Inhoudsopgave

Voorwoord

Als laatstejaarsstudent Industrieel Ingenieur in de Elektronica aan de KHLim kreeg ik de kans een onderzoek te doen rond clustering filesystems in Linux. Ik heb deze kans met beide handen aangenomen om zo meer kennis en ervaring op te doen rond het steeds populairder wordende besturingssysteem Linux.

Graag zou ik mijn promotor Leo Rutten willen bedanken voor deze kans en ook voor zijn nodige technische hulp en voor de goede opvolging van mijn eindwerk.

Ook bedank ik mijn ouders omdat zij mij de mogelijkheid en de middelen gegeven hebben om het diploma van industrieel ingenieur te behalen.

Tot slot bedank ik iedereen die ik niet in dit voorwoord opgesomd heb maar die mij toch gedurende dit laatste zware jaar de nodige steun en hulp geboden heeft.

Inleiding

De opdracht voor dit afstudeerwerk bestaat erin een concreet en duidelijk naslagwerk te creëren waarin alle installatie- en configuratieprocedures voor cluster filesystems duidelijk terug te vinden zijn, samen met alle essentiële theoretische uitleg nodig om deze Linux-toepassingen naar opbouw en gebruik te leren.
In de ICT-sector duiken begrippen zoals Linux, clustering, Open Source software steeds vaker op. Mijn interesse was al eerder gewekt, en mijn promotor, ing. Leo Rutten gaf me de kans met dit onderzoekswerk mijn belangstelling en basiskennis over het besturingssysteem Linux te verruimen, te verdiepen en voor anderen beschikbaar te maken.
Het is de bedoeling dat een ‘à la carte’ werkinstrument wordt geschreven waarin de gebruikers van cluster filesystemen vlot een antwoord kunnen vinden op hun concrete vragen. We zullen ook een verklarende woordenlijst aanleggen met uitleg bij de vele Engelse termen waarop de lezer bij zijn lectuur over dit onderwerp onvermijdelijk botst.

Hoofdstuk1: Situering van het eindwerk

1.1 Het besturingssysteem Linux

Wat is Linux, waar komt dit besturingssysteem vandaan, en hoe is het op vrij korte tijd in het ICT-wereldje zo populair kunnen worden?

Dit is Tux, de mascotte van Linux

1.1.1 Korte geschiedenis

Eén van de bekendste besturingssystemen in de wereld is Unix. Dit besturingssysteem dat reeds populair was in de jaren zeventig kreeg het moeilijk in de jaren negentig toen Windows NT op de markt kwam.

Ondanks deze concurrentie heeft Unix toch stand kunnen houden dankzij zijn talrijke voordelen in vergelijking met Windows. Vooral als het op beveiliging aankwam was Unix een stapje voor. Deze troef heeft veel bijgedragen aan de ontwikkeling van de huidige Linux, die later voortvloeide uit Unix.

Linux werd geschreven door Linus Torvalds die in 1991 aan de universiteit van Helsinki in Finland studeerde. Hij experimenteerde op deze universiteit met Minix, één van de eerste Unix-gebaseerde besturingssystemen voor gewone PC's die, speciaal voor ontwikkeling, gratis ter beschikking van de universiteiten werd gesteld. Op dat ogenblik bestond er geen enkele Unix-versie die op een klein systeem kon draaien. Omdat Torvalds niet over een mainframe of microcomputer beschikte besloot hij een kernel te schrijven die er in kon slagen om de multi-tasking eigenschap te behouden en dat op een 386 processor. In 1994 maakte Linus Torvalds ook een eerste Linux kernel voor Alpha-systemen. Na verloop van tijd werd er voor bijna ieder platform een Linux versie ontwikkeld.

Door zijn kernel over het internet te verspreiden en de mening van andere Unix-experts te vragen zorgde hij voor een echte ontwikkelingshype. Hieruit vloeiden heel wat toepassingen voort voor zijn systeem en werd er een goed ondersteund besturingssysteem geboren.

Vandaag de dag is Linux een zeer populaire Unix-variant die zelfs door grote bedrijven ondersteund wordt en een steeds grotere schare aanhangers kent.

1.1.2 Waarom mensen voor Linux kiezen en niet voor Windows?

Zoals bekend, heeft Linux veel voordelen ten opzichte van Windows:

- Het besturingssyteem is gratis te downloaden;
- Het is open source software;
- Het systeem is platformonafhankelijk;
- Het systeem is zeer stabiel;

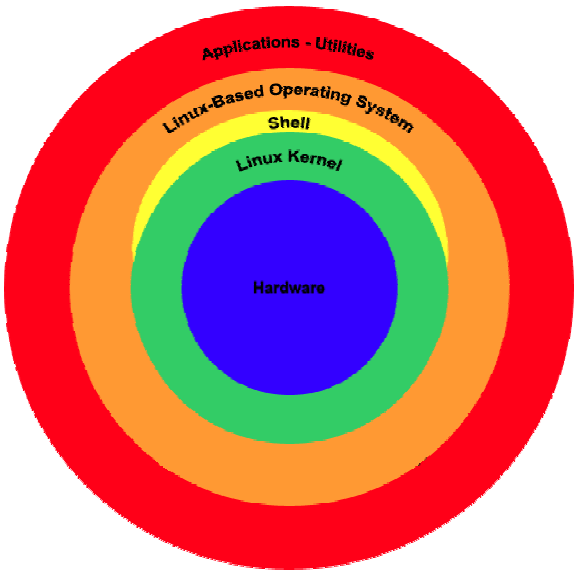
Dit eerste voordeel spreekt voor zich, Linux is volledig legaal gratis te downloaden van het Internet wat helemaal niet van Windows of andere besturingssystemen kan gezegd worden.

Open source betekent dat wanneer u Linux downloadt, u er ook gratis de broncode bijkrijgt zodat u zelf wijzigingen in het systeem kan aanbrengen. Hiervoor dient u wel een zeer grondige kennis te hebben van programmeren in C.

Linux is ook bijna volledig platform-onafhankelijk. Dit is toch wel één van de grootste voordelen aangezien u hierdoor Linux kan gebruiken voor heel wat verschillende toepassingen. Windows daarentegen ondersteunt enkel Intel 386 processoren en hoger.

Linux is door zijn speciale opbouw ook een zeer stabiel besturingssysteem. We werken namelijk met een kernel die bijna 100% crashfree is. Deze kernel vormt een link tussen de programma's die we draaien en de hardware. Bovenop de kernel draait dan een interfaceprogramma, dit is een programma van waaruit men processen kan starten, commando's kan uitvoeren, … . Dit kan tekstueel (bvb. bash) of grafisch (bvb. X window) zijn. En pas hierop worden de programma's gedraaid. Dit heeft als voordeel dat wanneer er toch een programma mocht crashen, dit steeds kan worden afgesloten zonder dat we andere processen verstoren die op hetzelfde moment draaien.

Opbouw van Linux:



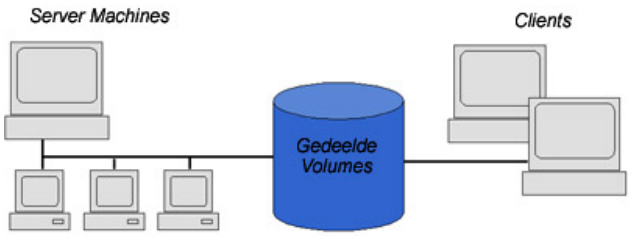
1.2 Cluster Filesystems

In dit onderdeel zal ik u met een korte uitleg proberen te verduidelijken wat een cluster filesystem is.

Cluster filesystems worden ook vaak distributed filesystems genoemd.
In de naam cluster filesystem zit enerzijds het woord cluster dat in deze context een verzameling of groep computers betekent die samenwerken om een hoger doel te bereiken en zich naar buiten toe voorstellen als één systeem. Anderzijds hebben we het woord filesystem dat bestandssysteem betekent. Eigenlijk betekent een cluster filesystem dus bestandssysteem verspreid over verschillende computers.
Een bestandssysteem bestaat uit de methode en gegevensstructuren die een besturingssysteem gebruikt om bestanden op een schijf of partitie bij te houden. Eigenlijk is dit dus de manier waarop bestanden op een schijf zijn georganiseerd.
Bij geclusterde bestandssystemen is het zo dat de bestanden over verschillende computersystemen verspreid zijn en vooral gebruikt worden in computernetwerken.

Deze technologie maakt het mogelijk dat meerdere clients een filesystem op een gedeelde harde schijf tegelijk kunnen benaderen. Via een snel intra-cluster netwerk wordt de toegang tot het gedeelde filesystem gesynchroniseerd. Cluster Filesystems zijn tevens meestal Journaling Filesystems, en in het geval van een server crash draagt één van de nog levende servers zorg voor het terugdraaien van de openstaande filesystem transacties.

Overzicht cluster filesystem:



Nu u bekend bent met een Cluster File System en het doel hiervan kent, kan ik alvast enkele belangrijke cluster filesystems opnoemen: CODA, Lustre, AFS (Andrew File System), AFS2, Veritas, GFS (Global File System), OpenAFS, Intermezzo, …
Hierbij zijn zowel CODA als OpenAFS, de systemen die ik zal gebruiken, ontstaan uit het originele Andrew File System.

1.3 Het Andrew File System

Daar de systemen die ik tijdens mijn afstudeerwerk gebruikt heb beide afstammelingen zijn van het Andrew File system vind ik het nodig uit te leggen wat dit systeem inhoudt en hoe het werkt. Ook zal ik kort de systemen beschrijven die ik gebruikt heb maar uiteindelijk blijft de filosofie achter AFS, OpenAFS en CODA dezelfde.

1.3.1 Wat is AFS?

Het Andrew File System is men beginnen te ontwerpen in 1983 aan de Amerikaanse Carnegie Mellon-universiteit en het werd er in 1985 voor het eerst in gebruik genomen. AFS is een uitbreiding op vlak van veiligheid en interoperabiliteit van het bekende Network Filesystem (NFS). Met dit systeem kunnen bestanden of programma's op andere computers in het netwerk worden gebruikt alsof ze op de lokale harde schijf staan.

Voordelen van dit systeem:

- caching mogelijkheid (op client, tussen 100 MiB tot 1 GiB)
- veiligheidsfeatures (Kerberos 4 gebaseerd, access control lists)
- eenvoud van adressering (je hebt maar 1 bestandssysteem)
- schaalbaarheid (voeg servers toe aan de cel indien nodig)
- communicatieprotocol

Om de voordelen van dit systeem aan te geven, verduidelijken we enkele van de voorgaande termen.

Caching: is het opslaan van veel gevraagde informatie om zo een sneller systeem te verkrijgen. Wanneer we bijvoorbeeld bij AFS een tekstbestand openen dat we reeds geopend hebben, dan zal er in de cache reeds een verwijzing naar dit bestand staan, zodat onze computer veel minder lang zal moeten zoeken naar dit bestand. Met AFS kunnen we de grootte van dit cachegeheugen kiezen.
Kerberos: is een authentificatieprotocol voor server-client applicaties dat ongeveer 15 jaar geleden ontworpen is aan het MIT (Massachusetts Institute of Technology). Kerberos wordt zelfs in Microsoft systemen zoals Windows 2000 gebruikt.
Access Control Lists: zijn lijsten waarin staat welke gebruikers bepaalde acties in een bepaalde directory mogen uitvoeren. Iedere AFS-directory beschikt over zo een ACL. Een ACL kan tot 20 verschillende entries voor gebruikers of groepen of een combinatie van beide bevatten.
Schaalbaarheid: Hiermee wordt bedoeld dat dit systeem gebruikt kan worden in een netwerk van twee systemen tot een zeer groot aantal systemen.

Nadeel van dit systeem:

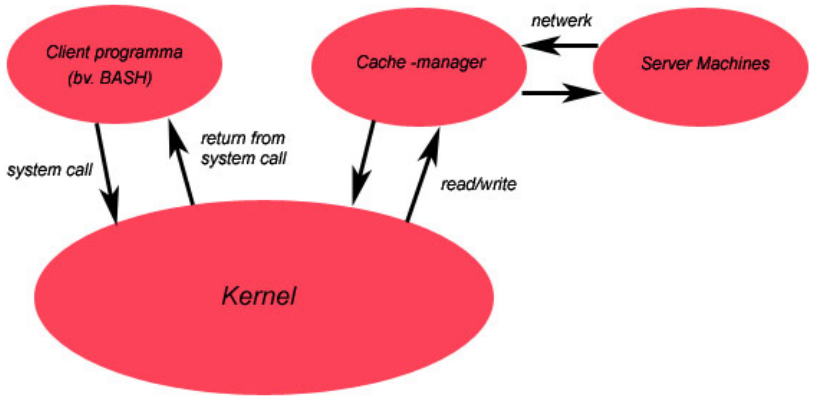
Het grote nadeel aan het originele AFS was dat het niet geschikt was voor Linux en enkel werkte onder Unix.

1.3.2 Hoe werkt een AFS-gebaseerd systeem?

Wanneer we met eender welk AFS-gebaseerd systeem werken, werken we met cellen en clients. Een AFS-cel is een groep van volumes. Elk volume is opgebouwd uit bestanden en directories die samengebracht zijn en een naam gekregen hebben. Al deze files en directories bevinden zich op servermachines. Iedere AFS-cel beschikt over zijn eigen lijst van gebruikers, groepen en systeembeheerders.
Op deze servers kan men bestanden plaatsen en wijzigen. Deze bestanden kunnen dan gedeeld worden over het volledige netwerk. Het grote voordeel van een AFS-gebaseerd systeem is dat deze bestanden allemaal onder dezelfde root-directory verschijnen. Het lijkt dus alsof ze op één server staan terwijl ze eigenlijk verspreid staan over verschillende servers.

Wanneer we inloggen met een client op ons systeem zullen we ons moeten authentificeren met een login en paswoord. Als we dan in de AFS-directory gaan kijken krijgen we een overzicht van alle bestanden en directories waar we toegang tot hebben.
Bij het uitvoeren van een request om een gedeeld bestand te openen zal de cache-manager automatisch de lokale cache controleren op de aanwezigheid van een kopie van dit bestand. Als er nog geen kopie bestaat zal de cache-manager een request sturen naar de AFS-servers, waarop hij het bestand toegestuurd krijgt. Hierbij ontvangt hij ook meteen een zogenaamde “call back” belofte in geval dat het bestand door iemand anders wordt aangepast. De cache-manager zet het bestand dan in de lokale cache op een FIFO-basis.
FIFO staat voor first-in-first-out, dit wil zeggen dat de eerste file die in de cache gezet is er ook als eerste terug uitgelezen zal worden.
Telkens wanneer we een bestand opslaan zal de cache-manager dit ook automatisch op de server zetten.

Schematische werking



Dit volledige proces is transparant, wat dit betekent is dat wij geen verschil merken tussen het werken in lokale bestanden of gedeelde bestanden. Enkel wanneer we een gedeeld bestand voor het eerst openen, dient dit nog eerst van de server afgehaald te worden, waardoor de wachttijd verhoogt. Deze wachttijd is bovendien ook afhankelijk van uw netwerkverbinding.

1.3.3 OpenAFS

Waarvoor staat OpenAFS?
Zoals de naam zegt staat het voor Open Source AFS dit wil zeggen dat het Open-source software is en dat de broncode openbaar is.
OpenAFS vloeide voort uit het originele AFS. AFS werd gecommercialiseerd in 1989 door Transarc en pas in 2000 werd de broncode door IBM openbaar gemaakt, waardoor OpenAFS geboren werd.

Daar OpenAFS dezelfde concepten en features als het originele AFS omvat, en ik deze reeds uitgelegd heb in het onderdeel over AFS, verwijs ik naar dit onderdeel voor meer info over het concept.

Omdat een OpenAFS systeem heel wat complexer is dan het originele AFS zou ik u graag meer uitleg geven over welke sytemen in een OpenAFS-cell kunnen bestaan, wat hun doel is en welke processen ze draaien.

Ook een OpenAFS-cell is opgebouwd uit servermachines. Deze servermachines kunnen een verschillende rol spelen in de cell naargelang welke processen ze draaien.

De verschillende processen die ze kunnen draaien zijn:

- **Files server proces**: levert data van de file servermachine aan clients als ze een request doen en schrijft de data weg als de clients data opslaan.
- **Basic OverSeer server proces (BOSserver proces)**: verzekert dat andere serverprocessen op de server draaien
- **Authentification server proces**: zorgt voor veilige netwerkcommunicatie door verificatie van identiteiten en authentificatie en onderhoudt ook de authentificatiedatabase.
- **Protection server proces**: helpt de toegang te beheren van gebruikers tot hun bestanden en directories. Maakt gebruik van de protection database.
- **Volume server proces**: voert alle bewerkingen met volumes uit. Het helpt bijvoorbeeld de beheerdervolumes van één server naar een andere verplaatsen.
- **Volume Location server proces**: onderhoudt de volume location database (VLDB), in deze database wordt bijgehouden waar volumes zich bevinden.
- **Update server proces**: verdeelt nieuwe versies van AFS process server software en verdeelt ook de configuratiebestanden naar alle servermachines.
- **Backup server proces**: onderhoudt de backup database, voorziet de mogelijkheid om volumes naar backup-tapes te schrijven.
- **Salvager proces**: wordt pas geactiveerd wanneer het fileserverproces of het volumeserverproces crasht. Het repareert de inconsistensies veroorzaakt door het crashen.
- **Network Time protocol Daemon**: synchroniseert de interne klok van een file servermachine met die van andere machines.

De verschillende soorten servermachines die er bestaan zijn:

- een simpele fileservermachine
- een database servermachine
- een binary distribution machine
- een system control machine (SCM)

Een simpele fileservermachine draait enkel de processen die zorgen voor het opslaan en verdelen van AFS-bestanden van en aan clients. Ook draaien ze processen voor het weergeven van processtatussen, en voor het afhalen van binaries en configuratiebestanden.

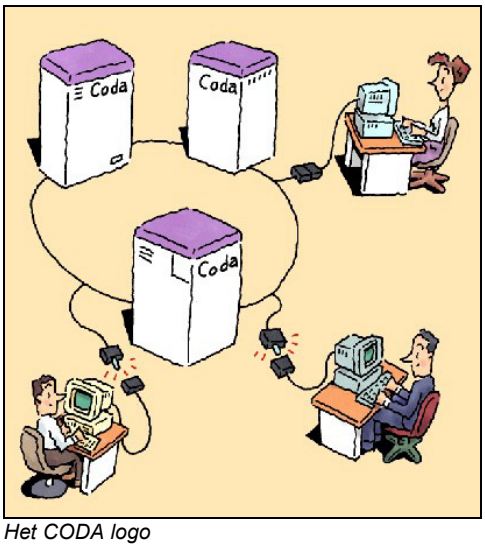
Een database servermachine draait de vier processen voor het onderhouden van de administratieve databases. Wanneer een cell groter is dan één servermachine is het best ervoor te zorgen dat er ook meer dan één database servermachine aanwezig is. Zo verkrijgt men het effect dat er hogere beschikbaarheid en een hogere betrouwbaarheid van de gegevens is. Als er dan één database servermachine uitvalt zijn de databases nog steeds bereikbaar op een andere database servermachine.

Een binary distribution machine zorgt voor het verdelen en opslaan van binaire bestanden voor de AFS-processen. Zo kan iedere servermachine dezelfde versies van processen draaien.

Een system control machine verdeelt en slaat systeemconfiguratiebestanden op die gedeeld zijn door alle servermachines in de cell. Wanneer we dan een configuratiebestand willen aanpassen, hoeven we dit enkel op de SCM te doen, en dan wordt dit bestand automatisch over alle andere servermachines verspreid. Belangrijk is ook dat er in een cell maar één SCM bestaan kan bestaan.

1.4 CODA

CODA is ook ontsproten uit het originele Andrew File System, maar is al snel een eigen leven gaan leiden. Het concept blijft nog steeds hetzelfde, maar CODA heeft vele features die in geen enkel ander cluster filesystem gekend zijn. Ook CODA werd ontwikkeld aan de Carnegie Mellon-universiteit door een groep van M. Satyanarayanan.



Het CODA logo

CODA beschikt dus over een aantal eigenschappen waardoor het een zeer performant systeem is, die ik hier even zal vernoemen:

- het werkt ook bij een verbroken verbinding (mobiele clienten)
  - Bij verbinding worden de data automatisch geïntegreerd
  - Bandbreedteadaptatie
- het voorziet in foutafhandelingsmethodes
  - lost server/server conflicten op
  - behandelt automatisch netwerkfouten
  - behandelt automatisch verbroken verbindingen van clients
- het is performant en schaalbaar
  - constante caching van bestanden, directories en attributen op de client
  - write back caching
- het is adequaat beveiligid
  - Kerberos authentificatie
  - access control lists (ACL's)

Een groot nadeel van CODA is dat het nog steeds in een soort van ontwikkel-testfase zit. CODA is een zeer goed systeem om mee te experimenteren en om kennis over geclusterde bestandssystemen op te doen. Maar het is nog steeds niet het meest stabiele systeem om een geclusterd bestandssysteem op te zetten. Hier zal waarschijnlijk nog verandering in komen, want men is nog voortdurend bezig met de verdere ontwikkeling en verbetering van dit systeem.

Zeer belangrijk is zijn zogenaamde disconnected operation. In grote netwerken is het zo dat regelmatig wel eens een server offline gaat; bij het originele AFS zorgde dit voor nogal wat hinder. Bij CODA werd dit probleem verholpen.

Disconnected operation:

Hoe gaat CODA dan met zo een netwerkfout om?
Wanneer de kernel van Linux voor het eerst een bestand vraagt aan de cache-manager (Venus in CODA) dan zal deze zoals uitgelegd in het originele AFS, het volledige bestand afhalen van de servers. Dit bestand wordt dan automatisch met alle bijkomende info in de cache gezet.
Stel nu dat we dit bestand opslaan zonder dat de server die dit bestand bezit beschikbaar is, dan zal er normaal geen update van het bestand op de server plaatsvinden en krijgen we een time-out error. Venus zal deze fout echter niet melden aan de gebruiker want hij zal zich realiseren dat de server niet beschikbaar is. Deze foutmelding zal dan op de client gelogd worden in de CML (Client Modification Log). Hetzelfde verhaal speelt zich af wanneer we bijvoorbeeld op een laptop zouden werken terwijl we niet verbonden zijn met het netwerk. Alle updates worden dan opgeslagen in de CML. Deze CML wordt regelmatig weggeschreven naar de harde schijf van de client.

De gebruiker merkt dus niets van deze disconnection mode. Wanneer de servers dan weer beschikbaar zijn, zal Venus automatisch alle updates die in de CML zitten uitvoeren en de CML aanpassen.

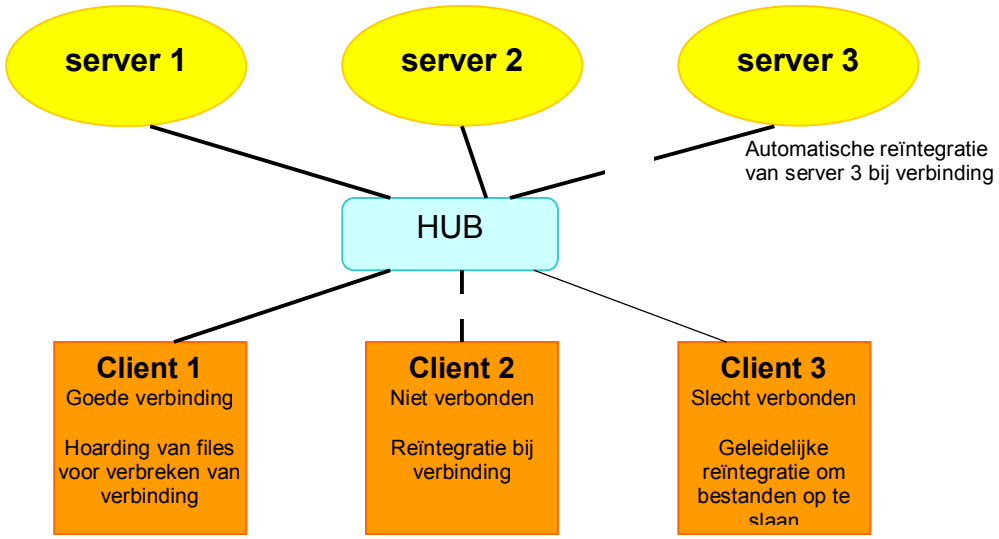
Wat als er twee personen zijn die tijdens het weekend allebei in disconnected mode een nieuwe directory maken met identiek hetzelfde pad (bv. /coda/usr/danny/eindwerk) en hier bestanden in zetten?
Wel het antwoord is simpel, bij de reïntegratie van de twee systemen wordt er automatisch een local/global conflict gedetecteerd. De bestanden worden dan symbolische links. Dit conflict wordt aan de gebruiker getoond en zal dan hersteld moeten worden. Deze herstelling kan in sommige gevallen automatisch gebeuren, maar kan ook handmatig worden afgehandeld. Meer informatie hierover vindt u terug in de installatie- en configuratieprocedures van CODA in één van de volgende hoofdstukken.

Hoarding files

Dit is nog een belangrijk voordeel van CODA. We kunnen er als het ware voor zorgen dat bepaalde bestanden een hoge prioriteit hebben in het up-to-date houden. Dit doen we door deze bestanden in de gebruikers hoard database te zetten zodat er automatisch gevraagd wordt aan de servers deze bestanden up-to-date te houden. Hoe dit juist in zijn werk gaat om bestanden te hoarden, zal ook in de installatie- en configuratieprocedures van CODA uitgelegd worden.

In onderstaande figuur ziet u nog eens de automatische foutafhandelingsmethodes van CODA afgebeeld.

Automatische foutafhandelingsmethodes:



1.5 Alternatieven voor CODA of OpenAFS

Wanneer we naar een mogelijk alternatief voor de gebruikte systemen zoeken, moeten we een systeem vinden dat het mogelijk maakt bestanden te delen. Ook moet dit systeem redundant zijn, dit wil zeggen dat wanneer er bijvoorbeeld een harde schijf crasht, er geen gegevens verloren gaan. Het systeem moet ook weten met verbroken netwerkverbindingen om te gaan. En een zeer belangrijk punt is dat het, zoals ieder goed systeem, over een goede beveiliging moet beschikken.

Alternatief 1: NFS
NFS wordt zeer veel gebruikt maar er zijn heel wat tekortkomingen aan wanneer we NFS met CODA of OpenAFS vergelijken. Zo heeft NFS al een groot nadeel op gebied van snelheid, ook de beveiliging binnen NFS is niet up-to-date. Onder Linux is het zo dat NFS nog steeds niet betrouwbaar is, hiermee bedoel ik dat het nogal graag instabiel wordt. Ook is het zo dat als bijvoorbeeld een NFS-server down is, dat alle clients die een gedeeld bestand gebruiken in een instabiele toestand raken.
Nog een belangrijk punt is dat wanneer we met meerdere servers werken onder NFS, we nooit alle gedeelde bestanden onder één gemeenschappelijke directory kunnen plaatsen. Ook heeft een client onder NFS steeds een up-to-date lijst nodig van alle servers en hun geëxporteerde directories in het netwerk terwijl de client bij CODA gewoon moet weten waar hij de CODA root-directory `/coda` kan vinden.
Uit al deze tekortkomingen blijkt dus al snel dat NFS geen mogelijk alternatief is voor OpenAFS of CODA.

Alternatief 2: RAID
De naam RAID staat voor Redundant Array of Inexpensive Disks, wat letterlijk vertaald redundante groep van goedkope schijven betekent. Uit deze naam kunnen we afleiden dat deze techniek de nodige redundantie biedt, we kunnen met RAID namelijk mirroring toepassen. Mirroring is een techniek die ervoor zorgt dat data automatisch op twee harde schijven opgeslagen worden. Hierbij hebben we dus al dadelijk tweemaal zoveel ruimte nodig. Er bestaat in RAID wel nog een andere techniek namelijk RAID4. Er wordt namelijk afwisselend op de verschillende schijven geschreven en op een andere schijf worden de pariteitsgegevens opgeslagen. Deze techniek wordt niet veel gebruikt, maar RAID5 wel, dit systeem schrijft afwisselend over de schijven data en pariteitsgegevens zodat alle mogelijke problemen bij verlies van data uitgesloten worden.
Maar uit onderzoek blijkt snel dat we voor een degelijk RAID-systeem extra hardware nodig hebben, namelijk RAID-controllers. Dit is al een behoorlijke tekortkoming daar we met OpenAFS of CODA zonder bijkomende hardware een goed redundant systeem kunnen verwezenlijken. Weer geen goed alternatief voor onze systemen dus.

Uit de beide methodes kunnen we besluiten dat het niet makkelijk is een mogelijk alternatief te vinden voor CODA of OpenAFS dat het goede concept van beide systemen bevat. Tuurlijk zouden we andere cluster filesystems gaan gebruiken als alternatief, maar waarom we nu net voor OpenAFS en CODA in dit eindwerk gekozen hebben zal in een later hoofdstuk nog aan bod komen.

1.6 De beginsituatie van het eindwerk

Er bestaan vandaag de dag massa's boeken over Linux in de boekhandels, veel van deze boeken behandelen dan ook dezelfde onderwerpen. Maar wanneer je een boek zoekt over het onderwerp clustering filesystems zal je al ver moeten zoeken voor je er één vindt, als je al een boek over dit onderwerp vindt. Je zal dus ook geen boek vinden met daarin een goede handleiding voor de installatie en configuratie van deze systemen.
Zoals iedereen weet die met Linux bezig is, moet je hierover informatie van het internet plukken. Ook moet je een selectie maken tussen de slechte en dus onbetrouwbare informatie en de goede informatie. Deze zal je nodig hebben en moeten behouden. Welke de goede informatie is kan je meestal enkel te weten komen nadat je het concept achter de systemen hebt proberen te begrijpen en grondig met de systemen geëxperimenteerd hebt. Het onderzoek alleen al naar deze systemen neemt heel wat tijd in beslag, en wanneer we deze systemen dan nog eens moeten uittesten, hebben we reeds heel wat tijd opgebruikt daar deze systemen niet eenvoudig te installeren en te configureren zijn.

1.7 De essentie en het gewenste resultaat

Mijn eindwerk bevat alle essentiële theoretische informatie over hoe enkele geclusterde bestandssytemen werken en hoe je beide geclusterde bestandssytemen kan installeren en configureren.
Na mijn eindwerk tot een goed resultaat gebracht te hebben hoop ik dat het zelfs voor een volkomen Linux-nieuweling mogelijk is één van beide systemen binnen linux op te zetten. Het gewenste resultaat is dus niet dat hij in dit eindwerk terug kan vinden hoe hij een bepaalde Linux-distributie installeert, maar wel dat hij op dezelfde distributies of hoger met dezelfde kernel of hoger zo een systeem kan implementeren zonder hierbij nog informatie op het internet te moeten opzoeken.

De instructies voor deze implementatie zullen stap voor stap of in Linux termen commando per commando beschreven worden in deze handleiding. Ook de veel voorkomende problemen die u eventueel kan tegenkomen, worden hierin toegelicht met de mogelijke oplossingen.

Ook zal ik er persoonlijk voor zorgen dat dit eindwerk net als alle andere informatie over Linux gratis te raadplegen zal zijn over het internet.

1.8 Beperkingen

Eén van de beperkingen bij dit eindwerk is de materiële beperking. Ik zal tijdens dit eindwerk geen reëel bedrijfsnetwerk kunnen simuleren met vele computersystemen, daar ik maar over enkele computersystemen beschik. Ik zal deze systemen dus onderzoeken op enkele pc's waarvan er sommige als server en andere als client gebruikt worden, en kan niet vertellen welke resultaten deze systemen zullen hebben in een bedrijfssituatie.

Ook beschik ik slechts over een beperkte tijd die ik aan dit eindwerk kan spenderen en is het dus waarschijnlijk niet mogelijk om nog andere geclusterde bestandssystemen (bvb. Lustre, Intermezzo, …) te onderzoeken. Ik kan deze systemen dus ook niet op basis van prestaties vergelijken met CODA en OpenAFS, die ik wel effectief getest heb op hun prestaties in probleemsituaties. Ik zal wel in dit eindwerk vermelden waarom ik voor een bepaald systeem kies.

Hoofstuk2: Aanpak van het eindwerk

2.1 Gevolgde werkwijze

In dit onderdeel zal ik trachten de uitgevoerde werkwijze te verduidelijken, en hierbij duidelijk de structuur in deze werkwijze te tonen.

2.1.1 Vooropgestelde werkschema

In dit vooropgestelde werkschema heb ik geprobeerd op voorhand de verschillende fases te schetsen die ik tijdens mijn afstudeerwerk zal moeten ondernemen.

Fase 1:	Bestaande informatie opzoeken
Fase 2:	De te testen systemen selecteren
Fase 3:	Planning opstellen
Fase 4:	Uitgebreide informatie verzamelen voor het testen van systeem 1
Fase 5:	Systeem 1 installeren en configureren
Fase 6:	Probleemsituaties creëren voor dit systeem
Fase 7:	Analyse van systeem 1 beschrijven
Fase 8:	Uitgebreide informatie verzamelen voor het testen van systeem 2
Fase 9:	Systeem2 installeren en configureren
Fase 10:	Probleemsituaties creëren voor dit tweede systeem
Fase 11:	Analyse van systeem 2 beschrijven
Fase 12:	Een goede en duidelijk gestructureerde handleiding opstellen

2.1.2 Uitleg bij dit werkschema

Fase 1:
In eerste instantie is het heel belangrijk alle mogelijke informatie te vergaren over wat een cluster filesystem juist betekent, wat het doel hiervan is en welke cluster filesystems er zoal bestaan. Hierna is het van groot belang de belangrijkste cluster filesystems te bestuderen en te vergelijken.

Fase 2:
De tweede stap die ik onderneem is het selecteren van de systemen. Dit is een stap die wordt gezet in samenspraak met mijn promotor, daar hij reeds ervaring heeft met enkele systemen.

Fase 3:
Een derde stap omvat het opstellen van een planning voor de uitvoering van de testen die ik zal ondernemen. Deze ruwe planning moet rekening houden met onverwacht opduikende problemen. In ieder geval moeten er enkele belangrijke deadlines in genoteerd worden.

Fase 4:
Na deze ruwe planning, start ik met het eerste systeem. Eerst zal ik zoveel mogelijk informatie over dit systeem vergaren en deze informatie naar betrouwbaarheid selecteren.

Fase 5:
Wanneer ik dan een duidelijk overzicht heb over hoe dit eerste systeem werkt en hoe ik te werk moet gaan om dit systeem te installeren en te configureren, kan ik hieraan beginnen. Voor deze stap is het natuurlijk ook heel belangrijk dat ik mijn basiskennis van Linux eerst opfris.

Fase 6:
Misschien wel één van de belangrijkste stappen voor het analyseren van het systeem, namelijk het creëren van probleemsituaties. Hierbij heeft mijn promotor mij geadviseerd zeer extreme situaties te gebruiken zoals het verbreken van een netwerkkabel, loskoppelen van een harde schijf, … .

Fase 7:
Eénmaal ik deze probleemsituaties gecreëerd heb en de reacties van de systemen op een dergelijke probleemsituatie ken, is het belangrijk een schriftelijke analyse te doen van dit systeem, om zo de notities genomen tijdens het onderzoek in een goed gestructureerd verslag om te zetten.

Fase 8:
Na een goede analyse van het eerste systeem, kan ik met het tweede systeem van wal steken. De eerste stap hierbij bestaat er opnieuw in zoveel mogelijk informatie rond dit systeem te vergaren en enkel de goede informatie te selecteren.

Fase 9:
Als ik dan de goede informatie uit alle informatie gefilterd heb en bestudeerd heb kan ik aan de installatie en configuratie van het tweede systeem beginnen.

Fase 10:
Dezelfde probleemsituaties creëren als bij het eerste systeem om zo een goede analyse van het systeem te kunnen doen. En misschien even de resultaten van beide systemen vergelijken.

Fase 11:
Ook bij dit systeem zal ik na het uitgevoerde onderzoek weer een goed gestructureerd verslag moeten opstellen zodat ik hieruit een goede handleiding voor dit systeem kan opstellen met de oplossingen voor problemen die zich kunnen voordoen.

Fase 12:
Dit is als het ware de finale van het onderzoek waarin ik voor beide systemen een goede handleiding en de essentiële theoretische kennis in een verslag zet, namelijk het bundeltje dat nu voor u ligt.

2.2 Informatiebronnen

Tijdens dit onderzoek naar deze twee systemen heb ik heel wat informatiebronnen geraadpleegd. De voornaamste bron zoals iedereen wel weet voor onderwerpen in verband met Linux is het internet. Wanneer ik voor een groot probleem stond kon ik ook altijd terecht bij mijn promotor om mij door dit probleem heen te helpen. Verder heb ik op het internet ook vaak gebruik gemaakt van speciale forums en mailinglists waar veel voorkomende problemen besproken worden.
Ook heb ik om mijn basiskennis Linux op te frissen enkele boeken gebruikt.
Van al de belangrijke informatiebronnen die ik tijdens mijn eindwerk gebruik heb zoals websites, boeken, vakliteratuur, … zal ik ook een duidelijk overzicht maken in het onderdeel bibliografie.

2.3 Waarom hebben we voor CODA en OpenAFS gekozen?

Waarschijnlijk vraagt u zich af, waarom nu CODA en OpenAFS worden onderzocht en geen ander cluster filesystem?
Deze selectie is mij geadviseerd door mijn promotor, die als fervente linux-gebruiker al met enkele cluster filesystems heeft geëxperimenteerd.
Zo hebben we bijvoorbeeld Intermezzo, Intermezzo is voortgevloeid uit CODA maar omdat er veel minder ondersteuning voor is werd dit bestandssysteem geschrapt.
We hebben dan ook nog Lustre, een zeer goed en geavanceerd bestandssysteem, maar omwille van commerciële redenen en materiële beperking werd dit bestandssysteem geschrapt. Het was de bedoeling een gratis systeem te onderzoeken, en dit is bij Lustre niet volledig het geval. Er zijn bepaalde opties waarvoor men een bijdrage moet betalen. Ook is het opzetten van een Lustre cluster filesytem veel complexer dan een ander cluster file system. Je hebt namelijk al zes computersystemen nodig om dit cluster filesystem in zijn eenvoudigste vorm op te zetten.
Dan is er ten slotte ook nog het Global File System (GFS) dit wordt enkel ondersteund door Red Hat en is ook niet gratis te downloaden.
Dit zijn zeker niet alle cluster filesystems, maar enkele van de belangrijkste die niet in dit onderzoek zijn betrokken.

Hoofdstuk3: De praktische realisatie

3.1 Voorbereiding, installatie en configuratie van CODA

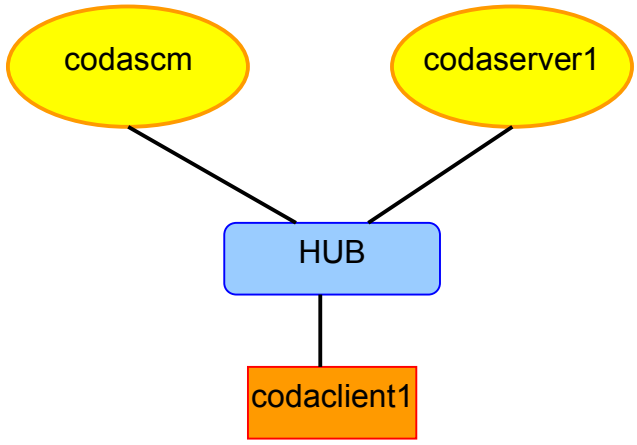
In dit onderdeel zal ik de volledige praktische realisatie van CODA behandelen: de nodige voorbereidingen die getroffen moeten worden, de installatieprocedure en het configureren van CODA tot een functionerend systeem.

Toch wil ik eerst nog even vermelden dat ik dit systeem opgezet en getest heb onder de distributie Slackware 10.0 met kernel 2.4.26.

3.1.1 Voorbereidingen

Om aan de installatie van CODA te kunnen beginnen, is het belangrijk even na te gaan welke voorbereidingen moeten getroffen worden. Als eerste punt in deze voorbereidingen moet men zich enkele vragen stellen zoals bijvoorbeeld, is CODA geschikt voor onze behoeften, hoe groot is het netwerk waarin CODA geïmplementeerd moet worden, moet er een backup-systeem voorzien worden voor de server-control-machine, enz… .
Wanneer al deze vragen beantwoord zijn, kan men best overgaan tot het schematiseren van het volledige CODA-systeem. In onderstaande tekening heb ik mijn “simpel” CODA-systeem grafisch voorgesteld.

Schema van mijn CODA-systeem:



Zoals u uit de tekening kan afleiden had ik tijdens het opzetten van mijn CODA systeem slechts 3 pc’s ter beschikking. Hiervan heb ik één pc geconfigureerd als server control machine (codascm), een tweede pc als tweede server (codaserver1) en de laatste pc als simpele client (codaclient1).

Wanneer u klaar bent met alle voorbereidingen en ongeveer een volledig overzicht heeft van hoe het volledige systeem er gaat uitzien kan u verder gaan met de installatie van de eerste server.

3.1.2 Installatie en configuratie van de eerste server

De eerste pc waarop men CODA installeert, wordt steeds de SCM of server control machine. Vooraleer ik begin met de installatie van deze SCM vind ik het belangrijk deze server een reële benaming te geven. Dit doen we door de naam in het bestand `/etc/HOSTNAME` aan te passen.

```
pico /etc/HOSTNAME
```

De huidige naam wordt veranderd in codascm.

Let op, na deze naamsverandering moet ook op ieder systeem dat communiceert met deze pc het bestand `/etc/hosts` aangepast worden, dit om ervoor te zorgen dat de verwijzingen tussen het ip-adres en de hostnaam juist zijn.
De naamsverandering vindt plaats bij de eerstvolgende reboot.

De bestanden die je nodig hebt voor de installatie van Coda zijn:

- Coda-6.0.6
- Lwp-1.11 library
- Rpc2-1.22 library
- Rvm-1.9 library

Deze bestanden vinden we op `www.coda.cs.cmu.edu/pub/coda/src/` en zullen tegen de publicatie van dit eindwerk waarschijnlijk niet meer de meest recente versies zijn. Wanneer we deze bestanden gedownload hebben is het belangrijk te controleren of er reeds een CODA-module aanwezig is. Dit doen we door in onderstaande directory een kijkje te nemen.

```
cd /lib/modules/2.4.26/kernel/fs/coda
```

Normaal zit er bij deze kernel reeds een module voor CODA, tijdens het installeren van CODA heb ik echter ondervonden dat er een probleem was met deze module. Het is namelijk enkel mogelijk om met deze module een oudere versie van CODA op te zetten, en omdat dit geen mogelijke oplossing was vanwege de onopgeloste bugs in oudere versies van CODA heb ik ervoor gekozen een nieuwe kernel-module te compileren die wel compatibel is met Coda-6.0.6.
Om deze module te compileren gaan we als volgt te werk. Eerst halen we het bestand linux-coda-6.0.0.tgz af op `ftp.coda.cs.cmu.edu:/pub/coda/linux/kernel` .

Hierna voeren we volgende commando's in:

```
tar –xzf linux-coda-6.0.0.tgz
cd linux-coda
make config
make coda.o
cp linux2.4/coda.o /lib/modules/2.4.26/kernel/fs/coda
```

Na deze commando's hebben we de juiste kernel-module om coda 6.0.6 op te zetten.

Nu beginnen we aan het installeren van de nodige programma's en bibliotheken voor het opzetten van een CODA-bestandssysteem. Allereerst is het nodig volgende bestanden te downloaden:

- coda-6.0.6.tar.gz
- lwp-1.11.tar.gz
- rvm-1.9.tar.gz
- rpc2-1.22.tar.gz

Daarna beginnen we met het uitpakken en installeren van de verschillende bibliotheken door volgende commando's :

```
tar –xvzf lwp-1.11.tar.gz
cd lwp-1.11
./configure
make
checkinstall

tar –xvzf rvm-1.9.tar.gz
cd rvm-1.9
./configure
make
checkinstall

tar –xvzf rpc2-1.22.tar.gz
cd rpc2-1.22
./configure
make
checkinstall
```

Misschien even ter verduidelijking, ik maak gebruik van het commando "`checkinstall`", dit is geen standaard commando in Slackware 10.0, je kan dit commando enkel uitvoeren indien je het checkinstall pakket geïnstalleerd hebt. Dit commando mag ook vervangen worden door "`make install`". Ik gebruik "checkinstall" omdat bij dit commando automatisch een lijst aangemaakt wordt met de reeds geïnstalleerde pakketten in de directory `/var/adm/packages`. Het is hierdoor heel makkelijk om een met checkinstall geïnstalleerd pakket te verwijderen door het commando "`removepkg naam_van_pakket`" te gebruiken.

Nadat we al deze pakketten geïnstalleerd hebben, kunnen we ook coda uitpakken en installeren.

```
tar –xvzf coda-6.0.6.tar.gz
cd coda-6.0.6
./configure
make
make server-install
```

Wanneer we al deze pakketten geïnstalleerd hebben, kunnen we even onze home directory opruimen door twee nieuwe directory's aan te maken namelijk "`src`" en "`coda-archieven`". We plaatsen dan alle uitgepakte bestanden in de "`src`" directory en de gedownloade pakketten in de "`coda-archieven`" directory. Voor we nu kunnen beginnen met configureren moeten we eerst onze directory-structuur in orde maken.

Eerst formatteren we een partitie die CODA kan gebruiken om de data die voor de clients bereikbaar zijn in op te slaan. Dit moet een standaard ext2 partitie zijn. Ik heb bij mijn onderzoek gebruik gemaakt van partitie hda6.

```
mkfs.ext2 /dev/hda6
```

Nu kunnen we een directory `/vicepa` aanmaken en de geformatteerde partitie in deze directory mounten.

```
mkdir /vicepa
mount /dev/hda6 /vicepa
```

Om deze partitie nu automatisch bij het opstarten te laten mounten in `/vicepa` voegen we een regel toe aan het bestand `/etc/fstab`.

In de directory `/vicepa` komen later de data te staan die bereikbaar zijn voor de clients. Nu moeten we nog een directory aanmaken met twee bestanden waarmee de logging van CODA mogelijk wordt. Indien we deze CODA-opstelling in een echte bedrijfsomgeving zouden gaan gebruiken, is het beter partities aan te maken voor deze logging omdat dit in een hogere performantie resulteert. Maar omdat wij enkel van plan zijn om onderzoek naar CODA te doen, maken we een directory aan waarin we de twee bestanden kunnen opslaan.

```
mkdir /usr/coda/logs
```

Nu zijn we klaar om van start te gaan met de configuratie. Wanneer we vice-setup uitvoeren wordt er een shell-script gestart. Door voor de uitvoer een # te zetten heb ik duidelijk willen maken dat dit uitvoer is van het shell-script. Ik heb de antwoorden op deze vragen er in het grijs bijgezet, de meeste antwoorden zijn natuurlijk persoonlijk en hangen af van de instellingen die u wenst voor uw server. De kleine tekst tussendoor is commentaar.

```
vice-setup
```

#Welcome to the Coda Server Setup script!

#Setting up config files for a coda server.
#Do you want the file /etc/coda/server.conf created? [yes] **yes**
Hierdoor wordt er een configuratiebestand gecreeërd in /etc/coda met de naam server.conf
#What is the root directory for your coda server(s)? [/vice] **/vice**
In deze directory komen alle configuratiebestanden, server-logs, …
#Setting up /vice.
#Directories under /vice are set up.

#Is this the master server, aka the SCM machine? (y/n) **y**
De eerste server is meestal de master server (SCM) en is de enige server die de configuratie databases kan aanpassen die gedeeld worden door alle servers.

#Setting up tokens for authentication.
#The following token must be identical on all servers.
#Enter a random token for update authentication : **updateto**
#The following token must be identical on all servers.

#Enter a random token for auth2 authentication : **authtok**
#The following token must be identical on all servers.
#Enter a random token for volutil authentication : **volutilt**
#tokens done!

#Setting up the file list for update client
#Filelist for update ready.
#/etc/services already has new services registered! Good.
#/etc/services ready for Coda
#Now installing files specific to the SCM...

#Setting up servers file.
#Enter an id for the SCM server. (hostname distfs)
#The serverid is a unique number between 0 and 255.
#You should avoid 0, 127, and 255.
#serverid: 1
#done!
#Initializing the VSGDB to contain the SCM as E0000100
#/vice/db/VSGDB set up

#Setting up ROOTVOLUME file
#Enter the name of the rootvolume (< 32 chars) : **codascmvol1**

#Setting up users and groups for Coda

#You need to give me a uid (not 0) and username (not root)
#for a Coda System:Administrator member on this server,
#(sort of a Coda super user)

#Enter the uid of this user: **600**
Hier kan u best een hoge id ingeven.
#Enter the username of this user: **danny**
#Going to rebuild the protection databases
#moving /vice/db/prot_users.db to /vice/db/prot_users.db.old
#moving /vice/db/prot_index.db to /vice/db/prot_index.db.old
#An initial administrative user danny (id 600)
#with Coda password changeme now exists.

#A server needs a small log disk partition, preferably on a disk by
#itself. It also needs a metadata partition of approx 4% of your filespace.

#For trial purposes you may give ordinary files instead of raw
#partitions. Keep all size small if you do this.
#Production servers will want partitions for speed.

#--------------------------
#WARNING: you are going to play with your partitions now.
#verify all answers you give.

```
#--------------------------

#WARNING: these choices are not easy to change once you are up and running.

#Are you ready to set up RVM? [yes/no] yes

#What is your log partition? /usr/coda/logs/log_disk_file.log

#The log size must be smaller than you log partition.  We
#recommend not more than 30M log size, and 2M is a good choice.
#What is your log size? (enter as e.g. '2M') 2M

#What is your data partition (or file)? /usr/coda/logs/data_disk_file.log

#The data size must be approx 4% of you server file space.  We
#have templates for servers of approx: 500M, 1G, 2.2G, 3.3G, 8G
#(you can store less, but not more on such servers).
#The corresponding data sizes are  22M, 44M, 90M, 130M, 315M.
#Pick one of the defaults, otherwise I will bail out

#What is the size of you data partition (or file)
#[22M, 44M, 90M, 130M, 200M, 315M]: 22M

#----------------------------
#WARNING: DATA and LOG partitions are about to be wiped.

#  log area: /usr/coda/logs/log_disk_file.log, size 2M.
#  data area: /usr/coda/logs/data_disk_file.log, size 22M.

#Proceed, and wipe out old data? [y/n] y

#LOG file has been initialized!

#Rdsinit will initialize data and log.
#This takes a while.
#rvm_initialize succeeded.
#Going to initialize data file to zero, could take awhile.
#done.
#rds_zap_heap completed successfully.
#rvm_terminate succeeded.

#RVM setup is done!



#Your server directories will hold the files (not directories).
#You can currently only have one directory per disk partition.

#Where shall we store your file data [/vicepa]? /vicepa
In deze directory zal de data opgeslagen worden die bereikbaar is voor de clients onder /coda.
#Shall I set up a vicetab entry for /vicepa (y/n) y
```

#Select the maximum number of files for the server.
#[256K, 1M, 2M, 16M]: **16M**

#Server directory /vicepa is set up!

#Congratulations: your configuration is ready...and now
#to get going do the following:

start the auth2 server as: auth2
start rpc2portmap as: rpc2portmap
start updatesrv as: updatesrv
start updateclnt as: updateclnt -h pcb10214
start the fileserver: startserver &
wait until the server is up: tail –f /vice/srv/SrvLog
create your root volume: createvol_rep codascmvol1 E0000100 /vicepa
setup a client : venus-setup codascm.khlim.be 20000
start venus : venus
enjoy Coda.
For more information see http://www.coda.cs.cmu.edu.

Nadat dit script uitgevoerd is, moeten we de server starten, door de nodige processen te starten. Hiervoor geven we volgende commando's in:

```
/usr/local/etc/auth2.init start
/usr/local/etc/update.init start
/usr/local/etc/codasrv.init start
```

Met het eerste commando starten we de authentificatie- en beveiligingsprocessen die nodig zijn voor het inloggen op de server. Met het tweede commando starten we twee update-processen: updateclnt en updatesrv.
Hierbij dient updateclnt (updateclient) voor het ontvangen van update-gegevens van veranderde files en databases. Updatesrv (updateserver) dient voor het versturen van update-gegevens.
Met het derde commando starten we de server, dit proces zal bijna alle handelingen voor zijn rekening nemen.
Nu kunnen we controleren of de server draait door volgend commando in te geven.

```
xterm –e tail /vice/srv/SrvLog &
```

Wanneer nu de server opgezet is moeten we een root volume creëren:

```
createvol_rep codascmvol1 E0000100 /vicepa
```

De server is nu klaar, het is nu mogelijk om in te loggen met een client op deze server. We zullen nu verder gaan met het installeren van een client.

3.1.3 Installatie en configuratie van een client

Allereerst kunnen we misschien ook deze pc een reële naam geven door in `/etc/HOSTNAME` de naam aan te passen.

```
pico /etc/HOSTNAME
```

We veranderen de huidige naam in codaclient1 en we zetten in het bestand /etc/hosts twee regels. Eén regel met verwijzing tussen de hostnaam en het IP-adres van de client en één regel met een verwijzing naar de codascm en zijn IP-adres. Ook op de eerste server (codascm) in de file `/etc/hosts` moet een verwijzing naar deze client (codaclient) en zijn IP-adres zitten.

Voor het opzetten van een client hebben we identiek dezelfde pakketten en bibliotheken nodig als bij het opzetten van een server. De werkwijze voor het installeren van een server wordt dus ook herhaald en komt hier niet aan bod. Enkel bij het installeren van coda-6.0.6 moeten we een ander commando ingeven.

```
tar –xvzf coda-6.0.6.tar.gz
cd coda-6.0.6
./configure
make
make client-install      ➔ Ander commando
```

Na deze installatie is het mogelijk de cache-manager van CODA namelijk Venus te starten.
We gaan allereerst Venus configureren om verbinding te maken met een testserver, door volgend commando in te geven:

```
venus-setup testserver.coda.cs.cmu.edu 20000
```

Door dit commando in te geven worden alle configuratiebestanden aangepast. Ook wordt er een directory geïnitialiseerd voor de cache. De 20000 achteraan het commando geeft de grootte van de cache aan in kilobytes. Deze moet altijd tussen 10Mb en 300Mb liggen.

Door volgend commando in te geven maken we een verbinding met de server:

```
venus &
```

Nu zijn we verbonden met de testserver, de volgende stap is het verbinden van de client met de persoonlijke server. Dit doen we door de venus stop te zetten en de directory `/coda` te unmounten.

```
vutil shutdown
umount /coda
```

Dan passen we alle configuratiebestanden en de directory `/etc/services` weer aan om verbinding te maken met onze persoonlijke server en starten we venus.

```
venus-setup codascm.khlim.be 20000
/usr/local/etc/venus.init start
```

Misschien is het goed even te verduidelijken wat Venus juist is. Venus is de client cache manager, en zorgt dus voor een snellere werking wanneer vaak dezelfde files gevraagd worden. Want wanneer er een bepaalde file-request binnenkomt gaat Venus eerst in de cache van de client kijken of deze file al opgevraagd is, zit deze request niet meer in de cache dan wordt er een request doorgegeven naar de CODA-servers. Voor meer informatie over de gedetailleerde werking verwijs ik graag naar “1.3 Het Andrew File System” of naar `http://www.coda.cs.cmu.edu` .

Nu willen we inloggen op de server, dit doen we als volgt:

```
clog “danny”
```

Dan wordt er naar een paswoord gevraagd, dit is standaard steeds “changeme”. Wanneer we nu in de coda directory op de client kijken zullen we zien dat we over onze eigen persoonlijke directory beschikken. In deze directory kunnen we nu files zetten, directories aanmaken enzoverder …

Om ons paswoord te veranderen, gebruiken we het commando “`cpasswd`” gewoon vanop de client terwijl we ingelogd zijn.

Wanneer dit allemaal werkt, is het misschien interessant om eens een nieuw volume bij te creëren op de server. Eerst en vooral moeten we er zeker van zijn dat volgende processen draaien: codasrv, updatecln and updatesrv. Dan doen we het volgende:

```
createvol_rep codascmvol2 E0000100 /vicepa
```

Hierbij is codascmvol2 willekeurig gekozen als volumenaam, E0000100 is het VSG adres en kunnen we terugvinden in `/vice/db/VSGDB`, `/vicepa` is de directory waarin we het nieuwe volume aanmaken.

Om het volume nu zichtbaar te maken vanop de client, moeten we eerst inloggen indien we nog niet ingelogd waren. Dan moeten we het nieuwe volume mounten.

```
cfs mkmount /coda/codascm.khlim.be/codascmvol2 codascmvol2
```

In vorige regel was cfs mkmount het commando,
`/coda/codascm.khlim.be/codascmvol2` is het absolute pad waarin het volume gemount wordt en codascmvol2 is de naam van het te mounten volume. Nu kunnen we ook in dit nieuwe volume bestanden zetten enz…
Let op: Wanneer we geen ander volume in de coda-directory zouden mounten, werken we automatisch in het zogenaamde rootvolume van de SCM, in ons geval dus codascmvol1.

Nu het opzetten van een client en de verbinding hiervan met een server uit de doeken gedaan is, kunnen we overgaan tot het opzetten van een 2de server binnen het coda systeem.

3.1.4 Installatie en configuratie van een bijkomende server

Voor we van start kunnen gaan met de installatie moet natuurlijk weer de naam in `/etc/HOSTNAME` gewijzigd worden en moeten de verwijzingen in `/etc/hosts` aangepast worden.
Zowel op de eerste server (codascm) als op de client (codaclient1) moeten ook verwijzingen naar deze tweede server (codaserver1) aangebracht worden in `/etc/hosts`.

Dan zijn we nu klaar voor de installatie van een tweede server: codaserver1
De installatie verloopt geheel identiek aan de installatie van codascm in punt 3.1.2 en hoeft dus niet herhaald te worden, de configuratie daarentegen verschilt een beetje van de configuratie van codascm.

We geven het volgende commando in:

```
vice-setup
```

Hierna volgt weer hetzelfde configuratiescript als bij codascm. We zullen onder andere moeten ingeven dat we nu niet te maken hebben met een SCM. We zullen dan de hostnaam van de SCM, in mijn geval “codascm”, moeten opgeven en het update token.
Na dit configuratiescript is het belangrijk enkele bestanden in de configuratiedatabase van codascm aan te passen, in het bestand `/vice/db/servers` moeten we de naam en het unieke serverID van codaserver1 bijvoegen. Aan het bestand `/vice/db/VSGDB` moeten we het volume van deze server toevoegen en een verwijzing van de scm naar de server.

Vb: E000104 codaserver1
E000104 codascm codaserver1

Hierna kunnen we best alle processen betreffende coda zowel op codascm als op codaserver1 herstarten.
We moeten hiervoor op beide systemen onderstaande commando’s uitvoeren.

```
/usr/local/etc/codasrv.init stop
/usr/local/etc/update.init stop
/usr/local/etc/auth2.init stop
```

Dan starten we eerst codascm en daarna codaserver1.

```
/usr/local/etc/auth2.init start
/usr/local/etc/update.init start
/usr/local/etc/codasrv.init stop
```

Nu kunnen we vanop codascm nieuwe volumes creëren op codaserver1.

```
createvol_rep codaservervol1 E0000104 /vicepa
createvol_rep codaservervol2 E0000104 /vicepa
```

Nu kunnen we op dezelfde manier als eerder beschreven is onze codaclient1 verbinden met codascm en codaserver1.

```
venus-setup codascm.khlim.be 20000
/usr/local/etc/venus.init start
```

Nu kunnen we inloggen met "`clog`" en de gewenste volumes mounten met "`cfs mkmount`".
Vanaf nu is ons CODA-systeem klaar om te gebruiken en uit te testen.
Indien u de servers wenst stop te zetten, doet u dit best met het commando: volutil shutdown.
Een client kan men best stopzetten met het commando: "`vutil shutdown`".

Tot zover de installatie en configuratie van een coda-bestandssysteem.
In volgende hoofdstukken zal ik het hebben over reeds uigevoerde testen op dit bestandssysteem.

3.2 Uitgevoerde testen onder CODA

Alle volgende operaties worden uitgevoerd vanop de client terwijl beide servers werken en de client verbonden is met beide servers.

Om een soort van real-time controle te hebben, kan je enkele vensters openen met daarin de status van codascm en codaserver1.

```
xterm –e cmon codascm.khlim.be &
xterm –e cmon codaserver1.khlim.be &
```

Ook van de client kan je real-time informatie over de connectie met de servers verkrijgen.

```
xterm –e codacon &
```

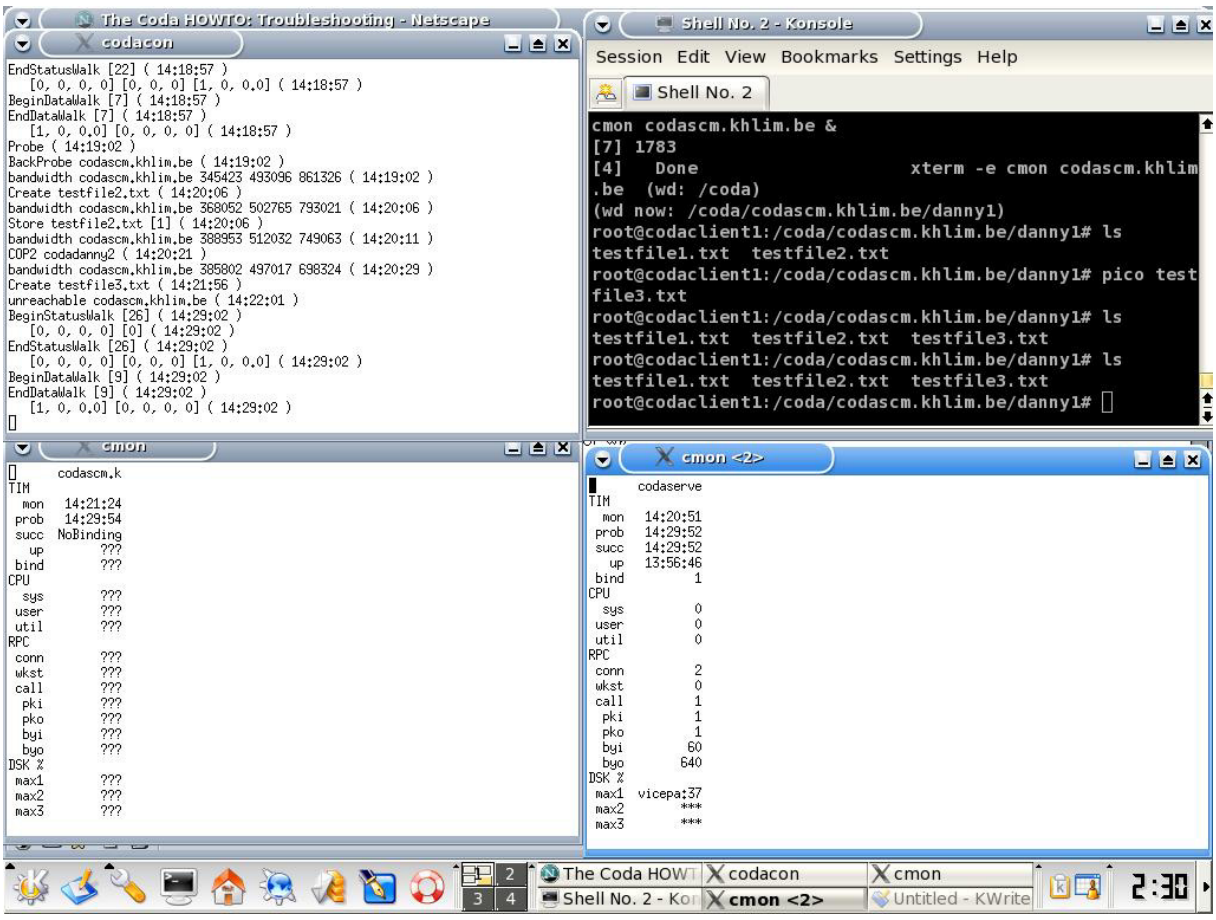
3.2.1 Test 1 : Disconnected operation

Om nu de eigenschap "disconnected operation" te testen, kan je de utp kabel van de client eens uittrekken en een bestand proberen te maken in een servervolume in `/coda/codascm.khlim.be/…` . Er zal geen foutmelding komen, maar bij een nieuwe verbinding wordt dit bestand automatisch aangemaakt.

Deze eigenschap werkt dus en zal zeer goed van pas komen in netwerken met minder goede netwerkprestaties, bvb: draadloze netwerken, beperkte bandbreedte, netwerk met draagbare computers, …

3.2.2 Test 2: Server-outage

Om te testen wat de reactie is bij het uitvallen van een server kan je de utp-kabel van codascm eens uittrekken om een server-outage te simuleren. Wanneer we dan de real-time informatie bekijken zien we het volgende:



In het venster linksonder zien we vraagtekens, dit wijst erop dat we geen informatie kunnen krijgen over de connectie met codascm en dat deze dus offline is.
Wanneer we nu op een reeds gemount volume van codascm een bestand proberen te schrijven krijgen we ook geen fout, in het venster linksboven verschijnt dan wel:

unreachable codascm.khlim.be

Bij het terug online gaan van codascm wordt automatisch het bestand geüpdatet op de server.
Er zal dus een soort van back-up systeem moeten voorzien worden dat automatisch de touwtjes in handen neemt wanneer de SCM uitvalt om zo de databases van de andere servers up-to-date te houden. Ook is het naar mijn mening beter om op de SCM geen data te plaatsen, en enkel simpele fileservers hiervoor de gebruiken. Zo heeft ieder systeem zijn eigen taken en wordt de SCM enkel belast met het up-to-date houden van de databases.

3.2.3 Test 3: Replication

Replication is een heel goed systeem om bij harde schijf en server crashes geen data te verliezen. Zo blijven de data beschikbaar voor iedereen via het gerepliceerde volume. Natuurlijk zal de systeembeheerder wel een goede controle moeten uitvoeren om zo snel mogelijk fouten op te sporen. Want wanneer dan ook nog het gerepliceerde volume in gevaar komt, is er wel sprake van dataverlies.

Om het fenomeen replication te testen kunnen we eens het volume van de tweede server mounten.

```
cfs mkmoint /coda/codascm.khlim.be/codaservervol codaserver1
cd /coda/codascm.khlim.be/codaservervol
```

Hier maken we een tekstbestand aan. Wanneer we nu op beide servers gaan kijken in `/vicepa/0/0` zien we dat op beide servers dit tekstbestand aanwezig is. Hieruit kunnen we besluiten dat replication werkt.

3.2.4 Test 4: Mounten van een onbestaand volume

Wanneer we een volume proberen te mounten dat niet bestaat bvb:

```
cfs mkmount /coda/codascm.khlim.be/codascmvol4 codascmvol4
```

Dan zien we in `/coda/codascm.khlim.be/` de directory codascmvol4 in het rood staan.
Deze directory is ook niet toegankelijk!
Om het volume weg te krijgen, gebruiken we het commando "`cfs rmmount`".

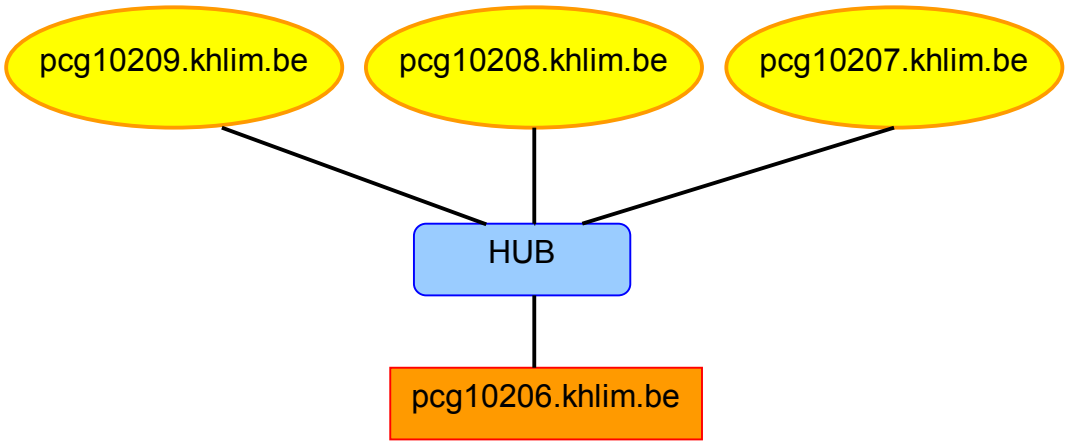
3.3 Voorbereiding, installatie en configuratie van OpenAFS

Nu zal ik de volledige praktische realisatie van OpenAFS behandelen. Hiermee bedoel ik ook weer de nodige voorbereidingen die getroffen moeten worden, de installatieprocedure en het configureren van OpenAFS tot een goed functionerend systeem.
Ik heb dit systeem onder de distributie Gentoo getest met kernel 2.4.26.

3.3.1 Voorbereiding

Zoals we reeds gezien hebben in 1.3.3 OpenAFS , bestaan er in een OpenAFS vier servertypes. Deze servertypes verschillen in de processen die ze uitvoeren. Omdat het bijvoorbeeld voor onze toepassing niet nodig is meer dan drie database servermachines te hebben, zullen we weer een soort van schema van ons systeem moeten maken. Zo hebben we een duidelijk overzicht hebben van het systeem voor we aan de installatie ervan beginnen.

Schema van mijn OpenAFS-systeem:



Hierbij is pcg10209.khlim.be zowel een fileserver, als een databaseserver, als een binary distribution server, als een system control machine (SCM).
De andere twee servers (pcg10208 en pcg10207) zijn identieke servers en draaien beide zowel alle databaseprocessen als de fileserverprocessen.
Ik heb ook op alle servermachines clientfunctionaliteit voorzien, die we achteraf wanneer het volledige systeem werkt eventueel kunnen verwijderen.
De pcg10206.khlim.be is de simpele clientmachine.
Wanneer al de taken van iedere machine gekend zijn, moeten we de datapartities en cachepartities gaan aanmaken.
Op iedere servermachine heb ik 2 datapartities van elk 3,6Gb voorzien en 1 cachepartitie van 559Mb.
Het aanpassen van de partities en partitietabellen gebeurt met het commando cfdisk.

```
cfdisk /dev/hda
```

Wanneer de partities aangemaakt zijn, moeten we er nog bestandssystemen op installeren: op de datapartities heb ik het ext3-bestandssyteem geïnstalleerd en op de cachepartitie het ext2-bestandssysteem omdat dit noodzakelijk is voor de juiste werking van de client.

```
mkfs.ext2 /dev/hda9
mkfs.ext3 /dev/hda8
mkfs.ext3 /dev/hda7
```

De nummering van `/dev/hda` is natuurlijk volledig afhankelijk van de manier waarop uw partities ingedeeld en geordend zijn.
Nu is het belangrijk enkele directories aan te maken waarin we deze partities dan kunnen mounten.

```
mkdir /vicepx
mkdir /vicepx
mkdir /usr/vice/cache
```

Met `/vicepx` is de x afhankelijk van welke server u bent aan het installeren: de eerste datapartitie van de eerste server wordt gemount in `/vicepa`, de tweede datapartitie in /vicepb en zo gaat u verder tot `/vicepz`, als u dan nog meer datapartities heeft kan u verdergaan met `/vicepaa` tot `/vicepzz`.
Nu dit klaar is, kunnen we de partities automatisch laten mounten door enkele regels bij te voegen in `/etc/fstab`.

```
nano /etc/fstab
```

In dit bestand plaatsen we dan volgende regels bij, ook weer afhankelijk van de indeling en volgorde van partities.

```
/dev/hda9       /usr/vice/cache ext2    auto
/dev/hda8       /vicepx         ext3    auto
/dev/hda7       /vicepx         ext3    auto
```

Wanneer we nu een reboot doen van ons systeem worden de data- en cachepartities automatisch gemount.
Nu rest ons nog enkel twee bestanden aan te passen.

```
nano /usr/portage/net-fs/openafs/files/ThisCell
```

Hierin zet u de persoonlijke cellnaam.

```
nano /usr/portage/net-fs/openafs/files/CellServDB
```

Hierin zet u de cellnaam en het ip-adres van deze machine.

Nu zijn we klaar om verder te gaan met de installatie van OpenAFS!

3.3.2 Installatie en configuratie van de eerste server

De eerste stap bij de installatie van OpenAFS is de nodige pakketten downloaden. Omdat we ervoor gekozen hebben deze installatie onder de distributie Gentoo uit te voeren, dienen we slechts één commando in te geven om alle bestanden van OpenAFS te downloaden.

Op dit ogenblik is de laatste stabiele versie OpenAFS 1.2.13, wanneer je nu met het commando emerge alle bestanden wil afhalen van OpenAFS zal je merken dat deze automatisch de versie 1.2.10 afhaalt.
Omdat deze versie nog de zogenaamde Quorum-bug bevat, is het niet mogelijk met deze versie een goed werkend systeem op te zetten.
De oplossing hiervoor is werken met een andere distributie of de laatste versie gebruiken die Gentoo ondersteunt. Ik heb gekozen voor de tweede optie.
Om de laatste versie te kunnen downloaden, moeten we in `/usr/portage/profiles/package.mask` de voorwaarde voor OpenAFS uitcommentariëren door er een # voor te plaatsen.

In dit bestand kunnen namelijk versies uitgeschakeld of als het ware gemaskeerd worden omwille van verschillende redenen. Bijvoorbeeld door een serieuze bug, of omdat het nog een beta-versie is, … .
Wanneer dit gebeurd is, moeten we de bestanden downloaden.

```
ACCEPT_KEYWORDS=”~x86” emerge –v openafs
```

Dit commando zal heel wat tijd in beslag nemen, want nu wordt OpenAFS 1.2.11 afgehaald, gecompileerd en geïnstalleerd.
Wanneer dit commando uitgevoerd is moeten we twee bestanden verwijderen.

```
rm /usr/vice/etc/ThisCell
rm /usr/vice/etc/CellServDB
```

Dan voeren we volgend commando uit voor de initialisatie van de bosserver die alle andere AFS-processen controleert op deze server.

```
/usr/afs/bin/bosserver –noauth &
```

Bij dit commando wordt een optie -noauth meegegeven om de authorisatie uit te schakelen omdat we nog geen administrator aangemaakt hebben.
Het & teken wil zeggen dat we het commando in de achtergrond uitvoeren.
Door dit commando worden ook de twee bestanden terug aangemaakt in `/usr/vice/etc/` , namelijk ThisCell en CellServDB.
Nu gaan we de cell een naam geven.

```
/usr/afs/bin/bos setcellname pcg10209.khlim.be khlimafs -noauth
```

Ik geef mijn cell dus de naam khlimafs vanop pcg10209.khlim.be met authorisatie uitgeschakeld.
Nu gaan we de vier databaseserverprocessen starten.

```
/usr/afs/bin/bos create pcg10209.khlim.be kaserver simple /usr/afs/bin/kaserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10209.khlim.be buserver simple /usr/afs/bin/buserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10209.khlim.be ptserver simple /usr/afs/bin/ptserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10209.khlim.be vlserver simple /usr/afs/bin/vlserver –cell \
        khlimafs -noauth
```

Nu kunnen we een kijkje gaan nemen in het bestand `/usr/afs/local/BosConfig`, en dan zien we dat er vier entries gecreëerd zijn voor de vier databaseprocessen.

```
nano /usr/afs/local/BosConfig
```

Ook zijn deze vier processen reeds gestart, dit kunnen we verifiëren met het commando pstree of met het commando “`bos status`”.

```
pstree
```

Of

```
usr/afs/bin/bos status pcg10209.khlim.be –noauth
```

Nu deze 4 processen gestart zijn, kunnen we beginnen aan de initialisatie van de beveiliging van onze cell. Hiervoor gaan we twee entries creëren in de authentificatie database. Eén entry voor de systeembeheerder of administrator en één voor de afs server processen.
Na ingave van volgend commando komen we in een interactieve mode terecht

```
/usr/afs/bin/kas –cell khlimafs –noauth
```

Wanneer er ka> voor een commando staat, wil dit zeggen dat het in interactieve mode wordt ingegeven.

```
ka> create afs
```

Na dit commando wordt er een paswoord gevraagd dat u zelf mag kiezen.

```
ka> create admin
```

Ook voor deze account wordt er weer een paswoord gevraagd dat u zelf mag kiezen.

```
ka> examine afs
```

Nu zal de inhoud van de entry afs op het scherm getoond worden

```
ka> setfields admin -flags admin
```

We moeten ervoor zorgen dat bij de account voor de systeembeheerder een admin vlag geset wordt, zodat deze speciale commando's voor het beheren van AFS mag uitvoeren.

```
ka> examine admin
```

Nu wordt de inhoud van de entry admin op het scherm getoond. Let goed op de admin-vlag.

```
ka> quit
```

Na dit commando gaan we uit de interactieve mode.

Nu gaan we de gebruiker admin toevoegen aan de gebruikerslijst namelijk `/usr/afs/etc/UserList`.

```
/usr/afs/bin/bos adduser pcg10209.khlim.be admin –cell khlimafs –noauth
```

Met het volgende commando gaan we een geëncrypteerde sleutel aanmaken in `/usr/afs/etc/KeyFile`. Wanneer er na dit commando gevraagd wordt naar een input key moeten we het paswoord ingeven dat we in de interactieve mode hebben opgegeven voor de entry afs.

```
/usr/afs/bin/bos addkey pcg10209.khlim.be –kvno 0 –cell khlimafs –noauth
```

Om te verifiëren of de key bijgevoegd is, gebruiken we volgend commando.

```
/usr/afs/bin/bos listkeys pcg10209.khlim.be –cell khlimafs –noauth
```

Nu gaan we in de protection database een entry creëren voor de administrator. Standaard krijgt de administrator een uid 1. Stel dat we al een admin account in `/etc/passwd` hebben, dan kunnen we best de optie –id meegeven met erachter hetzelfde id als in `/etc/passwd` zit.

```
/usr/afs/bin/pts createuser –name admin –cell khlimafs –noauth
```

Nu gaan we de gebruiker admin aan de groep system:administrators toevoegen.

```
/usr/afs/bin/pts adduser admin system:administrators –cell khlimafs –noauth
```

Om dit lidmaatschap nu te verifiëren, gebruiken we volgend commando.

```
/usr/afs/bin/pts membership admin –cell khlimafs -noauth
```

Nu herstarten we alle AFS serverprocessen.

```
/usr/afs/bin/bos restart pcg10209.khlim.be –all –cell khlimafs –noauth
```

Dan starten we de fileserver, volumeserver en salvager.

```
/usr/afs/bin/bos create pcg10209.khlim.be fs fs /usr/afs/bin/fileserver \
        /usr/afs/bin/volserver /usr/afs/bin/salvager –cell khlimafs –noauth
```

Even verifiëren of alle processen normaal werken.

```
/usr/afs/bin/bos status pcg10209.khlim.be –long –noauth
```

Omdat dit de eerste server is in ons OpenAFS systeem moeten we een eerste volume aanmaken en dit noemen we root.afs.

```
/usr/afs/bin/vos create pcg10209.khlim.be /vicepx root.afs –cell khlim –noauth
```

Let wel op dat u op de plaats waar de x staat in `/vicepx` de juiste letter invult, u moet de naam van de directory geven waarin de datapartitie gemount is.

Omdat dit de eerste server is, maken we van deze server ook de Server Control Machine of SCM, dit wil zeggen dat alle bestanden in `/usr/afs/etc` door deze machine verdeeld worden over alle andere servers. Deze server moet dus ook het upserverproces draaien omdat dit het serverdeel is van de update server.

```
/usr/afs/bin/bos create pcg10209.khlim.be upserver simple "/usr/afs/bin/upserver \
        –crypt /usr/afs/etc –clear /usr/afs/bin" –cell khlimafs –noauth
```

Na dit commando hebben we alle processen aan de gang die nodig zijn voor onze eerste server en gaan we verder met het implementeren van de clientfunctionaliteit op deze server.

3.3.3 Clientfunctionaliteit implementeren in de eerste server

Om deze server ook als client te kunnen gebruiken, moeten we enkele aanpassingen doorvoeren.
De eerste stap is namelijk een bestand aanmaken in `/usr/vice/etc` met de naam cacheinfo.

```
nano /usr/vice/etc/cacheinfo
```

In dit bestand komt volgende regel.

```
/afs:/usr/vice/cache:500000
```

Het eerste deel van de regel slaat op de directory waarin alle afs data gemount worden en het tweede deel geeft de directory aan waarin de cachepartitie gemount is.
Het laatste deel is de grootte van de cache in kb, deze mag niet groter zijn dan 90% van de cachepartitie bij een cache op harde schijf.

Dan maken we een directory aan waarin we alle data van de AFS-servers kunnen mounten.

```
mkdir /afs
```

Nu kunnen we de afs client starten.

```
/etc/init.d/afs start
```

En inloggen met het klog commando.

```
klog admin
```

Nu kunnen we enkele Acces Control Lists (ACL's) installeren, met deze lijsten kunnen we instellen wie toegang heeft tot bepaalde directories en wie niet. We kunnen ook instellen wie deze directories enkel mag bekijken en wie er ook dingen mag in wijzigen, enz…

```
/usr/afs/bin/fs setacl /afs system:anyuser rl
```

Dit betekent dat iedereen toegang heeft tot de directory `/afs` maar enkel om te lezen en te zoeken.

Als volgt gaan we een root volume aanmaken en het readonly mounten in `/afs/khlimafs` en lees/schrijf mounten in `/afs/.khlimafs`.

```
/usr/afs/bin/vos create pcg10209.khlim.be /vicepx root.cell
/usr/afs/bin/fs mkmount /afs/khlimafs root.cell
/usr/afs/bin/fs setacl /afs/khlimafs system:anyuser rl
/usr/afs/bin/fs mkmount /afs/.khlimafs root.cell –rw
```

Om de afs client te stoppen, moeten we ervoor zorgen dat we niet in de directory `/afs` zitten en dan volgend commando ingeven.

```
/etc/init.d/afs stop
```

Ziezo, de clientfunctionaliteit is volledig opgezet en we kunnen nu alvast beginnen met experimenteren onder OpenAFS. In het volgende hoofdstuk zullen we het hebben over het opzetten van een NTP-client voor we aan het hoofdstuk over het toevoegen van een tweede server kunnen beginnen.

3.3.4 Opzetten van een NTP-client

De NTP in NTP-client staat voor Network Time Protocol, zoals u waarschijnlijk al kan raden is dit een protocol om tijdssynchronisatie uit te voeren via een netwerk. Bij onze toepassing is dit nodig omdat de klokken van zowel alle servers als alle clienten goed gesynchroniseerd moeten zijn om OpenAFS te laten werken. Deze synchronisatie is cruciaal voor zowel het beveiligingsprotocol Kerberos als voor de gedistribueerde databasetechnologie Ubik.

We zullen dus verplicht zijn de klok van iedere server en client te laten synchroniseren met een externe NTP-server. Dit doen we als volgt.
Eerst moeten we alle pakketten van NTP downloaden, compileren en installeren.

```
emerge ntp
```

Als dit gebeurd is, moeten we twee configuratiebestanden aanpassen namelijk `/etc/ntp.conf` en `/etc/conf.d/ntp-client`.

```
nano /etc/ntp.conf
```

In dit bestand plaatsen we de volgende regel bij :

```
server 195.13.23.5
```

Dan openen we `/etc/conf.d/ntp-client`.

```
nano /etc/conf.d/ntp-client
```

In dit bestand wijzigen we onderstaande regels :

```
NTPCLIENT_CMD=”ntpdate”
NTPCLIENT_OPTS=”-b 195.13.23.5”
```

Dan kunnen we de opstartscripts toevoegen aan de default runlevel.

```
rc-update add ntpd default
rc-update add ntp-client default
```

Nu zijn we klaar voor een reboot zodat deze deamons automatisch worden gestart bij het booten van de pc.

Om toch nog even terug te komen op de wijzigingen in de configuratiebestanden wil ik vermelden dat ik het IP-adres gekozen heb van een NTP-server die geschikt is voor mijn OpenAFS systeem. Indien u bijvoorbeeld in Amerika of ergens anders in Europa een OpenAFS systeem wil opzetten zal u het internet moeten raadplegen om een NTP-server te vinden die in de buurt opgesteld staat om zo weinig mogelijk vertraging te bekomen. Hier alvast een URL om u op weg te helpen: http://ntp.isc.org/bin/view/Servers/StratumTwoTimeServers
Onze NTP-client is klaar en we kunnen dus verder met het toevoegen van onze tweede server aan ons OpenAFS systeem.

3.3.5 Installatie en configuratie van een tweede server

Zoals ik reeds in het vorige hoofdstuk verteld heb, is het noodzakelijk op de eerste en op deze server eerst een NTP-client te installeren om zo de synchronisatie van de klokken te verzekeren.
Dan pas kunnen we beginnen aan de installatie en configuratie van de tweede server. Ik geef hier enkel nog de stappen die na mekaar uitgevoerd moeten worden omdat ik bij de installatie van de eerste server reeds uitleg gegeven heb bij deze stappen. We zullen enkel de nieuwe commando's nader bekijken.

```
nano /usr/portage/profiles/package.mask
```

De voorwaarde voor net-fs/openafs uitcommentariëren in dit bestand.

```
ACCEPT_KEYWORDS="~x86" emerge –v openafs
rm –r /usr/vice/etc/CellServDB
rm –r /usr/vice/etc/ThisCell
/usr/afs/bin/bosserver –noauth &
/usr/afs/bin/bos setcellname pcg10207.khlim.be khlimafs –noauth
```

In het volgende commando zullen we het clientdeel van de update server starten voor het up-to-date houden van de configuratiebestanden zodat we enkel op de SCM de configuratiebestanden moeten aanpassen.

```
/usr/afs/bin/bos create pcg10207.khlim.be upclientetc simple "/usr/afs/bin/upclient \
        pcg10209.khlim.be –t 180 /usr/afs/etc" –cell khlimafs –noauth
```

Nu gaan we alle configuratiebestanden kopiëren van de eerste server naar deze server, dan zijn we zeker dat alle instellingen gelijk zijn.

```
scp –r root@pcg10209.khlim.be:/usr/afs/etc/* /usr/afs/etc
```

Na dit commando zal er naar het paswoord van de gebruiker root gevraagd worden op pcg10209.khlim.be.
Volgend commando initialiseert het clientdeel van de update server voor het automatisch aanpassen van de `/usr/afs/bin` directory.

```
/usr/afs/bin/bos create pcg10207.khlim.be upclientbin simple "/usr/afs/bin/upclient \
        pcg10209.khlim.be –t 180 –clear /usr/afs/bin" –cell khlimafs –noauth
```

Nu gaan we op de SCM een nieuwe server toevoegen door het volgende commando in te geven, de configuratiebestanden worden dan op de SCM aangepast en automatisch verspreid naar deze server via het clientdeel van de update server.

```
/usr/afs/bin/bos addhost pcg10209.khlim.be pcg10207.khlim.be
```

Nu gaan we verder zoals bij de installatie van de eerste server.

```
/usr/afs/bin/bos create pcg10207.khlim.be fs fs /usr/afs/bin/fileserver \
        /usr/afs/bin/volserver /usr/afs/bin/salvager –cell khlimafs –noauth
/usr/afs/bin/bos create pcg10207.khlim.be kaserver simple /usr/afs/bin/kaserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10207.khlim.be buserver simple /usr/afs/bin/buserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10207.khlim.be ptserver simple /usr/afs/bin/ptserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos create pcg10207.khlim.be vlserver simple /usr/afs/bin/vlserver –cell \
        khlimafs -noauth
/usr/afs/bin/bos restart pcg10209.khlim.be kaserver buserver ptserver vlserver –noauth
/usr/afs/bin/bos restart pcg10207.khlim.be kaserver buserver ptserver vlserver –noauth
```

Nu moeten we enkel nog de Volume Location Database synchroniseren.

```
/usr/afs/bin/vos syncvldb pcg10209.khlim.be –cell khlimafs –noauth
/usr/afs/bin/vos syncserv pcg10209.khlim.be –cell khlimafs –noauth
```

De tweede server is toegevoegd en werkt, om nu nog een derde server namelijk pcg10208.khlim.be toe te voegen gaan we identiek te werk als bij het toevoegen van deze tweede server. Dit doen we omdat deze derde server dezelfde functionaliteit heeft als de tweede. Maar indien je bijvoorbeeld nog een simpele fileserver wil bijplaatsen moet je de databaseprocessen niet initialiseren. Dan mag je dus stoppen na het bos create commando van de fileserver, volserver en salvager.

3.3.6 Opzetten van een simpele client

Zoals in het schema dat ik tijdens de voorbereiding gemaakt heb, gaan we nu nog een machine namelijk pcg10206.khlim.be als simpele client installeren.
Voor we verder kunnen gaan, moet een cachepartitie gecreëerd worden en gemount in `/usr/vice/cache`, ook de NTP-client moet opgezet zijn en werken.
Omdat we reeds clientfunctionaliteit in de eerste server hebben ingebouwd, is uitleg bij de commando's overbodig.

```
rm –r /usr/vice/etc/ThisCell
rm –r /usr/vice/etc:CellServDB
scp root@pcg10209.khlim.be:/usr/afs/etc/ThisCell /usr/vice/etc
scp root@pcg10209.khlim.be:/usr/afs/etc/CellServDB /usr/vice/etc
mkdir /afs
scp root@pcg10209.khlim.be:/usr/vice/etc/cacheinfo /usr/vice/etc
/etc/init.d/afs start
```

En de client werkt, we kunnen nu inloggen met het klog commando en dan kunnen we in `/afs/khlimafs` experimenteren.

3.3.7 Aanmaken van user-accounts en volumes

Nu ons OpenAFS systeem werkt, ga ik uitleggen hoe u op een makkelijke manier gebruikers kunt aanmaken met hun eigen rechten en homedirectory.
Het is belangrijk dat wanneer u gebruikers aanmaakt, u ervoor zorgt dat deze gebruiker ook steeds bestaat in `/etc/passwd` op de lokale client. En dat de Linux UID en de AFS UID aan elkaar gelijk zijn. Zo kan een correcte werking worden gegarandeerd.

Enkele begrippen die ik vooraf wil uitleggen zijn:

- Protection database: Zet de gebruikersnaam om naar userID (UID) en zoekt op tot welke groepen deze gebruiker behoort.
- Authentication database: Neemt het paswoord op in een geëncrypteerde vorm.
- Home volume: Slaat alle bestanden op in de homedirectory van de gebruiker op 1 partitie.
- Mountpoint: Maakt de inhoud van het gebruikersvolume zichtbaar en toegankelijk.

Om verder te gaan, moeten we eerst als admin op een client inloggen met het commando klog.
Allereerst gaan we een nieuw volume aanmaken om de gebruikers in onder te brengen en we mounten dit tweemaal. De eerste keer als read-only versie, en de tweede keer als read/write. Om hiertussen een onderscheid te maken, hebben we het read/write mountpoint voorzien van een . voor de naam.

```
vos create pcg10209.khlim.be /vicepb usr.
fs mkmount /afs/khlimafs/users usr.
fs setacl /afs/khlimafs/users system:anyuser rl
fs mkmount /afs/.khlimafs/.users usr. –rw
```

Om een gebruiker toe te voegen met zijn eigen rechten en zijn eigen homedirectory maken we een USS template file aan, dit is een bestand waarin alle eigenschappen staan die aangemaakt moeten worden. Hieronder vindt u het bestand dat ik aangemaakt heb.

```
nano /etc/uss.template
```

In dit bestand komen volgende twee regels :

```
A $USER 0 noreuse 5 01
V usr.$USER pcg10209.khlim.be /vicepb 51200 /afs/.khlimafs/users/$USER $UID \
        $USER all system:anyuser none
```

Daarna kunnen we gebruikers beginnen toe te voegen. Let wel op dat u op de plaats van UNIXUID de UID van de gebruiker uit `/etc/passwd` in volgende commando's zet.

```
uss.add –admin admin –user dkrawinc –uid UNIXUID –pass changeme \
        –template /etc/uss.template -verbose
uss.add –admin admin –user lrutten –uid UNIXUID –pass changeme \
        –template /etc/uss.template –verbose
```

Nu zijn er twee gebruikers voor AFS aangemaakt namelijk dkrawinc en lrutten. Ze hebben beiden hun eigen homedirectory in `/afs/khlimafs/users` en hun eigen paswoord namelijk “changeme”.
Ze kunnen beiden inloggen en hun eigen directory raadplegen, ze hebben geen toegang tot andermans directory.

3.3.8 Verplaatsen van volumes tussen servers

Waarom zouden we volumes willen verhuizen van één server naar een andere?

- Om alle volumes van een server te verhuizen naar een andere server, zodat de lege server dan zijn onderhoud kan ondergaan. Bvb: een OS-upgrade of een hardware herstelling.
- Om ruimte vrij te maken omdat de partitie overvol raakt en de gebruikers hun bestanden niet meer kunnen opslaan.
- Omdat een server overladen wordt omdat deze veel meer volumes bevat dan andere servers van dezelfde grootte.

Om deze redenen hebben de ontwerpers van OpenAFS ervoor gezorgd dat volumes met slechts één commando te verplaatsen zijn van de ene partitie op een server naar een andere partitie op een andere server.
Zelf heb ik het ook uitgetest en hieronder vindt u het commando dat ik getest heb.

```
vos move usr.dkrawinc pcg10209.khlim.be /vicepb pcg10208.khlim.be /vicepe
```

Tijdens het verplaatsen van het volume ondervindt de gebruiker geen last van deze verplaatsing, ook de Volume Location Database wordt automatisch aangepast zonder dat de administrator moet tussenkomen.

3.4 Toemaatje: Implementatie van Kerberos

Dit hoofdstuk betreft een extra opdracht. De integratie van Kerberos in OpenAFS wordt hier toegelicht. Omdat in Gentoo de nodige migration tools niet beschikbaar zijn heb ik voor deze opdracht gebruik gemaakt van de Linux-distributie Debian Sarge. De gebruikte kernel-versie is 2.4.27.

3.4.1 Voorbereiding

Bij deze opdracht is het niet zo belangrijk een schema te ontwerpen van het systeem omdat ik slechts één pc gebruik. Het spreekt dan ook voor zich dat deze pc zowel over server- als clientfunctionaliteit beschikt.
Net als bij de installatie van OpenAFS in Gentoo moeten we datapartities en cachepartities gaan aanmaken. Op iedere servermachine heb ik twee datapartities van elk 3,6Gb voorzien en één cachepartitie van 559Mb.
Het aanpassen van de partities en partitietabellen gebeurt in normale omstandigheden met het commando cfdisk, maar omdat ik reeds over deze partities beschik is dit overbodig.

```
cfdisk /dev/hda
```

Wanneer de partities aangemaakt zijn moeten we er bestandssystemen op installeren, op de datapartities heb ik het ext3-bestandssyteem geïnstalleerd en op de cachepartitie het ext2-bestandssysteem omdat dit noodzakelijk is voor de juiste werking van de client.

```
mkfs.ext2 /dev/hda9
mkfs.ext3 /dev/hda8
mkfs.ext3 /dev/hda7
```

De nummering van `/dev/hda` is natuurlijk opnieuw volledig afhankelijk van de manier waarop uw partities ingedeeld en geordend zijn. Nu is het belangrijk enkele directories aan te maken waarin we deze partities kunnen mounten.

```
mkdir /vicepa
mkdir /vicepb
mkdir /usr/vice/cache
```

Nu kunnen we de partities automatisch laten mounten door enkele regels bij te voegen in `/etc/fstab`.

```
nano /etc/fstab
```

In dit bestand plaatsen we dan volgende regels bij, ook weer afhankelijk van de indeling en volgorde van partities.

```
/dev/hda9       /usr/vice/cache ext2    auto
/dev/hda8       /vicepb         ext3    auto
/dev/hda7       /vicepa         ext3    auto
```

Wanneer we nu een reboot doen van ons systeem worden de data- en cachepartities automatisch gemount.

3.4.2 Installatie en configuratie van Kerberos en OpenAFS

Nu de nodige voorbereidingen getroffen zijn kunnen we beginnen aan de installatie. Bij de eerste stap van de installatie zien we dat Debian net als Gentoo gebruik maakt van één commando (nl. apt-get install) om een pakket af te halen van het internet, te compileren en te installeren.

```
apt-get install openafs-fileserver openafs-dbserver openafs-krb5 krb5-client \
    libpam-openafs-session krb5-user krb5-admin-server krb5-doc kernel-source-2.4.27
```

Nu moeten we een module compileren om ervoor te zorgen dat onze client kan werken. Ik heb heel wat problemen ondervonden bij het compileren van deze module en heb daarom een speciale tool gebruikt, namelijk “module-assistent”.
Door volgende vier commando’s in te geven worden zowel de module-assistent als de module zelf geïnstalleerd:

```
aptitude install module-assistant
module-assistant update
module-assistant prepare
module-assistant auto-install openafs
```

Na het compileren van deze module kunnen we verder met de configuratie van Kerberos en OpenAFS.
Allereerst zullen we twee configuratie bestanden moeten aanpassen `/etc/krb5kdc/kdc.conf` en `/etc/krb5.conf`.
Deze bestanden heb ik als volgt ingesteld:

/etc/krb5kdc/kdc.conf

```
[kdcdefaults]
        kdc_ports = 750,88

[realms]
KHLIMAFS = {
                database_name = /var/lib/krb5kdc/principal
                admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
                acl_file = /etc/krb5kdc/kadm5.acl
                key_stash_file = /etc/krb5kdc/stash
                kdc_ports = 750,88
                max_life = 10h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                master_key_type = des3-hmac-sha1
                supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal des:normal \\
                des:v4 des:norealm des:onlyrealm des:afs3
                default_principal_flags = +preauth
        }
```

/etc/krb5.conf

```
[libdefaults]
        default_realm = KHLIMAFS
# The following krb5.conf variables are only for MIT Kerberos.
        krb4_config = /etc/krb.conf
        krb4_realms = /etc/krb.realms
        kdc_timesync = 1
        ccache_type = 4
        forwardable = true
        proxiable = true
# The following encryption type specification will be used by MIT Kerberos
# if uncommented.  In general, the defaults in the MIT Kerberos code
# are correct and overriding these specifications only serves to disable
# new encryption types as they are added, creating interoperability problems.
#default_tgs_enctypes =aes256-cts arcfour-hmac-md5 des3-hmac-sha1 des-cbc-crc des-cbc-md5
#default_tkt_enctypes = aes256-cts arcfour-hmac-md5 des3-hmac-sha1 des-cbc-crc des-cbc-md5
#permitted_enctypes = aes256-cts arcfour-hmac-md5 des3-hmac-sha1 des-cbc-crc des-cbc-md5

# The following libdefaults parameters are only for Heimdal Kerberos.
        v4_instance_resolve = false
        v4_name_convert = {
                host = {
                        rcmd = host
                        ftp = ftp
                }
                plain = {
                        something = something-else
                }
        }

[realms]
KHLIMAFS = {
        kdc = pcg10207.khlim.be
        admin_server = pcg10207.khlim.be
}

[domain_realm]
        .khlimafs = KHLIMAFS
        khlimafs = KHLIMAFS

[login]
        krb4_convert = true
        krb4_get_tickets = true

[logging]
        kdc = FILE:/var/log/krb5kdc.log
        admin_server = FILE:/var/log/kadmin.log
        default = FILE:/var/log/krb5lib.log
```

Wanneer deze bestanden aangepast zijn met uw persoonlijke instellingen kunnen we overgaan tot het creëren van onze realm. Deze realm kan je best dezelfde naam geven als de naam van de openafs-cell.

```
/usr/sbin/kdb5_util create –r KHLIMAFS –s
```

De -s optie slaat op het feit dat we een zogenaamd stash bestand gebruiken om de master key bij te houden.

Hierna zal u merken dat er automatisch 5 bestanden aangemaakt zijn in `/etc/krb5kdc/` en in `/var/lib/krb5kdc/`
Nu gaan we aan één van deze automatisch gecreëerde bestanden (nl. kadm5.acl) een regel toevoegen. Dit om een gebruiker “admin” alle rechten te geven.

```
pico /etc/krb5kdc/kadm5.acl
```

In dit bestand plaatsen we volgende regel:

```
*/admin@KHLIMAFS *
```

Dan starten we kadmin.local, dit is de administratieve interface van Kerberos. In deze interface kan een adminstrator principals toevoegen, wijzigen en verwijderen.
Wij gaan een nieuwe principal aanmaken die root/admin heet.

```
/usr/sbin/kadmin.local
kadmin.local: addprinc root/admin@KHLIMAFS
kadmin.local: ktadd /var/lib/krb5kdc/kadm5.keytab kadmin/admin kadmin/changepw
kadmin.local: quit
```

Met addprinc wordt een nieuwe principal aangemaakt en met ktadd maken we een keytab bestand dat de kadmin principals bevat die aangemaakt werden bij de initialisatie van onze realm.

Nu kunnen we zowel het KDC-proces (Key Distribution Center) als het administratieproces starten:

```
/usr/sbin/krb5kdc
/usr/sbin/kadmind
```

Voor we verder kunnen moeten we de rechten van de directory /etc/openafs/server instellen en onze `/etc/hosts` juist configureren!

```
chmod 755 /etc/openafs/server
pico /etc/hosts
```

Dan gaan we AFS als principal aan kerberos toevoegen.

```
/usr/sbin/kadmin.local
kadmin.local: addprinc –kvno 5 –e des-cbc-crc:afs3 afs
kadmin.local: addprinc –kvno 5 –e des-cbc-crc:afs3 afs/khlimafs
kadmin.local: quit
```

Na het toevoegen van beide principals wordt er naar een paswoord gevraagd, u kiest best hetzelfde paswoord voor beide principals.

Dan starten we het Basic OverSeer Server proces.

```
bosserver –noauth &
```

Met het volgende commando gaan we een geëncrypteerde sleutel maken. Wanneer er na dit commando gevraagd wordt naar een input key, moeten we het paswoord ingeven dat we in de administratieve interface van Kerberos hebben ingegeven, bij het toevoegen van de AFS-service als principal.

```
bos addkey -server pcg10207.khlim.be -kvno 5 -cell khlimafs -noauth
```

Dan starten we het ptserverproces en voegen dit toe aan het BosConfig bestand.

```
bos create -server pcg10207.khlim.be -instance ptserver -type simple \
        -cmd /usr/lib/openafs/ptserver -cell khlimafs -noauth
```

We gaan een nieuwe gebruiker creëren, deze toevoegen aan de protection database en aan de groep system:administrators en achteraf moeten we alle processen herstarten.

```
bos adduser pcg10207.khlim.be admin -cell khlimafs -noauth
pts createuser -name admin -noauth
pts adduser admin system:administrators -noauth
bos restart pcg10207.khlim.be -all -noauth
```

Nu starten we de overige processen en voegen ook deze toe aan het BosConfig bestand zodat ze automatisch opgestart worden bij het opstarten van het bosserver proces.

```
bos create -server pcg10207.khlim.be -instance fs -type fs –cmd \
        /usr/lib/openafs/volserver -cmd /usr/lib/openafs/salvager –cmd \
        /usr/lib/openafs/vlserver -cell khlimafs -noauth
```

In het volgende commando creëren we het eerste volume, ook wel het rootvolume genoemd.

```
vos create -server pcg10207.khlim.be -partition /vicepa -name root.afs -cell khlimafs \
        -noauth
```

Dan sluiten we alle processen af.

```
vos shutdown pcg10207.khlim.be
```

We passen in `/etc/openafs/afs.conf.client` volgende regel aan:

“AFSCLIENT=true”

Nu starten we alles opnieuw.

```
bosserver -noauth &
/etc/init.d/openafs-client start
```

Vanaf dit moment is het mogelijk om via kinit in te loggen, als we dan achteraf `aklog` als commando ingeven zullen we automatisch toegang krijgen tot de afs ruimte.
We hoeven dus maar één keer ons paswoord in te geven bij kinit en krijgen dan een ticket van de KDC dat een bepaalde tijd geldig blijft. Achteraf hoeven we voor de services die door Kerberos gekend zijn geen paswoord meer in te geven.

We kunnen nu in OpenAFS gebruikers en volumes gaan aanmaken met de juiste commando's. Ook kunnen we deze volumes mounten in de afs-directory en de toegangsrechten op deze volumes aanpassen met Acces Control Lists.
Maar omdat dit reeds bij de installatie van OpenAFS besproken is verwijs ik naar hoofdstuk "*3.3.3 Clientfunctionaliteit implementeren in de eerste server*" .

Om er nu voor te zorgen dat bij het inloggen, onmiddellijk na het opstarten gebruik gemaakt wordt van Kerberos 5 gebruiken we PAM. Dit staat voor Pluggable Authentication Modules en is een systeem waarmee op modulaire wijze een toegangscontrole tot een toepassing geregeld kan worden.

Toepassingen die geschikt zijn voor PAM roepen binnen een gestandaardiseerd framework (API) modules aan om zo authentificatie te bekomen. De systeembeheerder kan regelen hoe bevoegdheden worden uitgegeven: de authentificatie wordt als het ware volledig omgebogen, buiten de toepassing.
Een PAM module kan een bestand lezen, of een smartcard, of via een netwerk aan een database om gegevens vragen. In feite kan een PAM module alles wat de systeembeheerder ook zou kunnen, er is bijvoorbeeld een module die kijkt of je wel een Home-directory hebt en zo niet maakt PAM deze aan. De modules zijn stapelbaar, je kan een serie van modules doorlopen, die onder elkaar data uitwisselen. De totale keten van modules "beslist" dan of men al dan niet bevoegd is.

Al deze modules bevinden zich in `/lib/security`. De configuratiefiles die dienen aangepast te worden zijn `/etc/login.defs` , `/etc/pam.d/common-auth` , `/etc/pam.d/common-account` , `/etc/pam.d/common-session` en `/etc/pam.d/common-password`.
In `/etc/login.defs` moet volgende regel aangepast worden:

```
CLOSE_SESSIONS yes
```

`/etc/pam.d/common-auth` ziet er als volgt uit:

```
auth    required        pam_unix.so nullok_secure
auth    sufficient      pam_krb5.so use_first_pass
```

`/etc/pam.d/common-account` ziet er als volgt uit:

```
account         required        pam_unix.so
account         [default=bad succes=ok user_unknown=ignore service_err=ignore system_err=ignore] \\
                pam_krb5.so
```

`/etc/pam.d/common-session` ziet er als volgt uit:

```
session         required        pam_unix.so
session         optional        pam_krb5.so
session         optional        pam_openafs_session.so
```

`/etc/pam.d/common-password` ziet er als volgt uit:

```
password        required        pam_unix.so     nullok obscure  min=4   max=8  md5
password        sufficient      pam_krb5.so     use_authok
```

Wanneer al deze bestanden aangepast zijn kan je inloggen met je loginnaam en paswoord dat zowel in kerberos als in `/etc/passwd` en `/etc/shadow` bekend is. Je wordt automatisch ook ingelogd op het OpenAFS-systeem en krijgt dan toegang tot je persoonlijk AFS-volume als dit ingesteld is als home-directory in `/etc/passwd`.

3.5 Vergelijking CODA-OpenAFS

Na dit onderzoek is het belangrijk beide systemen met elkaar te vergelijken. Dit heb ik op verschillende gebieden gedaan, ieder systeem heeft namelijk zijn voor- en nadelen.
Als eerste heb ik het systeem CODA onderzocht, zoals ik reeds vermeld heb is CODA geboren uit AFS, maar dit systeem biedt zeer interessante voordelen wanneer we in een netwerk zitten met zwakke internetverbindingen en mobiele systemen.

Ik vermeld ze hier nog even:

- het werkt ook bij een verbroken verbinding (mobiele clienten)
  - Bij verbinding wordt de data automatisch geïntegreerd
  - Bandbreedte adaptatie
- het voorziet in foutafhandelingsmethodes
  - lost server/server conflicten op
  - behandelt automatisch netwerkfouten
  - behandelt automatisch verbroken verbindingen van clients

Deze eigenschappen vinden we niet terug bij OpenAFS. Tijdens dit onderzoek heb ik opgemerkt dat er veel actiever gewerkt wordt aan OpenAFS dan aan CODA, dit wil dus ook zeggen dat er meer ondersteuning is voor OpenAFS.

Beide systemen beschikken over een persistente caching, waarmee bedoeld wordt dat eens een bepaald bestand opgehaald is van een server het steeds bereikbaar blijft in de lokale cache, ook bij het herstarten van het systeem. Bij OpenAFS is dit bestand enkel leesbaar, bij CODA zowel lees- als schrijfbaar door de eigenschap: disconnected operation.

Wanneer we het aspect configuratie bekijken is CODA het snelst en makkelijkst te configureren, dit kunnen we verklaren door het bijgeleverd shell-script “Venus-setup”.

Een zeer belangrijk voordeel van CODA is de automatische replicatie van volumes. U kan namelijk een groep servers (Volume Storage Group) aanmaken met enkele volumes die automatisch op alle servers van deze groep worden bijgehouden. Dit werkt via het “write_all/read_one principe”, er wordt naar alle servers geschreven bij het aanpassen van een bestand, maar bij het lezen wordt er slechts van één server gelezen. Bij OpenAFS bestaat er ook volume replication, maar dit moet handmatig of via een cron job gebeuren.

Op gebied van beveiliging beschikken beide systemen over een Kerberos-gebaseerde authentificatie.

Momenteel is CODA enkel beschikbaar voor Linux, FreeBSD, NetBSD, Windows 95 en Windows NT. OpenAFS is beschikbaar voor bijna alle populaire besturingssystemen (Linux, IBM AIX, MacOS, Solaris, Unix, …).

Zoals u ziet kan u hieruit concluderen dat CODA over heel wat geavanceerde technieken beschikt maar dat het nog een hele weg af te leggen heeft op gebied van

flexibiliteit en ondersteuning. OpenAFS daarentegen biedt een goede ondersteuning en flexibiliteit en betere stabiliteit.
U zal dus heel wat moeten wikken en wegen om te beslissen welk systeem voor uw toepassing best gebruikt kan worden. Natuurlijk kan u net als ik beide systemen een tijdje testen en op die manier uw keuze bepalen.

Besluit en toekomstperspectieven

Besluit

Met dit eindwerk hoop ik een goed gestructureerde handleiding gemaakt te hebben voor iedere nieuweling binnen het thema Cluster Filesystems. Dit is noodzakelijk omdat deze systemen niet zonder problemen te installeren zijn. Met dit eindwerk zou iedereen, zelfs een “Linux-newbie”, in staat moeten zijn CODA of OpenAFS te installeren.
Bovendien hoop ik dat dit afstudeerrapport een grote hulp is voor mijn opdrachtgever zodat hij nu deze systemen niet meer hoeft te onderzoeken, en toch een goede handleiding ter beschikking heeft om dit thema te gebruiken en te onderwijzen.

Voor mij was het een zeer boeiende en leerrijke ervaring waar heel wat opzoekwerk en zelfstudie bij kwam kijken. Ook leerde ik bepaalde problemen op een andere, betere manier aanpakken zodat ik heel wat tijd heb leren uitsparen bij het opsporen van problemen.

Ten slotte wil ik nog vermelden dat dit eindwerk zeker niet alle details omtrent CODA en OpenAFS omvat, maar enkel een handleiding is voor installatie en basisconfiguratie ervan, met de essentiële theoretische uitleg.

Toekomstperspectieven

De mogelijke uitbreidingen en toekomstperspectieven van dit eindwerk zijn:

- Integreren van OpenLDAP in het cluster filesystem.
- Inloggen op het cluster filesystem via een grafische interface die connectie maakt met OpenLDAP om de home-directory te achterhalen.

Pas wanneer deze twee uitbreidingen gerealiseerd zijn, zal het mogelijk zijn om dit systeem veilig en gebruiksvriendelijk in een bedrijfsnetwerk te integreren.

Bibliografie

Tijdens het uitvoeren van dit afstudeerwerk heb ik heel wat informatie op het internet geraadpleegd, maar ook in enkele boeken en tijdschriften heb ik informatie gevonden. Welke URL's, boeken en tijdschriften ik geraadpleegd heb kan u in dit onderdeel terugvinden.

Internet:

- "*CODA file system*", online, http://www.coda.cs.cmu.edu/
- Von Hagen, B., "*The Coda Distributed Filesystem for Linux*", online, http://linuxplanet.com/linuxplanet/tutorials/4481/7/
- "*OpenAFS*", online, http://www.openafs.org
- "*Gentoo Linux*", online, http://www.gentoo.org
- "Gentoo Forums", online, http://forums.gentoo.org/index.php
- "*Kerberos: The Network Authentication Protocol*", online, http://web.mit.edu/kerberos/www/
- "*NTP: The Network Time protocol*", online, http://www.ntp.org
- Wachsmann, A., "*Part III: AFS-A Secure Distributed Filesystem*", online, http://www.linuxjournal.com/article/7521

Boeken en tijdschriften:

- Barret, D.J., *Linux pocket guide*, O'Reilly, 2004
- LinuxJournal, 132, april 2005, (*Part III: AFS-A Secure Distributed Filesystem*), Wachsmann, A.
- Garman, J., *Kerberos The Definitive Guide*, O'Reilly, 2003

Bijlage 1: Verklarende woordenlijst

Hier vindt u de Nederlandse betekenis van alle minder frequente woorden en termen die in deze thesis gebruikt worden.

Beta versie: Een versie waarop men nog steeds testen uitvoert betreffende de performantie en die nog fouten kan bevatten omdat ze nog niet voldoende getest is.

Bosserver: Het Basic OverSeer serverproces of het proces dat alles AFS-processen onder controle houdt.

Deamons: Zijn processen die in de achtergrond draaien.

Entries: Datainvoer onder een bepaalde vorm in een database of bestand.

Geëncrypteerd: Geëncrypteerde data is met een bepaalde sleutel gecodeerd, dit wil zeggen dat enkel een ander systeem dat ook over deze sleutel beschikt de data kan lezen.

Keytab: Dit bestand bevat de hostsleutel van de server waardoor de client en de KDC elkaars identiteit kunnen bevestigen.

Mounten: Toevoegen van een bestandssyteem aan het systeem.

Principal: Een gebruiker of service in Kerberos.

Quorum-bug: Een bepaalde fout in het openAFS-systeem die pas vanaf versie 1.2.11 opgelost is, al de versies voor 1.2.10 werken dus niet correct.

Real-time: Betekent zonder vertraging. Een handeling die real-time gebeurt, wordt op dit ogenblik uitgevoerd.

Replication: Het kopiëren van volumes met als doel een beveiliging tegen gegevensverlies bij systeemcrashes.

Rootvolume: Wordt hier gebruikt om aan te duiden dat dit het eerste volume is dat bovenaan de boom van volumes staat. We kunnen enkel volumes onder dit volume creëren en niet erboven.

SCM: Server Control Machine is de machine die de controle van alle database- en bestandstransacties uitvoert.

ServerID: Het identificatienummer van de server en is uniek voor iedere server.

Server-outage : Het uitvallen van een server door een netwerkverbreking of door een systeemcrash.

Stash bestand : Een bestand dat de KDC gebruikt om zich te authentificeren tegenover de database benodigdheden als kadmin, kadmind, krb5kds, …

Token: Token heeft verschillende betekenissen maar wordt hier gebruikt als paswoord.

Unmounten: Verwijderen van een bestandssysteem van het systeem.

Uitcommentariëren: Het plaatsen van bepaalde commando's of tekst in commentaar zodat wanneer het script uitgevoerd wordt, deze tekst niet als commando aanschouwd wordt.

Venus: De cachemanager in CODA

Bijlage 2: Commando's in CODA

Hier vindt u een lijst van de meest gebruikte commando's in CODA en hun betekenis.

cfs checkservers

Met dit commando kan men nagaan of alle servers bereikbaar zijn. Wanneer een server niet verbonden is of gecrasht is zullen we de foutmelding "server is unreachable" te zien krijgen.

cfs mkmount *Bestandsnaam Volumenaam*

Wordt gebruikt voor het mounten van een volume in een bepaalde directory. In tegenstelling tot het mounten in Linux, gebeurt het mounten in CODA éénmalig! Dit commando kan alleen uitgevoerd worden vanop een CODAclient door een gebruiker die het recht heeft om te schrijven in de directory waaronder dit volume gemount zal worden.

cfs checkpointml

Dit is een handig commando om in disconnected mode na te gaan welke directories aangepast zijn.

cfs la *Directory*

Commando om de acces control list van een bepaalde directory te tonen.

cfs sa *Directory*

Hiermee kunnen we een acces control list maken voor een bepaalde directory.
We kunnen nog 2 opties meegeven:

> **-negative**: om rechten af te nemen
> **-clear**: om de acces control list volledig leeg te maken voor er nieuwe rechten in te zetten.

clog

Commando om zich te authentificeren tegenover CODA servers.

cpasswd

Met dit commando kan u uw paswoord voor de authentificatie in CODA wijzigen.

createvol_rep *Uw-volume VSG /dataruimte*

Dit commando wordt gebruikt bij het creëren van een nieuw volume in de door u opgegeven dataruimte. Met de optie VSG wordt de Volume Storage Group meegegeven. Op de plaats van "uw-volume" zet u de naam voor het volume.

ctokens

Bij dit commando zal de lijst van tokens die uw client kent getoond worden. Wanneer de client de CODA-realm dus niet succesvol heeft kunnen mounten zullen er geen tokens getoond worden.

filcon isolate -s *Servernaam*

Instellen van een netwerkfilter om zo een door u opgegeven server van de CODA-cell af te scheiden.

filcon clear *Servernaam*

Verwijderen van een server uit de netwerkfilter.

hoard

Met dit commando komt u in een soort van interactieve mode terecht waarin u dan bestanden prioriteiten kan meegeven voor het up-to-date houden van deze bestanden.

hoard list

Hiermee kunnen we de lijst tonen van de gehoarde bestanden en hun prioriteiten.

pdbtool

Ook met dit commando komt u in een interactieve mode terecht waarmee u de protectiondatabase kan gaan aanpassen. Nu volgen enkele administratieve taken die uitgevoerd kunnen worden in deze mode.

- **nui *gebruiker gebruikersID***

  -- Maak een nieuwe gebruiker aan met het gekozen ID

- **ng *groep eigenaarsID***

  -- Maak een nieuwe groep aan met de gespecifieerde eigenaar

- **ci *gebruiker/groep nieuweID***

  -- Verander het ID van een bestaande groep of gebruiker

- **ag *groepID gebruiker/groepID***

  -- voeg een gebruiker of groep toe aan een bestaande groep

- **n *gebruiker/groep***

  -- Geef alle informatie van de gebruiker of groep weer

repair

Tool voor het herstellen van inconsistente bestanden en directories.

spy

Geeft alle CODA-bestanden weer in een lijst. Deze lijst kan gebruikt worden voor het maken van een hoard-bestand.

tail –f *Bestand*

Hiermee kunnen we een bepaald bestand (meestal logbestanden) onder toezicht houden. Wanneer we voor dit commando nog het commando xterm gebruiken wordt dit in een mooi venster getoond.

venus-setup *Met,komma,gescheiden,lijst,van,servers Cachegrootte_in_kb*

Gebruikt men voor de initialisatie van Venus (de lokale cachemanager). Hierdoor worden bepaalde configuratiebestanden aangepast. Achteraf kan verbinding worden gemaakt met de opgegeven server door het command venus &.

venus &

Commando om deze client te verbinden met de opgegeven server of servers bij de initialisatie. Wanneer we venus voor de eerste keer starten moeten we ook nog de optie –init meegeven.

vice-setup

Hierdoor wordt het script uitgevoerd waarmee de configuratie van de CODAserver gebeurd. Volgende scripts worden achtereenvolgens uitgevoerd:

vice-setup-scm
vice-setup-user
vice-setup-rvm
vice-setup-srvdir

volutil clone *VolumeID*

Dit commando creëert een read-only kopie van het door u opgegeven volume. Nadien kan men dan met een “volutil dump” dit volume in een bestand zetten.

volutil dump *Volumeld Bestandsnaam*

Zoals reeds gezegd wordt hiermee het gekozen volume in een bestand gedumpt.

volutil makevrdb

Met dit commando wordt er automatisch een nieuwe volume replication database aangemaakt.

volutil purge *VolumeID VolumeNaam*

Dient om het opgegeven volume te verwijderen.

volutil restore *Bestand Partitie[Volumenaam [VolumeID]]*

Wordt gebruikt om van een gedumpt volume in een bestand opnieuw een reëel volume te maken.

volutil shutdown

Hiermee wordt de codaserver gestopt!

vutil –shutdown

Wordt gebruikt om venus te stoppen en dus de verbinding met de server te verbreken.

/etc/rc.d/init.d/auth2.init start

Starten van het serverproces dat de nodige authentificatie uitvoert.

/etc/rc.d/init.d/update.init start

Starten van het serverproces dat zorgt voor het up-to-date houden van de databases.

/etc/rc.d/init.d/codasrv.init start

Starten van het codaserverproces.

Bijlage 3: Commando's in OpenAFS

Hier vindt u een lijst van de meest gebruikte commando's in OpenAFS en hun betekenis.

afsd

Initialisatie van de cachemanager en start de juiste processen.

afsmonitor

Controleert Fileservers en cachemanagers.

bos addhost -server <*machine name*> -host <*host name*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Voegt een databaseserver toe aan /usr/afs/etc/CellServDB.

bos addkey -server <*machine name*> [-key <*key*>] -kvno <*key version number*>
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Voegt een nieuwe geëncrypteerde sleutel toe aan /usr/afs/etc/KeyFile.

bos adduser -server <*machine name*> -user <*user names*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Voeg een gebruiker toe aan /usr/afs/etc/UserList

bos create -server <*machine name*> -instance <*server process name*>
-type <*server type*> -cmd <*command lines*>+
[-notifier <*Notifier program*>] [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Starten van een nieuw proces en gelijktijdig toevoegen aan /usr/afs/local/BosConfig

bos delete -server <*machine name*> -instance <*server process name*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Verwijderen van een serverproces uit /usr/afs/local/BosConfig

bos exec -server <*machine name*> -cmd <*command to execute*>
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Uitvoeren van een commando op een machine vanop een andere machine.

bos getdate -server <*machine name*> -file <*files to check*>+
[-dir <*destination dir*>] [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Tonen van de "time-stamp" van een uitvoerbaar AFS bestand

bos getlog -server <*machine name*> -file <*log file to examine*>
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Tonen van een logfile van een proces.

bos getrestart -server <*machine name*> [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Laat de automatische herstart instelling zien van de serverprocessen.

bos install -server <*machine name*> -file <*files to install*>+
[-dir <*destination dir*>] [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Installeren van een nieuwe versie van een uitvoerbaar bestand.

bos listhosts -server <*machine name*> [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Toont de inhoud van /usr/afs/etc/CellServDB

bos listkeys -server <*machine name*> [-showkey] [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Toont de geëncrypteerde sleutels uit /usr/afs/etc/KeyFile.

bos listusers -server <*machine name*> [-cell <*cell name*>] [-noauth]
[-localauth] [-help]

Geeft een lijst van de gebruikers uit /usr/afs/etc/UserList.

bos prune -server <*machine name*> [-bak] [-old] [-core] [-all]
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Verwijdert verouderde versies van bestanden uit /usr/afs/bin en /usr/afs/logs.

bos removehost -server <*machine name*> -host <*host name*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Verwijdert een databaseserver uit /usr/afs/etc/CellServDB.

bos removekey -server <*machine name*> -kvno <*key version number*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Verwijdert een geëncrypteerde sleutel uit /usr/afs/etc/KeyFile.

bos removeuser -server <*machine name*> -user <*user names*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Verwijdert een gebruiker uit /usr/afs/etc/UserList.

bos restart -server <*machine name*> [-instance <*instances*>+] [-bosserver]
[-all] [-cell <*cell name*>] [-noauth] [-localauth] [-help]

Herstarten van een serverproces.

bos salvage -server <*machine name*> [-partition <*salvage partition*>]
[-volume <*salvage volume number or volume name*>]
[-file <*salvage log output file*>] [-all] [-showlog]
[-parallel <*# of max parallel partition salvaging*>]
[-tmpdir <*directory to place tmp files*>]
[-orphans <ignore | remove | attach>]
[-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Herstellen van de consistentie van een bestandssysteem of volume.

bos setcellname -server <*machine name*> -name <*cell name*>
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Instellen van cellnaam in /usr/afs/etc/ThisCell en /usr/afs/etc/CellServDB

bos setrestart -server <*machine name*> -time <*time to restart server*>
[-general] [-newbinary] [-cell <*cell name*>]
[-noauth] [-localauth] [-help]

Instellen van tijd en datum voor het herstarten van BOS Server processen.

bos shutdown -server <*machine name*> [-instance <*instances*>+] [-wait]
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Stoppen van een proces zonder de status vlag aan te passen in /usr/afs/local/BosConfig.

bos start -server <*machine name*> -instance <*server process name*>+
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Starten van een proces nadat de status vlag aangepast is in /usr/afs/local/BosConfig.

bos startup -server <*machine name*> [-instance <*instances*>+]
[-cell <*cell name*>] [-noauth] [-localauth] [-help]

Starten van een proces zonder de status vlag aan te passen in /usr/afs/local/BosConfig.

bos status -server <*machine name*> [-instance <*server process name*>+]
[-long] [-cell <*cell name*>] [-noauth] [-localauth] [-help]

Toont de status van de serverprocessen.

bos stop -server <*machine name*> -instance <*server process name*>+
[-wait] [-cell <*cell name*>] [-noauth] [-localauth] [-help]

Stoppen van een proces nadat de status vlag aangepast is in /usr/afs/local/BosConfig.

bos uninstall -server <*machine name*> -file <*files to uninstall*>+
[-dir <*destination dir*>] [-cell <*cell name*>] [-noauth] [-localauth] [-help]

Een uitvoerbaar bestand terugdraaien naar een vorige versie.

bosserver [-noauth] [-log] [-enable_peer_stats] [-enable_process_stats] [-help]

Initialiseren van de BOS Server.

buserver [-database <*database directory*>] [-cellservdb <*cell configuration directory*>] [-resetdb] [-noauth] [-smallht] [-servers <*list of ubik database servers*>+] [-enable_peer_stats] [-enable_process_stats] [-help]

Initialiseren van de backupserver.

fileserver [-d <*debug level*>] [-p <*number of processes*>] [-spare <*number of spare blocks*>] [-pctspare <*percentage spare*>] [-b <*buffers*>] [-l <*large vnodes*>] [-s <*small nodes*>] [-vc <*volume cachesize*>] [-w <*call back wait interval*>] [-cb <*number of call backs*>] [-banner (print banner every 10 minutes)] [-novbc (whole volume cbs disabled)] [-implicit <*admin mode bits: rlidwka*>] [-hr <*number of hours between refreshing the host cps*>] [-busyat <*redirect clients when queue > n*>] [-rxpck <*number of rx extra packets*>] [-rxdbg (enable rx debugging)] [-rxdbge (enable rxevent debugging)] [-m <*min percentage spare in partition*>] [-lock (keep fileserver from swapping)] [-L (large server conf)] [-S (small server conf)] [-k <*stack size*>] [-realm <*Kerberos realm name*>] [-udpsize <*size of socket buffer in bytes*>] [-enable_peer_stats] [-enable_process_stats] [-help]

Initialiseren van het fileserver deel van het fs proces.

fs checkservers [-cell <*cell to check*>] [-all] [-fast] [-interval <*seconds between probes*>] [-help]

Toont de status van de servers.

fs checkvolumes [-help]

Dwingt de cachemanager haar volume-gerelateerde informatie te updaten.

fs cleanacl [-path <*dir/file path*>+] [-help]

Verwijdert verouderde entries van de ACL.

fs copyacl -fromdir <*source directory (or DFS file)*>
-todir <*destination directory (or DFS file)*>+
[-clear] [-id] [-if] [-help]

Kopiëren van een ACL van één directory naar een andere directory.

fs diskfree [-path <*dir/file path*>+] [-help]

Toont informatie over de partitie die het bestand of de directory bezit.

fs examine [-path <*dir/file path*>+] [-help]

Toont informatie over het volume die het bestand of de directory bezit.

fs getcacheparms [-help]

Toont de huidige grootte van de cache en de grootte die in gebruik is.

fs getcellstatus -cell <*cell name*>+ [-help]

Rapporteert of de machine setuid programma's kan draaien vanuit een specifieke cell.

fs getclientaddrs [-help]

Toont de clientinterfaces.

fs getserverprefs [-file <*output to named file*>]
[-numeric] [-vlservers] [-help]

Toont de instellingen van de cachemanager voor fileservers of Vlservers.

fs listacl [-path <*dir/file path*>+] [-id] [-if] [-help]

Toont ACL's.

fs listcells [-numeric] [-help]

Toont de databaseservers in iedere cell die de cachemanager kent.

fs listquota [-path <*dir/file path*>+] [-help]

Toont quota informatie over het volume dat de opgegeven directory of dit bestand bezit.

fs lsmount -dir <*directory*>+ [-help]

Geeft het mountpoint weer van de opgegeven directory.

fs mkmount -dir <*directory*> -vol <*volume name*> [-cell <*cell name*>]
[-rw] [-fast] [-help]

Creëert een mountpoint voor een volume.

fs quota [-path <*dir/file path*>+] [-help]

Toont het percentage van quota in gebruik door het volume van opgegeven directory of bestand.

fs rmmount -dir <*directory*>+ [-help]

Verwijdert een mountpoint.

fs setacl -dir <*directory*>+ -acl <*access list entries*>+
[-clear] [-negative] [-id] [-if] [-help]

Instellen van een ACL voor een directory.

fs setcachesize [-blocks <*size in 1K byte blocks (0 => reset)*>]
[-reset] [-help]

Instellen van de grootte van de cache.

fs setcell -cell <*cell name*>+ [-suid] [-nosuid] [-help]

Toelaten of verbieden van het uitvoeren van setuid programma's vanuit een specifieke cell.

fs setclientaddrs [-address <*client network interfaces*>+] [-help]

Instellen van de clientinterfaces.

fs setquota [-path <*dir/file path*>] -max <*max quota in kbytes*> [-help]

Instellen van de maximum quota voor het volume dat opgegeven directory of bestand bezit.

fs setserverprefs [-servers <*fileserver names and ranks*>+]
[-vlservers <*VL server names and ranks*>+]
[-file <*input from named file*>] [-stdin] [-help]

Instellen van de cachemanager voor fileservers of Vlservers.

fs whereis [-path <*dir/file path*>+] [-help]

Rapporteert de naam van iedere fileserver die opgegeven bestand of directory bezit.

fs whichcell [-path <*dir/file path*>+] [-help]

Toont de naam van de cell waartoe opgegeven bestand of directory behoort.

fs wscell [-help]

Toont de naam van de cell waartoe de machine behoort.

kadb_check -database <*kadb_file*> [-uheader] [-kheader] [-entries]
[-verbose] [-rebuild <*out_file*>] [-help]

Controleert de integriteit van de authentificatiedatabase.

kas create -name <*name of user*> [-initial_password <*initial password*>]
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Creëren van een entry in de authentificatiedatabase.

kas delete -name <*name of user*>
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Verwijderen van een entry uit de authentificatiedatabase.

kas examine -name <*name of user*> [-showkey]
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Toont de informatie van een authentificatiedatabase entry.

kas forgetticket [-all] [-help]

Met geen enkel ticket van de uitvoerder meer rekening houden.

kas interactive [-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Activeren van de interactieve mode.

kas list [-long] [-showadmin] [-showkey]
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+]
[-noauth] [-help]

Tonen van alle entries in de authentificatiedatabase.

kas listtickets [-name <*name of server*>] [-long] [-help]

Toont alle tickets van de uitvoerder.

quit [-help]

Verlaat de interactieve mode.

kas setfields -name <*name of user*>
[-flags <*hex flag value or flag name expression*>]
[-expiration <*date of account expiration*>]
[-lifetime <*maximum ticket lifetime*>]
[-pwexpires <*number days password is valid ([0..254])*>]
[-reuse <*permit password reuse (yes/no)*>]
[-attempts<*maximum successive failed login tries ([0..254])*>]
[-locktime <*failure penalty [hh:mm or minutes]*>]
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Instellen van opties in een authentificatiedatabase entry.

kas setpassword -name <*name of user*> [-new_password <*new password*>]
[-kvno <*key version number*>]
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Veranderen van het paswoord in een authentificatiedatabase entry.

kas statistics [-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Toont statistieken van een authentificatieproces.

kas unlock -name <*authentication ID*>
[-admin_username <*admin principal to use for authentication*>]
[-password_for_admin <*admin password*>] [-cell <*cell name*>]
[-servers <*explicit list of authentication servers*>+] [-noauth] [-help]

Deblokkeren van een geblokkeerde gebruiker.

kaserver [-noAuth] [-fastKeys] [-database <*dbpath*>] [-localfiles <*lclpath*>]
[-minhours <*n*>] [-servers <*serverlist*>] [-enable_peer_stats]
[-enable_process_stats] [-help]

Initialiseert de authentificatieserver.

klog [-x] [-principal <*user name*>] [-password <*user's password*>]
[-cell <*cell name*>] [-servers <*explicit list of servers*>+] [-pipe] [-silent]
[-lifetime <*ticket lifetime in hh[:mm[:ss]]*>] [-setpag] [-tmp] [-help]

Authentificatie tegenover de authentificatieserver.

kpasswd [-x] [-principal <*user name*>] [-password <*user's password*>]
[-newpassword <*user's new password*>] [-cell <*cell name*>]
[-servers <*explicit list of servers*>+] [-pipe] [-help]

Aanpassen van het paswoord van de uitvoerder in de authentificatiedatabase.

prdb_check -database <*ptdb_file*> [-uheader] [-pheader] [-entries]
[-verbose] [-help]

Controleert de integriteit van de protectiondatabase.

pts adduser -user <*user name*>+ -group <*group name*>+
[-cell <*cell name*>] [-noauth] [-force] [-help]

Voeg een gebruiker of machine toe aan de protectiondatabase groep.

pts chown -name <*group name*> -owner <*new owner*>
[-cell <*cell name*>] [-noauth] [-force] [-help]

Veranderen van de eigenaar van een entry in de protectiondatabase.

pts creategroup -name <*group name*>+ [-owner <*owner of the group*>]
[-id <*id (negated) for the group*>+] [-cell <*cell name*>]
[-noauth] [-force] [-help]

Creëren van een lege entry in een protectiondatabase groep.

pts createuser -name <*user name*>+ [-id <*user id*>+] [-cell <*cell name*>]
[-noauth] [-force] [-help]

Creëren van een gebruiker of machine entry in de protectiondatabase.

pts delete -nameorid <*user or group name or id*>+ [-cell <*cell name*>]
[-noauth] [-force] [-help]

Verwijderen van een protectiondatabase entry.

pts examine -nameorid <*user or group name or id*>+ [-cell <*cell name*>]
[-noauth] [-force] [-help]

Toont een protectiondatabase entry.

pts listentries [-users] [-groups] [-cell <*cell name*>][-noauth] [-force] [-help]

Toont alle gebruikers of groep entries in de protectiondatabase.

pts listowned -nameorid <*user or group name or id*>+ [-cell <*cell name*>]
[-noauth] [-force] [-help]

Toont de groepen in de protectiondatabase die door een bepaalde gebruiker of groep beheerd worden.

pts membership -nameorid <*user or group name or id*>+ [-cell <*cell name*>]
[-noauth] [-force] [-help]

Toont de lijst van groepen waar een bepaalde gebruiker of groep lid van is.

pts removeuser -user <*user name*>+ -group <*group name*>+
[-cell <*cell name*>] [-noauth] [-force] [-help]

Verwijdert een gebruiker uit de protectiondatabase groep.

pts rename -oldname <*old name*> -newname <*new name*>
[-cell <*cell name*>] [-noauth] [-force] [-help]

Verandert de naam van een entry uit de protectiondatabase.

pts setfields -nameorid <*user or group name or id*>+
[-access <*set privacy flags*>]
[-groupquota <*set limit on group creation*>]
[-cell <*cell name*>] [-noauth] [-force] [-help]

Instellen van privacy vlaggen of de groepquota voor een entry uit de protectiondatabase.

ptserver [-database <*db path*>] [-p <*number of processes*>] [-rebuildDB]
[-enable_peer_stats] [-enable_process_stats] [-help]

Initialiseren van de protectionserver.

rxdebug -servers <*server machine*> [-port <*IP port*>] [-nodally]
[-allconnections] [-rxstats] [-onlyserver] [-onlyclient]
[-onlyport <*show only <port>*>] [-onlyhost <*show only <host>*>]
[-onlyauth <*show only <auth level>*>] [-version] [-noconns] [-peers] [-help]

Debuggen van Rx activiteiten.

salvager [initcmd] [-partition <*Name of partition to salvage*>]
[-volumeid <*Volume Id to salvage*>] [-debug]
[-nowrite] [-inodes] [-force] [-oktozap]
[-rootinodes] [-salvagedirs] [-blockreads]
[-parallel <*# of max parallel partition salvaging*>]
[-tmpdir <*Name of dir to place tmp files*>]
[-showlog] [-showsuid] [-showmounts]
[-orphans <ignore | remove | attach>] [-help]

Initialiseren van het Salvager deel van het fs proces.

scout [initcmd] -server <*FileServer name(s) to monitor*>$^{+}$
[-basename <*base server name*>] [-frequency <*poll frequency, in seconds*>]
[-host] [-attention <*specify attention (highlighting) level*>$^{+}$]
[-debug <*turn debugging output on to the named file*>] [-help]

Controleren van het fileserverproces.

tokens [-help]

Toont de tokens van de uitvoerder.

udebug -servers <*server machine*> [-port <*IP port*>] [-long] [-help]

Rapporteren van de status van het databaseserverproces.

unlog [-cell <*cell name*>$^{+}$] [-help]

Geen rekening meer houden met de tokens van de uitvoerder.

upclient <*hostname*> [-crypt] [-clear] [-t <*retry time*>] [-verbose]* <*dir*>$^{+}$ [-help]

Initialiseren van het clientdeel van de update server.

upserver [<*directory*>$^{+}$] [-crypt <*directory*>$^{+}$] [-clear <*directory*>$^{+}$]
[-auth <*directory*>$^{+}$] [-help]

Initialiseren van het serverdeel van de update server.

uss add -user <*login name*> [-realname <*full name in quotes*>]
[-pass <*initial password*>]
[-pwexpires <*password expires in [0..254] days (0 => never)*>]
[-server <*FileServer for home volume*>]
[-partition <*FileServer's disk partition for home volume*>]
[-mount <*home directory mount point*>]
[-uid <*uid to assign the user*>]
[-template <*pathname of template file*>]
[-verbose] [-var <*auxiliary argument pairs (Num val)*>+]
[-cell <*cell name*>] [-admin <*administrator to authenticate*>]
[-dryrun] [-skipauth] [-overwrite] [-help]

Creëren van een gebruikersaccount.

uss bulk -file <*bulk input file*> [-template <*pathname of template file*>]
[-verbose] [-cell <*cell name*>] [-admin <*administrator to authenticate*>]
[-dryrun] [-skipauth] [-overwrite]
[-pwexpires <*password expires in [0..254] days (0 => never)*>]
[-pipe] [-help]

Uitvoeren van meerdere uss commando's die opgesomd staan in een bestand.

uss delete -user <*login name*> [-mountpoint <*mountpoint for user's volume*>]
[-savevolume] [-verbose] [-cell <*cell name*>]
[-admin <*administrator to authenticate*>] [-dryrun] [-skipauth] [-help]

Verwijderen van een gebruikersaccount.

vldb_check -database <*vldb_file*> [-uheader] [-vheader] [-servers]
[-entries] [-verbose] [-help]

Controleert de integriteit van de VLDB.

vlserver [-p <*lwp processes*>] [-nojumbo] [-enable_peer_stats]
[-enable_process_stats] [-help]

Initialiseren van de volumelocationserver.

volinfo [-online] [-vnode] [-date] [-inode] [-itime]
[-part <*AFS partition name (default current partition)*>+]
[-volumeid <*Volume id*>+] [-header] [-sizeOnly] [-fixheader]
[-saveinodes] [-orphaned] [-help]

Toont gedetailleerde informatie over één of meerdere volumes en de partitie waarop deze zich bevindt.

volserver [-log] [-p <*number of processes*>]
[-udpsize <*size of socket buffer in bytes*>]
[-enable_peer_stats] [-enable_process_stats] [-help]

Initialiseren van het volumeserver deel van het fs poces.

vos addsite -server <*machine name for new site*>
-partition <*partition name for new site*>
-id <*volume name or ID*> [-cell <*cell name*>]
[-noauth] [-localauth] [-verbose] [-help]

Toevoegen van een read-only site definitie aan een VLDB entry van een volume.

vos backup -id <*volume name or ID*> [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Creëren van een backupvolume.

vos backupsys [-prefix <*common prefix on volume(s)*>+]
[-server <*machine name*>] [-partition <*partition name*>]
[-exclude] [-xprefix <*negative prefix on volume(s)*>+]
[-dryrun] [-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Creëren van een backupvolume voor meerdere volumes.

vos changeaddr -oldaddr <*original IP address*> [-newaddr <*new IP address*>]
[-remove] [-cell <*cell name*>] [-noauth] [-localauth]
[-verbose] [-help]

Aanpassen of verwijderen van een fileserver entry in de VLDB.

vos create -server <*machine name*> -partition <*partition name*>
-name <*volume name*> [-maxquota <*initial quota (KB)*>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Creëren van een read/write volume en de VLDB entry.

vos delentry [-id <*volume name or ID*>+]
[-prefix <*prefix of volume whose VLDB entry is to be deleted*>]
[-server <*machine name*>] [-partition <*partition name*>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Verwijderen van een volume entry uit de VLDB.

vos dump -id <*volume name or ID*> [-time <*dump from time*>]
[-file <*dump file*>] [-server <*server*>] [-partition <*partition*>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Converteren van een volume in een ASCII formaat en dit in een bestand dumpen.

vos examine -id <*volume name or ID*> [-extended] [-cell <*cell name*>]
[-noauth] [-localauth] [-verbose] [-help]

Toont informatie over het volume en over de entry in de VLDB.

vos listaddrs [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Toont alle VLDBserver entries.

vos listpart -server <*machine name*> [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Toont alle AFS partities op een fileserver.

vos listvldb [-name <*volume name or ID*>] [-server <*machine name*>]
[-partition <*partition name*>] [-locked] [-quiet] [-nosort]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Toont een entry uit de volume's VLDB.

vos listvol -server <*machine name*> [-partition <*partition name*>]
[-fast] [-long] [-quiet] [-extended] [-cell <*cell name*>]
[-noauth] [-localauth] [-verbose] [-help]

Toont informatie van een volume.

vos move -id <*volume name or ID*> -fromserver <*machine name on source*>
-frompartition <*partition name on source*>
-toserver <*machine name on destination*>
-topartition <*partition name on destination*>
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Verplaats een read/write volume naar een andere site.

vos partinfo -server <*machine name*> [-partition <*partition name*>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Toon de beschikbare en totale ruimte op een partitie.

vos release -id <*volume name or ID*> [-f] [-cell <*cell name*>]
[-noauth] [-localauth] [-verbose] [-help]

Update de inhoud van read-only volumes.

vos remove [-server <*machine name*>] [-partition <*partition name*>]
-id <*volume name or ID*> [-cell <*cell name*>]
[-noauth] [-localauth] [-verbose] [-help]

Verwijder een volume van een site.

vos remsite -server <*machine name*> -partition <*partition name*>
-id <*volume name or ID*> [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Verwijderen van een read-only site definitie uit een VLDB entry.

vos rename -oldname <*old volume name*> -newname <*new volume name*>
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Wijzigen van de naam van een volume.

vos restore -server <*machine name*> -partition <*partition name*>
-name <*name of volume to be restored*> [-file <*dump file*>]
[-id <*volume ID*>] [-overwrite <abort | full | incremental>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Converteren van een ASCII bestand naar een volume format en dit terug toevoegen aan het bestanssysteem.

vos status -server <*machine name*> [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Rapporteren van de status van een volumeserver.

vos syncserv -server <*machine name*> [-partition <*partition name*>]
[-cell <*cell name*>] [-noauth] [-localauth] [-verbose] [-help]

Synchroniseren van een server voor een specifieke site.

vos syncvldb [-server <*machine name*>] [-partition <*partition name*>]
[-volume <*volume name or ID*>] [-cell <*cell name*>] [-noauth]
[-localauth] [-verbose] [-help]

Synchroniseren van VLDB entries voor een specifieke site.

vos zap -server <*machine name*> -partition <*partition name*> -id <*volume ID*>

Verwijdert een volume zonder dit aan te passen in de VLDB.