System Administration and Maintenance

System Administrator

- Systems administrators design, install, and support an organization’s computer systems. Responsible for LANs, WANs, network segments, and Internet and intranet systems. Works in a variety of environments, including large corporations, small businesses, and government organizations. Install and maintain network hardware and software, analyze problems, and monitor networks to ensure their availability to users. Gather data to evaluate a system’s performance, identify user needs, and determine system and network requirements.

Three Pillars of SAM:

- Scalability
- Security
- Simplicity

UNIX Operating System

- Unix is an inherently open system. Developed at a renowned research institution, it was released and licensed together with the source code long before the formal idea of “Open Source” had manifested itself.
- The Unix shell, takes a special place in the list of utilities and commands available on the system. The shell provides the primary user interface, allowing for the invocation and execution of the other tools. was designed as
- a command interpreter both for interactive use as well as for non-interactive use. That is, it included a scripting language, allowing for complex series of commands to be executed; for example, by system start-up scripts at boot time.

UNIX Operating System

- The Unix operating system was, from the very beginning, designed as a portable, multitasking, multiuser system.

Documentation Techniques

- System administrators values good documentation since all systems must be properly documented
- System documentation writing tips: Know your audience, (yourself, other system administrators, other technical people, other people anywhere)
- it is important to clearly distinguish internal from external documentation
- Online Documentation(put specific information, short and simple, becomes immediately searchable and can be linked to other sources of information, To make it easier for your readers, be sure to use the right keywords in the text or use “tags” – remember to include or spell out common acronyms or abbreviations depending on what you and your colleagues or users most commonly use

Documentation Techniques

- Knowing your audience and understanding what you wish to communicate to them are the most important aspects to understand before starting to write documentation.
- Different Document types
  - Processes and Procedures
    - Contents(a. Purpose: describe in detail how to perform a specific task;

      document the steps to follow to achieve a specific goal;

      illustrate site-specific steps of a common procedure; b. Target audience all system administrators in the organization;c. Structure simple, consecutive steps;

      checklist;bullet points with terse additional information)

Documentation Techniques

- Different Document types
  - Policies
    - Purpose: establish standards surrounding the use of available resources
    - Target audience: all users of the given systems
    - Structure: full text, including description and rationale
  - Online Help and Reference
    - Purpose list and index available resources;illustrate common tasks;describe common problems and provide solutions
    - Target audience: all users of the given systems;possibly restricted set of privileged users (depending on

      the resources indexed)
    - Structure: simple catalog or itemization; short question-and-answer layout;simple sentences with example invocations

Documentation Techniques

- Different Document types
  - Infrastructure Architecture and Design
    - Purpose describe in great detail the design of the infrastructure;illustrate and document information flow;document reality
    - Target audience: other system administrators in the organization
    - Structure : descriptive sentences with detailed diagrams;
    - references to rationale and decision making process behind the designs
  - Program Specification and Software Documentation
    - Purpose describe a single software tool, its capabilities and limitations;
    - illustrate common usage;
    - provide pointers to additional information
    - Target audience all users of the tool, inside or outside of the organization
    - Structure short, simple, descriptive sentences;
    - example invocations including sample output;
    - possibly extended rationale and detailed description, command or syntax reference etc. via in-depth guide

Documentation Techniques

- Collaboration
    - collaborate with your colleagues to produce the most accurate information possible, and allowing end users to update at least some of the documents you provide has proven a good way to keep them engaged and improve the quality of your information repository.
    - Documentation should be kept under some kind of revision control, much like software code is. Many different solutions exist, ranging from storing simple text or markup files in your own repository, to using a database backed "Wiki", to hosting documents online using a cloud provider, to using e.g. Google Docs.

Documentation Techniques

- Formats
  - To increase the readability of your documents use a short line-length and the copious use of paragraphs.
  - Breaking up your text into smaller paragraphs helps the reader relax and facilitates reading comprehension, speed, and ease.
  - Use short sentences
  - Use proper punctuation
  - Resist the temptation to use images instead of text
    - Use illustrations as supplementary, not instead of information.

Documentation Techniques

- The Components of Good Documentation
- Good documentation is:
    - ▪ Useful
    - ▪ Accessible
    - ▪ Accurate
    - ▪ Available

- Web-based Documentation

Pros and Cons

Pros:

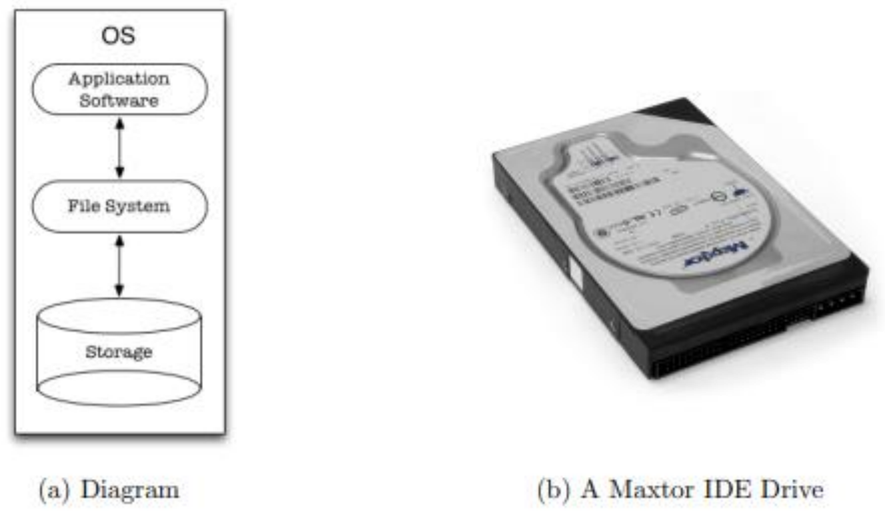
- ■ searchable
- ■ intangible
- ■ high information density
- ■ selective printing
- ■ low maintenance cost
- ■ instantaneous change propagation
- ■ easy to back up
- ■ revision history
- ■ dynamic and interactive
- ■ inherently nonlinear

Cons:

- ■ searchable
- ■ intangible
- ■ inherently nonlinear
- ■ many prerequisites for use

File Systems and Storage Models

- In order to be able to optimize our systems on this level, it is important for us to understand the principal concepts of how data is stored, the different storage models and disk interfaces.
- Storage Models
  - Direct Attached Storage
    - Alternatively, multiple direct attached disks can be combined to create a single logical storage unit through the use of a Logical Volume Manager (LVM) or a Redundant Array of Independent Disks (RAID). This allows for improved performance, increased amount of storage and/or redundancy



(a) Diagram (b) A Maxtor IDE Drive

Figure 4.1: Direct Attached Storage

File Systems and Storage Models

  - DAS can easily become a shared resource by letting the operating system make available a local storage device over the network.
- Network Attached Storage
  - allows multiple clients to access the same file system over the network, but that means it requires all clients to use specifically this file system. The NAS file server manages and handles the creation of the file systems on the storage media and allows for shared access, overcoming many limitations of direct attached storage.

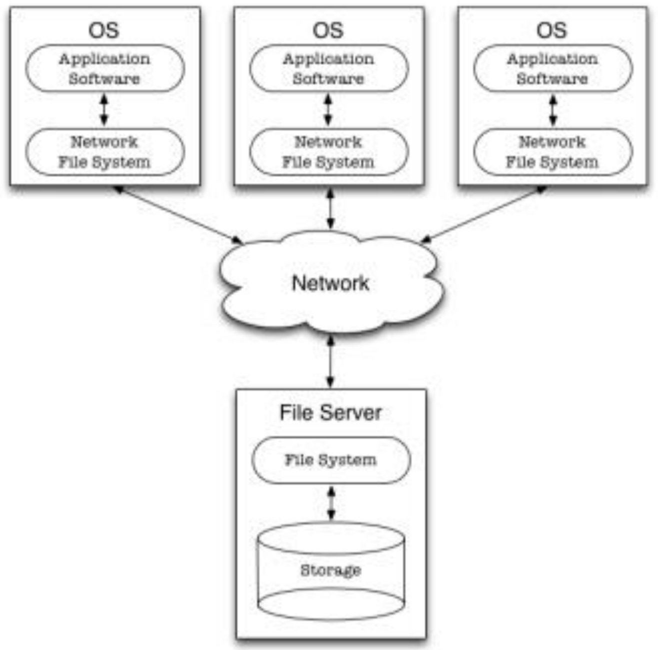


Figure 4.2: Three hosts using Network Attached Storage, or *NAS*

File Systems and Storage Models

- Storage Area Networks
  - Scaling up requirements with respect to storage size, data availability, data redundancy and performance is desirable to allow large chunks of storage on a block level. To accomplish this, we build high performance networks specifically dedicated to the management of data storage, and that is SAN.

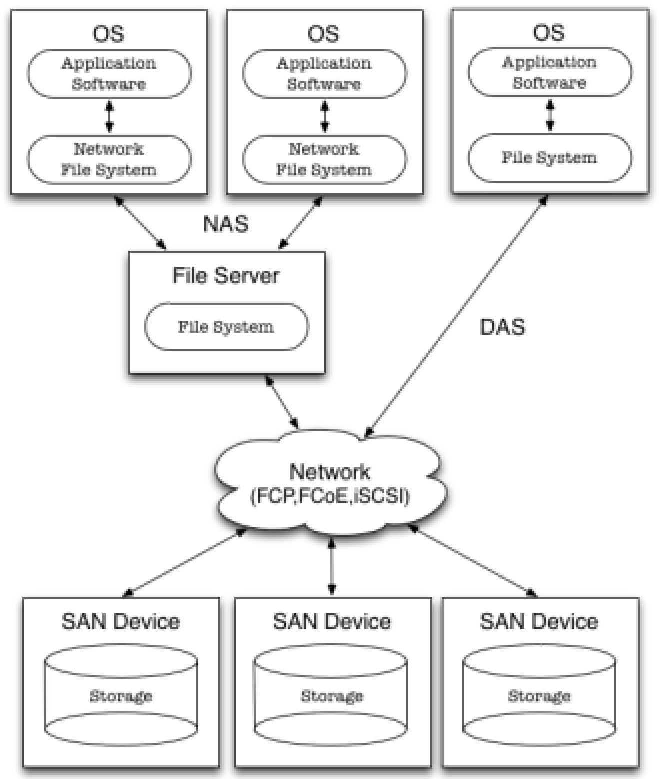


Figure 4.4: A SAN providing access to three devices; one host accesses parts of the available storage as if it was DAS, while a file server manages other parts as NAS for two clients.

File Systems and Storage Models

- Cloud Storage

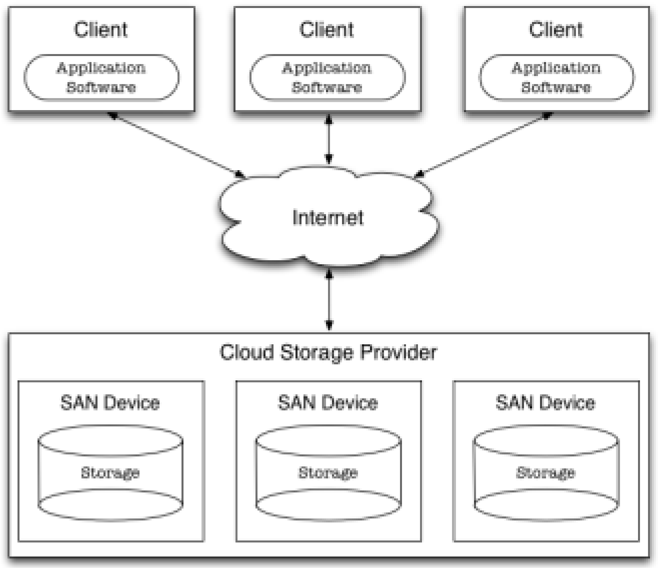
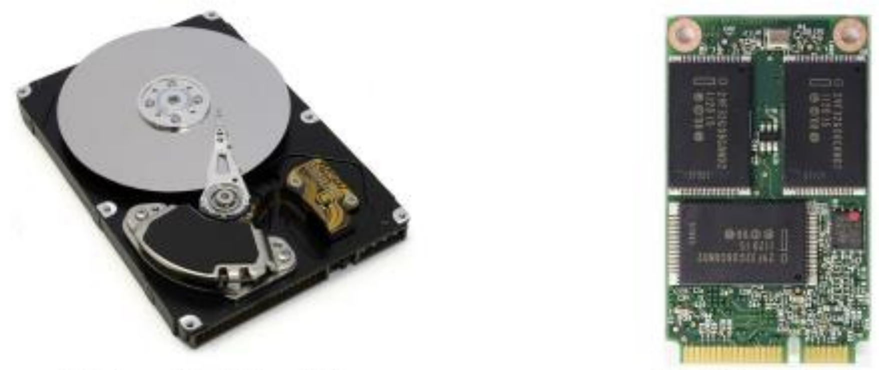


Figure 4.5: A possible cloud storage model: an internal SAN is made available over the Internet to multiple clients. In this example, the storage provider effectively functions as a NAS server, though it should generally be treated as a black box.

File Systems and Storage Models

- Storage Model Considerations
  - System administrators need to understand all of the storage models



(a) Open Hard Disc Drive

(b) SSD Mini PCIe Card

Figure 4.6: An open PATA (or IDE) hard drive (left) and a Solid State Drive (right). The HDD shows the rotating disk platters, the read-write head with its motor, the disk controller and the recognizable connector socket.

File Systems and Storage Models

- Disk Devices and Interfaces
    - Devices or media used to store data include tape drives (for use with magnetic tape), optical media such as CDs, various non-volatile memory based devices such as flash drives, DRAM-based storage, and of course the hard-disk drive (HDD).
    - Solid State Drive (SDD) offer significant advantages such as

    lower power consumption and generally higher performance, the dominant medium in use especially in enterprise scale storage solutions remains the ubiquitous hard drive

File Systems and Storage Models

- Disk Devices and Interfaces

Hard drives can be made available to a server in a variety of ways. Individual disks are connected directly to a Host Bus Adapter (HBA) using a

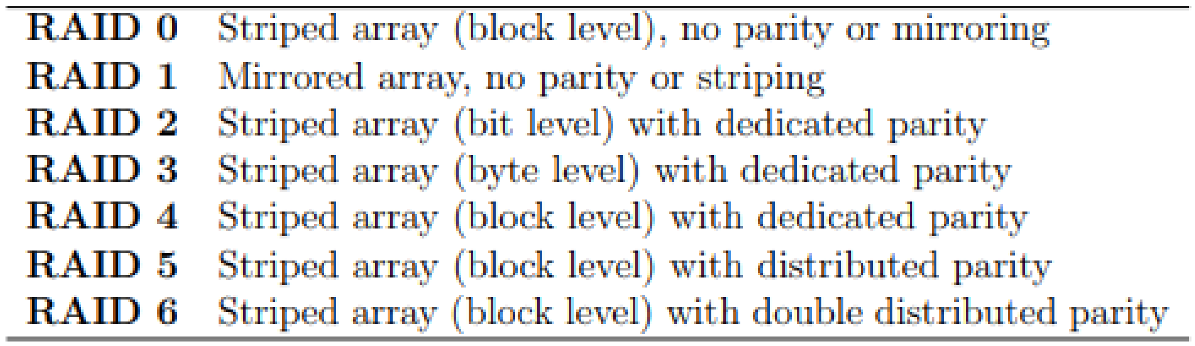
Single data/control cable and a separate power cable. The traditional interfaces here are SCSI(Small Computer System Interface), PATA(Parallel Advanced Technology Attachment) and SATA(Serial Advanced Technology Attachment), as well as Fibre Channel.

File Systems and Storage Models

- Dividing and Combining Disks
    - Disk space is a finite and fixed resource. Each hard drive we purchase has a given capacity, and it is up to the system administrators to use it efficiently.
    - Finally, the ways in which we divide or combine disks have implications on system performance and data redundancy.
- Partitions
    - Different file systems and anticipated uses of the data on a disk require different kinds of partitions. First, in order for a disk to be bootable, we require it to have a boot sector, a small region that contains the code that the computer's firmware (such as the Basic Input/Output System (BIOS)) can load into memory. I

File Systems and Storage Models

- Partitions
  - RAID
    - Logical Volume Managers provide a good way to consolidate multiple disks into a single large storage resource from which individual volumes can be created. An LVM may also provide a performance boost by striping data, or redundancy by mirroring data across multiple drives. Another popular storage technology used for these purposes is RAID, which stands for Redundant Array of Independent Disks.
    - When combining disks in a RAID configuration, the system administrator has to carefully consider the advantages and disadvantages of the available options, so-called RAID levels, where performance, redundancy, and efficiency of disk usage need to be weighed.



RAID 0	Striped array (block level), no parity or mirroring
RAID 1	Mirrored array, no parity or striping
RAID 2	Striped array (bit level) with dedicated parity
RAID 3	Striped array (byte level) with dedicated parity
RAID 4	Striped array (block level) with dedicated parity
RAID 5	Striped array (block level) with distributed parity
RAID 6	Striped array (block level) with double distributed parity

Table 4.1: A brief summary of standard RAID levels.

File Systems and Storage Models

- File Systems
  - is responsible for storing, managing and updating data on the storage device
  - When the operating system is installed, it will create a file system on the specified disk(s)
  - it needs to manage the available space and be able to store files as well as their attributes (i.e. metadata)
  - For different OS, it depends on means to identify and distinguish between different users; access control and the concepts of file permissions are a basic requirement.
  - disk file systems are designed to manage hard disk storage

File Systems and Storage Models

- Disk File System
    - manages block device storage, stores

    “files” in “directories”, maintains a file system hierarchy, controls file metadata, and allows for simple I/O via a few basic system calls
- Distributed File Systems
    - Unlike a disk file system, which provides access to the data it holds only to the processes running in the operating system the disk is attached to, a distributed file system may allow different client systems to access a centralized storage resource simultaneously, thus forming the basis for any NAS solutions.
    - Sun Microsystems’ Network File System (NFS), created in 1985, was the first such file system utilizing the Remote Procedure Calls (RPC) via the Internet Protocol (IP)Internet Protocol for the server to communicate with the clients, and remains to this day the standard distributed file system in the Unix world.

File Systems and Storage Models

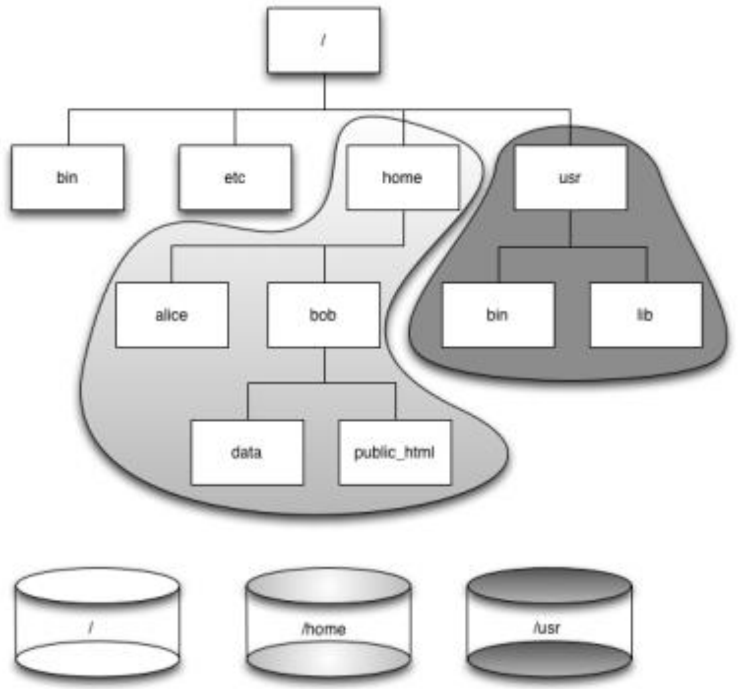


Figure 4.10: The Unix file system is a tree-like structure, rooted at /; different file systems can be attached at different directories or mount points. In this illustration, /home and /usr reside on separate disks from /.

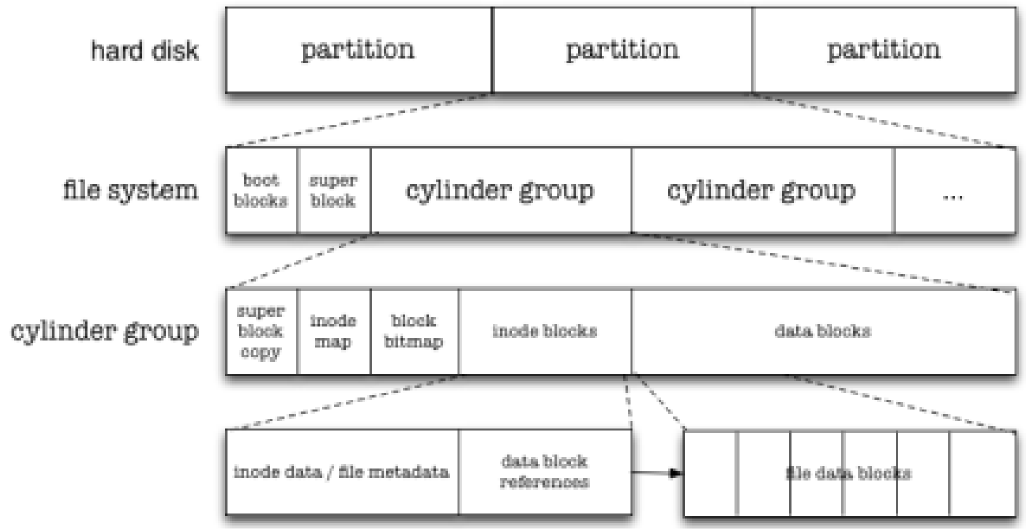


Figure 4.11: A disk may be divided into multiple partitions; a partition may contain a file system with multiple cylinder groups; each cylinder group contains some file system meta data as well as inode and data blocks.

File Systems and Storage Models

- A file system's primary task is to store data on behalf of the users. In order to read or write this data, it needs to know in which logical blocks it is located. That is, the file system needs a map of the blocks, a way to identify and address each location. This is accomplished by way of the inode and data block maps: the total number of inodes represents the total number of files that can be referenced on this file system, while the data blocks represent the space in which the file data is stored.

File Systems and Storage Models

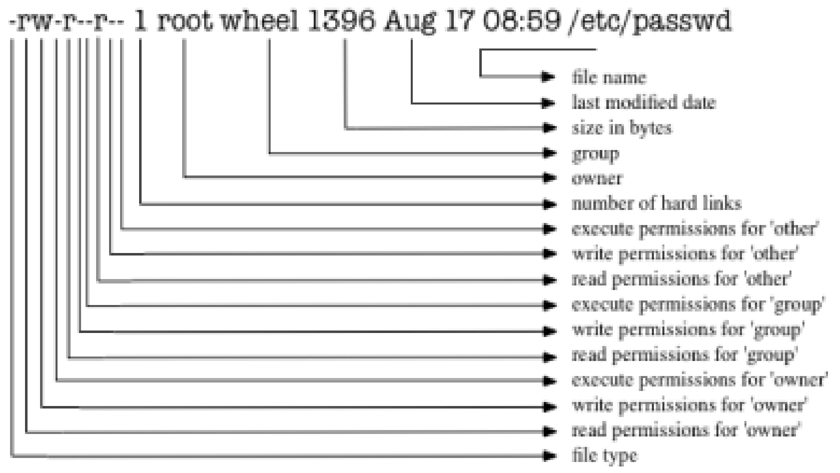
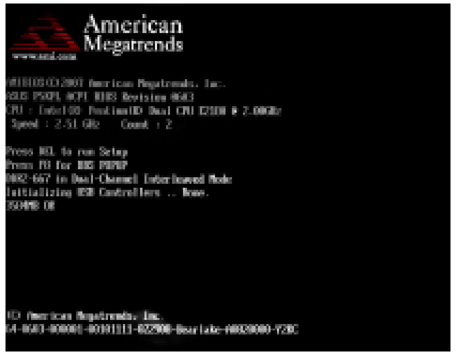

-rw-r--r-- 1 root wheel 1396 Aug 17 08:59 /etc/passwd
file name
last modified date
size in bytes
group
owner
number of hard links
execute permissions for 'other'
write permissions for 'other'
read permissions for 'other'
execute permissions for 'group'
write permissions for 'group'
read permissions for 'group'
execute permissions for 'owner'
write permissions for 'owner'
read permissions for 'owner'
file type

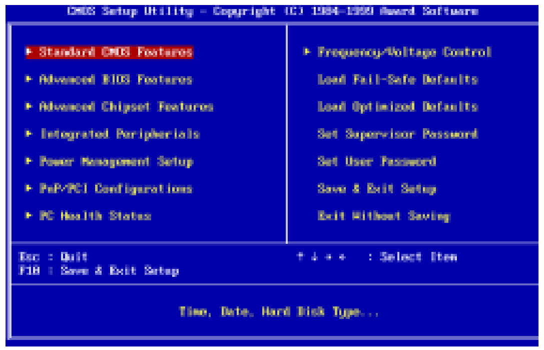
Figure 4.12: The default output of the `ls -l` command includes most of the metadata of a given file.

Software Installation and Package Management



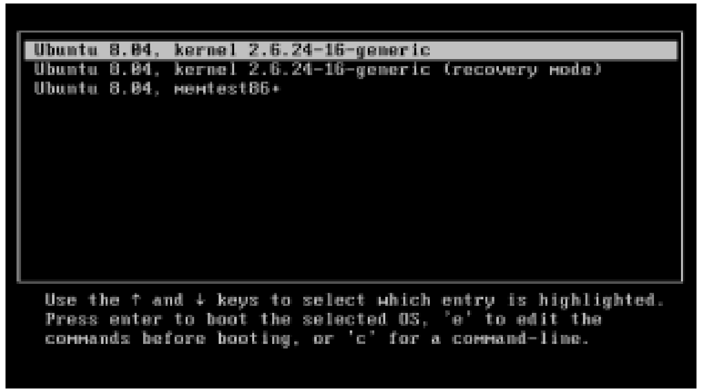
```
AMIBIOS(C)2007 American Megatrends, Inc.
ASUS P5KPL ACPI BIOS Revision 0603
CPU : Intel(R) Pentium(R) Dual CPU E2180 @ 2.00GHz
 Speed : 2.51 GHz    Count : 2

Press DEL to run Setup
Press F8 for BBS POPUP
DDR2-667 in Dual-Channel Interleaved Mode
Initializing USB Controllers .. Done.
3584MB OK

(C) American Megatrends, Inc.
64-0603-000001-00101111-022908-Bearlake-A0820000-Y2KC
```

(a) American Megatrends POST



(b) Award Software BIOS

Figure 5.1: A typical POST and BIOS screen



```
Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+

  Use the ↑ and ↓ keys to select which entry is highlighted.
  Press enter to boot the selected OS, 'e' to edit the
  commands before booting, or 'c' for a command-line.
```

(a) GNU GRUB

```
>> NetBSD/i386 BIOS Boot, Revision 5.2
>> (builds@b7, Sun Feb 7 00:30:50 UTC 2010)
>> Memory: 639/130048 k
Press return to boot now, any other key for boot menu
booting hd0a:netbsd — starting in 30
```

(b) NetBSD boot(8)

Figure 5.2: Second stage boot loader examples

Software Installation and Package Management

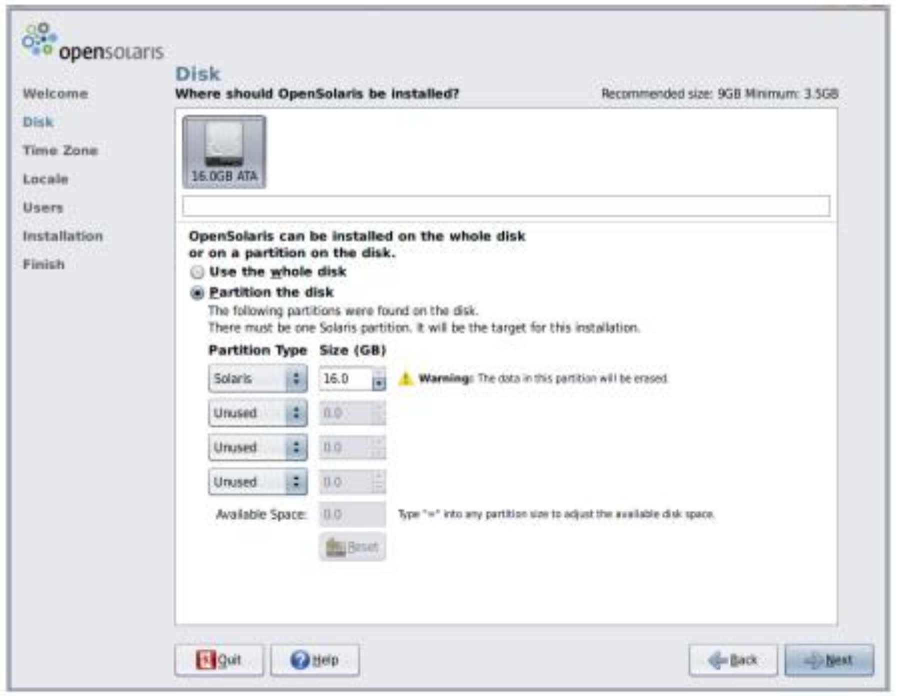
- At each step of the boot process, we find a different type of software active as the dominant component. At the final stage, when the system is up and running, we can easily identify certain applications or services that interact directly with the user. This kind of software does not require any special privileges and does not rely on any particular hardware. Even though these applications may have a dependency on specific features or capabilities in the OS, they are effectively isolated from the kernel's privileged execution mode. They are entirely optional and installed by the system administrator as needed; common examples include a web browser or a database management system.

Software Installation and Package Management

- Operating System vs. System Software vs. Third Party Software
  - Example of OS
  - Example of system software
  - Example of 3rd party software
- Static versus variable data: Files that are not expected to change during the runtime of the system are considered static. This includes the kernel, all system libraries and binaries, and many data files. Note that these files may, of course, be changed when the system is upgraded, for example, but are not modified under normal circumstances. Files that are updated by the OS, by any running applications, or by end-users at runtime, are termed variables. That is, we anticipate that these files will be created, modified, updated, or removed. Some may see frequent or even continuous updates (such as the various log files), while others only see comparatively rare modifications (such as when a system dæmon is restarted).

Software Installation and Package Management

- OS Installation



Disk

Where should OpenSolaris be installed? Recommended size: 9GB Minimum: 3.5GB

OpenSolaris can be installed on the whole disk or on a partition on the disk.

- **Use the whole disk**
- [✓ **Partition the disk**]

The following partitions were found on the disk.
There must be one Solaris partition. It will be the target for this installation.

Partition Type	Size (GB)	
Solaris	16.0	**Warning:** The data in this partition will be erased.
Unused	0.0	
Unused	0.0	
Unused	0.0	
Available Space:	0.0	Type "=" into any partition size to adjust the available disk space.

Reset

Figure 5.4: A graphical installer showing an interactive disk partitioning tool.

Software Installation and Package Management

- OS Installation
  - Identifying server requirements
    - Before a system is installed, a number of important choices have to be made:

      What file system will be used? What are the requirements of the final system with respect to file I/O? How many partitions will be created? What OS will be installed? What add-on software? What is the final purpose of the machine?
    - For example, if you plan on building a highly performant database server able to handle a large number of transactions per second, you can quickly identify the number of CPUs and CPU speed needed, the size of RAM to provide a large in-memory buffer cache, as well as hard drive disk speed and network throughput requirements, but not all operating systems are able to handle with equal efficiency the same amount of CPUs, RAM, or perhaps link aggregation, for example. As a result, your database server may have to run e.g. Solaris, even though the rest of your hosts are all running Linux (or vice versa).

Software Installation and Package Management

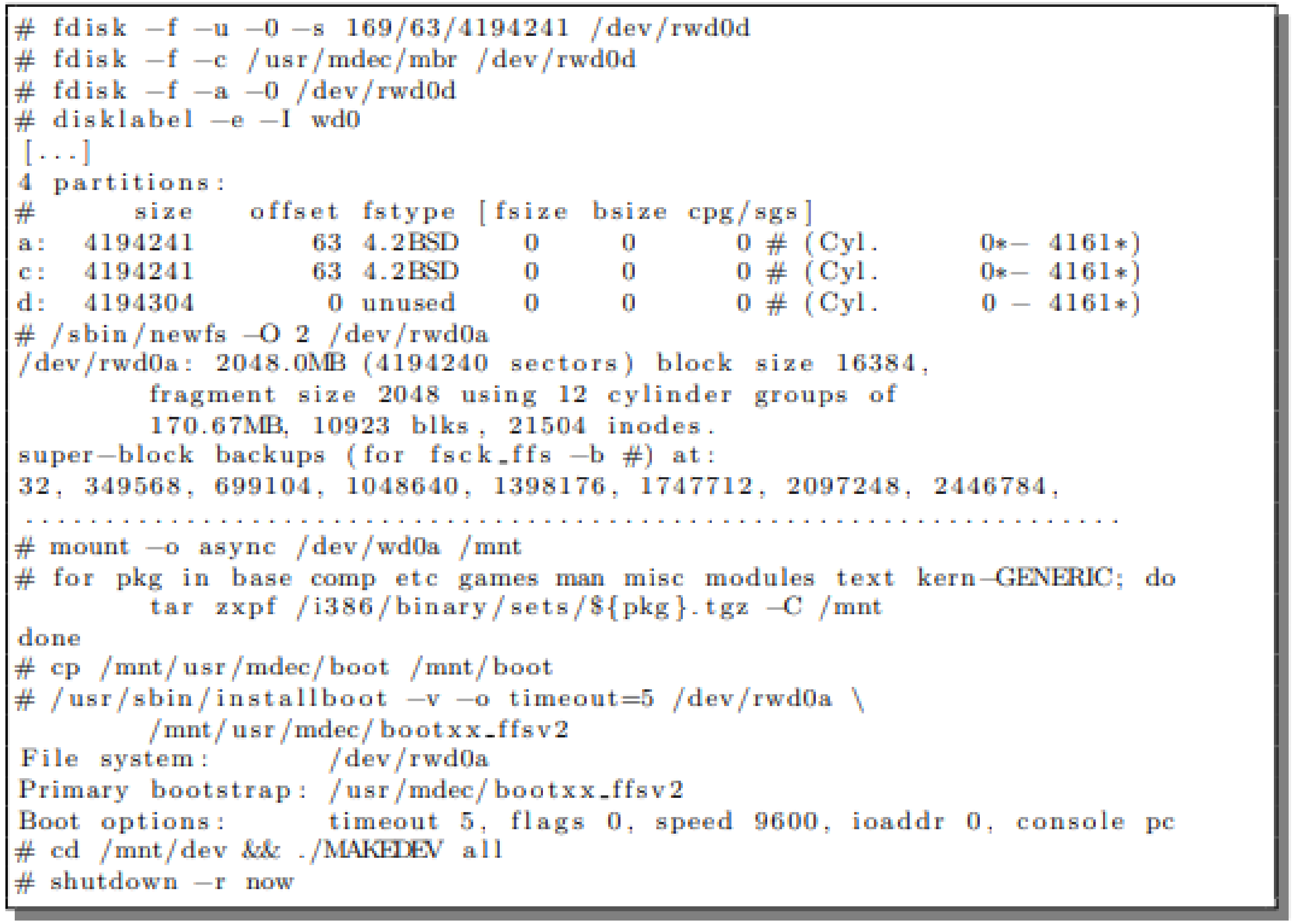
- OS Installation Overview
  - Hardware identification, provisioning and registration
  - Base OS installation.
  - Installation of add-on applications
  - Initial minimum system configuration
  - System registration
  - System restart

  Each of these steps consists of many smaller sub-steps; how exactly each of these main objectives is accomplished depends heavily on the size of the organization, the resources available, and the level of automation required. In addition, there is a thin line between system deployment and system configuration. As mentioned, the OS installation always includes at the very least a few minimal configuration steps, but some of the steps noted above may be considered part of a configuration management system’s first run

Software Installation and Package Management

- OS Installation Details
  - Boot the system
  - Identify disk(s)
  - Create partition table(s)
  - Create file system(s)
  - Make the system bootable
  - Install the OS
  - Install add-on software
  - Basic system configuration
  - Reboot

Software Installation and Package Management



```
# fdisk -f -u -0 -s 169/63/4194241 /dev/rwd0d
# fdisk -f -c /usr/mdec/mbr /dev/rwd0d
# fdisk -f -a -0 /dev/rwd0d
# disklabel -e -I wd0
[...]
4 partitions:
#       size    offset  fstype  [fsize bsize cpg/sgs]
a:   4194241        63  4.2BSD      0     0     0 # (Cyl.      0*- 4161*)
c:   4194241        63  4.2BSD      0     0     0 # (Cyl.      0*- 4161*)
d:   4194304         0  unused      0     0     0 # (Cyl.      0 - 4161*)
# /sbin/newfs -O 2 /dev/rwd0a
/dev/rwd0a: 2048.0MB (4194240 sectors) block size 16384,
        fragment size 2048 using 12 cylinder groups of
        170.67MB, 10923 blks, 21504 inodes.
super-block backups (for fsck_ffs -b #) at:
32, 349568, 699104, 1048640, 1398176, 1747712, 2097248, 2446784,
..................................................................
# mount -o async /dev/wd0a /mnt
# for pkg in base comp etc games man misc modules text kern-GENERIC; do
        tar zxpf /i386/binary/sets/${pkg}.tgz -C /mnt
done
# cp /mnt/usr/mdec/boot /mnt/boot
# /usr/sbin/installboot -v -o timeout=5 /dev/rwd0a \
        /mnt/usr/mdec/bootxx_ffsv2
File system:       /dev/rwd0a
Primary bootstrap: /usr/mdec/bootxx_ffsv2
Boot options:      timeout 5, flags 0, speed 9600, ioaddr 0, console pc
# cd /mnt/dev && ./MAKEDEV all
# shutdown -r now
```

Listing 5.4: Manual installation of NetBSD using the fdisk(8), disklabel(8), newfs(8), and installboot(8) commands.

Package Management Systems

- not only installs software, but, as the name suggests, manages it. That is, it provides tools that can identify which files belong to which package, that allow you to delete a package (and only do so if the files provided by this package are not required by other software), and pull in additional software to meet prerequisites.
- all software package management systems need to be able to not only install files, but to run certain commands required for the proper installation and configuration of the software in question. To this end, they provide the capability for a given package to execute an arbitrary shell script provided in the package, with the privileges of the invoking using – commonly the superuser. In other words, it is important to understand that any time we use a package manager to retrieve and install software for us from the Internet, we effectively let it fetch and execute unknown code on our systems. That is, not only do we trust the package manager locally – as well as the software author(s) – we also (and implicitly) trust the people who run and maintain the package repositories.

Package Management Systems

- Managing Software Updates and Patches
    - “If it ain’t broke, don’t fix it.” has become many a system administrator’s mantra, meaning that unless you have a very good reason to change something on a system that is currently running and not experiencing any problems, you shouldn’t do so. This simple lesson needs to be learned over and over again by every system administrator, and it appears it cannot be learned without a modicum of self-inflicted pain.
    - The benefit of a package manager’s dependency resolution has the often undesired result that a number of pre-requisites or dependencies further down the graph need to also be updated when a simple software change to a single package was desired. Therefore, all software updates need to be treated with great care.

Package Management Systems

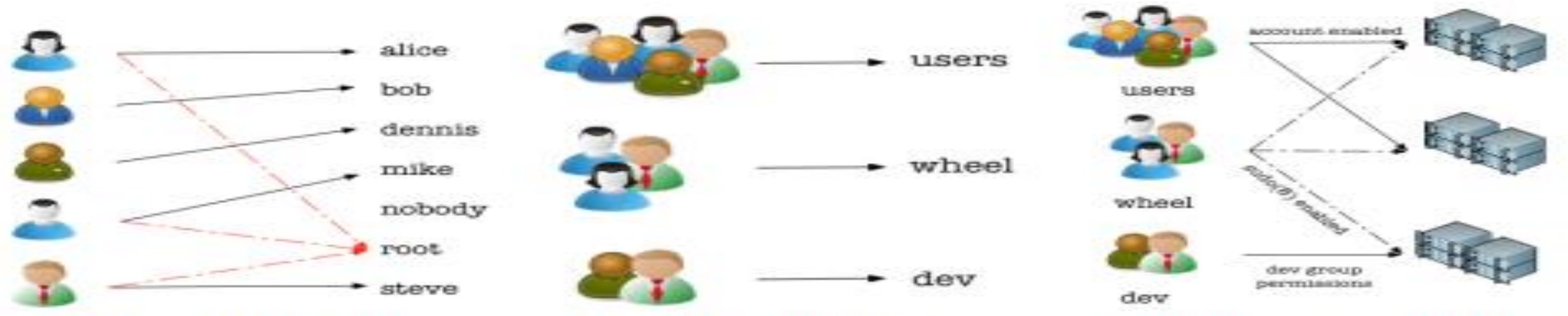
- Managing Software Updates and Patches
  - When evaluating the need to upgrade a piece of software, it is important to consider a number of factors:
    - Be aware of software updates
    - Track security updates and announcements
    - Consider privilege escalation possibilities
    - Consider the impact of updating shared libraries
    - Consider the impact of updating static libraries
    - Restart applications after the software update
    - Verify the integrity of your updates
    - Create your own patches

Users and Groups

- Allowing more than one person access to the resources of a system requires the computer to be able to distinguish between different users, which of course is done by providing distinct user accounts
- When a user wishes to use the system, she identifies herself to the computer (for example by entering her login name), upon which the system authenticates her (for example by prompting her for a password). Upon successful authentication, she is then granted access to her files and/or is allowed to run certain programs.

Users and Groups

- In order to manage the resources to be made available to different users, control the system, add or remove software, etc., we require the use of an omnipotent account, known as the superuser. And is commonly called the root account.



(a) User mappings (b) Group mappings (c) Groups to machines mappings

Figure 6.1: The set of users mapping to the set of account names is neither bijective (or "one-to-one"), nor surjective (or "onto"): some accounts are not used by an actual user, while others may be used by more than one user. Users are associated with local groups on a given host, and group membership may imply specific privileges across a set of hosts.

Groups of Users

- In addition to different types of users, we can also identify different groups of users. Privilege separation is a fundamental best practice in the area of system administration: every user should be given exactly the level of access they require, with elevated or superuser privileges being particularly strictly controlled. But even outside this special group, there are a number of users who need to share resources with one another. The nature of the environment you are in, plays an important role here, as (implicit) trust or lack thereof create vastly different use cases.

Groups of Users

- As the environment grows in size, however, you will find that you need to group accounts more carefully.
- In even larger companies, you may find that groups sharing access to the same set of servers may have competing resource requirements, even if we assume that every user has been given an account for the ultimate benefit of the organization at large. In addition, the larger your organization grows and the more users you have, the higher the risk of any one account getting compromised becomes.
- Sometimes other users are not authorized to gain access, but they are given access due to the fact that they are the boss

User Authentication

- Authentication (frequently referred to as AuthN) is the act of providing a proof of identity, not a proof of authorization (AuthZ). Although frequently entangled, the former only answers the question “Are you really who you say you are?”, while the latter concerns itself with the question “Are you allowed to perform this action?”.
- In many cases, it is desirable to combine some of these authentication methods to yield the desired level of security and usability. So-called “twofactor authentication” or 2FA has recently become a popular protection mechanism used by many internet sites: in order to gain access to a service, the user needs to provide e.g. a password (something you know) as well as a code generated by an app on their smartphone (the phone thus being something you have). There are also other access mechanism that provide multifactor authentication.

Authentication Examples

- Most simple password authentication(email access, social media access)
- another method of authentication so frequently usedthat we hardly ever think about it. Here, we see asymmetric key cryptography as a means of authentication, effectively using a something we have (a private SSH key). But note that at the same time, the server also offers us a way to authenticate it: the SSH hostkey fingerprint presented on the first connection allows us to verify that the server is in fact the one we intended to connect to.

Authentication Examples

- the use of a central authentication service by way of the Kerberos protocol. Here, we are authenticating ourselves to the central service using a password (something we know). We then use a time-based token (a “ticket” in Kerberos lingo, i.e. something we (now) have) to authenticate ourselves to the server we wish to access. At the same time, the Kerberos network authentication protocol also takes care of authenticating the server to us. This setup thus includes multiple authentication methods as well as an example of mutual authentication.

Authentication Examples

- the use of several strong factors combined with time-based access credentials to form a particularly strong example of authentication. Here, we are making use of a Certificate Authority (CA) for use with SSH to issue short-lived access credentials. In order to receive such a certificate, the user must first authenticate to the sshca service, which requires both a password (again: something we know) as well as a cryptographic One-Time Password (OTP) generated by a hardware token (something we have). In addition to the client certificate we received from the CA, the server we are finally accessing also implements another form of two-factor authentication, offering to send a message to the user’s cell phone and requiring an interactive acknowledgement. If the cell phone in question requires a fingerprint to unlock, then we are even adding a biometric factor (something weare).

System administrators with experience managing deployments in diverse environments are able to recognize and apply these general rules:

- All users are equal.
- Some users are more equal than others
- All users are to be given precisely the access rights they need, but no more
- Trust does not scale.
- You will always face tradeoffs

Configuration Management

- computer systems are not static: files are created, modified, or removed; users log in and run commands; services are started or terminated. In addition, the requirements of the systems, dictated at least in part by evolving business needs or emerging technologies are changing all too frequently as well. This leads to new software being added, patched or upgraded; user accounts are added or removed; jobs are scheduled or their frequency changed; interactions with other systems are enabled or prevented. In other words, our systems do continuously undergo change.

Configuration Management

- As we configure our machines, we may create detailed documentation about how to set up a given service, and the more systems we have, the more often we have to repeat the same steps to configure them, upgrade software, or to rebuild them when they inevitably fail.
- Since we cannot possibly keep track of hundreds of changes across thousands of systems ourselves, we delegate the task of applying well defined sets of changes to (possibly very large) numbers of systems to a class of software known as Software Configuration Management (SCM) systems,or “CMs”.

Configuration Management

- CM Requirements
  - Different CM systems allow you to specify your service requirements in different ways. Due to their particular evolution, choice of programming language, and internal architecture, they differ significantly in detail but exhibit conceptual similarities.
    - Software Installation
      - The CM needs to be able to install software on the hosts it manages. More specifically, it needs to be able to assure that software of a given version is installed (or perhaps not installed). It does not need to duplicate the capabilities of the package management system – rather, it relies on the package manager as a tool to accomplish the desired result. The system administrator provides a definition of which packages need to be present on the system, optionally specifying the exact version or perhaps simply requesting the "latest" available version. In order to accomplish this, the CM needs to have support for the package manager in use on the target system.

Configuration Management

- Service Management
  - The CM needs to be able to define which software services are supposed to be running on a given host. A machine functioning as a web server, for example, had better be running an HTTP dæmon. In order for some configuration changes to take effect, this dæmon may need to be restarted, and in order to make certain other changes, one may need to (temporarily) shut down a service.
- File Permissions and Ownership
  - The CM needs to be able to set file permissions and ownerships. If these are not explicitly defined for a given resource, then most systems will simply use the OS defaults.
  - Usermanagement by itself may be performed by the CM directly (that is, the CM systems adds or removes users, adjust their account settings, creates home directories etc.), or it may be handled via a central account management system which the CM system configures the host for

Configuration Management

- Installation of static files
  - The CM needs to be able to install a given, static file on all hosts. This is an obvious requirement: the management of local files lies at the heart of every CM system, but it is worth identifying a few distinct use cases. Here, we provide a single file that will be identical across all systems.
  - The CM system ensures that this file will be installed – as provided, with no further modifications – on the hosts. If any local modifications were made since the last time the CM ran, those will be overwritten. The ability to modify an existing file on the target system may seem like an important requirement for any CM, as this is how we configure our systems manually.
- Generation of host-specific data
  - The CM needs to be able to install host-specific files. For example, you may have different DNS or Syslog servers available to your servers based on the subnet they are deployed in. Or you may wish to add some information determined at runtime on the system itself.

Configuration Management

- Command Execution
  - The CM needs to be able to run a given command. This is about as generic a requirement as we can define, but at the same time, it is both one of the most important as well as one of the most dangerous. In order to perform system configuration, the software needs to run with superuser privileges, so any command it may run could have disastrous consequences (especially when run on every single host in your organization).
- Deployment roles
  - The final product needs to be tested before it is shipped to the customers. In the context of configuration management, this might be a service definition that has to be verified to not inadvertendly break the systems on which is should be deployed, including new releases of the service software.

Configuration Management

- Test
  - A small number of hosts on which to perform end-to-end tests after initial development make up the test environment. Once we have defined a new service, created a new package, or made any other changes to our CM, we let them be applied in this environment. The most important difference to the “development” environment is that we do not perform any manual changes here: everything goes through the full configuration management cycle. This allows us to make sure that the module we put together does in fact include all required changes, can be applied by the configuration management software and does not cause any obvious problems.
- Pre-Production
  - Once testing has determined that the changes we plan to roll out do indeed yield the desired state, we can push them into the pre-production environment, sometimes referred to as “staging”. This group consists of a representative sample of all production serving hosts, including different hardware configurations and different operating systems.

Configuration Management

- Production
  - All hosts serving production traffic or providing a crucial (possibly internal only) service.
  - Sometimes it is difficult to account for all eventualities, actual production traffic is so difficult to simulate. Hence, it may be a good idea to create a so-called “canary” role as a special method of detecting possible errors in your configuration.

Automation

System administrators need to be flexible and ingenuine to identify new and obvious solutions to the problems encountered

One needs to pay attention to details and provide strict adherence to results to produce reliable results, collect usable data and keep systems maintainable

Needs persistence and patience when coding softwares as well as debugging it and deploying new systems

Automation

Benefits of Automation:

- Repeatability
  - Reliability
  - Flexibility

Who benefits from automation?

- Ourselves
- Our Peers
- All Users

Automation

Levels of automation

- Configuration management
- Monitoring tools
- Self adaptation

Automation Pitfalls

- Increased complexity and impact
- Loss of audit trail
- Loss of Accountability
- Safeguards

Building Scalable Tools

- How software evolves?
    - Scripts
    - Programs
    - Software products
- Principles of developing Robust Software tools
    - Unix Philosophy and User Interface
    - Simplicity
    - Tools as filters
  - Test streams

Building Scalable tools

Explicit and Predictable tools

It is imperative that the user can easily determine whether or not the program succeeded.

System administrators often operate under significant pressure especially whn production reaches an outage and its the job of the SA, to bring back the solution at once to a technical problem. To address this one, one should be are of the flaws in a quickfix,

Building Scalable tools

- -Code that only requires a minium amount of commentary is easy to understand, but it is still important to explain your program to your colleagues. But raising awareness amongst your peers how the systems you create or maintain work has other beneficial side effects.
- -Code reviews.Understanding that peer review is an important and efficient method to ensure quality, many organizations require code review for certain changes. That is, before code can be committed in a repository, it requires somebody else to sign off. So-called “commit hooks” in a code repository can enforce this by requiring the commit messages to include the works “reviewed by: username”.

Building Scalable tools

Additional Guidelines for development approach of systems and tools:

-Write meaningful commit messages

-Follow a common style

-Value proper spelling

-Accept command line options

-Write the final manual

-Package your tool

Building Scalable tools

THANK YOU FOR LISTENING!