



NCommander @FOSSfirefighter

Oct 30 · 156 tweets · [FOSSfirefighter/status/1586511195858644994](https://twitter.com/FOSSfirefighter/status/1586511195858644994)

So, uh, I started script writing on a POSIX script, and this sentence just reached paper:

"To sum up Stratus VOS in a sentence, well, it's the operating system equivalent of Basque."

Other things I've written tonight:

[Use EU4 map here]

"Existing experience really doesn't count for much when every basic assumption is wrong."

"Just take this comment on the topic from the OpenSSL project if you don't believe me."

For those who want to know the comment,



I'm also debating if "POSIXLY Correct" or "[#define](#) POSIX" is a better subtitle heading :)

Well, I just cracked two pages of script. Current highlights

"It's somewhat difficult to show a lack of something, but for this adventure of vaporware, I have found it easiest to start at the beginning ..."

Cursed README file discussion the POSIX bits. I *probably* should track down a copy of the Resource Kit book which probably has all the documentation. I have the 95 one in dead tree form, but not any of NT ones ...

Notice that the partition to which you copy the source code must have the NTFS file system. Also, from the directory to which you copied all the source code, you must run the LONGNAME.BAT file before attempting to compile any of the utilities. This batch file renames many of the files to their correct name, which is often longer than the standard 8.3 filename. If you do not run this batch file, the utilities will not compile correctly.

The following utilities may require code changes to make them work on RISC-based computers: AR.EXE, CC.EXE, DEVSRV.EXE, LD.EXE, MAKE.EXE. For example, CC.EXE invokes "CL386" to compile other programs; on RISC-based computers, you must change the code to invoke the proper compiler.

The following copyright notice applies to both the POSIX utilities provided in binary form as well as the source code on the compact disc.

Copyright (c) 1988, 1989, 1990 The Regents of the University of California. All rights reserved. This code is derived from software contributed to Berkeley by Adam de Boor.

Decided to take a look, the IA has the original Resource Kit book for available for borrowing, and I got to tell you, this thing is cursed (I'm talking about it in the supporter lounge on Discord)

PART III Using Windows NT	
Chapter 5 Windows NT File Systems and Advanced Disk Management	157
File System History	158
About Disks and Disk Organization	159
FAT File System	159
Using the FAT File System with Windows NT	161
HPFS	161
Using HPFS with Windows NT	164
NTFS	164
Master File Table	164
NTFS File Attributes	166
NTFS System Files	171

This actually does document the original STREAMS TCP/IP stack, which was not integrated into Winsock in this era.

Interesting they supported routing on that, which wouldn't reappear until 2000 I think?

Breaking the 254-Session Limit	637
Chapter 19 TCP/IP on Windows NT	639
Components of TCP/IP on Windows NT	640
Internet Protocol Suite	640
Transmission Control Protocol and Internet Protocol	641
User Datagram Protocol	642
Address Resolution Protocol and Internet Control Message Protocol	642
IP Addressing	643
Windows NT TCP/IP Architecture	646
Streams Environment and the Registry	647
User-Mode Interfaces to TCP/IP	647
Windows NT TCP/IP Drivers and Services	648
Using and Administering TCP/IP with Windows NT	649
Name Resolution and NetBIOS over TCP/IP (NBT)	649
The LMHOSTS File	651
Network Browsing with Windows NT TCP/IP	658
Using IP Routing Under Windows NT	661
Using a Multihomed Windows NT Workstation as an IP Router	661
Taking Advantage of Multiple Default Gateways	665
Using Windows NT TCP/IP as a Connectivity Protocol	667
Tips for Using TCP/IP	669
Using IPINFO.INF to Prevent Configuration Errors	669
Using Additional Features of the FTP Server Service	670
Using SNMP	673
How Does the SNMP Service Work?	673
LAN Manager MIB II for Windows NT Objects	674

Also an entire section of talking about Ingres, but I'm getting distracted ...

Chapter 22 Client-Server Connectivity on Windows NT	693
Overview	694
SQL Server for Windows NT	694
Data Access Mechanisms	696
Data Stream Protocols	697
Interprocess Communication Mechanisms	698
Network Protocols	698
Net-Library Architecture	699
Win32 DB-Library Architecture	702
Configuration of the Net-Library Architecture	705
Ingres for Windows NT Client-Server Architecture	709
Ingres General Communication Architecture	710
Ingres Open Connectivity Products	711
Ingres/Net	711
Ingres/Star	714
Client-Server Applications Using GCA Architecture	716
Ingres Data Management	718

For its own video ...

Chapter 24 OS/2 Compatibility	731
Running Applications	732
Supported Applications	732
Unsupported Applications	732
Partially Supported Applications	733
APIs	733
Supported APIs	734
Unsupported APIs	734
Partially Supported APIs	734
Implementation of Subsystem	735
Memory Map of an OS/2 Application	735
Architecture Diagram	736
Multitasking	737
User Interface	738
Dynamic Linking	738
Memory Management	738
Interprocess Communication	739
I/O Architecture	740
I/O Privilege Mechanism	741
Filters	741
Device Monitors	741

The POSIX section is about 20 pages, slightly longer than the OS/2. I'm going to read the OS/2 one as well, cause there might be useful insights to be had.

Chapter 25 POSIX Compatibility	747
Definition of POSIX	748
POSIX Conformance	749
Application Compliance to POSIX.1	749
Running Applications	752
File Systems	752
Bypass Traverse Checking	752
Printing	753
Network Access	753
Restrictions on POSIX Applications	753
Implementation of Subsystem	754
Files Used	755
Communicating with Other Subsystems	755
Further Information	756

Cursed command: FORCEDOS.

Forces a OS/2 and DOS family mode binary to run under NTVDM's DOS box instead of the OS/2 subsystem

If you want to run an OS/2 application that is not supported, you have the following choices:

- If this is a bound application (one that can run under both OS/2 and MS-DOS), you can try to run it under the MS-DOS subsystem. To do so, run FORCEDOS from the command line:

```
FORCEDOS [/D directory] filename [parameters]
```

where:

Variable	Description
<i>/D directory</i>	the current directory for the application to use
<i>filename</i>	the application to start
<i>parameters</i>	the parameters to pass to the application

- If this is not a bound application and you have the source code, you can recompile the source without the unsupported APIs (which are specified in the error message that displays when you try to run the application). If you don't have the source, contact the application's developer.

The OS/2 text mode support is very complete, and this is very well documented. Interesting that LANMAN support is explicitly stated as supported.

That's probably intended to ease migration off OS/2 Server.

Interprocess Communication

The OS/2 subsystem implements all OS/2 IPC mechanisms (semaphores, pipes, shared memory, queues, and signals).

Named Pipes

The OS/2 subsystem implements named pipes on top of the Windows NT named pipe file system. They are supported transparently between Win32, MS-DOS, Win16, and OS/2 applications, both locally and remotely. All of LM 2.x named pipe functionality is supported.

Anonymous Pipes

Anonymous pipes are fully supported, including inheritance. They are integrated into the OS/2 file handle space.

Shared Memory

The full functionality of OS/2 1.x shared memory, including Get and Give semantics, is implemented using Windows NT shared memory features. The discardable segments property is ignored, invisible to the OS/2 application.

Semaphores

The OS/2 subsystem supports the full range of OS/2 1.x semaphore APIs, including RAM semaphores in private and shared memory, system semaphores, and fast-safe RAM semaphores. Association of semaphores with timers and named pipes is fully supported. The OS/2 subsystem uses a combination of the Windows NT semaphore object and the Windows NT event object to implement and OS/2 semaphore.

Queues

OS/2 1.x queues are fully supported, using shared memory between OS/2 processes and OS/2 semaphores as required.

Signals

OS/2 signals are fully supported, using Windows NT APIs to manipulate thread context. The OS/2 subsystem controls the address space of OS/2 processes and uses it to manipulate the register content and the stack of thread 1 of the process to be signaled.

More about LAN Manager support in the OS/2 subsystem. Full remote IPC; you could probably run the 16-bit OS/2 server apps under this, including old MS Mail.

Network Connectivity

The OS/2 subsystem implements some LAN Manager APIs. It also implements NetBIOS (both version 2.x and version 3.0 functionality), named pipes, and mailslots.

The OS/2 subsystem maintains remote drives compatible with OS/2. With these, any OS/2 application can use redirected drives transparently with the file I/O APIs. UNC naming is supported as well. Redirected drives of various network operating systems can be used, provided that the related Win32 Windows NT device drivers (redirectors) are installed.

OS/2 based BBS but its secretly Windows NT :)

Communication with Other Subsystems

Subsystems communicate by passing messages to one another. When an OS/2 application calls an API routine, for example, the OS/2 subsystem receives a message and implements it by calling Windows NT system services or by passing messages to other subsystems. When finished, the OS/2 subsystem sends a message containing the return values back to the application. The message passing and other activities of the subsystem are invisible to the user.

Communication between OS/2 and Windows NT processes can be accomplished through named pipes, mailslots, NetBIOS, files, and COM devices. The Win32 subsystem directs user input to an OS/2 application; it handles all screen I/O for OS/2 applications.

Getting to the POSIX part of the book ...

Definition of POSIX

POSIX stands for *Portable Operating System Interface* for computing environments. POSIX began as an effort by the IEEE community to promote the portability of applications across UNIX environments by developing a clear, consistent and unambiguous set of standards. POSIX is not limited to the UNIX environment, however. It can be implemented on non-UNIX operating systems, as was done with the IEEE Std. 1003.1-1990 (POSIX.1) implementation on VMS, MPE, and CTOS operating systems. POSIX actually consists of a set of standards that range from POSIX.1 to POSIX.12.

As the following table shows, most of these standards are still in the proposed state. This section deals with the Windows NT implementation of a POSIX subsystem to support the international ISO/IEC IS 9945-1:1990 standard (also called POSIX.1). POSIX.1 defines a C-language source-code-level application programming interface (API) to an operating system environment.

Family of POSIX Standards

Standard	ISO Standard	Description
POSIX.0	No	A guide to POSIX Open Systems Environment. This is not a standard in the same sense as POSIX.1 or POSIX.2. It is more of an introduction and overview of the other standards.
POSIX.1	Yes	Systems application programming interface (API) [C language]
POSIX.2	No	Shell and tools (IEEE approved standard)
POSIX.3	No	Testing and verification
POSIX.4	No	Real-time and threads
POSIX.5	Yes	ADA language bindings to POSIX.1
POSIX.6	No	System security
POSIX.7	No	System administration
POSIX.8	No	Networking A. Transparent file access B. Protocol-independent network interface C. Remote Procedure Calls (RPC) D. Open system interconnect protocol-dependent application interfaces
POSIX.9	Yes	FORTLAN language bindings to POSIX.1
POSIX.10	No	Super-computing Application Environment Profile (AEP)
POSIX.11	No	Transaction Processing AEP
POSIX.12	No	Graphical user interface

This is what Microsoft says is supported, there was apparently an official NIST test suite for this. I very much doubt that's survived, but the POSIX subsystem isn't as bare bones as most people think it is ...

POSIX Conformance

For a system to be given a certificate of POSIX.1 conformance it must meet the following requirements:

- The system must support all of the interfaces as defined in the ISO/IEC 9945-1.
- The vendor must supply a POSIX.1 Conformance Document (PCD) with the vendor's implementation as specified in ISP/IEC 9945-1.
- The implementation must pass the appropriate National Institute of Standards and Technology (NIST) test suite.

Windows NT is in the process of being verified for POSIX.1 conformance, and will be submitted to NIST for the Federal Information Processing Standards (FIPS) Publication 151-2 certification. FIPS 151-2 incorporates POSIX.1 as a reference standard and also requires a number of the optional features defined in POSIX.1 to promote application portability among conforming implementations. An implementation that conforms to FIPS 151-2 also conforms to POSIX.1. Note that conformance is specific to the manufacturer, hardware platform, and model number on which the implementation is tested.

POSIX.1 is a source-level standard; it does not provide any binary compatibility.

Application Compliance to POSIX.1

For POSIX.1, there are four categories of compliance, ranging from a very strict compliance to a very loose compliance. The various categories are outlined below.

Strictly Conforming POSIX.1 Applications

A *strictly conforming POSIX.1 application* requires only the facilities described in the POSIX.1 standard and applicable language standards. This type of application accepts the following:

- Any behavior described in ISO/IEC 9945-1 as unspecified or implementation-defined
- Symbolic constants
- Any value in the range permitted in ISO/IEC 9945-1

Officially, 110 calls are supported, with support for libraries, although I don't know if DLLs are supported; it seems to be static linking only.

750 Part VI Migration and Compatibility

This is the strictest level of application conformance, and applications at this level should be able to move across implementations with just a recompilation. At this time, the only language interface that has been standardized for POSIX.1 is the C-language interface. (As shown in the figure below, a strictly conforming POSIX application can use 110 calls from the standard C libraries.)

```
graph TD; A[Strictly conforming POSIX application] --- B[POSIX.1 libraries]; A --- C[Standard C libraries (110 calls)]; B --- D[Operating system]; C --- D;
```

Strictly conforming POSIX application	
POSIX.1 libraries	Standard C libraries (110 calls)
Operating system	

So applications in the POSIX sandbox are and get case sensitivity, and must be run on NTFS partitions.

This isn't actually enforced (I've test it), but I didn't notice it was case sensitive enabled. You also have to reconfigure some ACLs ...

File Systems

POSIX requires a certain amount of functionality from the file system, such as the ability for a file to have more than one name (or *hard links*) and case-sensitive file naming. Neither FAT nor HPFS supports these features, which is another reason why a new file system was required for Windows NT. NTFS supports both hard links and case-sensitive naming. If you want to run in a POSIX-conforming environment, you need at least one NTFS disk partition on your computer.

You can run POSIX applications from any Windows NT file system. If the application does not need to access the file system, the application will run with no problems. However, if the application does require access to the file system, it may not behave correctly on a non-NTFS disk partition.

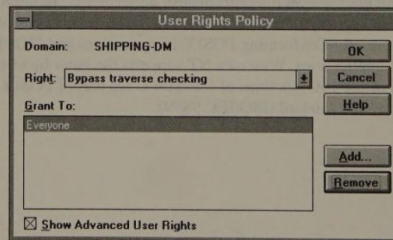
The documentation talks about disabling traverse checking, although I don't think that's a hard and fast requirement ...

Bypass Traverse Checking

By default, when you install Windows NT for the first time, the user right Bypass Traverse Checking is granted to everyone. This right allows a user to change directories through a directory tree even if the user has no permission for those directories.

If you want to run in a POSIX-conforming environment, you must disable this privilege for your account by using either the User Manager or User Manager for Domains tool as follows (you must be an administrator to do this):

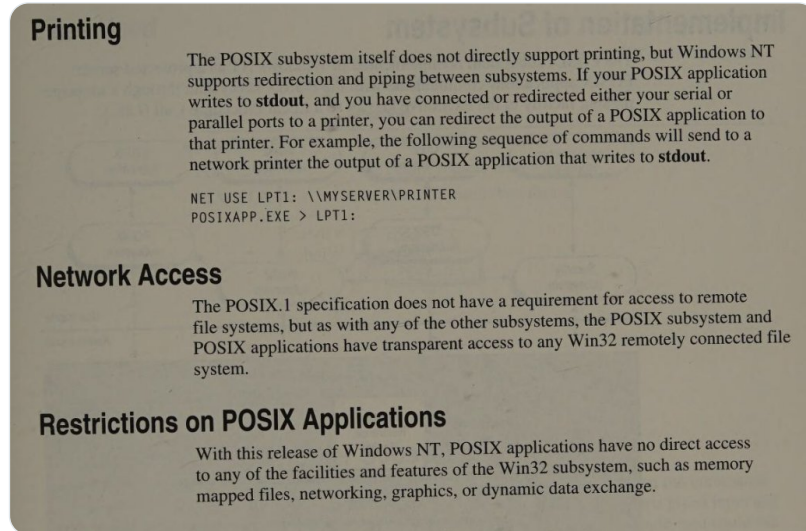
Select the account, and then choose User Rights from the Policies menu to display the following dialog box. (Be sure the Show Advanced User Rights check box is marked.) Specify the Bypass traverse checking right and choose Remove.



And here we get the limitations:

- No network access (defined as NFS)
- No printing
- No access to the Win32 API

Oof ...



and we've reached the end of the documentation ... like beyond this is just some notes.

half-assed doesn't even being to describe this



- I just released the book for anyone else who wants to take a look, I think I need to load up NT 3.1 next ...

Actually, a better question is, did the POSIX subsystem change from NT 3.1 to 2000?

(the last version that included it out of the box)

XP replaced it with Services for UNIX, although it was fundamentally the same thing. I also wonder if you could run POSIX binaries in SFU ...

Well the Windows 2000 Resource Kit book doesn't document anything. Let me grab the actual Resource Kit and see if anything significant changed ...

Well, at first glance, it doesn't, but I cracked open the README and uh ...

cc.exe *was* a wrapper around cl ... This requires more investigation ...

NAME
cc -- GNU project C Compiler

SYNOPSIS
cc [options] file ...

DESCRIPTION
cc is a version of the GNU C compiler. It accepts a dialect of ANSI C with extensions; this dialect is different from the dialect used in 4.3 BSD and earlier distributions. The **-traditional** flag causes the compiler to accept a dialect of extended Classic C, much like the C of these earlier distributions. If you are not already familiar with ANSI C and its new features, you will want to build your software with **-traditional**.

Well, it looks like the "cc" wrapper has a bunch of GNU stuff, but that was there before.

GCC used to have *some* support for the POSIX environment, there's a defined i386|alpha-winnt triplex for it, but I've never heard of someone using it ..

```
gnu_list gnu_lst =
{
  { "p"      , NO , RNULL , SNULL , nat_lst },
  { "traditional" , NO , RNULL , SNULL , NNULL },
  { "go"     , NO , RNULL , SNULL , nat_lst + 1 },
  { "f"      , NO , RNULL , SNULL , NNULL },
  { "t"      , NO , RNULL , SNULL , NNULL },
  { "c"      , NO , RNULL , SNULL , nat_lst + 2 },
  { "o"      , OPEN , RNULL , "<file>" , nat_lst + 21 },
  { "S"      , NO , RNULL , SNULL , nat_lst + 4 },
  { "E"      , NO , RNULL , SNULL , nat_lst + 5 },
  { "v"      , NO , RNULL , SNULL , NNULL },
  { "pipe"   , NO , RNULL , SNULL , NNULL },
  { "B"      , OPEN , RNULL , "<path-prefix>" , NNULL },
  { "b"      , NO , RNULL , SNULL , NNULL },
  { "ansi"   , NO , RNULL , SNULL , nat_lst + 6 },
  { "pedantic" , NO , RNULL , SNULL , NNULL },
  { "0"      , NO , RNULL , SNULL , nat_lst + 7 },
}
```

The NT 4 resource kit book does in-fact still have a section on POSIX compatibility, but I'm not seeing anything different.

Chapter 29 POSIX Compatibility	951
Definition of POSIX	951
POSIX Conformance	952
Application Compliance to POSIX.1	953
Strictly Conforming POSIX.1 Applications	953
Applications Conforming to ISO/IEC and POSIX.1	954
Applications Conforming to POSIX.1 and <National Body>	955
POSIX.1-Conformant Applications That Use Extensions	955
Running Applications	956
File Systems	956
Bypass Traverse Checking	956
Printing	957

this amuses me ...

This chapter describes the Windows NT implementation of a POSIX subsystem. It includes information about the following topics:

- Definition of POSIX
- Conformance and compliance to POSIX.1
- Running applications
- Implementation of subsystem
- Windows NT POSIX files

Note This chapter is not intended to be a POSIX tutorial.

Yeah this is 100% the same, the only difference is the removal of a screenshot, and some slight changes to account for the Explorer based UI.

Printing

The POSIX subsystem itself does not directly support printing, but Windows NT supports redirection and piping between subsystems. If your POSIX application writes to **stdout**, and if you have connected or redirected either your serial or parallel ports to a printer, you can redirect the output of a POSIX application to that printer. For example, the following sequence of commands will send to a network printer the output of a POSIX application that writes to **stdout**:

```
NET USE LPT1: \\MYSERVER\PRINTER
POSIXAPP.EXE > LPT1:
```

Network Access

The POSIX.1 specification does not have a requirement for access to remote file systems, but as with any of the other subsystems, the POSIX subsystem and POSIX applications have transparent access to any Win32 remotely connected file system.

Restrictions on POSIX Applications

With this release of Windows NT, POSIX applications have no direct access to any of the facilities and features of the Win32 subsystem, such as memory mapped files, networking, graphics, or dynamic data exchange.

Going to take a bit of a break, and watch Lashmark's most recent GT:NH video (

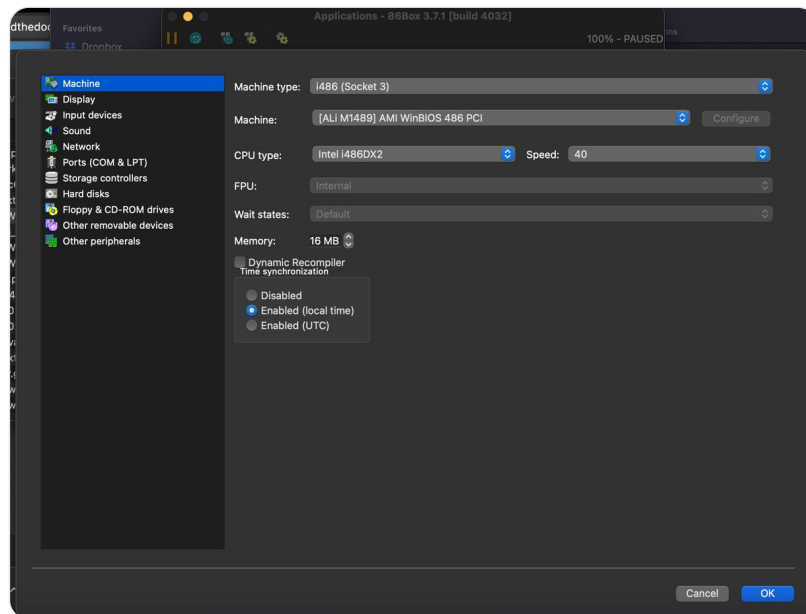


<https://www.youtube.com/embed/oOrhaP1NvIQ>

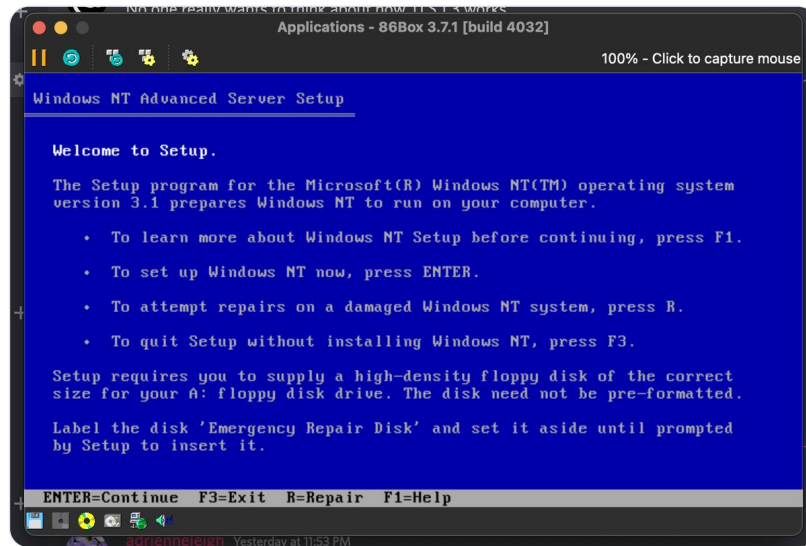
) before digging back in.

I'm writing the script in tandem with the research.

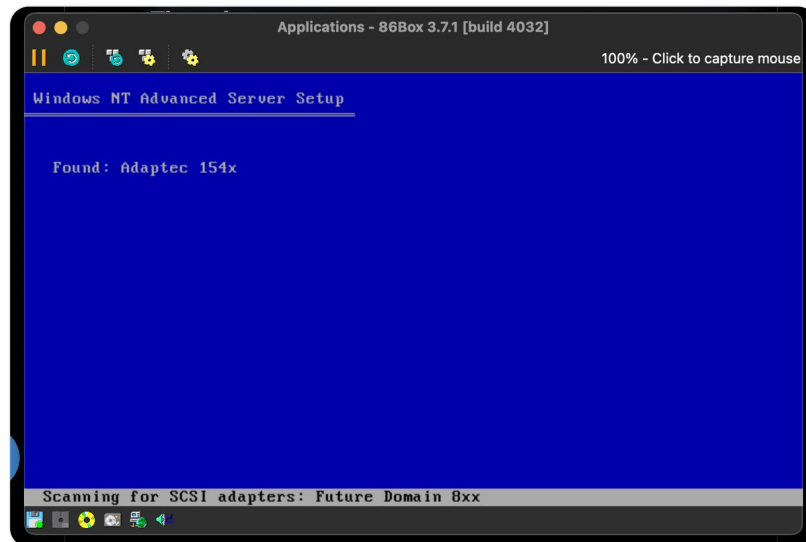
Ok, back to the grind, 86Box has an Apple Silicon port and since I don't really want to get on the desktop, this works. The trick is now divining the correct setup for NT 3.1.



Let's see if we can install; getting NT 3.1 installed is a major crapshoot, and I'm not sure if this media set is good.

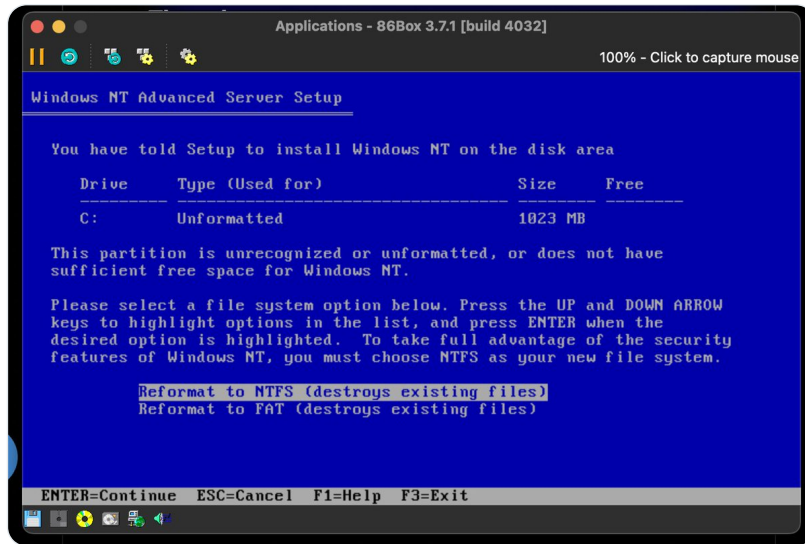


Ok, this is promising ...

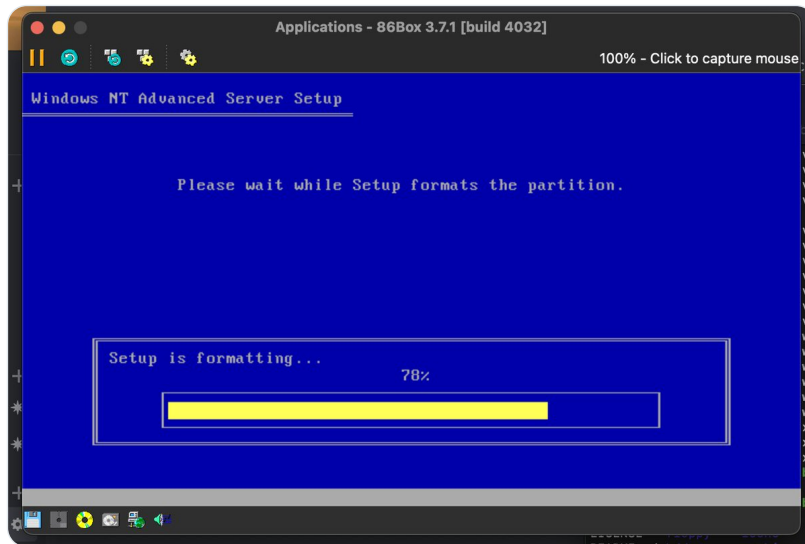


Interesting, on a blank drive, the option to format to HPFS isn't provided. I could install to FAT and convert though. No point though.

HPFS is mostly meant from OS/2 upgrades I think.

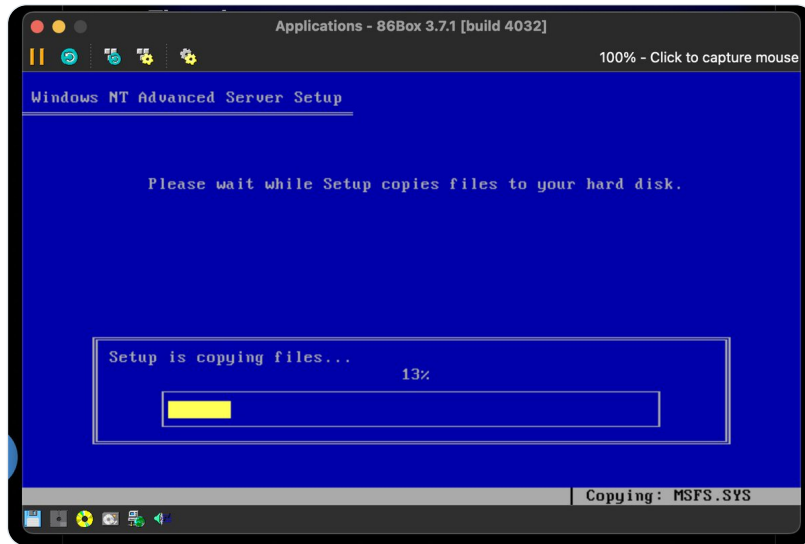


I know using 86box is more accurate, but man, I miss it completing instantly :)

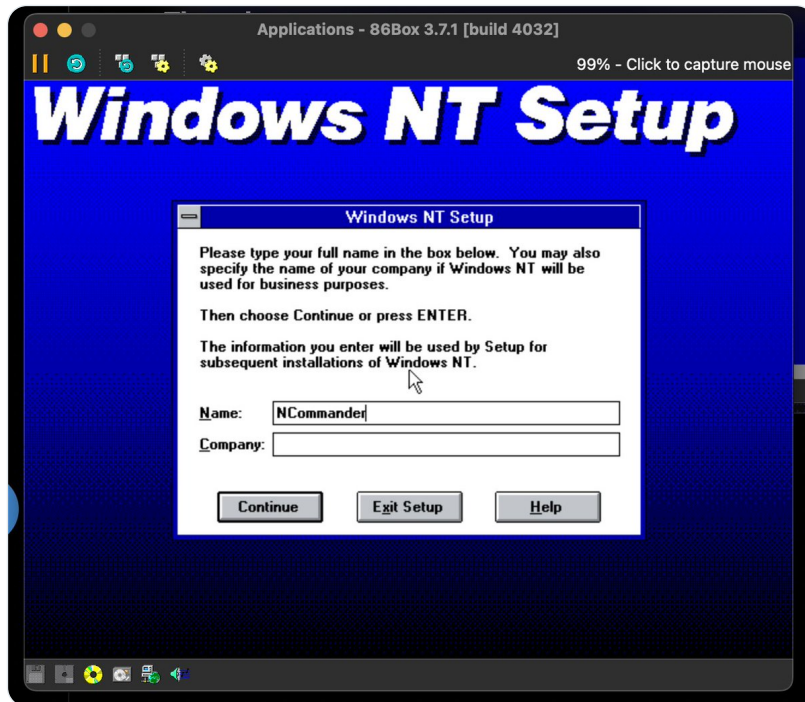


Well *it found the CD* which is the usual failure point.

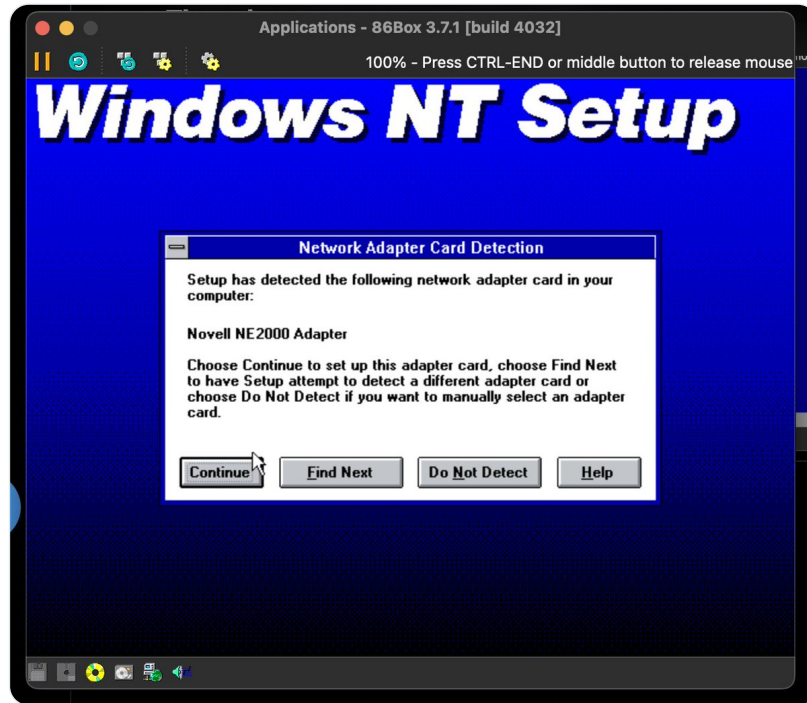
The real question now is if it will blow up in the graphical install, which means this is a bad TXTSETUP copy of the disk.



I have to admit, I really do like the old 3.1 ascetic

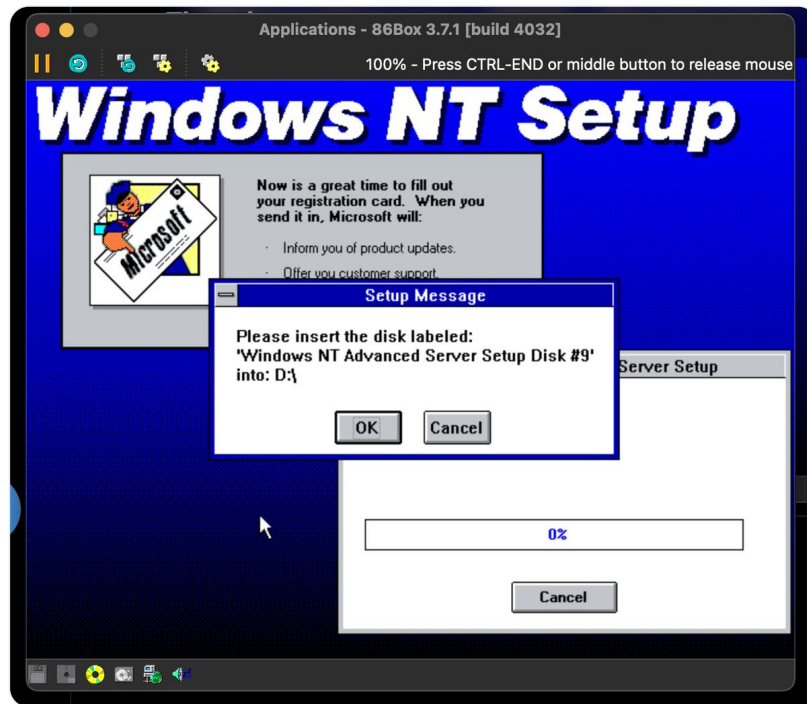


Well, it sees the NE2000 card ... one of the big things is this version of NT doesn't have proper TCP/IP support, so this is very questionable on how useful it really is ...



Yeah that's what I was afraid of, this is the corrupted media set.

OS2 Museum has a write up on this issue. There's a fairly sneaky way to get around it: <https://www.os2museum.com/wp/bad-ntas-iso/>

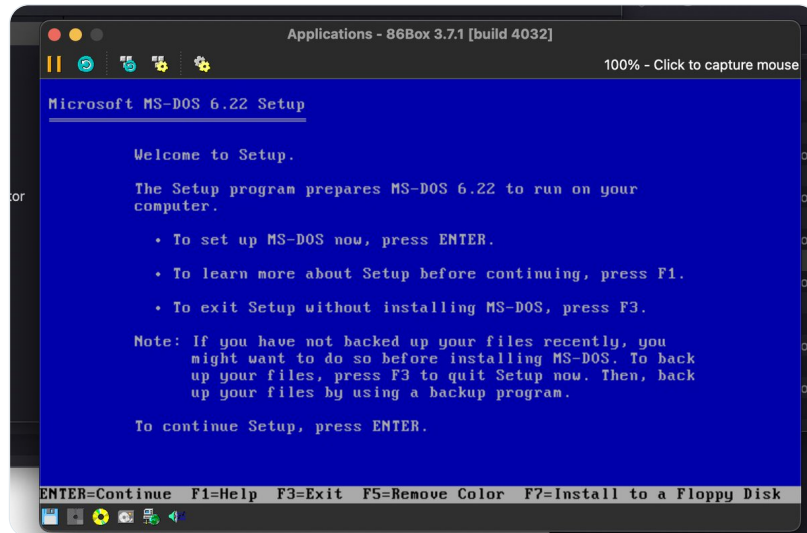


The problem is to install on Pentium+, you need to patch a file on the CD, and then either install from the HDD, or reburn the disc. This works fine for network and HDD installs, but falls over on clean installs.

The trick is to install from DOS, and manually copy files over.

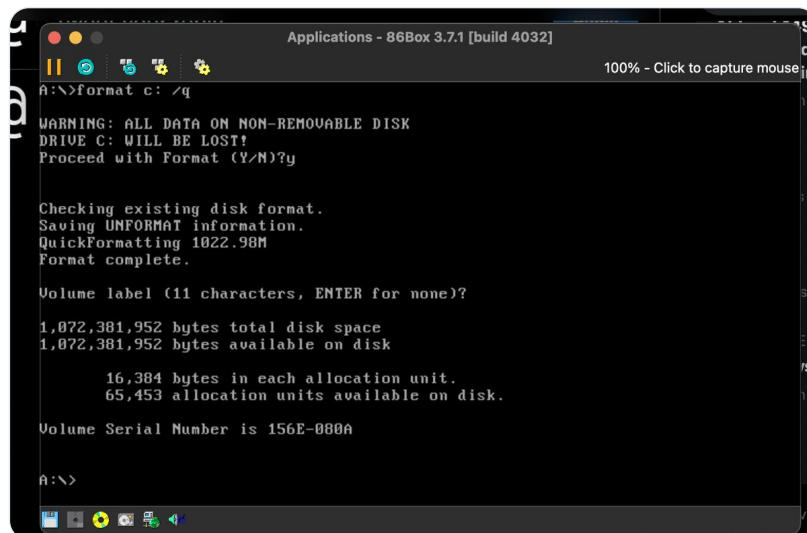
... I'm having trouble getting into BIOS to change the settings cause its a Mac ... stand by :)

Some fiddling later (I just deleted the nvram file)

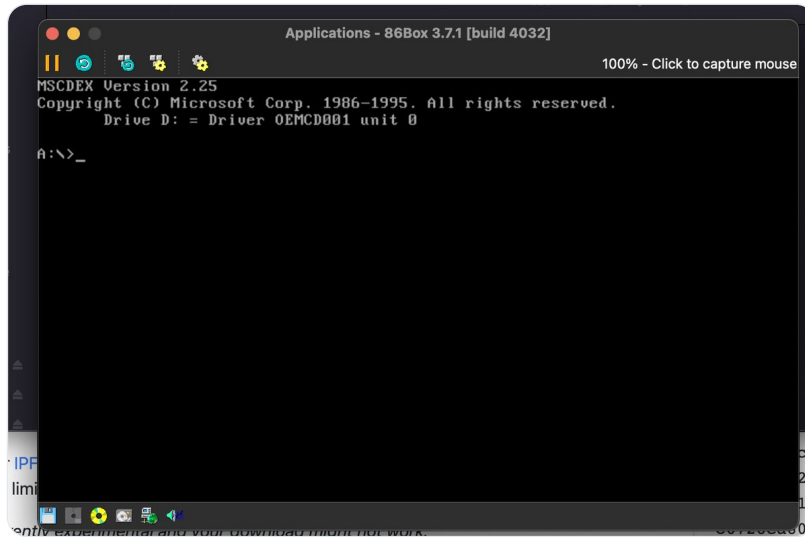


One quick format later.

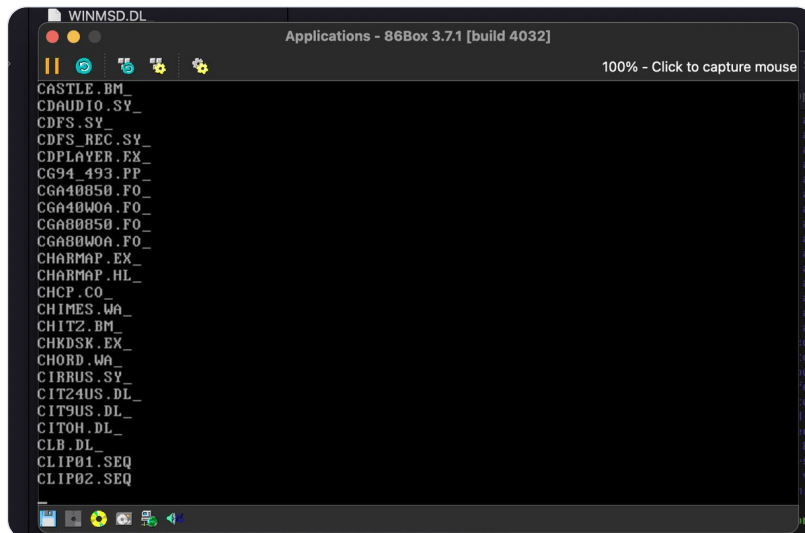
Now I can use the Windows 98 boot disk to copy the CD over (I didn't want to risk getting a FAT32 system)



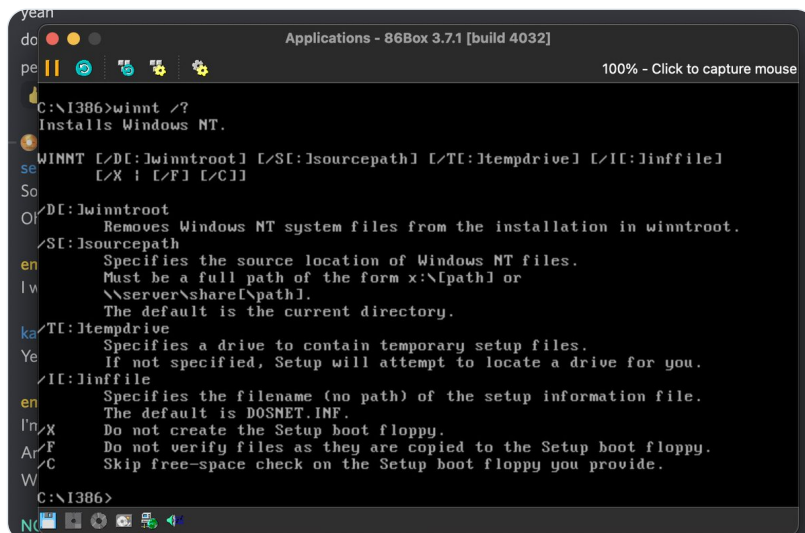
Cool, the SCSI drive showed up on the first try. That makes life much easier ...



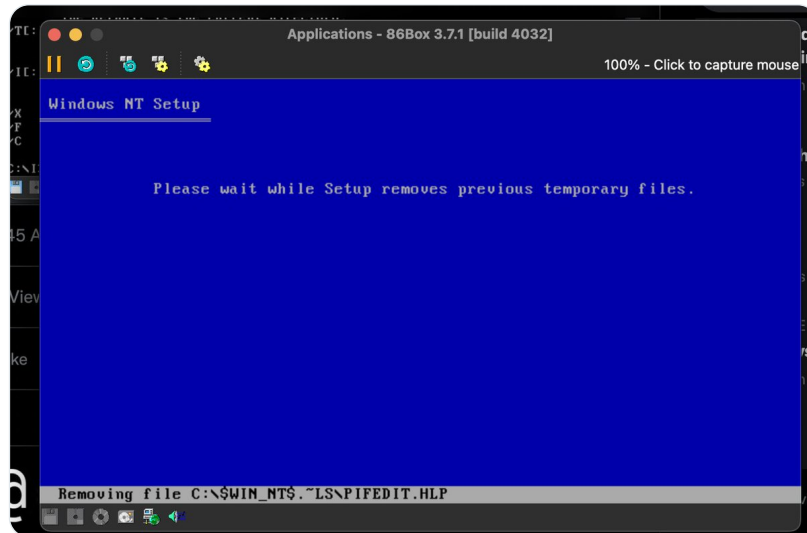
Copying files ...



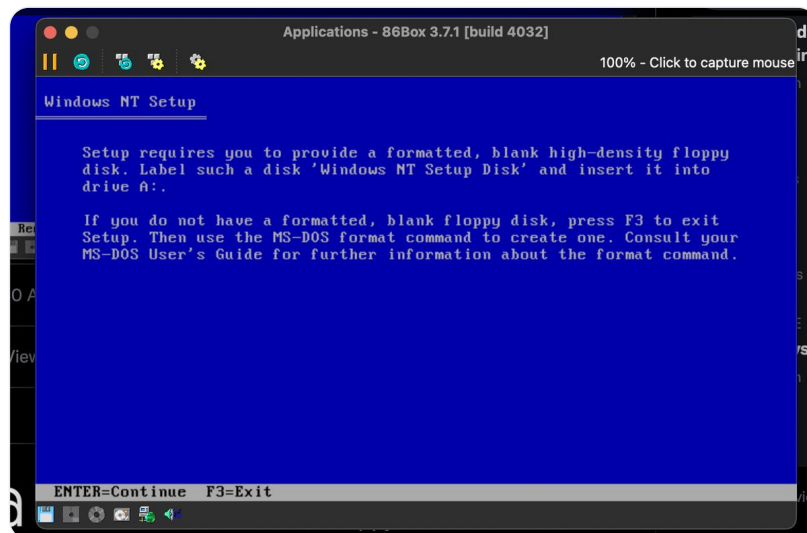
Lets do this ...



Attempt 3 or 4: I forgot how fiddly this is. You need a boot sector on the drive (aka, I need to run sys c: or format with /s), and I might need the setup specific boot floppy, and not just give it /x :P

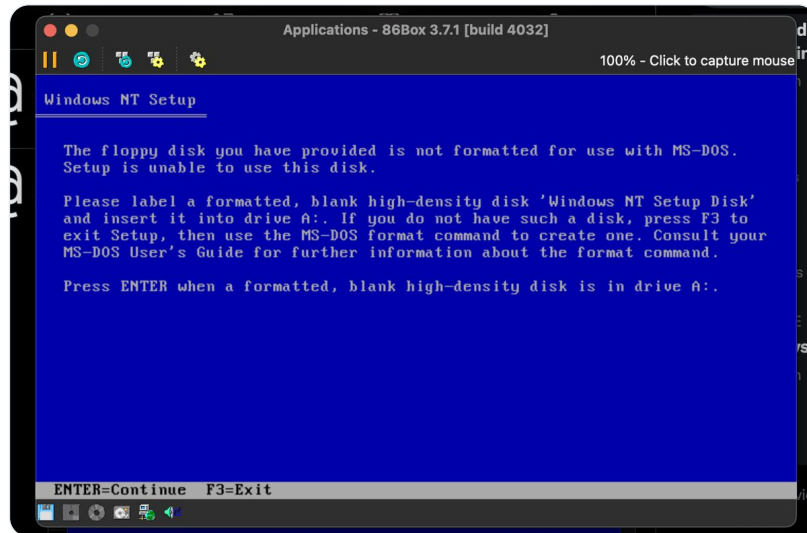


Ok, fine, we'll make a boot floppy ...

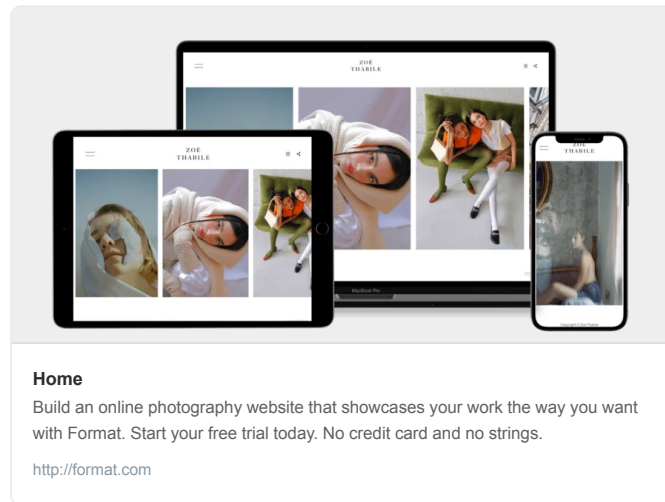


ARGH, *IT HAS TO BE FORMATTED*.

WHY CAN'T YOU DO THIS?!

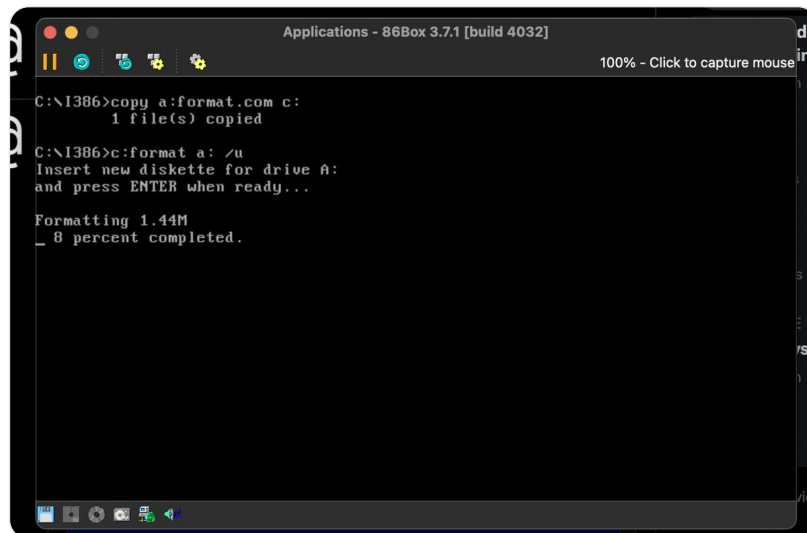


Just take the damn

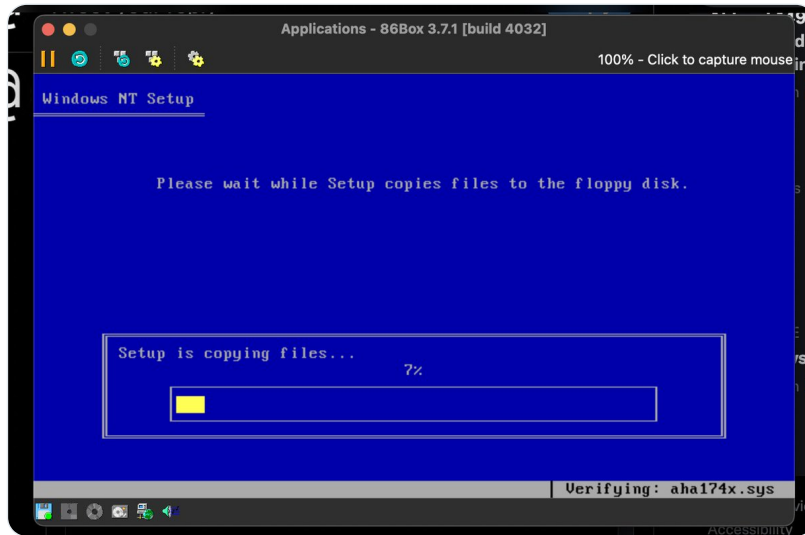


and choke on it :P

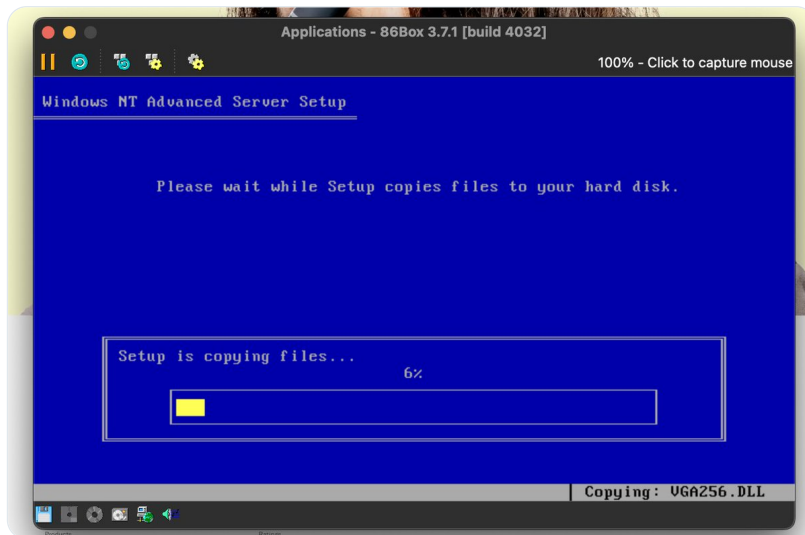
Ugh ... the things I do for content ...



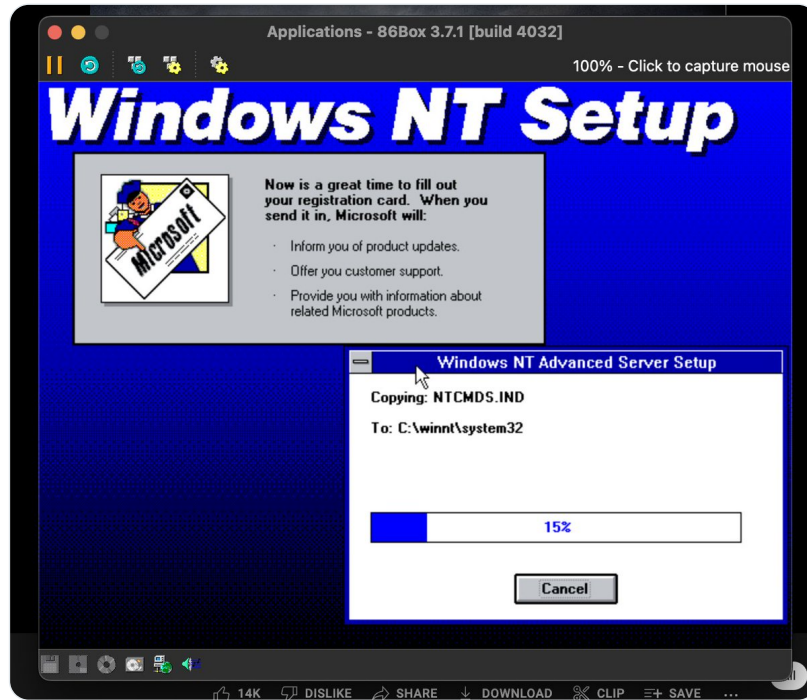
Let there be boot disk ...



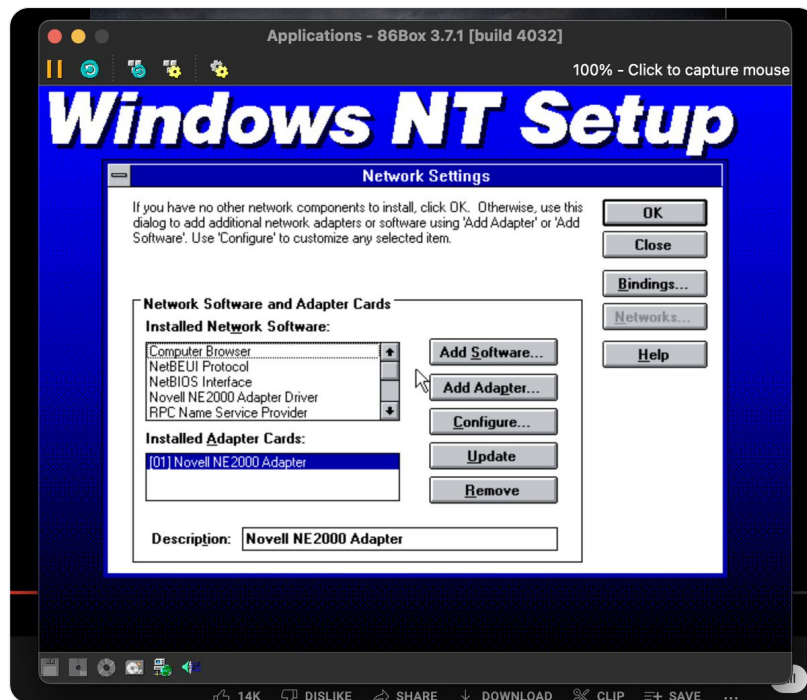
Ok, that seems to have done the trick ...



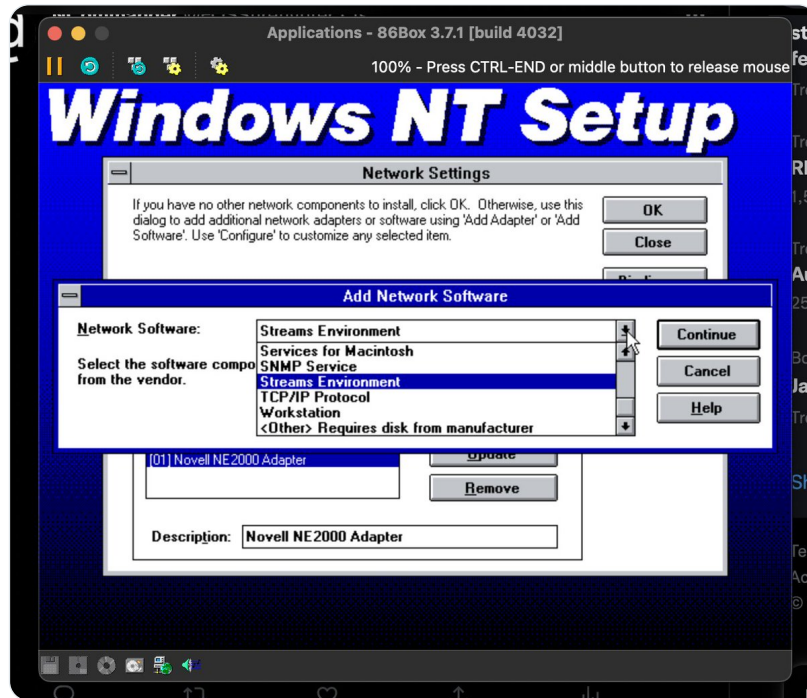
Let's go, NT 3.1 Advanced Server is installing!



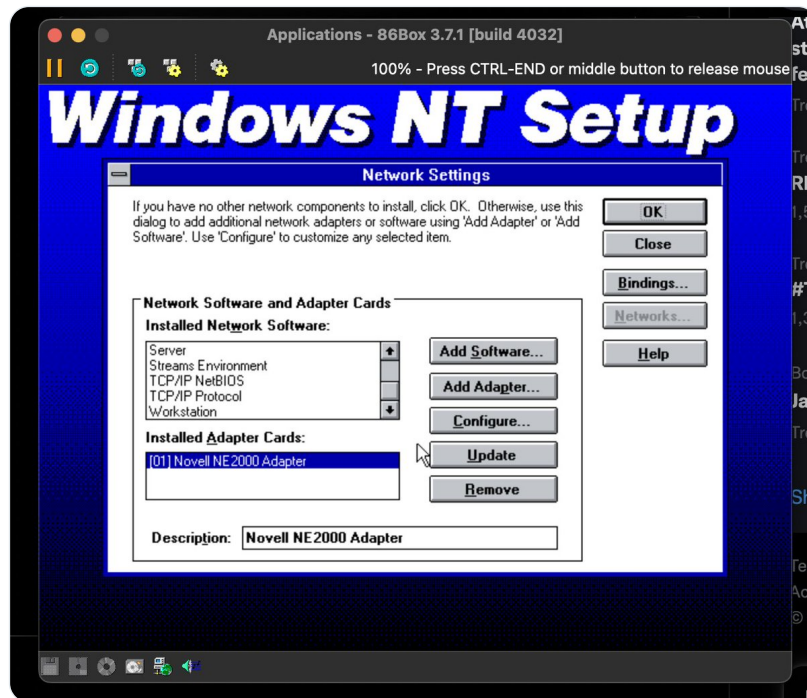
so on NT 3.1, there's no TCP/IP support listed in Network Settings ...



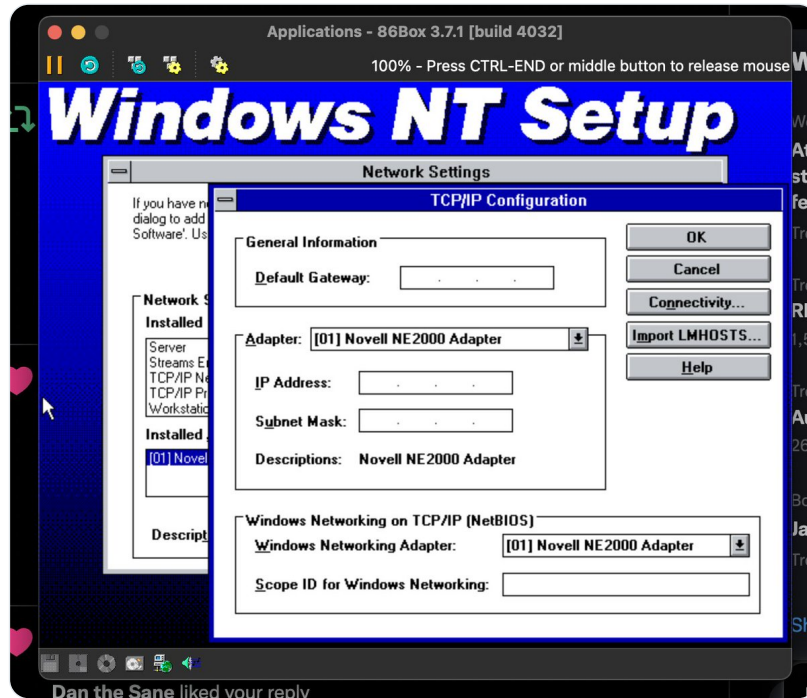
Instead you have to add it as a separate protocol, and it's not well integrated. I haven't actually tried to do much with it, but there's a fairly length section in the resource kit about it.



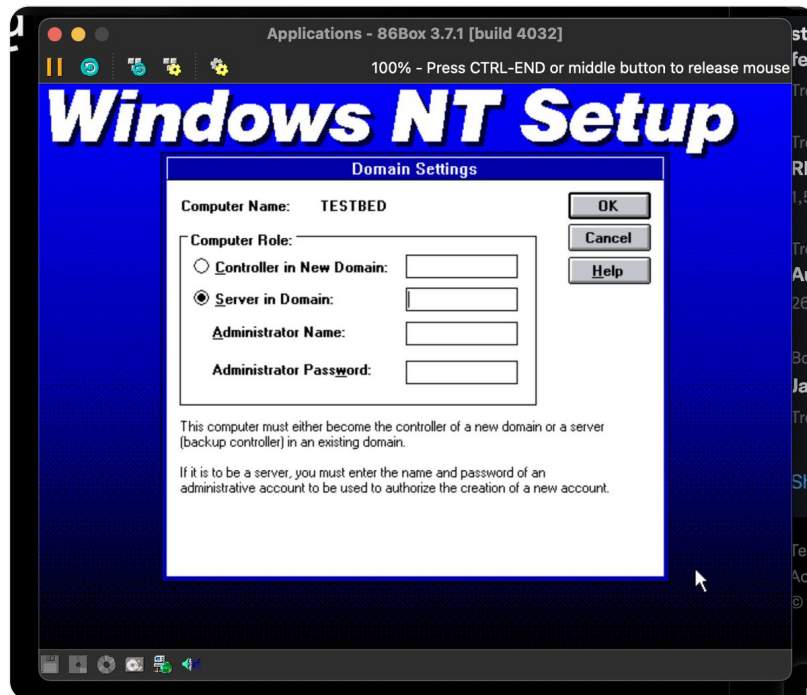
Installing TCP/IP automatically installs the STREAMS environment and NetBIOS over TCP/IP however. So in theory, this should be able to talk to SMB file servers that support LANMAN mode.



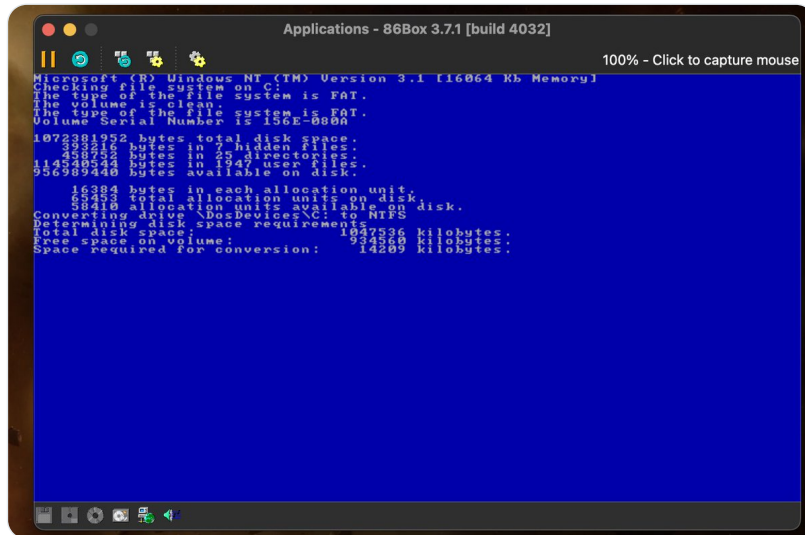
No DHCP support (hasn't been invented yet). I don't think WINS exists either, given it has an option right there to import LMHOSTS.



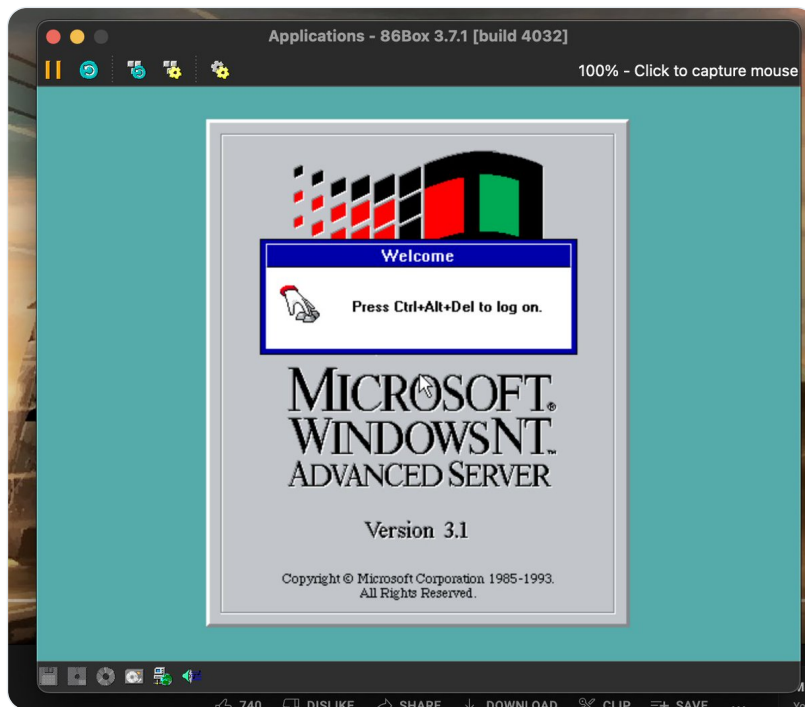
Oh ... I forgot about this. NT 3.1 Advanced Server *must* be in a domain. You can't have it freestanding. That came later ...



anyway, we're through setup, and the drive gets converted to NTFS on the first reboot ...



Let there be advanced server!



Doing some script writing about the install process. Cursed sentence follows:

"[MIPS] considerably better behaved than the x86 version, albeit with a distinct lack of software add-on software."

I am honestly debating just installing it in QEMU and saving myself headache

Anyway, back to script writing, FIPS-151-1 actually includes the text of the POSIX.1-1988 specification it seems, and that's freely available from NIST:

<https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub151-1.pdf>

I'm not actually sure if NT is certified to 1988 or 1990 versions, but they're not very different.

I love the idea of the POSIX standards group holding a WeirdNIX contest, especially "Most Demented", and I will put this in the video. By any chance, is there ANYONE on my feed who might have had first hand experience with this?

(please RT for visibility)

During the process of developing this standard, the Working Group sought to find problems with the standard in a manner that was both fun and which would publicize the standard. The contest is described in the Rationale (see **WeirdNIX** §B.1.2.12). Part of the prize was publication of the names of the winners.

Our special thanks to:

- Paul Gootherts (winner in *Most Serious* category).
- Michael Gersten (winner in *Most Demented* category).

The following persons were members of the 1003.1 Balloting Group that approved the standard for submission to the IEEE Standards Board:

I'm thumbing through the specification now, since you know, before I throw daggers at 90s Microsoft, I'm actually curious what they were on the hook to provide.

It's so far pretty much what you'd expect ...

Well, here's `fork()`, the thing that Win32 **doesn't** have, and POSIX does, as well as being the reason why running `configure` on Cygwin (or MSYS) is an exercise in patience.

(if you value your sanity, don't look at how cygwin does `fork()`)

3.1 Process Creation and Execution.

3.1.1 Process Creation.

Function: `fork()`

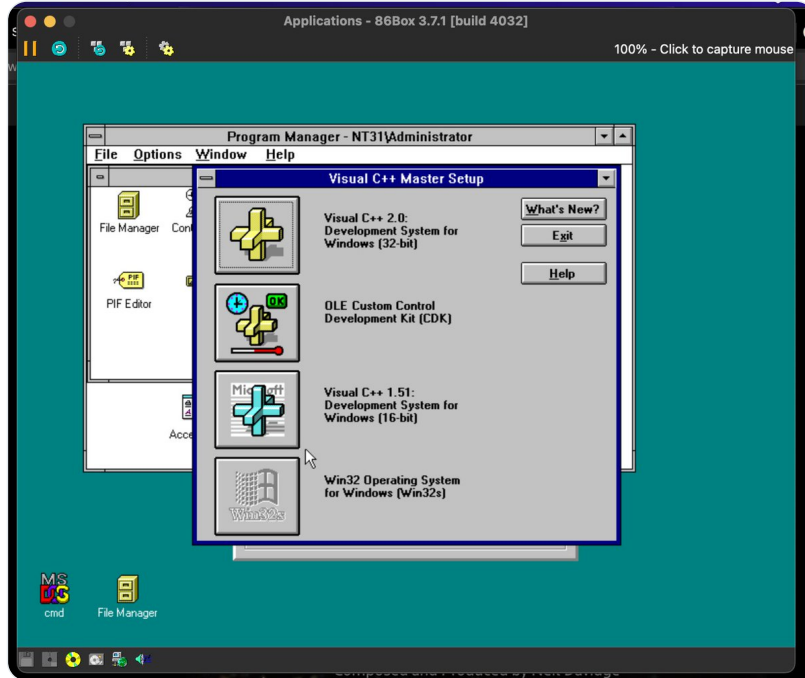
3.1.1.1 Synopsis.

```
#include <sys/types.h>
pid_t fork ( )
```

3.1.1.2 Description. The `fork()` function creates a new process. The new process (child process) shall be an exact copy of the calling process (parent process) except for the following:

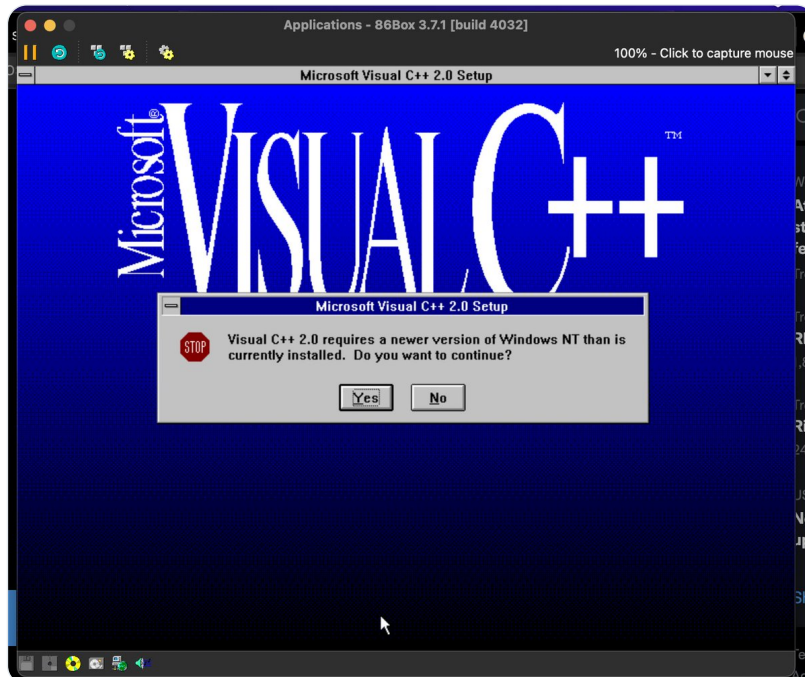
- (1) The child process has a unique process ID. The child process ID also does not match any active process group ID.
- (2) The child process has a different parent process ID (which is the process ID of the parent process).
- (3) The child process has its own copy of the parent's file descriptors. Each of the child's file descriptors refers to the same open file description with the corresponding file descriptor of the parent.
- (4) The child process has its own copy of the parent's open directory streams (see **Directory Operations** §5.1.2). Each open directory stream in the child process may share directory stream positioning with the corresponding directory stream of the parent.
- (5) The child process's values of `tms_utime`, `tms_stime`, `tms_cutime`, and `tms_cstime` are set to zero (see `times()` §4.5.2).
- (6) File locks previously set by the parent are not inherited by the child. (See `fcntl()` §6.5.2.)

Let me install Visual C++ 2.0 real quick. I just want to confirm that POSIX support was not included out of the box. You need the full SDK on NT 4, but I didn't try it on this version.



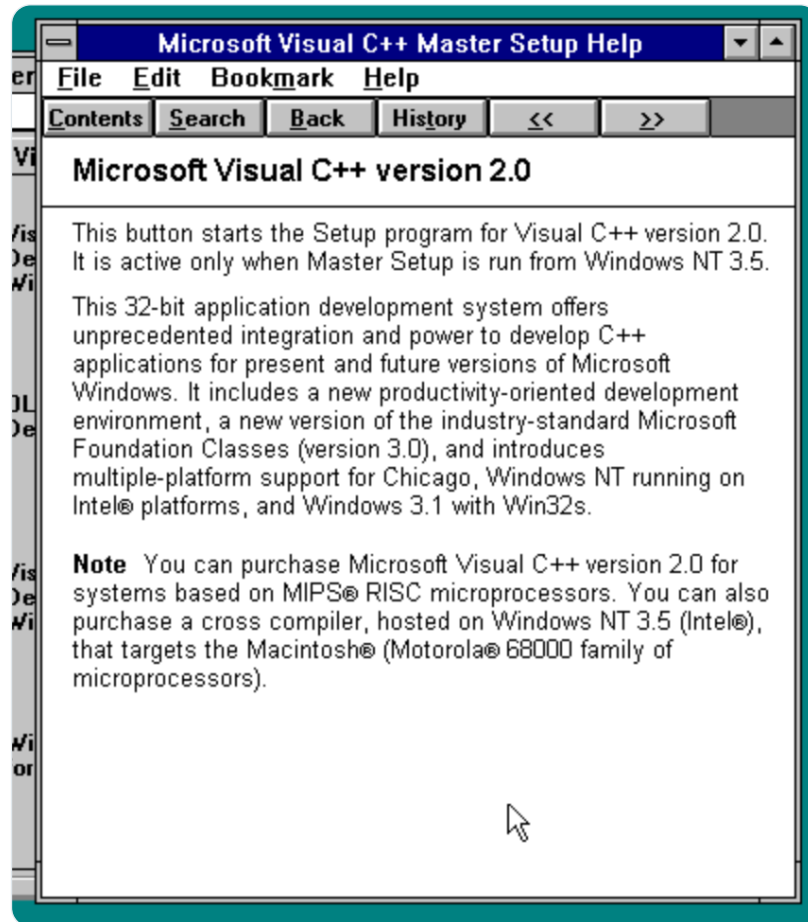
Uhhhhh

... what ...



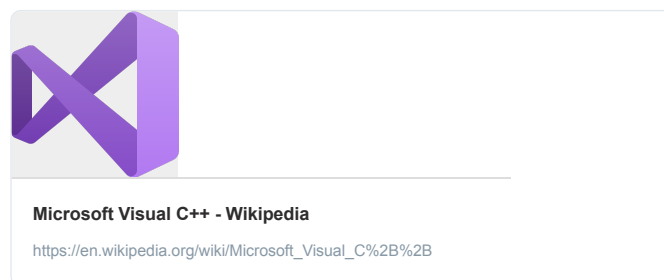
Apparently it's only for NT 3.5?!

... what the hell was the NT 3.1 development tool then?



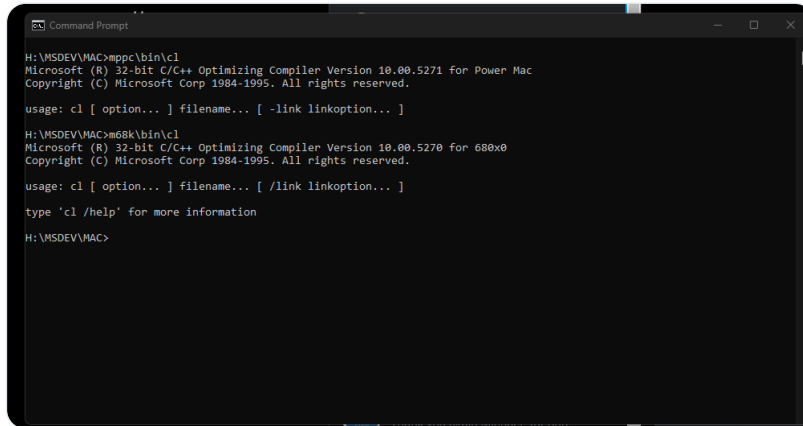
Ok, there's apparently a Visual C++ 1.0 for NT which is 32-bit. Thank you random commenter on Winworld for pointing that out.

Wikipedia rather notably doesn't make this clear:



Oh, incidentally, since it came up in a reply, there was a copy of VC++ for Macintosh, I actually do have a copy and the physical disc of it. Have a screenshot of some cursed compilers.

If this gets enough likes, I'll spend a day installing and documenting it more.



```
Command Prompt
H:\MSDEV\MAC\mpc\bin\cl
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 10.00.5271 for Power Mac
Copyright (C) Microsoft Corp 1984-1995. All rights reserved.

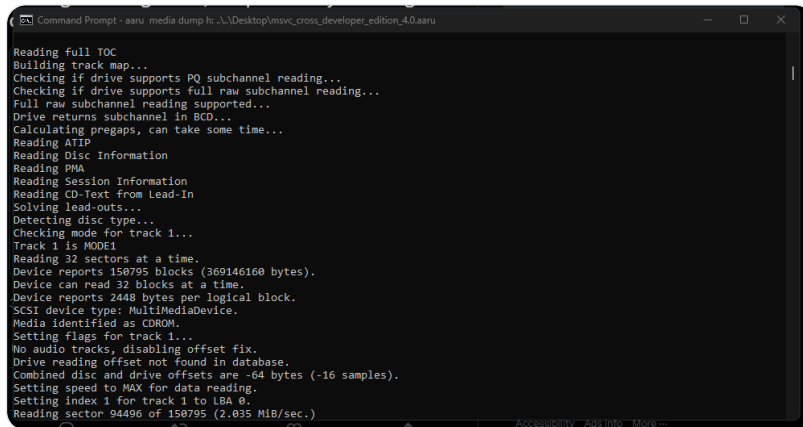
usage: cl [ option... ] filename... [ -link linkoption... ]

H:\MSDEV\MAC\m68k\bin\cl
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 10.00.5270 for 680x0
Copyright (C) Microsoft Corp 1984-1995. All rights reserved.

usage: cl [ option... ] filename... [ /link linkoption... ]

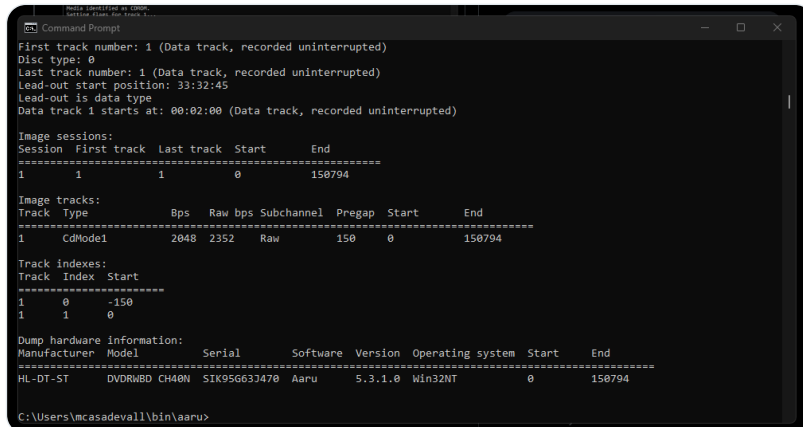
type 'cl /help' for more information
H:\MSDEV\MAC>
```

The disk actually has a ISO9660 and HFS partition on it, and I'm not certain if there's an actually good dump of this disk in existence, so I'm running it through [@TinkeringDaemon's aaru](#) to get the whole thing preserved.



```
Command Prompt - aaru media dump h:\Desktop\msvc_developer_edition_40.aaru
Reading full TOC
Building track map...
Checking if drive supports R0 subchannel reading...
Checking if drive supports full raw subchannel reading...
Full raw subchannel reading supported...
Drive returns subchannel in BCD...
Calculating pregaps, can take some time...
Reading ATIP
Reading Disc Information
Reading PMA
Reading Session Information
Reading CD-Text from Lead-In
Solving lead-outs...
Detecting disc type...
Checking mode for track 1...
Track 1 is MODE1
Reading 32 sectors at a time.
Device reports 150795 blocks (369146160 bytes).
Device can read 32 blocks at a time.
Device reports 2448 bytes per logical block.
SCSI device type: MultiMediaDevice.
Media identified as CDROM.
Setting flags for track 1...
No audio tracks, disabling offset fix.
Drive reading offset not found in database.
Combined disc and drive offsets are -64 bytes (-16 samples).
Setting speed to MAX for data reading.
Setting index 1 for track 1 to LBA 0.
Reading sector 94496 of 150795 (2.035 MiB/sec.)
```

[@TinkeringDaemon](#) aaru says its a single session disk, which is similar to what I've seen from other MS Macintosh hybrids. The existing ISOs are probably fine as is, but I'll make sure this gets preserved properly.



```
Command Prompt
First track number: 1 (Data track, recorded uninterrupted)
Disc type: 0
Last track number: 1 (Data track, recorded uninterrupted)
Lead-out start position: 33:32:45
Lead-out is data type
Data track 1 starts at: 00:02:00 (Data track, recorded uninterrupted)

Image sessions:
Session First track Last track Start End
-----
1 1 1 0 150794

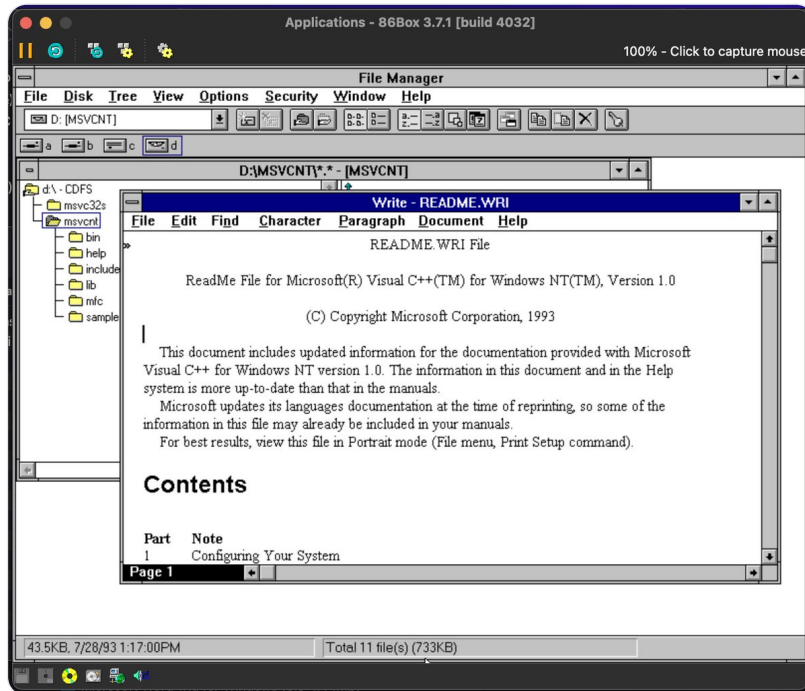
Image tracks:
Track Type Bps Raw bps Subchannel Pregap Start End
-----
1 CdMode1 2048 2352 Raw 150 0 150794

Track indexes:
Track Index Start
-----
1 0 -150
1 1 0

Dump hardware information:
Manufacturer Model Serial Software Version Operating system Start End
-----
HL-DT-ST DVD RWBD CH40N SIK95G63J470 Aaru 5.3.1.0 Win32NT 0 150794

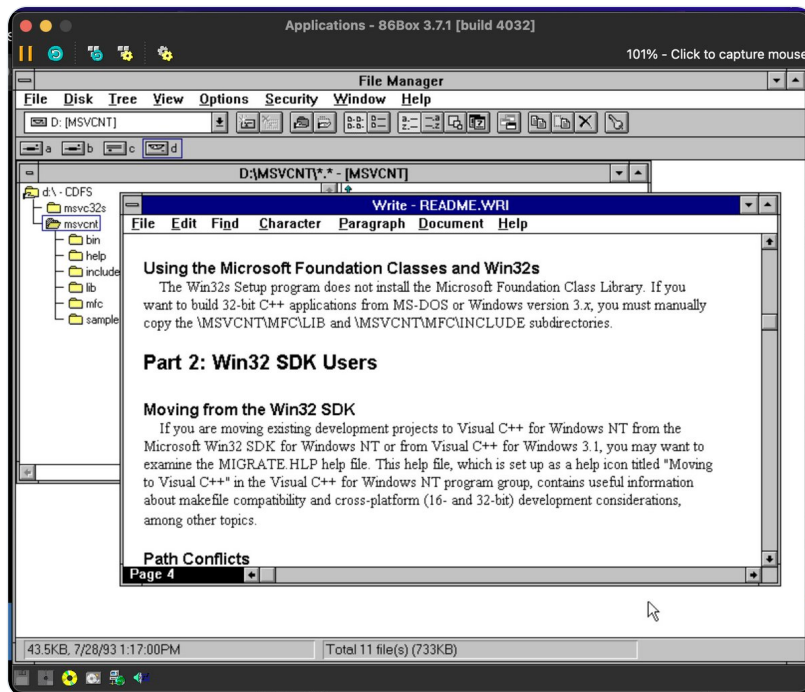
C:\Users\mcasadevall\bin\aaruu>
```

@TinkeringDaemon Ok, back to the original topic, it's usually worth thumbing through the README on these things ...

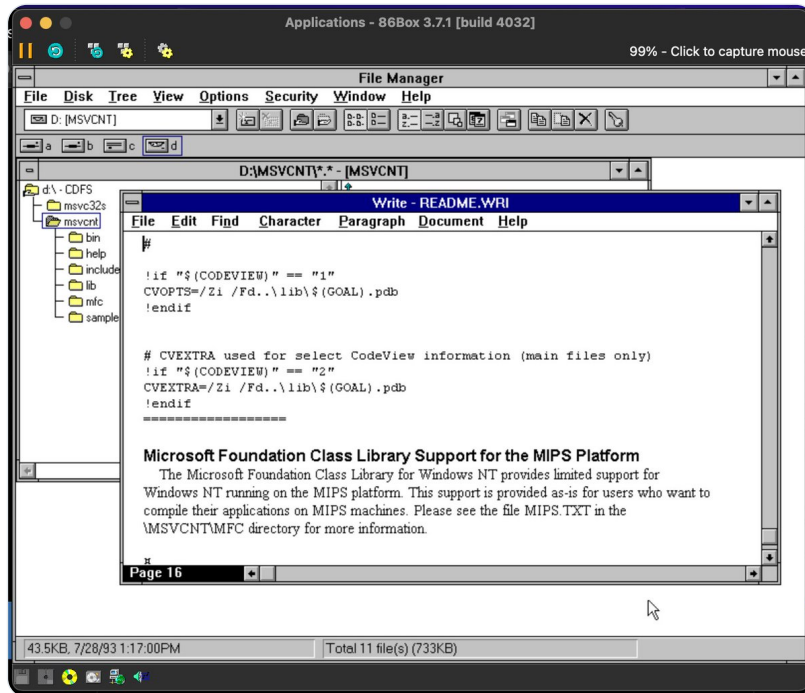


So this is an important note, there was a specific Win32 SDK with its own cl386 compiler (which was also used on OS/2).

I've seen this in pre-release copies of NT 3.1, but I should also look in the freestanding SDK to see if that compiler was there.

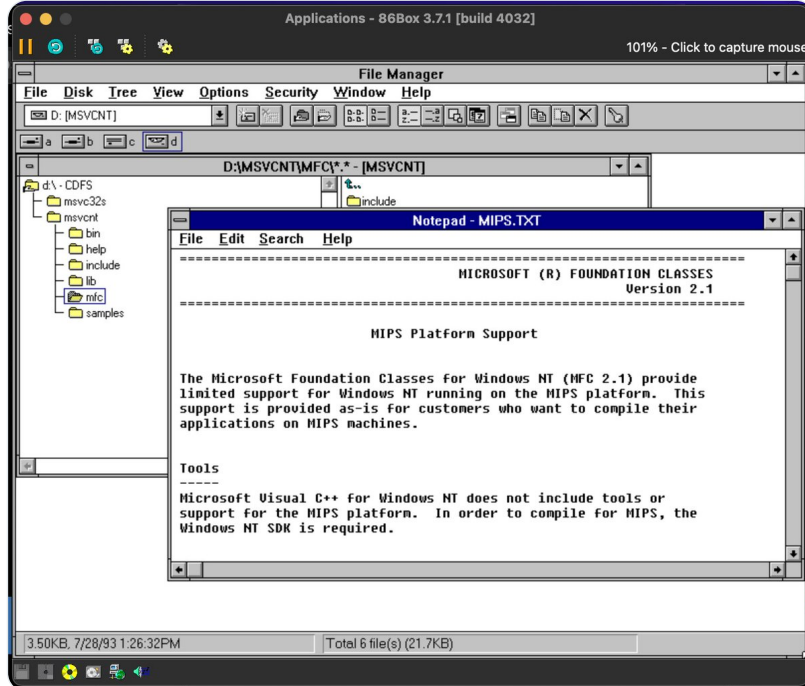


Interesting, MFC wasn't officially support on MIPS ...



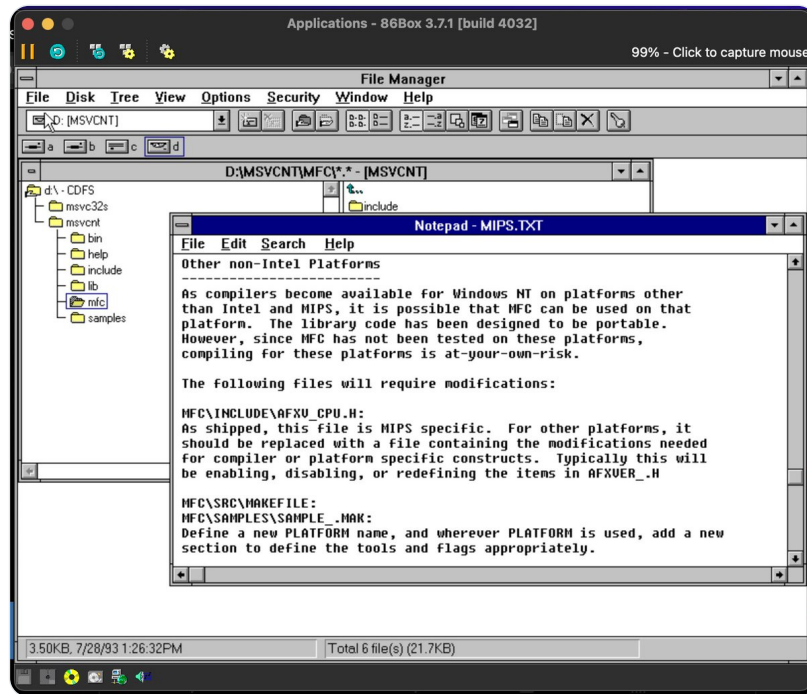
Ah, there's the reason.

No Visual C++ for MIPS. There were special "Visual C++ for RISC Editions", that could be used, but apparently those didn't exist yet; you just had the command line tools.

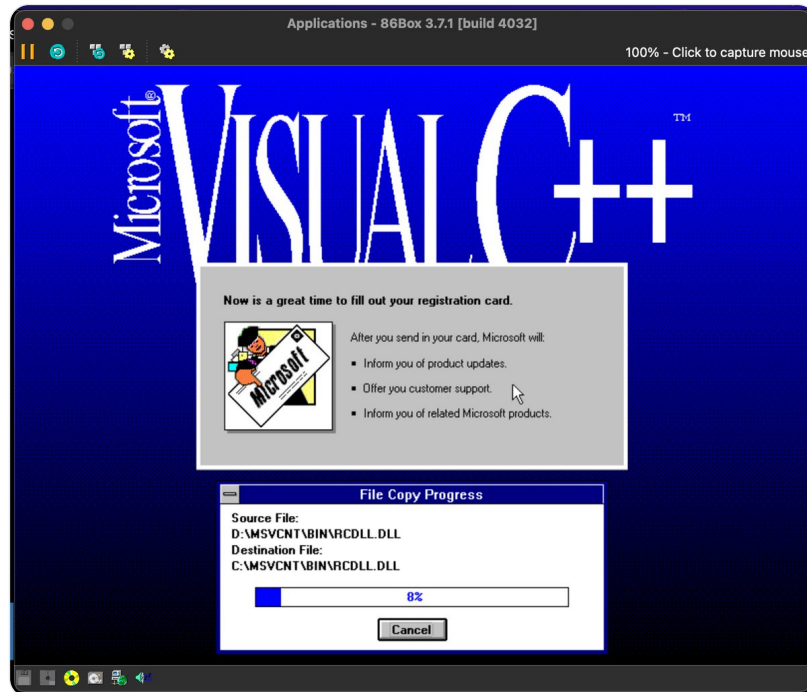


Interesting, the README actually talks a fair bit about using non MSVC compilers, and porting it. DEC Alpha support wasn't long off at that point, and there was a MFC port to Macintosh as well ...

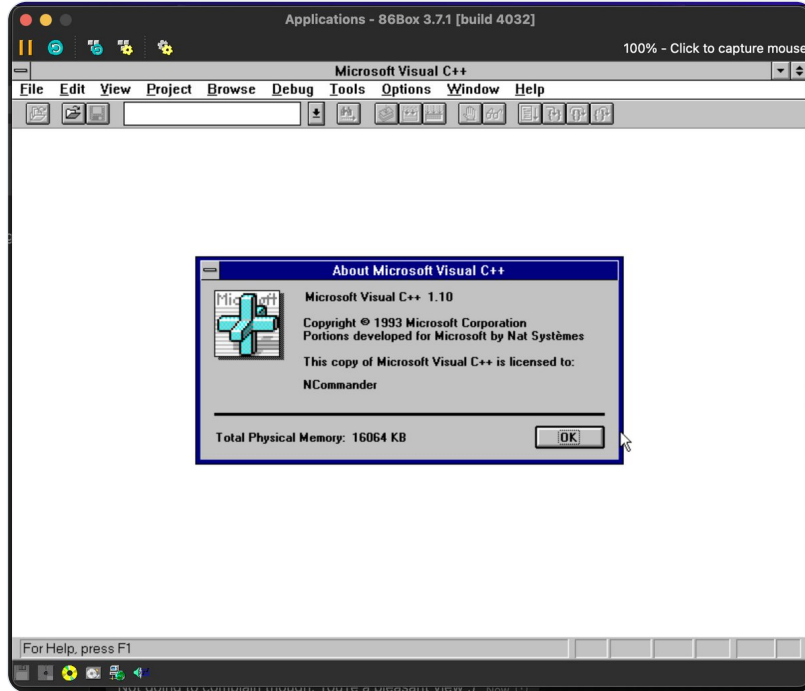
You could even get MFC to build under winelib if you really liked pain ...



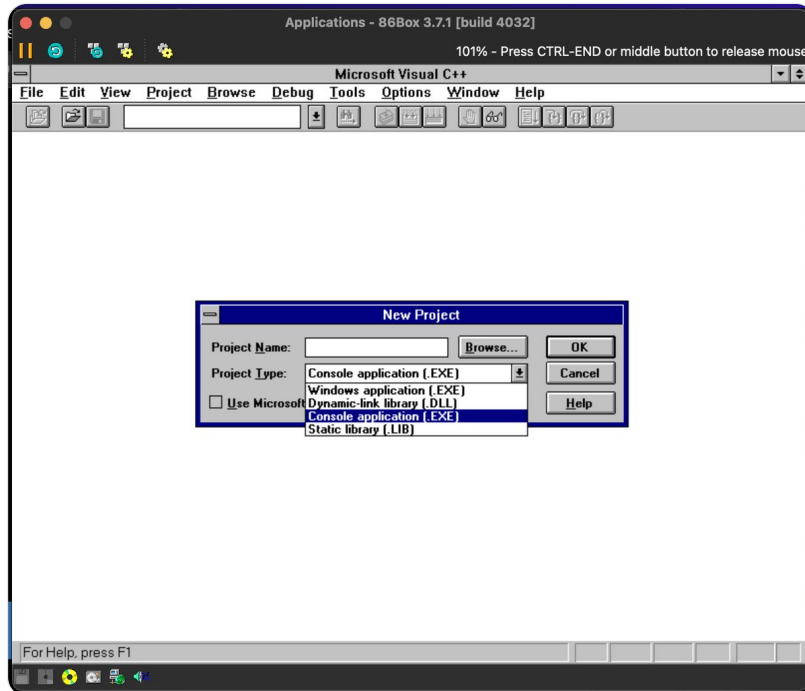
Anyway, let's go ...



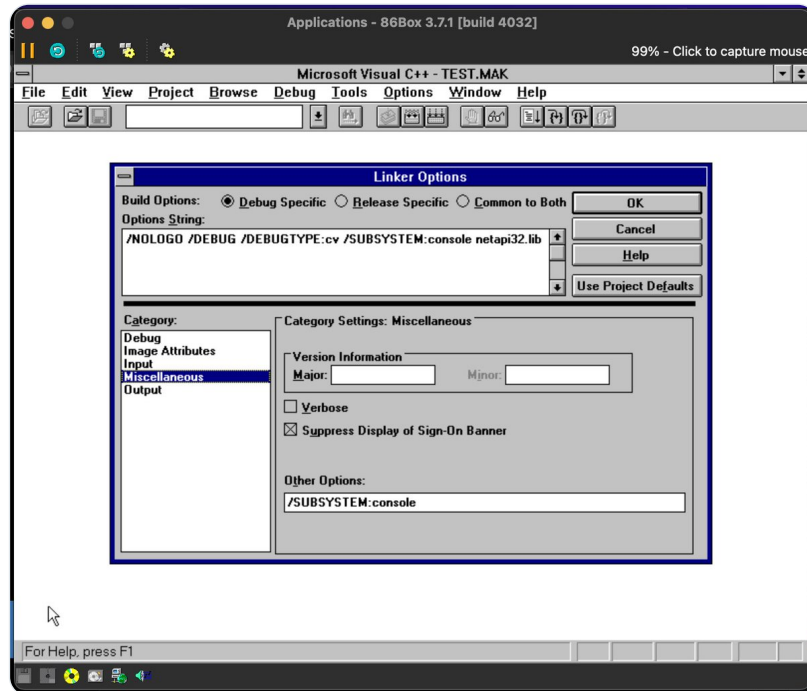
Well, that's pretty much what I expected. This is identical to the 16-bit VC++. Let me see if it can generate a POSIX binary directly. I suspect not, but let me rule it out ...



Yeah I'm going to go with "it can't" ...

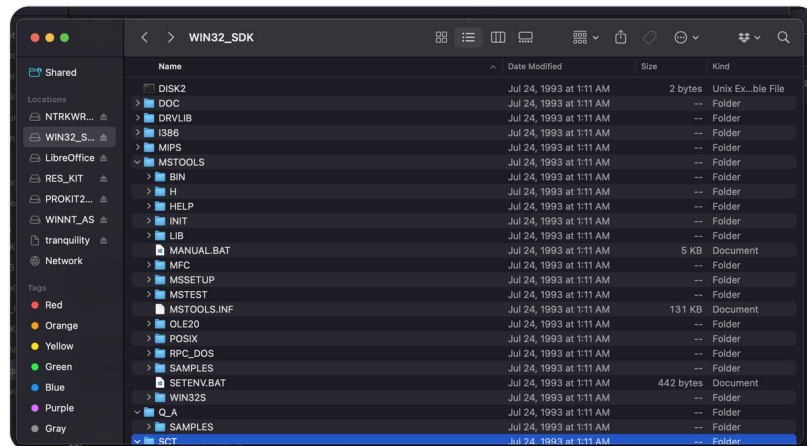


There's no option to set the necessary SUBSYSTEM flags, I haven't checked yet, but the necessary files are likely also missing ...



Ok, so this is the Win32 SDK disk referenced. It actually has a full copy of NT 3.1 on it, a copy of the C compiler for MIPS and i386 *and* the POSIX headers.

So yeah, Visual C++ never supported building POSIX binaries. Good to know.



There should actually be a later edition of this disc or an update, which has DEC Alpha support (NT 3.1 was originally released with MIPS and x86, a re-release added DEC Alpha), but I suspect that's been lost to time.

so uh, I just looked at when I started on this thread, and I've been at this for 8 hours ... WTF

...

how ...

Anyway, if you've enjoyed this thus far, consider supporting me on Ko-Fi (ko-fi.com/ncommander), or Patreon (



NCommander is creating retrocomputing videos | Patreon

Become a patron of NCommander today: Get access to exclusive content and experiences on the world's largest membership platform for artists and creators.

<https://patreon.com/ncommander>

) ...

I'm mulling if I want to work on this more today ...

Alright, I think I want to start by summarizing the technical points going forward:

"It's here we start needing to get into the technical nitty gritty, and more specifically, let's talk about what an application on NT really is."

The next question is, how deep do I want to touch on OS/2 support ... it's somewhat easier to demonstrate than most but that's a deep rabbit hole.

"Internally, the core of Windows NT, its kernel, has its own defined programming interface and methodology, and it is possible to write programs that use this Native API using the driver development kit"

I fear I might end up doing a Hello World native mode application ...

So yeah, I think I'm going to end up doing this, mostly because I really want to demonstrate just how different these things are ...

So, looking at the PE specification, it does define a OS/2 value, even though OS/2 never used PE. This might have been a leftover for NT OS/2, but I do wonder if you could actually compile a 32-bit binary that uses the OS/2 text mode API ...

Constant	Value	Description
IMAGE_SUBSYSTEM_UNKNOWN	0	An unknown subsystem
IMAGE_SUBSYSTEM_NATIVE	1	Device drivers and native Windows processes
IMAGE_SUBSYSTEM_WINDOWS_GUI	2	The Windows graphical user interface (GUI) subsystem
IMAGE_SUBSYSTEM_WINDOWS_CUI	3	The Windows character subsystem
IMAGE_SUBSYSTEM_OS2_CUI	5	The OS/2 character subsystem
IMAGE_SUBSYSTEM_POSIX_CUI	7	The Posix character subsystem

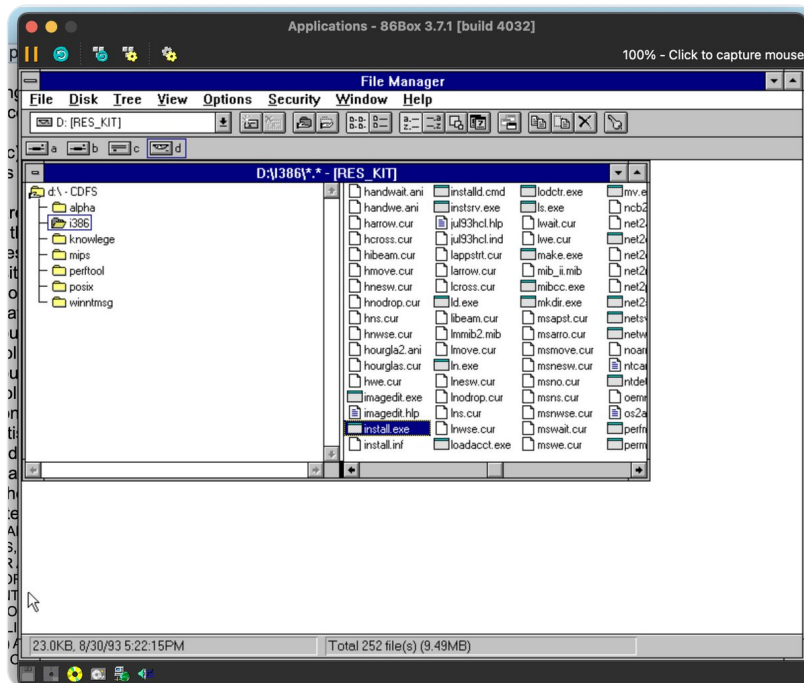
OS/2 does actually have some support for 32-bit code in text mode, but the API is very clearly unfinished and abandoned. the zine project, investigated this awhile ago, very cool project: <https://www.patreon.com/posts/project-zine-16513790>

(Patreon post, but the article is freely available)

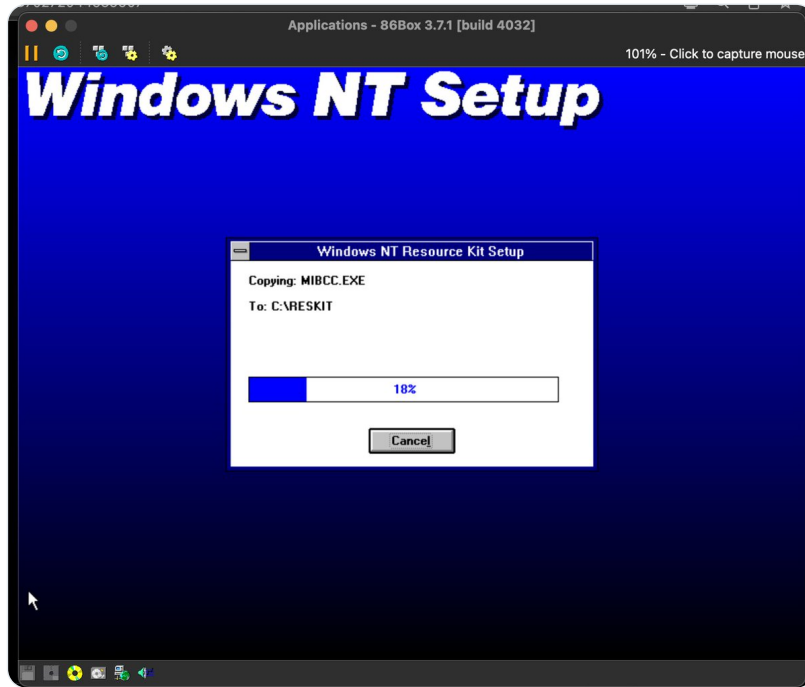
At this point, starting to write up about the Windows NT Resource Kit and software development kits ...

the deep dive is deep ...

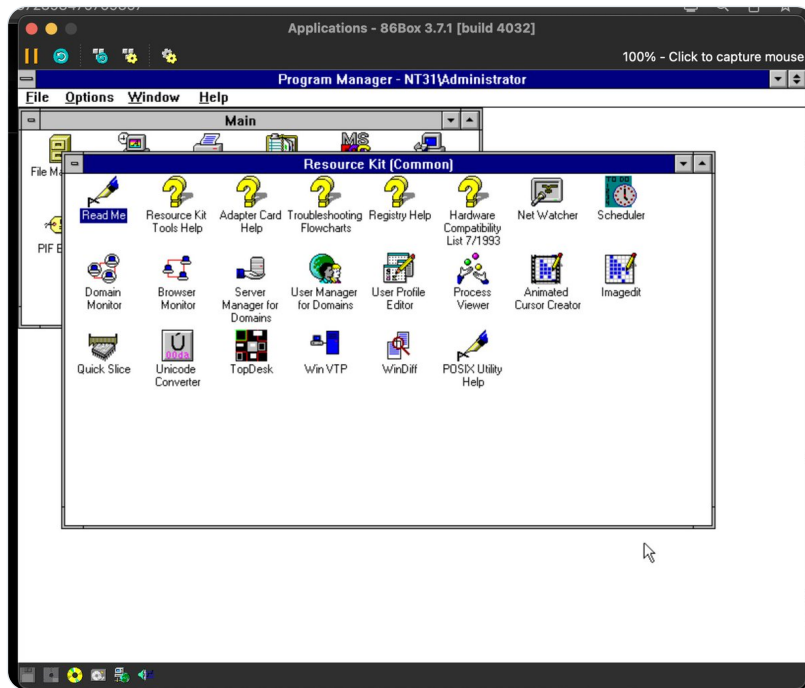
Alright, let's do a bit more research as I script write, let's install the NT resource kit on 3.1 ...



This is about as bare bones as it gets ...



That installs a lot of things ...



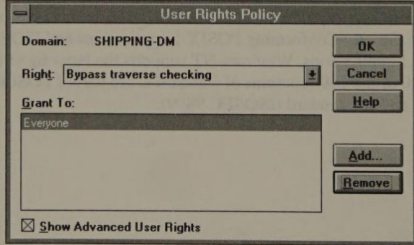
The documentation states we need to disable Traverse checking. I'm not actually sure if this is truly needed, or just needed to make the POSIX environment actually compliant ...

Bypass Traverse Checking

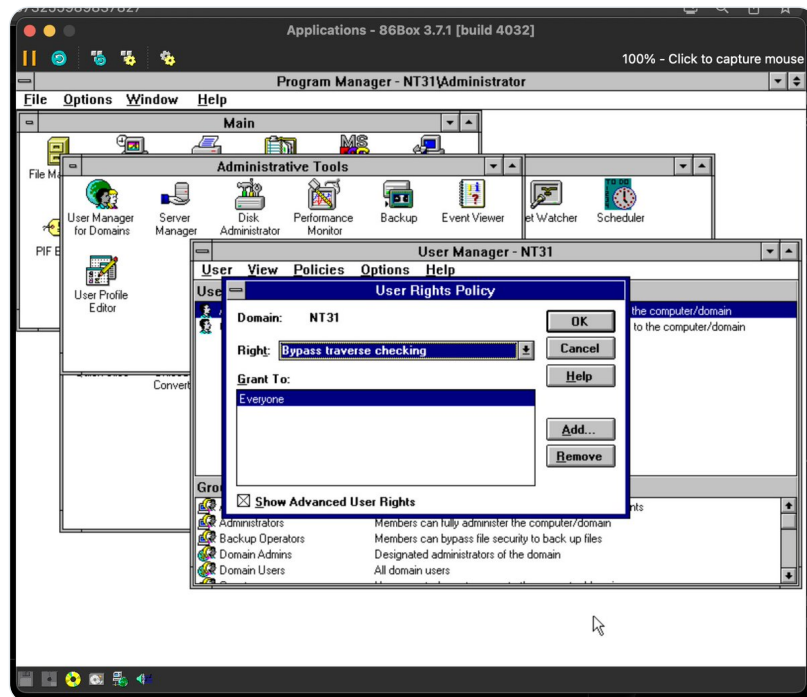
By default, when you install Windows NT for the first time, the user right Bypass Traverse Checking is granted to everyone. This right allows a user to change directories through a directory tree even if the user has no permission for those directories.

If you want to run in a POSIX-conforming environment, you must disable this privilege for your account by using either the User Manager or User Manager for Domains tool as follows (you must be an administrator to do this):

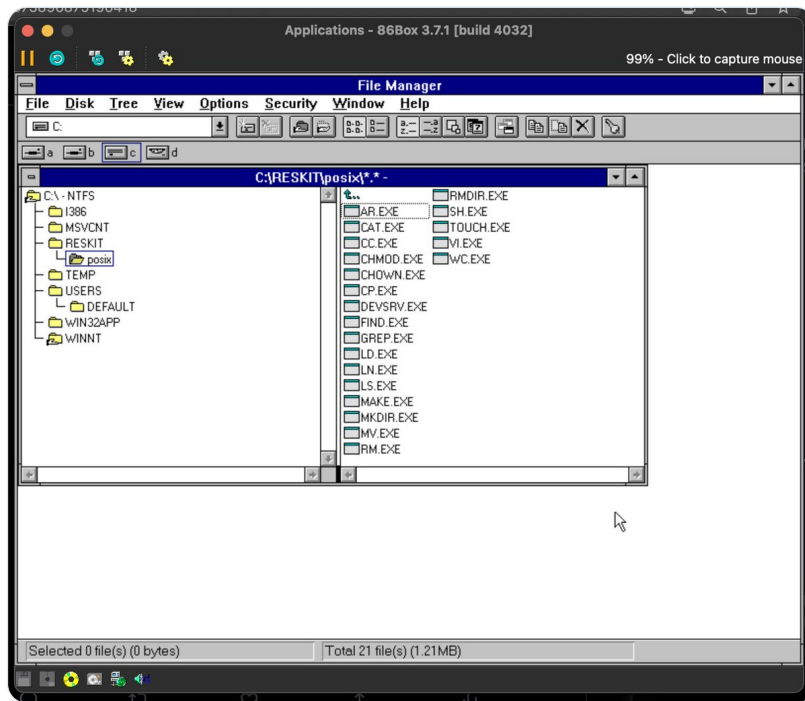
Select the account, and then choose User Rights from the Policies menu to display the following dialog box. (Be sure the Show Advanced User Rights check box is marked.) Specify the Bypass traverse checking right and choose Remove.



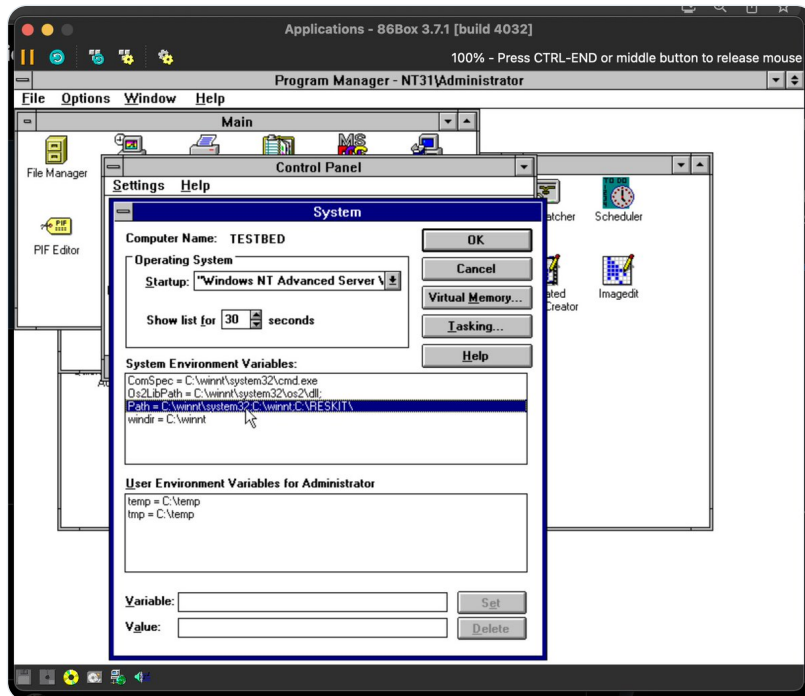
That's easy enough ... *please don't hose my install*



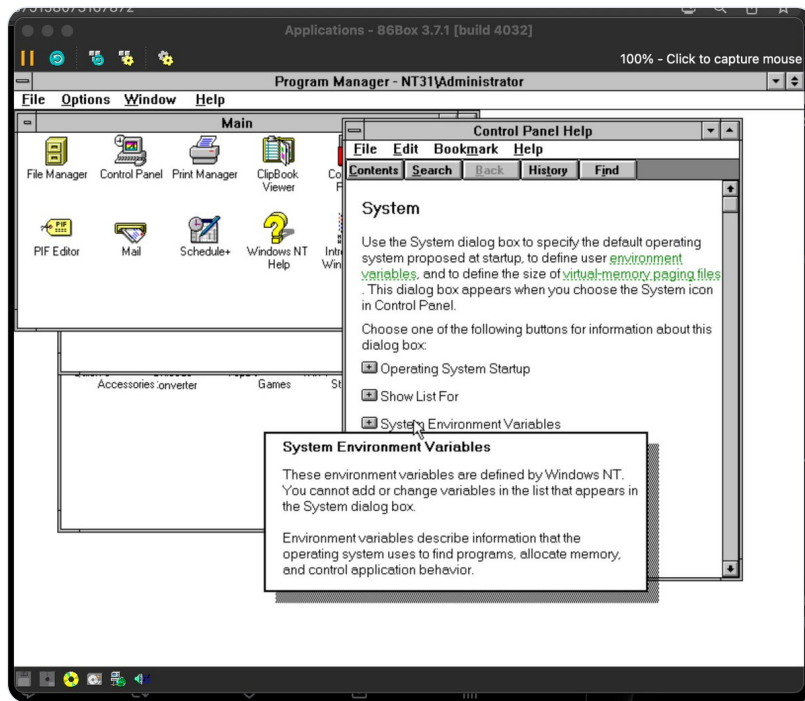
Resource kit binaries, I think I just need to add these to the PATH ...



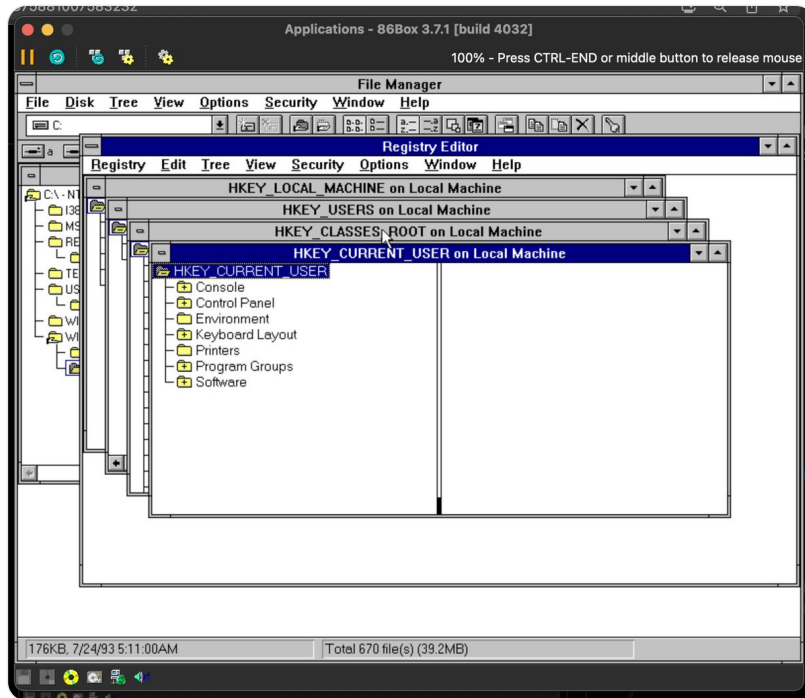
Cute problem, it doesn't appear there's a UI for editing the system PATH in NT 3.1 ... I think I need to go to the registry for this >.;>



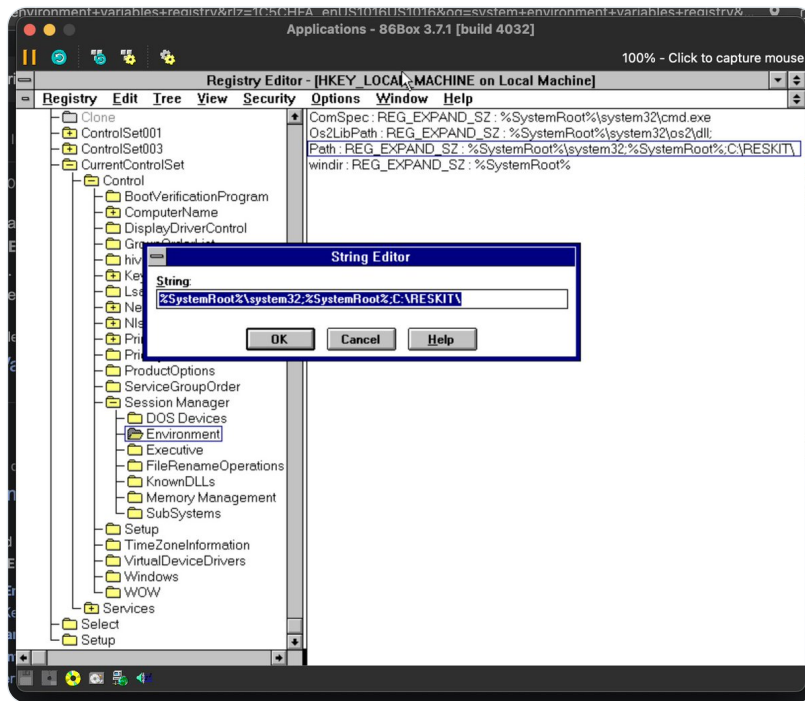
what the hell Microsoft ... *sigh* ...



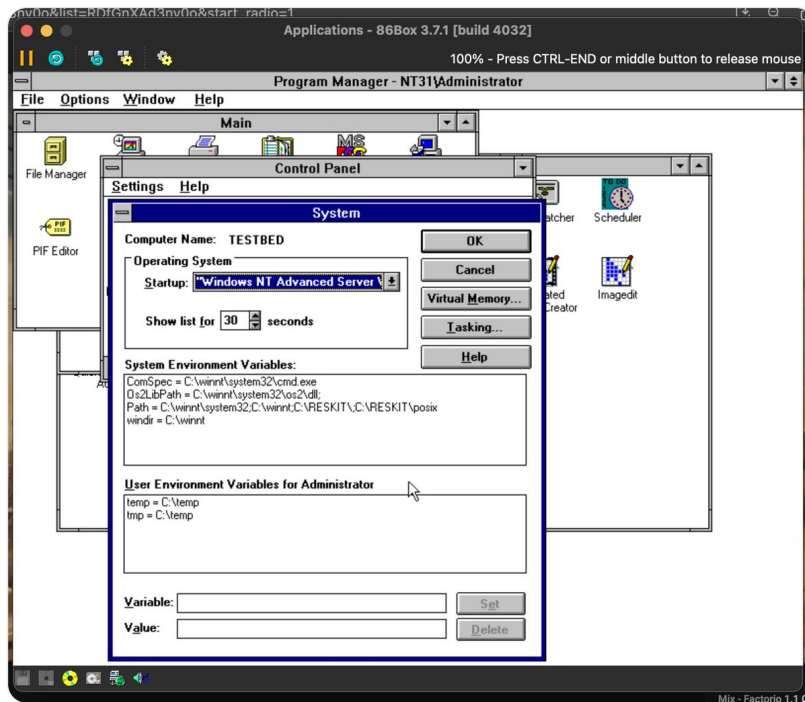
So, NT 3.1 doesn't have regedit. Instead we have rededt32 which is uh ... nowhere near as nice. Time to go hunting ...



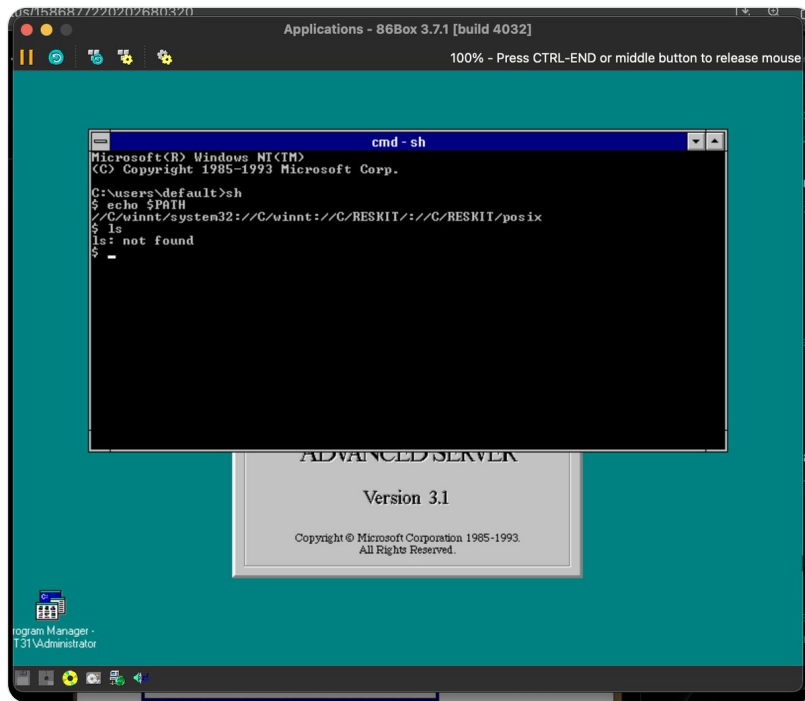
Do you trust the user?



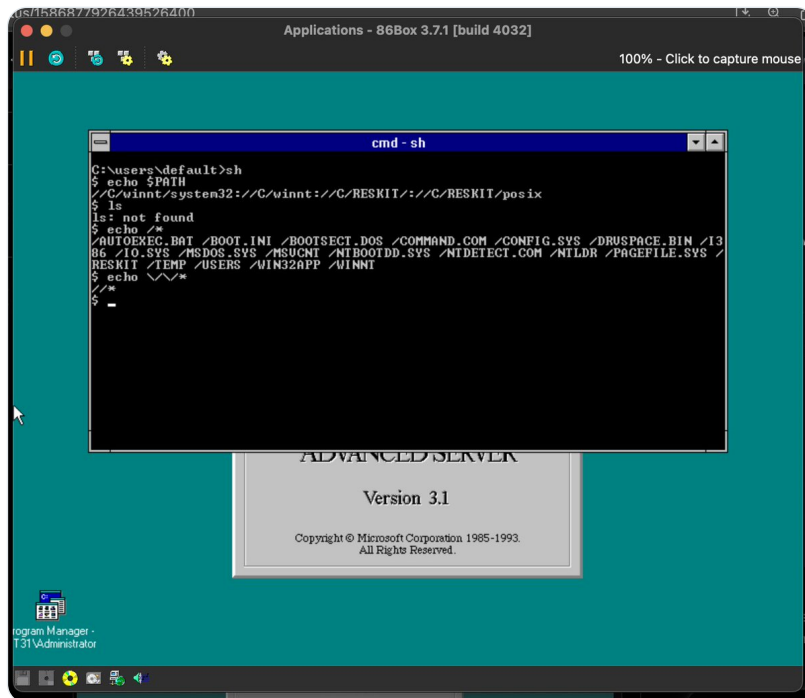
Well, I didn't brick it!



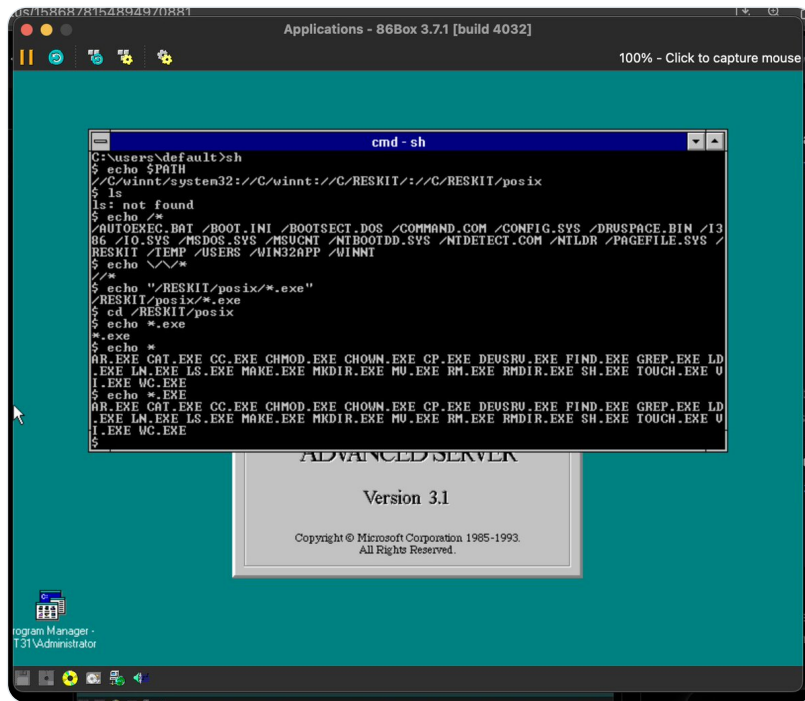
Partial success?



Hrm ... the path mapping doesn't work anywhere how I'd expect and *of course this isn't documented* ...



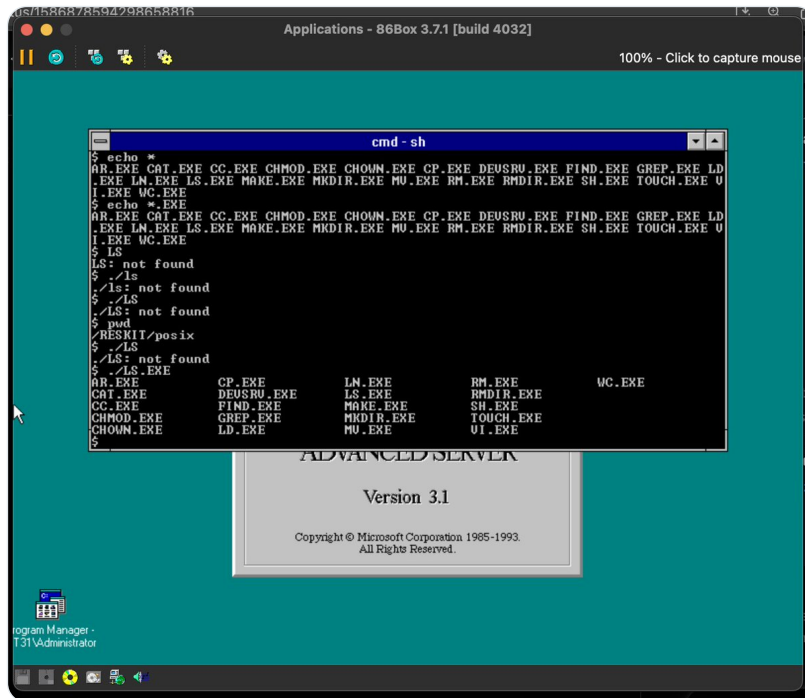
God this is surreal. POSIX applications are case sensitive(!) ... wait ...



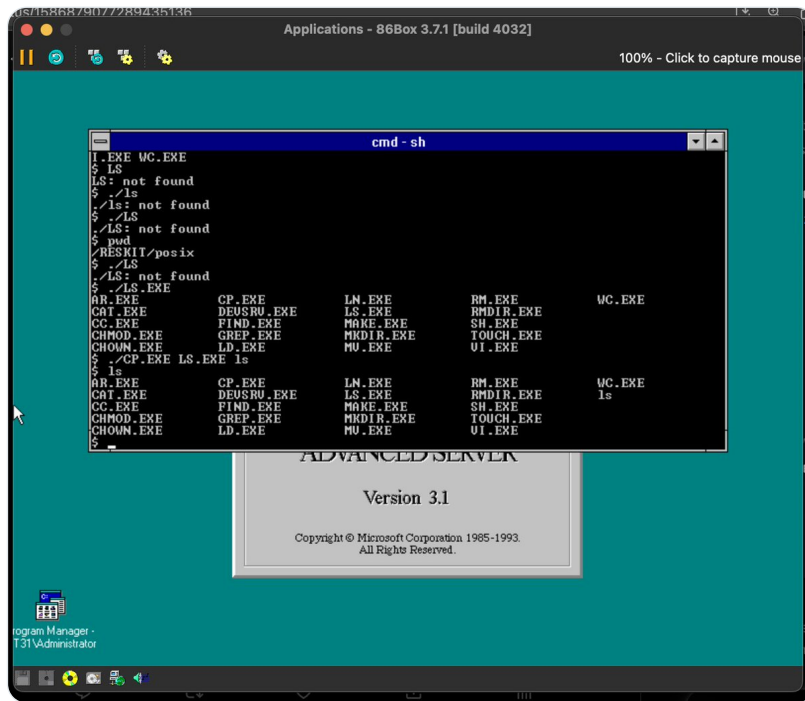
... so it needs to be "LS.EXE" to work ...

That's ... *what* ...

I knew this was half baked but that's *impressively bad*

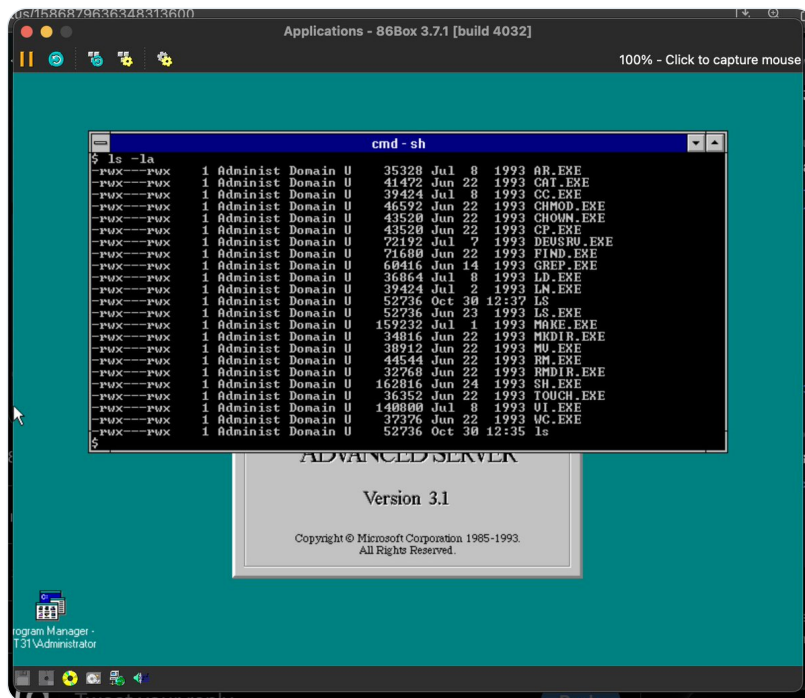


... mom, come pick me up, I'm scared

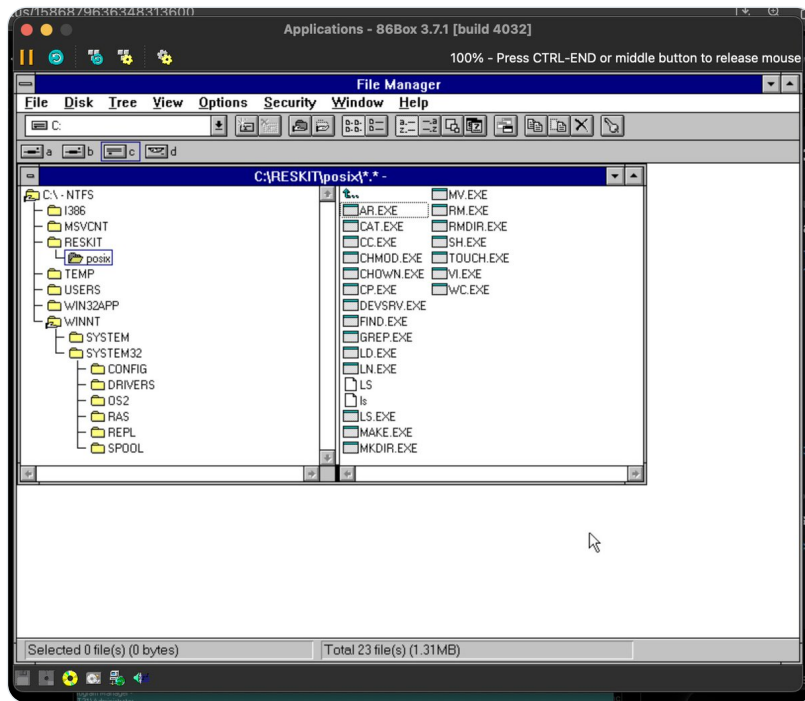


So, in this screenshot

- you can see NT users are mapped into POSIX
- you can have filenames with differing case
- for shit to work with ksh, it needs to be allow lower case. Honestly, I might write a patch for this ...

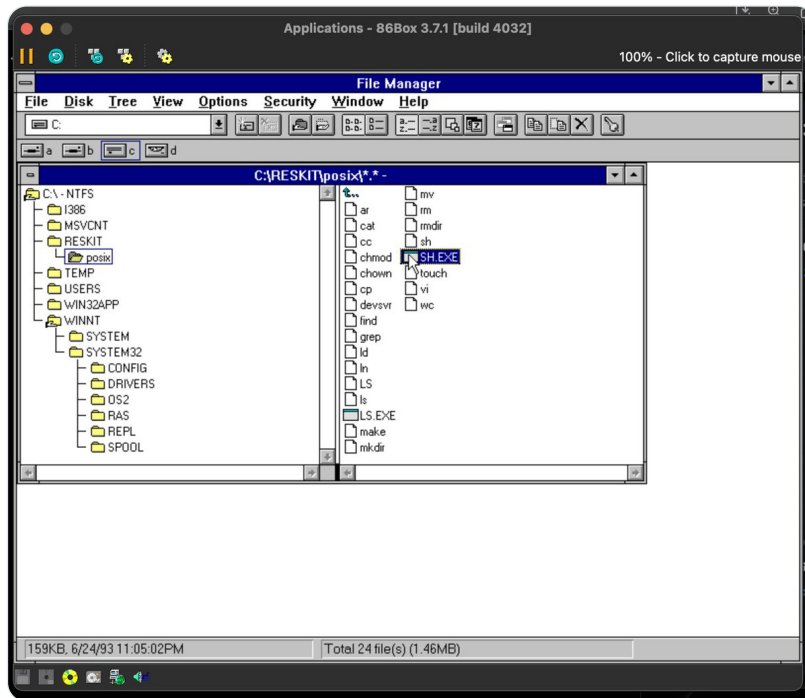


Seeing this in fileman is cursed. Properly cursed. I haven't even talked about that CC.EXE ...

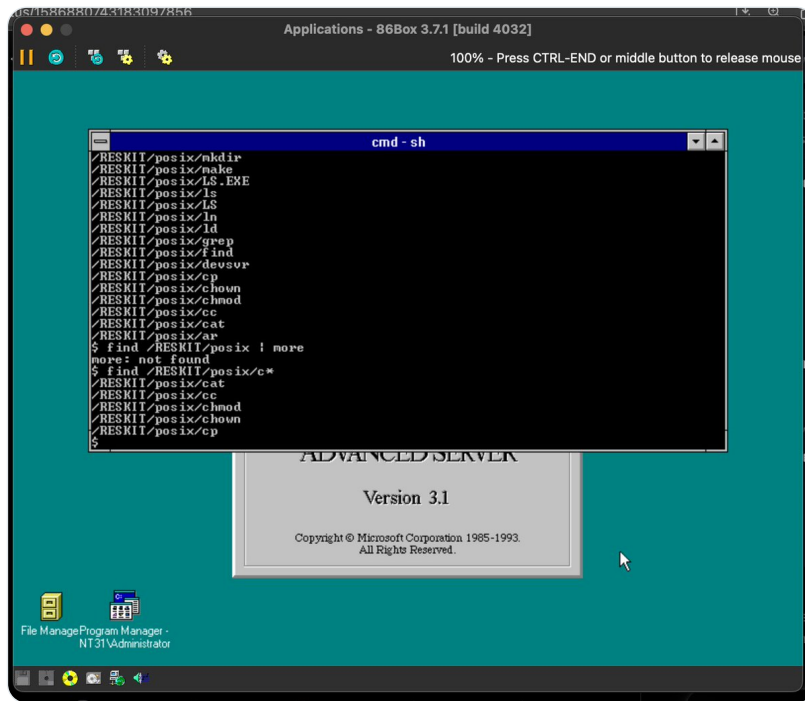


Some renaming later ...

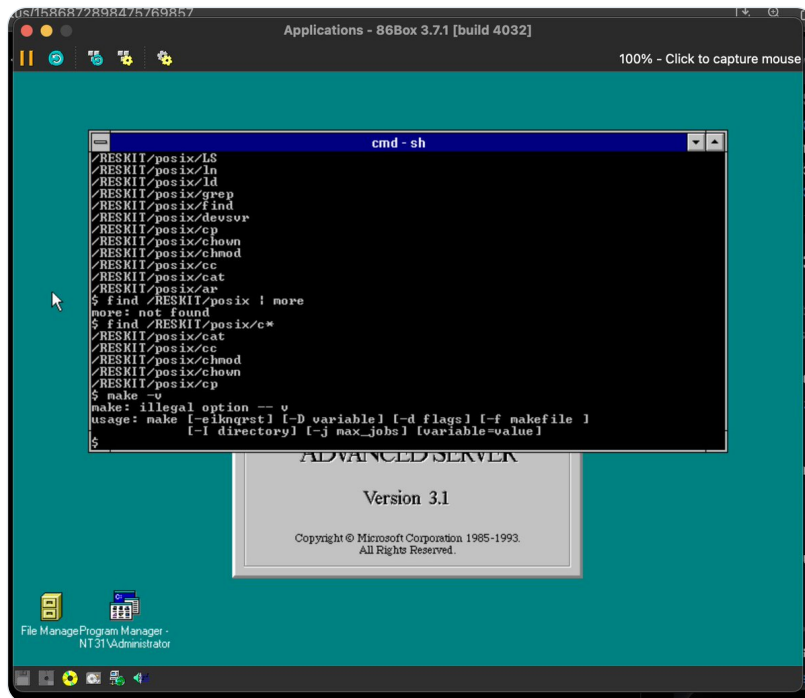
(I should probably link these to /bin)



Well *find works*

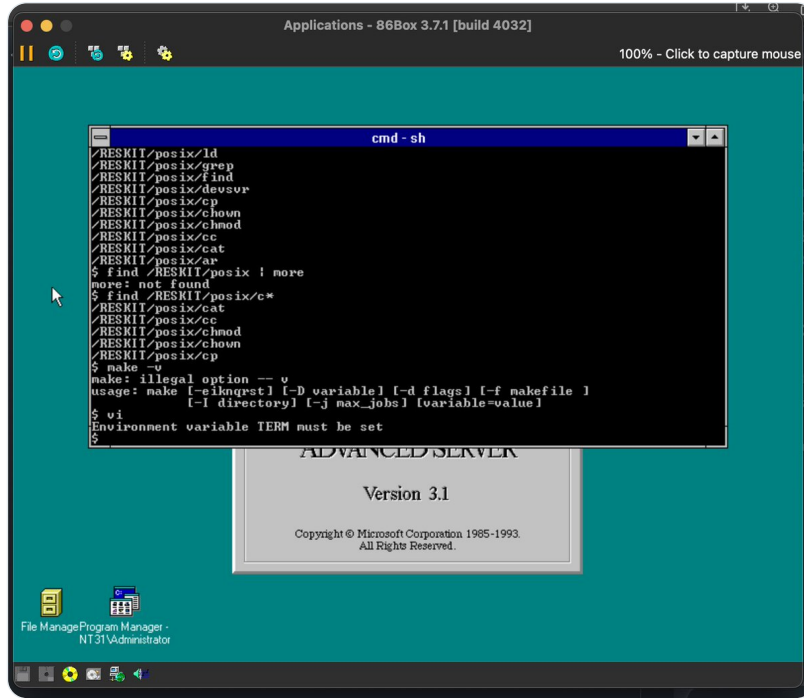


We also have make. It appears to be BSD make of some sort ...



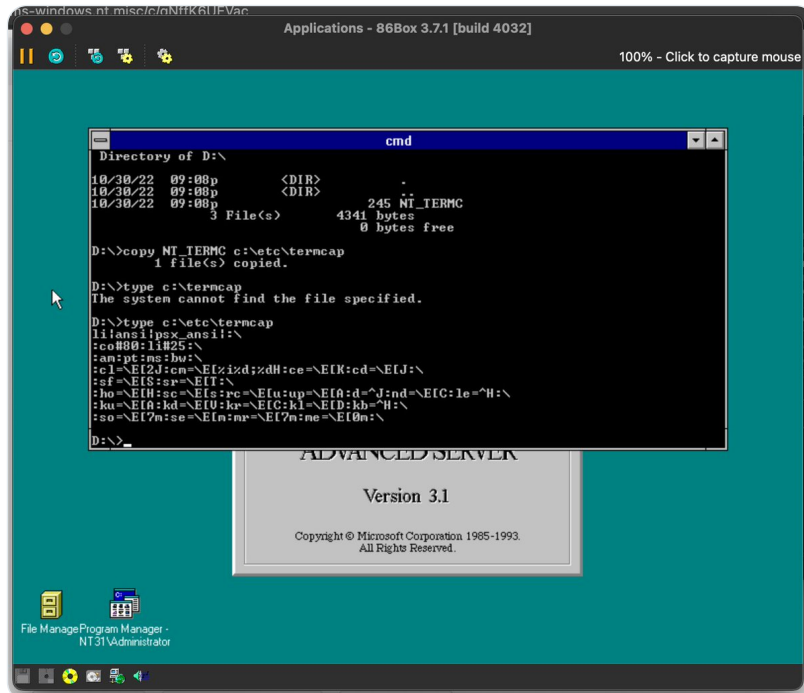
... so what's the TERM for Windows NT?

(I think I need to look at the termcap databae for this. Microsoft actually did ship it, I just need to look it up)

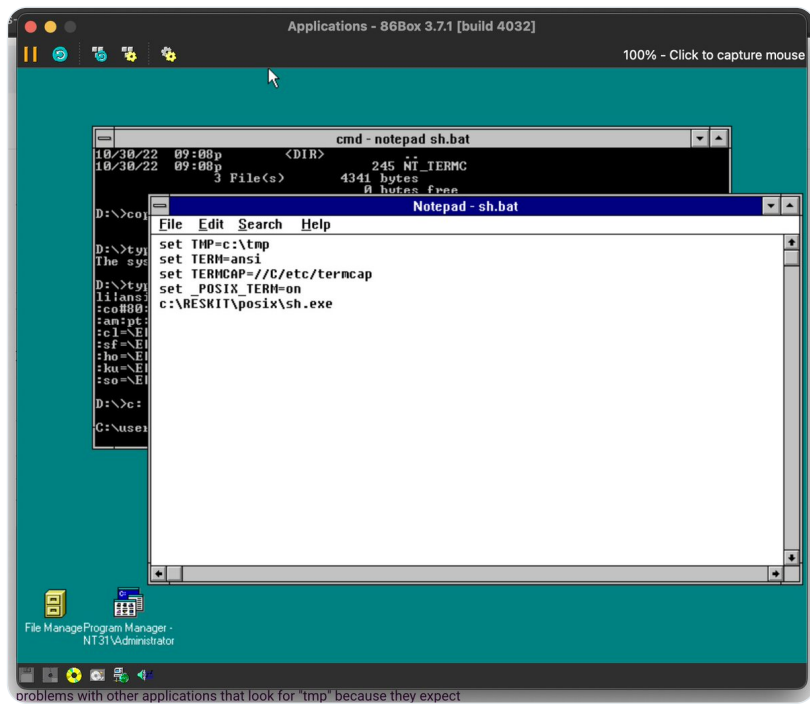


https://groups.google.com/g/comp.os.ms-windows.nt.misc/c/qNffK6UEVac/m/4Frg-Ms_FIYJ - omg, someone actually documented it. There's a knowledge base article (long gone), but someone in 1994 copy and pasted it FOR THIS MOMENT

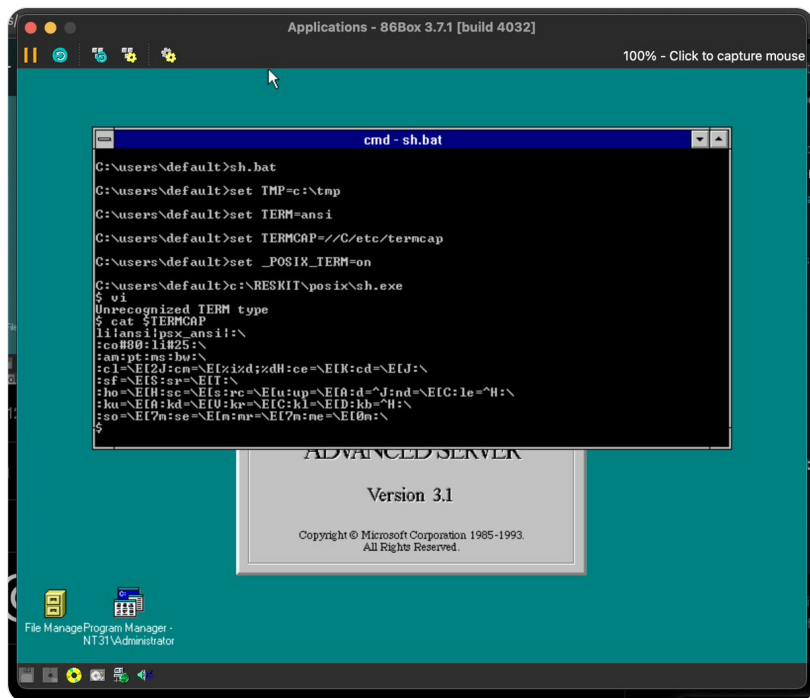
Ok, so that's the TERMCAP file setup, need to figure out the rest. I don't know if I want to set TMP, cause there are windows programs that care about that ...



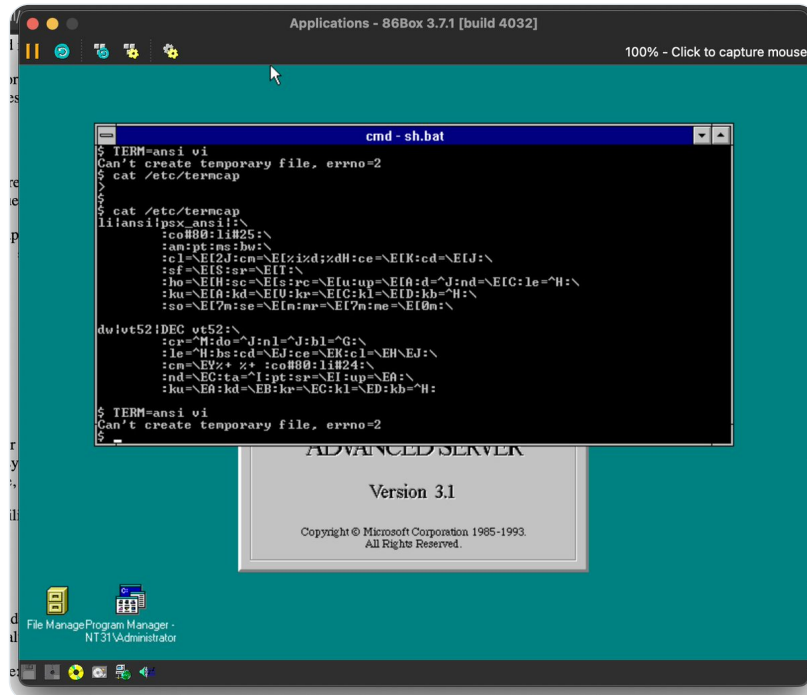
so maybe something like this?



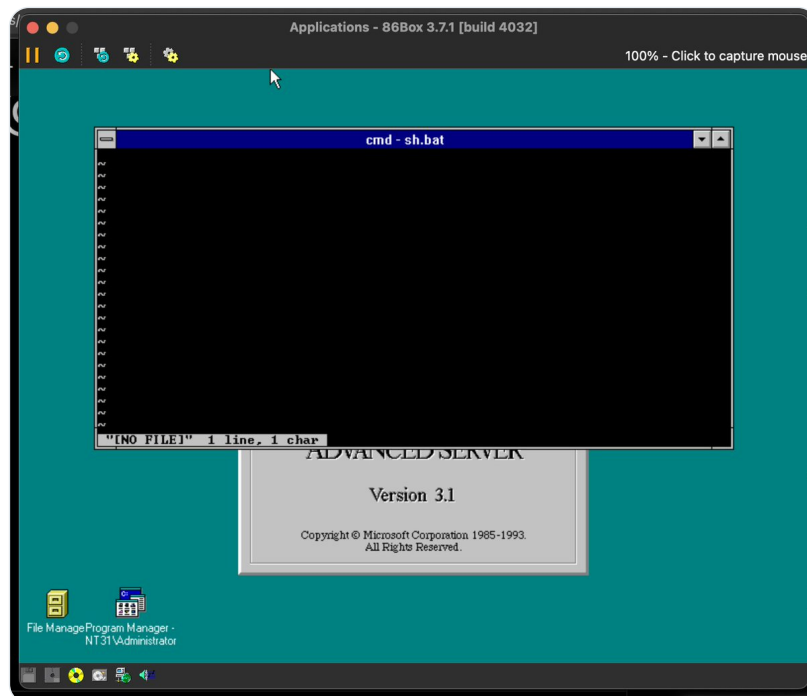
Not having a lot of luck ... I probably have the TERMCAP entry messed up though ...



It appears whitespace actually matters, more fiddling required ...

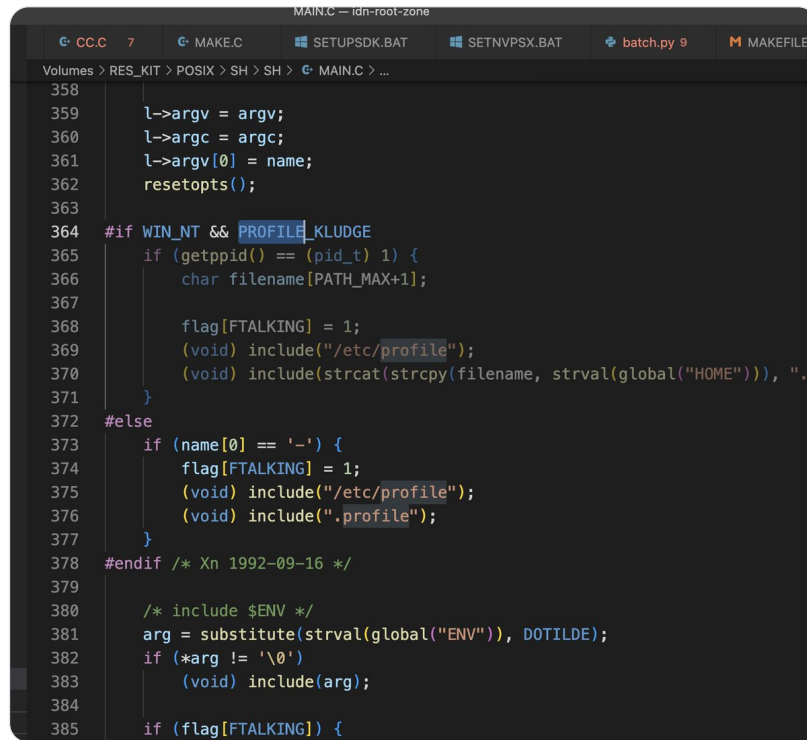


Oh *that's cool*. BSD vi running on NT on the POSIX subsystem.



So since Microsoft actually shipped the source code to KSH in the resource kit, I can actually see `/etc/profile` `*is loaded*`; that's the proper place to shove these variables.

Interesting the date code is from '92.

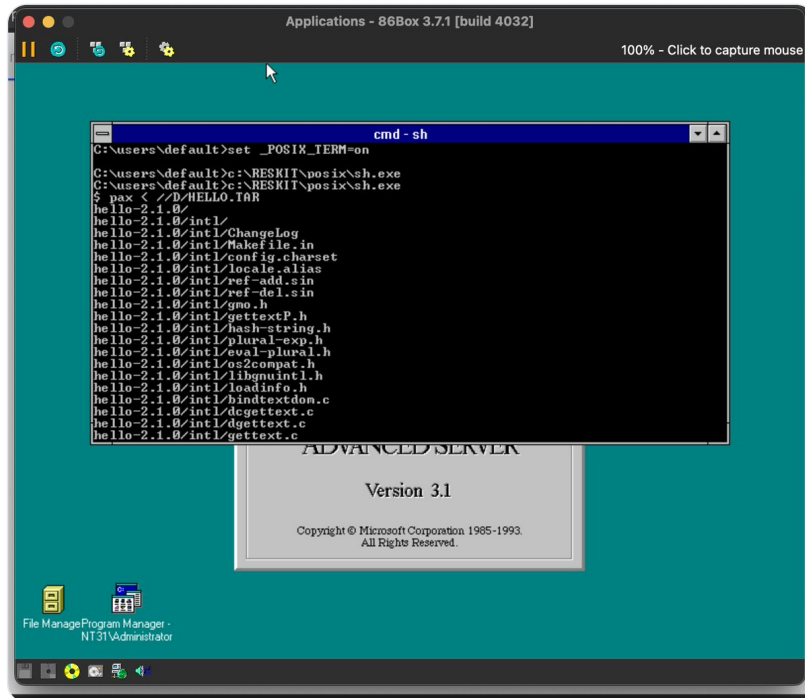


```
MAIN.C - idn-root-zone
CC.C 7 MAKE.C SETUPSDK.BAT SETNVPSX.BAT batch.py 9 MAKEFILE
Volumes > RES_KIT > POSIX > SH > SH > MAIN.C > ...
358
359     l->argv = argv;
360     l->argc = argc;
361     l->argv[0] = name;
362     resetopts();
363
364     #if WIN_NT && PROFILE_KLUDGE
365         if (getppid() == (pid_t) 1) {
366             char filename[PATH_MAX+1];
367
368             flag[FTALKING] = 1;
369             (void) include("/etc/profile");
370             (void) include(strcat(strcpy(filename, strval(global("HOME"))), ".p
371         }
372     #else
373         if (name[0] == '-') {
374             flag[FTALKING] = 1;
375             (void) include("/etc/profile");
376             (void) include(".profile");
377         }
378     #endif /* Xn 1992-09-16 */
379
380     /* include $ENV */
381     arg = substitute(strval(global("ENV")), DOTILDE);
382     if (*arg != '\0')
383         (void) include(arg);
384
385     if (flag[FTALKING]) {
```

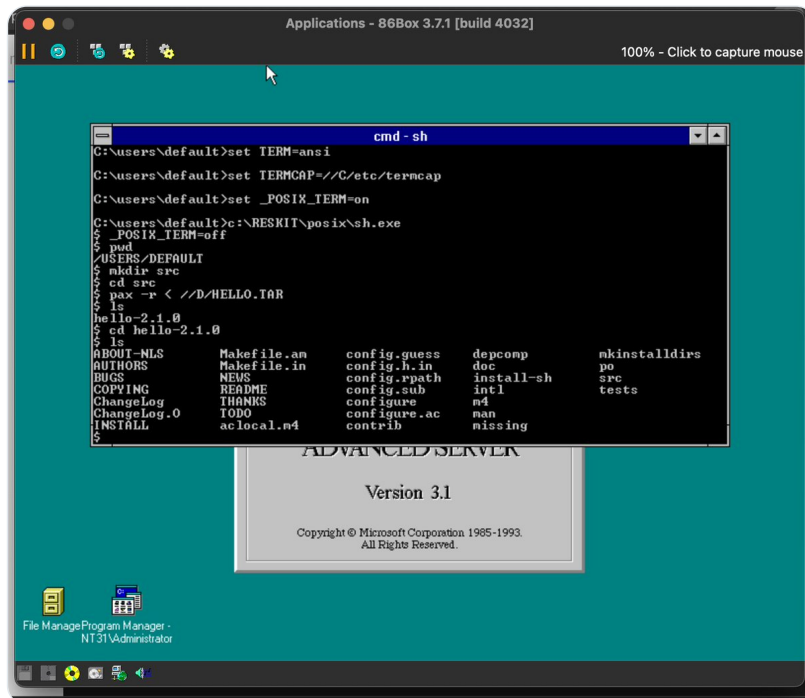
So I suppose I should install a toolchain but honestly, I have enough bits that ...
`*theoretically*` ... `configure` should run ...

It's still using `*MSVC 1.0*` as a compiler ... but I mean ... maybe?

I think this is the first time I've *ever* used pax for its intended purpose ...

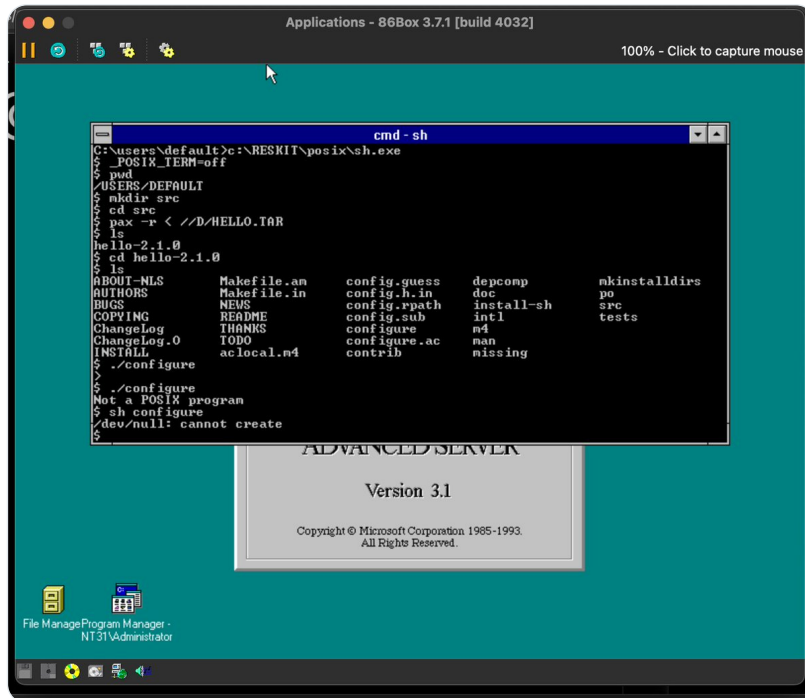


Fortunate favors the brave ...



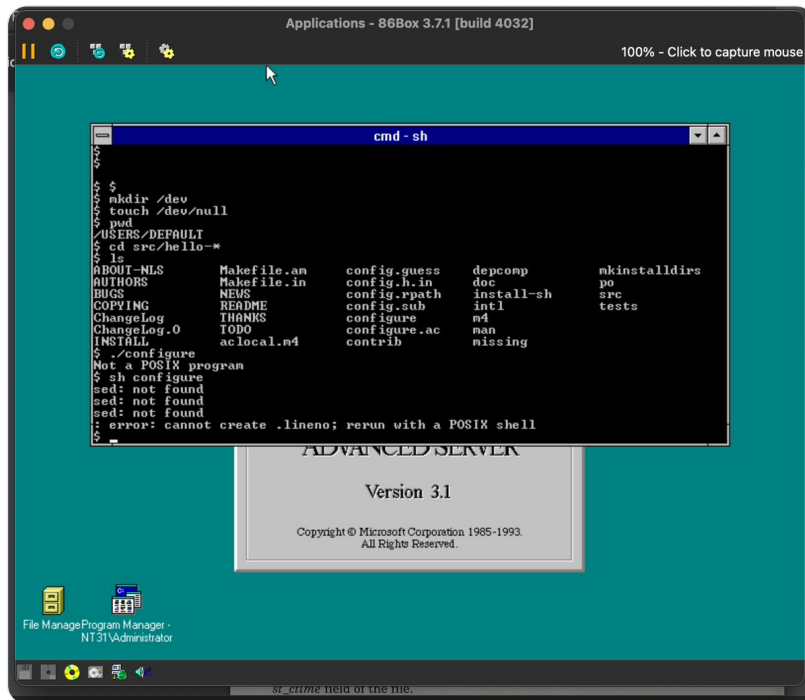
... *wow* ...

That's a failure mode *I didn't expect* ...

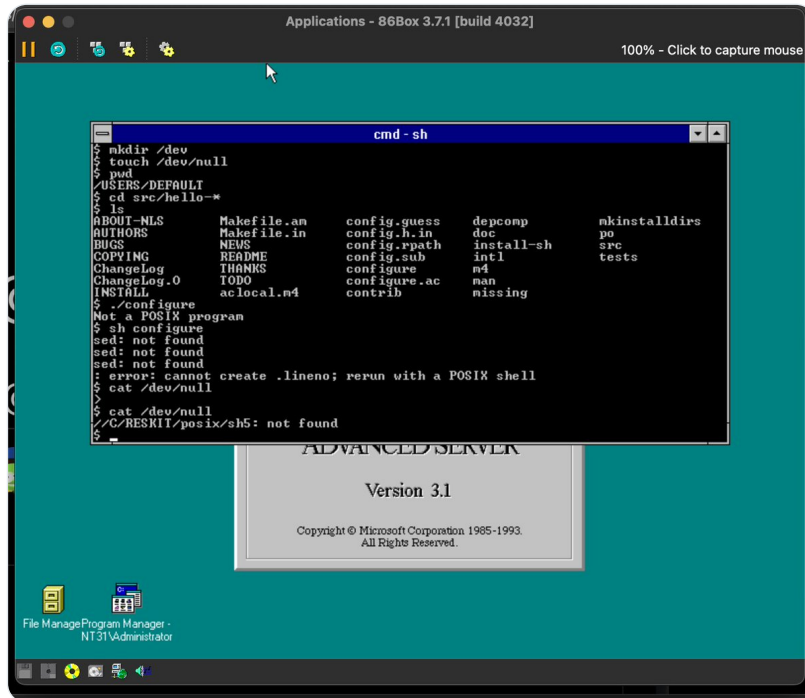


Trying to figure out if /dev/null is actually in the POSIX.1 standard or not ...

So I made a fake /dev/null, and well, it gets a bit further ...



Incidentally, I have successfully looked into the bit bin and it looked back ...



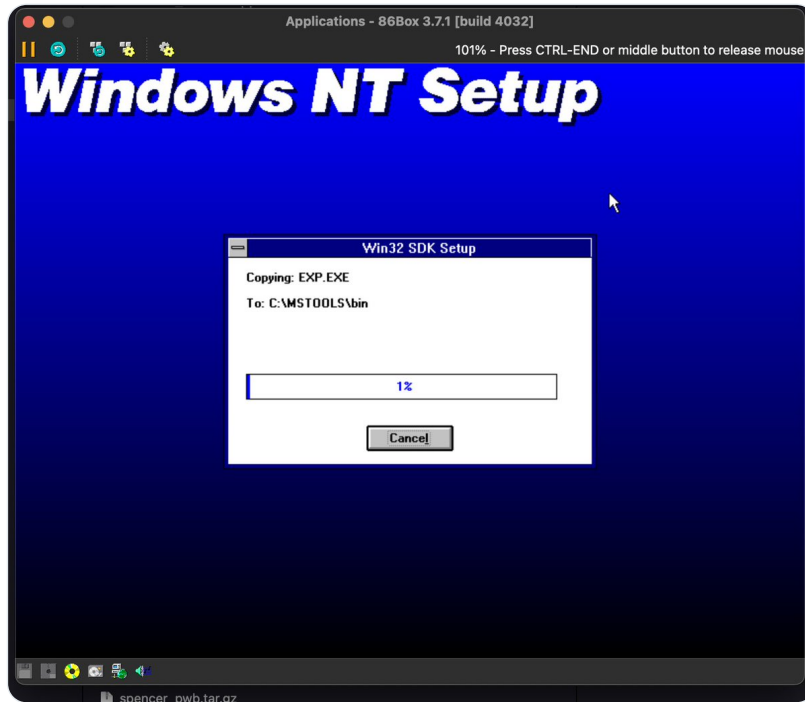
I'm debating right now if I am really ready to deal with the madness going further would require. There's seemingly nothing specifically stopping me from getting to the point I could run a configure script ...

That said, I'm thinking I might want to move to NT 4, better MSVC ...

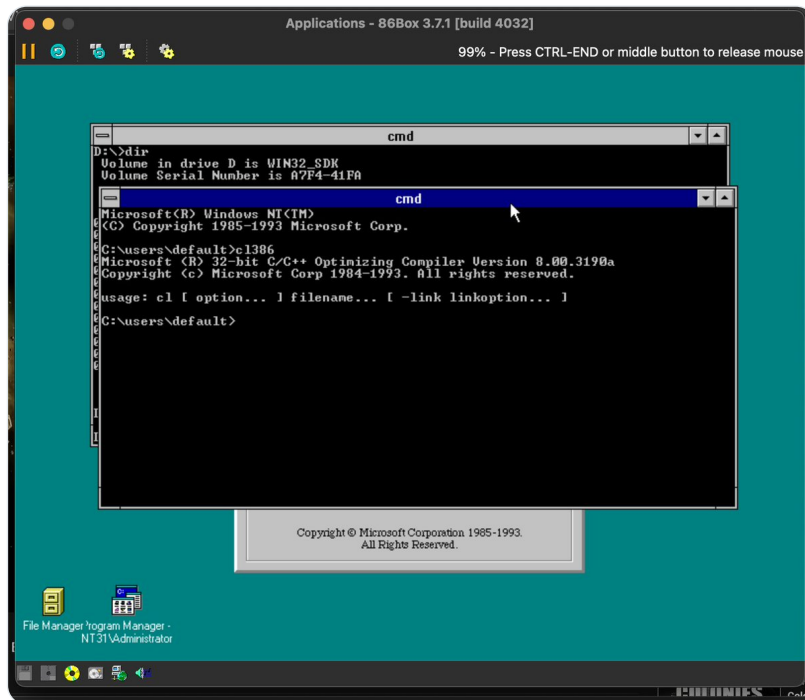
So GNU sed 1.16 from 1993 is actually surprisingly simple, and its configure script is very short. It's pretty much two files, I could hand compile this if needed. That probably would be enough?

Let me install the devtools, it's time :)

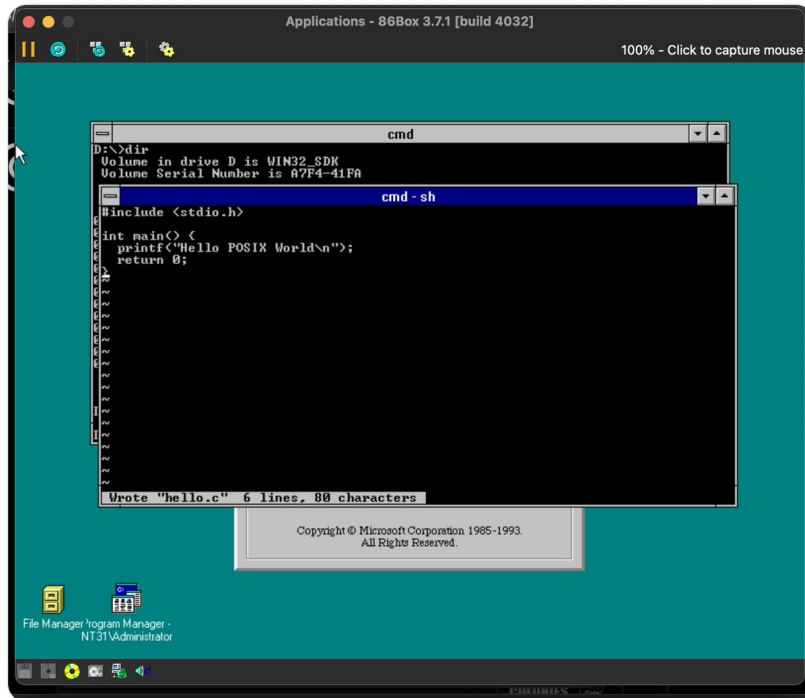
Installer goes brrr



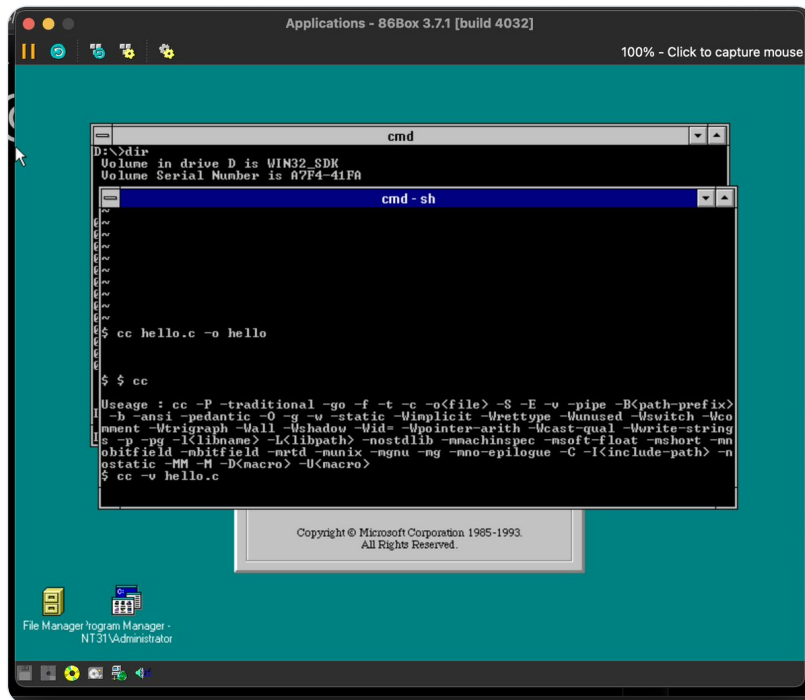
Let there be compiler ...



Hello world test ...



It seems to hang ...



So it appears I need to have DEVSVR running on the NT side to handle path conversion and shit. Of course its not documented at all, and the debug messages are commented out ...

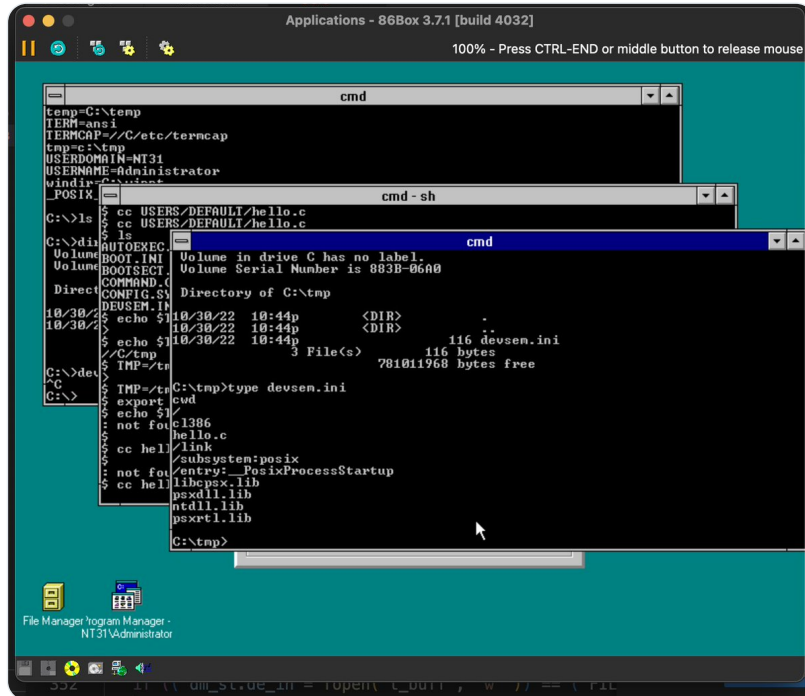
screaming intensifies

No wonder Microsoft just made makefiles for everything ...

How this works is *literally possessed*. the cc wrapper writes a file for devsvr, which is a daemon running on the NT side, runs it through cl386 and ...

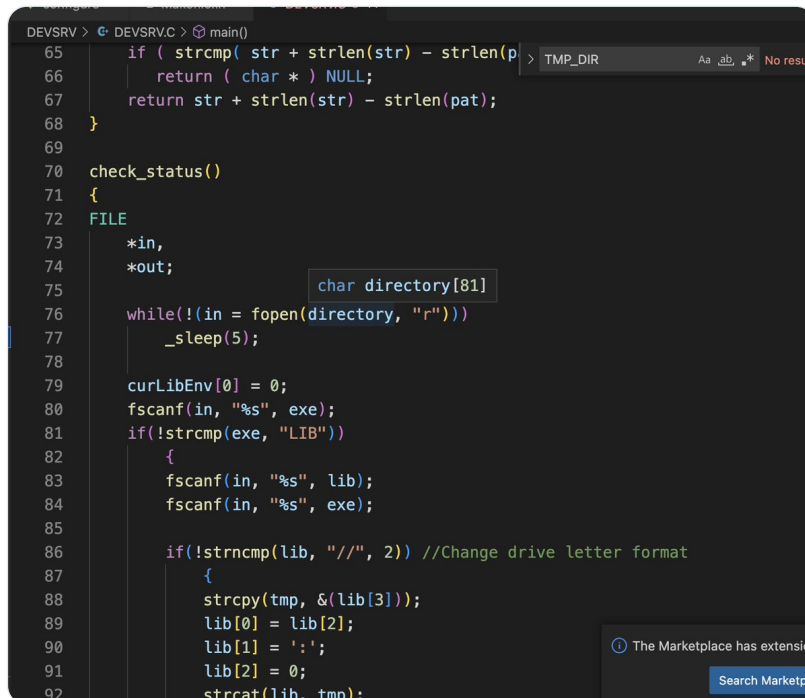
like ...

screaming intensifies



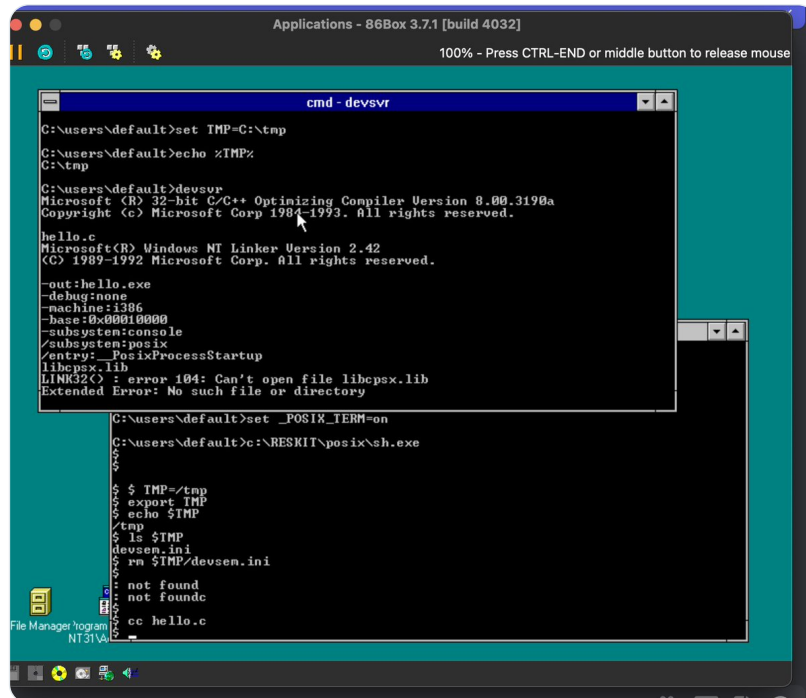
DEVSVR doesn't even do anything intelligent, it just waits every 5 seconds!

(I still can't get it to work but WTF)



Make sure your POSIX and NT TMP variables agree with each other, but this is ...

like *wow* ...



```
Applications - 86Box 3.71 [build 4032]
100% - Press CTRL-END or middle button to release mouse

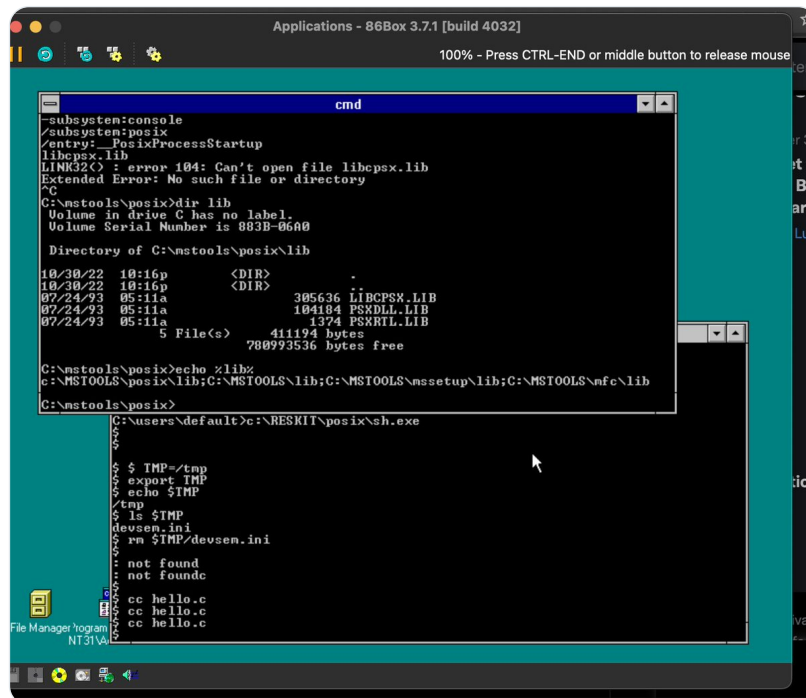
cmd - devsvr
C:\users\default>set TMP=C:\tmp
C:\users\default>echo %TMP%
C:\tmp
C:\users\default>devsvr
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

hello.c
Microsoft (R) Windows NT Linker Version 2.42
(C) 1989-1992 Microsoft Corp. All rights reserved.

-out:hello.exe
-debug:none
-machine:i386
-base:0x00010000
-subsystem:console
/subsystem:posix
/entry:__PosixProcessStartup
libpsx.lib
LINK32(>) : error 104: Can't open file libpsx.lib
Extended Error: No such file or directory

C:\users\default>set _POSIX_TERM=on
C:\users\default>c:\RESKIT\posix\sh.exe
$
$
$ TMP=/tmp
$ export TMP
$ echo $TMP
/tmp
$ ls $TMP
deven.ini
rm $TMP/deven.ini
$
$ : not found
$ : not found
$
$ cc hello.c
```

Well, the only problem is it can't find its runtime library now ... "progress" ...



```
Applications - 86Box 3.71 [build 4032]
100% - Press CTRL-END or middle button to release mouse

cmd
-subsystem:console
-subsystem:posix
/entry:__PosixProcessStartup
libpsx.lib
LINK32(>) : error 104: Can't open file libpsx.lib
Extended Error: No such file or directory
C
C:\nstools\posix>dir lib
Volume in drive C has no label.
Volume Serial Number is 883B-06A0

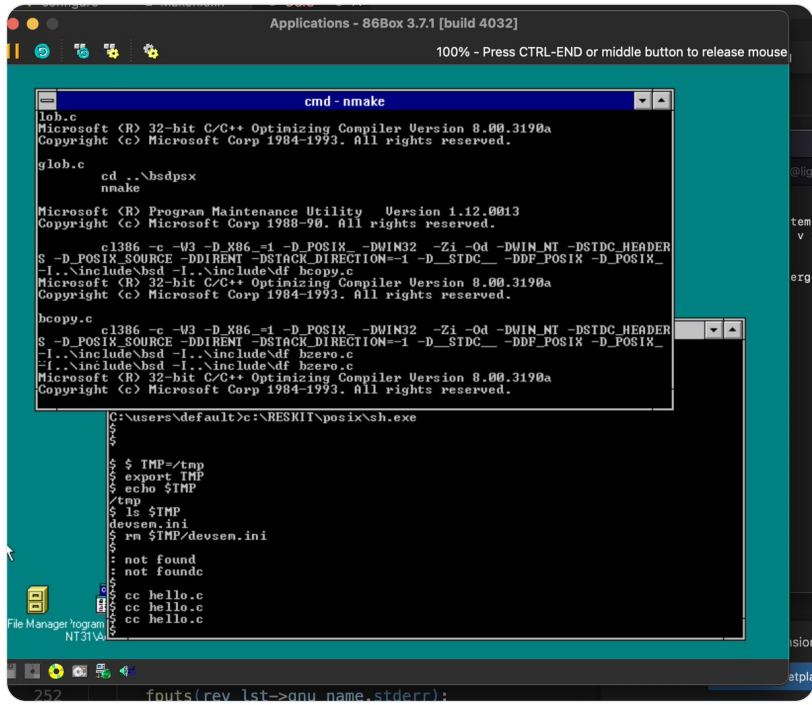
Directory of C:\nstools\posix\lib

10/30/22 10:16p <DIR> .
10/30/22 10:16p <DIR> ..
07/24/93 05:11a 385636 LIBCPSX.LIB
07/24/93 05:11a 104184 PSXDLL.LIB
07/24/93 05:11a 1374 PSXRLL.LIB
5 File(s) 411194 bytes
780993536 bytes free

C:\nstools\posix>echo %lib%
c:\MSTOOLS\posix\lib;c:\MSTOOLS\lib;c:\MSTOOLS\nssetup\lib;c:\MSTOOLS\nfc\lib
C:\nstools\posix>

C:\users\default>c:\RESKIT\posix\sh.exe
$
$
$ TMP=/tmp
$ export TMP
$ echo $TMP
/tmp
$ ls $TMP
deven.ini
rm $TMP/deven.ini
$
$ : not found
$ : not found
$
$ cc hello.c
$ cc hello.c
$ cc hello.c
```

So, I can build the POSIX environment from the Windows side properly, so I have MSTOOLS setup properly. Why is the devsvr wrapper unhappy ...



```
Applications - 86Box 3.71 [build 4032]
100% - Press CTRL-END or middle button to release mouse

cmd - nmake
glob.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

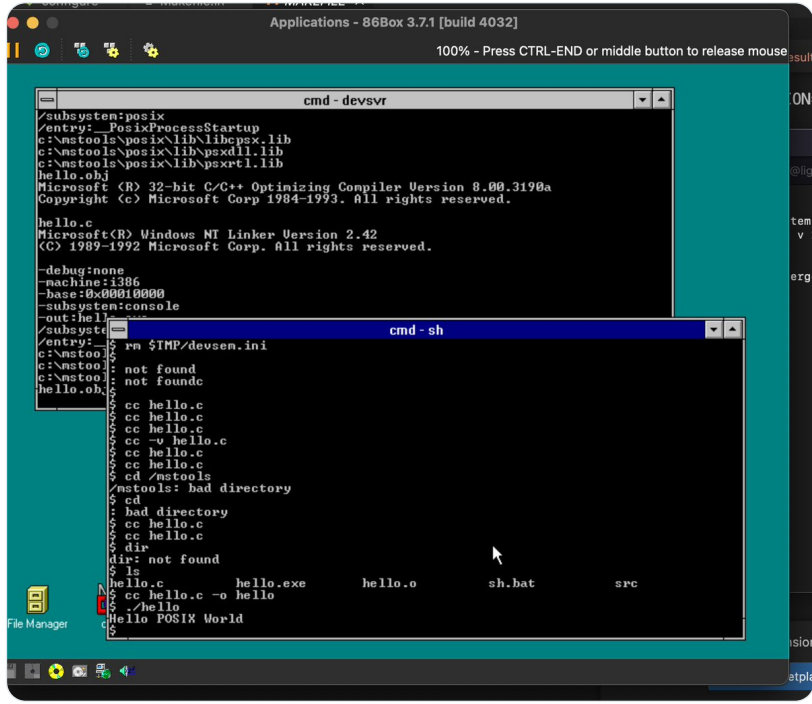
glob.c
cd ..\bsdpsx
nmake
Microsoft (R) Program Maintenance Utility Version 1.12.0013
Copyright (c) Microsoft Corp 1988-90. All rights reserved.

c1386 -c -W3 -D_X86_=-1 -D_POSIX_ -DWIN32 -Zi -Od -DWIN_NT -DSTDC_HEADER
-D_POSIX_SOURCE -DDIRENT -DSTACK_DIRECTION=-1 -D__STDC__ -DDF_POSIX -D_POSIX_
-I..\.include\bsd -I..\.include\df bcopy.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

bcopy.c
c1386 -c -W3 -D_X86_=-1 -D_POSIX_ -DWIN32 -Zi -Od -DWIN_NT -DSTDC_HEADER
-D_POSIX_SOURCE -DDIRENT -DSTACK_DIRECTION=-1 -D__STDC__ -DDF_POSIX -D_POSIX_
-I..\.include\bsd -I..\.include\df bzero.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

C:\users\default>c:\RESKIT\posix\sh.exe
$
$
$ TMP=/tmp
$ export TMP
$ echo $TMP
/tmp
$ ls $TMP
devsen.ini
rm $TMP/devsen.ini
:
: not found
:
: not foundc
cc hello.c
cc hello.c
cc hello.c
fputs(rev lst->gnu_name,stderr);
```

That required code patches *and far too much effort* ...



```
Applications - 86Box 3.71 [build 4032]
100% - Press CTRL-END or middle button to release mouse

cmd - devsvr
/subsystem:posix
/entry:_PosixProcessStartup
c:\mstools\posix\lib\libcpxx.lib
c:\mstools\posix\lib\psxdll.lib
c:\mstools\posix\lib\psxrt1.lib
hello.obj
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

hello.c
Microsoft (R) Windows NT Linker Version 2.42
(C) 1989-1992 Microsoft Corp. All rights reserved.

-debug:none
-machine:i386
-base:0x0010000
-subsystem:console
-out:hell

/subsystem:
/entry:
c:\mstoo
c:\mstoo
c:\mstoo
hello.obj
cc hello.c
cc hello.c
cc -v hello.c
cc hello.c
cc hello.c
cd /mstools
/mstools: bad directory
cd
bad directory
cc hello.c
cc hello.c
dir
dir: not found
$ ls
hello.c hello.exe hello.o sh.bat src
cc hello.c -o hello
./hello
Hello POSIX World
```

Actually building real software is still a challenge ...

```
Applications - 86Box 3.71 [build 4032] 100% - Click to capture mouse
cmd - devsvr
/subsystem:posix
/entry: PosixProcessStartup
c:\msotools\posix\lib\libcpx.lib
c:\msotools\posix\lib\sedll.lib
c:\msotools\posix\lib\psxrtl.lib
hello.obj
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

sed.c
sed.c(43) : fatal error C1083: Cannot open include file: 'strings.h': No such file or directory

utils.c
utils.c(31) : fatal error C1083: Cannot open include file: 'strings.h': No such file or directory

regex.c
regex.c(63) : fatal error C1083: Cannot open include file: 'strings.h': No such file or directory

getopt.c
getopt1.c
alloca.c

$ cd
$ cd directory
$ dir
dir: not found
$ ls
hello.c      hello.exe    hello.o      sh.bat      src
$ cc hello.c -o hello
$ ./hello
Hello POSIX World
$ pwd
.

$ cd
$ cd /USERS/DEFAULT
$ cd src/sed-1.8
$ cc sed.c utils.c regex.c getopt.c getopt1.c alloca.c -o sed
```

Wait, I think I realized the problem, I think I need the environment variables set on the sh terminal, not in devsvr ...

With *a lot of pain*, I managed to build sed ...

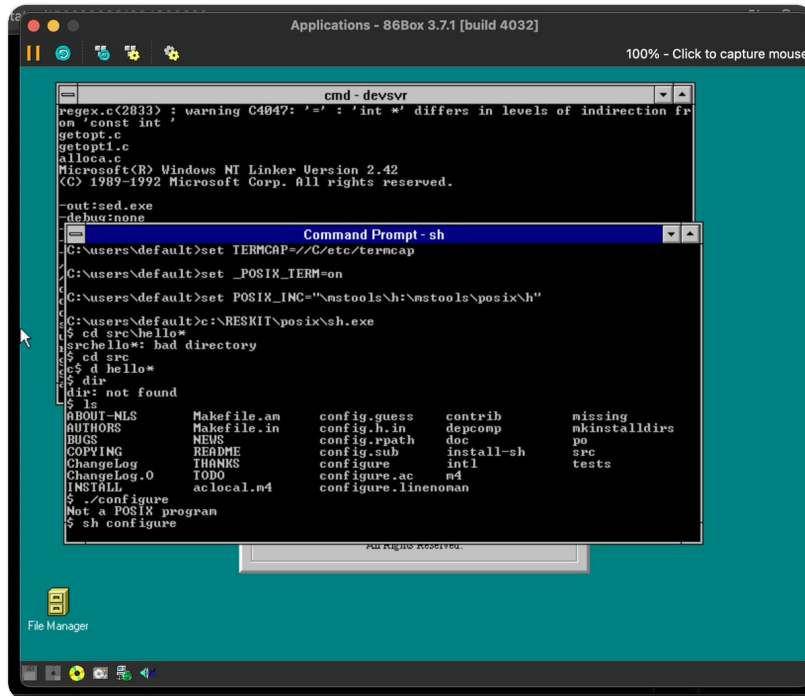
```
Applications - 86Box 3.71 [build 4032] 100% - Click to capture mouse
cmd - devsvr
regex.c(2833) : warning C4047: 'int *' : 'int *' differs in levels of indirection from 'const int'
getopt.c
getopt1.c
alloca.c
Makefile
Makefile.in
configure.in
configure
sed.o
utils.o
regex.o
sed.exe
Usage: ./sed [-n] [-quiet] [--silent] [--version] [-e script]
[-f script-file] [--expression=script] [--file=script-file] [file...]

environment variable TERM must be set
C:\users\default\src\sed-1.8>TERM=ansi
The name specified is not recognized as an
internal or external command, operable program or batch file.
C:\users\default\src\sed-1.8>sh

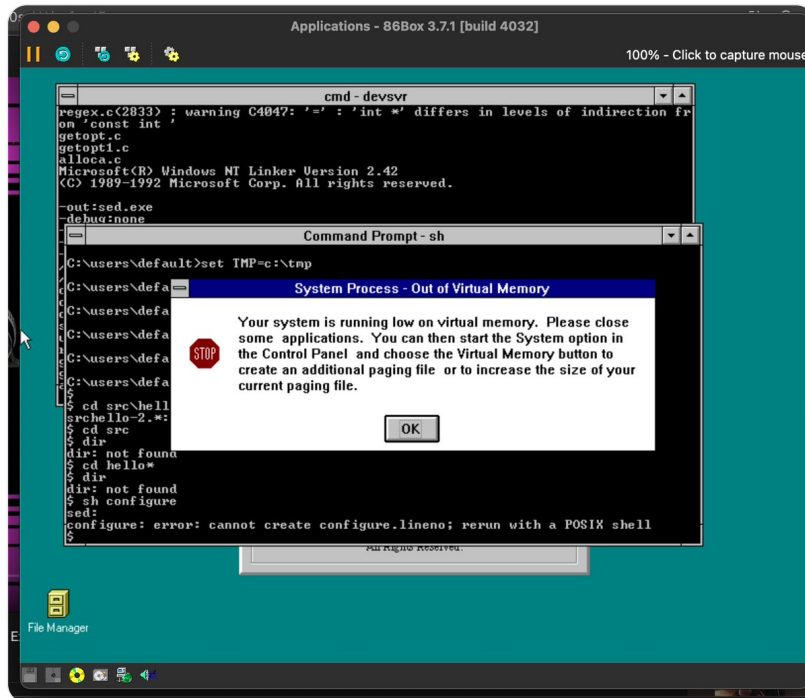
All Rights Reserved.

rm -f sed *.o core
```

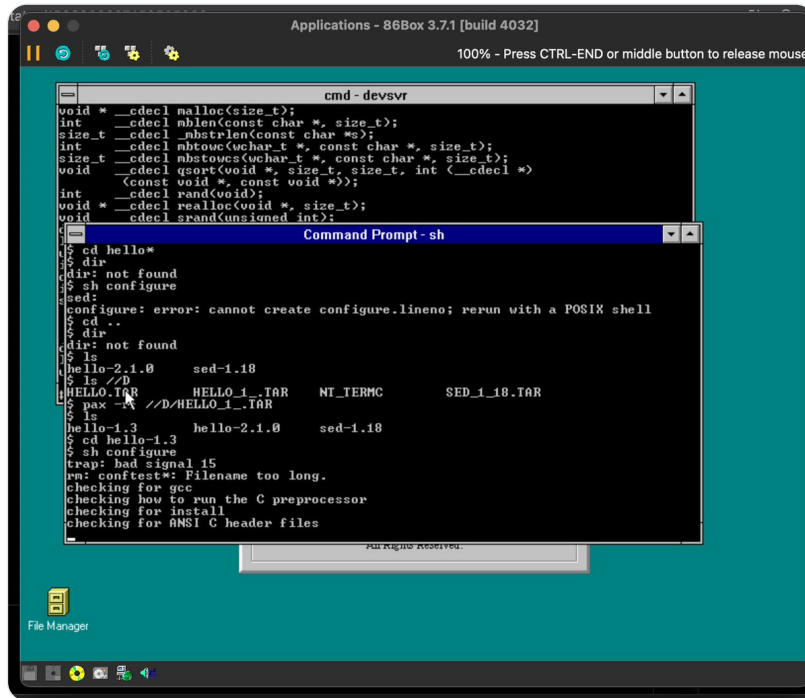
Well running a configure script just hangs now, and terminal output is so slow with -x its impressively bad ...



So I let configure just run for awhile and well ...



Ok, just for shits and giggles, I tried a much older version of "hello" from 1993, and shockingly, the configure script did start ...



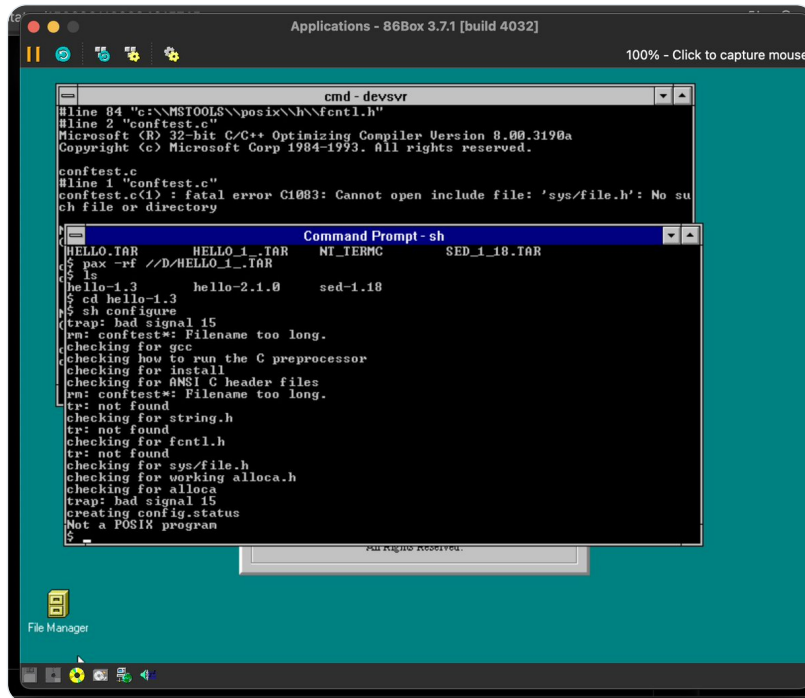
```
Applications - 86Box 3.7.1 [build 4032]
100% - Press CTRL-END or middle button to release mouse

cmd - devsvr
void * __cdecl malloc(size_t);
int __cdecl strlen(const char *, size_t);
size_t __cdecl mbstrlen(const char *);
int __cdecl mbtous(uchar_t *, const char *, size_t);
size_t __cdecl mbstows(uchar_t *, const char *, size_t);
void __cdecl qsort(void *, size_t, size_t, int (__cdecl *)
(const void *, const void *));
int __cdecl rand(void);
void * __cdecl realloc(void *, size_t);
void __cdecl srand(unsigned int);

Command Prompt - sh
$ cd hello*
$ dir
dir: not found
$ sh configure
configure: error: cannot create configure.lineno; rerun with a POSIX shell
$ cd ..
$ dir
dir: not found
$ ls
hello-2.1.0 sed-1.18
$ ls /D
HELLO.TAR HELLO_1_TAR NT_TERM SED_1_18.TAR
$ pax -f //D/HELLO_1_TAR
$ ls
hello-1.3 hello-2.1.0 sed-1.18
$ cd hello-1.3
$ sh configure
trap: bad signal 15
rn: conftest*: Filename too long.
checking for gcc
checking how to run the C preprocessor
checking for install
checking for ANSI C header files

File Manager
```

It didn't get far though ... by and large, we're not going to be running much of anything >.>:



```
Applications - 86Box 3.7.1 [build 4032]
100% - Click to capture mouse

cmd - devsvr
#line 84 "c:\msi\tools\posix\h\fcntl.h"
#line 2 "conftest.c"
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 8.00.3190a
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.

conftest.c
#line 1 "conftest.c"
conftest.c(1) : fatal error C1083: Cannot open include file: 'sys/file.h': No su
ch file or directory

Command Prompt - sh
HELLO.TAR HELLO_1_TAR NT_TERM SED_1_18.TAR
$ pax -f //D/HELLO_1_TAR
$ ls
hello-1.3 hello-2.1.0 sed-1.18
$ cd hello-1.3
$ sh configure
trap: bad signal 15
rn: conftest*: Filename too long.
checking for gcc
checking how to run the C preprocessor
checking for install
checking for ANSI C header files
rn: conftest*: Filename too long.
tr: not found
checking for string.h
tr: not found
checking for fcntl.h
tr: not found
checking for sys/file.h
checking for working alloca.h
checking for alloca
trap: bad signal 15
creating config.status
Not a POSIX program
$

File Manager
```

How I summed this up in the script thus far "After doing so, you'll end up with something semi-resembling a UNIX system."

Feelings on just getting it working:

"I wish to say it gets better but we're still got a way to go before we hit peak abomination, since ... sigh ... I need to show you how to compile stuff with it. After all, that was the entire point."

Script just broke the 9 page mark. On average, each page is about 3 minutes of VO, so ... yeah, this is going to be *a video* ...

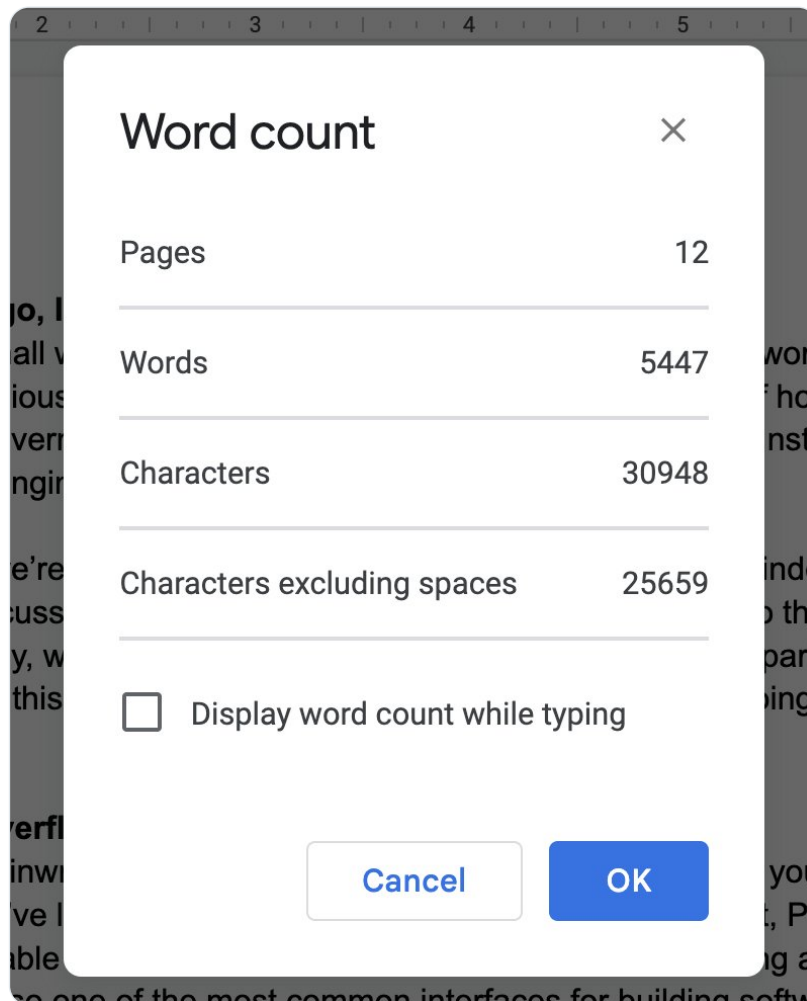
I think I found my vibe for the next two sections:

"What you're about to see is an exercise in spite ..."

Yeah, I'm def. throwing some daggers here

"[...] it's not only possible to implement POSIX support on top of the Win32 programming interface, it works far better than anything Microsoft shipped. Let me introduce you all to Cygwin."

We're probably at about 40 minutes of runtime, but I still need to add two sections, and I need to do the "project" part, which is likely going to be trying to get rogue or nethack running on both this, NT/mips, and NT/powerpc.



Ok, so once again, I've been at this for another 8 hours, so its time to do something else. If you've enjoyed this content, feel free to drop something in the tip jar: ko-fi.com/ncommander

• • •