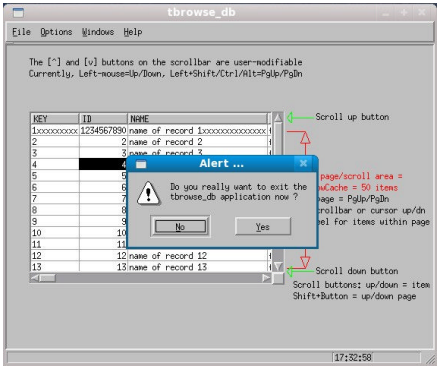
to select this directory, then start the standard Make utility

```
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.

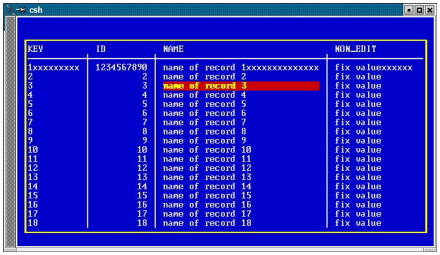




where the above also shows a prompt when aborting the execution by click on the [X] button at top right.

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

```
make terminal
```



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), 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

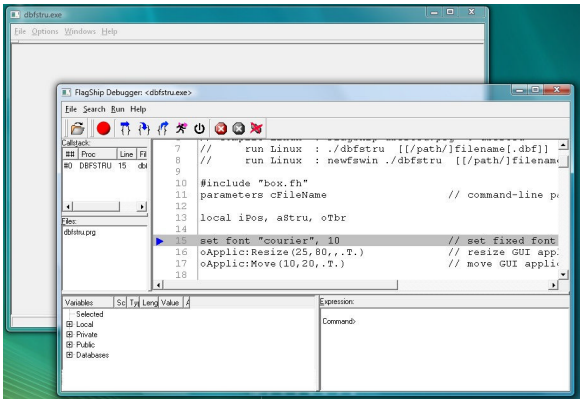
```
cp /usr/local/FlagShip7/examples/dbfstru.prg .
cp /usr/local/FlagShip7/examples/tdbedit.dbf .
```

c. Compile the application with debugger

```
FlagShip -d dbfstru.prg -o dbfstru
```

d. Invoke the application

```
./dbfstru
```



The debugger-window pops-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 breakpoints (where the application should stop before executing the statement) by a click on the line number, e.g. on line#39 and line#91.

Continue the execution, until breakpoint is reached, by F9 or click on the  button. You alternatively may step thru by  buttons (or by F11, F10), where the first steps over procedure or function, whilst the second steps into (but only if the procedure is available in source code).

The program 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/FlagShip7/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
- **foxconvert** is a helper for porting Foxbase and FoxPro sources. Note: when compiling Foxbase or FoxPro sources, use the `-fox` compiler switch and study the /usr/local/FlagShip7/system/foxpro_api.prg as well as the /usr/local/FlagShip7/include/stdfoxpro.fh files

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/FlagShip7/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 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

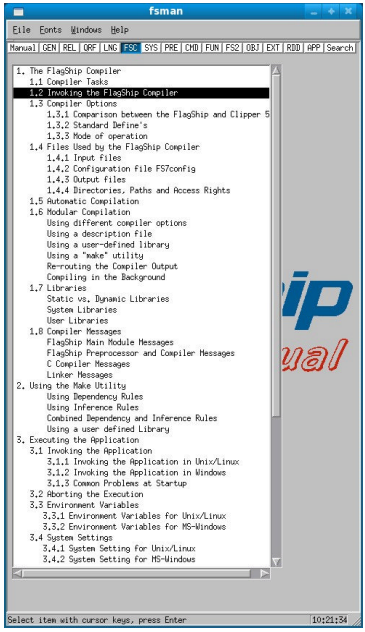
This first, short overview of FlagShip cannot and will not replace the full manual. In any FlagShip distribution, there are two complete manuals available:

- the manual in .pdf format (requires e.g. Acrobat reader) which can also be printed (caution: more than 2.500 pages), available in the /usr/local/FlagShip7/manual/pdf directory, and
- the on-line manual (named *fsman*) is available in /usr/local/FlagShip7/bin/fsman 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/FlagShip7/manual/relnotes.asc

In your 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.

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!