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.

Until now, the main emphasis was 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 easy.

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 task memory protection and task switching is not preemptive. Therefore, not completely debugged processes can block each other or even crash the system.

What can it be used for?

Mac06 is a solution if you want to

- learn UNIX e.g. to prepare yourself for MacOS X,
- use your low-end Mac (68k or PPC Performa) and want a simple solution to run UNIX,
- use your Performa 5200 on which neither MkLinux nor Linux68k will run,
- switch between MacOS (Finder) and UNIX without any rebooting,

- want to use standard MacOS tools like StuffIt, Norton Utilities, Netscape etc.,
- do software development for embedded controllers (Z180, M68K, PPC, ARM, ...) on a Mac with familiar UNIX tools,
- run legacy UNIX software (written in C/C++ and requiring POSIX libraries) on a Mac,
- need a file and console terminal interface for your application.

Features

- easy to install
- UNIX like file system using HFS/HFS+
- Finder integration including Apple Events
- 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++/Code Warrior
- `/dev/console`, `/dev/tty` are mapped to terminal windows
- `/proc` and `/vol` file system
- `-lsockets` for TCP/IP
- `c89` ISO-C compiler, COFF assembler, linker and archiver
- `-lcurses` library
- HTML based online manual
- X library for external X server (e.g. MI/X) in preparation

Installation

First of all, download the packed and binhexed files (`.hqx`) that you want to install-from

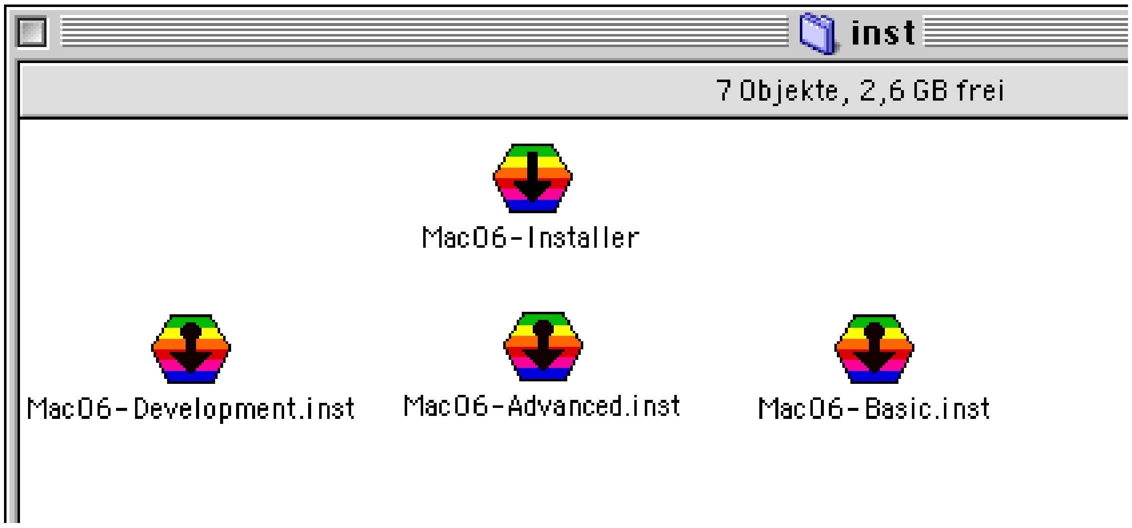
```
http://www.dsitri.de/projects/mac06/index.html
```

The following packages are available

Package	single user licence	site licence	contents
Mac06-Basic	free	free	Kernel, shell, demo software
Mac06-Advanced	US $10	US $100	Manuals, advanced utility programs, include headers and libraries
Mac06-Development	US $20	US $200	ISO-C compiler, assembler, archiver, linker, make tool, libraries
Mac06-Communication	US $10	US $100	telnet, ftp, mail and daemons
Mac06-GUI	tbd.	tbd.	Xlib, Xt, Smalltif

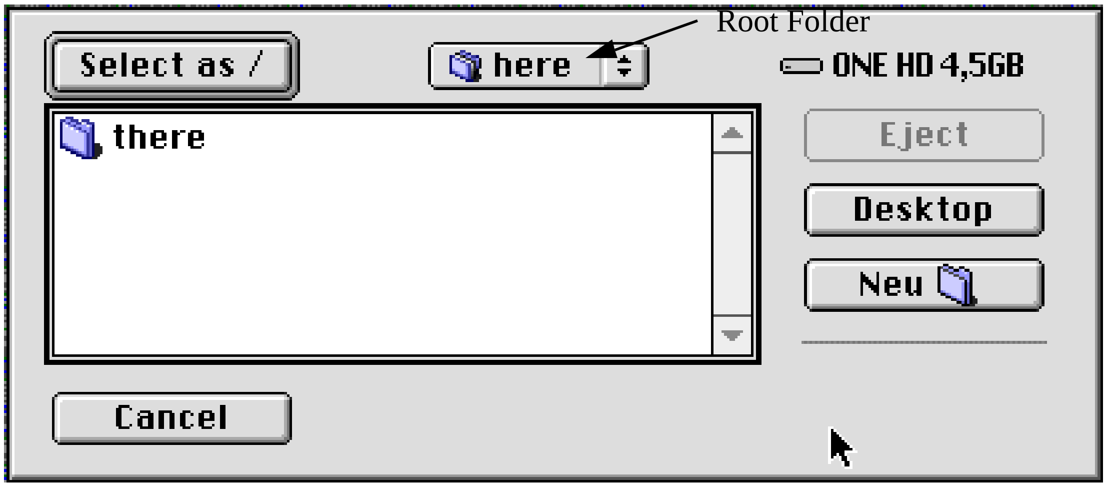
Depending on the rules built into your FTP client you may have to drop these files onto `Binary Pump` first to set the appropriate creator and type.

Expand the downloaded package(s) by usinga decompression tool like `StuffitEx-pander`. This results in file(s) like the following



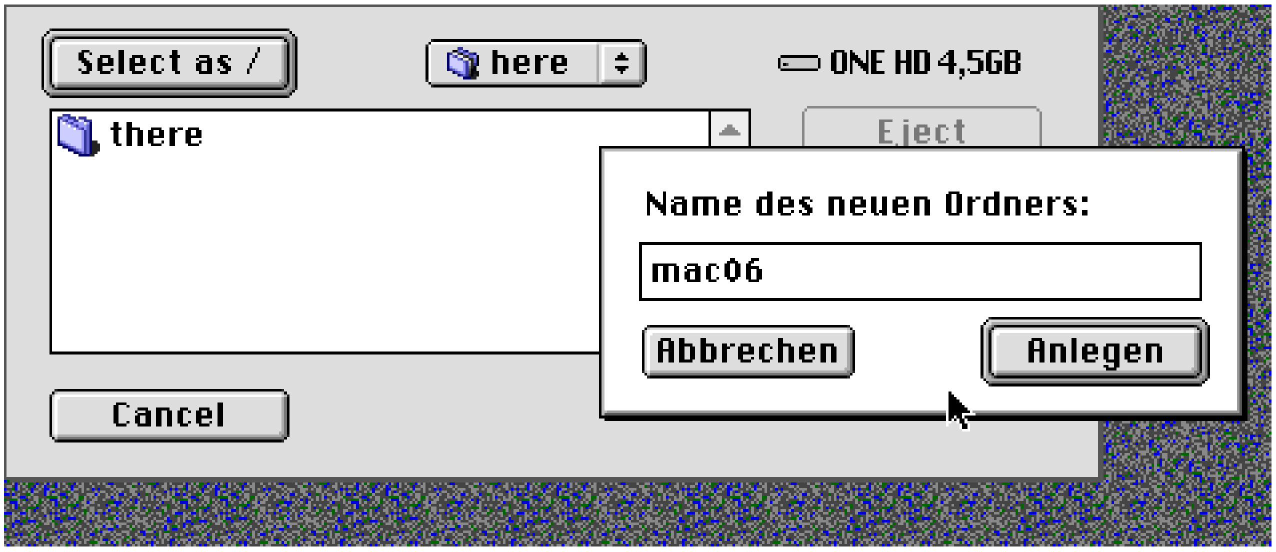
Introduction(1)

Then, double click each of the packages you want to install or select and drag them onto the `Mac06 Installer` icon. This will first open a destination folder selection box:

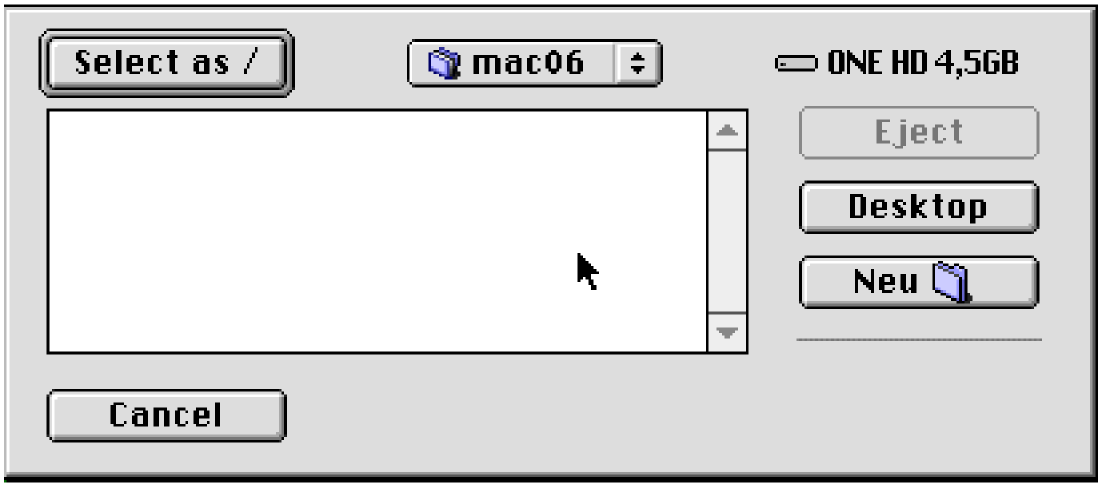


To select the folder you want to install Mac06 into, click through the folder hierarchy until the desired destination folder is shown at the position where ”here“ is shown above. This folder will become the root (/) of the Mac06 file system. Note, that you may see a localized version of this selection box, depending on your version of MacOS.

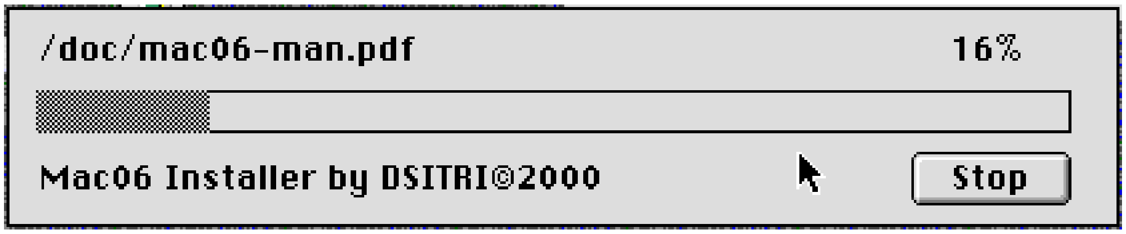
If the Mac06 root folder is not yet prepared or existing, you can also create a new folder as shown below:



This folder will become the ”here“ folder.

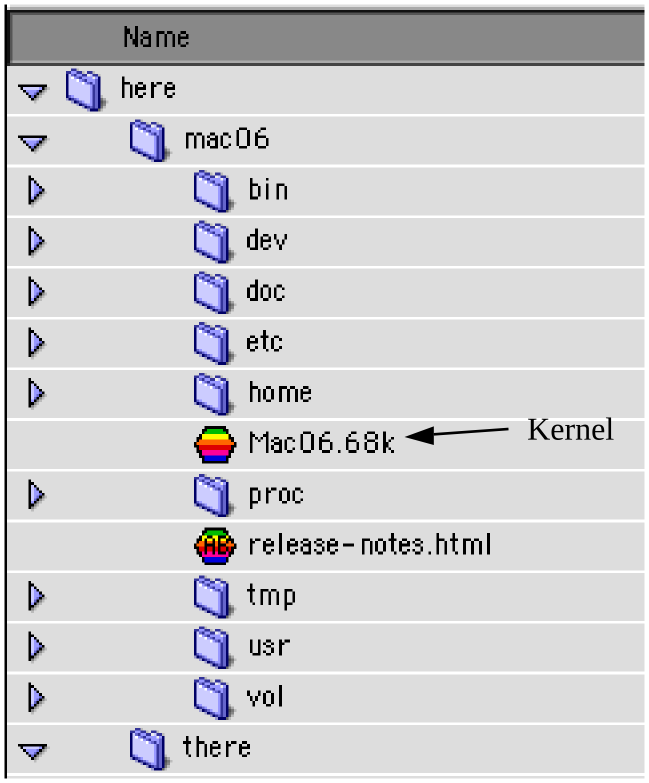


When the desired root folder is selected as ”here“, press the ”`Select as /`“ button to start installation of the files. The progress of the installation process is shown like



Introduction(1)

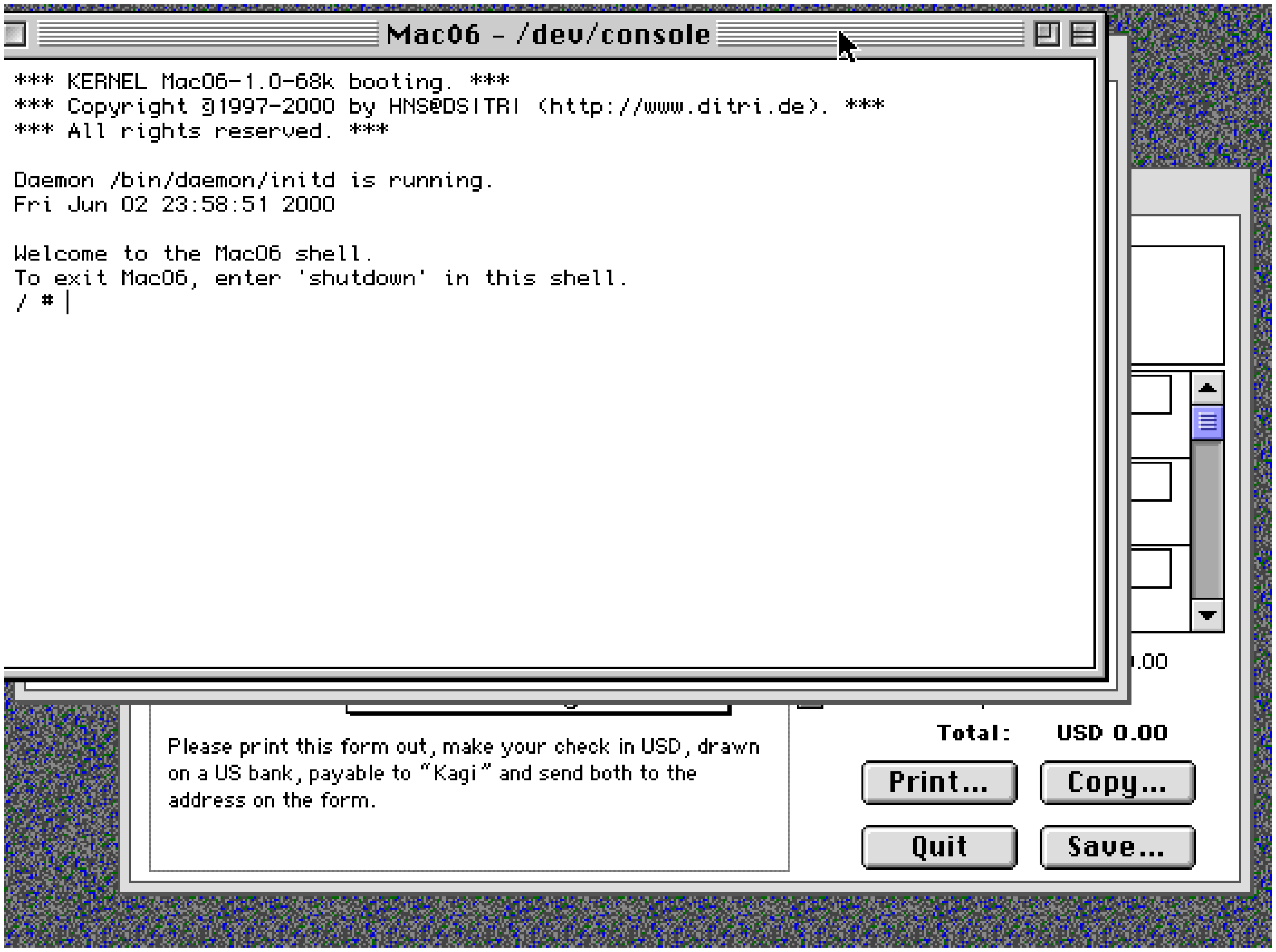
After completion, you will find the installed files in the root directory looking like in the example shown below:



To continue installation, double click the `Mac06` or `Mac06.68k` application. This will start the Mac06 kernel. Then, read the `release-notes.html` file in your HTML browser (e.g. Netscape).

Introduction(1)

Starting the kernel will bring up a console, a terminal window and a register application looking like this



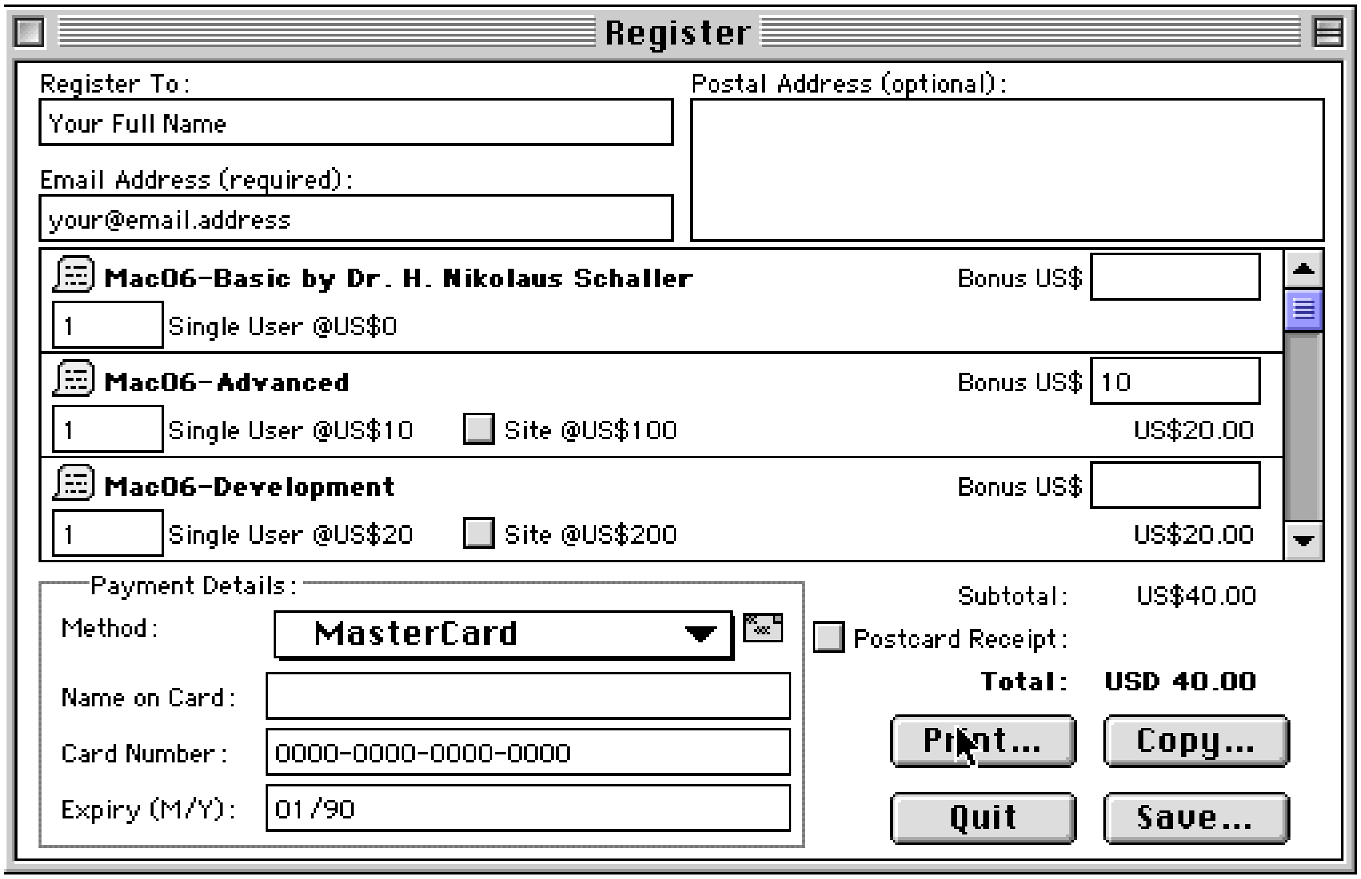
```
*** KERNEL Mac06-1.0-68k booting. ***
*** Copyright @1997-2000 by HNS@DSITRI (http://www.ditri.de). ***
*** All rights reserved. ***

Daemon /bin/daemon/initd is running.
Fri Jun 02 23:58:51 2000

Welcome to the Mac06 shell.
To exit Mac06, enter 'shutdown' in this shell.
/ # |
```

Please print this form out, make your check in USD, drawn on a US bank, payable to “Kagi” and send both to the address on the form.

Click into the register application to bring it to the front to process your Shareware Payment.

Shareware Fee Payment

Payment is fairly simple. Enter your name, email-address, and the number of Single User or Site Licences you want to purchase for each package in the register application as shown below



Register

Register To: Your Full Name

Email Address (required): your@email.address

Postal Address (optional):

Mac06-Basic by Dr. H. Nikolaus Schaller		Bonus US$	
1 Single User @US$0			
Mac06-Advanced		Bonus US$	10
1 Single User @US$10	☐ Site @US$100		US$20.00
Mac06-Development		Bonus US$	
1 Single User @US$20	☐ Site @US$200		US$20.00

Payment Details:

Method: MasterCard

Name on Card:

Card Number: 0000-0000-0000-0000

Expiry (M/Y): 01/90

Subtotal: US$40.00

☐ Postcard Receipt:

Total: USD 40.00

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.

A Bonus is greatly appreciated.

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
```

Introduction[1]

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.

Please note the legal stuff below. The software is not public domain.

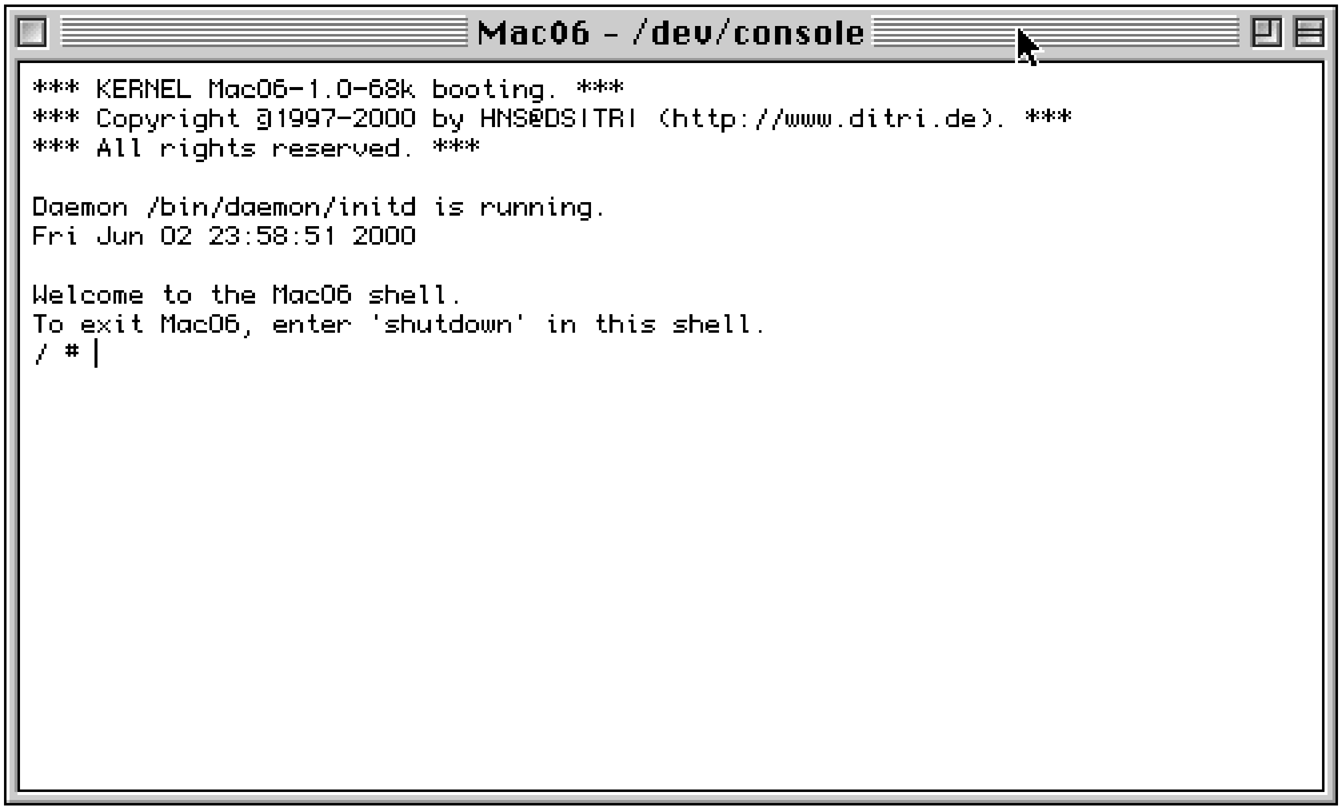
After payment has been sent to Kagi you are allowed to remove the entry from `/etc/inittab` that automatically starts `/bin/register` by editing this file in any text editor.

1. Falls Sie in Deutschland leben und bar oder per Scheck bezahlen möchten, senden Sie bitte vorab eine E-Mail an uns (`service@dsitri.de`), 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.

Short Description

Console Window

A console window will open showing boot and greeting messages from the kernel (as shown below). In this window, you can enter shell commands. It looks like this



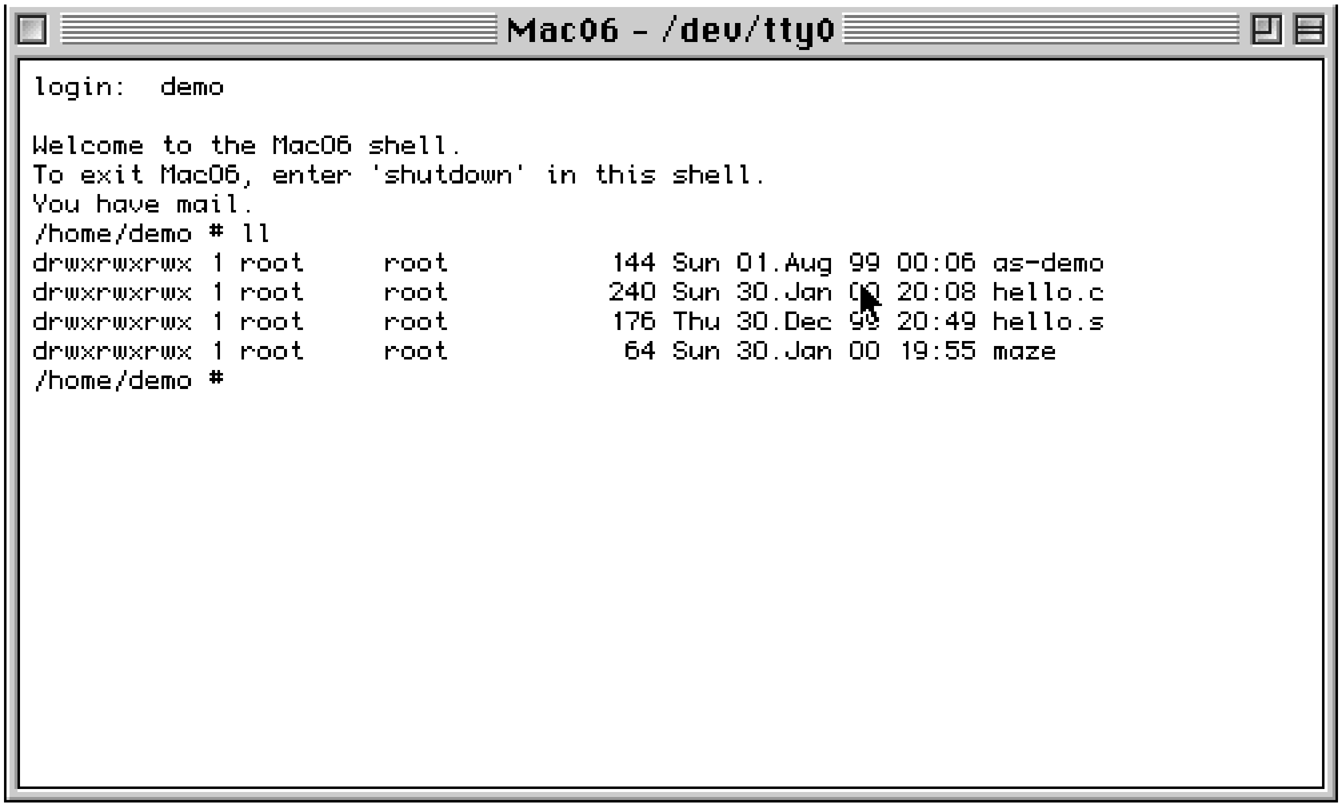
```
*** KERNEL Mac06-1.0-68k booting. ***
*** Copyright @1997-2000 by HNS@DSITRI (http://www.ditri.de). ***
*** All rights reserved. ***

Daemon /bin/daemon/initd is running.
Fri Jun 02 23:58:51 2000

Welcome to the Mac06 shell.
To exit Mac06, enter 'shutdown' in this shell.
/ # |
```

Login Window

In the standard configuration, a second terminal window pops up presenting a

```
login:
```

prompt.



```
login:  demo

Welcome to the Mac06 shell.
To exit Mac06, enter 'shutdown' in this shell.
You have mail.
/home/demo # ll
drwxrwxrwx 1 root     root          144 Sun 01.Aug 99 00:06 as-demo
drwxrwxrwx 1 root     root          240 Sun 30.Jan 00 20:08 hello.c
drwxrwxrwx 1 root     root          176 Thu 30.Dec 99 20:49 hello.s
drwxrwxrwx 1 root     root           64 Sun 30.Jan 00 19:55 maze
/home/demo #
```

As the user name, you can enter any valid user name as defined in `/etc/passwd`. Passwords are asked for but currently not yet verified. The main purpose of this mechanism is that you can control the startup environment by entering different names.

Shutting down

To shut down Mac06, issue the `shutdown` command in the console window. This will force all Mac06 processes to quit. If the registration program is still running, quit it by hand.

Deinstallation

Simply move the `mac06` folder to the Trash. Mac06 does not modify any system file and does not install `INIT`s or device drivers.

System Requirements

- Mac System 7.x or later
- a Mac or Performa with at least 68020 or PPC 60x and 8 MB of RAM
- about 15 MB hard disk space
- Mac06 uses either fat binaries or 680x0 code and therefore should run on all Macs

Some Legal Stuff

Licence Conditions

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

The software is licenced under the following conditions:

- This file and this legal notice are not removed.
- The software is used only for legal purposes.
- The software is redistributed as the original package (.hqx file) and not as restuffed parts of it.
- Redistribution (i.e. copying to other media) of the package is free.
- There is a shareware fee for using the package (i.e. copying to the memory and processor). The price is described in the Shareware Fee section. The right to change the pricing in future releases is reserved.
- Donations to support further development are greatly appreciated! To send a donation, please register with a Bonus.
- Payments through credit cards and checks are processed by Kagi (`mailto:sales@kagi.com`)
- Trademarks (like UNIX, MacOS, Macintosh, Code Warrior, PowerPC, Altivec and so on) are used without further indication. They are property of the respective holder. They are used only as a reference to describe the relationship with these products.
- The software is still under development. Therefore, there is a risk of some bugs where data loss - although assumed to be of very low probability - can not be excluded. Also, the system is not at all secure to hacking, especially if Internet

Introduction(1)

daemons (`telnetd`, `ftpd`) are used. Functions described in standards and text books may be different or missing in all areas of the system. Some deviations are already known and some are not yet.

- This software is distributed as is and neither a hotline nor service can be provided.
- Guarantees or warranties of any kind are excluded as far as legally possible.
- If you run across a bug, please decide yourself to send a mail (`mailto:service@dsitri.de`). Most bugs will be fixed in the next release and may result in some new ones...

Operation

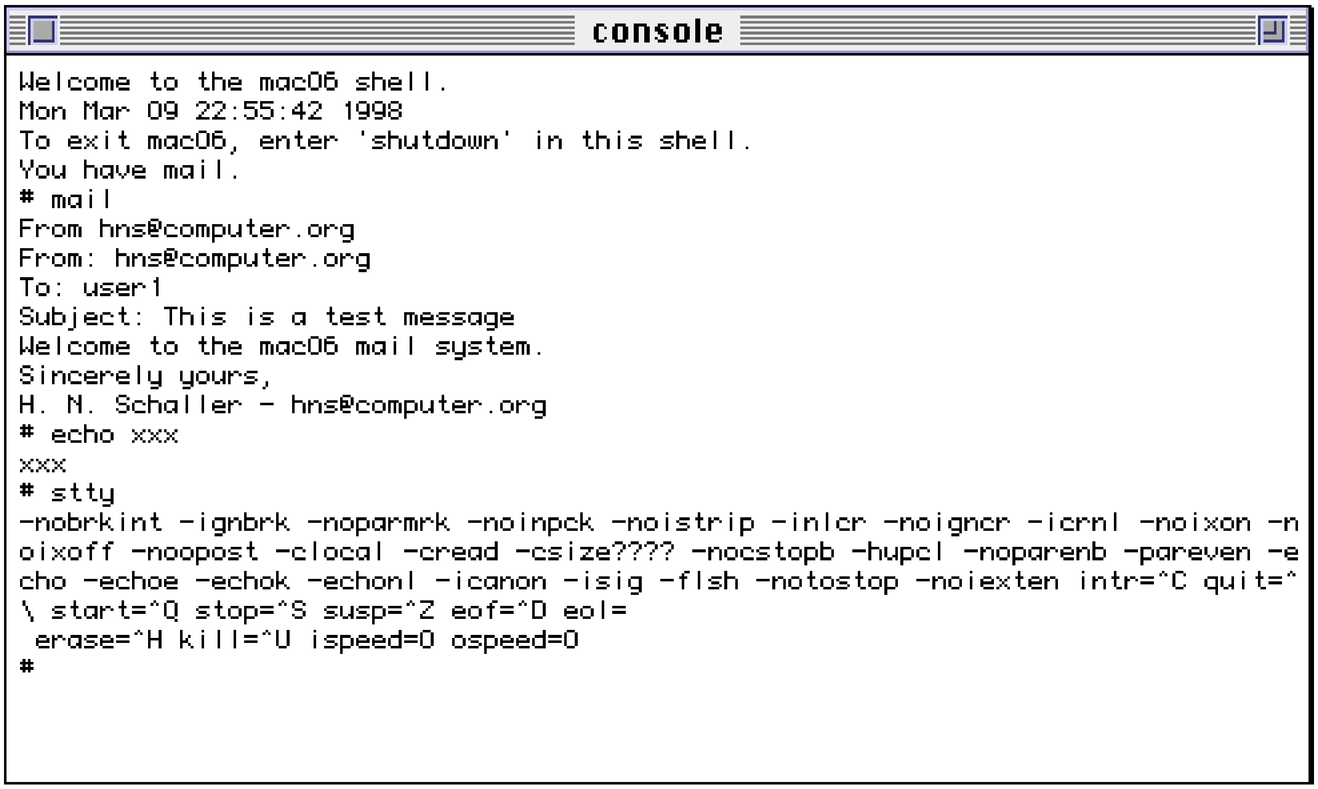
Starting Mac06

Simply double-click on the `Mac06` or `Mac06.68k` application found in the root directory of the distribution package. You may also add a link to this program to the auto start folder. Then, Mac06 will automatically come up whenever your machine is restarted.

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

The Terminal Window

The typical terminal window looks like this:



```
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 with 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. Note that `initd` is instructed to bring up a new terminal window.

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 corner. 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 above, the commands `mail`, `echo xxx` and `stty` have been started and the output of these commands is printed below the command enty line.

So, what are commands and what is output? Commands can be typed in 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.

Operation(1)

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 hints and tricks for shell programming.

The Online Manual

Mac06 provides an online manual that can be addressed with the `man` command. Try

```
$ man sh
$ man man
```

and others.

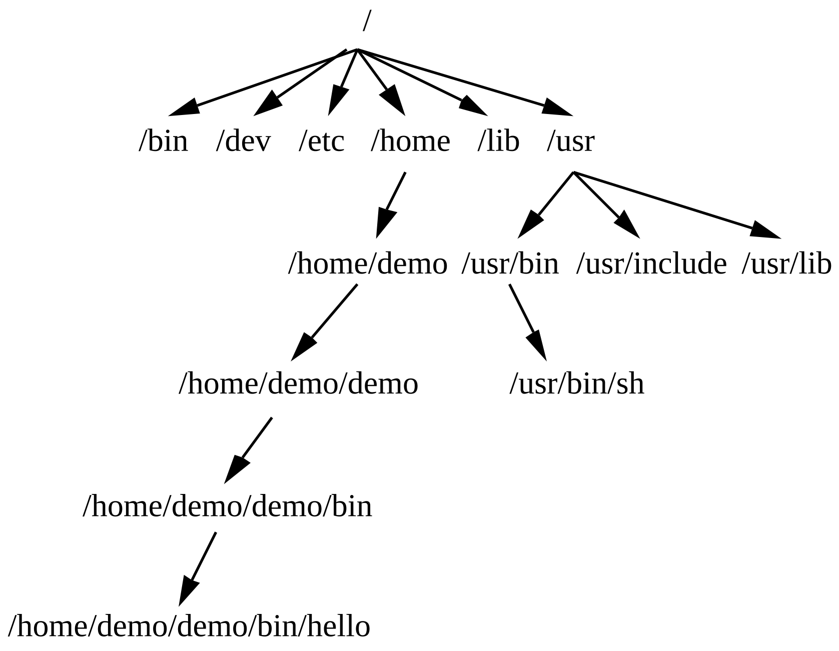
Since the manual is written in HTML, you can also directly go into the manual with your Internet Browser (e.g. Netscape). Just look up the `/usr/man/start.html` file in the Finder and double click.

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.

Operation(1)



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 unless 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 space 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 Permissions

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 Filenames

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, therefore, filenames are mapped according to certain rules:

Finder name	Mac06 name
.	...
..	....
.x	.x
..z	..z
/x	:x
x y	x?y

Operation(1)

Finder name	Mac06 name
x?y	x y

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 visible under Mac06 only and have no direct MacOS naming equivalent.

And, finally, please note that upper and lower case characters are distinguished by MacOS and not in the Finder!

CRLF Mapping

There is one more important aspect to be mentioned, CRLF mapping or line oriented, text file formats. MacOS separates lines by a CR character (`0x0d`) while UNIX typically separates lines by a LF (`0x0a`).

MacOS transparently (or at least mostly transparently) maps CR to LF and vice versa. The rules are as follows:

- Files with MacOS File Type ‘`TEXT`’ are read or written CRLF-mapped.
- All others are not, i.e. are read and written unchanged.
- A call for a new file to `fopen()` without mode `”b“`[1] (binary) or a call to `open()` or `creat()` without mode `O_BINARY` will generate a text file, i.e. a file with MacOS File Type ‘`TEXT`’ and CRLF mapping.
- A call for a new file to `fopen()` with mode `”b“`[2] (binary) or a call to `open()` or `creat()` with mode `O_BINARY` will generate a binary file, i.e. a file with a different MacOS File Type than ‘`TEXT`’.

These rules guarantee that files written from applications within Mac06 are always read back correctly - independently of the proper use of the `”b“` or `O_BINARY` attribute. Therefore, applications ported to Mac06 will function directly. Only of data exchange to MacOS applications is required, special care must be taken that the application uses `”b“` or `O_BINARY` properly in its source code. Then, files can also be handled properly by all Mac applications. The only step to be done manually may be to drag the file onto Binary Pump or change the File Creator and Type by some other tool to enable double clicks on the file in the Finder.

1. e.g. `fopen(“newfile.txt“, “w“);`
2. e.g. `fopen(”newfile.bin“, “w+b“);`

Sending Apple Events

MacOS provides a mechanism for sending Apple Events to applications running under MacOS. This can be used to automate tasks.

An Apple Event is sent by

```
$ AE -A ‘TTXT’ -E ‘AEVTODOC’ x
```

Please refer to the man page for more details.

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` or .- 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. a `SIGTERM` can be sent by using the File menu and selecting the Terminate item.

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 capable 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. The command help will show the list of commands.

dial

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

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

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 produce preprocessor output only, use the `-E` option and for assembler output only, use `-S`.

If `-c` is omitted from the `c89` call, `-lc` (this loads `/usr/lib/libc.a`) 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.o ...`	to specify the creation of a library `l.a`
`PROGRAM p s.o ...`	to specify the creation of a program `p`

Libraries to be linked can be specified 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 one is supported by Mac06 (unless some freely available source code will be ported). But the tool `mk` is provided which has comparable features. Therefore, the `makefile` provided by the software package 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 -llib libname.a ...
```

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 does 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()`[1] 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 cordially invited to have a link to your WWW pages added on

`http://www.dsitri.de/projects/mac06/mac06-APPLICATIONS.html`

Please send an e-mail to `service@dsitri.de`.

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://www.dsitri.de/projects/mac06/mac06-APPLICATIONS.html`

For MPW, there is no recipe available yet.

1. In the POSIX.4 compatibility mode, use `sched_yield()` which is portable if you `#include <sched.h>`.

Generating Installers

Generating installers (like those used for Mac06 itself) is fairly simple by using `mk`. Please refer to the online manual.

Shutting down 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. The Mac06 kernel itself can be killed by pressing alt-apple-.

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.

Removing register application

After paying your shareware fee, you are permitted to remove the automatic startup of the register application. To do this, edit the file `/etc/inittab` and remove or comment out (add a # character at the beginning of the line) the entry that starts `/bin/register`.

Some useful Literature

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

Bill O. Gallmeister, POSIX.4: Programming for the Real World, O'Reilly&Assoc., Inc., Sebastopol, ISBN 0-56502-074-0

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`

J. Strang, Programming with curses, O'Reilly&Associates, Cambridge, ISBN: 0-937175-02-1

Available Things

Please refer to the man pages for a full description of the commands and its arguments. Commands crossed out are currently not available or not yet properly working versions.

Shell

Builtin

. - switch shell script file

cd - change working directory

chroot - change root directory

exec - replace shell by new process/script

exit - exit from shell script

~~export - mark variables to be exported~~

~~newgrp - set new group id~~

read - read from file

~~readonly - mark variables as readonly~~

return - return from function

set - print variable values and set shell options and arguments

shift - shift shell script arguments

~~su - set new user id~~

unset - delete variable

Operators and Control Commands

| - pipeline

& - background execution

&& and || - conditional execution

( ) - subshell invocation

{ } - command grouping

! - negate exit value

~~case x in *) ;; esac - pattern decoder~~

~~for x in do done - argument loop~~

~~if x then elif else fi - conditional execution~~

~~while x do done - repeated loop~~

~~until x do done - repeated loop~~

Runtime

: - comment

AE - send AppleEvent

~~base64 - encode and decode base64 format (MIME)~~

basename - extract basic file name

~~binhex - encode and decode binhex format~~

cal - print calendar

cat - catenate files

~~chgrp - change file group ownership~~

chmod - change file mode

~~chown - change file ownership~~

cmp - compare files

~~code - encode and decode specified format~~

cp - copy file(s)

date - print current date and time

dirname - extract directory path

echo - echo (i.e. print) arguments

ed - launch editor

Available Things(1)

~~egrep - extended "get regular expression and print" command~~

errno - convert system error number to message

expr - evaluate numerical expressions

false - always fail

fgrep - fast "get regular expression and print" command

~~file - determine file type~~

find - find files with given properties

~~grep - "get regular expression and print" command~~

head - print header of a file

hroff - HTML formatter (run-off)

kill - send signal to a process

ll - long directory listing

~~ln - link file(s)~~

login - ask for new user-id and password

~~lp - spool files to printer~~

ls - directory listing

man - lookup in manual

mesg - permit others to write to terminal

~~mime - encode/decode MIME files~~

mkdir - create new directory

more - paginate and wait for user interaction

~~mv - move file(s)~~

~~newgrp - switch to new group id~~

~~nice - run process with different priority~~

nohup - run process in background

od - octal dump

~~passwd - change password~~

~~pr - paginated print~~

Available Things(1)

printf - formatted print

ps - process status

pwd - print working directory

~~rev - reverse lines~~

rm - remove files to the trash

rmdir - remove (empty) directory

sh - shell (command line interpreter, job control)

shutdown - shutdown Mac06

sleep - sleep for specified time

~~split - split file into smaller junks~~

stty - set tty parameters

~~su - switch to new user id~~

sync - flush all buffered data to disk

~~tail - print end of file~~

tar - tape archive

tee - T-connector for pipelines

test - test for conditions (file properties, numerical an logical values)

time - print running time of a command

touch - mark file as modified

~~tr - transpose characters~~

true - always successful

tty - print tty file name

uname - print system information

uudecode - decode uu-format

uuencode - encode uu-format

wait - wait for child process to terminate

wc - word (and character and line) counter

who - print user names

Available Things(1)

~~write - write message to other terminal~~

xarg - extend arguments

xd - hex dump

[ ] - test for conditions (file properties, numerical an logical values)

Communication

~~at - schedule events and commands~~

dial - dial up mailbox through modem

~~ftp - file transfer client~~

~~mail - send and read mail~~

nslookup - look up name server entries

telnet - remote terminal access

Development

ar - archiver

as - assembler

c89 - ISO-C compiler

~~coff2hex - convert COFF to hex format (for EPROMMer)~~

~~c++ - C++ compiler~~

errno - convert system error number into message

~~hdb - HNS debugger~~

ld - linker

mk - make tool

nm - print names in object

size - print size of object

Libraries

crt.o - C runtime

-lc - standard C library

-lcurses - terminal independent, character based window manager

-lld - COFF linker/loader support library

-lm - math library

-lmac - Macintosh support library

-lnet - networking library (Internet)

~~-lPt - Internet protocol toolkit for -lXt~~

-lsocket - socket library

~~-lX11 - small x library~~

~~-lXt - small x toolkit~~

~~-lSmalltif - small Widget library~~