

UKUUG Linux Conference 2003



Using GNU/Linux to Deliver Windows XP to the Desktop at The University of Edinburgh

J.E. Jarvis, University Computing Services

July 14, 2003

Abstract

The scalable management of desktop computing requires automated installation or re-installation of operating systems and software. Ideally the configuration data should be held centrally. For Microsoft operating systems this has often been difficult because the delivery environment has been DOS-based. In addition, filesystem image delivery techniques add additional per-seat licensing costs.

Our approach has been to use GNU/Linux to deliver the files and configuration information to networked PC's to build the operating system. Our system - PIE, the Pre-Installation Environment - is being used to deliver Windows XP but it has been designed so that we can deliver dual boot Windows XP and Linux if desired.

Introduction

At The University of Edinburgh the Microsoft Windows family is the dominant desktop operating system. Desktop provision for students in computing suites has to date been a tightly managed Windows NT 4 system. The mechanisms used to deliver the NT 4 student desktop were too inflexible to be extended widely to the more heterogeneous staff desktop. With the decision to migrate the student provision from Windows NT 4 to Windows XP Professional came the bold step to extend the project to include a staff managed desktop, also based on Windows XP Professional. There was an additional commitment to migrate from Novell Directory Services to Microsoft Active Directory.

Installing operating systems can be extremely time consuming and require considerable skilled labour. Potentially every workstation would have to be visited. In a large organisation it is essential that the time spent per workstation install is minimised and that install can be invoked by users if required. Ideally a generic boot media should be available which would query a network resource¹ to determine how a workstation should be installed. An install requires the placing of common components of the operating system plus workstation-specific overlays on the hard disk. On reboot the workstation should use the target operating system's automated build process, querying the specific overlay data, to build to its unique workstation identity. The only manual operation at the workstation should be the decision to insert and boot the media².

The existing methodology did just this. It involved the use of a DOS based boot floppy which connected via IPX protocol to Novell Netware servers to access a build database and the build software. The build software used was Symantec's Ghost³ which provides disk imaging⁴ and cloning. Large computing suites could be built synchronously using a multicast of the disk image.

The methodology was very robust but relatively inflexible - limited by hardware and by configuration diversity. It did however only require the insertion of floppy disk in the target machine to invoke a rebuild for which the configuration was held on central servers.

The requirement for a new delivery mechanism was that it must use TCP/IP network protocol rather than IPX. It also must provide at least equivalent functionality to the legacy system.

Preliminary investigations using DOS with a TCP/IP stack had limited success. At Christmas 2002 a think tank posed the question of whether using GNU/Linux and GNU tools could provide the delivery mechanism. After the Christmas break, proof of concept was achieved from scratch in just five days.

¹The network resource is not necessarily a database. Workstation builds generally require "recipe" information. Automated build systems used by Microsoft, Sun and RedHat all used text based instruction files. For workstation builds, databases are not flexible enough.

²For both the installation of proprietary, licensed operating system code and the rights to enter the workstation into a domain, an authentication step was mandated.

³Symantec Ghost requires a per-seat license.

⁴An image is a "copy" of a hard disk or partition on the hard disk such that it can be recreated on a different storage device.

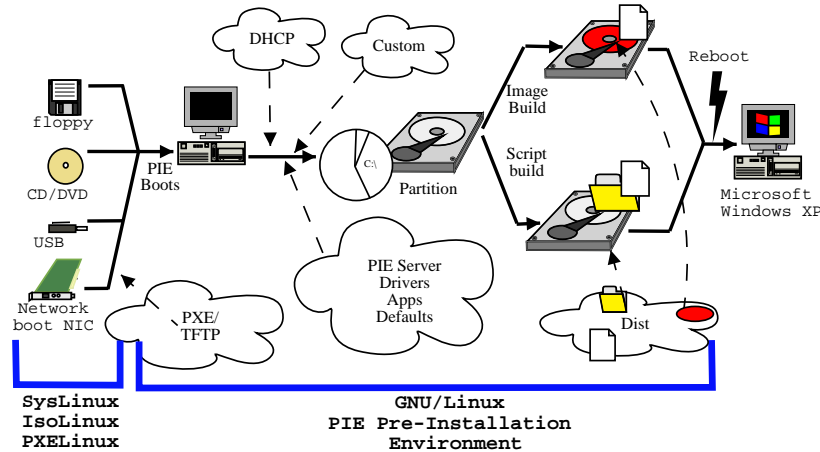


Figure 1: How PIE builds Windows XP

PIE - the Pre-Installation Environment

The Pre-Installation Environment - PIE - is a GNU/Linux based image delivery suite which is a highly configurable, open source and portable. The target task was to facilitate the delivery of Windows XP to the desktop (see Figure 1), but an underlying motivation of the two main developers was to produce a tool that resolved the generic set of problems related to automated operating system installation.

The design goals were to have a pre-installation environment that could

- Be small enough to fit on a single floppy⁵.
- Be able to utilise PXE or boot CDROM's if required.
- Support a wide range of PC hardware.
- Access network resources via TCP/IP protocol.
- Write and read images to and from disks or partitions.
- Use compression to keep image sizes small and comparable with those achieved with Ghost.
- Create, remove and resize image partitions.
- Add, remove and edit files on the imaged filesystems.
- Send or receive images via multicast.
- Have minimal "hard-wired" assumptions.
- Reduce or eliminate per-workstation license fees.

⁵The floppy disk may be considered a deprecated boot media. However being able to use floppy disks as the boot media small has other advantages - cost, availability, and read/write capability.

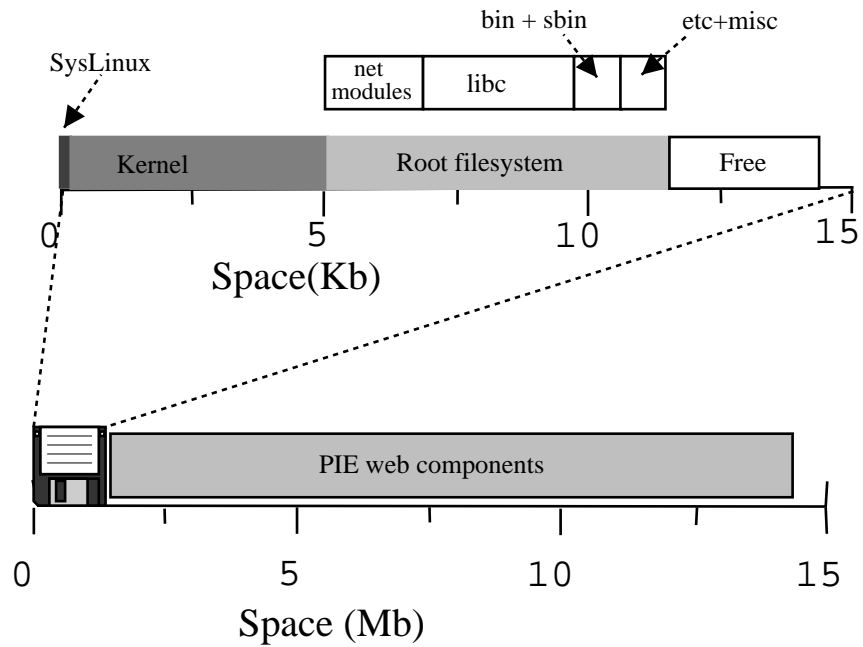


Figure 2: How PIE fits onto a floppy.

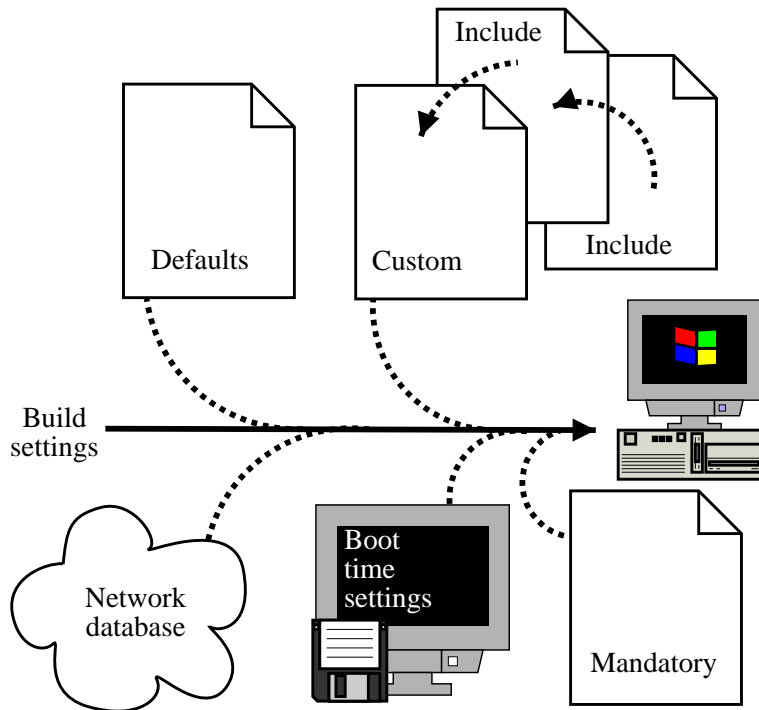


Figure 3: PIE is controlled by attribute/value pairs sourced into environment variables. Precedence is given to those applied further to the right.



Figure 4: PIE uses `dialog` to provide a basic user interface.

- Be future-proofed for PC architecture changes.
- Adhere to a sound security model.

Broad hardware support conflicts with the goal of keeping the boot media small. However, as will be demonstrated, a broad hardware support is attainable with space still left on the floppy. SYSLINUX⁶ is used as it is a small flexible bootloader code for floppy disks.

Support for other boot media was possible using ISOLINUX⁷ or PXELINUX⁸.

Access to network resources is an inherent part of GNU/Linux. Therefore this presents no problems.

For imaging a disk or partition `dd` is used. This is a tried and tested Unix tool.

GNU `parted` is perhaps the most critical tool, allowing the creation, deletion and resizing of partitions (with or without filesystems). A key bonus feature of `parted` is that if a partition is written to with a smaller `dd` image, a resize partition operation to the same (partition) sizes, stretches the image to fill the partition.

The `gzip` program provided a standard and stable compression tool. When taking images, compression could be enhanced by padding out unused space

⁶SYSLINUX is a bootloader for Linux which sits on a FAT filesystem so its configuration files are easily edited from a Windows environment (see <http://syslinux.zytor.com/>).

⁷ISOLINUX is a SYSLINUX equivalent for creating bootable CDROM's.

⁸PXELINUX is a SYSLINUX equivalent for network-booting using PXE - the pre-execution environment.

with zeros to fill the partition. This gave image size comparable with those achieved with Ghost.

The standard `mount` program allows the mounting of filesystems after imaging for file access. The package `busybox`⁹ is invaluable as a compact way of delivering standard Unix utilities - `[`, `ash`, `busybox`, `cat`, `chmod`, `chroot`, `echo`, `false`, `grep`, `gunzip`, `ifconfig`, `insmod`, `ln`, `lsmod`, `makedevs`, `mount`, `mv`, `rm`, `rmdir`, `route`, `sed`, `sh`, `tar`, `test`, `tr`, `true`, `umount`, `wget`, `zcat`. These utilities allow the adding, removing and editing of files on the image based on the instructions obtained from the network.

The tools - `udp-sender` and `udp-receiver` - proved to be sufficient for the multicast operations (see <http://udpcast.linux.lu/>).

To avoid hard-wired assumptions the functionality of PIE is controlled by attribute/value pairs sourced from the network into environment variables.

The use of GNU/Linux provides a strong future-proofing to the project. Gnu/Linux already supports 64 bit architectures. All the source code is available so there are no proprietary code problems or changing license conditions. No per-workstation license fees are due - nor indeed is a site license fee due!

The security model required that no passwords be stored on the boot media. Choices about authentication and authorisation should be decided at the point and time of implementation. It is trivial to incorporate credentials into the boot media by assigning values to the appropriate authentication variables, if that is an implementation requirement.

PIE comprises of a custom compiled modular Linux 2.4.20 bzImage kernel¹⁰ and a compressed root filesystem. The kernel is compiled with ability to run from a root filesystem in RAM. The kernel and root filesystem are loaded into memory by SysLinux (or PXELinux or IsoLinux).

When uncompressed and loaded into memory, the root filesystem uses 12Mb of RAM. Most of the 12Mb is unused at this point.

One of the most notable features is the size of both the kernel (480 Kb) and the filesystem (660 Kb). PIE can boot from a single floppy with space on the floppy to spare (see Figure 2), yet still support a very wide range of network cards!

This incredible compactness derives from an elegant design philosophy - that nothing need go on the boot media if it can be downloaded from the network. The boot media therefore contains only enough to detect its network card and reach the network. The kernel and root filesystem do not even have a module driver for the boot media itself¹¹!

The network drivers are stored as modules on the root filesystem. In the 660kb filesystem the network modules are:

⁹<http://www.busybox.net/>

¹⁰For linux kernels see <http://www.kernel.org/>

¹¹SysLinux uncompresses and loads both the PIE kernel and filesystem into memory using BIOS calls before the kernel boots.

3c509	3c59x	8139too	8390
e100	e1000	eeepro100	mii
ne2k-pci	ne	pcnet32	sis900
tg3	tulip		

There is scope for adding more.

Several additional techniques have been used to keep the root filesystem small. The uClibc library, a compact replacement for glibc weighs in at just 645 kb uncompressed compared with 1444 kb of the standard Linux glibc. UPX¹² is an executable packer. It provides excellent compression ratios on other binary packages.

When booting completes control is passed to a custom init script, `/linuxrc`. The `linuxrc` script calls `load-net-modules` which probes for network cards using the modules listed above. If it fails to find a card it halts with an error message. If it succeeds it call `start-network` which invokes `udhcpc`, a UPX'd DHCP client program. If DHCP fails¹³ the operative is informed and has the option to manually allocate network settings.

If networking fails, PIE fails and drops to an impotent shell. This shell has no access to any block devices - not even the boot media. Networking is a fundamental requirement for PIE to work. To create a compact GNU/Linux that did not need network access is possible, but it would not be PIE.

Once the network is available PIE can access functionality and drivers from the PIE web server. A fully functional C library (glibc) is installed first. The busybox on the root filesystem at boot only contains essentials, so a bigger busybox is the next download. This busybox has a lot more functionality...

Defined busybox functions:

```
[, ash, basename, busybox, cat, chmod, chroot, chvt,
clear, cmp, cp, dd, df, dirname, dmesg, dos2unix,
du, echo, env, expr, false, find, free, getopt,
grep, gunzip, gzip, head, hostname, ifconfig, init,
insmod, kill, killall, klogd, ln, logger, logname,
ls, lsmod, makedevs, md5sum, mkdir, mkfifo, mknod,
mkswap, mktemp, modprobe, more, mount, mv, pidof,
pivot_root, ps, pwd, readlink, reboot, rm, rmdir,
rmmmod, route, sed, sh, sleep, sort, stty, syslogd,
tail, tar, tee, telnet, test, tftp, touch, tr,
traceroute, true, tty, umount, uname, uniq,
unix2dos, uudecode, uuencode, wc, wget, which,
xargs, yes, zcat
```

Busybox is such an elegant package that it inspired the naming of another: the PIE install-on-demand script - `wizzybox`. Binaries such as `parted` and all the key PIE scripts exist only as links to `wizzybox` until they are first invoked. At that point `wizzybox` fetches the application or script (using `wget` and HTTP

¹²see <http://upx.sourceforge.net/>

¹³The use of DHCP is highly recommended for PIE, however since it was trivial to accommodate the manual configuration of networking, this was added for convenience.

protocol) from the PIE web server, installs it over the link and runs complete with the command line options and arguments. Wizzybox means that most of the PIE filesystem exists only on the web server until it is required.

The `wizzybox` scripts are with but one exception written in Bourne shell script¹⁴. From the outset it was decided that Bourne shell scripts are more standard and easier to support.

One of the most pivotal scripts is one called `start-block` which when first called is downloaded by `wizzybox` and itself downloads block device modules for floppy, cdrom, SCSI and IDE disks. It also creates the block devices in `/dev` using `mknod` for the relevant hardware. Finally, for IDE disks the `start-block` script consults a blacklist of IDE devices for which DMA should be turned off. The output from `start-block` is a list of block device file descriptors and the hardware type. This is cached and subsequent calls just return the cached data.

The PIE environment variables and how they are set underpin the flexibility of the system. The attribute/value pairs are set in several locations, the first of which is the boot media itself. To minimise skilled intervention the boot media should be as generic as possible. The only two hard-wired variables on the boot media are the `$pie_base` which is a URL which points to a web server address on which PIE is located, and the the build mode, `$pie_bm`. A defaults file is sourced from the PIE server and this sets values with sensible defaults. These are sufficient to perform a build of Windows XP but some user prompting will be required.

One crucial variable is the `$db` variable which points to a network database URL. This URL is queried in such a way that the hardware (MAC) address of the workstation is passed. At the University of Edinburgh the `$db` variable returns the name of the machine, the domain (Active Directory Domain) and the OU (Organisation Unit) into which to build the machine. Crucially, it also returns an optional extra URL, which if non-null is sourced (with `?[MAC address]` appended). If the custom URL is a plain text file, this text file is returned and sourced. If it is CGI script, the appended MAC address can be used as a key for further lookups.

There is some redundancy in this approach but it does improve flexibility. At Edinburgh the `$db` variable points to a database of nodes on the network. It was not appropriate to hi-jack this database for all possible machine settings however it did represent the definitive, compulsory registration database for networked devices. The `$pie_custom` URL is used to hold setting for particular machine classes - perhaps those to be partitioned differently or those which require a particular Windows XP image. To further enhance this the `include` option allows the inclusion of further attribute/value files. The call is to a shell script call `include` which can call itself recursively - therefore it can be used in a nested manner.

The final location in which attribute/value pairs might be set is in the mandatory file. This exists on the PIE server and is sourced last thus it takes precedence. Currently the only variable set in their is an error code.

¹⁴The exception is `calc_partition.pl` which is a Perl script. This may be rewritten at a future date using Bourne shell and `bc`.

The order of precedence of the variables is given by Figure 3. Note that the boot media overrides both the defaults and the custom values. This capability does however only apply to a few variables that can be passed at boot. They can either be passed by manually typing in at boot or edited into the `syslinux.cfg` file (or equivalent) to hard-wire them.

Fundamental to the successful operation of PIE is the ability to access network resources. Initially the scripts to build in subtly different ways were heavily laden with repeated code. For example the script to image build windows XP required conditional code to accommodate the image source being described as a URL. The source of the image could be from a SMB or NFS share, from a CD-ROM, from a multicast or via HTTP. The equivalent script to script build¹⁵ windows XP required the same functionality. This led to the development of a resource locator called `url`: `url` can access files from standard input or from URL type addresses and direct them to standard output or a URL specified location. The documentation for `url` states

Supported protocols are

```
http://, https://, ftp:// all by curl(1).  Read only.
file:// looks for a file on a mounted filesystem.
cdrom://, floppy:// looks for the image on the first CDROM or
                floppy disc that can be mounted.
pie:// looks for the image on the PIE partition.
mc://  looks for the image from a multicast stream. Read only.
smb:// uses mount.net to temporarily mount the share.
```

The PIE partition referred to above is a 50 Mb `ext2` (Linux) partition that is used in the University of Edinburgh implementation to store bootloader code. Grub is used and a branded splash image is inserted at boot. Grub allows us to accommodate boot menus, required for those needing dual boot Windows XP and Linux.

The actual Windows XP build scripts come in two flavours: `image-build-xp` and `script-build-xp`. These scripts demonstrate an adherence to Unix conventions of using command line switches to modify core behaviour. For example, `script-build-xp` has a `--prep` option to facilitate the building of a reference machine from which to obtain a master image. Other switches include one to indicate whether no authentication is required to access the proprietary images.

One advantage of using GNU/Linux over DOS is the ability to use pipelines and FIFO's¹⁶ *effectively*. PIE is reading and writing images of several hundred megabytes and only has 12 megabytes of filesystem and an unknown amount of RAM. Pipelines ensure that the data is never stored in full in any intermediate stage between network server share and workstation hard disk.

PIE has to be used by non-Linux aware operatives. To avoid "command-line terror" the `dialog` package is used to provide a consistent if somewhat basic

¹⁵Script building Windows XP uses an answer file to provide an unattended installation, analogous to a RedHat kickstart or a Solaris jumpstart. Script building is very useful for creating reproducible reference installs from which to take images.

¹⁶A FIFO is a "first in, first out" file. They are effectively a buffer accessed as a file would be. FIFO's are available in DOS

user interface (see Figure 4). The default setting for the build mode variable `$pie_bm` brings the user to an interactive menu unless this has been overridden at in the custom file, in the boot media config file or at boot prompt. The operative need never be confronted by a shell prompt.

PIE has proved such a potent environment that additional utility functionality has been added. An audit utility can indicate what core hardware is present. A decommission utility can securely blank or overwrite¹⁷ disk devices.

To ease the change management of PIE the source code has been committed to a CVS repository. The code can be checked out, configured and a custom kernel built within half a day to an independent site.

PIE contains complex scripting code but a sound underlying design philosophy ensures that the code is manageable and the functionality broad. PIE is not a secure environment. All processes run as root. However access to networked resources can be password protected. However the ability to boot PIE requires that physical security of the workstation is already compromised.

¹⁷One can choose to blank the disks with data from `/dev/zero` or select multiple disks overwrites with data from `/dev/urandom`.

Conclusions

PIE provides a powerful, portable networked utility environment which exists only in memory. The underlying design of PIE - just enough on the boot media to reach network - takes advantage of the flexible, modular design of GNU/Linux.

PIE, the Windows XP installation product, has capabilities associated with expensive commercial image delivery software. It obviates the need for per-workstation licence fees for distribution software. It is designed with good security in mind. It demonstrates the power and versatility of open source software even to the extent that it can facilitate easier installation of commercial operating systems. It promises to be a useful tool for organisations wishing to deploy operating systems.

Credits

PIE was written by Kenny MacDonald and James Jarvis of the Desktop Services Team, University Computing Services, The University of Edinburgh.

Special credit goes to Colin Higgs, Justin MacNeil, Angus Rae, Matthew Richardson and Shane Voss for initial brainstorming and subsequent feedback. Thanks go to our managers, especially Graham Newton, for having the faith in what we were doing. Credit for testing goes to Dave Ross, Mark Sammons, Mark Wiseman and Vladimir Zirojevic. Finally thanks to our wives and children for putting up with our late nights tapping away.