System Administration Part 1

Lecture Notes, Theory Questions and Lab Exercises

Erik Hjelmas

August 11, 2011
## Contents

1 Introduction and Basic Host Configuration ........................................ 1
   1.1 Introduction ............................................................................ 1
       1.1.1 Meta ........................................................................... 1
       1.1.2 Course design .............................................................. 2
       1.1.3 Org/Ed ........................................................................... 3
       1.1.4 Core .............................................................................. 4
       1.1.5 Problems ......................................................................... 5
       1.1.6 ITIL .............................................................................. 6
       1.1.7 Tools .............................................................................. 7
       1.1.8 Punchline ....................................................................... 7
   1.2 Host Basics ............................................................................... 8
       1.2.1 Distributions ................................................................... 9
       1.2.2 CLI ............................................................................... 11
       1.2.3 Configuration ................................................................. 13
       1.2.4 Services ......................................................................... 16
       1.2.5 Logging ......................................................................... 17
       1.2.6 Hardening ....................................................................... 17
   1.3 Common OS Adm Tasks ............................................................. 18
   1.4 Theory questions ..................................................................... 19
   1.5 Lab exercises .......................................................................... 20
2 Software Package Management

2.1 Introduction

2.1.1 Software App

2.1.2 Licensing

2.2 Software Install Issues

2.2.1 Install Types

2.2.2 File Placement

2.2.3 Dependencies

2.2.4 Modified Configs

2.2.5 Services

2.2.6 Files in Use

2.2.7 Target Users

2.2.8 Updates

2.2.9 Uninstall

2.2.10 Repair

2.2.11 Environments

2.2.12 Integration

2.3 Package Management

2.3.1 Pkg Man/Installer

2.3.2 Pkg Man System

2.3.3 Security

2.3.4 Protection

2.3.5 Good Practice

2.4 Windows

2.5 Unix/Linux

2.6 Theory questions

2.7 Lab exercises

2.7.1 Preparing your environment

2.7.2 Download and compile

2.7.3 Create software package

2.7.4 Query the package
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7.5 Install software package</td>
<td>37</td>
</tr>
<tr>
<td>2.7.6 Query software package database</td>
<td>37</td>
</tr>
<tr>
<td>2.7.7 Remove package and reinstall it</td>
<td>37</td>
</tr>
<tr>
<td>3 Automatic Installations</td>
<td>39</td>
</tr>
<tr>
<td>3.1 Install Sequence</td>
<td>39</td>
</tr>
<tr>
<td>3.2 Automated Install</td>
<td>40</td>
</tr>
<tr>
<td>3.2.1 Approaches</td>
<td>40</td>
</tr>
<tr>
<td>3.2.2 Level of attention</td>
<td>40</td>
</tr>
<tr>
<td>3.3 dd-netcat</td>
<td>41</td>
</tr>
<tr>
<td>3.4 Preseeding</td>
<td>41</td>
</tr>
<tr>
<td>3.5 Theory questions</td>
<td>45</td>
</tr>
<tr>
<td>3.6 Lab exercises</td>
<td>46</td>
</tr>
<tr>
<td>3.6.1 Preparation</td>
<td>46</td>
</tr>
<tr>
<td>3.6.2 Disk-image based: disk cloning (dd/netcat)</td>
<td>46</td>
</tr>
<tr>
<td>3.6.3 Configuration-file based: preseeding</td>
<td>47</td>
</tr>
<tr>
<td>4 Configuration Management: One Host</td>
<td>49</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>49</td>
</tr>
<tr>
<td>4.1.1 Host life cycle</td>
<td>50</td>
</tr>
<tr>
<td>4.1.2 Cfengine</td>
<td>51</td>
</tr>
<tr>
<td>4.1.3 Hello world</td>
<td>51</td>
</tr>
<tr>
<td>4.1.4 Syntax</td>
<td>52</td>
</tr>
<tr>
<td>4.2 Body</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1 Control</td>
<td>53</td>
</tr>
<tr>
<td>4.2.2 Body part</td>
<td>53</td>
</tr>
<tr>
<td>4.3 Bundle</td>
<td>53</td>
</tr>
<tr>
<td>4.4 Promise Type</td>
<td>54</td>
</tr>
<tr>
<td>4.4.1 Any Bundle</td>
<td>54</td>
</tr>
<tr>
<td>4.4.2 Agent Bundle</td>
<td>54</td>
</tr>
<tr>
<td>4.4.3 Other</td>
<td>55</td>
</tr>
<tr>
<td>4.5 Promise</td>
<td>55</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction and Basic Host Configuration

1.1 Introduction

1.1.1 Meta

- Teacher is Erik Hjelmås
  www.hig.no/~erikh
- Mixed lectures and lab (A115)
  - 1-3 hrs lecture
  - 2-5 hrs tutorial lab
  - 0-1 hrs solutions and summary
- Course Fronter room IMT3292 Systemadministrasjon
- Erik’s office is K210

Browse Fronter room.
Feel free to drop by my office any time!

- 10 credits ~ 15 hrs per week (~ 300 hrs total)
- Evaluation based on:
  - Mid-term exam
  - Project
CHAPTER 1. INTRODUCTION AND BASIC HOST CONFIGURATION

– Final exam
  • Grading based on 100 point scale, see Course description

I expect you to spend approx 9 hrs pr week (outside class scheduled hours) reading, doing the lab and theory exercises. I recommend you prioritize your time approximately (1 = highest priority):

1. Lab exercises and Review questions and problems (~6 hours)
2. Lecture notes (~1 hours)
3. Readings (~2 hours)

Project: Define a case for three weeks to really dig into some technology you would like to learn. It should resemble a typical sysadm situation where you are given a time period of three weeks to evaluate a possible change of technology, introduction of a new service, etc.

1.1.2 Course design

• Good Textbooks:

• We want stronger alignment between theory and practical lab, with the practical lab dominating the course

There are many good books on system administration. Most books are dedicated to a specific system, some books are dedicated to a specific platform, and a few are general enough for this course. The ones who are general enough for this course, are the ones shown above. Both are really good books, which are strongly recommended reading. Both books have been used in some version of this course (Burgess in 2001-2007, Limoncelli et. al. in 2008). From 2009 on we decided that the course should be even more focused on configuration management of standalone and networked hosts. With the introduction of separate courses on “network administration” and “database (including storage) and application management” our system administration course can be even more “host-focused”. During these years we also see its not possible to get a proper understanding of the complexity of system administration without getting our
1.1. INTRODUCTION

hands dirty in the lab. The lab exercises have to be complex enough to be close to real-
life scenarios, but at the same time simple enough to be comprehensible within the time
limits each week of the course.

Wikipedia is an especially relevant source of reading material for this course. System
administration is a practical discipline where we have to know the relevant protocols
and issues, combined with an overview of different implementation alternatives.

The course is focused on practical understanding of system administration: the lab is
the most important, the Wikipedia collection provides most of the relevant theory, and
the lecture notes fill in the missing parts.

• If you master this course, then graduate and start working, this course IS real life

• Equal weight and Unix/Linux and Windows, and always mention other alterna-
tives

• Not current practice or long-term theory, but Long-term practice (e.g. Unix CLI)

• Repeat and overlap on key practical concepts (e.g. AD Domain) to enforce knowl-
dge

• Integrate third-party software (e.g. Likewise, Samba)

This course has been taught at GUC since 1998, with the current teacher since 2002.
Throughout these years these key concepts have been gathered, modified and put in
action. The course is always in development, attempting to improve in every new ver-
sion.

1.1.3 Org/Ed

• Organizations

USENIX [http://www.usenix.org](http://www.usenix.org)
USENIX SAGE [http://www.sage.org](http://www.sage.org)
NUUG [http://www.nuug.org](http://www.nuug.org)

• Education

HiO-Master [http://www.hio.no](http://www.hio.no)
HiG-Master [http://www.hig.no](http://www.hig.no)
CW [http://itkurs.idg.no/](http://itkurs.idg.no/)
SANS [http://www.sans.org](http://www.sans.org)
CHAPTER 1. INTRODUCTION AND BASIC HOST CONFIGURATION

There are of course many many possibilities, these are just to mention a few. If you want to continue on a masters program, choose HiG if your interest are mostly in security, choose HiO is your interest are more pure system administration.

(Or choose to study abroad, I strongly encourage you to spend at least a semester abroad during your time as a student.)

1.1.4 Core

• Take care of our users data!

• Understand the viewpoint of the user! (its just supposed to work...)

• Core principles
  – Scalability (performance!)
  – Reliability (redundancy!)
  – Predictability (uniformity!)
  – Security

• Ethics, Teaching, Documentation, Helpdesk, Universal design/Usability, Economy!

• Technical expertise and Troubleshooting

We have all at some point experienced what its like to loose data. It is a horrible feeling, and in a business setting it can also directly be translated to a monetary cost, so protecting our users data should be our number one priority.

A user will communicate problems in many different ways and we have to try to understand the root cause of a problem which is probably not described with the symptoms we would typically hear if it was a colleague sysadm who described the symptoms. A user at GUC once reported the start button has an orange light and nothing happens when I press it. This would normally mean in this kind of workstation that the power supply had failed (which had happened many times before). However, when one of the sysadms showed up with a new power supply, it turned out that the user had a new laptop with a docking station which she tried to turn on by pressing the button on the monitor without the laptop being present in the docking station...

Scalability means that if you need five servers for fifty clients, you should not need ten servers for one hundred clients. Similarly if ten sysadms can serve five hundred users, you should not need twenty sysadms for one thousand users. We have to have a properly designed physical infrastructure and corresponding work routines (ITIL!) such that it will scale nicely.
1.1. **INTRODUCTION**

*Reliability* means that things will work all the time and the principle is closely linked with redundancy: duplication. Ideally everything should be duplicated (or replicated by at least a factor two): sysadm competence (there cannot be just one MS SQLServer expert), physical communication lines, server rooms in multiple locations, failover protocols, loadbalancers, DNS records, directory servers (master and slave), power supplies, etc etc.

*Predictability* means that components that fail should not come as a surprise. If we run a large number of DNS servers, their behaviour is predictable if they are all the same software and run on the same hardware. Homogeniety (why do you think we only have workstations of one specific brand in our labs?) helps predictability.

*Security* for sysadm means proper configuration of all hosts. Security is an important topic (although different aspects) in all systems, and there are many ways of introducing security mechanisms, but it is the sysadms who have to realize them. Proper setup of access control, patch management, careful configuration of all services, following least-privilege principles and avoiding unnecessary complexities, etc.

The bottom line is that a good system administrator have to have deep technical knowledge and be willing to always learn to keep up to date on current developments. Troubleshooting is a daily task and it is important that we are able to find root causes of problems instead of just patching symptoms. We want to spend time fixing the leaking pipe, not mopping the floor.

1.1.5 **Problems**

- Poor tools for sysadmin
- Poor existing routines and documentation
- Proprietary standards/closed systems
- Getting different OS’s to work together
- Users of all kinds...
- AND: Sysadmins of all kind!

There are no problems, only challenges!

Social skills are one of the most important factors when hiring sysadms...
This course is about technical knowledge of IT infrastructures, thereby most of what we will be doing is in the ITIL lifecycles Service Transition and Service Operation. The above mapping is just one proposed mapping, there can be alternative views, but this should give you a good link to the ITIL framework.

Service Asset and Configuration Management is about keeping an up to date over-view of all hosts (including their services of course) in terms of hardware and software configuration. This is typically done by maintaining a configuration management database (CMDB) which is synchronized with/updated by agents such as cfengine.

Release and Deploy Management is about deploying services into production, in other words: installations. Important technology we need to know about in general is how software packages work and how installations can be automated (remember: automation means scalability and consistency). Core services that have to be deployed in most infrastructures are DHCP, DNS, directory services, mail and network file systems.

Event Management is about monitoring hosts to learn about events that arise which are meaningful for the management of the infrastructure.

Incident Management is about dealing with unplanned interruptions in the infrastructure, typically involving restore of data and services.

Access Management is about identity management and controlling access to services. Usernames and advanced authentication methods based on directory lookups are an important part of this, but also host-based access from DHCP and DNS configurations are core topics for access management. A user’s access to a service is closely linked to the mappings in DNS for that service.

Monitoring and Control is about interpreting and working with the huge amounts of data that can be gathered with monitoring systems in a useful way.

IT Operations is day-to-day operations when no major events or incidents occur, meaning routine work of monitoring and following backup schedules.
1.1. INTRODUCTION

1.1.7 Tools

- PowerShell
- Bash
- apt/dpkg/msiexec/WiX
- dd/netcat, preseeding
- Cfengine
- Dhcp3-server
- MS DNS
- MS Active Directory
- MS GPO
- Likewise
- Samba
- MS Exchange
- Nagios

We will learn many useful practical skills, but always remember that the most valuable knowledge is the underlying principles. Don't focus on the djbdns configuration details, focus on the working of the DNS protocol. Don't see Active directory and Samba as two separate products which you know how to configure, see what they have in common, you will most likely encounter another directory server, authentication mechanism or file/print server, and it's critical that you can transfer your knowledge to new products.

1.1.8 Punchline

This course is not about network management or application/database management, it's about learning to manage standalone and networked hosts, and basic infrastructures, so let's get started ...

TAVLE:
Work or Master’s studies

- 

Thesis

- 

AppDBadm + NetworkAdm + SysAdm

- 

IT Foundations + InfoSec + ITIL

1.2 Host Basics

The first part of this course will be concerned about standalone hosts. Standalone does not mean that the host in never in a network, it just means that we treat them administratively by themselves.

We have to start by learning a lot about standalone hosts, and there are plenty of sysadm jobs where you will end up working mostly with standalone host configuration. One good example is this, where each county in Norway have to offer the students in high school some way of having their own laptop: if you were put in charge of this, how would you do it?

- Would you install and harden the hosts the best you can and leave it to the students to do the maintenance?
- Would you reinstall all hosts periodically? (after each semester?)
1.2. HOST BASICS

- Would you attempt to do remote maintenance and support yourself (of course through your sysadm team)?
- Would you attempt to make each host self-maintaining (autonomous)?

So far, each county have chosen different solutions, but the most cost-effective seems to be periodic reinstalls combined with some self-maintenance (e.g. a properly configured (automated) microsoft update).

1.2.1 Distributions

An operating system bundled with software packages.

- Commercial/Non-commercial
- Users: Enterprise/Power/Home
- Multi-hardware or Platform-specific
- Server/Desktop/Embedded devices
- i18n, li10n, just for musicians, ...
- Targeted at Security, Usability, Portability, ...

The term distribution is commonly used to refer to Linux distributions, but we might just as well talk about Windows distributions since a version of windows is much more than just the operating system. Windows comes bundled with many of the same features (package manager, standard application like a text editor, ...) as a linux distribution, just on a much smaller scale.

Linux


Note that many of the distribution are closely related (Ubuntu is based on Debian, and exists in Ubuntu, Kubuntu, Xubuntu, ...).
Debian is divided into unstable, testing and stable. It is common to always use Debian stable on servers and workstation/laptops. However, it is also fairly safe to use testing on non-critical hosts if the users require more up to date services. Each distribution also has a dedicated name, unstable is always called *sid*.

**Windows**

- Windows Server 2008 R2 (*Server Core* install option)
  - Foundation
  - Standard
  - Enterprise
  - Datacenter
- Windows 7
  - Home Premium
  - Professional
  - Ultimate

Windows distributions are called *Windows releases*, and are characterized as commercial, multi-hardware distributions which are divided based on company-level (small, medium, enterprise) and target use.
1.2. HOST BASICS

Overview of Server 2008 R2 features:

Overview of Windows 7 features:

1.2.2 CLI

- In many situations you do not want a GUI
- With CLI you can script your tasks with consistency!

“I need to do something N times on M hosts”

N=1, M=1  Use a GUI
N>1, M=1  Use CLI
N=1, M>1  Use CLI
N>1, M>1  Use CLI

What is the difference between running a program in a shell or running a program directly?
If the program is run in a shell, it is slower, but it will have additional environment variables available in memory and have shell features available such as piping and redirection. Some program might depend on shell-specific environment variables and some might not. In a configuration management system were we do remote command execution we should not embed the program/script in a shell unless it is necessary.

Bash/PowerShell

- Remember Bash from OpSys course?
- Startup scripts for login shells:
  1. /etc/profile
  2. Searches and executes first match of the files (in this order):
(a) ~/.bash_profile
(b) ~/.bash_login
(c) ~/.profile

• Startup scripts for non-login shells:
  1. ~/.bashrc

DEMO: view contents of these files, see how they link to each other (.profile includes .bashrc)

• Remember PowerShell from OpSys course?
  – All cmd.exe commands included
  – piping of objects instead of text lines

• Startup scripts for non-login shells:
  1. $env:windir/system32/WindowsPowerShell/v1.0/profile.ps1
  2. $env:windir/system32/WindowsPowerShell/v1.0/Microsoft.PowerShell_profile.ps1
  3. $env:UserProfile/Documents/WindowsPowerShell/profile.ps1
  4. $env:UserProfile/Documents/WindowsPowerShell/Microsoft.PowerShell_profile.ps1

http://msdn.microsoft.com/en-us/library/bb613488%28VS.85%29.aspx “The profiles are listed in load order. The most specific profiles have precedence over less specific profiles where they apply.”

ENV. variables

• Processes inherit their parents environment variable
• System environment variables
• User specific environment variables
• (Ordinary shell variables)
• Usage:
  – Bash: $HOME
1.2. HOST BASICS

- DOS: %HOMEPATH%
- PowerShell: $env:HOMEPATH

Environment variables play an important role in system configuration, e.g. they are used for deciding where users store their data (folder redirection).

DEMO: `sudo cat /proc/1/environ` (environment variables in the init process, the first process that is started on Unix/Linux systems). env in bash

`regedit` and see System environment variables

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment User Environment Variables Added by Applications`

`HKEY_CURRENT_USER\Environment` and also `HKEY_CURRENT_USER\Volatile Environment`

This is merged in the powershell provider ENV:

cd env:
Get-ChildItem

Also note difference between DOS and PowerShell:

```cmd
echo %HOMEPATH%
exit
echo %HOMEPATH%
echo $HOMEPATH
echo $env:HOMEPATH
```

1.2.3 Configuration

- Config data for services/applications, operating system, hardware devices, security, ...
- System wide and/or User specific
- Immediate effect or restart required?
- Unix/Linux: `/etc` and `. files and dirs in homedir`
- Windows: *The Registry*

As system administrators, we have to try and keep control of the configuration data. The more we get to know where we find the relevant configuration data, the more we know what is going on “under the hood”, the easier our job will be with making sure our system behave as they should, and our users will be happy.
etc & proc

- /etc/inittab
- /etc/fstab
- /etc/passwd
- /etc/init.d
- /etc/alternatives
- /etc/{apache,dhcp3,gimp,...}
- ...

The /etc directory contains most of the system wide configurations. We should know that all applications and services that have some system wide configuration store this in /etc/appname and often /etc/appname/appname.conf.

In other words, in the Unix/Linux world, configuration data is most often stored in plain text files. Windows takes a different approach and stores most configuration data in a database format (in the registry).

- /proc/PID
- /proc/sys
- /proc/meminfo
- /proc/sys/net/ipv4
- ...

proc is a run-time filesystem which works as a control interface for the operating system kernel.

Registry

**HKEY_LOCAL_MACHINE (HKLM)** SAM, SECURITY, SOFTWARE and SYSTEM, files in %SystemRoot%\System32\config

**HKEY_CURRENT_CONFIG (HKCC)** link to
HKLM\System\CurrentControlSet\Hardware Profiles\Current

**HKEY_USERS (HKU)** each user profile actively loaded on the machine
1.2. HOST BASICS

**HKEY_CURRENT_USER (HKCU)** link to currently logged on user in HKU

**HKEY_CLASSES_ROOT (HKCR)** a compilation of HKCU\Software\Classes and HKLM\Software\Classes

HKLM contains the subkeys SAM (user account database), SECURITY, SOFTWARE and SYSTEM which have corresponding files on disk. It also contains the subkey HARDWARE which is generated at runtime (similar to the proc filesystem on Linux).

HKCC just links into HKLM.

HKU is user specific config data for each active user profile (note: a user profile can be active without the user being logged in).

HKCU just links into HKLM.

HKCR stores file associations and is just linked into HKLM and HKCU.

Edit the registry using the GUI regedit.exe or with command-line powershell, Get-ItemProperty and Set-ItemProperty.

**DEMO:**

```bash
cd HKLM:
ls
Get-ChildItem hklm:\software\microsoft\windows\currentversion\uninstall
| ForEach-Object {Get-ItemProperty $_.pspath}
```

**WMI**

- CIM (Common Information Model)
- WMI providers
- Namespaces

The CIM standard defines a large set of management objects/tables which is supposed to represent “everything that can be managed on a host” in a standardized way. These CIM objects have properties and methods. WMI is Microsofts implementation of CIM with WMI providers (COM objects implemented as DLLs, approx 100 in total) which provides access to these CIM-based objects. Since the number of CIM objects/tables is extremely large, they are grouped into namespaces.

Note: much of the information that can be retrieved and manipulated through WMI comes from the registry (or “can also be found in the registry”).

We will get back to remoting (remote administration) later in the course.
1.2.4 Services

A Service is a process that can run on a host without a user being logged on, and it usually provides some services (web, mail, time sync, ...).

Runlevels

- Runlevels on Debian and others
  - 0 - Halt
  - 1 - Single user mode
  - 2-5 “Normal mode”
  - 6 Reboot
- Runlevel directories contain links to scripts in /etc/init.d
- S=Start, K=Kill, numbered sequence
- /etc/init.d/.depend.{boot,start,stop}

Services can be manually started/stopped/restarted directly with the init.d scripts, e.g. /etc/init.d/apache stop
but the preferred way on most linux distributions is to use the service command which does the same as above but in a more restricted environment (meaning different environment variables available when the service script is being executed), e.g. service apache stop

SvcHost/SCM

- Are commonly run through a service hosting process svchost
- Have to comply with the interface of the SCM (Service Control Manager)
- Managed through GUI or CLI

DEMO:

Get-Service

16
1.2.5 Logging

- Centralized logging
- Standard log format
- Synchronized time
- Unix/Linux: some "syslog" version
- Windows: Event log
  - System
  - Application
  - Security
  - ...

You probably want to have a centralized logging infrastructure for at least some parts of your log. For other parts of your logging you will just want to make sure log data is circulated and deleted when it gets old. For centralized logging, the question is whether you can log data from multiple platforms on the same log server, there should be a common log format which specifies how to record:

- Timestamps
- Log level (URGENT, ALERT, DEBUG, ...)
- Hostname
- Processname
- Log message

To log data in the correct sequence on a log server, timestamps are crucial, so time better be correctly in sync (ntp is our friend).

1.2.6 Hardening

*The process of shutting down unnecessary services and configuring the host to minimize the attack surface*

The security of a Windows host can be analyzed using Microsoft Baseline Security Analyzer and Secunia tools. On Linux there are similar tools such as bastille-linux. A good practice is to combine these host specific tools with vulnerability scanners such as nessus.
1.3 Common OS Adm Tasks

- Unix/Linux and Windows:
  - boot-up sequence
  - start and stop services
  - monitor (and kill!) processes and load
  - periodic processes
  - environment variables
  - logging
  - check filesystems
  - configure networking
  - add/delete users and account settings
  - interoperability
  - restrict/configure desktops

*we dont get to install applications yet...*

We will spend the laboratory exercise learning these elements in Unix/Linux and Windows.
1.4 Theory questions

1) Describe the boot process of a typical PC operating system.

2) What is the most important part of a system administrators job?

3) What does the concept of ‘Universal Design’ (NO: universell utforming) mean in system administration?

4) Describe the concept of scalability in system administration.

5) Describe the concept of reliability in system administration.

6) Describe the concept of security in system administration.

7) Describe the concept of predictability in system administration.

8) What does ”performing jobs consistently” mean?

9) What kind of data is extremely important to protect on servers (different from workstations/laptops)?

10) Describe the basic command sequence for editing and saving a file in vi.

11) Briefly describe at least five different but very typical single host administration tasks.

12) What is the consequence of using the Debian distribution name squeeze instead of testing in the repository-location-file (sources.list) of Debians package management system?

13) Briefly describe the structure of the Windows registry.
1.5 Lab exercises

The focus of this opening lab is to gain basic practical sysadm knowledge about command line usage of Unix/Linux and Windows. This is core technology competence which all sysadms have to possess, and which we will build on for the rest of the course.

Before you begin, make sure you have access to a Windows7 host with PowerShell, and a Linux (Debian) host with bash. If not otherwise stated, you are supposed to do all exercises on both Unix/Linux and Windows.

1) Prepare your working environment (environment variables)

Unix/Linux/Windows: Find out the order in which bash'/PowerShell’s configuration files are read when they start. Modify the path to include a directory where you can store your scripts (e.g. $HOME/scripts note: case matters in bash not in powershell, we might just as well say $home/scripts in PowerShell but that would not work in bash).

Windows: To allow for exeuction of scripts in powershell you need to set the correct execution policy:

Get-ExecutionPolicy and set it to Set-ExecutionPolicy RemoteSigned (you probably have to start PowerShell as admin to do this). Download and install vim (http://www.vim.org) and add vim’s bin path to your powershell profile.

(I recommend using vi/vim for all file editing since it is standard on all unix machines and easily available on most other platforms as well, and its exam material for this course :).

2) Boot-up sequence

Unix/Linux: After booting use dmesg | less (display message) to view all messages in the kernel ring buffer. Some of this is the results of the probing of all device drivers showing you all hardware the OS detects during boot. dmesg is very useful to examine e.g. when you plug-in an USB device, it will show you where the device is assigned. Be aware that since its a ring buffer old messages will be deleted. However, all messages from the last boot are usually stored in /var/log/dmesg.

Windows: Reboot and press F8, choose “enable boot logging”, and view drivers that have been loaded during boot with more c:/WINDOWS/ntbtlog.txt (note: this probably only works if you are using a windows boot loader).

3) Periodic tasks

Schedule a command to every minute output the current date to a file /dates/date or C:/dates/date.

What happens if the directory dates does not exist?
4) **Monitor (and kill!) processes and load**

List all processes and sort them by the amount of RAM usage (Unix/Linux: Resident Set Size, Windows: Working Set). Start a new process (e.g. launch a simple text editor) and kill it from the command line.

5) **Logging**

View the 10 most recently logged events.

6) **Check filesystems**

What filesystems are mounted? How much free space is available on each of them? Are there any files larger than 10MB present and if so when was those files most recently accessed?

7) **Configure networking**

What is the IP address, Netmask, gateway, dns servers and routing table? Change networking to static IP address (using the same address you currently have) and then change it back to DHCP again.

8) **Add and delete users**

Add a user data with password poL+Vot2, check that you can log in as that user, then delete the user and the corresponding account (verify that the home directory of the user no longer exists).

9) **Start and stop services**

List the services running. Find out how to map services to processes (PIDs). List all open ports and find the mapping between open ports and services. Find the scheduling services `cron` and `Task Scheduler`, stop and start them.

10) **Registry and Firewall**

Use PowerShell to change the registry value

```
HKLM:\System\CurrentControlSet\Control\Terminal Server\fDenyTSConnections
```

in such a way that the remote desktop service is enabled (check that a service starts listening on port 3389). Use `netsh advfirewall firewall add rule` ... to add a corresponding firewall rule allowing access to this service.
Chapter 2

Software Package Management

2.1 Introduction

2.1.1 Software App

- Executables
- Shared and non-shared libraries
- Images (icons), sounds
- Manual/Help files
- Directories
- Config files and registry/database entries
- Menu entries and shortcuts
- Environment variables

A software application is much more than just the executable file you run to start the application.
(On Unix/Linux use `ldd` to print shared library dependencies for an executable).

2.1.2 Licensing

- Open Source/Free software
- EULA's
- Product keys
• Dual licensing
• Product activation
• Time-restricted licenses
• Per user/Per host licenses
• License servers
• *Keeping track of license use ...*

You can make a lot of money just by being a software licensing expert and doing consultancy services ...

### 2.2 Software Install Issues

Developers spend hundreds of labor-months building that great new three-tier application. However, I wouldn’t be surprised if you’re reading this book because you’re the person who has to figure out how to install the application while everyone else is out celebrating the fact that they’ve finished it.


A typical application is distributed (e.g. client-server structure) including some middleware (thereby “three-tier”). Installing such an application is about much more than copying a few files.

#### 2.2.1 Install Types

• New installations
• Patches
• Minor upgrades
• Major upgrades/Service packs (*incremental or cumulative*)
2.2. SOFTWARE INSTALL ISSUES

There is no standardized terminology, but we might be talking about replacing just single files, or replacing an entire package or replacing a set of packages (service packs). Incremental means that previous service packs version have to be installed first while cumulative means they are included in the upgrade.

2.2.2 File Placement

- Local
  - C:/Program files
  - /usr, /usr/local
- Mounted from server
  - P:
  - /progs
- Advertised via shortcuts
- (Web applications, Terminal-server applications, ...)
- And what about all the shared files...

As with everything in system administration, we should strive for keeping it all standardized, meaning you will probably want to follow your own systems policy more than the software developers policy for placing files.

If you use the “good old approach” with installing much software on a network-mounted filesystem (a NFS/CIFS share), this might not be a good solution in todays mobile world. It raises the question of what to do with typical local changes such as registry- and menu-entries. Software installs always require some local configuration, so centralizing a software install to a network mounted file system does not necessarily save you any work if local changes still have to be made.
Advertised software is *installation-on-demand* where the software is not installed until it is attempted used for the first time. This is more common on Windows than on Unix/Linux.

### 2.2.3 Dependencies

- Two approaches:
  1. Dependencies between packages
  2. Dependencies between files
- DLL (and shared objects) hell

Dependencies between packages is the common approach on Linux (at least with the apt system), this is not common on Windows, Windows deals mainly with dependencies between files.

**DLL Hell** refers to the problematic situation where you want to install a software package where a certain DLL (shared library, .so in Unix/Linux) is required but this will overwrite a previous version of the same DLL which is required by a different package. This can be resolved with side-by-side installation.

### 2.2.4 Modified Configs

- *How do you deal with updating/upgrading a software package where the user has made her own modifications/additions to the configuration?*

Normally the only alternative is to retain the users version or overwrite, so if you are automating the install process you have to make this policy decision (probably by discussing this with your colleagues and boss first).

### 2.2.5 Services

- *What if you are installing a package which depends on other services that might be or might not be running?*
Installation of a software package might cause services to be stopped during installation and then restarted again together with the newly installed service.

2.2.6 Files in Use

- Will you have to replace files that are currently in use (open)?

Then of course you have to try (or get the user) to shutdown the application using the files, or if not prepare for replacement and schedule a reboot to replace the files during the reboot.

(on Unix/Linux, check out the man pages for `lsof` and `fuser` to deal find out if files are in use and by which processes).

2.2.7 Target Users

- Who are the target users?
  - a single user
  - a group of users
  - an administrator
  - is it an application where some users should have different access rights from others?

You might be installing software where some users should have administrator privileges for the application while others have reduced privileges and some might even just be guest users.

2.2.8 Updates

*Windows patch cripples XP with blue screen, users claim*

Angry customers blame MS10-015 for Blue Screen of Death, XP reboot hell

*My Digg Rektor*

February 17, 2010 9:01 AM PT

CompuServe - Recommended 64 128 256 512 768 1024 1280 1600

Complaints began early yesterday, and gained momentum throughout the day.

"I updated 11 Windows XP updates today and restarted my PC like it deserves me," said a user identified as "Tmiq" who kicked off a general support thread. "From then on, Windows cannot restart again. It is stopping at the blue screen with the following message: 'A problem has been detected and Windows has been shut down to prevent damage to your computer.'"

[http://www.computerworld.com](http://www.computerworld.com)
• Where are all the files to be updated?

• What if the update fails?

• Why are updates different from fresh installs?
  – host is in usable state
  – host is in an office
  – no physical access required
  – host is already in use
  – host may not be in a “known state”
  – host may have “live” users
  – host may be gone
  – host may be dual-boot

There has to be a way to locate the files installed from previous versions of the package, and during an update process everything have to be carefully backed up so it can be restored if something fails (e.g. a power shutdown while copying files).

2.2.9 Uninstall

• Installations have to transactional/atomic in behaviour

• There should also be a transactional uninstall

• Should the uninstall remove the dependencies that were installed together with the original installation?

Removing dependencies that were installed together with the original installation is also sometimes called cascading package removal.

2.2.10 Repair

• Should automatic repairs be performed if a file in an installed package is deleted?

This is common behaviour in some situations on Windows (performed by the Windows Installer service). Its useful but maybe not always good since you might want to delete a specific library as an ad hoc fix to a security problem until a proper patch is ready (on Windows you would have to modify the msi package to exclude to library, then reinstall it, to avoid automatic repair).
2.2. SOFTWARE INSTALL ISSUES

2.2.11 Environments

- Multiple operating systems
- Multiple versions of the same operating systems
- Multiple configurations of the same library environments

You probably need to have several different packages for different environments, maybe the application you are in charge of installing will be installed on both Windows laptops and Windows mobile phones?

2.2.12 Integration

Is it only Microsoft software on a client host?

- MS Office, MS Internet Explorer, MS Visual Studio, ...
- Adobe Photoshop, Illustrator, InDesign, Acrobat, Flash
- Mozilla Firefox, Thunderbird
- Opera
- Matlab
- Mathematica
- MiKTeX
- Symantic Internet Security

This is the really big issues: how do we deal with all different kinds of software which is packages each in its own way: Yes, we have to repackage, but that’s not always easy and not always an option. Probably the answer is a combination of repackaging and writing manual wrapper scripts.

Do note that if this was the list of software to be installed on both Windows and Unix/Linux, the situation would only be slightly better on Unix/Linux. It would be slightly better because it would be easier to repackage some of the commercial software as packages, but it wouldn’t be much better because we can’t find Adobe, Matlab etc together with the free software we are used to finding in the software repositories.
2.3 Package Management

Software package management is NOT a solved problem! This screenshot is from a workstation at Gjøvik University College in 2009.

We would like all software package in the same format, with the same mechanisms. This would allow us to keep a consistant database of installed software, allowing for:

- Version control
- Updates/Upgrades
- Rollback
- Dependency resolution
- Inventory overview
- Package grouping

DEMO: Get-WmiObject -class Win32_Product vs Secunia PSI or Putty install

2.3.1 Pkg Man/Installer

- Performs installations/removals/updates/...
- Sometimes comes bundled as part of package
- Maintain database of installed software
2.3. PACKAGE MANAGEMENT

- Unix/Linux: dpkg, rpm
- Windows: Windows Installer (msiexec)

A classical windows installer is a small program which is part of the compressed package you are going to install. More recently Windows also have a separate installer program which comes with the operating system called Windows Installer (msiexec).

An installer on Unix/Linux is commonly called a low-level package manager instead of an installer. As always there is no standard terminology.

2.3.2 Pkg Man System

- Dependency resolution
- Uses local package manager
- Checks for updates
- Unix/Linux: apt, yum
- (Windows: Windows Update/WSUS/pkgmgr *(not really...), Getit)

A local package manager/installer commonly stops/aborts installation if dependencies/conditions of the install package is not met. The package management system checks these requirements before installing, and resolves these by fetching additional packages needed and reordering the installation sequence of packages in such a way that the installer will not complain.

General workings of a package management system:

1. Download information about available packages from a remote repository
2. Check signatures of package lists
3. Decide whether packages should be installed or updated
4. Resolve dependencies
5. Download and install/update packages (incl checking signatures of packages)

(we sometimes talk about an application level package manager as well such as PERL’s CPAN)

DEMO:

cat /etc/apt/sources.list
aptitude update
aptitude safe-upgrade
2.3.3 Security

Installations are performed with high (root/administrator) privileges: We really need to trust the packages (and the source they come from) we are installing!

And of course, this means using cryptographic hashes and digital signatures, hashing and signing maybe repository index metadata (root metadata), package metadata and the package data itself.

Replay Attack

If the attacker is MITM, she can serve an old version of the repository even though the root metadata is signed ...

Protect by making sure you don’t accept metadata older than what you already have.

Freeze Attack

If the attacker is MITM, she can avoid updating the repository ...

Protect by limiting how long signed root metadata is valid.

Metadata Attack

If metadata is not signed (not root nor package metadata), MITM can easily offer newer versions of packages which are actually older version (with vulnerabilities the attacker know how to exploit) ...

Protect by requiring signed metadata.

DOS Attack

As root metadata the MITM attacker will just serve an endless file ...

Protect by monitoring system resources, setting hard limits or possibly by keeping package management cache on a separate partition.

2.3.4 Protection

- Consider using distributions who maintain control over their software repositories
- Maintain your own software repository for your infrastructure
2.4 WINDOWS

The problem is that its too easy to become a repository maintainer (mirror a repository) for many distributions, and as soon as you are a mirror you can initiate the mentioned attacks if the distribution is vulnerable.

2.3.5 Good Practice

Two common scenarios:

1. Warn if updates available (servers)
   - pre-defined list of software

2. Autoinstall updates (workstations/mobile devices)
   - user might install software
   - *One, Some, Many* technique of Limoncelli, Hogan and Chalup

In the second scenario we should audit the package database regularly, both for security and policy compliance reasons and for practical reasons. The user might install known malware by accident, or install software not approved by the organization (e.g. filesharing software). The host might have to be reinstalled, and having a recent list of installed software will ease this process.

The one, some, many technique of Limoncelli, Hogan and Chalup: start by installing/updating/upgrading a single host, then do it on your own workstation and you colleagues, then on an entire lab, then an entire user group, etc ...

2.4 Windows

Let’s take a look at Windows Installer in the lab ...

2.5 Unix/Linux

Let’s take a look at dpkg in the lab ...

A nice feature of dpkg is that you can replicate hosts by doing a
dpkg --get-selection > file
on the original host, and
dpkg --set-selection < file
and given the same distribution they will get the same set of software packages.

Note: when using *aptitude* search for searching packages and you want to understand what the flags to the left mean, the following link might be helpful (e.g. to understand that ‘v’ means ‘virtual’):

http://algebraicthunk.net/~dburrows/projects/aptitude/doc/en/ch02s02s02.html
2.6 Theory questions

1) Describe at least three key issues in Software Licensing.

2) What can a package manager do if a package is about to replace configuration files which a user have modified manually?

3) What is meant by “transactional behaviour” of package managers/ installers?

4) What is the goal of software package management?

5) Describe the how a typical package manager works (e.g. what happens when you run `aptitude install gcc`).

6) Describe how a “freeze” attack against a package manager can be performed. How can you protect against this attack?

7) Describe how a “metadata manipulation” attack against a package manager can be performed. How can you protect against this attack?

8) Why would you want to maintain your own software repository in your infrastructure (and direct all your clients package managers to this repository)?

9) Describe the “one, some, many” technique of software updates.

10) What is the difference between “Windows Update” and “Windows Server Update Services” (WSUS)? How are they related?
2.7 Lab exercises

In this lab we will download source code for a software package (Cfengine on Linux and Putty on Windows), compile it and create a package of the resulting binaries. On Windows we will create a msi package and on Unix/Linux we will create a deb package.

REMEMBER: you might be frustrated in this lab with all the seemingly unnecessary details you have to deal with, and you will be tempted to just point-and-click something instead. But you are not doing this lab just for installing this software package once or twice on a single host, you are doing it to manage software packages on 2200 hosts repeatedly at least twice pr year. Think Scalability!

2.7.1 Preparing your environment

Unix/Linux: Make sure you have some recent libdb-dev, libssl-dev and libpcre-dev packages (cfengine depends on BerkeleyDB database, OpenSSL crypto and Perl compatible regular expression libraries) in addition to tools needed for compiling (autotools-dev, fakeroot, dh-make, build-essential, flex and bison).

Windows: Make sure you have a development environment so you can compile C/C++ programs. You can probably use visual studio, the sample solutions have been compiled with MinGW (http://www.mingw.org, choose at least gcc and make when installing).

2.7.2 Download and compile

Unix/Linux: Download most recent version of cfengine from http://www.cfengine.org. You do not have to compile cfengine now, it will be compiled as part of the package creation process.

Windows: Download most recent version (source code) of Putty from http://www.chiark.greenend.org.uk/~sgtatham/putty/ and compile it.

2.7.3 Create software package


Windows: Download and install WiX on Windows (http://wix.sf.net). Install WiX with msiexec from the powershell command line. Follow the WiX tutorial on http://www.tramontana.co.hu/wix/lesson1.php
and create a putty-0.60.msi package which installs the executables putty.exe and pscp.exe in the directory C:\Program files\Putty. You can get GUIDs from http://www.guidgen.com. You might also need to read http://wix.sourceforge.net/manual-wix3/add_a_file.htm.

2.7.4 Query the package

Query the packages you have created to see which files will be installed (use dpkg-deb on Unix/Linux and google if you can’t use msiexec on Windows).

2.7.5 Install software package

Install the packages you just created with msiexec on Windows and dpkg on Unix/Linux.

2.7.6 Query software package database

Query the package databases to verify that the packages were successfully installed. For a few of the files, check that they have actually been installed to their intended locations (and check that their size is not zero...).

2.7.7 Remove package and reinstall it

Remove the packages with msiexec on Windows and dpkg on Unix/Linux. Verify that they have been removed by querying the package databases again. Install it again (too see that actions really can be reversed and repeated).
Chapter 3

Automatic Installations

3.1 Install Sequence

1. Booting a kernel
2. Select language and keyboard
3. Partition and mount disks
4. Install kernel and driver modules
5. Configure the network interface
6. Install base system
7. Make the system bootable
8. Reboot
9. Configure system time
10. Set root password and create user accounts
11. Select package locations
12. Select packages to install (and install them)

In the simplest systems (e.g. Ubuntu) this process is shortened to the minimal visible steps of

1. Booting a kernel
2. I18n & l10n (internationalization and localization)
3. Partition disks
4. User accounts
3.2 Automated Install

- Saves time
- Consistent behaviour (even on a small scale)
- Forces documentation of configuration/inventory

Remember that it’s not just how many hosts that need installation that you have to think about but also the frequency: how often they need to be reinstalled: reinstalling 10 hosts each month is as many installations in total as installing 120 hosts each summer.

3.2.1 Approaches

- Disk-image based (dd-nc, ghost, ...)
- Configuration-file based (FAI, preseed, ...)
- A combination (A hybrid approach) (RIS, WDS, ...)

The problem with disk-image based approach is that there might be unique parameters in software on each (not all hosts can be identical Linux with DHCP). The most obvious example is if the host uses static IP addresses (hosts cannot have the same IP address ...). A more specific problem is on Windows hosts which have one or more unique SIDs (Security IDentifier). Solutions which uses disk-image based approaches usually includes some tools for dealing with creating new SIDs in the process. There is some controversy whether new SIDs really are needed, see:


The disk-image based approach is also useful to know about since its a basic forensic technique. In forensics you always want to capture the exact copy of a disk image and do you investigation on that copy, not on the original harddrive.

Configuration-file based means that we are not installing a disk-image, we are doing a normal OS install, it’s just that we automate the entire process by providing predefined answers to all questions we normal have to answer during the install process.

3.2.2 Level of attention

- Either the process is unattended or it is not!

There is a very big difference between not having to visit your computers at all, and having to visit them just to press enter once. The process should be completely automated, even rebooting.
3.3 dd-netcat

Sometimes it can be wise to insert in the pipeline a `gzip -cf` on the server and a `gzip -dcf` on the client. When is this a good idea? This depends on the computational power vs the bandwidth. It is wise if you have a low bandwidth connection.

3.4 Preseeding
Notice that it’s the firmware on the NIC (network interface card) which runs first.

Consider why DHCP uses broadcast instead of unicast. For DHCPDISCOVER and DHCPOFFER, it is necessary based strictly on the fact that the client and the server don’t know or have IP addresses. For DHCPREQUEST and DHCPACK it’s the additional argument that its nice that other DHCP servers present are notified which DHCP server the client chose.
TFTP is a very simple lightweight protocol which transfers files over UDP one segment at a time (slow and simple).

PXELinux (a boot loader) runs and downloads its own configuration file, looking for this file in the special search sequence ending in a file named default (if it does not find any of the other files).
PXELinux downloads a kernel and an initrd (initial ram disk).

PXELinux starts a kernel with several boot parameters, one of them being the preseed file. This can be downloaded with other protocols than http, but http is commonly used. The kernel then performs the install with all the answers fetched from the preseed file (probably including download all software packages from repositories).
3.5 Theory questions

1) Describe the three main approaches to automatic/unattended installations.

2) What is meant by “level of attention” during an automatic/unattended installations?

3) Describe the main steps in installation of an operating system.

4) Describe the contents of the configuration file involved in automatic/unattended installations of operating systems (e.g. the preseeding file for Debian/Ubuntu, or the unattended.xml file in WDS).

5) Which problems can you encounter when doing disk-image based installation? Discuss with respect to different setups of different operating system.

6) Describe the process of a preseeded automatic install (including which firmware/software runs on the client at any given time, and which servers are contacted at each stage).
3.6 Lab exercises

We will now start working with virtual machines (vm’s) in a network. For this lab you will download prepared vm’s from [http://www.hig.no/~erikh/vmware](http://www.hig.no/~erikh/vmware). The username is always data (and/or root/administrator) and the password always poL+VoT2.

A few things that are good to know about vm’s in vmware workstation:

- When you have downloaded a vm, unzipped it (tar zxf file.tgz) and opened it in vmware, always choose “I moved it” when vmware asks you if you have moved it or copied it (if you answer “I copied it” vmware will change the mac address which might screw up our lab setup a bit since it will rename the network interface).

- You might need to press F2 for BIOS to change boot order so you can boot from cd.

- We will sometimes use ordinary vmare-supplied DHCP in vmnet1 but most times vmnet2 where we have our own dhcp server, so make sure you have set up vmnet2 with subnet 192.168.25.0 and without DHCP. You can set this up in vmware workstation by choosing the “virtual network editor” in one of vmware workstations menus, the “edit” menu if you are on Linux.

- When booting with knoppix as CD write knoppix lang=no at boot prompt or write loadkeys no at the shell prompt if not in X window system.

3.6.1 Preparation

Download knoppix(iso), drone, cube and cube-clone, unzip drone, cube and cube-clone and open them in vmware. Check that you can boot cube but not cube-clone or drone.

Boot both cube and cube-clone from knoppix iso-image (in the vmware configuration for each virtual machine, choose to mount the knoppix iso file as the CD-ROM drive) and open a terminal window on both of them. (we are now going to copy the entire content of the disk of cube to the disk of cube-clone)

3.6.2 Disk-image based: disk cloning (dd/netcat)

Cube have static IP address 192.168.25.10, configure cube-clone with static ip address 192.168.25.11 with

```
sudo ifconfig eth0 192.168.25.11 netmask 255.255.255.0
```

Start listening for datatransfer on cube-clone with

```
nl -l -p 12345  | sudo dd of=/dev/sda bs=16M
```

and initiate datatransfer from cube with
3.6. LAB EXERCISES

```bash
sudo dd bs=16M count=192 if=/dev/sda | nc 192.168.25.11 12345
```
(this will take a few minutes).

Reboot cube-clone without knoppix and verify that you can log in. Note: they now both have the same IP address (so this is usually only done for dhcp clients), and also be aware that on the cloned cube eth0 might be renamed to eth1). You have now successfully done an automatic install by disk cloning :)

### 3.6.3 Configuration-file based: preseeding

We are now going to do automatic install of a client computer drone by debian’s preseeding mechanism. Delete cube-clone (the one you have installed by disk cloning), and unpack it again. Change network for both cube, cube-clone and drone to vmnet2 (where vmwares dhcp server is not present).

Boot cube. (If there’s no network, set up networking with)

```bash
sudo ifconfig eth0 IPADDRESS netmask 255.255.255.0
```

```bash
sudo route add default gw xxx.xxx.xxx.2
```
(find out gw, netmask and IP from vmware’s setup of the vmnet2 network).

The packages isc-dhcp3-server, tftpd-hpa and apache2 have been installed on cube.

Check if you need to edit the dhcp3-server config file to reflect your network settings and restart the dhcp server.

The dhcp server on cube is not started when cube boots, so start it manually.

View the configuration file of tftpd-hpa.

```bash
cd /masterfiles/tftpboot and view the contents of the directory.
cd pxelinux.cfg and rename the two files starting with 01 to reflect the MAC address of cube-clone and drone.
```

Check if you need to edit these same two files to correct the boot parameter of the auto-labeled kernel to have the right IP address of the webserver containing the preseeding files.

```bash
cd /var/www and view the contents of the two preseeding files.
```

Boot cube-clone and drone and press F12 to do a pxeboot which will trigger the fully automatic installation.
Chapter 4

Configuration Management: One Host

4.1 Introduction

- We’ve built and deployed hosts: now let’s maintain them
- Convergence and stable state
- Stable state is defined by policy
- Cfengine is a declarative language for enforcing policy

We now know how to configure OS, install apps and automate the process, its time to see how we can maintain hosts during their lifetime. Our goal is that the “stable state” that the hosts have after our automatic installs should be maintained during the hosts life. In other words we want to do maintenance with convergent operations in such a way that a host that drifts away from its stable state will be moved closer to its original stable state. Two examples:

**Startup programs/helper toolbars/add-ons** which causes the host to use five times its original startup time. If our policy states that unknown applications should not be part of startup, we make sure these file/registry entries are not present.

**Disk usage** a disk that fills up causes poor performance for the host. If our policy dictates rotation and compression of log files, deletion of old cached data, etc, this can be enforced to bring the host back to a more stable state.

A declarative language is a language where you state the goals (what you want to achieve) instead of how to get there, as opposed to an imperative language (such as Perl, C, C++ and Java) where the focus is on the algorithm (how to do it).
For alternative systems to cfengine see
and

4.1.1 Host life cycle

New is new machine, Clean is a fresh install with no localization, Configured is correctly configured and operational environment, Unknown is misconfigured or outof date, Off is retired and turned off.

Build and initialize is usually one step.

Entropy is introducing noise, e.g. unwanted start-up programs or out-of-control internet cache files which slow down the host, which is fixed by maintenance (Debug) or by a fresh install

A host needs to be updated to stay in its proper configured (stable) state.
4.1.2 Cfengine


History: Cfengine 1 (original trust based on IP ala rsh), 2 (security) and 3 (improved language).

4.1.3 Hello world

```plaintext
body common control {
    bundlesquence => { "hello" };  
}

bundle agent hello {
    reports:
        linux|windows::
            "Hello world!";
}
```

To run a cfengine script file.cf, you either do cf-agent -f ./file.cf or you can do the following:

1. cp file.cf ~/.cfagent/inputs/promises.cf
2. cf-agent
In other words, cfengine by default runs the script in its own input directory which is
/.cfagent/inputs for ordinary users and /var/cfengine/inputs (possibly relocated
elsewhere on some distributions) for root.

TAVLE: Kjernen i cfengine3:

```
promise type:
   classes::
      "promise"
      parameters/attributes;
```

### 4.1.4 Syntax

- A cfengine script is divided into sections which are either bodies or bundles
  (cfengine is case-sensitive).

- Similar syntax to bash/perl:
  
    - sections are contained within `brackets`
    - `commas` separate parts of the same promise
    - promises are ended with a `semicolon`
    - body part lines end with a `semicolon`
    - variables are identified by `$` and usually contained in `brackets` to separate
      them from surrounding text
    - most user defined information is contained within double quotations.
    - `#-comments` or `comment` => "My comment" if you want them printed.
    - promise type end with `colon`
    - classes end with `double colon`

### 4.2 Body

`body` `TYPE` `NAME`

- Body control promises

- Body parts: collection of parameters for promises, so the promises themselves can
  be kept simple (easy to read: easy to link to policy)
4.3. BUNDLE

4.2.1 Control

- body common control *(with the bundle sequence)*
- body agent control (for cf-agent)
- body server control (for cf-serverd)
- body monitor control (for cf-monitor)
- body runagent control (for cf-runagent)
- body executor control (for cf-execd)
- body knowledge control (for cf-know)
- body reporter control (for cf-report)

All details can be found in the cfengine3 reference manual.

4.2.2 Body part

Hides the details of a promise, e.g. a file copy promise might have a parameter

```plaintext
perms => myperms("0700");
```

where we have hidden the details:

```plaintext
body perms myperms(p) {
    mode => "${p}";
    owners => { "root" }; 
    groups => { "root" }
}
```

e.g. copy parameters

4.3 Bundle

bundle TYPE NAME

- Group promises together in a bundle
- TYPE is who the bundle is intended for, e.g. body agent backups (for cf-agent)
4.4 Promise Type

- Define the types of actions cfengine can perform
- The most general one is commands which we can use to call bash and powershell

4.4.1 Any Bundle

- vars
- classes
- reports

Promise types were called actions in cfengine2.

vars A promise to be a variable, representing a value.

classes A promise to be a class representing a state of the system.

reports A promise to report a message.

4.4.2 Agent Bundle

- commands
- databases
- files
- interfaces
- methods
- packages
- storage

commands A promise to execute a command.

databases A promise to configure a database.

files A promise to configure a file, including its existence, attributes and contents.

interfaces A promise to configure a network interface.

methods A promise to take on a whole bundle of other promises.

packages A promise to install a package.

storage A promise to verify attached storage.
4.4.3 Other

- access
- measurements
- roles
- topics
- occurrences

**access** A promise to grant or deny access to file objects in cf-serverd.

**measurements** A promise to measure or sample data from the system, for monitoring or reporting in cf-monitord (Cfengine Nova and above).

**roles** A promise to allow certain users to activate certain classes when executing cf-agent remotely, in cf-serverd.

**topics** A promise to associate knowledge with a name, and possibly other topics, in cf-know.

**occurrences** A promise to point or refer to a knowledge resource, in cf-know.

4.5 Promise

cfengine3 reference manual, section 2.1:

Everything in cfengine 3 can be interpreted as a promise. Promises can be made about all kinds of different subjects, from file attributes, to the execution of commands, to access control decisions and knowledge relationships.

We won’t get comfortable with this before we get to some examples...
Also read about ordering and link to bundlesequence.

4.6 Classes

There’s no if-statement in cfengine, decisions are made using classes:

- *Hard classes* (built-in) and *soft classes* (user defined)
• Classes can be combined with operators (listed in order from highest to lowest precedence)

() grouping

! NOT

. AND (alternatively &)

| OR (alternatively ||)

DEMO:
cf-promises -v | grep classes

Classes are defined in bundles. Bundles of type common yield classes that are global in scope, whereas in all other bundle types classes are local.

### 4.6.1 Time

• Yr2009

• January

• Day1

• Monday

• Hr00

• GMT_Hr00

• Q1

• Min00_05

• Min00

### 4.6.2 Host

• hostname

• operating system / distribution

• architecture
4.6.3 Network

- ipaddress
- subnets
- net interfaces

4.7 Variables

- Dynamic typing:
  - int
  - real
  - string

- Scalar
- List
- Associative array

Scalars, Lists and Associative arrays are the key variable types in most scripting languages.

4.8 Loops

```plaintext
body common control {
    bundlesequence => { "example" };
}

bundle agent example {
    vars:
        "component" slist => { "cf-monitor", "cf-server", "cf-exec" };
    commands:
        "/bin/echo /var/cfengine/bin/$(component)";
}
```

Will run the /bin/echo promise three times
4.9 Locks and Splay

`ifelapsed` minutes to wait before next run

`expireafter` how long cf-agent should wait before killing an old instantiation of itself

`splaytime` wait a random amount of time (from zero to splaytime) before initiating

Load distribution and defending against DOS attacks.
4.10 Theory questions

1) Briefly explain the computer states New, Clean, Configured, Unknown and Off (in Evard’s life cycle model).

2) What is the difference between an imperative and a declarative programming language? Give examples of each language-type.

3) Describe the general syntax of cfengine promises and how they are included in bundles.

4) What is a “body part” in cfengine? What is its purpose?

5) What is the difference between hard and soft classes in cfengine?

6) What is the purpose of using ifelapsed in cfengine?

7) What is a scalar, a list and an associative array?

8) Write a cfengine script which checks if the ssh server (sshd) is running. If it is not running, the script should restart it. (Hint: focus on structure of the script, I dont expect you to get the exact syntax right.)

9) Write a cfengine script which does backup of /home/data to /backups/home/data (this is a typical backup where /backups is mounted from your file server, this protects against harddrive failure or laptop theft).

10) Write a cfengine script which warns if there is less than 20% free disk space on the volume which /home resides on (note: /home might be in a separate filesystem).
4.11 Lab exercises

The goal of this lab is to familiarize ourselves with the language of cfengine. Cfengine is a sysadm scripting language, but very different from powershell, bash, perl and python. We will not go far into cfengine details but primarily focus on using cfengine as a wrapper for powershell and bash.

Note: cfengine3 has a completely new language compared to cfengine2, so dont pay too much attention to articles and tutorials you find about cfengine2. They are nice to read, but the syntax would be very different in cfengine3.

Whenever you write a cfengine3 script myscript.cf3, check that the syntax is ok with cf-promises -f ./myscript.cf3 and execute it with cf-agent -f ./myscript.cf3 (add -v for verbose output or -d for debug style output)

4.11.1 Prepare environment

Make sure cfengine3 is installed by executing cf-agent -V. Check that keys have been created (you should find a few key files in /var/cfengine/ppkeys), if not run cf-key to create keys. Check that /var/cfengine/bin contains all the cfengine binaries (cf-*)

4.11.2 Hello world

Write and execute a cfengine3 hello world program hello.cf (feel free to make it say something else than “hello world”) using cfengine’s promise type reports. Make it apply to all hosts of class linux or windows. Expand it with the control parameter ifelapsed in such a way that it will run whenever you execute it and you don’t have to wait a minute between each run.

4.11.3 Hello world through powershell and bash

Repeat the exercise above, but instead of using reports, use commands as a wrapper around a call to powershell’s Write-Host and bash’ echo. Make sure you separate between classes of hosts in such a way that powershell is being used on Windows hosts and bash on Unix/Linux hosts.

4.11.4 Hello world through powershell and bash with time classes

Repeat the exercise above in such a way that the Write-Host is executed on Windows only on Tuesdays between 10:00 and 10:30, and echo is executed on Linux only if it is September and not between 10:00 and 10:30.