PCE/macplus, the ultimate 68K Macintosh emulator

November 25, 2016 ToughDev 4 Comments 68k, emulator, mac os, pce/macplus

5.00 avg. rating (**94%** score) - **1** vote

UPDATE: Retro68 together with CodeLite and pce/macplus emulator can form a very good 68k development environment. See my latest post for more details.

I have long been trying to find a fully-fledged 68k Macintosh emulator that can run System 7 and older, with support for sound and serial port communications. Mini vMac does not suit my requirements due to lack of serial port emulation. Basilisk II, despite being a very capable emulator with support for Ethernet, does not officially support 24-bit addressing and hence will not be able to run Mac systems older than 7.5.5, at least not without the MODE32 extension.

Fortunately things changed when I came across PCE/macplus, a highly customizable emulator that can emulate many 68k compact Macintosh models from the Macintosh 128K/512K up to the Macintosh SE. Serial emulation, which is what I need, is also supported as well. This article will describe my various experiments with PCE/macplus and share some interesting findings.

Downloading the binaries

The Windows emulator binaries for various Macintosh systems together with the source code can be downloaded from PCE's download page. If you are interested, on the same page you will also find emulators for PC XT, Atari, CP/M, and other machines.

Running PCE/macplus on Windows should be straight forward. Just extract the ZIP file, locate the **run.bat** batch script, and execute it. With some luck, you should be able to see your favorite 68k Macintosh system on the emulator screen shortly. However, since many exciting features of PCE/macplus such as PPP emulation via the serial port will not be supported on Windows, with some time at hand, I decided to spend some time compiling PCE/macplus for Ubuntu, my favourite Linux distro to be able to explore its full set of features.

Compiling pce/macplus from source

First download the source from the same download page and extract it. After that, install the required X11 and SDL development libraries:

```
sudo apt-get install libx11-dev libx11-doc libxau-dev libxcb1-dev
libxdmcp-dev x11proto-core-dev x11proto-input-dev x11proto-kb-dev xorg-
sgml-doctools xtrans-dev
sudo apt-get install libsdl1.2-dev libsdl1.2debian
```

Next, configure PCE/macplus for building with support for most of the available features:

```
./configure -with-x --with-sdl --enable-tun --enable-char-ppp --enable-
char-tcp --enable-sound-oss --enable-char-slip --enable-char-pty --enable-
char-posix --enable-char-termios
```

This part needs a bit of explanation. The provided options enable PPP emulation via the serial port (needed for Internet access since PCE/macplus does not emulate the Ethernet card) and enable building with SDL (required for sound support) and with X11 libraries (important, otherwise there

About Us

Since 2008, ToughDev has been publishing technical sharing articles on a wide range of topics from software development to electronics design. Our interests include, but are not limited to, Android/iOS programming, VoIP products, embedded design using Arduino/PIC microcontrollers, reverse-engineering, retro-computing, and many others. We also perform product reviews, both on new and vintage products, and share our findings with the community. In addition, our team also develops customized software/hardware solutions highly adapted to suit your needs.

Read our latest review of the DS1054Z/GDS1054B/DSO4084C oscilloscopes as well as our text-to-speech solution for the PIC24 for some examples of what we can do.

Contact us if you have interesting ideas to share and our friendly staff will get back to you!

Terms of use

Search

Recent Comments

Emery on Emulating dual Hercules+CGA monitors setup with DOSBox

D Knight on Opening Windows Write (.WRI) files on modern versions of Windows with CWordpad

ToughDev on Portable 500m AM transmitter from AliExpress

ToughDev on Repairing a failed Tecsun S-8800 radio due to damaged charging board

Ayman Ali on Repairing a failed Tecsun S-8800 radio due to damaged charging board

Peter on Portable 500m AM transmitter from AliExpress

Greg on System 7.5.5 on Mini vMac

will be no video output once the emulator is run). Finally, although Open Sound System (OSS) has been deprecated in Ubuntu 16, we also enable it with **–enable-sound-oss** just in case.

When configuration is done, check the output and make sure that the requested modules (PPP, SDL, etc) have been enabled:

```
pce 20170208-df19414 is now configured:
CC:                        gcc -g -O2
LD:                        gcc

prefix:                    /usr/local

Emulators built:           atarist cpm80 ibmpc macplus rc759 sim405 sim6502
simarm sims32
Emulators not built:
ROMs built:
ROMs not built:            ibmpc macplus
Terminals built:           null x11 sdl
Terminals not built:
Char drivers built:        null stdio posix ppp pty slip tcp termios
Char drivers not built:    wincom
Sound drivers built:       null wav oss sdl
Sound drivers not built:
Enabled options:           tun
Disabled options:          readline
```

If the results show that SDL or X11 is not built, check that you have the necessary development libraries installed.

The final step is to install the emulator to /usr/local/bin/pce-macplus. This will also install other emulators (atarist, cpm80, ibmpc. etc) that were built during the configuration process as well:

```
make
sudo make install
```

When everything is done, type **pce-macplus** into the command line and you should see the emulator terminal interface:

```
macdev@macdev-VirtualBox:~/Documents/pce-20170208-df19414$ pce-macplus
pce-macplus version 20170208-df19414
Copyright (C) 2007-2012 Hampa Hug <hampa@hampa.ch>
SYSTEM:   model=mac-plus
*** RAM not found at 000000
CPU:      model=68000 speed=0
VIA:      addr=0xefe000 size=0x2000
SCC:      addr=0x800000 size=0x400000
RTC:      file=pram.dat realtime=1 start=<now> romdisk=0
KEYBOARD: model=1 international=0 keypad=keypad
IWM:      addr=0xd00000
TERM:     driver=null ESC=ESC aspect=4/3 min_size=512*384 scale=1
mouse=[1/1 1/1]
type 'h' for help
```

You can type **q** to quit now as a few other steps are still needed before you can run your favorite Mac system software using this emulator.

Configuring the emulators

Unlike Mini vMac which has a user interface for user to load the disk images and configure various other options such as speed or display, PCE/macplus uses a text configuration file. As a start, download the Windows version of PCE/macplus for your favorite Macintosh machine and extract it. Then create a file call **run.sh** with the following command, place it into the same folder and execute it:

```
pce-macplus -v -c mac-se.cfg -l pce.log -r
```

Here **mac-se.cfg** is the name of the config file to be loaded (-c) and run in the emulator (-r) with results written to the log file (-l) in verbose mode (-v). You will then see two windows – one is the emulated Macintosh machine while the other shows various log messages:

ToughDev on System 7.5.5 on Mini vMac

Greg on System 7.5.5 on Mini vMac

ToughDev on Dial-up Internet access in 2024 using the Viking DLE-200B telephone line simulator

Alexander Jacocks on Dial-up Internet access in 2024 using the Viking DLE-200B telephone line simulator

Yannick on Getting xCode to run on unsupported versions of macOS

ToughDev on Getting xCode to run on unsupported versions of macOS

Yannick on Getting xCode to run on unsupported versions of macOS

ToughDev on Emulating dual Hercules+CGA monitors setup with DOSBox

Bill on Emulating dual Hercules+CGA monitors setup with DOSBox

jalal on Open 16-bit DOS applications natively on 64-bit Windows using NTVDMx64

ToughDev on System 7.5.5 on Mini vMac

Andrea on System 7.5.5 on Mini vMac

ToughDev on Troubleshooting hard-to-find Windows Subsystem for Linux (WSL) crash due to issues with NTVDMx64

Follow @mrtoughdev

Tags

3cx 68k 8086tiny 80386 adlib amiga c++ call control api cctv ch375 ch376 commodore data recovery deskmate dosbox dspic embedded ffmpeg intel atom iOS macintosh macintosh se mac os microsoft office mini pc msdos ms dos pcjs radio raspberry pi retro68 rtsp scsi2sd tandy tecsun turbo c ubuntu virtualbox voip windows windows 11 windows 98 WordPress word processor xCode

Archives

2024 (10)

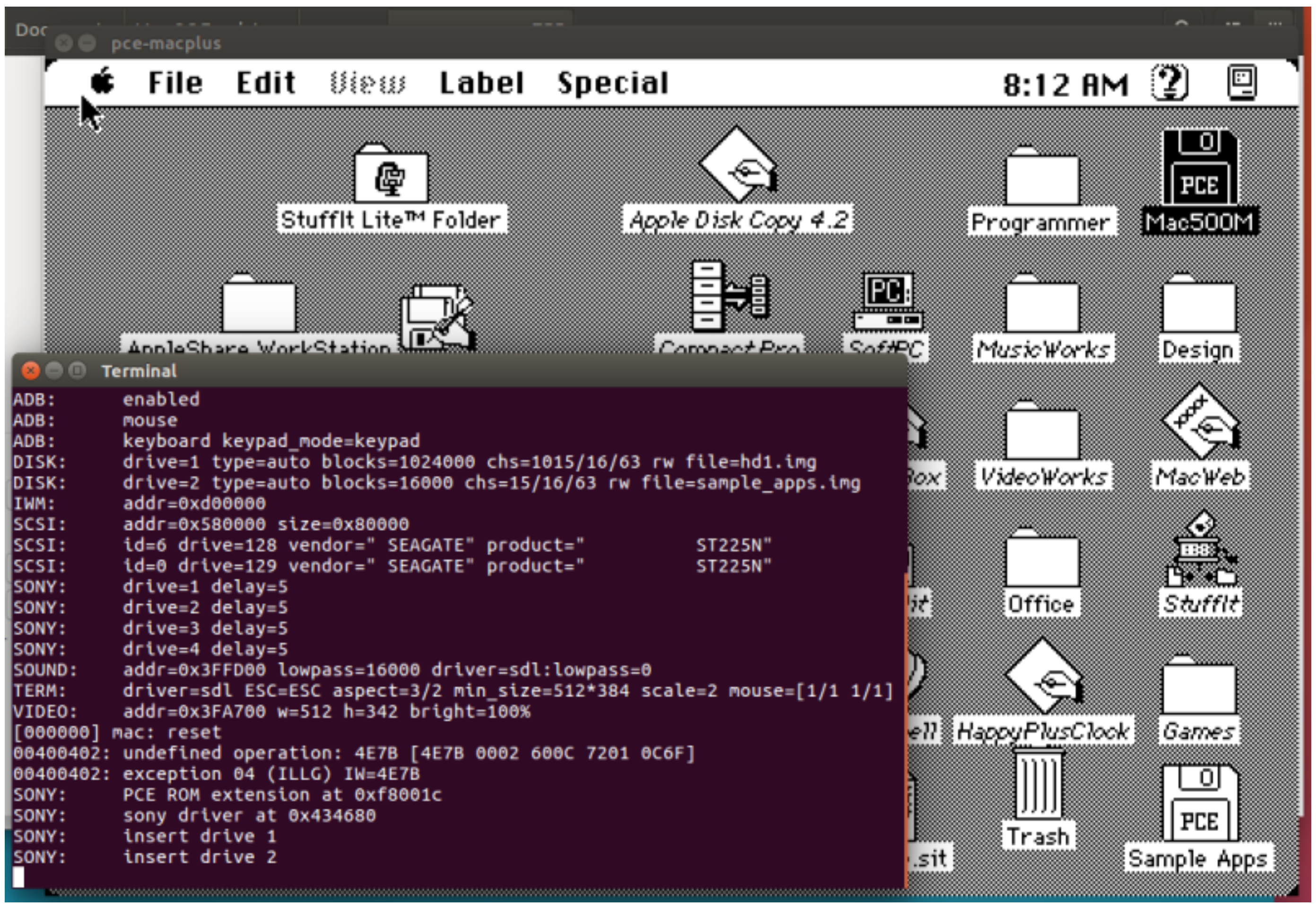
2023 (14)

2022 (12)

2021 (12)

2020 (12)

2019 (12)



```
ADB:      enabled
ADB:      mouse
ADB:      keyboard keypad_mode=keypad
DISK:     drive=1 type=auto blocks=1024000 chs=1015/16/63 rw file=hd1.img
DISK:     drive=2 type=auto blocks=16000 chs=15/16/63 rw file=sample_apps.img
IWM:      addr=0xd00000
SCSI:     addr=0x580000 size=0x80000
SCSI:     id=6 drive=128 vendor=" SEAGATE" product="          ST225N"
SCSI:     id=0 drive=129 vendor=" SEAGATE" product="          ST225N"
SONY:     drive=1 delay=5
SONY:     drive=2 delay=5
SONY:     drive=3 delay=5
SONY:     drive=4 delay=5
SOUND:    addr=0x3FFD00 lowpass=16000 driver=sdl:lowpass=0
TERM:     driver=sdl ESC=ESC aspect=3/2 min_size=512*384 scale=2 mouse=[1/1 1/1]
VIDEO:    addr=0x3FA700 w=512 h=342 bright=100%
[000000] mac: reset
00400402: undefined operation: 4E7B [4E7B 0002 600C 7201 0C6F]
00400402: exception 04 (ILLG) IW=4E7B
SONY:     PCE ROM extension at 0xf8001c
SONY:     sony driver at 0x434680
SONY:     insert drive 1
SONY:     insert drive 2
```

For more information on the available options, refer to this commented example configuration file.

Sound support

Various sound output methods are supported: null (no sound), OSS (using Open Sound System to output sound to DSP device) and SDL (Simple DirectMedia Layer). Recording the sound output to a WAV file is also supported. You can configure the output method by specifying the **driver** string in the **sound** section of the config file:

```
sound {
......
#driver = "null"
#driver = "wav:wav=speaker.wav:lowpass=0:wavfilter=0"
#driver = "oss:dev=/dev/dsp:wav=speaker.wav:lowpass=0:wavfilter=0"
driver = "sdl:wav=speaker.wav:lowpass=0:wavfilter=0"
}
```

In my experiments, SDL works best and is the most reliable method. OSS does not work well on Ubuntu 16, since it has been deprecated. It either produces no sound at all or plays garbled sounds over the speaker, probably because the frequency and bit rate of /dev/dsp is not configured properly. However, regardless of the method used, the Macintosh 128K emulated by PCE/macplus does not play any chime upon booting up – only some static noises are heard.

Hard drive emulation

I came across an interesting issue when playing around with PCE/macplus. I used Linux **dd** command to create a 1000MB hard disk image, attach it to the machine using the config file, and use Apple HD SC Setup to initialize the drive and create the partitions. Surprisingly, despite changing various options, the tool only created a 20MB partition while leaving the rest of the emulated hard drive unallocated. I have been using this tool on several large hard drives up to 2GB, both emulated and real, with no issues. So what was wrong this time?

After much debugging, the issue is due to PCE/macplus faithful attempt to emulate the original Apple hard disk drive, down to specifying SEAGATE and ST225N as the vendor and product strings in the config file respectively:

```
device {
   id      = 0
   drive   = 0x81

   vendor  = " SEAGATE"
   product = "         ST225N"
}
```

This config emulates the Seagate ST225N 20MB hard disk drive that comes with many compact Macintoshes of the time. Although this is good in a sense because the emulated hard drive will show up in Apple HD SC Setup immediately without the need for a patch, it also causes Apple HD SC Setup to think that the hard drive is only 20MB and creates a single 20MB partition, without

2018 (13)
2017 (12)
2016 (13)
2015 (29)
2014 (17)
2013 (16)
2012 (13)
2011 (16)
2010 (30)
2009 (53)
2008 (53)

Featured

Mobile Development
VoIP
Macintosh 68k
Logic Analyzer
Microcontroller
Oscilloscope
Radio/TV
Reverse-engineer

Tools

Log in
Entries RSS
Comments RSS
WordPress.org

even querying the drive for its real size. The solution is to change the vendor and product name to something else (while maintaining the original string length), and use the patched version of Apple HD SC setup to initialize the drive.

Disk image format

For supported Macintosh models, adding a new hard disk image is as simple as adding a **disk** section in the config file:

```
disk {
    drive    = 2
    type     = "auto"
    file     = "sample_apps.img"
    optional = 0
}
```

In my tests, both HFS partition images and HFS entire hard drive images are supported. An **insert_delay** option can be used in the **sony** section of the config file to indicate how long the emulator should wait after startup before inserting the disk image. This is needed if you are having multiple disk images, as inserting them all at once will confuse the Mac and may cause some of them not to show up, which happened several times during my experiment.

Unfortunately there seems to be no easy mechanism to insert or remove disk images when the emulator is running, unlike Mini vMac. The only method is to press **Ctrl-`** in the video window to pause the emulator and type **m emu.disk.insert <drive>:<fname>** or **m emu.disk.eject <drive>** into PCE/macplus terminal window. Not only is this method so user-unfriendly and counter-intuitive, it has never worked properly when I tried it. Either the disk could not be ejected, or does not show up without a reboot after being inserted. If you need to install software that requires the insertion/removal of multiple disks, just mount them all at once in the config file before starting up the emulator – most installers will know how to eject the last disk and continue with the next disk automatically if multiple disks are present.

Another problem that I have is with the disk image file extension. In a few cases, the emulator seems to guess the image type using the file extension. For example, the same disk image would work when its file extension is ".img", but will no longer be recognized if ".dsk" or ".image" is used. There is also no clear documentation on what type of images are supported, and what values other than "auto" are accepted as the disk type.

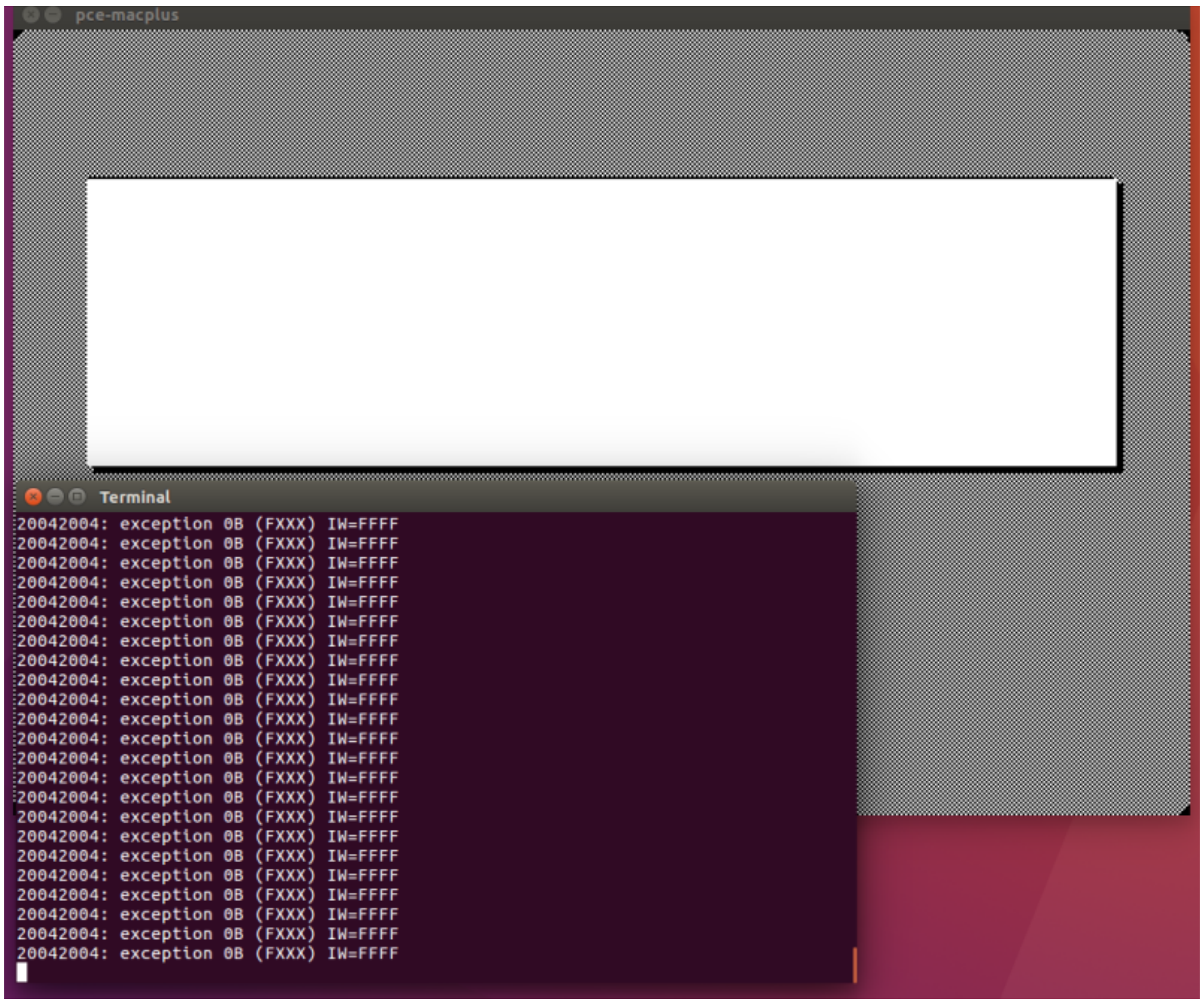
Macintosh 512K and HD20 INIT

When playing around with the Macintosh 512K emulator and trying to mount HFS disk images with it, I noticed something weird. As we know, the Macintosh 512K (and 128K) came with only 400K IWM disk drives and only support the original Macintosh File System (MFS). Although external 800K floppy drives may work, the internal floppy drive will still not read 800K disks even if an internal 800K drive is fitted, unless the ROM is also upgraded. To cater for this, the PCE/macplus config file has an option named **cfg.sony** which will enable support for 800K Sony disk drives if set to 1:

```
cfg.sony = 1
rom {
    file    = "mac-plus.rom"
    size    = 128K
    address = 0x400000
}
if (cfg.sony) {
    rom {
        file    = "macplus-pcex.rom"
        address = 0xf80000
        size    = 256K
    }
}
```

If **cfg.sony** is 1, then an extension ROM called **macplus-pcex.rom** is loaded, enabling support for 800K floppy drives. Otherwise, only 400K IWM floppy images are supported. Other than playing around with the sample MacWrite and MacPaint IWM images that come with PCE/macplus, I have no ideas how to create new IWM images.

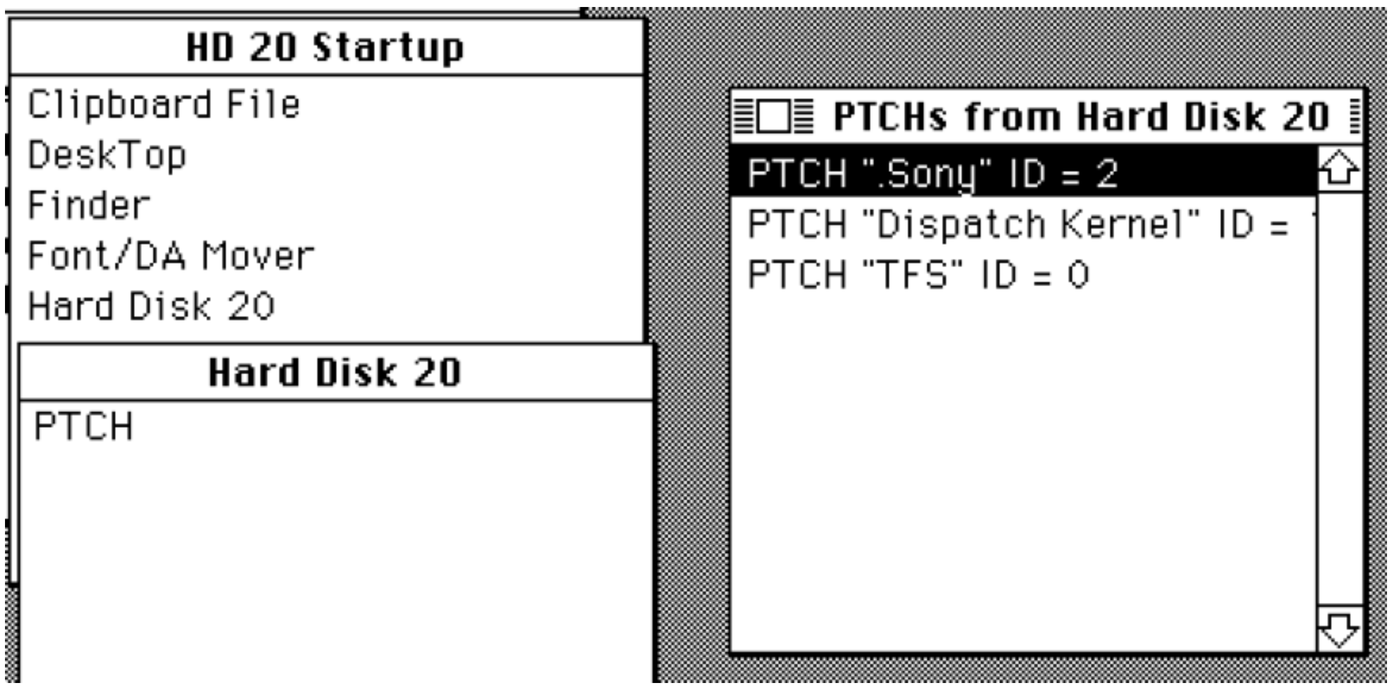
With the Sony drive support enabled, the next step is to enable support for HFS for the Macintosh 512K. This is usually done by adding the HD20 INIT to the System folder and should work out of the box. However, in my case, I was greeted with a flashing "Welcome to Macintosh" screen and tons of exception messages in the terminal window after a reboot:



```
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
20042004: exception 0B (FXXX) IW=FFFF
```

The terminal window displayed "exception 0B (FXXX) IW=FFFF" continuously and I could only close the emulator. What was wrong?

The root cause is because the HD20 INIT also contains Sony driver support, which conflicts with the macplus-pcex ROM file. To fix this, use ResEdit to open the Hard Disk 20 INIT file, select the PTCH resource, select the ".Sony" entry and choose Edit > Clear to remove the Sony driver support:



After a reboot, the Macintosh 512K emulated by PCE/macplus should now have HFS support and emulated HFS disk images should show properly in Finder.

The ZIP file containing the original HD20 Startup disk image as well as the modified disk image with the Sony driver removed can be downloaded here.

PPP emulation through serial port

One of the reasons why I like PCE/macplus is its abilities to support PPP (and SLIP, as well as TCP) on the serial port, allowing Internet access from System 6 and 7 machines. This cannot be done with Mini vMac, since it does not emulate the serial port. To test this, I am using Config PPP 2.0.1 and MacPPP 2.0.6. MacTCP should be set to use PPP on a static IP configuration and not LocalTalk. Config PPP should point to the modem port, although using the printer port is also possible, just that you may need to disable AppleTalk on that port using Chooser. The rest of the settings can be kept as default.

Next, you need to create a TUN interface on the machine which will be used for PPP. The following script, when run as root, will create a TUN interface named 'tun9':

```
tunctl -t tun9
ifconfig tun9 192.168.1.244 up
route add -host 192.168.1.245 dev tun9
bash -c 'echo 1 &gt; /proc/sys/net/ipv4/conf/tun9/proxy_arp'
arp -Ds 192.168.0.253 enp1s0f0 pub
```

Replace **enp1s0f0** with your Ethernet (or wireless) interface name and **192.168.0.253** with its IP address. Now run **ifconfig**, and you should see the tun9 interface showing up:

```
tun9 Link encap:UNSPEC HWaddr
00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:192.168.1.244 P-t-P:192.168.1.244 Mask:255.255.255.255
UP POINTOPOINT NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Modify the config file to enable PPP on the modem port:

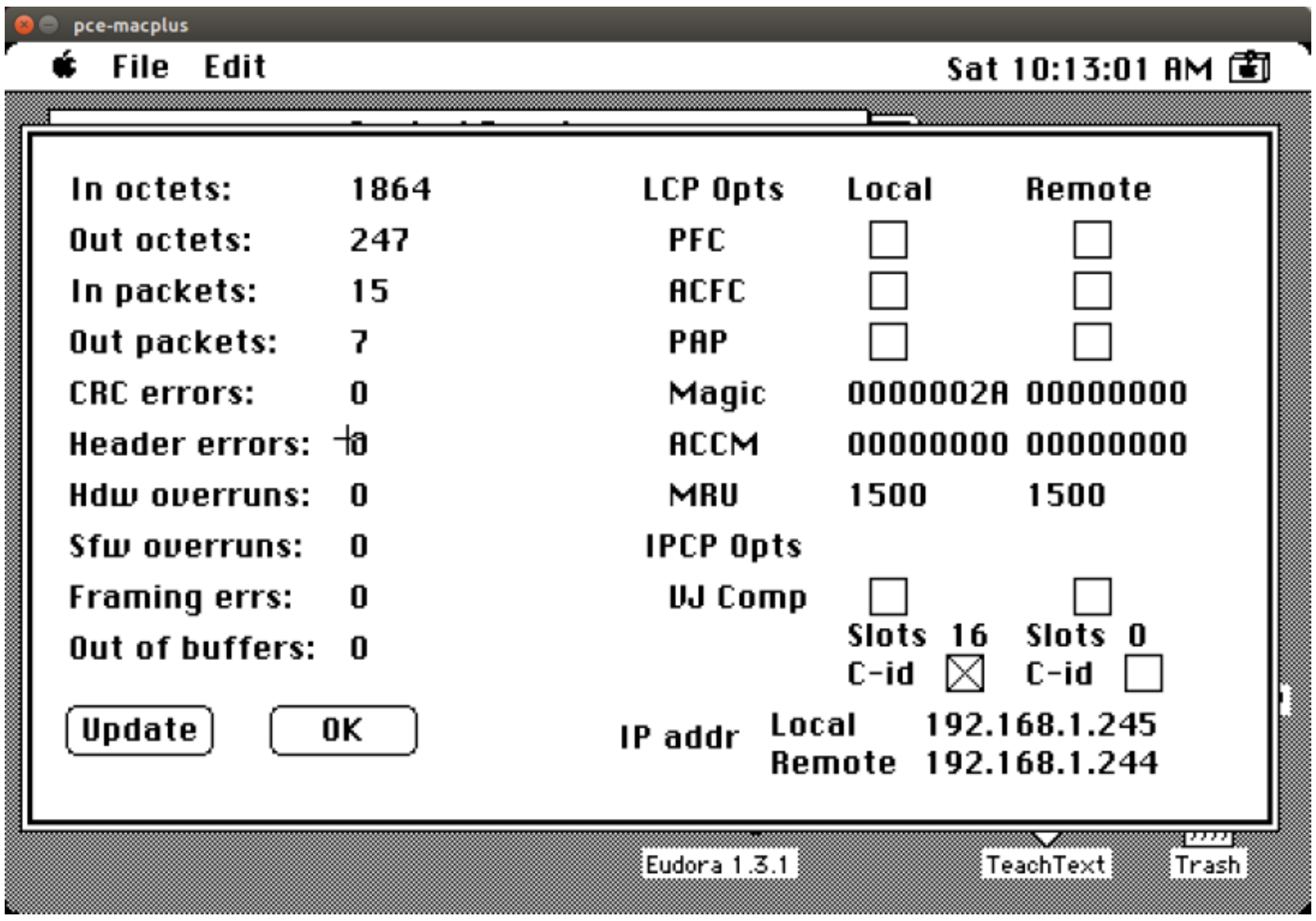
```
serial {
port = 0
multichar = 1
driver = "ppp:if=tun9:host-ip=192.168.1.244:guest-ip=192.168.1.245"
}
```

The multichar parameter, if set, will allow the specified number of characters to be sent or received without any transmission delay. This means that if multichar is set to 2 and the serial port is configured to be 19200bps, you can potentially have a serial transmission speed up to 2x19200bps = 38400bps. For a start, set it to 1 to use the default serial port speed. The guest IP address provided on the driver string should match the MacTCP static IP configuration.

Now start PCE/macplus, and check the terminal to make sure that the PPP interface has been set up with no errors:

```
SERIAL: port=0 multichar=1 driver=ppp:if=tun9:host-ip=192.168.1.244:guest-
ip=192.168.1.245
```

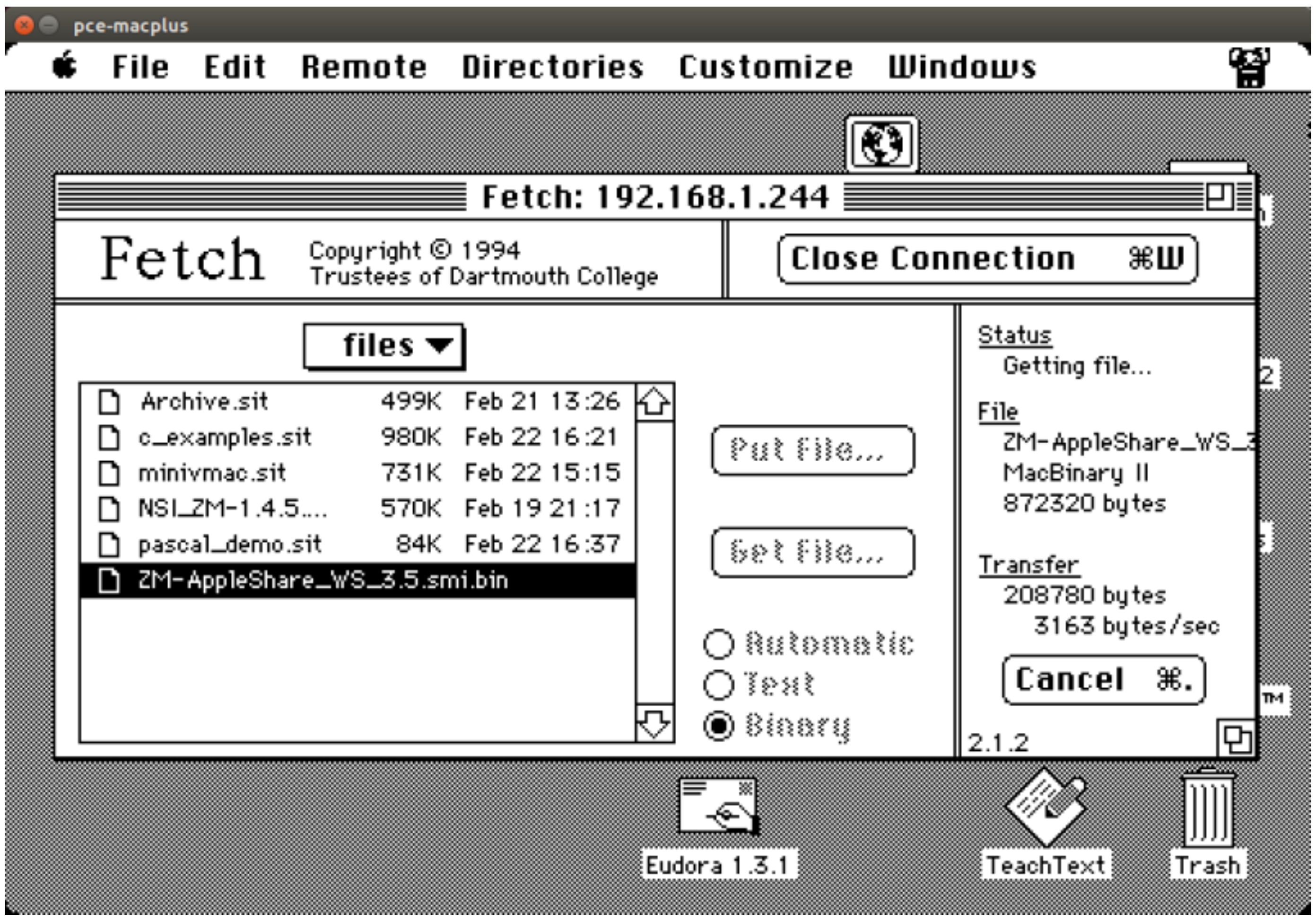
Once the system has booted up, click **Open** inside **Config PPP.** If the connection is successful, **PPP UP** will be shown and clicking the **Stats** button will show the transmission statistics:



```
In octets:       1864        LCP Opts   Local      Remote
Out octets:      247           PFC        ☐          ☐
In packets:      15            ACFC       ☐          ☐
Out packets:     7             PAP        ☐          ☐
CRC errors:      0             Magic      0000002A   00000000
Header errors:   0             ACCM       00000000   00000000
Hdw overruns:    0             MRU        1500       1500
Sfw overruns:    0           IPCP Opts
Framing errs:    0             VJ Comp    ☐          ☐
Out of buffers:  0                        Slots 16   Slots 0
                                          C-id ☒     C-id ☐

                             IP addr  Local   192.168.1.245
                                      Remote  192.168.1.244
```

With the PPP connection in place, Fetch now works properly to download files from an FTP server at 3KB/sec:



```
Archive.sit             499K  Feb 21 13:26
c_examples.sit          980K  Feb 22 16:21
minivmac.sit            731K  Feb 22 15:15
NSI_ZM-1.4.5....        570K  Feb 19 21:17
pascal_demo.sit          84K  Feb 22 16:37
ZM-AppleShare_WS_3.5.smi.bin
```

Status
Getting file...
File
ZM-AppleShare_WS_3
MacBinary II
872320 bytes
Transfer
208780 bytes
3163 bytes/sec

You will notice that the download speed will be slightly faster than the maximum theoretical speed for the serial port. In this example, the serial port is configured at 19200bps, yet we are able to download at 3163 bytes/sec (almost 28.8kbps, taking into account protocol overhead). This is due to the efficiency of the emulated serial port. On a real Macintosh SE with the same configuration, I can only get around 1.8-1.9KB/s download speed. You can make the download speed even faster by setting the **multichar** parameter to a reasonable value such as 5 or 10.

If the emulator says "can't open driver" on the TUN interface, you can try to use openvpn instead of tunctl to create the TUN interface. Try the following:

```
sudo openvpn --mktun --dev tun9
```

If you can open the PPP session but there seems to be no Internet access, you may need to enable IP forwarding on your Linux host by editing **/etc/sysctl.conf** and setting **net.ipv4.ip_forward=1**, then run **sudo sysctl -p** to update the system settings. If you still have issues, try to enable IP masquerade on your main network interface for traffic from the TUN interface.

Internet access via PPPD running on host machine

If the PPP emulation by PCE/macplus does not work at all, you can use tty0tt0 to create a pair of null modem ports:

```
$ ./start_tty0tty.sh
(/dev/pts/13) <=> (/dev/pts/18)
```

and configure the emulator to use one port of the pair (/dev/pts/13):

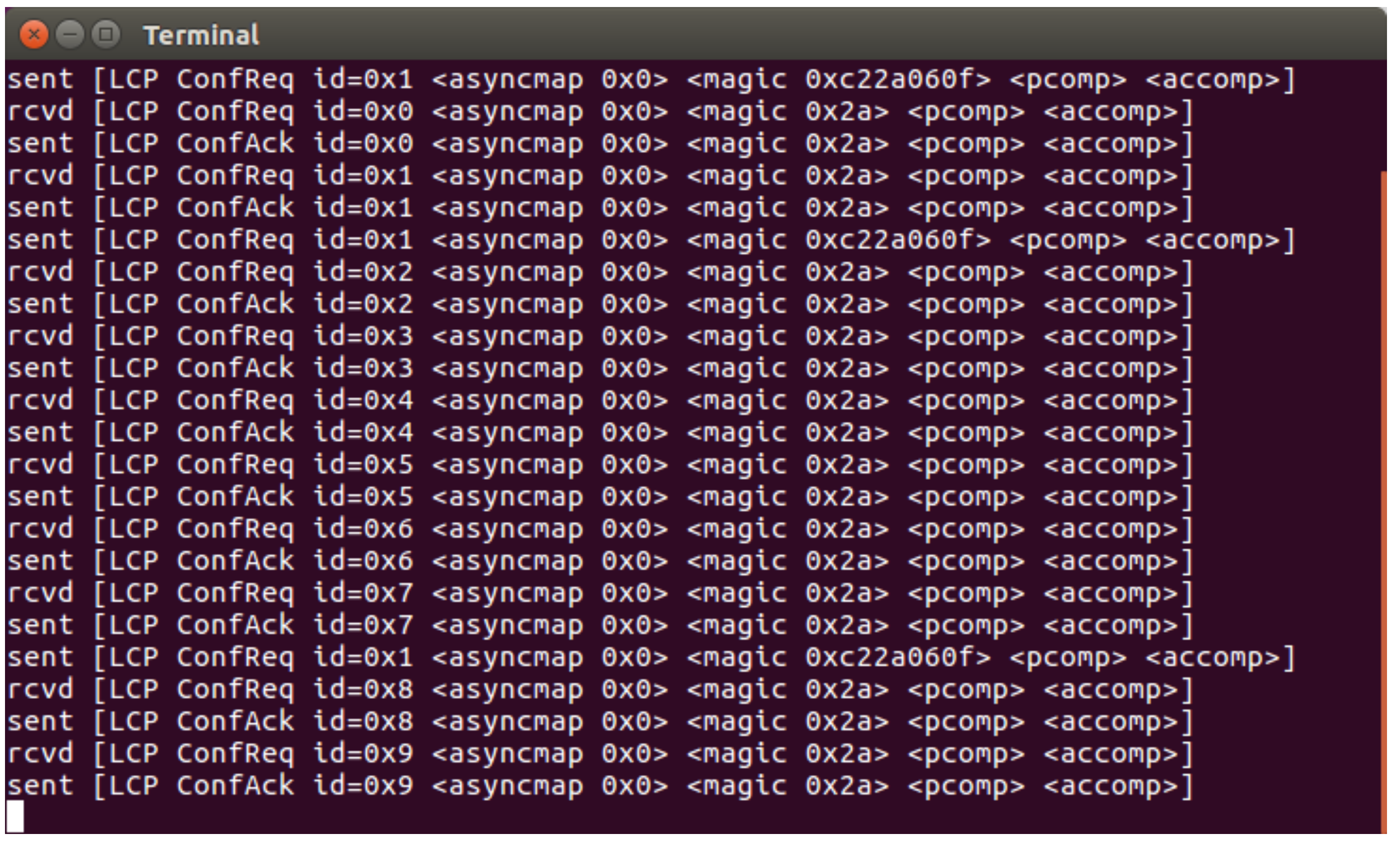
```
driver = "stdio:file=/dev/pts/13"
```

Use the following to run PPPD on the host machine to point to the other port (/dev/pts/18). You may need to change the IP to match your configuration :

```
pppd debug nodetach defaultroute proxyarp local nocrtscts
192.168.1.244:192.168.1.245 /dev/pts/18 19200
```

Remember to edit **/etc/ppp/options** and add (or uncomment) the **noauth** option to disable PPP authentication, which will just cause problems. You can also add the **passive** and **persist** options so that PPPD will not simply give up after a few connection attempts. After that, click **Open** in the **Config PPP** panel to connect to a PPP server, and quickly run the above PPPD startup script. The connection should be established and Internet access should be available from inside the emulator.

One thing to take note is that the PPPD request should be triggered from the emulator first (by clicking on the Open button from Config PPP), followed by quickly starting PPPD on the host machine. If the order is reversed (PPPD is started first), the emulated Macintosh will attempt to send out data packets, which are received by PPPD, but will not receive any response, and the connection will eventually timeout, with a "Link Dead" error. The issue is demonstrated in the following screenshot, with multiple LCP ConfReq and LCP ConfAck messages being generated:



```
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xc22a060f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xc22a060f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x2 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x2 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x3 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x3 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x4 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x4 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x5 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x5 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x6 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x6 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x7 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x7 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xc22a060f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x8 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x8 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x9 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x9 <asyncmap 0x0> <magic 0x2a> <pcomp> <accomp>]
```

I contacted the author of PCE/macplus and was advised that this may be due to assumptions made on the serial port behaviour, which may not apply for emulated serial ports. For example, a tool could assume that any data sent via the serial port is immediately received on the other end, which may not be true for an emulated port (due to data buffering, etc.), causing weird timing issues. Whatever the reason might be, I was not able to fix the issue. So for now, if you have to use PPPD on the host, just remember to initiate the connection from inside your emulator first.

Macintosh Classic ROM disk

The Macintosh Classic has a hidden System 6.0.3 disk image, which can be started by holding down Command-Option-X-O at startup. PCE/macplus emulates this as well, by having an **rtc.romdisk** option in the config file. If set to 1, the machine will boot from the ROM disk:

```
rtc {
# On startup the parameter RAM is loaded from this file. On
# shutdown it is written back.
file = "mac-classic-pram.dat"

realtime = 1

# Set the startup disk to the ROM disk. This only works with
# the Macintosh Classic ROM.
romdisk = 0
}
```

Booting from ROM disk can also be specified from the command line using **rtc.romdisk=1**:

```
pce-macplus -v -c mac-classic.cfg -l pce.log -I rtc.romdisk=1 -r
```

There is a minor issue with booting from ROM disk. The selected startup disk is stored in PRAM, which is the mac-classic-pram.dat file in the PCE/macplus emulator. Once the machine has booted from floppy/hard disk drive, the startup disk is remembered and used for the next boot, even if booting from ROM disk is specified. According to the author, this is because the code does not know how to set the startup disk. A workaround is to delete the **mac-classic-pram.dat** file for every startup so that the machine could boot from either the ROM disk or from the disk images correctly. This causes several system settings, include screen brightness, to be reset to default, and the screen will look dark upon first startup. In my tests, even after deleting **mac-classic-pram.data**, setting **rtc.romdisk=1** from command line will sometimes still boot from the disk drive if **romdisk = 0** is set in the config file. This seems to be a bug with PCE/macplus.

The verdict

Despite the limitations and some minor issues, i think PCE/macplus is a very capable emulator. I am currently using it together with Retro68 and CodeLite to improve Browsy, an open-source minimal System 6 web browser. I hope the author of PCE/macplus can spend some time making the emulator for user friendly, for example by allowing easy insertion/removal of disk images when the emulator is running. When that happens, I guess PCE/macplus will be much more popular than Mini vMac, currently the most well-known emulator for System 6 and System 7.

See also

Developing 68K Mac apps with CodeLite IDE, Retro68 and pce-macplus emulator
Exploring Retro68 GCC-based SDK for 68K Macintosh computers
System 6.0.8 on a vintage Macintosh SE with 4MB RAM
Exploring old Mac software on Mini vMac, Basilisk and SheepShaver
System 7.5.5 on Mini vMac

Share Tweet Share Share Share Share

5.00 avg. rating (**94%** score) - **1** vote

← Exploring Paralex paraphrase-driven open question answering engine

System 6.0.8 on a vintage Macintosh SE with 4MB RAM →

4 thoughts on “PCE/macplus, the ultimate 68K Macintosh emulator”

Em Adespoton
April 8, 2017 at 12:28 pm
Permalink

Hmm… I’m trying to do this with OS X instead of Linux; OSS gets disabled of course, but I’m having issues with tun, with the error “src/lib/tun.c:36:10: fatal error: ‘linux/if.h’ file not found”

Do you know if the tun source only compatible with linux and not with OS X tun?

ToughDev Post author
April 8, 2017 at 2:00 pm
Permalink

Hi,

This is a common problem when building code that uses TUN/TAP meant for Linux on OSX. Try to use this http://tuntaposx.sourceforge.net and see if it helps you.

Let me know how it goes.

Andrew
October 7, 2018 at 1:10 pm
Permalink

In VirtualBox (Windows host, Ubuntu VM) the mouse would fly to one of the four corners of the emulated screen no matter how I moved the mouse. After hours of trying different versions and settings of pce, the solution was to turn off “Mouse Integration” in the Input menu at the top of the VirtualBox screen.

Carlos De Matta
August 22, 2021 at 11:38 am

[redacted] Permalink

Thanks, I have no idea how to make the output to work, I’m a windows user, new on ubuntu Linux.

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Comment

☐ I'm not a robot reCAPTCHA Privacy - Terms

Post Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.