Mac06 - The POSIX Environment for MacOS

Installation and Operation Manual

Release Mac06-0.95

 hns@computer.org

(1)

Contents

Introduction

What is Mac06?

Mac06 („Mac oh six“) is a lightweight environment resembling the UNIX (POSIX) programming and user interface. It is built on top of the MacOS application programming interface and is therefore virtually hardware independent within the Macintosh computer series, as already existing device drivers are used.

The main emphasis was until now put on making the system running with full kernel support. Therefore, system call performance is not yet optimized. The copy command `cp` for example copies data with about 30kbyte/second.

The file system completely relies on the MacOS HFS and, therefore, transferring data between Mac06 and MacOS is very simple.

Programs can be compiled using Symantec C++/THINK C or Code Warrior. In this environment, even the MacToolbox (dialogs, windows etc) could be used within Mac06 executables. It is possible to start any Macintosh application from within the Mac06 system if you have acces to the file.

Not much effort has been spent to port all those useful commands found on large scale UNIX systems like `sed`, `awk`, ... but there are already many ideas to extend the system.

The overlay structure to MacOS has its drawbacks, of course. There is no memory protection and task switching is not preemptive.

What can it be used for?

Mac06 is a solution if you want to

- run legacy UNIX software (written in C/C++ requiring POSIX libraries) on a Mac,
- have a low-end Mac (68k or PPC Performa) and want a simple solution to run UNIX,
- have a Performa 5200 on which neither MkLinux nor Linux68k will run,
- learn UNIX,
- switch between MacOS (Finder) and UNIX without rebooting,
- do software development for embedded controllers (Z180, 68k, PPC, ARM) on a Mac with familiar UNIX tools.

Features

- easy to install
- UN*X like file system using HFS/HFS+
- Finder integration
- mostly POSIX compatible `libc.a` and `#include` headers
- commands like `sh`, `ll`, `cd`, `cat`, `xd`, `echo`, `find`, `fgrep`, ...
- can run applications written with Think C/Symantec C++
- `/dev/console`, `/dev/tty` are mapped to terminal windows
- `/proc` and `/vol` file system
- sockets for TCP/IP
- `c89` ISO-C compiler

Installation

Download the packed and binhexed file, for example from

`http://ourworld.compuserve.com/homepages/Nikolaus_Schaller/mac06.html`

You may have to drop the file onto `Binary Pump` to set the appropriate creator and type. Expand the package using `Dehqx` or `StuffitExpander`.

To run, launch the `mac06` application found in the root directory and the Mac06 kernel will start. A console window will open showing boot and greeting messages from the kernel (as shown below). In this window, you can enter shell commands.

Deinstallation

Simply move the `mac06` folder to the trash. Mac06 does not modify any system file and does not install INITs.

System Requirements

- System 7.x or later
- a Mac or Performa with at least 68020 and 8 MB of RAM
- about 5 MB hard disk space

Mac06 uses fat binaries or 680x0 code and therefore should run on all Macs (at least a 68020 is required and MacOS 7.0)

The software is still under development. Therefore, there are certainly bugs and missing functions in all areas of the system. Some are known and some are not. If you run across a bug, decide yourself to send a mail (`mailto:hns@computer.org`) to the author. Most bugs will be fixed in the next release and may result in some new ones...

Some Legal Stuff

Licence Conditions

The copyright of this Software is owned by the author H. Nikolaus Schaller, Munich - hns@computer.org. The software is not public domain.

The software is licenced under the following conditions:

- the COPYRIGHT, LICENCE and README files are not removed
- the software is used only for legal purposes
- the software is redistributed as the original package (.hqx file) and not as parts of it
- Redistribution (copying to other media) of the packed .hqx file is free.

There is currently no shareware fee for use (copying to the memory and processor) of the software. The right to change this in future releases is reserved as well as changing the pricing indicated below.

Donations to support further development are greatly appreciated!

This software is distributed as is and there is no hotline and no service.

Guarantees or warranties of any kind are excluded as far as legally possible.

Trademarks (UNIX, MacOS and so on) are used without further indication. They are used only as a reference to describe the relationship with these products. They are property of the respective owner.

Registration & Shareware Fee

Since Mac06 is still under development, we are strongly interested in users doing beta testing the releases coming before Mac06-1.0. Therefore, the price policy is currently as follows:

- payment of a shareware fee is completely voluntary.

Introduction(1)

- But, if you like to further support the development efforts, voluntary payments („donations“) are taken as a prepayment of the shareware fee for the first final release. This means, you will not have to pay again for Mac06-1.0 if you do a voluntary payment before its final release.

The pricing suggested for donations is as follows:

	single user	site	parts covered
Mac06	US$20	US$200	kernel, shell, files, device drivers, compilers, communication etc.

A site licence is equivalent to 10 users and covers all locations of your institution/organization within a 160km (100 miles) radius of your site. One big advantage of a site licence is that you do not need to keep track of how many people at your site are using the software.

Payment is fairly simple. Open the register program (found in `/bin/register`) either by using the Finder or by the `Donations...` menu item in the `File` menu of Mac06. Enter your name, email-address, and the number of single user or site licences you want to donate.

If you are paying by US$ check or cash[1], Print the data from the register program and send the data together with your check or cash to Kagi. Their postal address is

```
Kagi
1442-A Walnut Street #392-QM4
Berkeley, California 94709-1405
USA
```

If you are paying with credit card, fill in the required data and then either Print and Fax the data to Kagi using the Fax number +1 (510) 652-6589. Or Copy the data and paste it to an email to Kagi (sales@kagi.com)

That’s all. After a while (3 to about 10 days for processing) you will receive an email receipt from Kagi, provided that you have specified a valid internet email address.

1. Falls Sie in Deutschland leben und bar oder per Scheck bezahlen möchten, senden Sie bitte vorab eine E-Mail an uns (hns@computer.org), da in diesem Fall eine Bearbeitung über Kagi, USA unnötig aufwendig wird. Sie erhalten dann Hinweise über die Abwicklung. Zahlungen per Kreditkarte senden Sie bitte trotzdem über Kagi, da wir selbst keine Kreditkartenabrechnungen vornehmen.

Introduction(1)

Please note the COPYRIGHT and LICENCE files. The software is not public domain. The current release is distributed as freeware but the right is reserved to change this in future releases as well as changing the pricing indicated above.

Operation

Starting Mac06

Simply double-click on the ma06 application found in the root directory of the distribution package.

After a while, a terminal window will appear waiting for command input.

The Terminal Window

The terminal window looks like this:

```
console
Welcome to the mac06 shell.
Mon Mar 09 22:55:42 1998
To exit mac06, enter 'shutdown' in this shell.
You have mail.
# mail
From hns@computer.org
From: hns@computer.org
To: user1
Subject: This is a test message
Welcome to the mac06 mail system.
Sincerely yours,
H. N. Schaller - hns@computer.org
# echo xxx
xxx
# stty
-nobrkint -ignbrk -noparmrk -noinpck -noistrip -inlcr -noigncr -icrnl -noixon -n
oixoff -noopost -clocal -cread -csize???? -nocstopb -hupcl -noparenb -pareven -e
cho -echoe -echok -echonl -icanon -isig -flsh -notostop -noiexten intr=^C quit=^
\ start=^Q stop=^S susp=^Z eof=^D eol=
 erase=^H kill=^U ispeed=0 ospeed=0
#
```

Like any MacOS window, you have a „go away control“ in the top left corner. By clicking on this button, the terminal window (and the shell) can be closed.

By clicking in the title area (where the name console stands), you can move the window.

Resizing can be done either by clicking the „grow control“ in the top right corner or by clicking in the bottom right corner and dragging the window. In the current release, only the surrounding window is resized but the number of lines and columns is not changed.

Text entry into the window is sent to the application(s) controlling the window. Usually, this is the shell (command interpreter). Text output from the processes started by the shell goes to this window. In the example, the commands `mail`, `echo xxx` and `stty` have been started and the output of this command is printed below the command enty line.

So, what are commands and what is output? Commands can be entered after the shell prompt (usually a `#` or a `$` character). At this position, a cursor is blinking.

Note, that standard UNIX behavoiur is, that echoing of the characters typed in is immediately, even if the current command is still running. Therefore, typing while command output may disrupt the screen layout. This behavoiur can be controlled by setting special terminal options and some applications do this internally. Note also, that pressing the return key sends the line to the shell.

By the way, the terminal is a VT52 emulation.

The Shell and Shell Commands

The shell is the command interpreter. You can control processes, trigger operations, start applications, get information, and write programs by entering appropriate shell commands.

The shell follows a simple command line oriented principle: print a prompt, wait for the entry of a line, split up the line into command and arguments, analyze the command, find and run the appropriate program to execute the command. Then, wait for the completion of the command and print the next prompt.

The prompt is usually a `$` or `#` character followed by a blank character to indicate, that you can now enter a command. Submit the command by pressing the return key. An example:

```
$ echo my first command
```

In this example, the `$` is printed by the shell and all the characters from `e` to `d` are typed in. Now, the shell splits up the command separating at blank characters. Then it looks up the command in its internal table, and, in this case will find it there. But if not, it will examine the `$PATH` variable (described below) for a list of directories and search there for a program file with the name `echo`. In either case, the echo command will be started and the arguments „`my`“, „`first`“, and „`command`“ are passed. Then, the command starts execution and simply prints out its arguments. Therefore, you will see

```
$ echo my first command
```

```
my first command
$
```

Some commands have not only arguments but also options. Options begin by convention with a − and in some cases with a +. The shell passes these options simply as arguments as the options are decoded by the command itself. For many commands, you can figure out the list of available options by specifying `-?` as the first option.

The echo command has, for example, the option `-c` which suppresses the final newline character after printing the arguments.

```
$ echo -c +++
+++$
```

Now, what about variables? Variables are stored in the shell and have a name and a value. You can define new variables by typing in

```
$ newvar=123455
```

Variables can be used (referenced) in any command by entering a $ followed by the variable name (which must exist before).

```
$ echo $newvar
123455
```

All variables can be listed by

```
$ set
```

In this list, you will find some predefined variables like `$HOME`, `$PATH`, `$TERM`.

Now, how to quit the shell?

The shell has a builtin command

```
$ exit
```

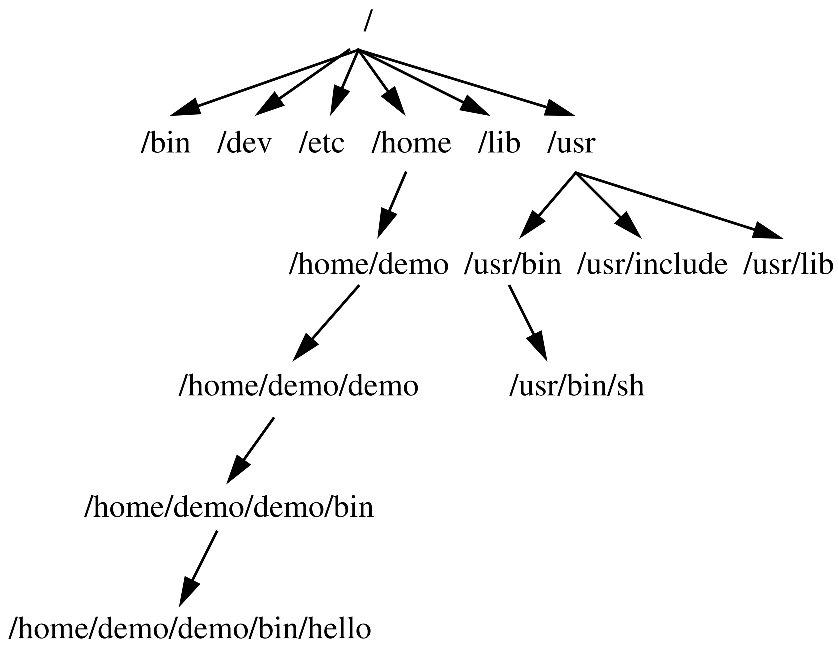
which makes the shell quit. The Mac06 kernel checks if this was the last process and will also quit in this case.

For more information, please have a look at the Internet Newsgroup `news:comp.unix.shell` where you will also find many hits and tricks for shell programming.

Managing Files

The File System

The file system is a hierarchical structure of directories (folders) and files. The root directory is denoted by the name `/` (slash). Files and directory objects are specified by typing in their path name which is a path starting at the root and traversing down through the directory tree. The next level is specified by a separating slash as shown in the figure.



The Current Directory

Additionally, each process remembers a „current directory“. File names not beginning with a slash (`/`) are relative to this current directory. Therefore, short names can be used. The shell provides the `pwd` command to print the current working directory:

```
$ pwd
/home/demo
$
```

The current directory can be referenced to certain commands by the special name **.** (single period).

Changing the Current Directory

The current directory can be changed by the `cd` command:

```
$ pwd
/home/demo
$ cd demo/bin
$ pwd
/home/demo/demo/bin
$
```

The special name **..** is recognized as the directory above the current. Therefore

```
$ cd ..
$ pwd
/home/demo/demo
$
```

File Name Expansion

The shell expands all command arguments (file names) containing a `*` or `?` character by trying to match all files in the current directory to the pattern specified. A `*` matches any substring while `?` stands for a single character. Note, that the reslt is alphabetically sorted. If the pattern does not match, it is passed as an argument. Therefore

```
$ echo *
abc b bcd efa
```

lists all file names, while

```
$ echo *b*
abc b bcd
```

lists all names containing a `b`,

```
$ echo *z*
*z*
```

does not match anything, and

```
$ echo b?*
bcd
```

does not find the file `b`.

The expansion also works for directories and absolute file names

```
$ echo /home/demo/demo/*
/home/demo/demo/bin /home/demo/demo/src
```

To pass a argument containing a `*` or `?` unmodified to a command (like the `find` command described below), enclose the argument in apostrophes or double quotes:

```
$ echo "*b*"
*b*
```

Listing Directories

The contents of a directory are listed by the `ls` or `ll` command. The command `ll` is an abbreviation of `ls -l`, which gives a *long* listing, while `ls` gives the file names only. Examples:

```
$ ls
bin  src
$ ll
drwxrwxrwx 1 root     root        64  Thu  10.Sep  98  12:05  bin
drwxrwxrwx 1 root     root       160  Thu  10.Sep  98  12:04  src
```

In this listing, the first column describes the file type (`d` = directory) and the access rights for the three personalities file owner, the group and others (`rwx` = read/write/execute). The third and fourth colums print out the owner and group. The number displayed in the fifth column is the total file size in bytes. Then comes the creation/modification date and finally the file name.

Note, that files beginning with a period are hidden unloess the option `-a` is used:

```
$ ll -a
drwxrwxrwx 1 root     root       336  Thu  06.Aug  98  09:36  ..
drwxrwxrwx 1 root     root        64  Thu  25.Dec  97  22:56  .
drwxrwxrwx 1 root     root        64  Thu  10.Sep  98  12:05  bin
drwxrwxrwx 1 root     root       160  Thu  10.Sep  98  12:04  src
```

You can also specify a certain directory to be listed:

```
$ ll bin
-rwxrwxrwx 1 root     root      6128  Thu  10.Sep  98  12:05  bin/hello
```

Displaying Contents of a File

The contents of a file can be displayed by the `cat` (catenate) command which simply copies the contents of the specified file to files to the terminal window.

```
$ cat file1 file2
```

But if the file is longer than about 20 lines, it will not fit onto the screen and the beginning of the file will scroll away. Therefore, the `more` command is available.

```
$ more file1 file2
```

The `more` command will display one file after each other but will stop after filling one page of the screen and await for user entry. There you can press the return key to start the next page or enter `q` and return to quit the `more` command.

Copying, Moving and Removing Files

Files can be copied with the `cp` command. Simply making a copy of a single file is done by

```
$ cp file copyoffile
```

This command will create the file `copyoffile` if it does not yet exist and copy all contents of the specified file.

Copying several files is only possible by specifying a directory as the destination (last argument). This directory must already exist.

```
$ cp file1 file2 file3 destdir
```

If you want to have the original removed, i.e. a file renamed, use

```
$ mv oldname newname
```

If you want to move the file (without changing the name), specify a different directory as the destination:

```
$ my file newdir
```

Files can be removed with the `rm` command:

```
$ rm file
```

Note, that Mac06 does not remove the file finally but copies the file to the MacOS Trash. Therefore, inadvertently removing a file is not as dangerous as in standard UNIX.

Finding Files

The command `fgrep` (fixed grep) scans a specified file or files for a certain substring. If the string is found, the line is printed. Options exist to print only the number of matching lines.

```
$ fgrep main *.c
```

will scan all C-sources for the string `main`.

The command `find` is very similar to the Find Files command of the MacOS Finder. Its purpose is to identify files which match certain conditions. You can look for the name, the size, the file type, the owner, the modification date etc.

An example:

```
$ find /home/demo -name '*.c' -print
```

will scan the demo directory and all its subdirectories for file names ending in .c and print their name. Note, that the pattern has ben put into apostrophes to prevent, that the shell expands the pattern.

If `-print` is missing, the find command will behave strange. Although it will find the files it will not print their full name.

Changing File Access Rights

The owner of a file can be modified by the `chown` command (which is curently not supported by the kernel).

Access rights can be modified by `chmod`. Currently, only the user's write permission can be modified. Therefore

```
$ chmod u+w file
```

permits writing to the file and

```
$ chmod u-w file
```

write protects the file.

Redirection and Pipelines

The output of a command (called standard output) goes by default to the terminal from which the command was started, i.e. the shell and the command share the same output window. This can be modified by file redirection:

```
$ ls
a b c
$ ls >file
$ ls
a b c file
$ cat file
a b c file
```

Note, that the output redirection has created a new file before the `ls` command was run. Therefore, the new file is included in the listing.

The command input (called standard input) can also be redirected by using a < charatcer instead of >. This is more rarely used, as most commands treat all arguments as file names and process the standard input only of no argument is specified. So

```
$ cat <file
a b c file
```

gives the same output as `cat file`.

Now, it could be useful, to save the output of one command into a file and then process this intermediate file by a second command. Doing this, wastes disk space and is slow. Therfore, there is a capability of connecting the output of a command directly to the input of another command (calles a pipe):

```
$ ll -a | fgrep 64
drwxrwxrwx 1 root     root       64 Thu 25.Dec 97 22:56 .
drwxrwxrwx 1 root     root       64 Thu 10.Sep 98 12:05 bin
```

Here, the output of the `ll` command is filtered for entries containing the number `64`.

The Finder Integration

The MacOS file system is just a different view to the files available through the Finder. Therefore, files can be either copied by Mac06 commands or by dragging them from a Finder window to another. This makes it possible to intermix working with MacOS applications and Mac06 by fo example using your favourite text editor (a MacOS application) to edit files for Mac06.

Both systems, MacOS and POSIX have their own rules for file names and, therefor, filenames are mapped according to certain rules:

Finder name	Mac06 name
.	...
..	....
.x	.x
..z	....z
/	:
blanks	?
?	blank

Examples

Finder name	Mac06 name
.	...
Test	Test
..Test x/y	....Test?x:y
File 1	File?1
C/C++	C:C++
a:b (a is directory)	a/b
Test?	“‘Test “
no representation	.
no representation	..

These rules are applied vice versa for creating new files and directories. All this is done to keep the directory structure transparent. Note that the virtual files (to be correct: directories) „.“ and „..“ are available under Mac06 only and have no direct MacOS equivalent.

And, finally, please note that upper and lower case characters are distinguished!

Managing Processes

Listing the Processes

To list all processes, enter `ps`. This command will display the Mac06 process number and the command. The command `ps -l` gives a *long* listing telling much about internals like the process status, open files, signals, etc. Note, that MacOS assigns different process numbers.

Signals and Killing Processes

Signals are a method of UNIX systems to asynchronously notify running or stopped processes about certain conditions.

Signals can be either generated by the running process itself (timers, system errors), by pressing keys or buttons, or by the `kill` command.

```
$ kill -15 7
```

would send a `SIGTERM` (15) signal to the process 7. If the process reacts on this signal, it would normally close all files, delete temporary files and exit.

By pressing ctrl-`C` oder Apple-`.` you can send a `SIGINT` or SIGQUIT signal to all processes controlled by the active terminal window. This will normally terminate the command and abort the output. Clicking into the Close box of a window sends a SIGHUP signal.

Running MacOS Applications

You can copy MacOS applications (68k or PPC or fat binaries) to the Mac06 file system and they will be recognized as executables by the shell. To run them in the background, add an ampersand (`&`) to the command. You can also kill these applications with `kill -9` - they will receive a ‘`quit`’ apple event (‘`aevt`’).

Communication

The communication part of Mac06 is not yet completed. Therefore, most of the following commands are not yet implemented or working properly. The terminal devices and `nslookup` are working. `Telnet` and `dial` are able to open connections.

Devices

Device files can be found in `/dev`. So, try

```
ll /dev
```

In most cases, device files behave like ordinary files. Therefore, they can be used for file redirection in the shell. Try

```
$ cat </dev/tty1 >/dev/tty2
```

This will open two new windows and will echo all lines typed into the `tty1` window to the `tty2` window (but not vice versa). The windows can be closed by entering ctrl-`D` which will notify the `cat` command to quit.

nslookup

This command is used to look up internet node names in the name-server (ns).

```
$ nslookup ftp.apple.com
```

```
ftp.apple.com = xxx.xxx.xxx.xxx
```

This command will open your internet connection (PPP, dialup modem or alike) and query the network for the network address.

telnet

This command opens a telnet remote terminal session to the specified host. The terminal emulation is a VT52.

dial

This command allows you to control the modem directly and dialing up a mailbox or dialup-host. The terminal emulation is a VT52.

ftp

This command allows to fetch or send files through the internet from or to a FTP server.

mail

This command allows to get or send e-mails from or to the internet by contacting a POP3 server.

The Development Package

The development package is still under construction. All commands documented here are working to some extent but the final compiled programs are not.

How to Compile Programs

Mac06 provides an ISO-C compiler `c89` with integrated preprocessor and assembler, a linker `ld` and an librarian `ar`. All three use the COFF file format.

To compile a source file, enter

```
c89 -c source.c
```

This will result in an `source.o` file in the current directory.

To producepreprocessor output only, use the `-P` option and for assembler output only, use `-S`.

If `-c` is omitted from the `c89` call, `-lc` and `crt.o` are added to the list of files and the linker `ld` is called automatically. Therefore,

```
c89 -o myprog soutce1.c source2.c
```

will compile both source files and link them to get the executable program file `myprog`.

The Make Tool mk

The tool `mk` is a non-standard make (program builder) utility.

It reads a file called `mkfile`. This file contains the following control commands:

`IMPORT project`	to specify a project for inclusion (i.e. headers and libraries)
`LIBRARY l.a s.c ...`	to specify the creation of a library
`PROGRAM p s.c ...`	to specify the creation of a program

Libraries can also be spcified as part of the `PROGRAM` command.

Running `mk` starts all appropriate actions like compiling the sources, adding objects to the libraries or calling the linker. Dependencies are also analysed, so that only those files are compiled, that have been modified.

Porting Applications

Porting of applications should go in the following way, although there is no general recipe and some UNIX experience is required.

First of all, create a new subdirectory. Then FTP the source files from the server to this directory. You can do this with the Mac06 `ftp` command (when available), with `fetch` or your browser.

Then `unpack/untar` the usually archived files.

The UNIX standard program for managing program packages is `make` and sometimes, also `imake` is used. Neither is supported by Mac06 (unless some freely available distribution is ported). But the much simpler tool `mk` is provided which has comparable features. Therefore, the `makefile` provided by the software pakkage has to be converted to a `mkfile` by hand. This usually requires to set up a `project/src` and a `project/include` directory and copying the source and include files to these directories. Then, create a `mkfile` in the `src` directory. Add a line like

```
LIBRARY libname.a libsrc1.c libsrc2.c ...
```

for each library that is created by the package

and

```
PROGRAM progname progsrc1.c progsrc2.c ...
```

for each application program generated by the package. Identifying which sources belong to which library and program needs some experience in reading `makefiles` but is usually straightforward.

Sometimes, makefiles are also used for special tricks, like extracting the embedded manual from the source files by using `sed/awk` or having special installation rules. This can not be ported to the `mkfile` and must be done by hand.

Finally, change to the `src` directory and enter `mk`. This will start the compilation process.

Typical adaptions to be made in addition are to modify the source to include the proper header files.

If the source code doesn not use some special code and is POSIX conformant, you should not experience any trouble.

Compute intensive applications need a special treatment. Since the Mac06 kernel does not provide preemptive multitasking, the applications have to cooperate. Fortunately, this is fairly simple to achieve: simply add a getpid() call within the calculation loop so that it is called several times per second. As each sytem call gives the other processes (including the Finder) a chance to interrupt, your application will no longer block the Mac user interface.

Finally, if you have succeeded in porting an application, you are invited to have a link to your WWW pages added on the Mac06 home page.

Writing native MacOS applications for Mac06

Using Think C or Symantec C++ for PowerPC, you can write MacOS applications running under Mac06. To do this, create an empty project, add the appropriate libraries from `/usr/lib` and the source files of your application. Make sure to add appropriate aliases to Symantec C so that `/usr/include` files are recognized.

Compile and save the application file within a directory of the search path (`$PATH`). An example is given in `/home/demo/demo/src`. By the way, you can copy `hello.π` as `@π` to the `(Project Models)` folder of the compiler.

For Metrowerks Code Warrior, there is a project file written by Erik Winkler. You can find a link on the Mac06 home page at Third Party Applications:

`http://ourworld.compuserve.com/homepages/Nikolaus_Schaller/mac06.html`

For MPW, there is no recipe available yet.

Exiting Mac06

To exit from Mac06, i.e. stop all processes and release all the occupied memory to MacOS, enter the `shutdown` command. This should normally terminate all processes and exit the `mac06` kernel application.

Or use the `Quit` menu entry in the `File` menu. Shutting down MacOS will also stop Mac06.

Occasionally, applications can get stuck. In this case, use a MacOS tool like `AppWatcher` to kill all processes by hand.

Administration

Adding new Users

To add new users, create a new home directory in the `/home` directory - preferably through the Finder. This can calso be an alias to a directory - even outside the Mac06 directory tree.

Then, add the user to the `/etc/passwd` file.

Some Literature

Donald Lewine, POSIX Programmer's Guide, O'Reilly&Assoc., Inc., Sebastopol, ISBN 0-937175-73-0

Thomas Horn, Systemprogrammierung unter UNIX, VTB, Berlin, ISBN 3-341-01090-4/0863-0860

Michael Beck et. al., Linux-Kernel-Programmierung, Addison Wesley, Bonn, ISBN 3-89313-939-X

M. Banahan, A. Rutter, UNIX: lernen, verstehen, anwenden, Carl Hanser, München, ISBN 3-446-13975-3 (translation of: UNIX - the book, John Wiley & Sons)

The Open Group: `http://www.opengroup.org/onlinepubs/7908799`

Index

Symbols

A

B

C

Index(1)

Index(1)

I

K

L

M

N

Index(1)

O

P

Q

R

S

Index(1)