

Running Classic on non Macintosh computers, including Intel Macintosh

Andreas Fischlin¹,
Systems Ecology², ETH Zurich³

CONTENTS

1	BASICS	1
2	EXPERIENCE	2
2.1	SheepShaver v2.3 (14.May.2006) on G4 17" PowerBook	2
2.2	Basilisk II v1.0 (port 19, 1.Aug.2006) on Intel Core Duo 2.33 GHz 17" MacBook Pro	3
2.3	Basilisk II v1.020091004 (port 19, 2.Jan.2010) on Intel Core i7 2.66 GHz 17" MacBook Pro	6
2.4	SheepShaver v2.3 (14.May.2006) on Intel Core Duo 2.33 GHz 17" MacBook Pro	11

Abstract

You can run Classic on Windows machines, Amigas, etc. and notably also on Intel Macintosh machines, for which Apple no longer supports Classic. This text assumes you are mostly interested to run Classic on an Intel Mac.

1 BASICS

You need a Macintosh ROM file, which may be difficult to get and you require the matching system software. Particularly copyright reasons make it considerably harder to meet those requirements. Note, that you are at least legally safe to run Classic, if you own a Mac containing the ROM file you are using.

- **To execute native 68K Code use open source Basilisk II**

http://www.users.bigpond.com/pear_computers
<http://www.gibix.net/dokuwiki/en:projects:basilisk2>
<http://basilisk.cebix.net/>

- **To execute native PPC Code use open source SheepShaver**

<http://www.gibix.net/projects/sheepshaver/help/>
<http://sheepshaver.cebix.net/>

Thus for MacMETH or RAMSES it is best to use Basilisk II, however it is also possible to use the 68k emulator running within SheepShaver. Without any glitch Easy ModelWorks, MacMETH, including the compiler, RAMSES etc. did fully work as expected.

Depending on which solution you choose, you need a different ROM and different System Software.

¹ <mailto:andreas.fischlin@env.ethz.ch>, <http://www.sysecol.ethz.ch/staff/af>

² <http://www.sysecol.ethz.ch>

³ <http://www.ethz.ch>

	ROM	MacOS	Ethernet	Sound
Basilisk II	Classic	0.x .. 7.5	no	no
	Mac II	7.0.1 .. 8.1	yes	yes (>7.5, Sound Manager 3.0)
SheepShaver	2 MB	7.5.2 .. 9.0.4	?	no

Only a portion of above characteristics were tested by me. I filled out this table using READ MEs and info I found on the web. However, I use regularly and successfully since many years Basilisk II 1.0, Mac OS X port 19 with Mac OS 7.6.1 and a ROM from a Quadra 800 (details given below).

Older ROM sizes range from 68K (Mac 128, 512) to 512K (e.g. Ilfx, Ilci). The most advanced Mac II ROM is the 1024K ROM present in following machines:

Mac Color Classic
 Mac Color Classic II
 Mac LC III, LCIII+, 550 (?), 475, 575, 580, 630
 Mac Ilvx, Ilvi
 Performa 275, 460, 475, 476, 520 (?), 550 (?), 560 (?), 575, 577, 578, 580, 588, 600, 630
 Centris 610, 650
 Powerbook 140, 145, 145B, 150, 160, 165, 165c, 170, 180, 180c,
 Powerbook Duo 210, 230, 250, 270c, 280, 280c
 Quadra 605, 610, 630, 650, 700, 800, 900, 950

The 2024 ROM you need for SheepShaver is available from Apple
http://download.info.apple.com/Apple_Support_Area/Apple_Software_Updates/MultiCountry/Macintosh/System/Mac_OS_ROM/Mac_OS_ROM_Update_1.0.smi.bin
 and is contained in following machines:

PowerBook 190, 190cs, 520, 520c, 540, 540c, 550c
 Quadra 660AV, 840AV

2 EXPERIENCE

2.1 SheepShaver v2.3 (14.May.2006) on G4 17" PowerBook

- During startup SheepShaver generates a confusing message that "SheepShaver" has crashed although it has actually not crashed. Simply dismiss this message.
- Double clicks do not work reliably. This requires some get used to.
- Setting the startup disk to another volume does not what you expect it to do. To really boot e.g. from a CD (pressing C during boot does not help, settings in SheepShaverGUI are ignored) you need to make the installed System Folder disfunc by e.g. moving the System file out of the System Folder.
- I could not connect to the internet. Don't know why. But I don't consider this to be a particular drawback, unless you wish to print.
- From what I read about Basilisk, I expect it to behave even better than SheepShaver. I expect it to be able to print and connect to the internet.

I believe the optimal solution for MacMETH and RAMSES would be MacOS 8.1 running under

Basilisk II using a 1024 K ROM file.

2.2 Basilisk II v1.0 (port 19, 1.Aug.2006) on Intel Core Duo 2.33 GHz 17" MacBook Pro

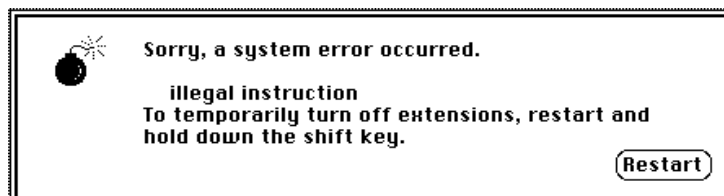
This concerns experience I made using "Basilisk II 1.0, Mac OS X port 19" (Copyright © 1997-2006 Christian Bauer et al. Freely distributable under the terms of the GNU GPL.), a port which was prepared by Nigel Pearson (http://web.netcall.com.au/~pear_computers/BasiliskII.html). Its icon is the following:



and I name this application now "BasiliskII-Nigel Pearson.app" to avoid confounding it with other variants (see next section). This Basilisk works amazingly fast and demonstrates how much performance we lost with making the software slow by burdening it with many whistles and bells. System 7.5.3 launches or restarts within a second on my 2.33 GHz Intel Core Duo!!!!

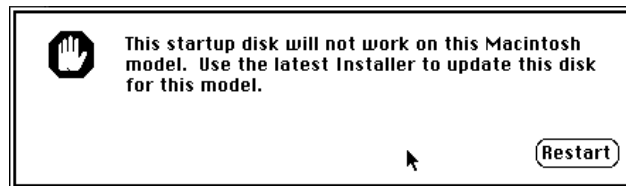
At <http://www.emaculation.com/articles/intro.html> I read "... 1MB ROM with the checksum "420DBFF3" from the Macintosh Quadra 900 does work with Basilisk II." I found only a ROM on a Quadra 800 with checksum "F1A6F343" (Major Version 067C, Minor Version 23F1, Sub-Release 0000 as returned by CopyROM application).

However, it seems that basic configurations (CPU, FPU, model etc.) that would fit precisely those of the Quadra 800 where the ROM is coming from are actually not well supported and lead to a system crash during startup phase. Notably, this is the case for one of the following configuration combinations: System 7.5.3 and 7.6.1: (i) CPU 68040 (with any other combination for FPU, model), (ii) FPU checked (with any other combination). The only specification combinations that seem to work are 68020 or 68030 and Ilci or Quadra 900, which can fortunately be combined freely (Note, Ilci had the 68030 CPU and Quadra 900 had the 68040 CPU). Note also that the Ilci supports only up to MacOS 7.6.1 whereas Quadras support up to MacOS 8.1.



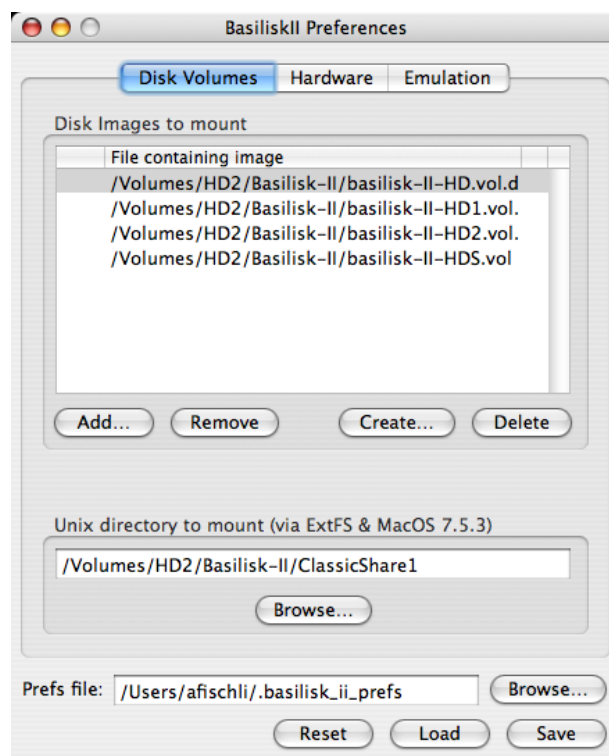
- Error message generated during startup of Basilisk II if the specification "With FPU" has been checked in Basilisk's preferences.

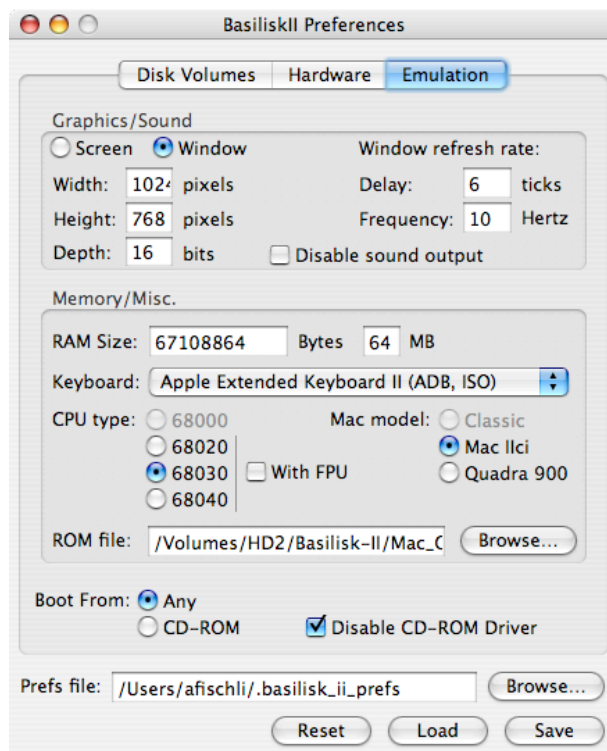
Using System 8.1 the situation is similar. The only difference is that we need now to specify model Quadra 900, since Mac IIci is officially no longer supported to run this MacOS.



- Error message generated during startup of Basilisk II if the specification "Mac IIci" has been selected in Basilisk's preferences and booting MacOS 8.1.

Yet, again we need to choose CPU 68030. The CPU 68040 of the true Quadra 900 crashes the system. The experience with MacOS 8.1 as copied from release server, however, is not satisfying for a lack of system stability. RAMSES shell or MEdit occasionally crash the system for unknown reasons. Yet, the very same configuration seems to work fine for several launches of the RAMSES shell. Either a reinstallation from scratch or further experience with different system configurations (see Extensions Manager) will hopefully clarify this. Otherwise it might be recommendable to work merely with 7.5.3, albeit 7.6.1 is considered the most commendable system 7. Those two systems have shown to support Classic reliably enough with the following Basilisk configurations:





Configuration of Basilisk II: Disk Volumes and Emulation tab as used with a System 7.6.1 custom installation.

The Control Panel "Startup Disk" is not helpful to switch to another startup disk. It seems that Basilisk uses the first disk listed as the startup disk. Editing `~/basilisk_ii_prefs` using `vi` works fine to quickly swap a second disk to use as another startup disk.

I had difficulties to change the screen size in window mode using the preference panel of Basilisk. Only one dimension could be changed, Basilisk generated error messages that the screen resolution would not be supported and Basilisk would attempt to revert to the default tiny 512x384 resolution. However, in fact MacOS did not work and hanged when I attempted to restart. However, using `vi` to edit `~/basilisk_ii_prefs` worked very well and I could set dimensions of the screen to my favored 1024 x 768 (I tested from 512x384 and 640x480). The control strip module "Monitor resolution" works. However, note, the module "Monitor BitDepth" fails and will corrupt the system (discard "Display Preferences", should this happen).

The problems with file exchange under Basilisk are of course the same as with SheepShaver. File type and creator end up corrupted and the same methods need to be used to successfully transfer files back and forth (see below). Note also, using special characters such as `f` in file names results in spurious disk errors when copying such a file to the 'Unix' volume from within Basilisk. Changing the file names so that only ASCII characters are used avoids this problem.

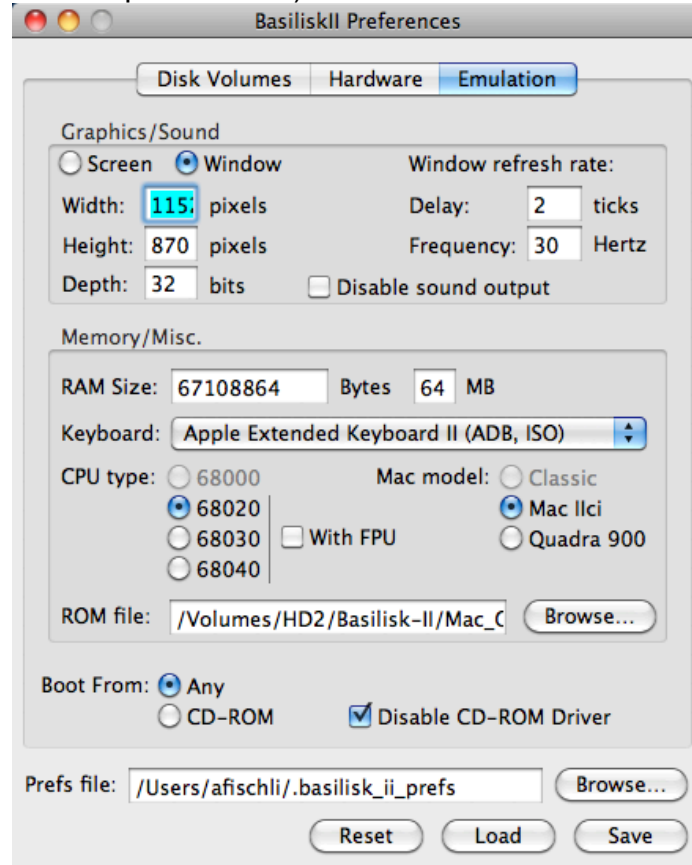
Networking did not function at ETHZ in my office despite of what the HowTos.html help page states (resides within Basilisk II).

Volume philosophy: First remember, that pre 8.1 MacOS do not support HFS+. Thus all volumes used by the Basilisk emulator should be formatted as HFS and not HFS+ volumes! It is easy to create additional volumes, given you have the disk space. My experience is good with creating a volume using Basilisk, using extension ".dmg" and format it while launching the emulator. Later on you can open such a volume from within OS X.

I keep for safety reasons a volume HDS with the 7.5.3 system software. On purpose I do

not use the extension ".dmg" to avoid to open this file ever under OS X. This allows any time to run the Basilisk emulator in case the system in daily use should get misconfigured or otherwise corrupted. I also kept a record of all software installed. Finally, any RAMSES work is done on the usual HD2 volume only, again keeping changes to the startup volume HD separate. The latter changes little, thanks to its fairly stable applications folder.

The nervousness of the progress bar can be reduced by setting the slider on left bottom to about 30 Frames/sec (see also preferences):



2.3 Basilisk II v1.020091004 (port 19, 2.Jan.2010) on Intel Core i7 2.66 GHz 17" MacBook Pro

My latest 17" MacBook Pro (grandeur3) runs as of this writing under Mac OS X 10.6.6. I used on there Basilisk II version v1.020091004 (Basilisk II 1.0 (2009-10-04), Copyright © 1997-2006 Christian Bauer et al.). The latter is much simpler, has e.g. no menu command "Preferences...". Instead it comes with a 'BasiliskIIGUI.app' (Basilisk II version 1.0, Copyright © 1997-2006 Christian Bauer et al. Mac OS X port 17). It comes with a nice icon:



This version has the advantage of converting file types and creator data correctly when exchanging files through the Unix/ '/Volumes/HD2/Basilisk-II/ClassicShare1' folder. This is very convenient. The binary seems also to have been built rather recently. However, the performance is by default very poor compared to the one of the Nigel Pearson variant (see previous section and below for improvements on this).

I also detected confusing numerical problems when the configuration was not properly set on my Intel. If I did set the CPU to 68020 with FPU, or 68030 with FPU, or 68040 some SANE functions such as Ln were no longer working properly and caused wrong calculations, in particular for small x values, DMMathLib.Ln (SANE variant) returned always the same wrong and large value 2.977044467802188E+009:

The screenshot shows the Basilisk II source code editor with the 'Source' tab selected. The code defines a module 'TestLn' with a procedure chain and data sections. The procedure chain shows a programmed HALT at address 19650. The data sections show the values of variables x, y, lx, and ly.

```

MODULE TestLn;

(*
  Implementation and Revisions:
  =====
  Author   Date       Description of change
  -----
  AF       03/15/2011  First implementation
*)

FROM DMMathLib IMPORT Ln;
FROM DMLongMathLib IMPORT LongLn;

VAR x,y: REAL; lx, ly: LONGREAL;

BEGIN (* TestLn *)
  x := 0.018;
  y := Ln(x);
  HALT;
  lx := 0.018D;
  ly := LongLn(lx);
END TestLn.

```

Procedure Chain

TestLn	in TestLn	in System
Address 19650		

Data 1

TestLn	Type	Value
x	REAL	+1.80000E-02
y	REAL	+2.97704E+09
lx	REAL	+1.800000000000000E-002
ly	REAL	+2.977044467802188E+009

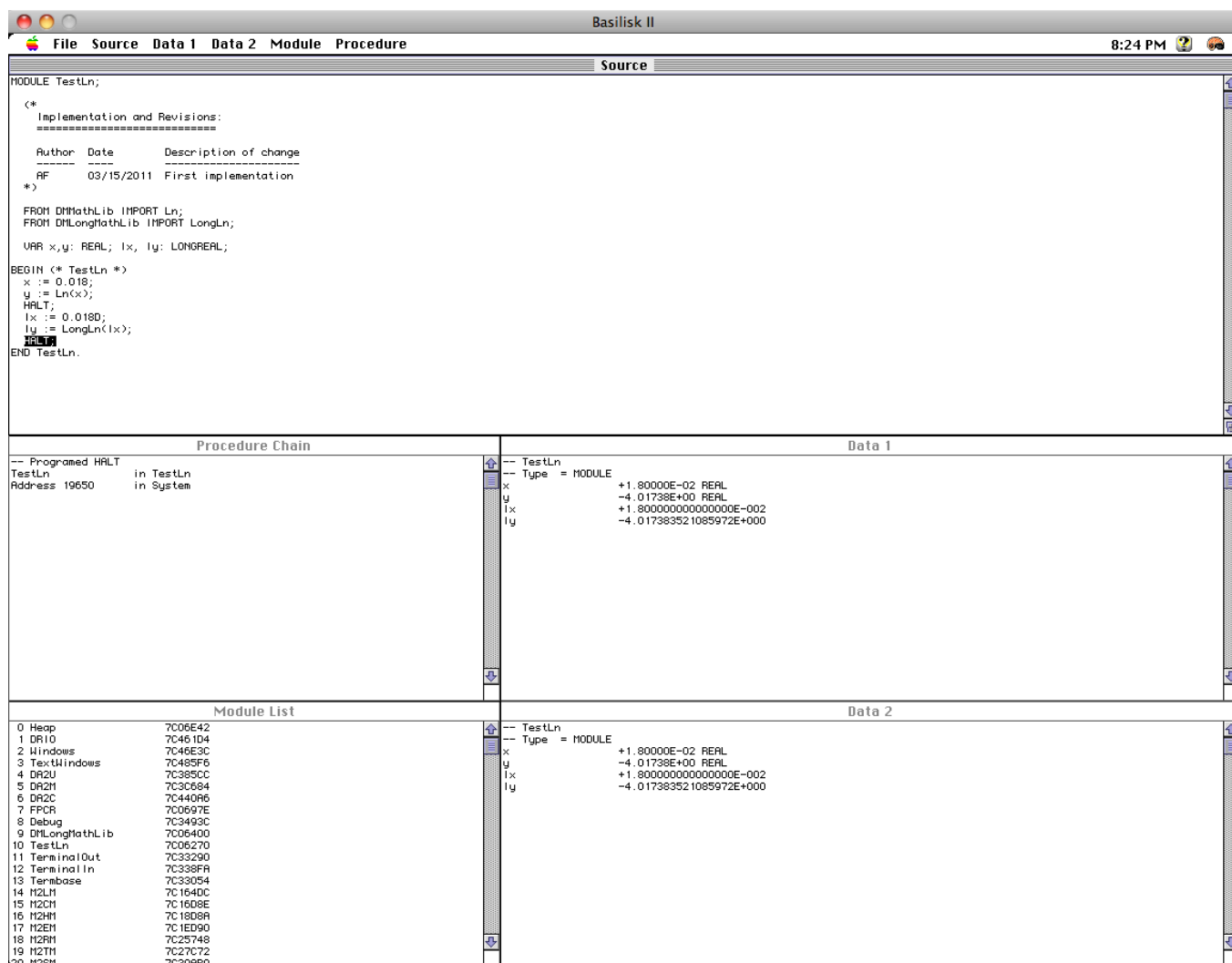
Module List

Module	Address
0 Heap	7C06E36
1 DR10	7C46200
2 Windows	7C46E68
3 TextWindows	7C48E22
4 DR2U	7C385F8
5 DR2M	7C3C6B0
6 DR2C	7C440D2
7 FPCR	7C06976
8 Debug	7C34968
9 DMLongMathLib	7C063F8
10 TestLn	7C06278
11 TerminalOut	7C332BC
12 TerminalIn	7C33926
13 TermBase	7C33078
14 H2LM	7C164F8
15 H2CM	7C16DAA
16 H2HM	7C18DA6
17 H2EM	7C1EDB4
18 H2PM	7C2576C
19 H2TM	7C27C96
20 H2SM	7C30A94

Data 2

TestLn	Type	Value
x	REAL	+1.80000E-02
y	REAL	+2.97704E+09
lx	REAL	+1.800000000000000E-002
ly	REAL	+2.977044467802188E+009

As soon as one avoids demanding a FPU, all is fine and logarithms are again computed properly:

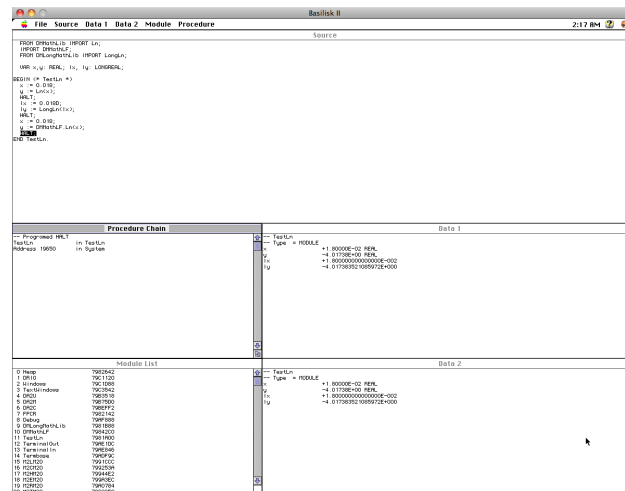


Note a difficulty when using BasiliskIIGUI.app to change preferences. It seems your edits are not really written into the file `~/basilisk_ii_prefs` unless you launch application BasiliskII by pressing the Start button in the lower left corner of the BasiliskIIGUI.app windows.

There were first great performance losses observable to the previous version. I increased memory availability from 64 to 128 MB, having read in a BasiliskII forum that this may help. However, what really makes a big difference is whether one checks the flag "Don't Use CPU When Idle". If that is left unchecked the performance is greatly increased and comparable to what it was using Nigel Pearson's Basilisk II. Speed comparison (LBM 8.0, 2005 parameter identification: 38.9, 38.3⁴ sec (Nigel Pearson's Basilisk II, 60 Frames/sec); 29.6, 30.2 sec (Nigel Pearson's Basilisk II, 30 Frames/sec); 35.8, 36.0 sec (Basilisk II v1.020091004)) showed that I have then about the same performance as with the version I used previously (see previous section). The only drawback from this high speed is the fact that the ventilator is caused to become very active and the 'Activity Monitor.app' show large values of 200.1 % CPU activity for the process BasiliskII. In any case, having memory at 128 MB seems to slightly improve performance as well: 31.1, 34.7 sec (Basilisk II v1.020091004, but with 128 MB memory).

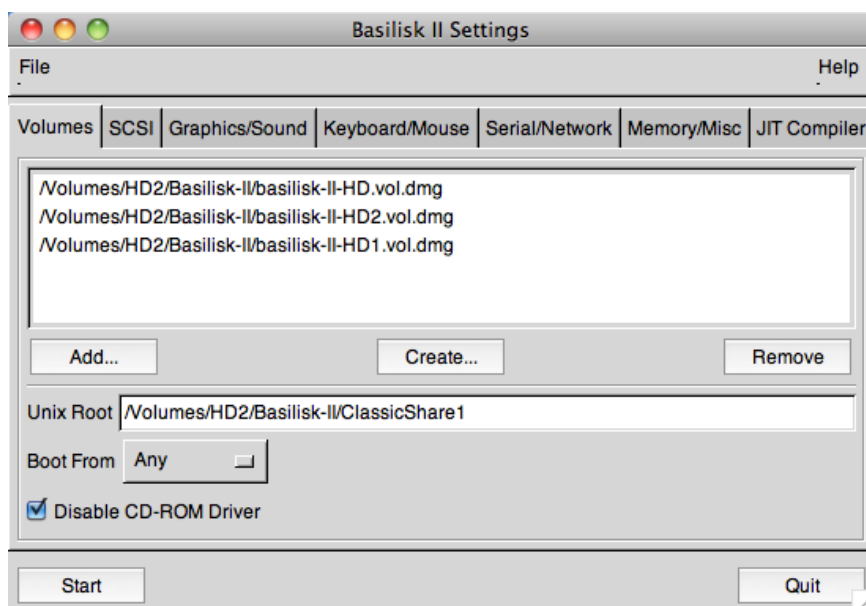
Finally, I started also using the Quadra 800 ROM. It seems this made it possible to activate the CPU Type "68020 with FPU" and to use the FPU dependent MacMETH compiler Compile20 and any herewith compiled OBMs (see e.g. test program TestLn.MOD). Ex.:

⁴ Made always two measurements of time needed



Moreover, using the FPU option did speed up things considerably more: The same tests as described above now needed only 18.4 sec⁵ (Basilisk II v1.020091004)!!

To conclude, the following settings made by BasiliskIIGUI.app seem to have worked best and BasiliskII v1.020091004 runs hereby in the end surprisingly efficiently⁶:



⁵ Twice the same number of seconds in the two tests I made

⁶ I now recommend to change little of these settings. Notably you must not activate “CPU Type“ 68040, although it might first give the impression to work, or you risk getting wrong numerical results. The only options that can be altered freely are (1) to check “Don’t Use CPU When Idle“ to slow down the ventilator, which becomes otherwise very active, but at the cost of significant performance decrease; (2) to use only “CPU Type“ 68020 without a FPU.

Basilisk II Settings

File Help

Volumes | SCSI | **Graphics/Sound** | Keyboard/Mouse | Serial/Network | Memory/Misc | JIT Compiler

Video Type: Window

Window Refresh Rate: Dynamic

Width: 1152

Height: 870

☐ Disable Sound Output

Audio Output Device: /dev/dsp

Audio Mixer Device: /dev/mixer

Start Quit

Basilisk II Settings

File Help

Volumes | SCSI | Graphics/Sound | **Keyboard/Mouse** | Serial/Network | Memory/Misc | JIT Compiler

☒ Use Raw Keycodes

Use Raw Keycodes: /Volumes/HD2/Basilisk-II/BasiliskII_keycodes Browse...

Mouse Wheel Function: Cursor Up/Down

Lines To Scroll: 3

Start Quit

Basilisk II Settings

File Help

Volumes | SCSI | Graphics/Sound | Keyboard/Mouse | Serial/Network | **Memory/Misc** | JIT Compiler

MacOS RAM Size (MB): 128

Mac Model ID: Mac IIci (MacOS 7.x)

CPU Type: 68020 with FPU

ROM File: /Volumes/HD2/Basilisk-II/Mac_OS_ROM Browse...

☐ Don't Use CPU When Idle

☐ Ignore Illegal Memory Accesses

Start Quit

A further caveat: Switching between the two versions of BasiliskII as described in this section and the previous one is likely to alter the preferences in a conflicting manner. BasiliskII from Nigel Pearson does cause a change in the „Memory/Misc“ settings so that BasiliskIIGUI.app needs to be rerun to restore above given settings. The reasons seems to be a bug in the preferences settings of Nigel Pearson's BasiliskII: It seems not to remember another preference file than the default one (~/.basilisk_ii_prefs), regardless what one chooses in the preference dialogs and therefore the two BasiliskII applications share the same preference file always. (This seems to be a bug Nigel Pearson's BasiliskII, since it forgets after a restart the changed text preference file, despite the fact it uses its own preference file in ~/Library/Preferences.)

However, although Basilisk II v1.020091004 appears to be superior to the previous one by Nigel Pearson if configured correctly, it occasionally still crashes and requires the startup disk to be fixed. Thus my little utility 'repairBasilisk.command' is still crucial for a smooth use of Basilisk II.

As of 21.Apr.2011 I found yet another ROM at <http://www.redundantrobot.com/sheepshaver-tutorial/>. According to that tutorial Basilisk II is supposed to work best with the Performa ROM, not the Quadra ROMs. I will try this out. Interestingly, the size of the two variants (Performa vs. Quadra 800) is equal. I will see whether I can detect a difference in the stability.

2.4 SheepShaver v2.3 (14.May.2006) on Intel Core Duo 2.33 GHz 17" MacBook Pro

Compared to Basilisk or actual Classic under MacOS X SheepShaver is very slow in running 68K code such as RAMSES. However, it supports the more recent MacOS 9 systems, yet not the last 9.22 as required for actual Classic and as available under MacOS X on PPC Macs.

The following problems arise only on Intel Macs. They are due to the little Endian philosophy of the Intel processor, an inheritance from the 8-bit age, one of the clear disadvantages of the Intel processor family.

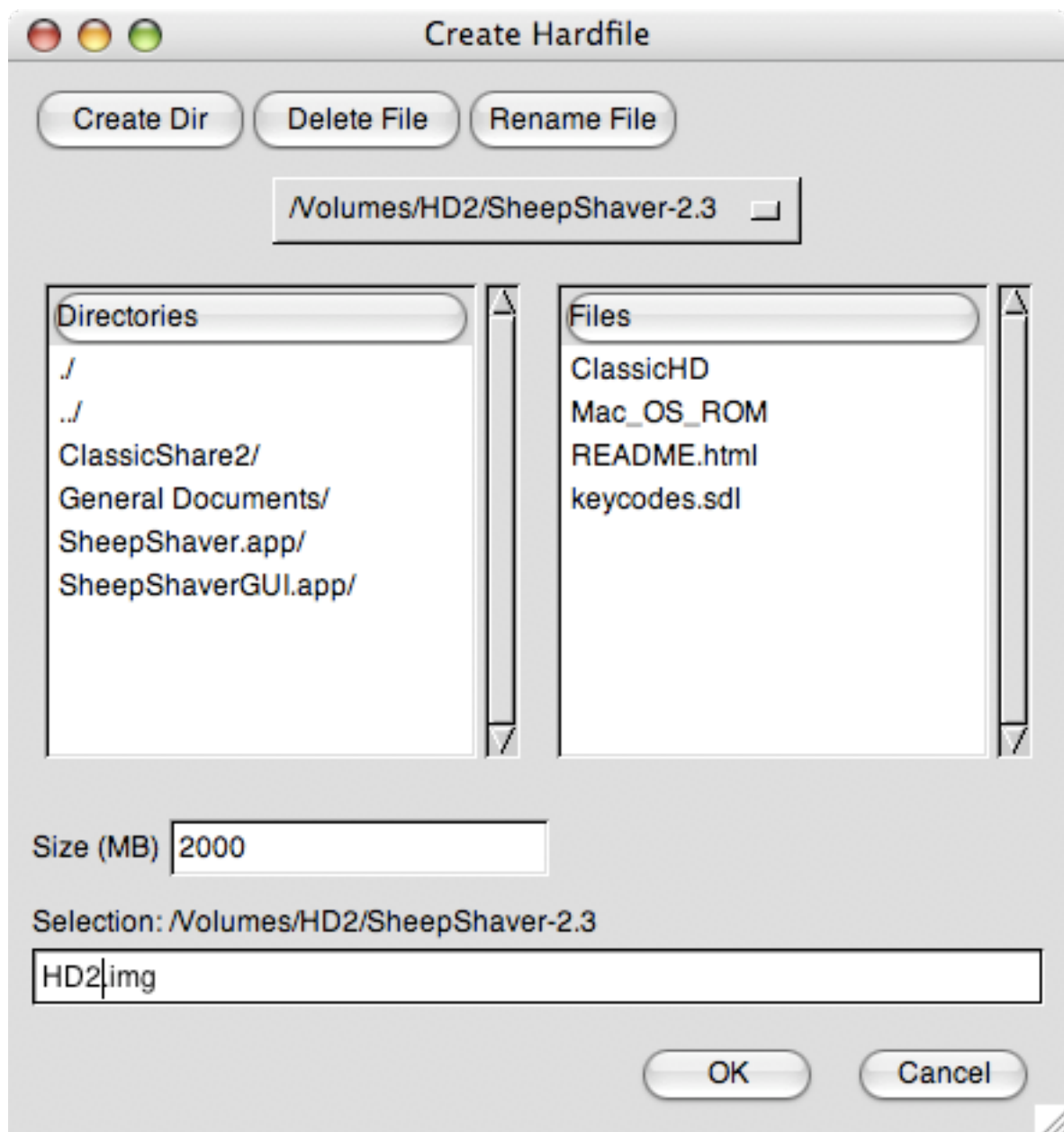
Transferring files between OS X and SS: It is important to note that the folder to share files under SheepShaver (SS) is a volume formatted as a "Unix Directory Tree". Such formatted disks do NOT support Macintosh file storage by default. If you use the Finder to copy under OS X files to this directory structure, on my machine this is directory '/Volumes/HD2/SheepShaver-2.3/ClassicShare2/', the result are typically files, which can't be used within SS. They appear no longer as recognizable by the Finder. For instance if you copy a self mounting Classic disk image (extension .smi, creator 'oneb', file type 'APPL') from OS X to that shared directory, it will look fine if seen from the OS X side, but from within SS the file looks not as the expected self mounting Classic disk image. Closer inspection, e.g. with ResEdit menu command 'File -> Get File/Folder Info...' shows that the file type has been corrupted to 'LPPA' and the creator to 'beno'. In some cases the files may also become hidden, which you can see under the section 'Finder Flags', where the check box 'Invisible' may be inadvertently checked. The latter is another consequence of the little Endian problem with Intel processors, depending what the Finder Flags of the original file are. You can fix all these settings by correcting the file type, the creator and the Finder Flags. It is recommended to uncheck the Finder Flag 'Inited' to ensure all is properly recognized by the Finder within SS and then save the changes. The result will be a functional a self mounting Classic disk image. However, you can't mount it on the exotic 'Unix' volume, you have first to copy it to another, HFS formatted, ordinary Macintosh volume, e.g. the startup volume, before you can really mount it. BTW, note that the changes are not only tedious to make, using ResEdit to fix for every file these properties is a task of no practical value. There is also another disadvantage of this technique: Under OS X the files are now corrupted and need to be fixed by reversing all settings before you can reuse them. Moreover, be warned, a file now corrupted under OS X may look still ok, since its extension may determine its behavior, despite the fact that creator and file type are now corrupted. Thus a smi disk image may mount properly under OS X, despite the fact its creator and file type are wrong. Copying such a file via Finder to a Classic system may result in a disfunc file. In some cases problems arise also under OS X notably on a G4 or G5 machine under Classic.

Conclusion: All in all the use of SS's Unix directory is most confusing and in general not recommended on an Intel machine.

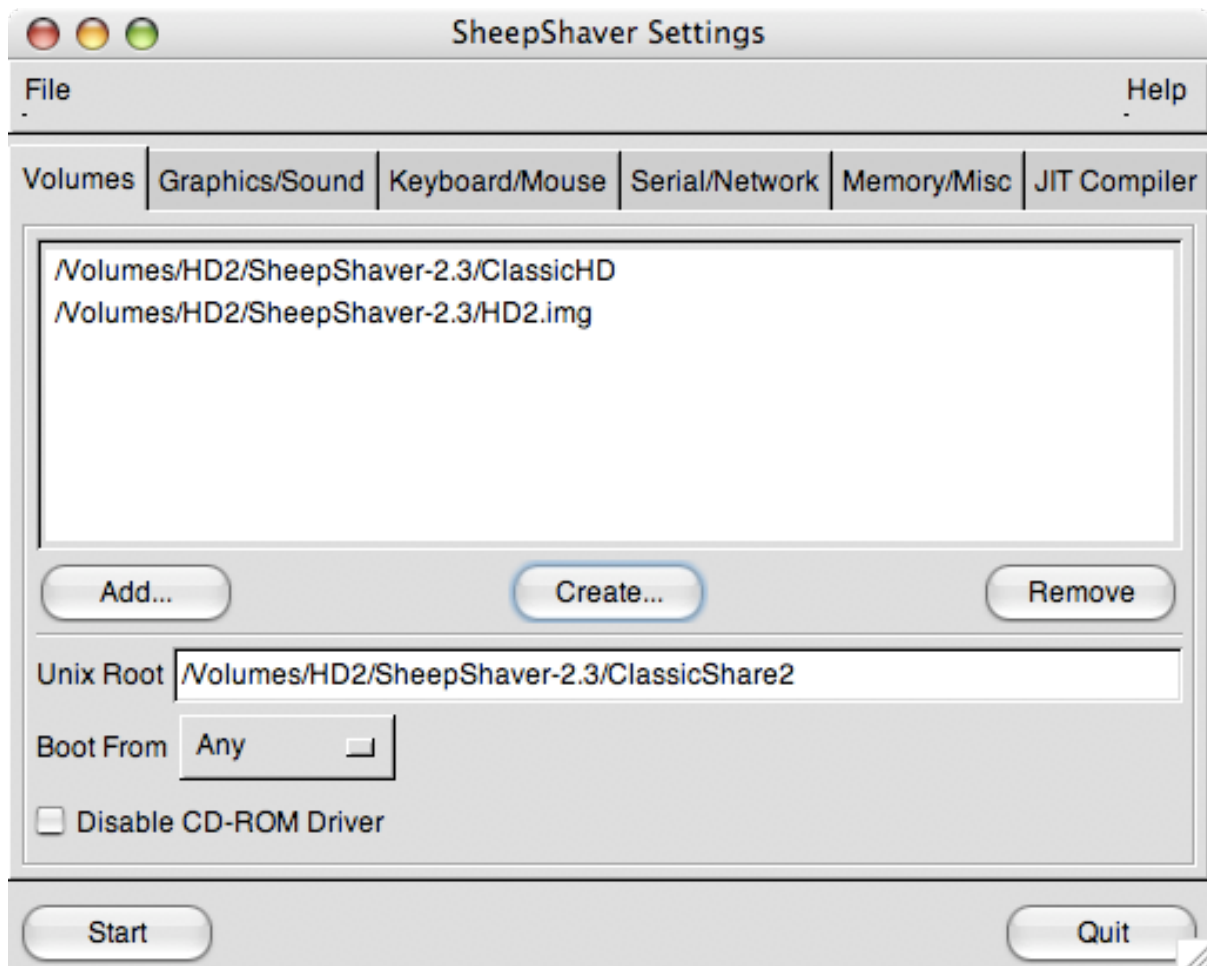
However, there is one use of it: While running SS you can transfer files quickly without having to go through a quit-relaunch cycle of SS. Following this procedure: Transfer the files as StuffIt archives. E.g. use StuffIt Deluxe, to stuff any files, including entire directory structures and copy the resulting .sit file(s) to the shared directory. Unstuff the .sit file(s) in the target system by dragging and dropping them onto 'StuffIt Expander' and you get the wanted files. This works regardless of creator and file type and 'StuffIt Expander' actually fixes those during the expansion process. Quite handy. Note, however, you should use the Unix volume only for transfer and not execute anything on it, many applications refuse to work on the file system of SS's 'Unix' volume.

There is one other technique you can use: Create a disk image, I recommend a Classic disk image with extension .img to transfer files without having to go through the StuffIt/Unstuff conversions. The disadvantage of that technique is that it can't be used on the fly. You need to go through a quit-relaunch cycle of SS. The advantage is that you can really use the same file structure from both systems at a time.

The most convenient technique to obtain such a shared volume is to use the SSGUI (SheepShaverGUI.app) to create a disk image file of the needed size and extension .img. Example: We wish to create file 'HD2.img' of 2 GB size, the largest possible volume under SS's Classic you can get. Use button "Create..." in the Volumes tab of the SSGUI application, double click the directories until you reach the directory where you wish to store the image file, in this case on one of my partitions called HD2 within the folder 'SheepShaver-2.3' and then give the disk image file the wanted name and don't forget to use extension .img. The result is a window similar to this:



Press button 'OK' and wait till the disk image file has been created, which may take a while if your image is large. You should get a window similar to this:



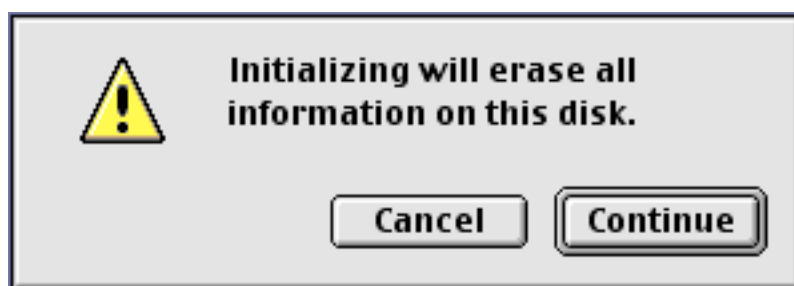
and you can now press button 'Start' to launch SS. If all goes well the first thing you will see is a request by the Finder within SS to format the file with a dialog similar to this:



Name the disk to your liking, I will here use 'HD2'. If you use a MacOS 8.1 or later within SS, I recommend to use the Mac OS extended format (also known as HFS+), especially if you plan to work with many small files on this volume. Otherwise use the default Standard format:



Allow the process to continue by clicking the default button "Continue"



and the result is the wanted disk. From within SS you can use this disk as a separate volume, from OS X you can mount the disk image too and access the files. However, you should refrain from doing that at the same time. Thus quit first SS before you mount the disk image under OS X. Other than that constrain you can freely copy files to and from the disk from both systems, i.e. the Finder within SS and the Finder within OS X on your Intel Mac.

Should you wish to exchange files on the fly, there is a technique available to do that easily. Select the volume and create an alias to it, e.g. similar to this:



Given you have quit all applications and closed all files stored on the disk you can move the volume to the trash (or Cmd-Y while having it selected, ignore the warning that the volume will reappear after restart)

Then you can mount the disk image in OS X, e.g. to copy files to it, unmount it (select it and Cmd-E) and you can regain the wanted volume by simply double clicking the alias without having to go through a quit/relaunch cycle with SS.

Note, I found no problem in formatting the disk image anywhere, i.e. the disk image worked if formatted in OS X or formatted in Classic within SS. To hide annoying folders such as '.Trashes' as created by OS X make them also invisible within Classic by setting the Finder Attribute 'invisible', e.g. using ResEdit.

There are of course other methods possible such as using a remote file server, burning a CD-RW, or using a USB stick. Also using commando tools such as /Developer/Tools/CpMac in combination with /usr/local/bin/stuff and /usr/local/bin/unstuff may be useful to write shell scripts or Apple Scripts doing the transfer. The tool stuff and unstuff are installed by installing StuffIt Deluxe, but can also be installed separately and independently from StuffIt Deluxe.

Editing preferences of SS: SS uses simple text file '~/.SheepShaver_prefs'. You can edit that with any text editor after having renamed the file in a Terminal session or using the SSGUI.

USB sticks and CDs: To mount removable devices such as a USB stick, use SSGUI after having quit SS and in the tab "Volumes" click button "Add New". In the subsequent menu navigate to your USB stick and select it. Proceed in the same way for a CD. (Re)launch SS.

Bootting from CD: Do the same thing to mount a CD. If you want to boot Sheep Shaver from a CD, under the volumes tab, change boot from Disk Image to CD Drive (I think)

Incompatibilities: Control Panel BeHierarchic crashes the system (boot with Shift pressed to turn extensions off and boot the corrupt system).

3 CREATING DISK IMAGES

I used under Snow Leopard, where some of the experiences described under section 2.4 (SheepShaver) may no longer be true (notably HFS disk images are no longer writable, only readable under Mac OS X 10.6.x (Snow Leopard) or later). The AppleScript based utility "Emulator Disk Image Chooser.app", however, works very well as of this writing (31.May.2011, Mac OS X 10.6.7) and can easily create new disk images and make them known to Basilisk II

very conveniently. I ran into a disk limit on volume HD2 and could easily swap it for a newer, much larger volume using this utility.

af, ETH Zurich, 12.Oct.2006, 30.Dec.2006, 30.May.2007, 2.Mar.2011, 16.Mar.2011, and 31.May.2011

<http://www.sysecol.ethz.ch/>
<http://www.sysecol.ethz.ch/people/afischli>
<mailto:andreas.fischlin@env.ethz.ch>