User Mode Linux, VMWare and Wine

Virtual Machines Under Linux

Brad Marshall

bmarshal@pisoftware.com

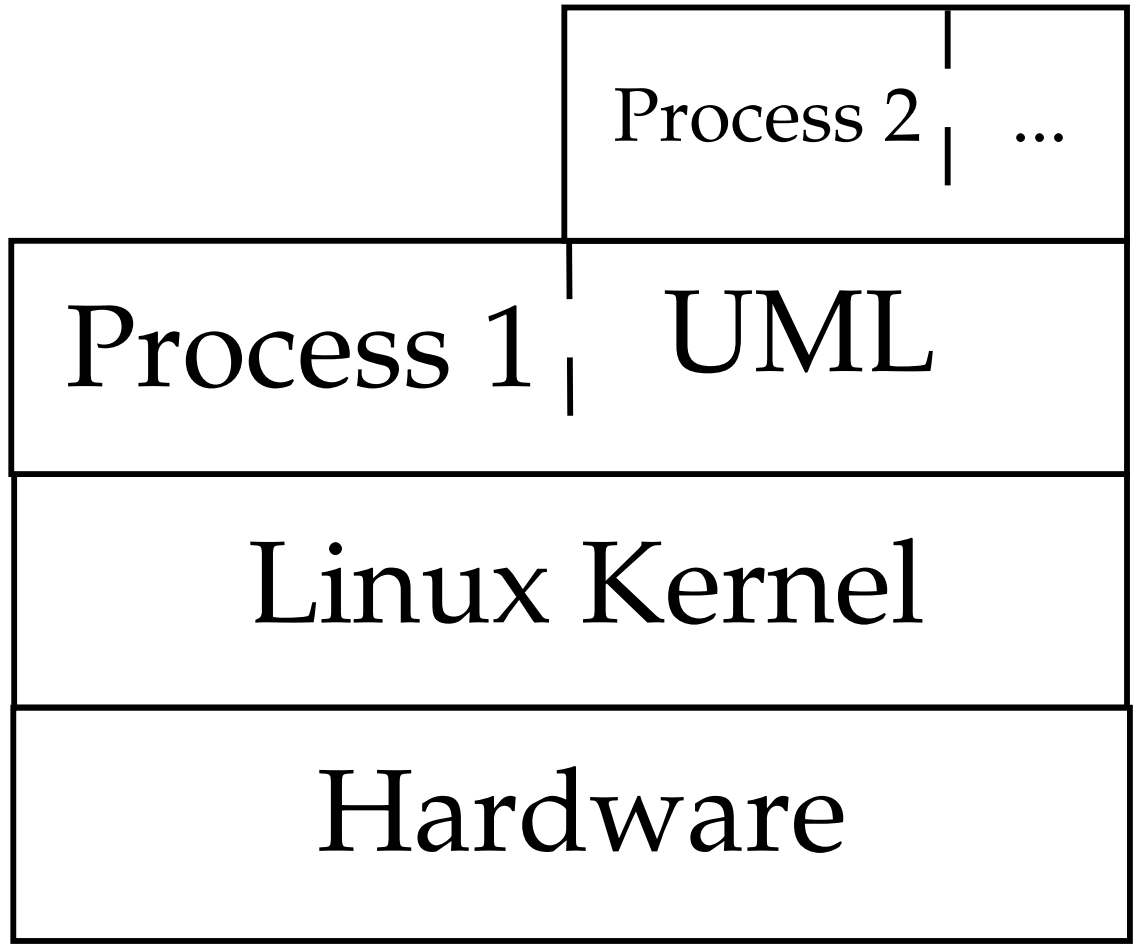
Plugged In Software

Contents

What is UML?

- Port of linux kernel to linux system calls, rather than hardware
- Gives a virtual OS - no machine emulation layer
- Disk storage is done via files on host system
- Can control what hardware the virtual machine can access
- Won’t damage real computers hardware or software
- Runs own scheduler and VM
- UML kernel and processes run as processes on host kernel
- Can run as any user

UML Architecture



What use is UML?

- Safe way of running Linux distributions
- Kernel development and debugging
- Process debugging
- Learning
- Secure sandbox / jail
- Honeypots
- Test environment
- Disaster recovery practice
- ISP's virtual hosting
- .... whatever!

What hardware can UML support?

- Block devices
- Consoles and serial lines
- Network devices
- SCSI devices
- USB devices
- Sound cards
- PCI hardware (in progress)

Installing UML

- Host machine needs 2.2.15 or 2.3.22 (or later)
- Patch available for older kernels
- Minimum needed:
  - UML kernel
  - Root filesystem to boot
- Optional host kernel patch for skas (Separate Kernel Address Space)

UML Utilities

- uml_moo - merge COW file with its backing file
- uml_mconsole - attach to UML management console
- uml_switch - switch daemon
- uml_net - setuid helper for network setup
- tunctl - create and control persistent TUN/TAP interfaces

Networking

- Transports can provide network to:
  - Local host
  - Other machines on local net
  - Rest of the internet
- Done using one of the available transports:
  - Exchange packets with host
    - ethertap
    - TUN/TAP
    - slip
    - slirp
    - pcap
  - Virtual network
    - Multicast
    - switch daemon
- Uses uml_net setuid helper for configuration

Choosing a transport

- ethertap
  - if you want access to the host networking and it is running 2.2
- TUN/TAP
  - if you want access to the host networking and it is running 2.4. Can use a preconfigured device, which doesn't require uml_net
- Multicast
  - if you want a purely virtual network and you don't want to set up anything but the UML
- a switch daemon
  - if you want a purely virtual network and you don't mind running the daemon in order to get somewhat better performance

Choosing a transport cont

- slip
  - there is no particular reason to run the slip backend unless ethertap and TUN/TAP are just not available for some reason
- slirp
  - if you don’t have root access on the host to setup networking, or if you don’t want to allocate an IP to your UML
- pcap
  - not much use for actual network connectivity, but great for monitoring traffic on the host

UML Networking Kernel Boot Args

- General format
  - eth<n>=<transport>,<transport args>
- Multicast
  - eth<n>=mcast
- TAP/TUN
  - eth<n>=tuntap„,<IP address>
- Ethertap
  - eth<n>=ethertap,<device>,<ethernet address>,<tap IP address>

UML Networking Kernel Boot Args

- Switch daemon
  - eth<n>=daemon,<ethernet address>,<socket type>,<control socket>,<data socket>
- Slip
  - eth<n>=slip,<slip IP>
- Slirp
  - eth<n>=slirp,<ethernet address>,<slirp path>
- Pcap
  - eth<n>=pcap,<host interface>,<filter expression>,<option1>,<option2>

Sharing filesystems between UMLs

- Uses copy-on-write layering in the ubd block device
- Layers a private read-write device over a shared read-only device
- Useful when using lots of virtual machines - saves lots of disk space
- Writes are done to private device, reads from either
- Do not boot directly from read-only backing files - will invalidate any COW files that use it

Creating COW files

- To create COW file, boot with
  - ubd0=root_fs_cow,root_fs_debian_22
- After creation, only need:
  - ubd0=root_fs_cow
- Name of backing file is stored in COW file header
- To merge a COW file and backing file:
  - uml_moo <COW file> <new backing file>

Host File Access

- hostfs allows mounting of files from host filesystem
- Check hostfs is available on virtual machine by looking at /proc/filesystem
- Mount it by:
  - mount none /mnt/host -t hostfs
- If you want to mount a subdirectory:
  - mount none /mnt/host -t hostfs -o /path

Serial Lines and Consoles

- Can attach serial lines and consoles to a variety of host I/O channels
  - ptys
  - ttys
  - file descriptors
  - ports
- Done via a command line option, of format <device>=<channel>
- Consoles use device con, serial lines use ssl
- Use device number to talk about specific device, without specifies all

Serial Lines and Consoles Channels

- Pseudo terminals - `<device>=pts`
- Terminals - `<device>=tty:<tty device>`
- Xterms - `<device>=xterm`
- Port - `<device>=port:<port no>`
- File descriptors - `<device>=<fd>`
- Nothing - `<device>=null`
- None - `<device>=none`

Specify different input and output channels by putting a comma between them

Management Console

- Low level interface to kernel, like SysRq
- Allows you to:
  - get the kernel version
  - add and remove devices
  - halt or reboot the machine
  - send SysRq commands
  - pause and resume the UML
  - make online backups without shutting down the UML
  - receive notifications of events of interest from within UML
- Needs uml_console (part of uml utilities) and `CONFIG_MCONSOLE`

Management Console Usage

- When booting UML, there will be a line like:

```
mconsole (version 2) initialized on
     /home/brad/.uml/4UUEHn/mconsole
```

- Can specify unique machine id passing `umid=debian`
- Attach by calling uml_console with mconsole socket or umid

```
$ uml_console debian
```

Mconsole Commands

- version
  - Prints UML version
- halt and reboot
  - Shuts down machine instantly
- config
  - Adds a new device or queries config of existing one
- remove
  - Removes device from system

Mconsole Commands cont

- sysrq
  - Takes one letter argument, calls kernel's SysRq driver
- help
  - Gives help
- cad
  - Calls Control-Alt-Delete on UML instance
- stop and go
  - Pauses until go is run

Tracing Thread Mode

- Each UML process is also a process on host
- Tracing thread that does system call tracing on UML processes
- Tracing thread nullified system calls, caused process to enter UML kernel (mapped to upper part of address space)
- Problems:
  - UML kernel is present in address space of its processes, and by default is writeable
  - UML's jail fixes this by making it read-only, but at a performance cost
  - Kernel can still be read and found out that it is a UML
  - UML uses signals to send control to UML kernel during system call or interrupt

Seperate Kernel Address Space Mode

- UML kernel runs in diff host address space from processes
- Address space is identical to what it would be on the host
- Requires kernel patch on host
- On virtual machine, make sure you have `CONFIG_MODE_SKAS`
- Will fall back to TT mode if host doesn't have support
- On bootup, will see:

```
Checking for the skas3 patch in the host...found
Checking for /proc/mm...found
```

Running X

- Run X clients on host X server
  - Setup network as normal
  - Set the display to the host's X server then run clients as normal

```
$ export DISPLAY=host-ip:0
```

- Run a local virtual X server
  - Setup networking as normal
  - Run Xnest
  - Set the display to Xnest, then run clients as normal

```
$ export DISPLAY=:0
```

Compiling a UML kernel

```
$ tar xvfj linux-2.4.20.tar.bz2
$ cd /path/to/kernel
$ zcat uml-patch.gz | patch -p1
$ make config ARCH=um
$ make linux ARCH=um
$ make modules ARCH=um
$ sudo mount /path/to/root_fs \
      /path/to/mnt -o loop
$ make modules_install \
      INSTALL_MOD_PATH=/path/to/mnt/ \
      ARCH=um
```

Building a UML filesystem

- mkrootfs - command line utility for building multiple filesystems
- UML Builder - Step by step UML filesystem builder
- gBootRoot - GUI app for creating UML filesystems and boot disks
- rootstrap - Debian filesystem creation util

UML Honeypot

- Allows the use of one host for the honeypot
- Traffic logged from that host via iptables
- Intruder can get root on UML without endangering host
- Kernel option, honeypot, to rearrange address space to allow stack smash exploits to work
- Kernel option, jail, protects kernel memory from processes
- Useful UML features for honeypots
  - tty logging - logs all tty traffic out to host
  - hppfs - HoneyPot ProcFS - allows /proc to be modified
  - skas mode - process address space is identical to host

UML Future

- SMP
  - Emulate more processes than host has
- Clustering
  - Run UML across multiple hosts (and multiple OSes)
- Security
  - Honeypot
  - Jailed services
  - Sandbox
- hostfs extensions
  - Access to local and remote filesystems (rsync, ssh)
  - Mount databases as filesystem

UML Future cont

- Ports to other OSes
  - Windows
  - Non-i386 linux
- Embed UML into applications
  - Mount application as filesystem, modify it via fs
  - See connections as processes
- Providing colocation
  - Already happening - some ISPs providing services using UML

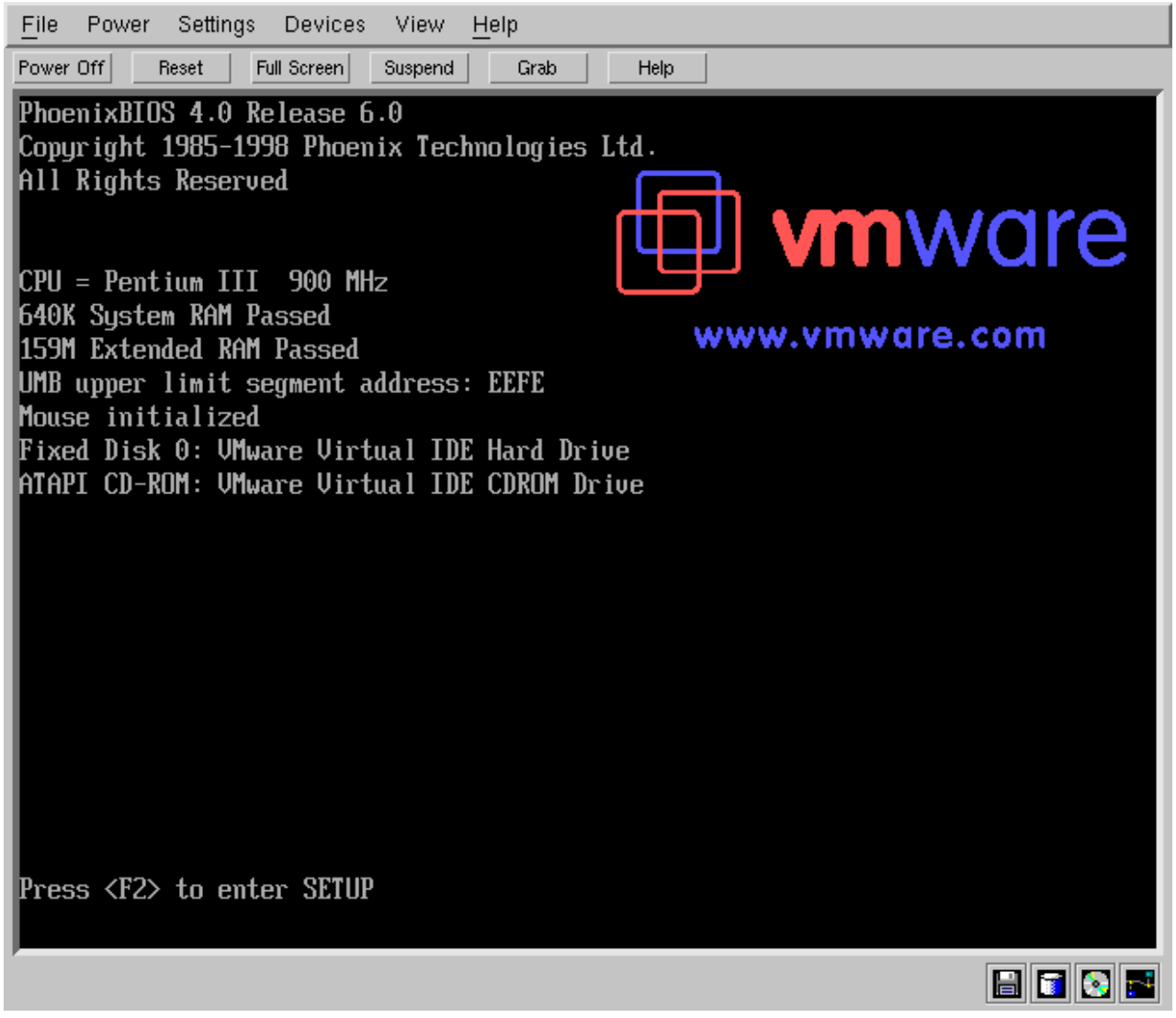
UML Conclusion

- Implementing UML has shown up bugs in Linux kernel
- Running applications in UML makes debugging easier
- Good for securely hosting services
- Can give Linux the ability to move into areas it wasn't
- Ability to run virtual machines can give good test environments
- Still in early stages of development

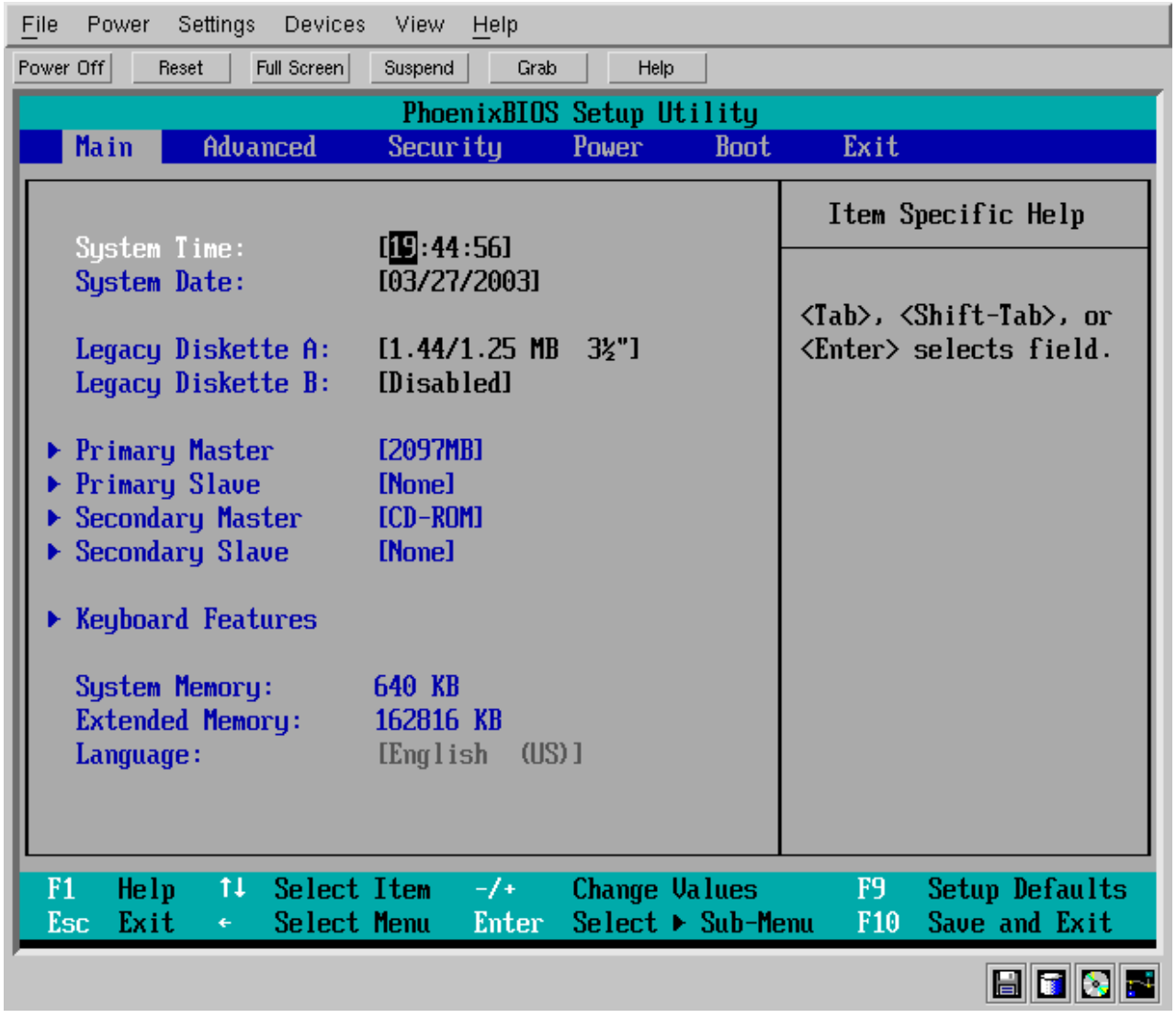
VMWare

- Provides a virtual machine to install an OS on
- 3 main products
  - Workstation
    - Run multiple OSes on the desktop
  - GSX Server
    - For providing virtual servers on enterprise hardware
    - Provides support for clustering of virtual machines
  - ESX Server
    - Has own operating system
    - Guarenteed service levels for virtual machines
    - Provides remote management functionality

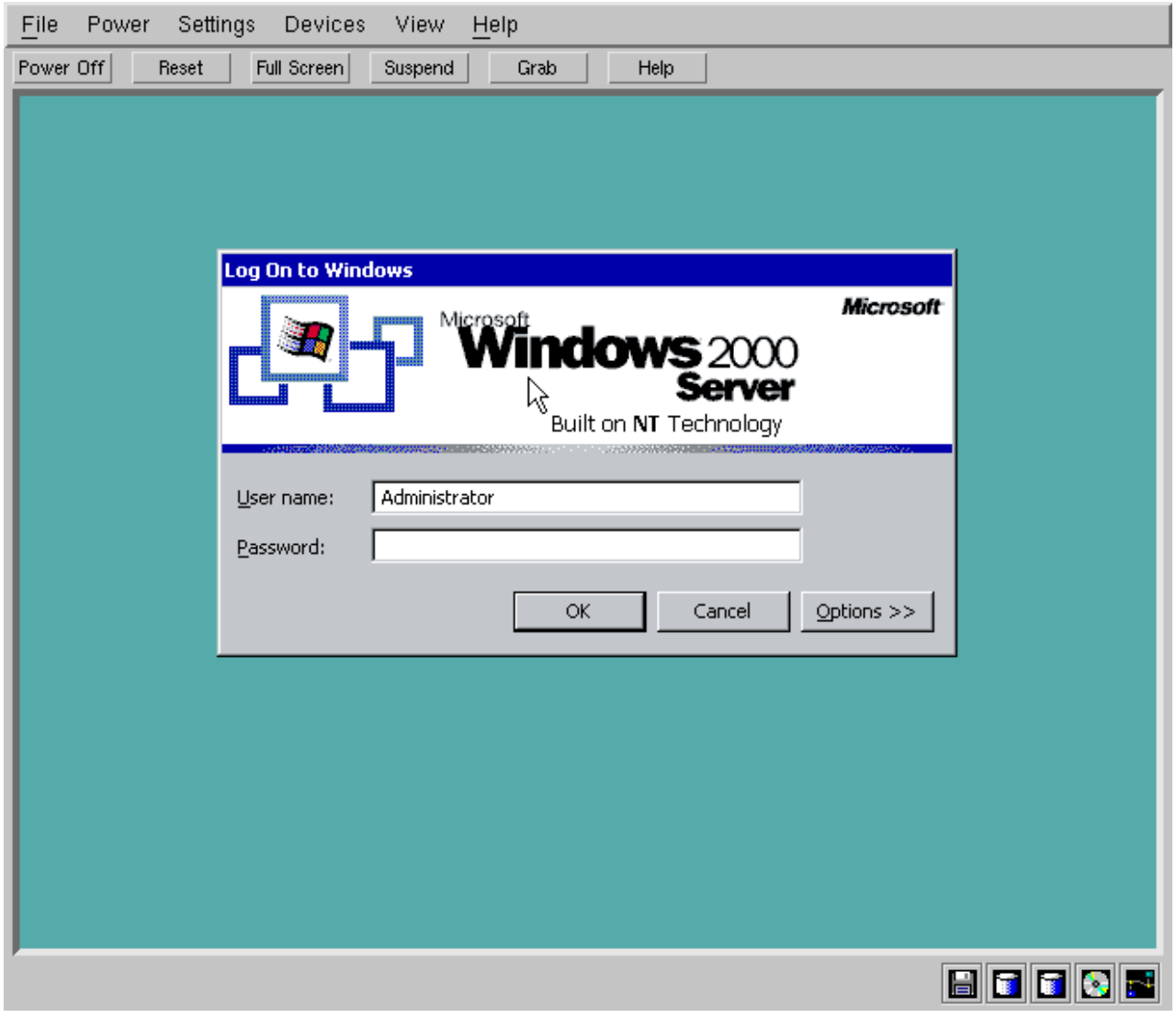
VMWare Booting



VMWare BIOS


File Power Settings Devices View Help
Power Off Reset Full Screen Suspend Grab Help
PhoenixBIOS Setup Utility
Main Advanced Security Power Boot Exit
System Time: [19:44:56]
System Date: [03/27/2003]
Legacy Diskette A: [1.44/1.25 MB 3½"]
Legacy Diskette B: [Disabled]
Primary Master [2097MB]
Primary Slave [None]
Secondary Master [CD-ROM]
Secondary Slave [None]
Keyboard Features
System Memory: 640 KB
Extended Memory: 162816 KB
Language: [English (US)]
Item Specific Help
<Tab>, <Shift-Tab>, or <Enter> selects field.
F1 Help ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ← Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit

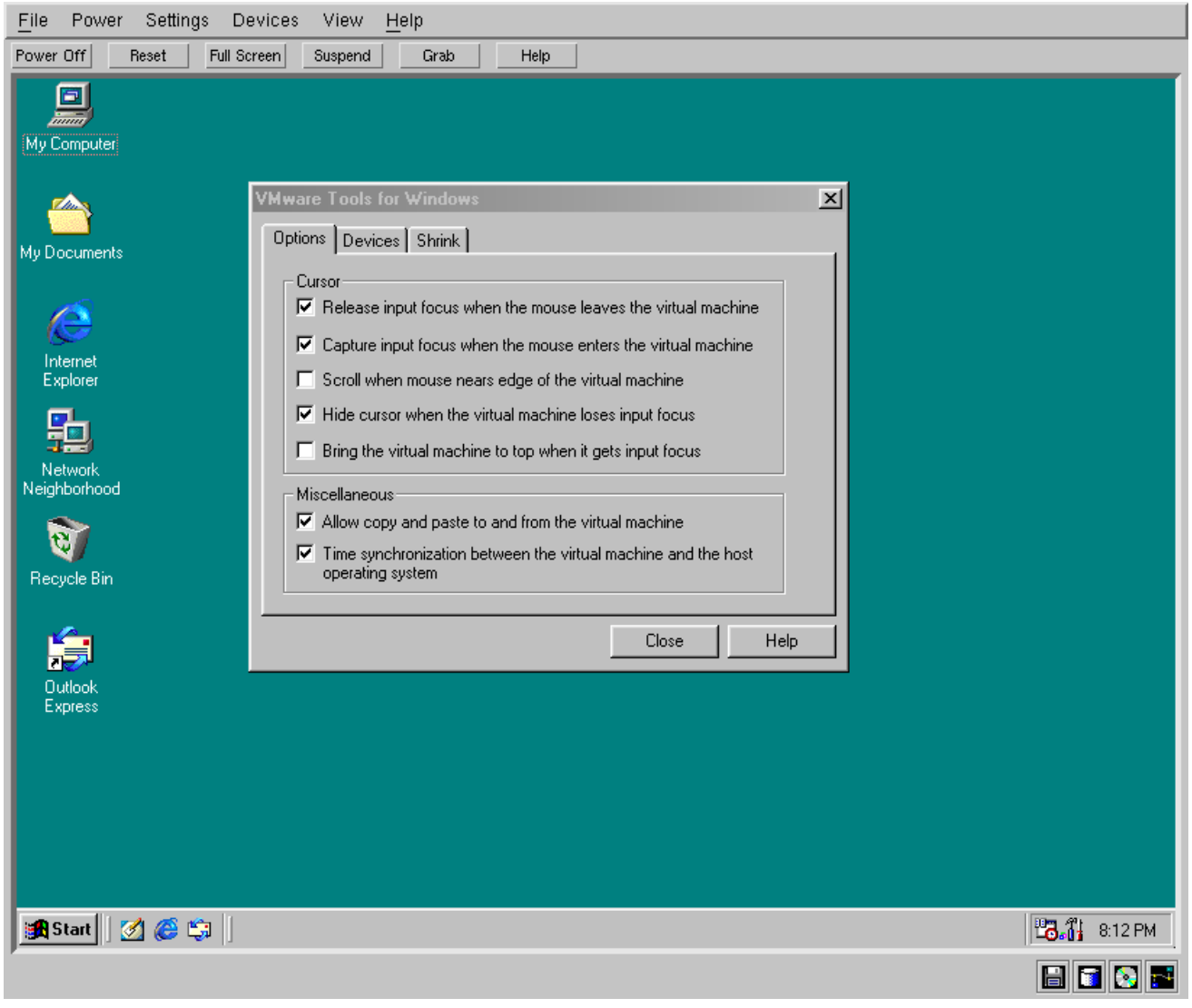
VMWare Login



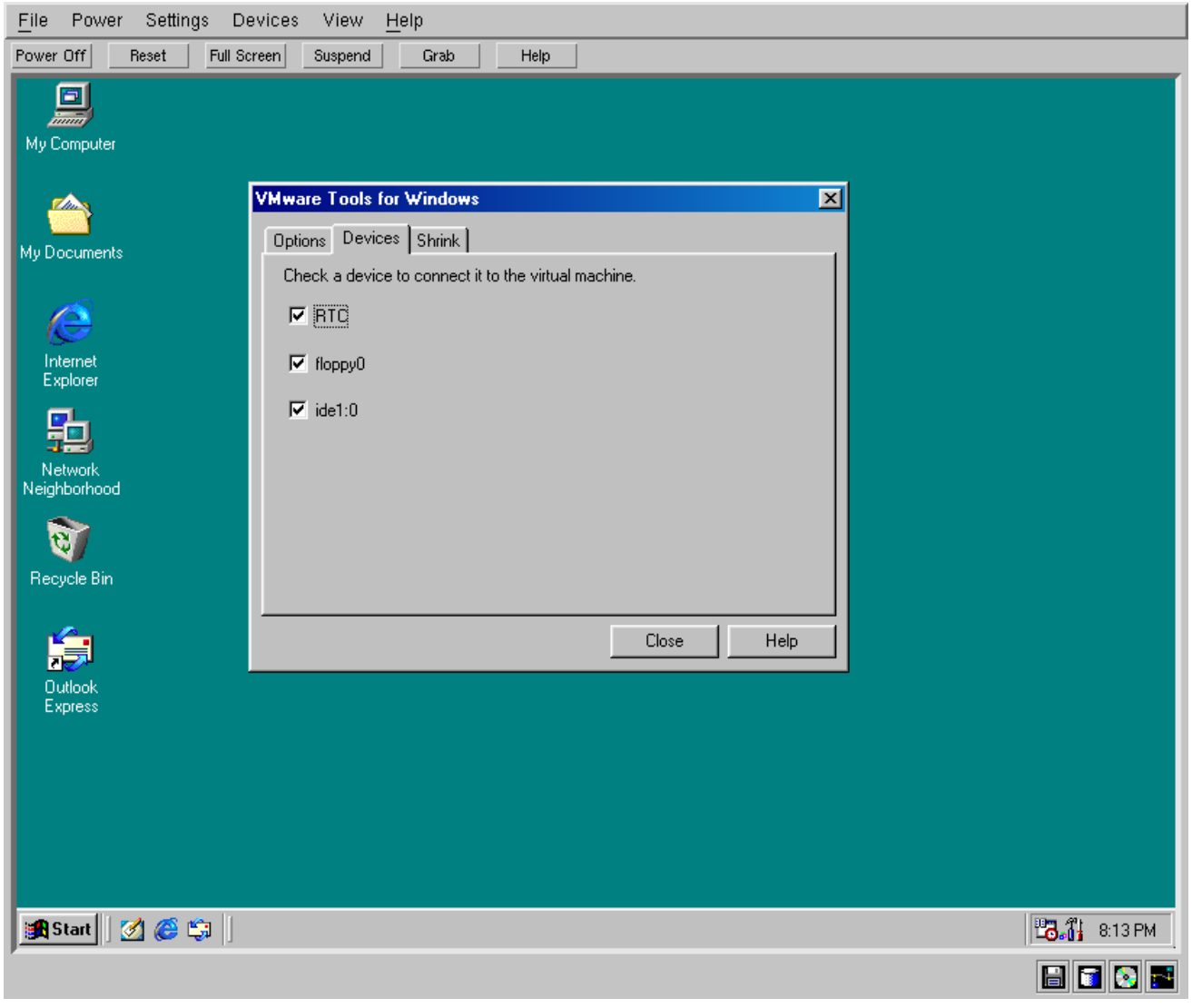
VMWare Tools

- Useful utilities available in vmware tools
- Includes:
  - SVGA driver
  - Guest OS service
  - Tools control panel

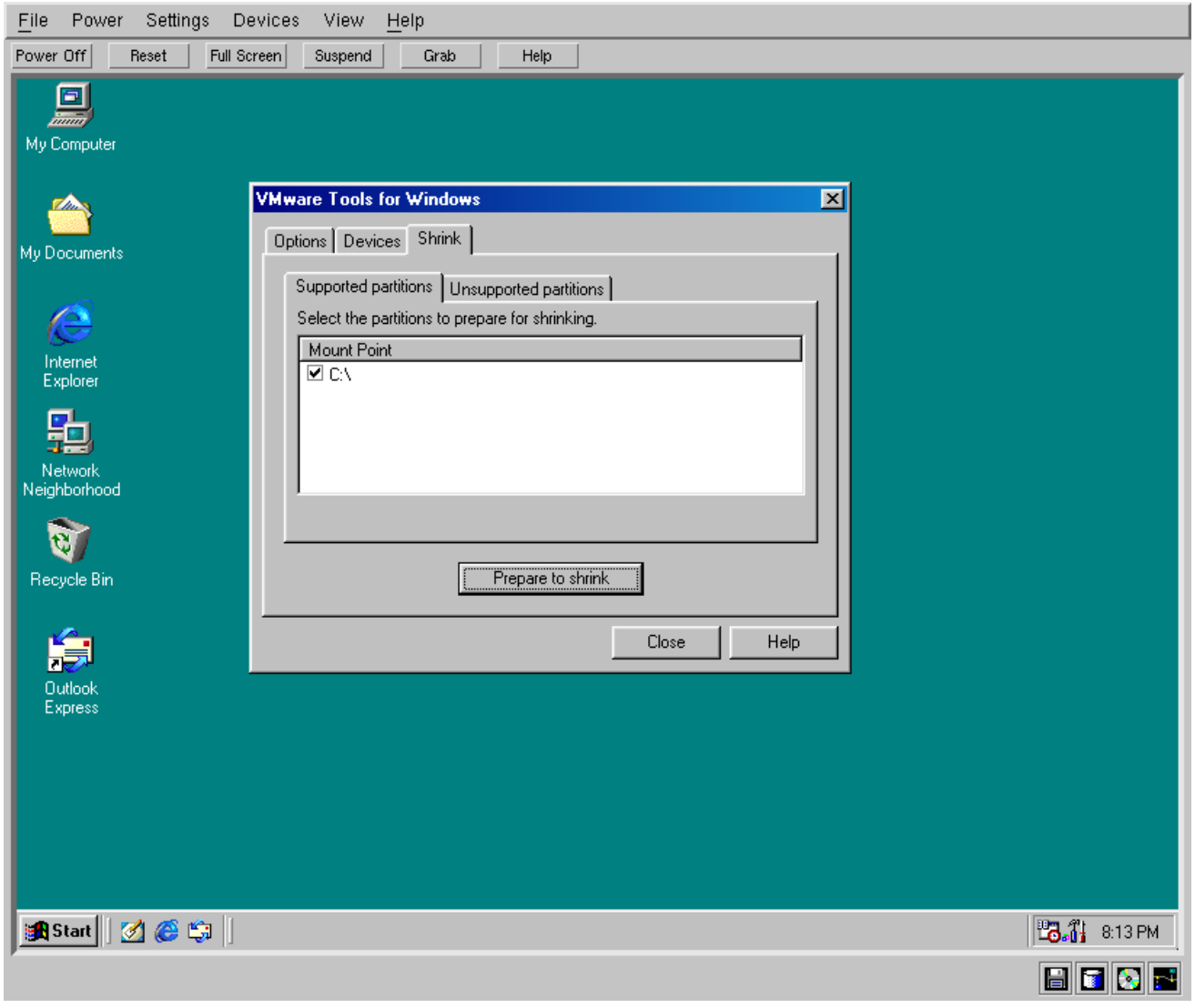
VMWare Toolbox



VMWare Toolbox



VMWare Toolbox



Wine

- Implementation of Windows win32 and win16 API on linux
- Supports Win32 (Win9x/NT and XP), Win3.x and DOS
- Doesn’t require MS Windows to run
- Can use native system DLLs if available
- Graphics - DirectX and OpenGL
- Supports networking, sound, serial ports, etc

Wine Versions

- ReWind
  - Forked version of Wine from when it changed from X11 license to LGPL
- Transgaming WineX
  - Designed for games
  - Includes Direct3D and copy protection support
- Codeweavers Wine preview
  - Nice setup program for easy install
- Codeweavers CrossOver plugin
  - Used to run Win32 browser plugins in Linux, eg QuickTime
- Codeweavers Office
  - Good support for MS Office

URLs

- User Mode Linux
  - http://user-mode-linux.sourceforge.net/
  - http://www.usermodelinux.org/
- VMWare
  - http://www.vmware.com/
- Wine
  - http://www.winehq.com/

Questions?

Questions?