IA 99-36: Installation Guide and Platform Specific Issues

Table of Contents

Table of Contents

Copyright Notice

Mjłlner Informatics Report
MIA 99-36
August 1999

Installation Guide

1 Installation Guide for PC's running Windows NT / 95 / 98 / 2000

This chapter contains the installation guide for the Mjłlner System on Windows NT and Windows 95 using the Microsoft or GNU SDK.

1.1 Minimum Requirements

Requirements for using the Mjłlner System on Windows NT/95/98/2000:

- CPU: 386/486/586/Pentium or compatible.
- RAM: At least 16Mb RAM. 32Mb (or more) recommended.
- Free disk space: Approximately 80Mb
- Linker and standard C libraries from either Microsoft or GNU
- Windows 95/98/2000 or Windows NT 3.5.1/NT 4.0 or later.

1.2 Installation

1.2.1 Run the `Setup` program

To install The Mjłlner System, run the `Setup` Program, which can be found on the Mjłlner System CD in the folder named

```
\windows\<SDK>\Disk1
```

where <SDK> is one of the following: `ms` or `gnu`.

The `Setup` Program installs The Mjłlner System on your system in the directory you specify. Throughout the rest of this document, BETALIB refers to the directory chosen here.

1.2.2 Set up Environment Variables

When the installation is completed you should make sure that the programs and batch files are in your path. If you installed the Mjłlner System in e.g. `C:\BETA`, the batch files are located in `C:\BETA\bin` and the programs are in `C:\BETA\bin\nti_<SDK>`.

BETALIB should point to the location of the installation directory of the Mjłlner System, e.g. `C:\BETA`. If not set, the system uses the path chosen during the installation.

The setup for the Microsoft tools could look something like:

```
set PATH=C:\MSDEV\BIN;%PATH%
set LIB=C:\MSDEV\LIB;
set PATH=C:\BETA\BIN;C:\BETA\BIN\NTI_MS;%PATH%
```

The setup for the Gnu tools could look something like:

```
set GNU=C:\GNU
set PATH=%GNU%\BIN;%PATH%
set PATH=C:\BETA\BIN;C:\BETA\BIN\NTI_GNU;%PATH%
set GCC_EXEC_PREFIX=%GNU%\LIB\GCC-LIB\
set C_INCLUDE_PATH=%GNU%\INCLUDE
set LIBRARY_PATH=%GNU%\LIB
```

You can make Windows NT set the environment variables automatically each time you log in via the System tool in the Control Panel.

You can make Windows 95 set the environment variables automatically each time you log in by setting them in the file c:\autoexec.bat.

If you install the Mjłlner System for more than one SDK, you can use the same location for all installations, as most of the files are shared.

1.3 Installing an SDK

To use the Mjłlner System you must also install one of the following:

- Microsoft Win32SDK including: a linker: `LINK.EXE`, a make program: `NMAKE.EXE` (optional) and C libraries. These can be obtained through e.g. Microsoft Visual C++ v4.0 or later. `NMAKE.EXE` is only needed if the MAKE property is used, which is not recommended.

- GNU Win32SDK including: a linker: `LD.EXE` and `GCC.EXE`, a make program: `GMAKE.EXE` (optional) and C libraries. `GMAKE.EXE` is only needed if the MAKE property is used, which is not recommended.

2.3.1. Installing the Microsoft Win32SDK

You will need to install the Microsoft SDK including all library files.
Also, make sure that you have the LINK.EXE program in your path. This program may not be installed automatically - if this is the case simply locate it on your CD-ROM and copy it to a directory in your path.

2.3.2. Installing the Gnu Win32SDK

You can download the Gnu Win32SDK from the internet. It is available at many sites, including

`http://agnes.dida.physik.uni-essen.de/~janjaap/mingw32/`

Download the packages marked [required]. Then follow the instructions for the installation.

For your convenience we have also included the files used to build the Mjłlner System on the Mjłlner System CD.

1.4 Notes

- You must NOT install the Mjłlner System in a directory containing spaces in the path.
- If you are installing on Windows NT you must have Administrator privileges. Otherwise no program group will be created.
- The file `DelsLog.1` located in BETALIB is a file used by the uninstall mechanism. It should not be deleted.

1.5 Windows Specific Issues

1.5.1 Using Windows Resource Files

You may use Windows resource files in your BETA programs. You do this by specifying a RESOURCE property, e.g.

```
ORIGIN '~beta/basiclib/betaenv';
RESOURCE nti 'foo.res';
-- PROGRAM:descriptor--
(# do ...
#)
```

This specifies, that when linking, the `foo.res` precompiled resource file should be included. You may also specify inclusion of a non-compiled resource file (`foo.rc`). This will make the compiler generate appropriate calls of the resource compiler for the SDK used. If you are using Microsoft SDK, this requires, that you have the `RC.EXE`tool installed, and that it can be found in your path.

1.5.2 Calling External Functions

As shown in, e.g. [MIA 94-24], functions and procedures produced by the C compilers can be called from within BETA. Normally you will just prefix your pattern with external to achieve this. On Windows there are, however, two different ways to call C:

1. If the C function is declared as `__stdcall`, you must specify `callStd`in the do-part of the BETA external declaration, e.g.

    - In C:

      ```
      int __stdcall foo1();
      ```
    - In BETA:

      ```
      foo1: external
        (# result: @integer;
      ```

```
        do callStd;
        exit result
        #);
```

Notice, that many C header files has defined `WINAPI`as an alias for `__stdcall`.

2. Otherwise you can specify `callC`in the do-part. This is default, so leaving out the do-part is equivalent.

1.5.3 Linking without Console

By default the programs are linked in such a way that they always include a console. To avoid this you can use 'switch 71' on the compiler, e.g., run

```
beta -s 71 foo.bet
```

to link `foo.exe` without a console.

1.6 Additional Information for the Windows Implementation

1.6.1 Limitations for the Windows Implementation

See the BETA FAQ:
`http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionVI`

2. Installation guide for UNIX workstations

2.1 Requirements

2.1.1 Sun workstations

- Solaris 2.4 or later
- 16 Mbytes RAM
- X window system (Rel. 11.5 or later)
- 150 Mbytes of available disk space (the full distribution occupies about 100 Mbytes)
- OSF/Motif 1.2 runtime libraries.

2.1.2 Silicon Graphics workstations

- IRIX 6.2 or later (32 bit)
- 16 Mbytes RAM
- X window system (Rel. 11.5 or later)
- 150 Mbytes of available disk space (the full distribution occupies about 100 Mbytes)
- OSF/Motif 1.2 runtime libraries

2.1.3 HP workstations

- HP-PA HP 9000 series HP-UX 9.0 or later
- 16 Mbytes
- X window system (Rel. 11.5 or later)
- 150 Mbytes of available disk space (the full distribution occupies about 100 Mbytes)
- OSF/Motif 1.2 runtime libraries.

2.1.4 PC's running Linux

- CPU: Intel 386/486/586/Pentium
- ELF based Linux
- 16 MB RAM
- X window system (Rel. 11.5 or later)
- LessTif/Motif 1.2 or later*[1]*
- 120 Mbytes of available disk space (the full distribution occupies about 80 Mbytes).

2.2 Installation

The easiest way to install the system is to place it in `/usr/local/lib/beta`, but it can be placed anywhere (see below). The location of the BETA system in your file directory is in the following referred to as `$BETALIB`. By default `$BETALIB` is `/usr/local/lib/beta`.

1. Unpack the tapes (skip this if tar-files have been obtained by ftp):
   The details below may vary on different systems.

   ```
   cd $BETALIB
   tar -xvfb device-file 2000
   ```
   where `device-file` is the UNIX file used for interfacing the tape station, e.g. `/dev/rst8`.
2. Uncompress and untar the system:
   The above command will have extracted a file called `system.tar.Z` or `system.tar.gz` and a number of other compressed tar-files. Each of the compressed files should now be unpacked; use either

   ```
   uncompress system.tar.Z
   tar -xvf system.tar
   ```
   or (if zcat is available)

   ```
   zcat < system.tar.Z | tar -xvf -
   ```
   On Linux systems (or elsewhere, if GNU tar is available) use:

   ```
   tar -xzvf system.tar.gz
   ```
   Repeat this for the other archive files too.
3. Include `$BETALIB/bin` in your search path by including

   ```
   set path=($path $BETALIB/bin)
   ```
   in, e.g., the `.login` file[2].
4. Set `BETALIB` environment variable.
   Set the environment variable `BETALIB` to point to the top directory of your BETA installation, e.g.:

   ```
   setenv BETALIB /usr/local/lib/beta
   ```
   or when using bash on Linux:

   ```
   bash$ BETALIB=/usr/local/lib/beta
   bash$ export BETALIB
   ```
   See also the section on environment variables.

2.3 UNIX Specific Issues

2.3.1 LD_LIBRARY_PATH

On `sgi`, `LD_LIBRARY_PATH` must be set to point to `$BETALIB/lib/sgi` in order to run programs generated using the Mjłlner System.

2.3.2 Dialogs

If you use the window manager (TV)TWM, you can enable automatic placement of dialogs etc. by including the following in your `.twmrc` file:

```
UsePPosition "non-zero"
```

2.3.3 Scripts in $BETALIB/bin

The scripts located in $BETALIB/bin use an internal script to perform some tests for e.g. machine type. This script is placed in

`$BETALIB/configuration/env.sh`

If you have a very special installation, it may be necessary to modify the `env.sh` script according to your system. If the `$BETALIB` is not set, `env.sh` sets it to `/usr/local/lib/beta`.

2.3.4 UNIX man pages

Standard UNIX manual pages for the tools in the Mjłlner System are placed in `$BETALIB/man/man1`. To include these in the search path of the man program, either include `$BETALIB/man/` in your `MANPATH` environment variable, or have your system administrator copy the files in `$BETALIB/man/man1` to the directory containing local manual pages (usually `/usr/man/manl`).

2.4 Additional Information for Linux

See the BETA FAQ:
http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionVIII

2.4.1 Motif on Linux

If you are running Linux, then Motif is normally not included in the standard libraries on your machine. You will then have to buy a separate Motif package (version 1.2 or later) in order to use the graphical libraries included in the Mjłlner System. Alternatively you may try using LessTif, see below.

After you have installed Motif/LessTif, you should change the two environment variables `MOTIFINC`and `MOTIFHOME`set in the `$BETALIB/configuration/env.sh` file to reflect the placement of your Motif/LessTif installation.

2.4.2 LessTif

Lesstif is a free Motif clone. The libraries and source code are free because it is developed on a non-commercial basis. It is available at

http://www.lesstif.org

Since the Mjłlner System uses Motif to draw windows, all errors in Lesstif, will also occur as errors in the Mjłlner System and you have to wait for a Lesstif update or work around it in your BETA program. More information on using LessTif with the Mjłlner System can be found in the LessTif FAQ:

`$BETALIB/doc/xwindows/LessTifFaq/index.html`

2.4.2.1 Installation of Lesstif

- Download Lesstif libraries (0.82 or later). You can download it from

  http://www.lesstif.org

- Install Lesstif libraries according to the Lesstif installation guide.
- Modify the environment variables `MOTIFHOME` and `MOTIFINC` as described above
- Make sure that `ld` (gnu linker) can find the libraries (normally locate in `/usr/local/lib`). Use `ldconfig -v | more` as root to ensure that. Perhaps you have to append `/usr/local/lib` to linker configuration file `/etc/ld.so.conf`
- Recompile a subset of the BETA system: (as the owner of BETALIB):

```
cd $BETALIB/guienv/private/X11
chmod -R u+w .
rm linux/guienv_unix.o
rm linux/Canvas.o
rm linux/Button.o
rm linux/IconButton.o
rm linux/ToggleButton.o
beta guienv_unixbody
chmod -R a-w .
```

- Try to compile a couple of the Lidskjalv demos in `$BETALIB/demo/guienv/`.

2.5 Additional Information for SGI IRIX

2.6.1 Limitations on SGI IRIX

See the BETA FAQ:
http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionIX

2.6 Additional Information for SPARC Solaris

2.6.1 Limitations on SPARC Solaris

See the BETA FAQ:
http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionX

2.7 Additional Information for HP-PA HP-UX

2.7.1 Limitations on HP-PA HP-UX

See the BETA FAQ:
http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionVII

---

[1] Motif is not required to compile simple programs, but programs using the platform independent GUI libraries as well as the graphical tools require motif. Most graphical BETA programs are known to work with LessTif, but at the time of writing this manual, the exact version of LessTif required is not known.

---

[2] Remember to do a rehash if you source your `.login` file manually.

---

3 Installation guide for Power Macintosh

3.1 The Mjłlner System on Macintosh

This page describes how to install and get started using the Mjłlner System on Macintosh.

3.1.2 MPW

The current version of the BETA compiler on the Macintosh is available as part of the mjolner tool ([MIA 99-39] , [MIA 99-40] , [MIA 99-34]) and as an MPW tool. MPW stands for *Macintosh Programmers Workshop*, which is the official programming environment for the Macintosh, developed by Apple, Inc. The BETA compiler runs as an MPW tool, that is, it is a command, that can be invoked from the MPW Shell (command interpreter).

MPW 3.4 or later is needed to use BETA. In addition to the MPW Shell, the compiler uses the MPW `PPCLink` and `Rez` tools to build the programs.

3.2 Requirements

- Power Macintosh (or Performa with RISC processor)
- MacOS 7 or 8
- 40 MB memory recommended.
- At least 60-100 MB disk space (depending on the size of the harddisk, if a HFS partition)
- MPW 3.4 or later

3.3 Installation

3.3.1 Unpack

This distribution is stored as a self-extracting archive. To install the Mjłlner System, double-click the self-extracting archive.

3.3.1.1. Configuring the Files

After the `beta5.0` folder is installed you need to move the file `UserStartup•beta` in the `beta5.0` folder to the MPW folder (the folder containing the MPW Shell).
The purpose of `UserStartup•beta` is to set up the environment used by The Mjłlner System.
When starting MPW the first time after you moved `UserStartup•beta` to the MPW folder, you will be asked to select the beta folder (`beta5.0`). The position of the beta folder will be remembered afterwards using the file `BetaStartup•Home` in the MPW folder. If this file is deleted or you move or rename the beta folder you will be prompted for the beta folder again.

During the first startup `UserStartup•beta` also creates the file `BetaStartup•Menu`.
In `BetaStartup•Menu` the ß menu is defined as described below. You can modify this file to suit your specific needs.
In case it is deleted a new standard `BetaStartup•Menu` is created during the next startup of the MPW.

3.4 Macintosh Specific Issues

3.4.1 Using MPW - Macintosh Programmers Workshop

This section briefly describes how to use MPW for compiling and running BETA applications.

As mentioned above, MPW is a program development environment for the Macintosh computers with tools for editing, compilation and execution of e.g. BETA programs. MPW is a command driven interface to the Macintosh operative system (analogous to the UNIX shell).

MPW is centered around the concept of a worksheet, where the commands are entered and thereafter executed. A worksheet saves its contents from session to session and can therefore be used to contain the most often used commands such that they can be easily executed in later sessions. The commands in the worksheet can be edited using the usual Macintosh editing facilities.[3]

Commands can be executed from the worksheet in two different ways:

1. The text cursor is placed somewhere in the command line and the Enter key [4] is pressed. The entire line will then be executed as one command.
2. Some text may be selected and the Enter key is pressed. The selected text will then be executed as one command.

When starting MPW after installation, an environment is set up to make it easy to use the BETA system. The BETA compiler can be executed either using the script beta or by using the special ß menu.

Notice that you only need to activate the BETA compiler on the program fragment that constitutes the application. If the program fragment uses other fragments (libraries), these are automatically included by the compiler and linker. If some fragments have been changed since the last compilation, these fragments will automatically be recompiled.

3.4.1.1 The ß Menu

The BETA environment defines a ß menu containing items which makes it easy to use the BETA compiler.

The items are:

Compile {Active}

Compile the front most window

Recompile

Compile the previously compiled file again

Compile File...

Prompts the user for a file, and compiles the selected file

Execute

Execute the last compiled file

Open Fragment

Tries to open the selection. The selection is treated as a BETA file name, e.g. `~beta/guienv/guienv`

Directory {Active}

Changes directory to the location of the file shown in the front most window

-Application

Compile the BETA program as an application (default)

-Tool

Compile the BETA program as an MPW Tool

In most cases, the ß menu defines the necessary interface to the BETA compiler. However, the advanced user may prefer to use the beta script instead, please see [MIA 90-02] for legal options etc.

When selecting `Compile {Active}` the beta script is invoked with the active window as argument. The script first executes the BETA compiler and then executes the job file generated by the compiler. The job file links the compiled application. Output during compilation is directed to a separate MPW window.

Application and Tool

The options `-Application` and `-Tool` specify whether the generated application should be a Macintosh stand-alone application or an MPW tool. The default is that the application is linked as a Macintosh stand-alone application. If a BETA program uses the input/output streams `keyboard` and `screen` in `betaenv` and is executed as a stand-alone Macintosh application, a simple console window is opened. The input/output stream is redirected to this window. The console also defines the standard File and Edit menus. The Cut, Copy, and Paste items in the Edit menu are available.

The execution can be stopped by selecting Quit in the console File menu.

Notice, that if the program is using GuiEnv, it must be compiled as a stand-alone application. The input/output is in this case redirected to a special GuiEnv window.

3.4.1.2. Free Download of MPW

October 1997, Apple Computer, inc decided to make MPW freely available for downloading. It can be downloaded from

`ftp://dev.apple.com/devworld/Tool_Chest/Core_Mac_OS_Tools/MPW_etc./MPW-GM_`

The above MacBinary Disk Copy archive contains the two folders

- MPW
- Interfaces&Libraries

Please notice, that new versions may be available when you read this.

3.4.1.3 CodeWarrior MPW

As an alternative to Apple's MPW, an MPW package is included in the MetroWerks CodeWarrior package. Information about MetroWerks can be found at

`http://www.metrowerks.com`

Information about MetroWerks CodeWarrior can be found at

`http://www.metrowerks.com/desktop/mac_os/`

Additional Configuration for CodeWarrior MPW

Using the CodeWarrior version requires additional configuration. The name of the CodeWarrior link tool is `MWLinkPPC`, whereas the name of the linker in the MPW from Apple is `PPCLink`. The BETA compiler must be instructed to use the `MWLinkPPC` tool instead. This is done by inserting the following line in a MPW UserStartup file:

```
set -e betaopts -s 70
```

A MPW UserStartup is a file with a name like `BetaStartup` •`Smith` located in the MPW folder.

An alternative method is to rename the `MWLinkPPC` tool to `PPCLink`.

3.4.2 Getting started on Macintosh

The following steps are usually performed when working with BETA programs under MPW:

1. MPW is started by double clicking the MPW Shell icon or on a BETA source file (`.bet`). This opens MPW with the worksheet document in a

window.

2. We now edit the BETA program text. This may be initiated in two different ways:

    a. We open an existing BETA file by selecting the `Open...` entry in the `File` menu. Or
    b. We create a new file for the BETA program by selecting the `New...` entry in the `File` menu. The file name must end with `.bet`.
3. Now edit the file using the ordinary Macintosh editing facilities in the window containing the program.
4. Save the changed BETA program using the `Save` entry in the `File` menu.
5. Select `Compile {Active}` in the ß menu.
6. Messages will now be written in the `{MPW}:compilerOutput` window. These messages informs about the progress of the compilation, and eventually about syntactic and semantic errors.
7. The program may now be executed by selecting `Execute` in the ß menu.
8. Close MPW using the `Quit` entry in the `File` menu

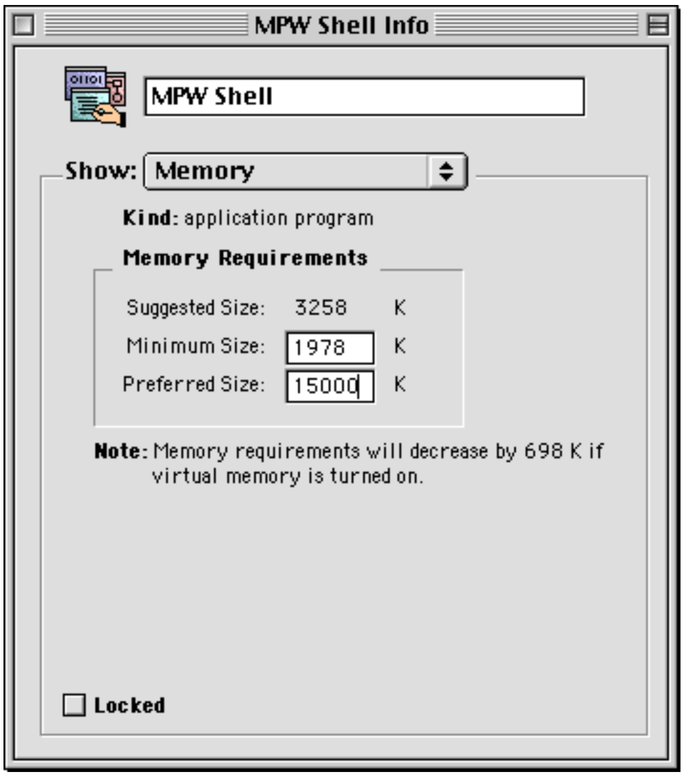
Steps 3, 4, 5 and 7 may be repeated over and over, while making changes to the same program.

3.4.3 Memory Requirements

3.4.3.1 Memory Requirements to MPW Shell

To ensure proper workings of MPW and the BETA compiler, you should ensure, that enough memory is allocated for MPW.

In order to run the BETA compiler at least 15 MB of memory must be allocated to the MPW Shell. The amount of memory can be set by selecting the MPW Shell icon and using the `Show Info` command in the Finder as illustrated in the figure below:


MPW Shell Info

MPW Shell

Show: Memory

Kind: application program

Memory Requirements

Suggested Size:	3258	K
Minimum Size:	1978	K
Preferred Size:	15000	K

Note: Memory requirements will decrease by 698 K if virtual memory is turned on.

Locked

More memory allocated to MPW makes the compiler and especially the linker run faster.

3.4.3.2 BETA Application Memory Requirements

You may also have to increase the memory requirements of a compiled BETA application (1.5 MB for a BETA application). This is done likewise by selecting the BETA application, and then use the `Show Info` command in the Finder.

3.4.4 Errors and Configuration

3.4.4.1 Memory Allocation Errors

When compiling programs or running BETA applications, memory is allocated when needed. During compilation memory is allocated in the MPW heap and during execution of BETA applications memory is allocated in the application heap. In case the execution runs out of memory, one of the following messages can appear:

- `IOA heapspace full`
- `Couldn’t allocate ToSpace`
- `Couldn’t allocate IOA`

You will then have to increase the memory heap as described above.

3.4.4.2 Virus detectors

Some virus detectors (e.g. Vaccine and Gatekeeper) do not allow MPW to operate correctly. Gatekeeper complains during compilation of a BETA program. Vaccine does not complain but causes MPW to run erroneously, typically to hang. Therefore Vaccine and Gatekeeper must be configured to accept MPW.

3.4.4.3 Problems with Memory

In case the system is unable to launch an application from MPW, the problem might be lack of memory. A solution could be to exit MPW and activate the application from the Finder by double-clicking on the application icon.

3.4.4.4 Directories in Fragments

Please notice, that directories in `ORIGIN`, `INCLUDE` etc. in fragments must be specified using the UNIX directory syntax. That is, the Macintosh file:

```
{betalib}guienv:guienv.bet
```

must e.g. in an `ORIGIN` property of a fragment be specified as:

```
ORIGIN '~beta/guienv/guienv'
```

Please also note, that `~beta/` is used instead of `{betalib}`.

3.4.5 Accessing the Macintosh Toolbox from BETA

Programming the Macintosh is done through the Toolbox. The Toolbox includes a large number of routines and they are all documented in Inside Macintosh. There exists a BETA interface to most of these routines such that they can be used in a BETA program.

The fragment `maclib` in file `{betalib}maclib:maclib` contains this toolbox interface.

3.4.5.1 BETA extensions to the Toolbox

Object oriented extensions to the interface to the Macintosh Toolbox have been built in order to make it possible to program the Macintosh in 'the BETA way.' Using purely Toolbox interface routines from BETA forces you to program in a Pascal-like fashion.

The fragments in the folder `{betalib}guienv:` contain an environment (GuiEnv) where windows, text editors, dialogs, menus, etc. have been lifted to 'real' BETA patterns with an object oriented interface.

The GuiEnv environment is documented in [MIA 95-30]

The Toolbox interface is documented in [MIA 90-10]

The folder `{betalib}demo:maclib:` contains demo programs using the

Toolbox interface and the folder `{betalib}demo:guienv:` contains demo programs using the object oriented extensions.

3.4.6 Adding Resources to Applications built with the BETA Compiler

You may use macintosh resource files in your application. You do this by specifying a `RESOURCE` property, e.g.

```
ORIGIN '~beta/basiclib/betaenv';
RESOURCE ppcmac 'foo.r;
-- program: descriptor --
(# do ... #)
```

The BETA compiler will automatically compile all the resources in `foo.r` into the application. The extension `'.r'` tells the system that the resource file is a textual description that must be compiled using the MPW `Rez` tool. If the extension is `'.rsrc'` the system knows that the resource is a compiled resource file, and it will include the compiled resources without calling the `Rez` tool.

You may need to use the lowlevel Toolbox interface to utilize these resource in the current version of the libraries.

3.5 Additional Information for Macintosh

3.5.1 Limitations for Macintosh

See the BETA FAQ:
`http://www.daimi.au.dk/~beta/doc/faq/beta-language-faq.html#SectionV`

---

[3] Please see your Macintosh manuals for how to use your Macintosh.

---

[4] The Enter key is located in the lower right corner of the numeric keypad. Pressing the Return key will just insert a carriage return

---

4 Miscellaneous

4.1 Environment Variables

The following environment variables are used in the Mjłlner System on all platforms.

`BETALIB`

> Specifies how `~beta` is expanded in BETA fragment properties. I.e. it should be set to the path indicating where you installed the Mjłlner System. It is used by many tools in the Mjłlner System.

`BETAOPTS`

> Specifies options that the beta compiler should be invoked with by default.

`BETALINKOPTIONS`

> Specifies the linker options to be used by the BETA compiler when linking. If set, it totally overwrites the default link options the compiler would have used otherwise.

`TMPDIR`

> Occationallyally, the link-directives in the job-file scripts will use a directory for temporary files. If a specific directory is to be used (e.g. because the default temporary directory used is full), setting `TMPDIR` to the name of a directory, prior to compilation, will cause the link-directives to place temporary files in this directory.

`BETART`

> See `BETALIB/doc/betarun/BETART.html`

4.1.1 Windows Specific Environment Variables

If `TMPDIR` is not set, the environment variables `TMP` and `TEMP` are searched. If none of these are set, the directory `C:\TEMP` is used.

4.1.1.1 Automatic Setting of Environment Variables

For most uses, the defaults registered in the registry by the installer are sufficient.

You can make Windows NT set the BETALIB and Path environment variables automatically each time you log in via the System tool in the Control Panel.

You can make Windows 95/98/2000 set the BETALIB and Path environment variables automatically each time you log in by setting it in the `autoexec.bat` file.

4.1.2 UNIX specific Environment Variables

`CC`

Set by the compiler in the jobfiles on all UNIX platforms. Thus `$(CC)` can be used in `BUILD` property commands (see [MIA 90-02]) and Makefiles invoked using the `MAKE` property.

`MACHINETYPE`

Is set automatically by the compiler during the execution of the jobfiles and make files. It may be necessary to set this variable manually, if these command files are executed manually.

`LD_LIBRARY_PATH`

This is a colon separated list of directories to search for external libraries during linking. Notice that not all standard UNIX linkers supports this variable directly, but the jobfiles generated by the beta-compiler will still use this variable.

`BETAREPORT`

If set, cause automatic sending of reports by e-mail to Mjłlner Informatics in case a tool in the Mjłlner System crashes with a fatal error. See the section on Error Reports below.

4.1.2.1 LD_LIBRARY_PATH on Silicon Graphics

On Silicon Graphics machines, `LD_LIBRARY_PATH` is especially important: Most BETA programs are linked using shared object files. This means, that a part of the linking process is postponed until runtime, and in order for this to work, the runtime loader must be able to locate the shared object files generated by the compiler. The compiler will output a suggestion for setting `LD_LIBRARY_PATH` after each compilation, that uses shared object files. In this case, `LD_LIBRARY_PATH` must be set before attempting to run the program. Otherwise you will get a loadtime error like

```
793:./foo: rld: Fatal Error: cannot map soname 'foo1..so'
using any of the filenames
  /usr/lib/foo1..so:/lib/foo1..so:
  /lib/cmplrs/cc/foo1..so:
  /usr/lib/cmplrs/cc/foo1..so:
-- either the file does not exist or the file is not
mappable (with reason indicated in previous msg)
```

4.1.3 Macintosh Specific Environment Variables

The following MPW environment variables are used by the Mjłlner System.

Warning: Except for the `Verbose`, `Time`, and `BETART` variables, you should probably not change these variables yourself.

`{Verbose}`

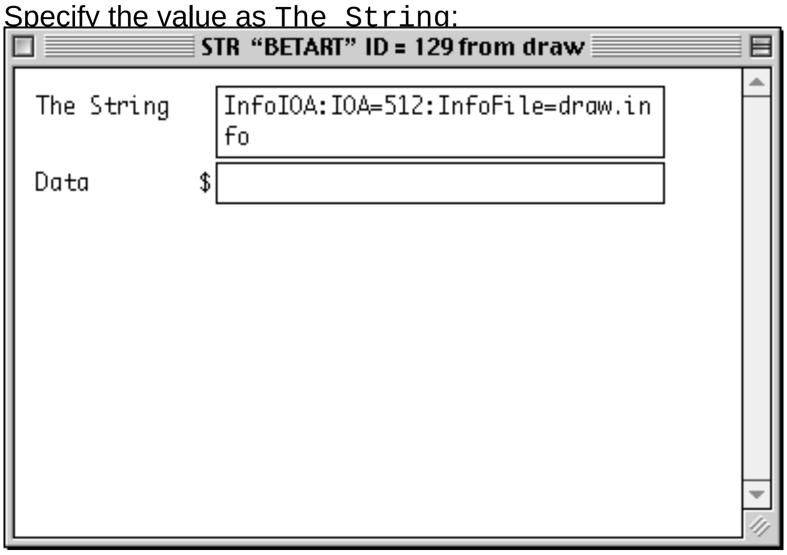
To analyse the BETA compiler memory usage is could be useful to set the Verbose option by issuing the following command in the MPW WorkSheet:
`Set Verbose 1 Export Verbose` Output from the garbage collector of the BETA compiler will then be directed to a special window.
The following command will turn off the garbage collector output:
`Unset Verbose Export Verbose`

`{Time}`

To analyse the time used by the BETA compiler and the linker set the Time option by:
`Set Time 1 Export Time` Time usage of the compiler and the linker will be printed in the compiler output window. The following command will turn off the time usage output:
`Unset Time Export Time`

`{betart}`

Example of setting it for an MPW tool:
`set BETART "InfoIOA:IOA=512:InfoFile=info.dump" export BETART` Example of setting for an application:
For applications the BETART environment variable is read from the resource of type ’`STR `’ with number `129` and name `BETART`.
To change this resource use a resource editor, e.g. `ResEdit`.
Specify the value as `The String`:


STR “BETART” ID = 129 from draw

The String: InfoIOA:IOA=512:InfoFile=draw.info

Data $:

Then specify the number and name using the Resource Info dialog.

`{betaLinkLibs}`

Internal linking variable. Specifies the linker libraries to be used by the BETA compiler when linking (using standard MPW linker). If set, it totally overwrites the default link options, the compiler would have used otherwise.

{BetaLinkCreator}

Specifies the application creator, default 'BETA'

{BetaLinkType}

Specifies the application type, default 'APPL'.

By using the `-Application` and `-Tool` items in the ß menu `BetaLinkCreator` and `BetaLinkType` is changed to 'MPS ' and 'MPST', respectively.

4.2 Using the GNU Emacs Editor

If you want to use the very popular GNU Emacs text editor as an alternative to the structure editor included in the Mjłlner System Integrated Tool ([MIA 99-39] , [MIA 99-40] , [MIA 99-34]), you may benefit from the beta-mode for Emacs located in the file

`BETALIB/emacs/beta-mode.el`

By putting the following lines of Emacs-Lisp code into your `.emacs` file, you can make Emacs automagically recognize your BETA source code file if they have the suffix `.bet` (textual BETA files).

```
(setq betalib (getenv "BETALIB"))
(if (not betalib) (setq betalib "/usr/local/lib/beta"))
;; or if you are on a PC:
;; (if (not betalib) (setq betalib "c:\\beta"))

(setq load-path (append load-path
                        (list (format "%s/emacs" betalib))))

(autoload 'beta-mode "beta-mode")
(setq auto-mode-alist (append (list (cons "\\.bet$" 'beta-mode))
                               auto-mode-alist))
```

When in beta-mode, you can get to know more about beta-mode by using `M-x describe-mode`. Also you may want to byte-compile `beta-mode.el` from within Emacs for improved performance. The directory `BETALIB/emacs` also contains various other contributions for using Emacs to edit BETA programs. For instance, the file `beta-hilit19.el` contains a setup for syntactic colouring of your BETA programs when using Emacs version 19 or later

Emacs is available for UNIX, PC and Macintosh.

Adding the BETA menu

The BETA menu in Emacs 19 or later can be added using the `beta-mode-hook`:

```
(defun mybeta ()
   "Adds BETA menu"
   (interactive)
   (load "beta-menu19" t t)
   )

(setq beta-mode-hook 'mybeta)
```

Other useful thing to hook in

You may do more things using the `beta-mode-hook`. Here is a larger example:

```
(defun mybeta ()
   "Make the following local bindings in beta-mode:
C-xC-rj   calls beta-comment-justify
C-xC-rC-r calls beta-comment-justify-region
C-xC-rC-c calls beta-convert-region-to-comment
C-xC-ru   calls beta-remove-comment
C-xC-ri   calls indent-buffer.

Also adds BETA menu and beta-hilit19.
"
   (interactive)
   (local-set-key "\C-x\C-rj"    'beta-comment-justify)
   (local-set-key "\C-x\C-r\C-r" 'beta-comment-justify-region)
   (local-set-key "\C-x\C-r\C-c" 'beta-convert-region-to-comment)
   (local-set-key "\C-x\C-ru"    'beta-remove-comment)
   (local-set-key "\C-x\C-ri"    'indent-buffer)
   (load "beta-menu19" t t)
   (load "beta-hilit19" t t)
   )

(setq beta-mode-hook 'mybeta)
```

4.3 Documentation on the World Wide Web

The latest version of these manual pages can always be found on the Internet at the URL

http://www.mjolner.com/mjolner-system/documentation/index.html

Various other information about BETA and The Mjłlner System can be found at
Mjłlner Informatics Homepage:

http://www.mjolner.com/

The BETA Language Homepage:

http://www.daimi.au.dk/~beta/

4.3.1 Introductory material

You may find introductory material in the distribution in

`BETALIB/doc/tutorials.html`

4.4 The comp.lang.beta Newsgroup

The USENET newsgroup `comp.lang.beta` is intended for discussions about the BETA language and the programs and systems written in or supporting BETA. Discussions concerning object-oriented programming principles based on the concepts known from BETA will also take place in `comp.lang.beta`, possibly cross-posted to `comp.object`.

The BETA language Frequently Asked Questions will be posted to `comp.lang.beta`, and the most frequently asked questions from `comp.lang.beta` will be included in the subsequent versions of this FAQ .

4.5 Error Reports

The following e-mail address can be used to send error reports and comments:

`support@mjolner.com`

This is *not* a hot-line support; but all e-mails will be answered as fast as possible. Mjłlner Informatics can offer a contract for hot-line support.

For errors specific to either the `mjolner` tool or the `beta` compiler, the following two e-mail addresses may be used instead:

Mjłlner Tool specific errors:

`bug-mjolner@mjolner.com`

BETA compiler specific errors:

`bug-beta@mjolner.com`

The following classification characters can be used to indicate which priority an error should have in the maintenance process.

- **B** - Serious bug
  The tool crashes or produces a wrong result and the error cannot be circumvented.
- **b** - Minor bug
  The tool produces a wrong result but the error can be circumvented
- **I** - Serious inconvenience
  E.g. lack of important functionality.

- **i** - Minor inconvenience
  E.g. lack of functionality that "would be nice to have", but is not crucial.

An error report should include

- a small description the steps that lead to the the error situation (if possible)
- output from the console (if available)
- dumps (if available)
- the smallest possible program that lead to the error situation (if possible). Here the betatar program might be useful.

On UNIX systems, the shell scripts in `$BETALIB/bin`, which invokes the `beta` compiler and the `mjolner` tool can be configured to cause automatic collecting of the necessary information and sending of it to Mjłlner Informatics in case of serious bugs. This is done by setting the `BETAREPORT` environment variable to the value "`yes`":

```
setenv BETAREPORT yes
```

Errors in manuals can be reported using the on-line Manual Error Reporting Facility.

Legal Notice

Apple and Macintosh are registered trademarks of Apple Computer, Inc.
MPW is a trademark of Apple Computer, Inc.
UNIX is a registered trademark of AT&T.
Motorola is a trademark of Motorola, Inc.