0. Contents

0. List of Tables

0. List of Figures

1. Intro to Linux Security

1.1. Do I need to worry

This is certainly a valid question. Your system is one of tens of millions of computers connected to the internet. You aren't a high profile bank that is likely to be targeted by criminals looking for bank account numbers. Should you really worry? After all, how will the hackers possibly find your system amongst all the others?

Let's explore this briefly and see if we need to take steps to protect our system. Like most people I have a DSL internet connection into my home provided by the local telephone company. They have assigned me an Internet Protocol (IP) address that distinguishes me from other users on the internet and supplied me with a DSL modem that is connected to the phone line.

So that everyone in my family can gain access to the internet connection without having to run network cables throughout the house I have a wireless network. This consists of a wireless router/base station that is connected to the DSL modem. The base station is a fairly common low cost device that, like most routers, includes a firewall that provides the first line of defense for my home network. I have configured my wireless network to use the highest level of encryption so the wireless transmissions are as safe as I can make them with current consumer grade technology.

Has anyone found my IP address and tried to get into my network? First we need to talk a little about the way a possible attack might begin – don't worry, we'll cover these topics in greater detail later. Computer systems talk to each through "ports". Specific applications are configured to talk to other systems through specific ports. For example computers might transfer files between each other using something called ftp (File Transfer Protocol). The ftp client and server talk to each other through port 21. The telnet command that allows users to log into one system from another over a network does so over port 25. In fact a Linux system has 65,535 ports that can be used for various forms of communication between different systems on a network. Most of these ports on a Linux system are closed by default – but some are left open simply because closing them renders the system inaccessible to anyone except the person at the keyboard.

It is not surprising to learn, therefore, that the first thing a potential intruder will try to do is see if any useful ports are open. When intruders find my IP address (usually by running a program that tries every IP address known to man until they get a response) they will scan a range of ports to see if any of them are open. So, has anyone tried to find an open port on my system? The firewall in my wireless router has a log file I can check using a web browser. In an 8 hour period the firewall logged 130 attempts to find an open port to enter my home network. The log file is full of entries that read:

2005/07/15 06:17:45 Connection attempt to base station from WAN blocked – src:¡xxx.xxx.xx.x:2364¿ dst:¡nn.nnn.nnn:3306¿

Each of these lines represents an attempt to break through the firewall and into a system on my network. Note that he IP addresses have been removed in the above log entry to protect the innocent (as the author of a book on Linux security it would be unwise to publish my IP address) and also, ironically, to protect the guilty (the IP address of the person trying to break into my system from outside).

Now let's take this one step further. I work outside my home from time to time and often need remote access to the Linux server on my home network while on the road. I do this using something called ssh (Secure Shell). The ssh utility is used to remotely log into a computer system from another system. ssh uses port 22. Needless to say I have port 22 open on my firewall and people have found this open port. I checked the logs on my Linux system to find any failed attempts to log in. I found 20 attempts to log in. All these attempts failed because invalid login and password information were entered.

Based on these experiences on what is probably a typical Linux configuration it is safe to say that no matter who you are, as long as you are connected to the internet, either directly via a cable modem or indirectly via a router there is a very good chance you will not escape the attention of those who make it their business, for what ever reason, to try to break into other computer systems.

1.2. The "Hacker" Word

The term "Hacker" is frequently used in the media when describing an individual who breaks into other people's computer systems. This is actually a misuse of a word that at one time did not have the negative connotations it now has. Years ago the word hacker was used to describe a talented computer programmer. Hackers were generally admired for their ability to rapidly write complex and efficient computer programs.

Sadly the term is now used in a derogatory sense to refer to what is essentially a criminal act and this new use for the word is now firmly rooted in popular culture. It is used by the news media, included in book titles and was even adopted by Hollywood the 1996 Angelina Jolie film titled "Hackers". For better or worse the new meaning is here to stay even if those of us who remember the old meaning of the word wish it wasn't so.

In this book we will bow to popular culture and use "hacker" in its new context with sincere apologies to those who would prefer that a hacker was still nothing more than a great programmer.

1.3. Security and Linux

As Linux users we have some inherent advantages over our fellow Windows users when it comes to security (or lack there of). Hackers, rather like gamblers, use the laws of odds and averages in their endeavors to find vulnerable computer systems to break into. They will typically target the types of systems that have the most security vulnerabilities. They will also mostly focus attention on areas where there are the most opportunities for unprotected systems – in other words the types are system that are most common on the internet. In both these cases Windows is the predominant operating system. In security circles they say Windows has a larger "surface area" to attack both in terms of vulnerabilities and numbers of systems.

Linux is both more secure and less common than Windows based systems with the consequence that attacks on Linux systems occur less frequently than on Windows systems. Having said that it would be foolish to be complacent about securing any system regardless of whether it runs Windows, Linux or any other operating system.

The purpose of this book is to provide a step by step approach to securing a Linux system from outside attack. It is designed to be used and understood by both new and experienced Linux users.

-§-

2. Firewalls - The First Line

The most important first step in developing a secure environment is to avoid, wherever possible, having your Linux system being the first line of defense from outside attack. The best way to do this is to ensure that you have a firewall installed between your Linux system (or the network on which it is installed) and the connection to the internet. If, for example, your Linux system is currently connected directly to a cable or DSL modem box then you will need to think seriously about installing a router or wireless base station that includes a firewall feature between the modem and your Linux system. Linux does come with a firewall that can be configured to protect you and we cover this later in the book. It is better, however, not to rely solely on this.

A firewall essentially stands between your computer or network on which your computer resides and shields it from the dangers lurking on the internet. It can either be a software program that runs on a computer system or it can be built into a hardware device such as a wireless base station or router hub. In this chapter we are going to look at firewalls as a part of a wired or wireless hub. In later chapters we will look at configuring the firewall software on a Linux system to provide a second layer of defense against attack.

2.1. What is a Firewall

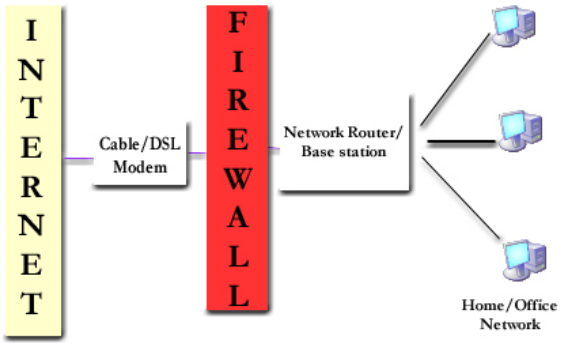
It is refreshing in an industry filled with cryptic terms and three letter acronyms (better known as TLAs) to come across a name that is at least somewhat self explanatory.

Consider the internet to be inferno of viruses, hackers and cyber criminals all looking for systems that they can invade. Rather like a firewall in real life it stops any of this unpleasantness from spreading into your environment. Another good analogy is that of a fortress wall that protects the inhabitants that live inside. The firewall stops unwanted connections from entering your internal network much like the wall around a fortress prevented the marauding hoards in medieval times. Rather like the gate in the fortress wall the firewall allows only data and connections that meet certain criteria to pass through the wall to the internal network.

A typical firewall configuration is shown in Figure 2.1. The firewall is positioned between the outside internet connection coming in through the modem and the internal network on which reside a number of Linux and Windows systems. The firewall controls all data traffic and filters out anything that is not permitted to enter the internal network.

Firewalls ?

FIGURE 2.1.



2.2. How a firewall works

A typical firewall can perform a number of tasks depending on the complexity of the firewall itself. The basic functions of a firewall are as follows:

2.2.1. Stealth Mode - Discarding Pings

This requires a little explanation. There is a common mechanism in networked environments for finding out if a particular system is up and running and connected to the network. Typically a utility called ping is given the IP address of the remote system. The ping utility sends a data packet to the remote system represented by the IP address and waits for a reply. If it gets a reply then the user knows that the system at that address is available on the network.

Whilst this seems innocuous enough there is actually good reason to configure your firewall to not respond to ping requests. You' ve probably seen the old war movies (and some new ones too) where the destroyer on the surface of the ocean uses sonar to try to locate a submarine somewhere in the depths below. The sonar sends out pings and waits to see if the sounds bounces back off the hull of the submarine. When the destroyer gets an echo it drops depth charges in an attempt to destroy the submarine. Compare this to your Linux system. The hacker will send out ping packets to every IP address on the planet and attack those that reply. By not responding to the ping packet you have a greater chance of remaining anonymous to the attacker – rather like a stealth submarine that is impervious to sonar.

Don' t be fooled by "experts" who try to tell you that ping stands for Packet Internet Groper. This just an attempt by those experts to make something sound more complicated than it is. The author of ping states that he chose that name because of the noise made by sonar.

2.2.2. Port Forwarding and Blocking

Port blocking is the most fundamental level of firewall security and will be used by most home or small business users to protect their systems.

As we mentioned previously computer systems communicate through ports. A firewall can be used to block any ports that you do not want to be open to your systems inside the firewall. For example FTP operates through port 21. If you do not wish anyone on the outside to have ftp access to your systems you will need to configure your firewall to block port 21.

Conversely, Port Forwarding is also a very useful tool to have. Suppose you have three Linux systems on your internal network and want to be able to telnet into one of those systems when you are outside your firewall (perhaps at the local café using the free Wi-Fi connection while you drink your coffee or while in a hotel on a business trip). In this situation you will configure your firewall to forward port 21 connections to the system you want to access from outside. When you connect to your IP address using telnet the firewall will see the packets arriving on port 21 and know that it must forward them to the IP address of the machine you have designated. If you have more than one system on your network it is essential that you set up port forwarding to handle this. After all, without port forwarding how would the router know which internal system you wanted to connect to?

2.2.3. Packet Filtering

Packet filtering is a much more advanced mechanism for providing security and is not available in typical small business or home use router devices.

Data is transmitted over networks and the internet in what are called packets. Each packet contains information about where the data came from and where it is going to (i.e the IP address of the sender and the your IP address). In fact a packet contains a great deal of information about the nature of the data being transmitted and many advanced firewall solutions allow you to filter the data packets coming in through your internet connection to allow or disallow packets depending on what are called filtering rules. For example you might allow a telnet session (which allows you to log into your Linux system from outside) but disallow ftp packets (which allow files to be transferred to and from of your Linux system). You may also choose to block packets arriving from an IP address that you know to be suspicious.

-§-

3. Understanding Services

In previous chapters we have touched on some of the services that a Linux system provides and the ports that those services communicate through. In this chapter we will provide an overview of the various communication related services. This information will make it easier to make an informed decision as to whether these are services you want to have running on your Linux system and, therefore, potentially accessible to the outside world.

3.1. Web Server - httpd - Port 80

The httpd service is the Hyper Text Transfer Protocol Deamon. If you plan to host your own web site on your Linux system you will need to activate this service. Without out it your web server will not serve any web pages.

http work through port 80 so you will need to make sure that you have this port open on your Firewall and configured to forward requests to the IP address of the Linux system on your network that is running the web server.

3.2. Remote Login - telnet - Port 25

The telnet service allows users to log into the Linux system from outside. For example you may want to be able to log into your Linux system to perform tasks when you are outside your office or home. You can also use telnet to log into one computer from another on the same network.

The telnet service communicates through port 21. Security experts now advise against the use of telnet these days. Telnet transmits data in plain readable text, which is readily intercepted by hackers leaving vital information (including login and password information) exposed to interception. These days SSH (Secure Shell) is recommended instead.

3.3. Secure Remote Login - ssh - Port 22

Rather like the telnet service the ssh (Secure Shell) service allows users to log into the Linux system from outside. The difference being that ssh uses an encryption mechanism to product the information being passed over the network thereby preventing others from capturing your login and password information.

The ssh service communicates through port 21.

3.4. File Transfer - ftp - Port 21

FTP is short for File Transfer Protocol and is the protocol for exchanging files over the Internet. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server. FTP uses port 21 so if you think you or others will need to transfer files to or from your Linux system make sure port 21 is configured correctly on your Firewall.

The vsftp (very secure ftp) server is recommended since it is more secure than the standard ftp server. It also considered to smaller and faster.

3.5. Mail Transfer - SMTP - Port 25

SMTP is short for Simple Mail Transfer Protocol and is a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another. The messages can then be retrieved with an e-mail client such as Evolution, KMail, or Balsa.

-§-

4. Linux Firewall - 2nd Line

In previous chapters we have covered the firewall located in the router or cable modem and viewed this as the first line of defense in protecting your Linux system from outside attack. In this chapter we will be looking at the second line of defense – the firewall on your Linux system.

During the installation of your Linux system you will have been asked a number of questions about the security settings you wanted to select. At the time you may not have understood what these settings meant or you may not recall which settings you chose. In this Chapter we will explore how to configure the security settings of your Linux system.

4.1. The lokkit Command

The lokkit command can be run at any time to change the security settings of Firewall installed on your system. To run this command you must first login as root or use the "su" command. If you are already super user on your Linux system start the lokkit command as follows:

```
/usr/sbin/lokkit
```

or to use the su command from a non-super user account as follows:

```
su {c "/usr/sbin/lokkit"
```

The lokkit command allows you to either enable or disable the Firewall. The first step if it is not already enabled is to enable it. Use the "Tab" key to move around and the "Space" key to select the "Enabled" option.

The second step is configure the Firewall. Use the Tab key to move the "Configure" button and press the "Space" key.

On the configuration screen simply select the service types that you want to support. Based on your selections lokkit will configure the Firewall to allow access to the appropriate ports. The services listed are HTTP, FTP, SSH, Telnet and Mail (SMTP). You can also specify other ports you wish to open on the Firewall in the "other ports" section.

The lokkit command also provides the option of specifying trusted devices on the "*Configure*"screen. In summary, it is possible to have more than one network device installed on a Linux system. In this scenario it might be that one device is connected to a trusted and secure network while the other is connected to a network that is connected to the outside world in some way (perhaps through a router or firewall to a broadband connection). The firewall feature allows you to disable the firewall settings for any connections coming in from the device connected to the trusted or secure network while applying the firewall rules to device connected ot the untrusted network.

-§-

5. Linux Wireless Security

The saying goes that a chain is only as strong as its weekest link and network security is no exception to this rule. In previous chapters we've looked a number of places where you can improve the level of security of your Linux system. We will now look at another area of Linux security that, if not implemented correctly, will make all the other security precautions ineffective. This is the area of wireless security.

Home and business networks are now increasingly going wireless. This has many benefits primarily in terms of convenience. Wireless networks mean that network cable doesn't have to run wherever a computer needs to be installed and you can get network and internet access anywhere as long as you are within the range of wireless base station. This, for example, gives users freedom to use their laptops in places in a building where there is no network connection available.

There is one big draw back to wireless networks in that they provide another point of security vulnerability in the network. No matter how secure you have made you have made your firewall and your Linux system if you have not also secured your wireless network a hacker can very easily eaves drop and monitor all the traffic on your network to get information such as system passwords and bank account login and password information. Even worse, if your wireless network is wide open intruders can connect to your network and potentially access all of your systems.

Carefully configuring your firewall and Linux system whilst leaving your wireless network unprotected is akin to locking the front door of your house but leaving the windows open.

Fortunately securing a wireless network is straightforward and can be implemented with a few simple steps. In this chapter we will explore the world of wireless security and show you how to ensure your wireless network is as secure is at can be made with current wireless technology.

5.1. Intro to Wireless Security

Wireless data differs from data traveling through a wired network in that the data is broadcast using radio waves. These radio transmissions pass through walls and floors and ceilings into the apartments above or below, the street outside or the house or office building next door. While data traveling through an eterhnet cable is almost impossible to intercept the data from a WiFi network can potentially be picked up by anyone with a wirless network card within the range of the wireless network.

In the wired world we rely on firewalls to protect networks and systems from intrusion. The wireless network is typically located behind the firewall and attack comes not from a hacker attempting to break in through your internet connection but from a person in the building or room next door or the opportunistic hacker who drives the streets at night with a laptop looking for unprotected wireless networks.

Wireless networks are protected from attack by using encryption. This ensures that the data passing between the computers on the network and the wireless base station/router can only be understood by other computers that know what key was used to encypt the data. It is very unlikely that a hacker will be able to find out what your encryption key is. In fact breaking into encrypted wireless networks is so difficult and time consuming that the hacker will simply take the path of least resistance and move on to one of the many unprotected wireless networks rather than try to break into yours.

There is no practical way to prevent these radio waves carrying our data from spreading outside our buildings (short of encasing them in lead) so we have to accept that the data is going to be visible to others. Rather than preventing the data from being seen by others, therefore, we instead rely on encryption to make the data unintelligable to the hacker. Whilst anyone in range of our wireless network can see the data they cannot read it without the correct encryption key.

5.2. What is Encryption ?

Encryption essentially involves taking data and subjecting it to mathematical algorithms that include a key making it unreadable to anyone else who does not know what that key is. The encrypted form of the data is know as cyphertext. Wireless networks use what is know as symmetrical encryption whereby the same key is used at both ends of the nework connection. For example, the encryption key is used as part of the mathematical equation on the sending system to encrypt the data. The receiving system then uses the same key to decrypt the data when it receives it. This key is specified by you when you configure the encryption for your wireless network and should be known only to you. The chances of a hacker guessing your encryption key are very remote and while it is possible to break the encryption code with enough time and computing power it is unlikely this kind of effort will be expended on your network. You can specify different lengths of key for the encryption process - the longer the key the stronger the encryption and the more secure the network.

WiFi wireless networks use a security standard known as Wired Equivalent Privacy (WEP). The aim of WEP is to provide a level of security in a wireless network environment that is equivalent to the security of a wired network. In practice it falls short of this

goal but for most purposes it provides an adequate level of protection.

Wireless encryption can be configured as either 64-bit or 128-bit. This refers to the length of the key that is used in the encryption algorithm and these relate directly to the strength of the encryption (128-bit encryption being stronger than 64-bit encryption). Using stronger encryption can impact the performance of the network because more time has to be spent encrypting and decrypting the data at each end of the communication. In practice it is unlikely the typical user would notice a significant difference and the strongest encryption (128-bit) is always recommended.

The encryption key are specified in hexadecimal. Unlike decimal which uses a number base of 10 (i.e digits between 0 - 9) hexadecimal uses a base of 16 (i.e digits between 0 - 9 and A - F). 64-bit encryption requires that you provide a 10 digit key whilst 128-bit encryption requires that you provide a 26 digit key.

-§-

6. File System

In computing, a file system or filesystem (often abbreviated to fs), controls how data is stored and retrieved. Without a file system, data placed in a storage medium would be one large body of data with no way to tell where one piece of data stops and the next begins. By separating the data into pieces and giving each piece a name, the data is easily isolated and identified. Taking its name from the way paper-based data management system is named, each group of data is called a “file”. The structure and logic rules used to manage the groups of data and their names is called a “ file system ”.

There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

File systems can be used on numerous different types of storage devices that use different kinds of media. As of 2019, hard disk drives have been key storage devices and are projected to remain so for the foreseeable future. Other kinds of media that are used include SSDs, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computer’s main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some file systems are used on local data storage devices; others provide file access via a network protocol (for example, NFS, SMB, or 9P clients). Some file systems are “ virtual ”, meaning that the supplied “ files ” (called virtual files) are computed on request (such as procfs and sysfs) or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

6.1. Origin of the Term

Before the advent of computers the term file system was used to describe a method of storing and retrieving paper documents. By 1961 the term was being applied to computerized filing alongside the original meaning. By 1964 it was in general use.

6.2. Architechture

A file system consists of two or three layers. Sometimes the layers are explicitly separated, and sometimes the functions are combined.

The logical file system is responsible for interaction with the user application. It provides the application program interface (API) for file operations — OPEN, CLOSE, READ, etc., and passes the requested operation to the layer below it for processing. The logical file system " manage[s] open file table entries and per-process file descriptors. " This layer provides " file access, directory operations, [and] security and protection. "

The second optional layer is the virtual file system. " This interface allows support for multiple concurrent instances of physical file systems, each of which is called a file system implementation. "[8]

The third layer is the physical file system. This layer is concerned with the physical operation of the storage device (e.g. disk). It processes physical blocks being read or written. It handles buffering and memory management and is responsible for the physical placement of blocks in specific locations on the storage medium. The physical file system interacts with the device drivers or with the channel to drive the storage device.[7]

6.3. Types of File Systems

File system types can be classified into disk/tape file systems, network file systems and special-purpose file systems. Disk file systems

A disk file system takes advantages of the ability of disk storage media to randomly address data in a short amount of time. Additional considerations include the speed of accessing data following that initially requested and the anticipation that the following data may also be requested. This permits multiple users (or processes) access to various data on the disk without regard to the sequential location of the data. Examples include FAT (FAT12, FAT16, FAT32), exFAT, NTFS, HFS and HFS+, HPFS, APFS, UFS, ext2, ext3, ext4, XFS, btrfs, ISO 9660, Files-11, Veritas File System, VMFS, ZFS, ReiserFS and UDF. Some disk file systems are journaling file systems or versioning file systems. Optical discs

ISO 9660 and Universal Disk Format (UDF) are two common formats that target Compact Discs, DVDs and Blu-ray discs. Mount Rainier is an extension to UDF supported since 2.6 series of the Linux kernel and since Windows Vista that facilitates rewriting to DVDs.

6.3.1. Flash File Systems

Flash file systems

Main article: Flash file system

A flash file system considers the special abilities, performance and restrictions of flash memory devices. Frequently a disk file system can use a flash memory device as the underlying storage media but it is much better to use a file system specifically designed for a flash device.

6.3.2. Tape File Systems

Tape file systems

A tape file system is a file system and tape format designed to store files on tape in a self-describing form[clarification needed]. Magnetic tapes are sequential storage media with significantly longer random data access times than disks, posing challenges to the creation and efficient management of a general-purpose file system.

In a disk file system there is typically a master file directory, and a map of used and free data regions. Any file additions, changes, or removals require updating the directory and the used/free maps. Random access to data regions is measured in milliseconds so this system works well for disks.

Tape requires linear motion to wind and unwind potentially very long reels of media. This tape motion may take several seconds to several minutes to move the read/write head from one end of the tape to the other.

Consequently, a master file directory and usage map can be extremely slow and inefficient with tape. Writing typically involves reading the block usage map to find free blocks for writing, updating the usage map and directory to add the data, and then advancing the tape to write the data in the correct spot. Each additional file write requires updating the map and directory and writing the data, which may take several seconds to occur for each file.

Tape file systems instead typically allow for the file directory to be spread across the tape intermixed with the data, referred to as streaming, so that time-consuming and repeated tape motions are not required to write new data.

However, a side effect of this design is that reading the file directory of a tape usually requires scanning the entire tape to read all the scattered directory entries. Most data archiving software that works with tape storage will store a local copy of the tape catalog on a disk file system, so that adding files to a tape can be done quickly without having to rescan the tape media. The local tape catalog copy is usually discarded if not used for a specified period of time, at which point the tape must be re-scanned if it is to be used in the future.

IBM has developed a file system for tape called the Linear Tape File System. The IBM implementation of this file system has been released as the open-source IBM Linear Tape File System — Single Drive Edition (LTFS-SDE) product. The Linear Tape File System uses a separate partition on the tape to record the index meta-data, thereby avoiding the problems associated with scattering directory entries across the

entire tape.

Tape formatting

Writing data to a tape, erasing, or formatting a tape is often a significantly time-consuming process and can take several hours on large tapes.[a] With many data tape technologies it is not necessary to format the tape before over-writing new data to the tape. This is due to the inherently destructive nature of overwriting data on sequential media.

Because of the time it can take to format a tape, typically tapes are pre-formatted so that the tape user does not need to spend time preparing each new tape for use. All that is usually necessary is to write an identifying media label to the tape before use, and even this can be automatically written by software when a new tape is used for the first time. Database file systems

Another concept for file management is the idea of a database-based file system. Instead of, or in addition to, hierarchical structured management, files are identified by their characteristics, like type of file, topic, author, or similar rich metadata.[12]

IBM DB2 for i [13] (formerly known as DB2/400 and DB2 for i5/OS) is a database file system as part of the object based IBM i [14] operating system (formerly known as OS/400 and i5/OS), incorporating a single level store and running on IBM Power Systems (formerly known as AS/400 and iSeries), designed by Frank G. Soltis IBM's former chief scientist for IBM i. Around 1978 to 1988 Frank G. Soltis and his team at IBM Rochester have successfully designed and applied technologies like the database file system where others like Microsoft later failed to accomplish.[15] These technologies are informally known as Fortress Rochester[citation needed] and were in few basic aspects extended from early Mainframe technologies but in many ways more advanced from a technological perspective[citation needed].

Some other projects that aren't " pure " database file systems but that use some aspects of a database file system:

Many Web content management systems use a relational DBMS to store and retrieve files. For example, XHTML files are stored as XML or text fields, while image files are stored as blob fields; SQL SELECT (with optional XPath) statements retrieve the files, and allow the use of a sophisticated logic and more rich information associations than " usual file systems ". Many CMSs also have the option of storing only metadata within the database, with the standard filesystem used to store the content of files. Very large file systems, embodied by applications like Apache Hadoop and Google File System, use some database file system concepts.

6.3.3. Transactional File Systems

Transactional file systems

Some programs need to either make multiple file system changes, or, if one or more of the changes fail for any reason, make none of the changes. For example, a program which is installing or updating software may write executables, libraries, and/or configuration files. If some of the writing fails and the software is left partially installed or updated, the software may be broken or unusable. An incomplete update of a key system utility, such as the command shell, may leave the entire system in an unusable state.

Transaction processing introduces the atomicity guarantee, ensuring that operations inside of a transaction are either all committed or the transaction can be aborted and the system discards all of its partial results. This means that if there is a crash or power failure, after recovery, the stored state will be consistent. Either the software will be completely installed or the failed installation will be completely rolled back, but an unusable partial install will not be left on the system. Transactions also provide the isolation guarantee[clarification needed], meaning that operations within a transaction are hidden from other threads on the system until the transaction commits, and that interfering operations on the system will be properly serialized with the transaction.

Windows, beginning with Vista, added transaction support to NTFS, in a feature called Transactional NTFS, but its use is now discouraged.[16] There are a number of research prototypes of transactional file systems for UNIX systems, including the Valor file system,[17] Amino,[18] LFS,[19] and a transactional ext3 file system on the TxOS kernel,[20] as well as transactional file systems targeting embedded systems, such as TFFS.[21]

Ensuring consistency across multiple file system operations is difficult, if not impossible, without file system transactions. File locking can be used as a concurrency control mechanism for individual files, but it typically does not protect the directory structure or file metadata. For instance, file locking cannot prevent TOCTTOU race conditions on symbolic links. File locking also cannot automatically roll back a failed operation, such as a software upgrade; this requires atomicity.

Journaling file systems is one technique used to introduce transaction-level consistency to file system structures. Journal transactions are not exposed to programs as part of the OS API they are only used internally to ensure consistency at the granularity of a single system call.

Data backup systems typically do not provide support for direct backup of data stored in a transactional manner, which makes the recovery of reliable and consistent data sets difficult. Most backup software simply notes what files have changed since a certain time, regardless of the transactional state shared across multi-

ple files in the overall dataset. As a workaround, some database systems simply produce an archived state file containing all data up to that point, and the backup software only backs that up and does not interact directly with the active transactional databases at all. Recovery requires separate recreation of the database from the state file after the file has been restored by the backup software.

6.3.4. Network File Systems

Main article: Distributed file system

A network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server. Programs using local interfaces can transparently create, manage and access hierarchical directories and files in remote netpwork-connected computers. Examples of network file systems include clients for the NFS, AFS, SMB protocols, and file-system-like clients for FTP and WebDAV.

6.3.5. Shared Disk File System

Shared disk file systems

Main article: Shared disk file system

A shared disk file system is one in which a number of machines (usually servers) all have access to the same external disk subsystem (usually a SAN). The file system arbitrates access to that subsystem, preventing write collisions. Examples include GFS2 from Red Hat, GPFS from IBM, SFS from DataPlow, CXFS from SGI and StorNext from Quantum Corporation. Special file systems

A special file system presents non-file elements of an operating system as files so they can be acted on using file system APIs. This is most commonly done in Unix-like operating systems, but devices are given file names in some non-Unix-like operating systems as well.

6.3.6. Device File System

Device file systems

A device file system represents I/O devices and pseudo-devices as files, called device files. Examples in Unix-like systems include devfs and, in Linux 2.6 systems, udev. In non-Unix-like systems, such as TOPS-10 and other operating systems influenced by it, where the full filename or pathname of a file can include a device prefix, devices other than those containing file systems are referred to by a device prefix specifying the device, without anything following it. Other special file systems

In the Linux kernel, configfs and sysfs provide files that can be used to query the kernel for information and configure entities in the kernel. procfs maps processes and, on Linux, other operating system structures into a filespace.

6.3.7. Minimal File System

Minimal file system / audio-cassette storage

In the 1970s disk and digital tape devices were too expensive for some early microcomputer users. An inexpensive basic data storage system was devised that used common audio cassette tape.

When the system needed to write data, the user was notified to press " RECORD " on the cassette recorder, then press " RETURN " on the keyboard to notify the system that the cassette recorder was recording. The system wrote a sound to provide time synchronization, then modulated sounds that encoded a prefix, the data, a checksum and a suffix. When the system needed to read data, the user was instructed to press " PLAY " on the cassette recorder. The system would listen to the sounds on the tape waiting until a burst of sound could be recognized as the synchronization. The system would then interpret subsequent sounds as data. When the data read was complete, the system would notify the user to press " STOP " on the cassette recorder. It was primitive, but it worked (a lot of the time). Data was stored sequentially, usually in an unnamed format, although some systems (such as the Commodore PET series of computers) did allow the files to be named. Multiple sets of data could be written and located by fast-forwarding the tape and observing at the tape counter to find the approximate start of the next data region on the tape. The user might have to listen to the sounds to find the right spot to begin playing the next data region. Some implementations even included audible sounds interspersed with the data. Flat file systems

Not to be confused with Flat file database.

In a flat file system, there are no subdirectories; directory entries for all files are stored in a single directory.

When floppy disk media was first available this type of file system was adequate due to the relatively small amount of data space available. CP/M machines featured a flat file system, where files could be assigned to one of 16 user areas and generic file operations narrowed to work on one instead of defaulting to work on all of them. These user areas were no more than special attributes associated with the files; that is, it was not necessary to define specific quota for each of these areas and files could be added to groups for as long as there was still free storage space on the disk. The early Apple Macintosh also featured a flat file system, the Macintosh File System. It was unusual in that the file management program (Macintosh Finder) created the illusion of a partially hierarchical filing system on top of EMFS. This structure required every file to have a unique name, even if it appeared to be in a separate folder. IBM DOS/360 and OS/360 store entries for all files on a disk pack (volume) in a directory on the

pack called a Volume Table of Contents (VTOC).

While simple, flat file systems become awkward as the number of files grows and makes it difficult to organize data into related groups of files.

A recent addition to the flat file system family is Amazon's S3, a remote storage service, which is intentionally simplistic to allow users the ability to customize how their data is stored. The only constructs are buckets (imagine a disk drive of unlimited size) and objects (similar, but not identical to the standard concept of a file). Advanced file management is allowed by being able to use nearly any character (including '/') in the object's name, and the ability to select subsets of the bucket's content based on identical prefixes.

-§-

7. POSIX

The Portable Operating System Interface (POSIX) is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.

7.1. Name

Originally, the name POSIX referred to IEEE Std 1003.1-1988, released in 1988. The family of POSIX standards is formally designated as IEEE 1003 and the international standard name is ISO/IEC 9945.

The standards emerged from a project that began circa 1985. Richard Stallman suggested the name POSIX to the IEEE instead of former IEEE-IX. The committee found it more easily pronounceable and memorable, and thus adopted it.

7.2. Overview

Unix was selected as the basis for a standard system interface partly because it was " manufacturer-neutral ". However, several major versions of Unix existed—so there was a need to develop a common denominator system. The POSIX specifications for Unix-like operating systems originally consisted of a single document for the core programming interface, but eventually grew to 19 separate documents (POSIX.1, POSIX.2, etc). The standardized user command line and scripting interface were based on the UNIX System V shell. Many user-level programs, services, and utilities (including awk, echo, ed) were also standardized, along with required program-level services (including basic I/O: file, terminal, and network). POSIX also defines a standard threading library API which is supported by most modern operating systems. In 2008, most parts of POSIX were combined into a single standard (IEEE Std 1003.1-2008, also known as POSIX.1-2008).

As of 2014, POSIX documentation is divided into two parts:

POSIX.1, 2013 Edition: POSIX Base Definitions, System Interfaces, and Commands and Utilities (which include POSIX.1, extensions for POSIX.1, Real-time Services, Threads Interface, Real-time Extensions, Security Interface, Network File Access and Network Process-to-Process Communications, User Portability Extensions, Corrections and Extensions, Protection and Control Utilities and Batch System Utilities. This is POSIX 1003.1-2008 with Technical Corrigendum 1.) POSIX Conformance Testing: A test suite for POSIX

accompanies the standard: VSX-PCTS or the VSX POSIX Conformance Test Suite.

The development of the POSIX standard takes place in the Austin Group (a joint working group among the IEEE, The Open Group, and the ISO/IEC JTC 1).

7.3. Some Dude on Soflw

The most important things POSIX 7 defines

C API

Greatly extends ANSI C with things like:

more file operations	:	mkdir
		dirname
		symlink
		readlink
		link (hardlinks)
		poll()
		stat
		sync
		nftw()
process and threads	:	fork
		execl
		wait
		pipe
		semaphors sem_*
		shared memory (shm_*)
		kill
		scheduling parameters (nice, sched_*)
		sleep
		mkfifo
		setpgid()
networking	:	socket()
memory management	:	mmap
		mlock
		mprotect
		madvise
		brk()
utilities	:	regular expressions (regex)

Those APIs also determine underlying system concepts on which they depend, e.g. fork requires a concept of a process.

Many Linux system calls exist to implement a specific POSIX C API function and make Linux compliant, e.g. sys_write, sys_read, ... Many of those syscalls also have Linux-specific extensions however.

Major Linux desktop implementation: glibc, which in many cases just provides a shallow wrapper to system calls.

CLI utilities

E.g.: cd, ls, echo, ...

Many utilities are direct shell front ends for a corresponding C API function, e.g. mkdir.

Major Linux desktop implementation: GNU Coreutils for the small ones, separate GNU projects for the big ones: sed, grep, awk, ... Some CLI utilities are implemented by Bash as built-ins.

Shell language

E.g., a=b; echo " $a "

Major Linux desktop implementation: GNU Bash.

Environment variables

E.g.: HOME, PATH.

PATH search semantics are specified, including how slashes prevent PATH search.

Program exit status

ANSI C says 0 or EXIT_SUCCESS for success, EXIT_FAILURE for failure, and leaves the rest implementation defined.

POSIX adds:

126: command found but not executable.

127: command not found.

¿ 128: terminated by a signal.

But POSIX does not seem to specify the 128 + SIGNAL_ID rule used by Bash: https://unix.stackexchange.com/questions/99112/default-exit-code-when-process-is-terminated

Regular expression

There are two types: BRE (Basic) and ERE (Extended). Basic is deprecated and only kept to not break APIs.

Those are implemented by C API functions, and used throughout CLI utilities, e.g. grep accepts BREs by default, and EREs with -E.

E.g.: echo ' a.1 ' — grep -E 'a.[[:digit:]]'

Major Linux implementation: glibc implements the functions under regex.h which programs like grep can use as backend.

Directory struture

E.g.: /dev/null, /tmp

The Linux FHS greatly extends POSIX.

Filenames / is the path separator portable filenames use at most max 14 chars and 256 for the full path can only contain: a-zA-Z0-9._-

See also: what is posix compliance for filesystem?

Command line utility API conventions

Not mandatory, used by POSIX, but almost nowhere else, notably not in GNU. But true, it is too restrictive, e.g. single letter flags only (e.g. -a), no double hyphen long versions (e.g. –all).

A few widely used conventions: - means stdin where a file is expected – terminates flags, e.g. ls – -l to list a directory named -l

See also: Are there standards for Linux command line switches and arguments?

" POSIX ACLs " (Access Control Lists), e.g. as used as backend for setfacl.

This was withdrawn but it was implemented in several OSes, including in Linux with setxattr.

Who conforms to POSIX?

Many systems follow POSIX closely, but few are actually certified by the Open Group which maintains the standard. Notable certified ones include:

OS X (Apple) X stands for both 10 and UNIX. Was the first Apple POSIX system, released circa 2001. See also: Is OSX a POSIX OS? AIX (IBM) HP-UX (HP) Solaris (Oracle)

Most Linux distros are very compliant, but not certified because they don't want to pay the compliance check. Inspur's K-UX and Huawei's EulerOS are two certified examples.

The official list of certified systems be found at: https://www.opengroup.org/openbrand/register/ and also at the wiki page.

Windows

Windows implemented POSIX on some of its professional distributions.

Since it was an optional feature, programmers could not rely on it for most end user applications.

Support was deprecated in Windows 8:

Where does Microsoft Windows' 7 POSIX implementation currently stand? https://superuser.com/questions/495360/does-windows-8-still-implement-posix Feature request: https://windows.uservoice.com/forums/265757-windows-feature-suggestions/suggestions/6573649-full-posix-support

In 2016 a new official Linux-like API called *Windows Subsystem for Linux* was announced. It includes Linux system calls, ELF running, parts of the /proc filesystem, Bash, GCC, (TODO likely glibc?), apt-get and more: https://channel9.msdn.com/Events/Build/2016/P488 so I believe that it will allow Windows to run much, if not all, of POSIX. However, it is focused on developers / deployment instead of end users. In particular, there were no plans to allow access to the Windows GUI.

Historical overview of the official Microsoft POSIX compatibility: http://brianreiter.org/2010/08/24/the-sad-history-of-the-microsoft-posix-subsystem/

Cygwin is a well known GPL third-party project for that *provides substantial POSIX API functionality* for Windows, but requires that you *rebuild your application from source if you want it to run on Windows.* MSYS2 is a related project that seems to add more functionality on top of Cygwin.

Android

Android has its own C library (Bionic) which does not fully support POSIX as of Android O: Is Android POSIX-compatible?

-§-

8. Unix

Unix was made by two guys Ken Thompson (inventor of utf-8) and Dennis Ritchie (inventor or C). So these two behmoths were working on a computing system called multex : Multiplexed information computer service.

The objective of this was to have multiple programs running at the same time, hence multiplexed. Anyway they got frustrated with this and began workingon thier own project in thier spare time called UNICS : Uniplexed Information and Computing service.

As time went on people only remebered the acronym and forgot what the ending CS stood for and it got transformed into UNIX.

By this time the C programming language was mature enough that they were able to write this entire program in just C.

UNIX vs. LINUX

This one has been a little muddled in my head for a while so I thought I would finally write it down here to explain it to myself.

So linux is an Operating System and a Kernel.

-§-

9. Linux Kernel vs Os

This

-§-

wik (????b)lik (????)lin (????a)wik (????a)lin (????b)

10. References

Likegeeks Filesystem. ???? Accessed: 2019-12-25.

Linux Filesystem Explained. ????a. Accessed: 2019-12-25.

Linuxtopia. ????b. Accessed: 2019-12-25.

Wikipedia : POSIX. ????a. Accessed: 2019-12-25.

Wikipedia Filesystem. ????b. Accessed: 2019-12-25.