Administração de Sistemas (ASIST)

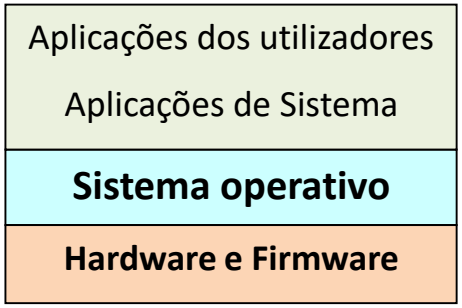
Aula Teórico Prática 01

Sistemas operativos servidores

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Sistema operativo

O sistema operativo (**Operating System - OS**) é um conjunto de componentes de *software* que permite estabelecer uma plataforma estável e homogénea sobre a qual as aplicações de sistema e dos utilizadores podem funcionar.



O hardware sobre o qual o sistema operativo trabalha pode ser muito diverso. O sistema operativo constitui uma camada de abstração, graças a ele, as aplicações não necessitam de ser desenvolvidas para tipos de hardware específicos. **As aplicações usam indiretamente o hardware através do sistema operativo.**

Algumas das funções mais importantes de um sistema operativo são a gestão de **processos** e ***threads***, a gestão de **memória central (RAM)** e gestão de **dispositivos externos.**

Os componentes do sistema operativo que asseguram estas três funções básicas fazem parte do que normalmente se designada o núcleo do sistema operativo ou ***kernel***.

Sistemas operativos servidores

Embora qualquer sistema operativo possa ser usado para implementar um servidor, existem versões de sistemas operativos especificamente desenhadas para esse efeito. Tipicamente batizadas com a designação ***server***.

Na maioria dos casos, as versões ***server*** são baseadas nas versões tradicionais (***desktop***), frequentemente o ***kernel*** é o mesmo.

Na atualidade, destacam-se três famílias de sistemas operativos servidores:

Microsoft Windows Server – versão servidora dos sistemas operativos da Microsoft, o primeiro foi o Windows NT Server, à data, a mais recente é o Windows 2019 Server.

Versões ***open source*** baseadas em Unix – destacam-se entre outras as família **LINUX** e **BSD** (NetBSD, FreeBSD e OpenBSD).

OS X – originalmente designado MAC OS Server, atualmente existe apenas uma versão que é simultaneamente server e desktop.

Sistemas de ficheiros (*File System*)

De entre os vários tipos de dispositivos externos que um sistema operativo gere destacam-se os dispositivos de armazenamento (***storage***). Ao contrário da memória central típica (**RAM**) são normalmente não voláteis, ou seja os dados armazenados persistem quando o sistema é desligado.

Uma caraterização básica de um dispositivo de armazenamento é a sua capacidade, normalmente expressa em bytes, cada byte é armazenado numa posição endereçável do dispositivo. Alguns dispositivos pode ser de leitura apenas, como por exemplo os CDs não regraváveis.

O sistema operativo acede a bytes individuais no dispositivo de armazenamento, mas esse tipo de acesso não é adequado para as aplicações.

Para permitir uma utilização mais expedita pelas aplicações, o sistema operativo estabelece um **sistema de ficheiros** sobre o dispositivo de armazenamento. Esta operação é designada **formatação** e consiste na organização do dispositivo de armazenamento em pastas (diretórios) e ficheiros.

Partições e sistemas de ficheiros

A gestão dos sistemas de ficheiros é também uma função base de qualquer sistema operativo e encontra-se integrada no *kernel*.

Uma vez criado o sistema de ficheiros no dispositivo de armazenamento externo o sistema operativo pode proporcionar funções de utilização de ficheiros (*open/create*, *read*, *write*, *close*, …) muito mais adequadas para as aplicações e mais fáceis de gerir (permissões, concorrência, …) pelo *kernel*.

Muitos dispositivos de armazenamento externo são discos fixos (não amovíveis): **HDD** (*Hard Disk Drive*) ou **SSD** (*Solid State Drive*). Normalmente os discos fixos tem de ser **divididos em partições antes de poderem serem formatados**, mesmo que se pretenda apenas uma partição no disco.

Uma partição é uma zona contínua que começa e termina em posições bem definidas do disco e é independente de outras partições. Para todos os efeitos pode ser vista como um disco independente.

Um sistema de ficheiros criado numa partição é totalmente independente de sistemas de ficheiros de outras partições.

Tabela de partições (partitions table)

A tabela de partições define as posições de início e fim de cada partição, encontra-se guardada no início do próprio disco numa zona designada *Master Boot Record* (**MBR**).

A primeira coisa que um sistema operativo faz quando tem acesso a um disco é ler a tabela de partições. Igualmente, o primeiro passo na configuração de um novo disco vazio é criar uma nova tabela de partições.

Uma vez definida uma partição, ela pode ser formatada usando um determinado tipo de sistema de ficheiros, contudo, após ser formatada, a alteração da partição (posições de início e fim) deixa de ser um processo trivial.

Alterar uma partição formatada sem mais cuidados implica a perda total do sistema de ficheiros lá existente (dados).

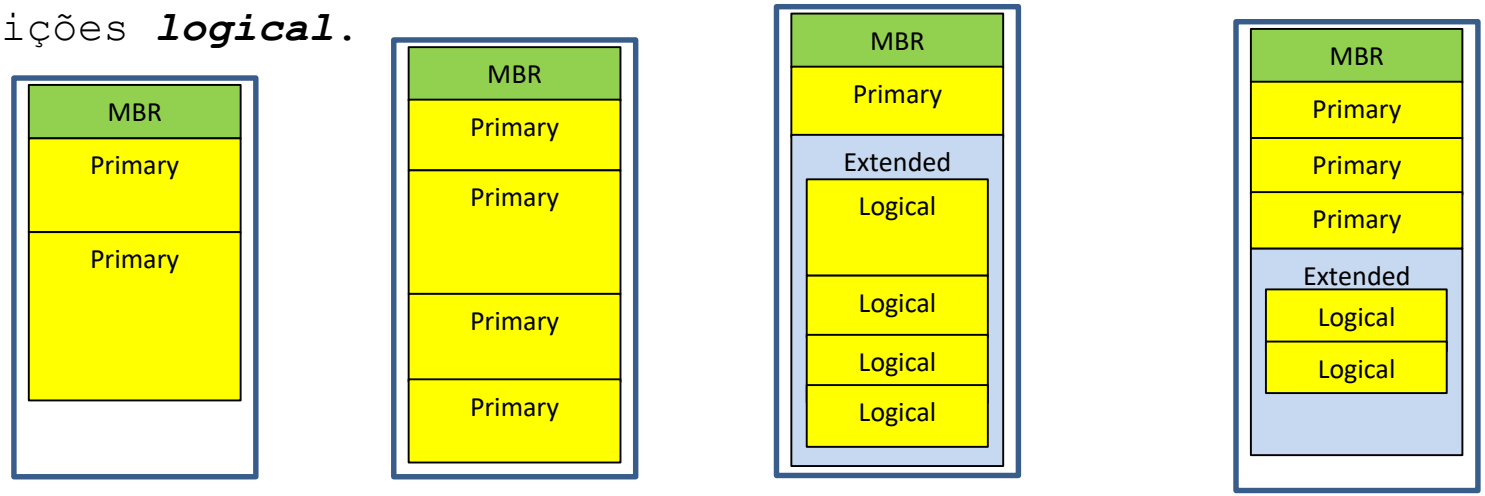
Alterar uma partição que contém dados (sistema de ficheiros) envolve a deslocação de todo o sistema de ficheiros. Normalmente é simples **expandir uma partição** (avançar a posição final), desde que o espaço seguinte no disco não esteja alocado a nenhuma outra partição.

MBR - Primary, Extended, and Logical Partitions

A tabela de partições do MBR foi herdada dos primórdios dos computadores pessoais e apresenta bastantes limitações. Atualmente existem algumas alternativas, mas tenta-se manter a compatibilidade com o formato MBR.

A tabela MBR é estática e tem quatro entradas que podem ou não estar definidas. Portanto permite um máximo de 4 partições. Destas, no máximo uma pode ser ***extended***, as restantes são ***primary***.

As partições ***extended*** existem com o objetivo de contornar o limite de 4 partições, uma partição ***extended*** pode conter um número ilimitado de partições ***logical***.



As partições ***logical*** podem ser usadas da mesma forma que as partições ***primary***, podem por exemplo ser a partição de arranque do sistema operativo – no pressuposto que exista uma partição *primary* que contenha o *boot loader*.

Identificação de discos e partições

Diferentes sistemas operativos utilizam formas diferentes de identificar os discos e partições, neste curso vamos centrar a nosso estudo nos sistemas Unix (Linux) e sistemas MS-Windows.

Nos sistemas Linux os discos atuais (interfaces SCSI e SATA) são identificados por **/dev/sda**, **/dev/sdb**, … de acordo com a ordem em que são detetados pelo *kernel*. As partições existentes em cada disco são identificadas através de sufixos numéricos sequenciais começados por um. Por exemplo **/dev/sdb5** identifica a quinta partição do segundo disco.

Nos sistemas Windows os discos são identificados por valores numéricos começando pelo valor zero: **"Disk 0", "Disk 1" ,** …

Tal como no Unix, as partições em cada disco são identificadas por valores numéricos sendo atribuído à primeira partição o valor 1. Por exemplo a designação "***Disk 1 partition 3***" refere-se à terceira partição do segundo disco. Nos sistemas Windows utiliza-se a designação volume para referir uma partição. Para que uma partição (volume) possa ser usada pelas aplicações é necessário ainda **atribuir-lhe uma letra de drive única.**

Gestão da tabela de partições - Linux

Em Linux o comando tradicional para gerir a tabela de partições é o ***fdisk***. Com este comando é possível de forma interativa criar e alterar partições:

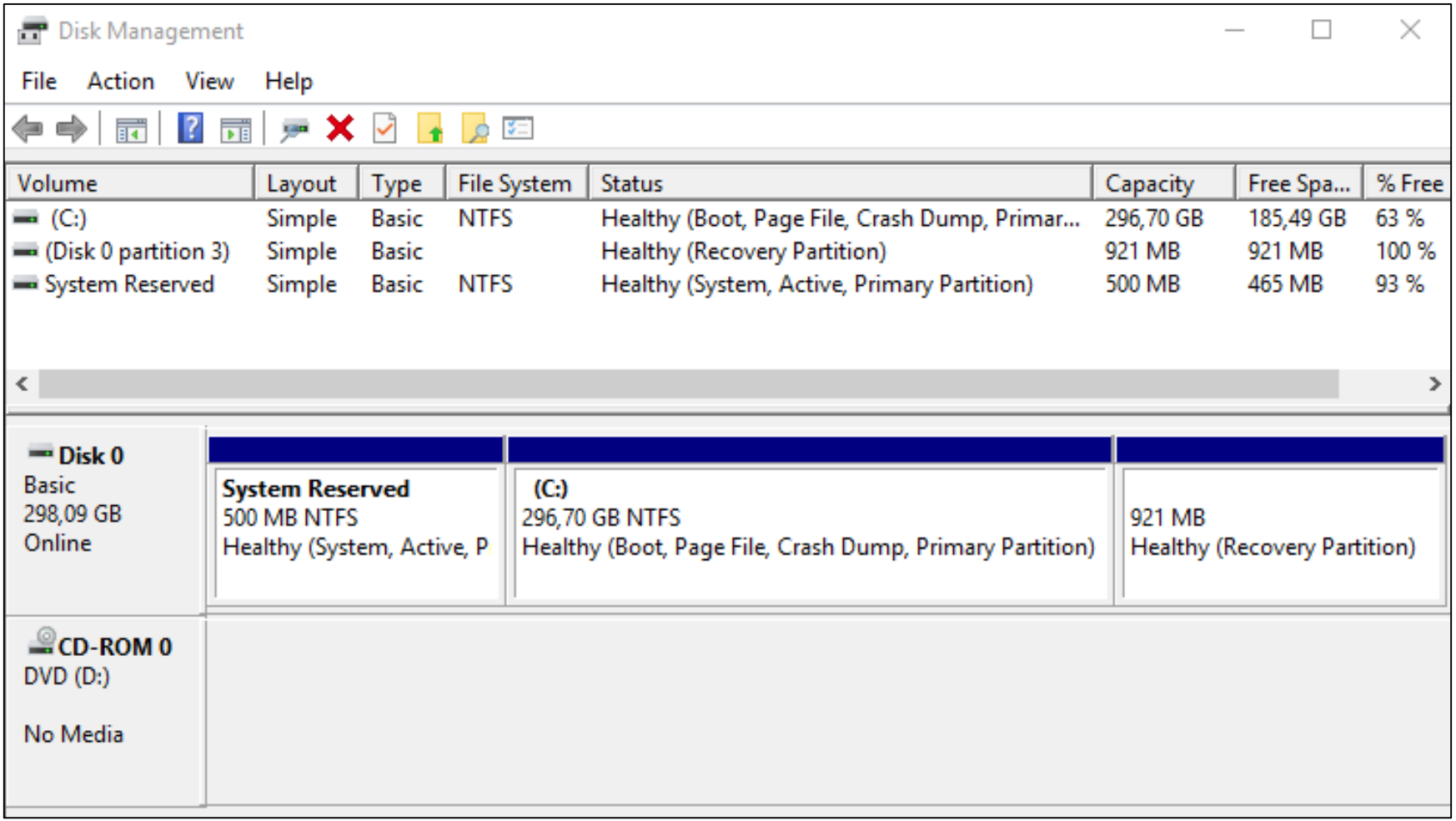
```
HOST1# fdisk /dev/sda
Command (m for help): p
Disk /dev/sda: 73.4 GB, 73407868928 bytes
255 heads, 63 sectors/track, 8924 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device    Boot      Start     End       Blocks    Id        System
/dev/sda1 *         1         101       811251    83        Linux
/dev/sda2           102       8924      70870747+ 5         Extended
/dev/sda5           102       202       811251    82        Linux swap
/dev/sda6           203       711       4088511   83        Linux
/dev/sda7           712       4757      32499463+ 83        Linux
/dev/sda8           4758      8797      32451268+ 83        Linux
/dev/sda9           8798      8924      1020096   83        Linux
```

No exemplo acima apresenta-se (argumento **-p** do ***fdisk***) a tabela de partições do disco **/dev/sda.** Como se pode observar uma tabela de partições contém diversa informação, além das posições de cada partição também o tipo de sistema de ficheiros de cada uma e a indicação de qual é a partição de arranque (*boot*), no caso a partição primária **/dev/sda1.**

Podemos também observar uma partição do tipo *extended* (**/dev/sda2**) que contém as partições lógicas **/dev/sda5** até **/dev/sda9.**

Gestão da tabela de partições - Windows

Em Windows a aplicação ***Disk Management*** permite realizar o mesmo tipo de operações e adicionalmente atribuir a um volume formatado uma letra de drive:



GUID Partition Table (GPT)

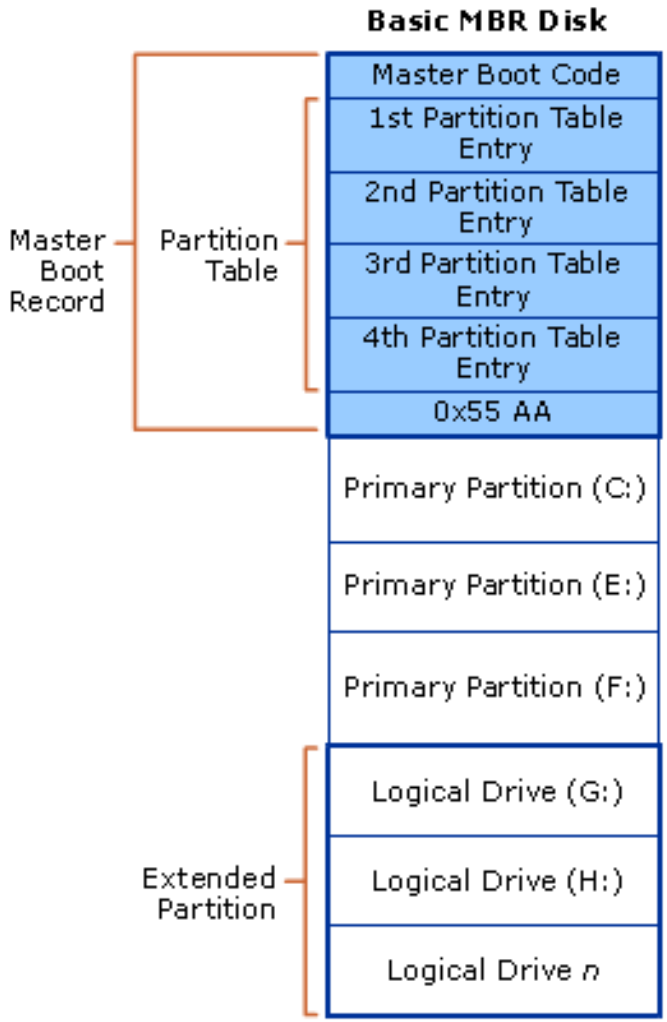
Embora o limite de quatro partições por disco do MBR seja ultrapassado através das partições lógicas, existem outros problemas.

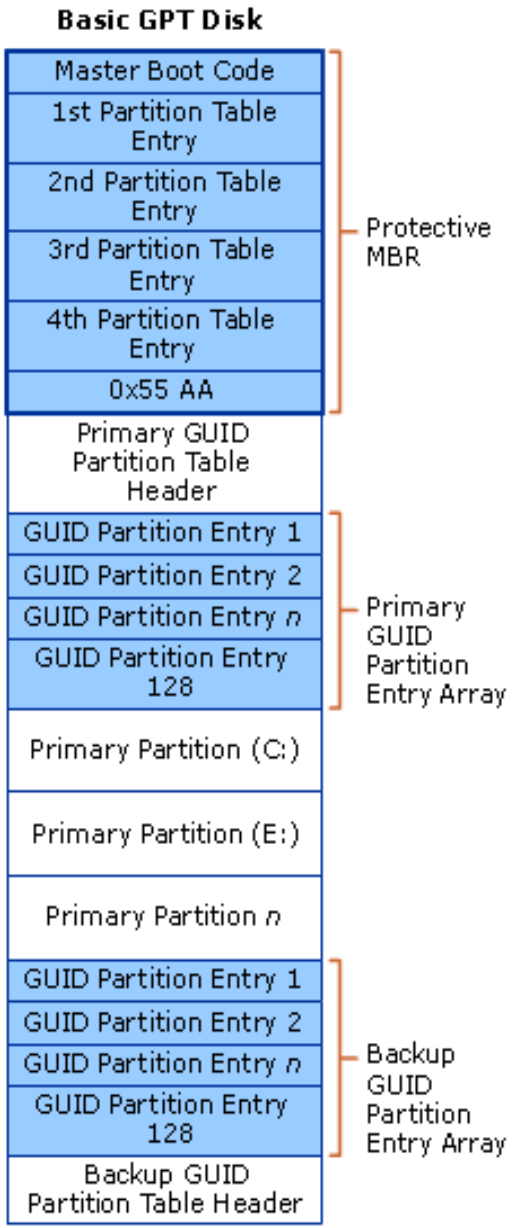
Com o MBR, devido à forma dos endereços usados na referenciação das posições no disco, não é possível definir partições para além da posição correspondente a 2 terabytes. Se o disco tem dimensão superior, ficara desaproveitado. Outro problema é que só existe um exemplar do MBR, se for corrompido perde-se a referência às partições existentes.

A GUID (*globally unique identifiers*) *Partition Table* (GPT) resolve estes problemas. Suporta até 128 partições primárias, permite a validação da integridade da tabela e possui uma cópia de segurança numa posição alternativa do próprio disco.

Os discos com GPT mantêm o MBR para compatibilidade limitada (*protective MBR*) com sistemas operativos que não reconhecem a GPT.

MBR Partition Table and GPT





Fonte: https://msdn.microsoft.com

Formatação de partições

Definir uma nova partição ou volume (por exemplo recorrendo ao comando ***fdisk*** em Linux ou à aplicação ***Disk Management*** em Windows) é uma simples operação de edição da tabela de partições, não tem qualquer efeito sobre a zona do disco correspondente a essa partição. Para poder ser utilizada para armazenamento de dados a zona do disco referenciada na tabela de partições tem de ser preparada (**formatada**).

Existem muitos formatos de sistema de ficheiros, na atualidade os ambientes Windows da Microsoft usam o **NTFS** (*New Technology File System*), nos sistema Linux o mais usado é o **EXT4**.

O sistema de ficheiros guarda várias informações sobre os objetos, nomeadamente a sua localização dentro da partição e atributos tais como o nome e **ACL** (permissões). Toda esta informação fica armazenada em estruturas guardadas na própria partição, o processo de criar estas estruturas é conhecido por operação de **formatação**.

Após a operação de formatação obtém-se um sistema de ficheiros completamente vazio.

Windows – *Basic* and *Dynamic disks*

No Windows Server, inicialmente os discos são do tipo ***basic***, mas podem ser convertidos para ***dynamic***. Os discos *basic* utilizam a tabela de partições do MBR e a GTP.

Os discos convertidos para *dynamic* também possuem a tabela de partições MBR/GTP, mas contêm apenas uma partição designada partição LDM (*Logical Disk Manager*). Os sistemas Linux também suportam LDM.

Os discos *dynamic* permitem diversas configurações de volumes multipartição não suportados para discos *basic*. Para encorajar a utilização de discos *dynamic*, a Microsoft deixou de suportar configurações de volume multipartição nos discos *basic*.

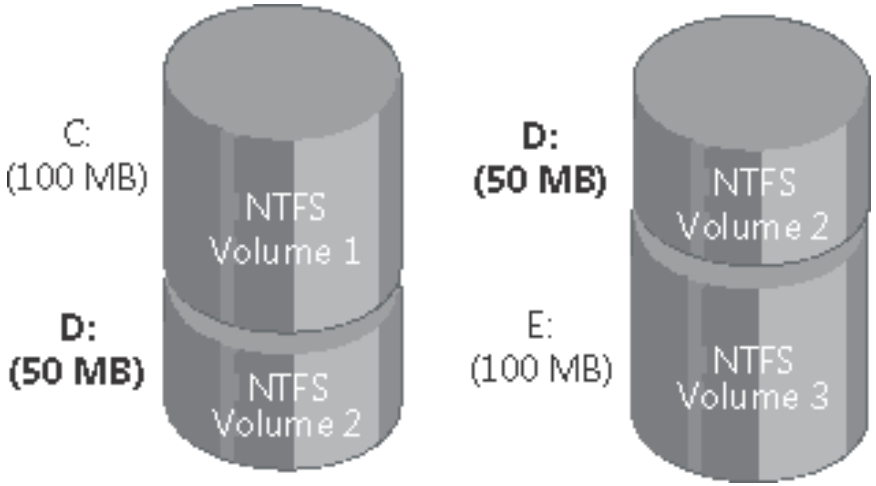
Atualmente um *disco basic* apenas pode conter *simple volumes*, no entanto, um *simple volume* de um disco *basic* pode ser expandido desde que exista espaço livre imediatamente a seguir à partição correspondente e esteja a usar o sistema de ficheiros NTFS. Os volumes definidos num discos *basic* são designados ***basic volumes***, os volumes definidos num disco *dynamic* são designados ***dynamic volumes***.

Dynamic volumes – simple and spanned

Um *dynamic volume* pode ser um ***simple volume***, um *simple volume* encontra-se num único disco, mas pode estar disperso por vários locais do disco. Enquanto existir espaço livre no disco o volume pode ser sempre expandido, desde que não esteja formatado ou esteja formatado com NTFS.

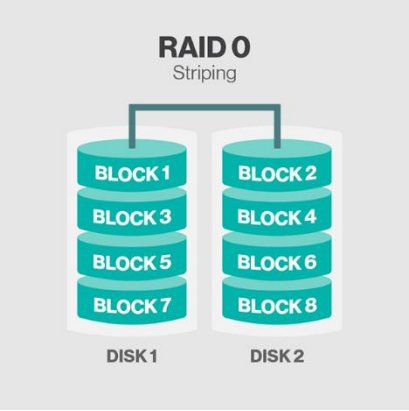
Spanned Volume, é um volume que está disperso por mais do que um disco. Se for necessário expandir um *simple volume* e não existir mais espaço no disco pode ser transformado num *volume spanned*. Uma vez adicionado espaço de um disco a um *volume spanned*, não pode ser removido, todo o volume terá de ser eliminado e reconstruído.

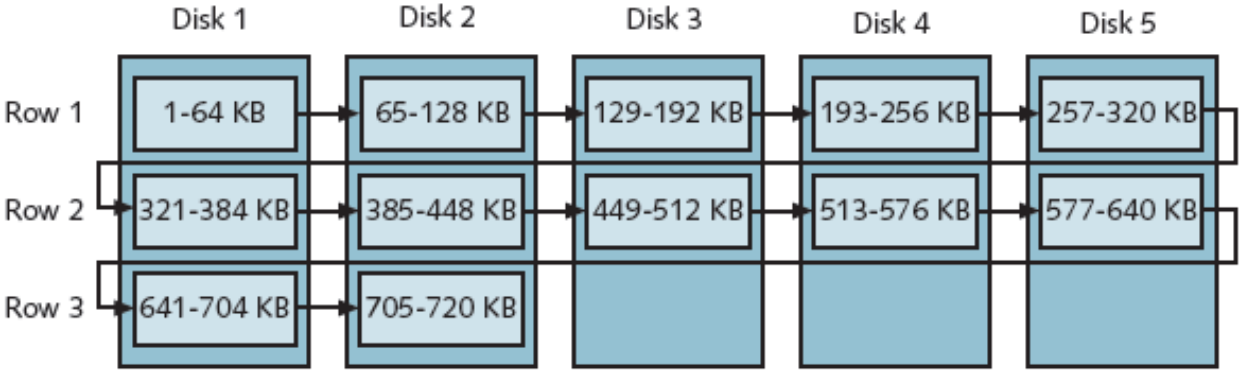
A imagem ilustra um *Spanned Volume* (D:), os espaços usados pelo volume em cada disco podem ser diferentes. Para transformar um *simple volume* em *spanned*, ele tem de estar formatado com NTFS ou não formatado.



Dynamic striped volumes

Um ***striped Volume*** é também um volume que está disperso por mais do que um disco, no entanto, além de permitir obter um volume de maior dimensão, também melhora a performance, correspondendo a um modo de funcionamento **RAID0**. Num volume *striped* são alocados blocos de igual dimensão em todos os discos e os dados são acedidos transversalmente através dos blocos dos diferentes discos.



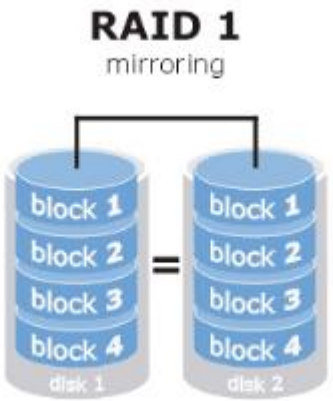


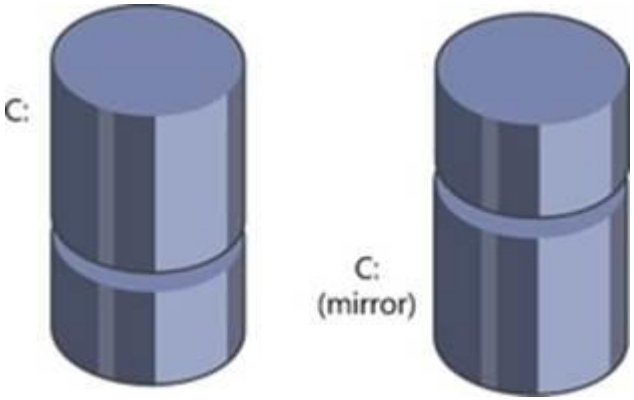
Consegue-se uma melhor performance porque cada ficheiro fica distribuído pelos vários discos, consequentemente as operações de acesso são realizadas em paralelo nos vários discos, algo que não acontece com um volume *spanned* onde só após esgotado o espaço num disco se passa ao seguinte.

Dynamic mirrored volumes

Em termos simples um ***mirrored volume*** é um volume *simple* que tem um duplicado exato em outro disco, transformando-se num *mirrored volume*, corresponde por isso a uma configuração **RAID1.**

O objetivo é proporcionar tolerância a falhas, se um dos discos falhar será utilizada a outra cópia do volume existente no outro disco.



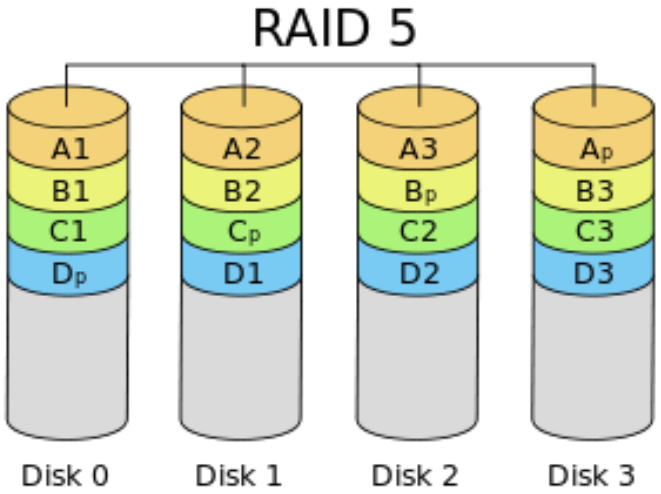


O grande inconveniente é que o espaço disponível no volume é apenas metade do espaço total usado no disco pelo volume. Um volume *mirrored* pode ser desativado obtendo-se dois volumes *simple* que são cópias exatas um do outro.

Dynamic RAID5 volumes

Os volumes RAID5 também proporcionam tolerância a falhas, mas exigem pelo menos 3 discos. O modo de utilização dos discos é semelhante ao *striping*, no entanto para cada sequência de blocos escrita transversalmente ao longo dos discos, num dos discos é guardada informação de recuperação em caso de falha.

Se um dos discos falha, a informação ou não se perde ou é recuperada automaticamente. A figura ilustra este tipo de configuração, A1/A2/A3, B1/B2/B3, C1/C2/C3 e D1/D2/D3 contêm os dados propriamente ditos, Ap/Bp/Cp/Dp contêm informação de recuperação.



Por exemplo se o Disk 0 falha: A1 recupera-se usando A2+A3+Ap, B1 recupera-se usando B2/Bp/B3 e assim sucessivamente.

Pelo facto de se usar *striping*, o RAID5 garante uma melhor performance do que o RAID1, além disso apenas o espaço correspondente a um dos volumes fica indisponível.

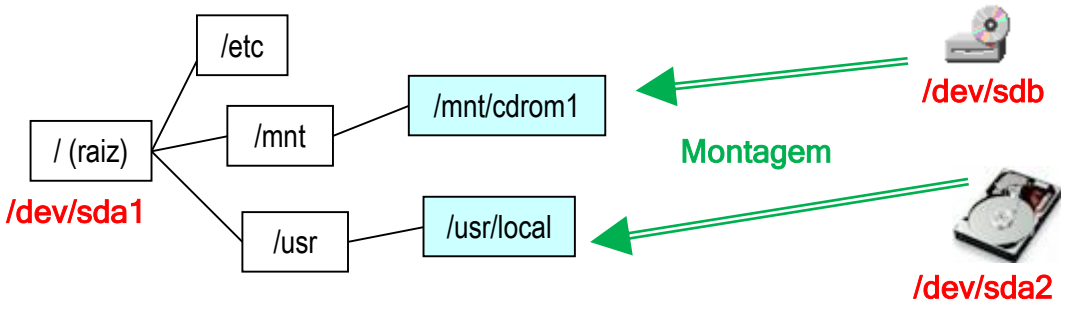
Acesso aos sistemas de ficheiros

O sistema operativo faculta às aplicações acesso os sistemas de ficheiros residentes nas várias partições dos vários discos. Nos sistemas DOS/Windows usam-se as letras de drive para esse efeito, as letras **A:** e **B:** foram tradicionalmente reservadas para os discos flexíveis (em desuso) e por isso as partições (volumes) disponíveis são identificadas por letras a partir de **C:**.

Os sistemas Linux (e Unix em geral) usam uma filosofia diferente, escolhe-se um sistema de ficheiros (partição) para assumir o papel de raiz (***root***). Os outros sistemas de ficheiros existentes, seja qual for o seu tipo, são integrados no sistema de ficheiros raiz através de uma operação designada de montagem (***mount***).

A montagem consiste numa associação lógica criada pelo sistema operativo entre um diretório vazio e um sistema de ficheiros residente numa outra partição ou disco.

Após a montagem, o sistema operativo redireciona para o sistema de ficheiros da partição montada, todos os acessos ao diretório vazio.



Distribuições Linux

Embora apenas exista um único *kernel* Linux, *open source*, gratuito e em desenvolvimento permanente, para obter um sistema servidor funcional é necessário um vasto conjunto de software adicional. O conjunto completo que permite instalar um servidor Linux é conhecido por **distribuição**. Porque algum do software adicional necessário pode não ser gratuito, nem todas as distribuições Linux são gratuitas.

Uma questão fundamental com que um administrador que pretende instalar um servidor Linux se depara é "**Que distribuição usar?**". Existem dezenas de distribuições atuais que se podem agrupar em famílias segundo a distribuição inicial de que derivaram, podendo-se destacar aqui a família **RedHat** e a família **Debian** entre várias outras.

Sendo o *kernel* o mesmo, as diferenças entre famílias residem principalmente no gestor de pacotes de aplicações usado e na forma como os ficheiros de configuração do sistema são organizados. Certamente para um administrador que habitualmente gere sistemas de uma dada família haverá a tendência a manter-se na mesma família pois já está familiarizado com os comandos e a localização dos ficheiros de configuração.

Distribuições Linux

Normalmente as distribuições estão disponíveis em duas versões: *server* e *desktop/workstation*.
No caso do Linux, a maior diferença entre as duas reside no facto de a versão *server* não incluir o suporte de interface gráfica, isso significa que esses recursos ficam disponíveis para a função de servidor.

Num servidor pretendemos acima de tudo estabilidade e longevidade, devemos por isso optar por distribuições estáveis e com suporte de atualizações durante alguns anos.
Por exemplo as distribuições **Ubuntu** (família Debian) existem em versão **LTS** (*Long Term Support*), são disponibilizadas gratuitamente de 2 em 2 anos e são fornecidas atualizações durante 5 anos.
Neste momento a última versão LTS do Ubuntu server é 18.04 e utiliza o *kernel* Linux versão 4.15.0-34.

De referir também a existência de distribuições não gratuitas que incluem o suporte técnico direto, como por exemplo o *Red Hat Enterprise Linux* (RHEL).

A arquitetura inerente a um Linux server tem como desvantagens a necessidade de pacotes adicionais para algumas tarefas de gestão de grupo (como autenticação, por exemplo) e eventual incompatibilidade entre pacotes. Como vantagens pode apontar-se a não obrigatoriedade de um servidor dedicado.

Sistemas operativos Windows Server

As versões servidoras dos sistemas operativos da Microsoft foram introduzidas em 1993 com o primeiro Windows NT Server. A última versão é o Windows 2019 Server.

O Windows Server representa uma alternativa à filosofia de grupo de trabalho (***workgroup***). Até ao Windows NT Server, o único ambiente de rede proporcionado pela Microsoft era ***peer-to-peer*** existindo apenas postos de trabalho que podiam partilhar recursos entre si (***workgroup***).

As versões server permitem um funcionamento ***server-based*** em que existem servidores dedicados sendo então o conceito de ***workgroup*** convertido em ***domain***.

A arquitetura ***server-based*** tem notórias vantagens:

- Autenticação centralizada e única (domínio).
- Maior segurança.
- Centralização de recursos e operações.

Como desvantagens pode apontar-se a necessidade de pelo menos um servidor dedicado e a maior complexidade ao nível da administração.

Sistemas operativos Windows Server - *editions*

O Windows Server não é gratuito, cada versão é disponibilizada em várias *editions*, cada uma das quais pode incluir diferentes funções (roles/features), limites ao número de utilizadores e limites quanto à quantidade máxima de memória (RAM) e número de CPUs que pode utilizar. O preço de cada *edition* pode diferir substancialmente.

Por exemplo o Windows 2019 Server tem as seguintes *editions*:

- Windows Server 2019 Essentials
- Windows Server 2019 Standard
- Windows Server 2019 Datacenter
- Windows Server 2019 Hyper-V

Para poder competir com sistemas operativos servidores que não possuem interface gráfica como o Linux, as versões mais recentes do Windows Server permitem uma instalação do tipo ***Server Core***. Com uma instalação *Server Core,* não é instalada a interface gráfica e toda a gestão do servidor terá de ser realizada através da linha de comando na consola.

Outra opção de instalação disponível no 2019 Server é ***Nano Server***, ainda consome menos recursos do que *Server Core*, mas neste caso toda a administração tem de ser realizada através da rede uma vez que nem sequer é possível o acesso à consola.

Instalação do sistemas operativo

Embora seja possível iniciar o processo de instalação de um sistema operativo numa máquina já em funcionamento, a forma mais habitual é fazer o arranque da máquina a partir de um dispositivo externo de armazenamento já preparado para o efeito, tipicamente um CD/DVD ou um *USB flash drive*.

O dispositivo de arranque contém um sistema operativo elementar (pode não ser o sistema operativo que se vai instalar), servindo de suporte às aplicações que vão proceder à instalação.

O processo de instalação desencadeado desta forma é quase totalmente automatizado, as aplicações que fazem a instalação analisam o hardware e tentam deduzir as configurações mais adequadas.

Normalmente serão solicitadas algumas informações e confirmações ao utilizador, por exemplo quanto à utilização dos discos e respetivo particionamento.

Dependendo do programa de instalação, podemos em alguns casos indicar qual o nosso nível de conhecimento sobre o sistema operativo e processo de instalação e desta forma alterar a quantidade de informação que será solicitada e não decidida automaticamente. Por exemplo o modo de instalação “expert”.

Instalação do sistemas operativo - partições

Na maioria dos casos o programa de instalação toma as decisões corretas, no entanto falta-lhe informação sobre os propósitos exatos que se pretende para o servidor que está a ser instalado.
É verdade que a maioria das configurações estabelecidas no processo de instalação pode facilmente ser alterada posteriormente quando o servidor estiver operacional. Um aspeto a ter em consideração é o particionamento dos discos, alterar partições que estão em uso não é uma tarefa simples, devemos por isso desde logo definir as partições de acordo com os objetivos de longo prazo.

As partições são limites estáticos, se uma partição é demasiado pequena, quando ficar cheia não há uma solução simples para a expandir se existir outra partição a seguir no disco.

Nos sistemas Unix/Linux os limites de quota (espaço máximo que um utilizador pode ocupar) são definidos para cada partição / sistema de ficheiros, consequentemente se forem pretendidos vários valores de quota para diferentes funções (web, mail, etc.) terá de existir uma partição distinta para cada função.

Nos sistemas Windows podem ser definidas quotas globais para uma pasta (quotas de pasta), mas não se referem a utilizadores individuais. As quotas de utilizador são definidas para cada volume (quotas de disco).

Instalação do sistemas operativo - hardware

Atualmente uma grande parte dos servidores são instalados em plataformas de virtualização como servidores virtuais e não sobre hardware físico.

Uma vantagem de usar uma máquina virtual é que ela foi criada especificamente para suportar o sistema operativo pretendido e como tal todos os dispositivos de hardware que estão a ser simulados são facilmente reconhecidos pelo programa de instalação.

Quando se instala um sistema operativo sobre hardware real tudo se pode complicar se o sistema de instalação não possui drivers adequados para esse hardware.

Notoriamente, se o sistema de instalação não possui um driver adequado para o controlador de discos a instalação torna-se de todo impossível.

Para contornar esse problema, alguns sistemas de instalação permitem a adição de drivers de controladores de disco durante o processo inicial de instalação.

Mesmo que o sistema de instalação não consiga suportar todo o hardware existente, o suporte para esse hardware pode mais tarde ser adicionado ao servidor já funcional, mas antes disso, para fazer a instalação é necessário usar o disco.

Arranque do sistema operativo (boot)

A colocação de um sistema operativo em execução numa plataforma de hardware começa com a execução das rotinas de arranque residentes normalmente em algum tipo de ROM (por exemplo o BIOS de um PC). No final são executados uma sequência de programas designados ***bootloader*** que culminam com o carregamento em RAM e execução do sistema operativo.

Num PC, o BIOS carrega e executa o *bootloader* existente no MBR do disco de arranque, no caso do sistema operativo Linux o primeiro *bootloader* executa de seguida um segundo *bootloader* que é habitualmente o **LILO** (*LInux LOader*) ou nas distribuições recentes o **GRUB** (GNU *GRand Unified Bootloader*).

Seja qual for o *bootloader* que coloca o Linux em execução, o processo de arranque não termina ai, depois de estar em execução o *kernel* Linux carrega e executa o **programa inicial**, normalmente **/sbin/init**, que iniciará todos os processos que deverão estar em execução no sistema.

Para que o *kernel* Linux possa funcionar e carregar o programa inicial e restantes necessita de um sistema de ficheiros inicial designado *root* (raiz).

Linux - Sistema de ficheiros raiz (*root*) - /

Quando o *kernel* Linux é colocado em execução pelo último *bootloader* necessita de ter acesso a um sistema de ficheiros inicial designado *root*, estas informações tem ser passadas ao *kernel* na altura em que é carregado através de argumentos, por exemplo:

root=/dev/sda1 **init=/sbin/init**

Significa que o *kernel* deve montar como root (/) a partição */dev/sda1* e no final executar o programa inicial */sbin/init*.

O *kernel* Linux suporta também a utilização de sistemas de ficheiros de rede como raiz (**NFSROOT**) usado por exemplo em máquinas *diskless*.

Uma alternativa muito usada é um *ramdrive* com um sistema de ficheiros raiz temporário:

root=/dev/ram0 **initrd=initrd.gz** **init=/sbin/init**

Todo o arranque inicial do sistema operativo é realizado com o *ramdrive* que é carregado para a memória pelo *bootloader* juntamente com o *kernel*. Na fase final do arranque o sistema de ficheiros raiz é normalmente trocado por outro e o *ramdrive* é descarregado.

Linux - partições SWAP

Os sistemas operativos atuais utilizam memória virtual, isto permite utilizar dispositivos de armazenamento externos para armazenar blocos de memória central (RAM) e desta forma disponibilizar para o sistema mais RAM do que existe fisicamente – operação designada por *paging*. *Swapping* é uma operação de certa forma semelhante, mas orientada por processos, trata-se de transferir temporariamente todos os dados de um processo para o disco.

Nos sistemas Windows a memória virtual baseia-se num ficheiro na pasta de sistema, o sistema Unix utiliza partições especiais do disco. Dada a lentidão do funcionamento dos discos em comparação com a memória central, deve ser vista como um mecanismo tampão que evita que a memória se esgote.

Num sistema Linux, desde que exista espaço em disco livre, é sempre possível acrescentar novas partições SWAP pois o sistema operativo tem capacidade de usar simultaneamente várias destas partições, eventualmente em discos distintos.

A quantidade de memória central de um **sistema servidor** deve ser tal que a memória virtual apenas seja usada muito esporadicamente. Por esta razão num servidor Linux não há interesse em ter partições SWAP muito grandes, geralmente usa-se a regra de **usar um espaço em disco igual à quantidade de memória central**.

Tipos de sistema de ficheiros

Os sistemas operativos atuais suportam vários tipos de sistemas de ficheiros, mas nem todos poderão possuir as funcionalidades mínimas necessárias para o sistema operativo em causa, alguns requisitos fundamentais num sistema de ficheiros de um servidor são:

- **ACL (*Access Control List*)** – cada objeto deve ter uma ACL associada definindo as permissões sobre o objeto.
- ***Journaling*** – registo de operações de escrita antes de serem aplicadas, deste modo havendo uma falha grave não é necessário verificar a integridade de todo o disco, apenas a zona que estava a ser escrita.

Os sistemas Linux atuais utilizam o sistema de ficheiros EXT4 (*fourth extended filesystem*), os sistemas Windows utilizam o NTFS. Ambos utilizam *journaling*, definem permissões (ACLs) e suportam a definição de quotas.

As ACL do NTFS são bastante mais complexas e flexíveis do que as ACL do EXT4, no entanto são parcialmente compatíveis através da interface POSIX (*Portable Operating System Interface*) disponibilizada no NTFS.

Linux - Tabela de sistemas de ficheiros - /etc/fstab

Quando o *kernel* Linux é carregado para a memória uma das suas primeiras tarefas é montar a partição *root* (de acordo com os parâmetros que recebeu). A montagem das restantes partições é realizada mais tarde segundo as instruções existentes no ficheiro de configuração **/etc/fstab**.

Neste ficheiro de texto cada linha representa uma partição a montar e tem o seguinte formato:

device	mounting-point	type	options	dump	fscheckseq

Os 6 parâmetros são separados por espaços ou TAB, consequentemente nenhum dos parâmetros pode conter espaços ou TAB:

- ***device*** – dispositivo ou partição – por exemplo **/dev/sdc3**. Existem várias formas alternativas para especificar o ***device***, nomeadamente através de uma etiqueta (*LABEL*) ou através de um *UUID* (*Universally Unique IDentifier*). O comando **blkid** permite ver os UUID e etiquetas que estão associados a cada partição existente.
- ***mounting-point*** – pasta vazia onde deve ser montada esta partição – por exemplo **/usr/local**

Linux - Tabela de sistemas de ficheiros - /etc/fstab

device	mounting-point	type	options	dump	fscheckseq

- ***type*** – tipo de sistema de ficheiros – por exemplo ***ext4***
- ***options*** – opções de montagem – por exemplo ***ro*** para montar em modo de leitura apenas
- ***dump*** – 0 ou 1. Zero significa que a partição não deve ser incluída nas cópias realizadas com o comando *dump*. Um significa que deve ser incluída. Apenas é relevante se o administrador tiver intenção de usar este comando para efeitos de cópia de segurança (*backup*) e/ou arquivo
- ***fscheckseq*** – ordem de reparação da partição (0, 1, 2 , …). No caso de ocorrer uma falha catastrófica, no arranque seguinte os sistemas de ficheiros devem ser verificados. O valor zero significa "não verificar", normalmente para a partição *root* recomenda-se o valor 1, sendo a primeira a ser verificada. Nas restantes partições que devam ser verificadas recomenda-se o valor 2, sendo desse modo verificadas (sequencialmente ou em paralelo) após concluída a verificação da *root*.

Linux - Tabela de sistemas de ficheiros - exemplos

```
LABEL=Debian    /    ext4    defaults         1      1

/dev/scd0  /media/cdrom0  udf,iso9660  user,noauto,exec,utf8  0  0

/dev/sdb1 /media/disk2 ext2 defaults 0 2

UUID=413eee0c-61ff-4cb7-a299-89d12b075093  /home  ext3  nodev,nosuid,relatime  0  2

UUID=cee15eca-5b2e-48ad-9735-eae5ac14bc90  none   swap   sw   0   0
```

```
UUID=be35a709-c787-4198-a903-d5fdc80ab2f8  /  ext4  relatime,errors=remount-ro  0  1

/dev/sda5          none         swap     defaults        0      0

//server/share  /media/windows  cifs  user=user,uid=1000,gid=100  0     0

server:/share  /media/nfs  nfs  rsize=8192,wsize=8192,noexec,nosuid  0   0
```

Configuração das interfaces de rede

Num sistema servidor as interfaces de rede devem ser configuradas manualmente

Utilizar configuração automática, tipicamente através do protocolo DHCP cria uma dependência externa no sistema (serviço DHCP externo). Num servidor, para o qual se pretende uma elevada fiabilidade/disponibilidade, todas as dependências externas são de evitar.

A configuração manual da interface de rede exige a colaboração do administrador da rede que deverá fornecer os vários elementos:

- Endereço IP (estático) + máscara da rede + endereço do encaminhador da rede (*default gateway*)

- Endereços dos servidores de nomes DNS e nome do domínio local

A única desvantagem da configuração manual é a necessidade de uma sintonia permanente entre o administrador da rede e o administrador do sistema servidor (muitas vezes são o mesmo). As alterações de configuração na rede têm de ser manualmente realizadas no servidor.

Administração de Sistemas (ASIST)

Aula Teórico Prática 02

Utilizadores e grupos.
Administração centralizada.

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Utilizadores, grupos e permissões

O conceito de utilizador é comum a todos os sistemas operativos *multiuser*, é fundamental para distinguir diferentes atores perante o sistema. Cada utilizador tem associado a si um conjunto de **atributos** que constituem o que vulgarmente se designa **conta de utilizador** (*user account*).

As contas de utilizadores são armazenadas pelo sistema operativo em bases de dados apropriadas. Diferentes sistemas operativos utilizam tipos de contas de utilizador diferentes com diferentes tipos de atributos, consequentemente este é um aspeto de difícil integração.

Entre os atributos que são comuns a todos os sistemas operativos encontra-se o ***login name*** através do qual o utilizador se identifica perante o sistema.

A cada utilizador o administrador pode atribuir permissões (o que lhe é permitido fazer). O conceito de grupo de utilizadores (*group account*) vem facilitar esta tarefa uma vez que as permissões atribuídas a um grupo têm efeito sobre todos os seus membros.

As permissões podem ser definidas sobre a forma de direitos (*user rights*) associadas à conta de utilizador ou de grupo, ou sob a forma de uma **ACL** (*Access Control List*) associada a um objeto.

Sistemas Unix – Utilizadores

Nos sistemas Unix a base de dados fundamental de utilizadores é o ficheiro de texto **/etc/passwd**. Outras bases de dados podem ser usadas em complemento a esta.

```
root:x:0:0:root:/root:/bin/bash
nobody:x:65534:65534:Nobody:/:/bin/sh
www:x:98:98:Gestor WWW:/users/home/www:/usr/bin/tcsh
i977805:x:2176:102:Joaquim Cardoso Morais:/users/home/i977805:/bin/csh
artur:$1$s2b9pyc6$46nIOl8G1fGrymmvsJejG/:1203:98:Artur:/users/home/artur:/bin/bash
```

A imagem acima ilustra um exemplo do conteúdo deste ficheiro, neste caso com 5 contas, começando pelo utilizador **root** (administrador do sistema, possui todas as permissões). Cada linha do ficheiro corresponde a uma conta de utilizador e contém **7** atributos separados pelo símbolo de dois pontos:

- *Login-name* – nome único que identifica o utilizador no sistema.
- *Hash* da *password*. O valor **x** indica que o *hash* se encontra no ficheiro **/etc/shadow**
- UID (*User IDentifier*) – número único de 16 bits que identifica o utilizador.
- GID (*Group IDentifier*) primário – número de 16 bits que indica o GID do grupo primário deste utilizador.
- Nome longo ou descrição do utilizador.
- Pasta inicial (*home*). Pasta corrente após o *login*.
- Programa inicial. Tipicamente uma *shell* executada após o *login* (se não estiver preenchido, aplica-se a *shell* por omissão).

Sistemas Unix – Autenticação de utilizadores

O *hash* da password guardado no ficheiro **/etc/passwd** é usado para autenticação, quando o utilizador fornece a *password* o sistema aplica-lhe a função *hash* e compara o resultado com o que está no **/etc/passwd**; sendo iguais a autenticação é válida.

O sistema operativo exige que o ficheiro **/etc/passwd** seja legível por todos os utilizadores, consequentemente concluiu-se que não é o local ideal para guardar os *hash* das *passwords* dos utilizadores.

Note-se que uma função *hash* não é reversível, por isso não é possível obter a *password* a partir do *hash*, mas o facto de haver acesso a essa informação abre a possibilidade de ataques de força bruta tentando várias *passwords* aleatórias e vendo se por acaso após aplicar a função *hash* a alguma delas o resultado é igual ao que está no **/etc/passwd**.

Para evitar esta vulnerabilidade os *hash* das passwords no **/etc/passwd** podem ser substituídos pelo símbolo **x** que significa que o *hash* se encontra no ficheiro auxiliar **/etc/shadow**, não sendo possível aos utilizadores lerem este ficheiro auxiliar.

De notar que a colocação da *hash* no ficheiro **/etc/shadow** não impede os ataques de força bruta, apenas os dificulta pois não é possível a um utilizador comparar a *hash* gerada com aquela que está armazenada.

Sistemas Unix – Grupos de utilizadores

Nos sistemas Unix, as contas de grupos de utilizadores são armazenadas no ficheiro de texto **/etc/group**, outras bases de dados podem ser usadas em complemento a esta.

O formato é idêntico ao do ficheiro **/etc/passwd**, na figura ao lado é apresentado um exemplo. Os atributos de cada grupo são o nome do grupo, o campo seguinte (**x**) não é usado atualmente, segue-se o **GID** (*Group IDentifier*). Quer o nome do grupo quer o GID têm de ser únicos.

```
root:x:0:
bin:x:1:
nogroup:x:65534:
sshd:x:77:
users:x:100:
inf:x:102:
profs:x:98:
dom_users:x:1003:artur,i977805
```

A primeira linha do exemplo define o grupo **root**. Embora nomes de grupos e nomes de utilizadores sejam totalmente independentes, por omissão é criado para cada utilizador um grupo com o mesmo nome que é o grupo primário desse utilizador.

O último atributo é uma lista separada por virgulas de nomes de utilizadores que são membros do grupo (como grupo secundário) para além do respetivo grupo primário.

Um utilizador pode pertencer a um grupo por duas vias: porque é o seu grupo primário (definido no ficheiro /etc/passwd) ou porque está na lista de membros do grupo (no ficheiro /etc/group).

Linux – Gestão de utilizadores e grupos

A gestão de utilizadores e grupos locais (definidos nos ficheiros **/etc/passwd**, **/etc/group** e **/etc/shadow**) deve ser realizada através de um conjunto de comandos padrão do Linux existentes na maioria das distribuições: **useradd** ; **usermod** ; **userdel** ; **groupadd** ; **groupmod** ; **groupdel** ; **chfn** ; **chsh** ; **passwd**; **pwconv**.

Estes comandos têm a vantagem de terem uma solidez comprovada e realizarem algumas validações e operações acessórias.

Uma alternativa que deve ser utilizada com cautela consiste na manipulação direta dos ficheiros, ou seja usar um editor de texto para alterar os ficheiros **/etc/passwd** e **/etc/group**. Neste caso há necessidade de evitar qualquer tipo de incoerência ou sobreposição de identificadores. Estes ficheiros são lidos sequencialmente, se houver uma duplicação de identificadores, apenas o primeiro será válido.

Note-se que o utilizador **root** (UID=0) e grupo **root** (GID=0) são fundamentais para o funcionamento do sistema, sem elas o sistema nem sequer poderá arrancar.

Sistemas Windows – Utilizadores e Grupos

No Windows Server só existem contas de utilizadores locais e grupos locais se o servidor não estiver integrado em nenhum domínio Windows, ou seja estiver a operar como um *Standalone Server*.

Num sistema Windows *standalone*, as contas de utilizador estão armazenadas numa base de dados do sistema operativo designada SAM (*Security Account Manager*). Quando um sistema Windows é integrado num domínio deixa de utilizar a SAM local e passa a usar a base de dados *Active Directory* do domínio.

A quantidade de atributos associada às contas de utilizador e grupos Windows é bastante extensa, podendo mesmo ser expandida no caso do *Active Directory*. Uma diferença que desde já se pode destacar relativamente ao Unix é que ao contrário daquele, nos sistemas Windows **um grupo pode ser membro de outro grupo**. Uma outra diferença é que nos sistemas Windows contas de grupo e de utilizadores são armazenadas conjuntamente e em consequência não é possível ter um nome de utilizador igual a um nome de grupo.

Quando o Windows Server é instalado são criadas automaticamente diversas contas de utilizador e grupo fundamentais para o funcionamento do sistema. O nome do utilizador que possui direitos totais de administração é ***Administrator*** e pertence ao grupo ***Administrators*** (se a versão do Windows for a inglesa).

Nomes de Utilizadores e de Grupos

Nos sistemas **Windows** os nomes de utilizadores e grupos podem ter até 64 caracteres. Alguns caracteres são proibidos, mas podem ser usados **espaços e pontos**, a Microsoft desaconselha a utilização de espaços no caso de nomes de utilizadores. Os nomes de utilizadores e de grupos podem ter maiúsculas e minúsculas, mas **não são *case sensitive*.**

Nos sistemas **Linux** as restrições são maiores, não devido ao sistema operativo propriamente dito mas devido a um vasto conjunto de *software open source* com o qual é necessário manter a compatibilidade. As recomendações são as seguintes:

- Não conter letras maiúsculas.
- O nome tem de começar por uma letra minúscula ou um *underscore*.
- Os caracteres seguintes também podem ser dígitos numéricos ou o sinal menos. Adicionalmente o último caractere também pode ser o dólar. Não são permitidos outros caracteres, incluindo o espaço.
- O nome não deve exceder os 31 carateres de comprimento.

Centralização de contas

Num ambiente de rede de trabalho típico, os vários utilizadores autorizados devem ter acesso a um conjunto de servidores e outros recursos. Não é obviamente praticável manter localmente em cada um deles uma base de dados de utilizadores e grupos independente.

A solução é ter uma única base de dados de contas, central, e que seja acessível a todos os equipamentos através da rede usando protocolos de aplicação apropriados. Dessa forma, qualquer alteração na base de dados central de contas tem efeito imediato sobre todos os equipamentos que a estão a usar.

Para os utilizadores é muito mais cómodo usar apenas um único conjunto de credenciais e não um diferente para cada recurso a que acede.

Com a centralização das contas obtém-se igualmente um **sistema de autenticação centralizada** dos utilizadores. Embora as credenciais sejam iguais em todos os recursos/serviços, o utilizador tem de se autenticar em cada um deles.

Um **sistema de autenticação *single sign-on* (SSO)**, é ainda mais cómodo para os utilizadores. Como o nome indica o utilizador apenas necessita de se autenticar uma vez, após isso tem acesso a todos os recursos. Este é o tipo de autenticação que temos nos domínios Windows da Microsoft.

Domínios Windows da Microsoft

Os domínios Windows surgiram com o Windows NT Server, trata-se de um sistema de autenticação centralizada ***single sign-on***.

O funcionamento do domínio é centralizado numa máquina Windows Server com a função ***Domain Controller*** (**DC**). O DC contém a base de dados com as contas de utilizadores, grupos e outros constituintes do domínio. Desde o Windows 2000 Server, as base de dados dos DC são do tipo *Active Directory* (**AD**), utilizando o *standard* **LDAP** (*Lightweight Directory Access Protocol*).

Antes da introdução do AD, cada domínio possuía um DC designado **PDC** (*Primary Domain Controller*) que continha a base de dados do domínio. Num domínio só pode existir um PDC, mas podem existir outros DC que contêm cópias de leitura apenas da base de dados.

Com a introdução do AD o conceito de PDC desaparece embora continue a ser emulado para compatibilidade com sistemas mais antigos. Um domínio AD pode conter vários DC, as bases de dados e funções de controlo necessárias (*Active Directory Domain Services* – **AD DS**) são distribuídas pelos vários DC existentes, alguns deles vão ter funções especais (***operations masters***).

Estrutura lógica do Active Directory - Tree

O ***Active Directory*** (**AD**) não se limita a domínios, existe uma infraestrutura lógica que integra os domínios.

Os nomes de domínio AD correspondem a nomes qualificados de domínios DNS. Uma super estrutura lógica de domínios que pode ser criada é a árvore de domínios (***Domain Tree***).

Uma árvore corresponde a um **conjunto de domínios DNS contíguos**, ou seja um domínio DNS e eventualmente subdomínios DNS. O domínio AD correspondente ao domínio DNS base (mais próximo da raiz) que pertence à árvore designa-se ***tree root domain***.

Um domínio AD pertence sempre a uma árvore, se for o único domínio da árvore é também o ***tree root domain***.

Quando um novo domínio é criado, por *promoção* de um Windows Server de ***Standalone Server*** a ***Domain Controller***, o novo domínio ou é integrado numa árvore já existente, se o nome DNS corresponde a um subdomínio de outro que já pertence a uma árvore, ou então será criada uma nova árvore, tendo o novo domínio como raiz.

Estrutura lógica do Active Directory - Forest

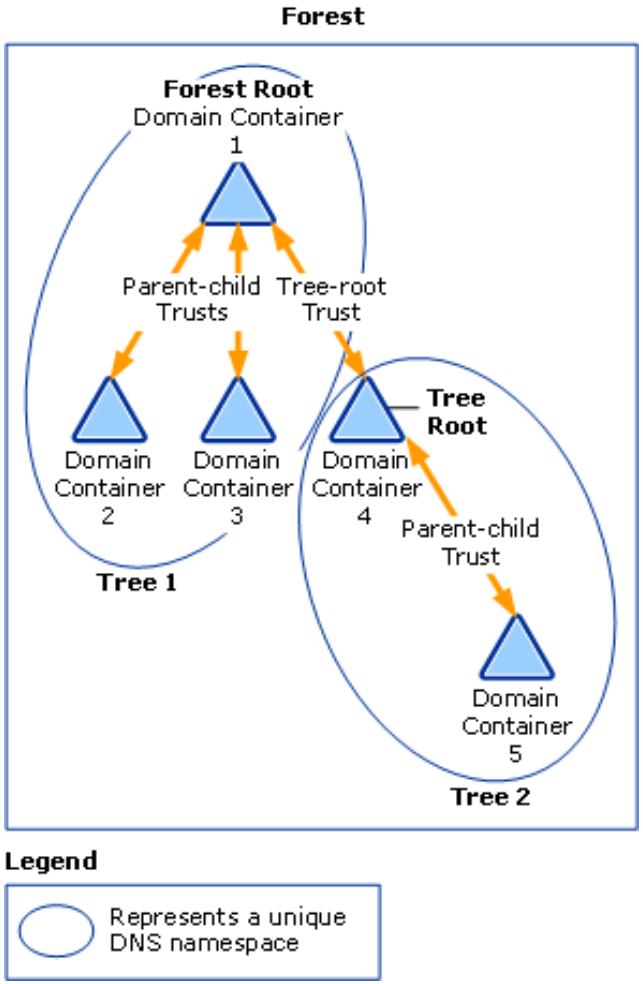
Uma floresta é um conjunto de árvores, e consequentemente um conjunto de domínios, e **ao contrário do que acontece com os domínios de uma árvore, pode não existir qualquer relação entre os nomes DNS das árvores que integram uma floresta.**

Quando um novo domínio é criado tem de ser integrado numa floresta já existente ou terá de ser criada uma nova floresta. O primeiro domínio a ser adicionado a uma floresta passa a ser o ***forest root domain*** e a floresta passa a ser identificada através do nome DNS desse domínio.

Uma floresta estabelece uma relação de confiança entre todos os seus domínios.

Dito de forma simples isto significa que um utilizador autenticado num dos domínios da floresta terá acesso aos recursos de outros domínios da floresta.

A fronteira da floresta no seu conjunto constitui uma *security boundary*. A floresta também pode ser vista como o nível mais elevado da organização lógica do AD.



Fonte: https://technet.microsoft.com

Active Directory Federation Services (AD FS)

Trata-se de um sistema de autenticação ***single sign-on*** (**SSO**) que permite ultrapassar os limites de confiança definidos pela floresta.

A federação não deve no entanto ser interpretada como sendo um nível lógico do AD acima das florestas.

Foi pensado para permitir a integração da autenticação entre organizações diferentes (***federation partners***) que desta forma constituirão uma **federação.**

Estes serviços estão vocacionados para serem usados diretamente por aplicações tais como serviços Web e utilizam a internet de forma segura para aceder a servidores de autenticação específicos designados ***AD FS servers.***

Os utilizadores de organizações que fazem parte da mesma federação (***federation partners***) necessitam de se autenticar apenas uma vez, após isso têm acesso às aplicações (eventualmente Web) disponibilizadas pelos ***federation partners*** sem necessidade de autenticações adicionais.

Active Directory Domain Controllers

A estrutura lógica do *Active Directory* é suportada por um conjunto de máquinas Windows Server designadas *Domain Controllers* (**DC**), ou seja disponibilizam os *Active Directory Domain Services* (**AD DS**).

Cada domínio AD tem de possuir pelo menos um DC, a Microsoft aconselha um mínimo de dois para garantir tolerância a falhas.

Uma máquina Windows Server pode encontrar-se em uma de 3 situações:

- Após a instalação inicial, não está integrado em nenhum domínio e designa-se ***Standalone Server***.

- Pode ser integrado num domínio sem ser um *Domain Controller*, neste caso designa-se ***Domain Member***, tal como os postos de trabalho.

- Pode ser promovido a ***Domain Controller***, eventualmente criando um novo domínio, uma nova árvore e uma nova floresta.

Dependendo do contexto em que é promovido a *Domain Controller* pode exercer ainda funções especiais no *Active Directory*, nomeadamente ***Global Catalogue Server*** e funções ***Operations Master***.

Em cada floresta existe um *global catalogue*, trata-se de uma base de dados que contém réplicas das partes mais importantes de cada uma das bases de dados AD de cada domínio que integra a floresta. O *global catalogue* é fundamental para permitir operações ao nível da floresta.

Active Directory Domain Controllers

O primeiro *domain controller* a ser criado numa floresta passa a ser o *global catalogue server* contendo essa base de dados, posteriormente, outros *domain controller* da floresta podem também tornar-se *global catalogue servers.*

As diferentes funções *operations master* têm de ser executadas por um único *domain controller (Single-Master Operations)* dentro do nível da estrutura a que se aplicam. Assim:

- Numa floresta: além dos *global catalogue servers*, tem de existir exatamente um DC a exercer a função ***Schema Master*** e um DC a exercer a função ***Domain Naming Master***.

- Num domínio: tem de existir exatamente um DC a exercer a função ***Primary Domain Controller (PDC) Emulator***, um DC a exercer a função ***Infrastructure Master*** e um DC a exercer a função ***Relative ID (RID) Master***.

Quando existem vários DC no mesmo domínio a sincronização do *Active Directory* é garantida (*Multimaster replication*) e é proporcionada tolerância a falhas.

Também é possível definir um DC como sendo *read-only* (**RODC**). Operações como uma pesquisa podem ser realizadas pelo RODC, operações que envolvem alterações não são possíveis. Podem ser instalados em locais remotos de segurança precária e com problemas de ligação aos DC.

Estrutura física do Active Directory - Sites

A infraestrutura lógica de domínios, árvores e florestas não possui qualquer reflexo da realidade física subjacente. Para armazenar informação sobre a localização física é introduzido o conceito de ***Site Object*** no contexto de floresta. Assim sob o ponto de vista físico **a floresta está dividida em sites**.

Isto só acontece porque a floresta é o topo da hierarquia lógica, não há relação direta entre Sites (conceito físico) e domínios, árvores e florestas (conceitos lógicos).

Assim: **um domínio pode estar disperso por vários *Sites***, também: **um *Site* pode fazer parte de vários domínios**.

A relevância de dividir a floresta em sites relaciona-se com a otimização dos mecanismos de replicação entre *domain controllers* dentro da floresta.

A definição dos Sites é realizada com base nas configurações de rede existentes nos DC. Cada ***Site Object*** está associado a um ou mais ***Subnet Objects***, um *subnet object* é uma representação de uma rede de nível 3 (IPv4 ou IPv6). Outros objetos como servidores e workstations estão associados a Sites, com esta informação consegue-se por exemplo determinar para uma workstation qual é o DC do domínio pretendido que está fisicamente mais próximo, eventualmente no mesmo Site.

Active Directory – Organizational Units (OU)

Sob o ponto de vista lógico, um domínio pode ser divido numa estrutura em árvore de ***Organizational Units*** (**OU**).

O objetivo das OU é facilitar a organização e a administração. Uma OU pode conter objetos do domínio tais como utilizadores, grupos, computadores e até outras OU.

As OU devem refletir a estrutura hierárquica ou lógica da organização, cada OU pode por exemplo representar um departamento ou uma secção, é importante que existe alguma relação lógica entre os objetos que se colocam na mesma OU.

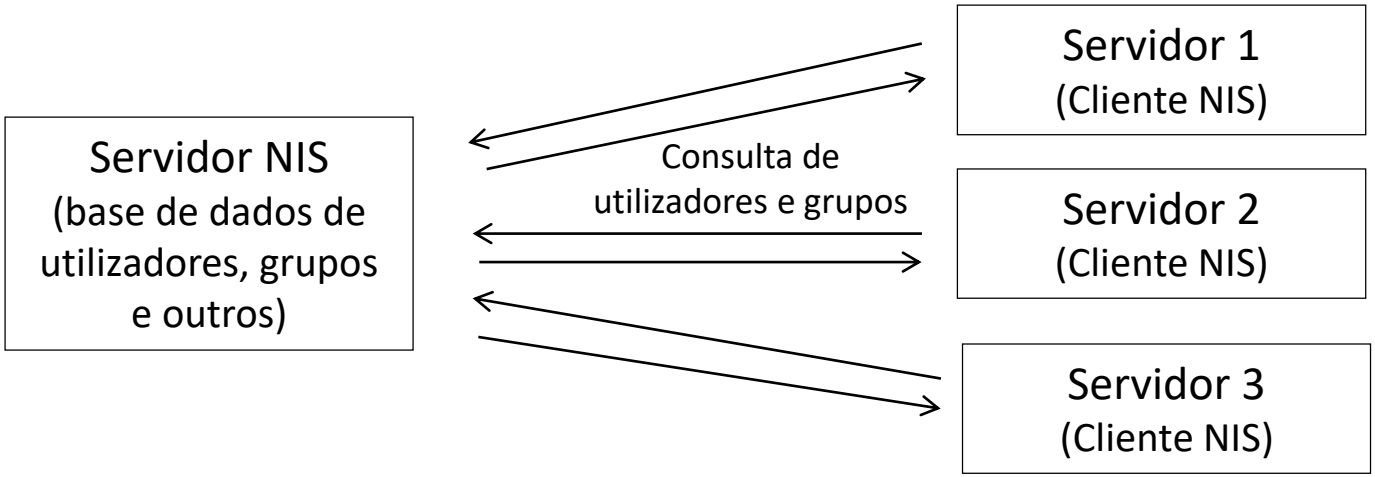
Sob o ponto de vista administrativo destacam-se duas utilizações:

- Uma ***Group Policy*** pode ser aplicada a uma OU tendo efeito sobre todos os objetos contidos na OU.
- Podem ser dadas a utilizadores e grupos permissões sobre uma OU. Desta forma é possível por exemplo criar um administrador de uma OU com permissões para criar utilizadores e grupos nessa OU apenas.

Unix - diversificação dos repositórios do sistema

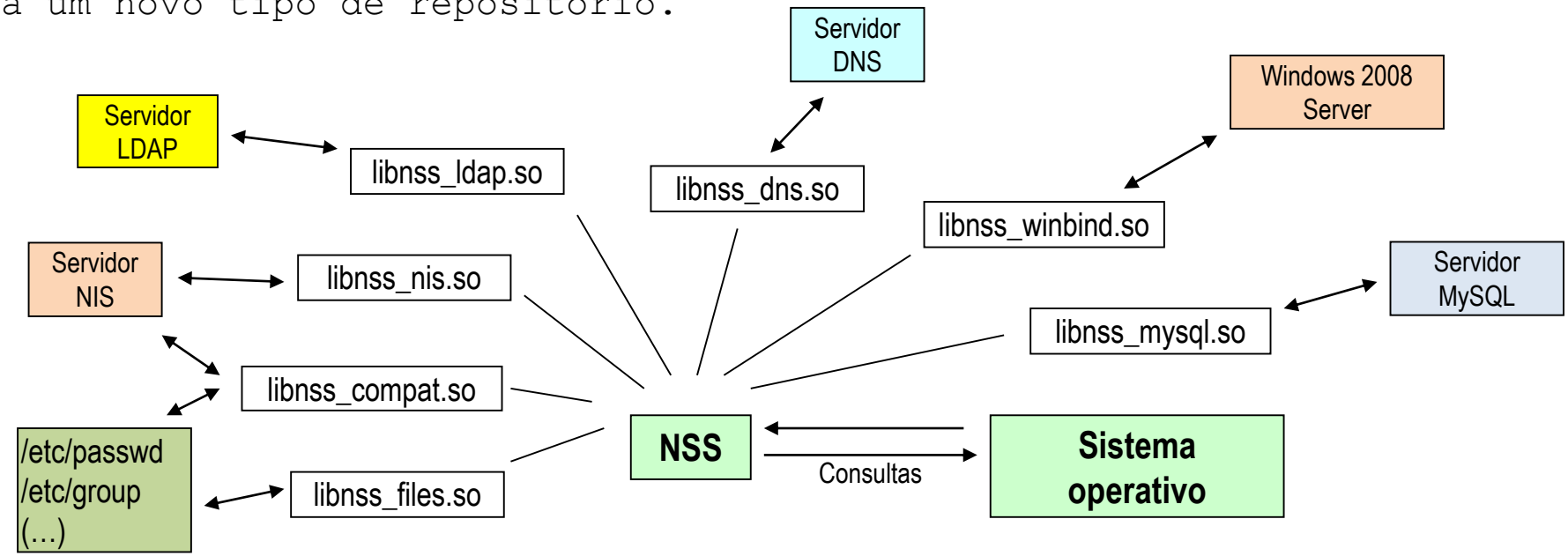
Nos sistemas ***Unix*** desde cedo se sentiu a necessidade de diversificar e centralizar os repositórios de informação de configuração do sistema (entre outros, bases de dados de utilizadores e grupos), essa necessidade levou ao desenvolvimento de sistemas alternativos, um dos mais importantes é o sistema centralizado **NIS** (*Network Information Service*), também conhecido por *Yellow Pages*.

A **centralização** destes repositórios tem enormes vantagens sob o ponto de vista de administração de um conjunto de servidores. Se todos os servidores usarem o mesmo repositório central o administrador consegue gerir toda a informação num único local, os utilizadores e grupos serão os mesmos em qualquer um dos servidores do conjunto.



Name Service Switch (NSS)

O NSS é um sistema modular que permite integrar nos sistemas Linux diversos tipos de repositórios de informação de sistema. Através da adição de um novo módulo (biblioteca dinâmica) acrescenta-se o suporte para um novo tipo de repositório.



Nem todos estes módulos são vocacionados para bases de dados de utilizadores e grupos, por exemplo o *libnss_dns* é normalmente usado para nomes de máquinas.

O módulo *libnss_compat* implementa um modo de compatibilidade com sistemas pré NSS consultando os ficheiros locais e recorrendo adicionalmente a servidores NIS.

Configuração do *Name Service Switch*

A configuração do NSS consiste em definir a ordem pela qual os vários repositórios disponíveis (módulos) vão ser usados. Por motivos de segurança e fiabilidade deve ser dada preferência aos repositórios locais. O ficheiro de texto **/etc/nsswitch.conf** contém a configuração.

Para cada tipo de informação define a sequência de pesquisa. Exemplo:

```
bash-3.00$ cat /etc/nsswitch.conf
#
passwd:     files ldap winbind nis
shadow:     files nis
group:      files ldap winbind nis

hosts:      files nis dns winbind
```

No exemplo as definições de utilizador (***passwd***) serão pesquisadas pela seguinte ordem:

1º Ficheiros locais (***files***)

2º Servidores LDAP (***ldap***)

3º Servidores Windows (***winbind***)

4º Servidores NIS (***nis***)

Será usado o primeiro que for encontrado, <u>terminando a pesquisa</u>

Configuração do *Name Service Switch*

A configuração do NSS propriamente dito é bastante simples, no entanto, muitos dos módulos NSS usados necessitam de configuração específica.

Por exemplo o módulo *libnss_ldap* que permite obter informação em servidores de base de dados através do **LDAP** (*Lightweight Directory Access Protocol*) usa normalmente o ficheiro de configuração **/etc/ldap.conf**. Neste ficheiro, entre outros elementos, vai ter de ser especificado:

- Endereço IP ou nome DNS do servidor LDAP
- DN (*Distinguish Name*) da raiz da base de dados LDAP
- Versão do protocolo LDAP

Ao integrar repositórios remotos de utilizadores e grupos devemos ter consciência de que, o anteriormente referido conjunto de comandos Linux para gestão de utilizadores e grupos, funciona apenas com contas locais armazenadas nos ficheiros **/etc/passwd**, **/etc/shadow** e **/etc/group**.

Em conjunto com o NSS normalmente é também usado o **PAM** (*Pluggable Authentication Module*), os dois não são indissociáveis qualquer um deles pode funcionar sem o outro. Entre outos aspetos, o PAM trata do processo de login dos utilizadores. Quando os utilizadores estão em repositórios remotos o PAM tem de recorrer a eles.

Linux Pluggable Authentication Module (PAM)

O Linux-PAM é usado por todas as distribuições Linux atuais. Trata-se de uma biblioteca modular usada pelas aplicações e serviços que processam as entradas e saídas de utilizadores no sistema.

O PAM trata 4 tipos de tarefas (***management groups***):

- **account** – permite realizar validações antes da autenticação (Exemplos: horário de acesso, endereço de origem) e após a autenticação (Exemplos: acesso permitido para o utilizador, password expirada).
- **auth** – trata-se da autenticação propriamente dita, normalmente através de *username/password*.
- **password** – alteração pelo utilizador da respetiva password.
- **session** – registo e preparação do ambiente. Regista a entrada do utilizador após autenticação bem sucedida, prepara o ambiente de trabalho para o utilizador. Pode ser novamente invocada no *logout*, por exemplo para registar a saída do utilizador.

Existe uma grande variedade de módulos para implementar estas tarefas, muitos deles especificamente desenvolvidos para funcionarem como interface com repositórios remotos. Alguns módulos podem ser usados para vários tipos de tarefas, outros são específicos para determinadas tarefas.

Linux-PAM e autenticação em Linux

A forma tradicional de autenticação de utilizadores em UNIX consiste em comparar o *hash* da *password* que foi fornecida pelo utilizador com o *hash* que está armazenado na conta do utilizador (/etc/passwd ou /etc/shadow).

Com a diversificação de repositórios de contas de utilizador, a forma de realizar a autenticação passou a depender do tipo de repositório. O sistema PAM resolve o problema, tal como o NSS é modular tipicamente para cada módulo NSS existe um módulo PAM para o mesmo tipo de repositório. Por exemplo para LDAP existe o **libnss-ldap** e o **pam-ldap**.

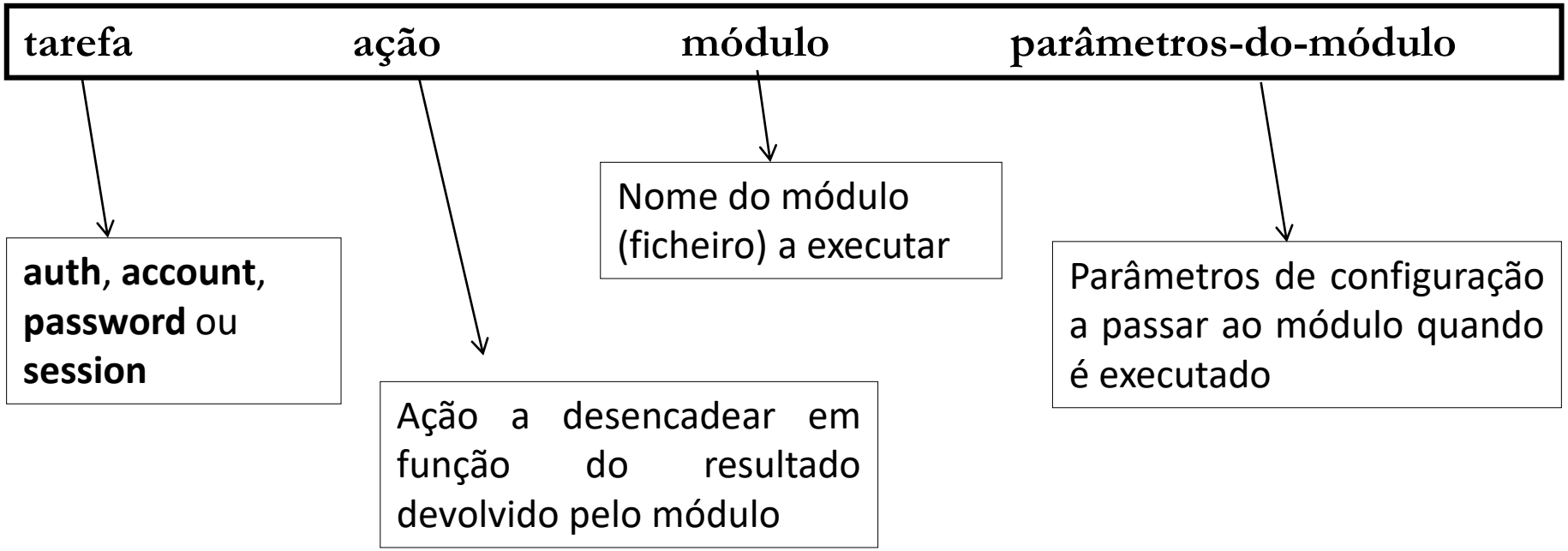
Com o PAM as formas de autenticação podem ser facilmente diversificadas deixando de estar limitadas aos tipos de *hash* suportados nos ficheiros /etc/passwd e /etc/shadow.

O fornecimento direto do par *username/password* ao sistema de autenticação é conhecido por **PAP** (*Password Authentication Protocol*), mas só deve ser usado através de canais de comunicação seguros.

O protocolo **CHAP** (*Challenge Handshake Authentication Protocol*), muitas vezes designado também de autenticação com password protegida, nunca transmite a password legível. O CHAP não pode ser implementado através dos *hash* tradicionais do Unix, mas pode ser implementado com recurso a módulos PAM apropriados.

Linux-PAM - configuração

Configurar o sistema PAM consiste em definir a sequência de módulos a utilizar em cada tipo de tarefa, isso é conseguido com ficheiros de texto, habitualmente residentes na pasta **/etc/pam.d/**, em que cada linha representa a invocação de um módulo no contexto de uma tarefa, na forma:



Os módulos PAM podem devolver vários resultados, o valor zero (**PAM_SUCCESS**) indica sucesso total, outros valores indicam vários tipos de erro. A ação a realizar é definida em função deste valor devolvido pelo módulo.

Linux-PAM – configuração - ação

tarefa	ação	módulo	parâmetros-do-módulo

required – em caso de sucesso continua processamento dos módulos seguintes, em caso de falha também continua a processar os módulos seguintes, mas o resultado final vai ser falha.

requisite - em caso de sucesso continua o processamento dos módulos seguintes, em caso de falha termina de imediato resultando em falha.

sufficient – em caso de sucesso e não existindo falhas *required* anteriores, devolve sucesso e termina a cadeia. Em caso de falha regista uma falha *optional* e continua a processar a cadeia.

optional - em caso de sucesso continua a cadeia, em caso de falha regista uma falha *optional* e continua a processar a cadeia.

Linux-PAM – configuração - ação

A ação a realizar pode ser especificada caso a caso para cada valor possível devolvido pelo módulo:

tarefa	ação	módulo	parâmetros-do-módulo

[value1=ação1 value2=ação2 value3=ação3 …]

Onde **value** representa o resultado devolvido, por exemplo **success** ou **default** e **ação** a ação a realizar nesse caso, as ações podem ser entre outras:

ok – assinala um sucesso e continua a processar os módulos seguintes

done – assinala um sucesso e termina o processamento com sucesso

die – assinala uma falha e termina o processamento com falha

{*NN*} – assinala um sucesso, ignora os *NN* módulos (linhas) seguintes e continua o processamento no módulo (linha) *NN+1*. ({NN} é um número inteiro superior a zero)

Podem referir-se as seguintes equivalências entre formas de especificar a ação:	required	⇔	[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
	requisite	⇔	[success=ok new_authtok_reqd=ok ignore=ignore default=die]
	sufficient	⇔	[success=done new_authtok_reqd=done default=ignore]
	optional	⇔	[success=ok new_authtok_reqd=ok default=ignore]

Linux-PAM – alguns módulos comuns

pam_unix.so	Pode ser usado em qualquer das tarefas, implementa as funcionalidades tradicionais do UNIX pré-PAM.
pam_deny.so	Pode ser usado em qualquer das tarefas, devolve sempre falha. Usado para testes ou para finalizar uma sequência.
pam_env.so	Módulo **auth** e **session** que permite manipular as variáveis de ambiente segundo um ficheiro de configuração, geralmente "/etc/security/pam_env.conf".
pam_mail.so	Módulo **auth** e **account** que faz a verificação de mail no inicio da sessão.
pam_ldap.so	Módulo de interface com repositórios LDAP, suporta as funcionalidades **auth**, **account** e **password**.
pam_issue.so	Módulo **auth** que permite apresentar uma mensagem antes da autenticação do utilizador.
pam_motd.so	Módulo **session** que apresenta uma mensagem após a entrada no sistema (***message of the day***).
pam_nologin.so	Módulo **auth** que devolve sempre sucesso para o administrador e devolve sucesso para os outros utilizadores, a menos que exista o ficheiro **/etc/nologin**.
pam_listfile.so	Pode ser usado em qualquer das cadeias, permite devolver sucesso ou falha em função de uma lista residente num ficheiro, por exemplo com uma lista de utilizadores autorizados.
pam_cracklib.so	Módulo **password** que verifica a solidez da nova *password* que o utilizador pretende usar. Além de a verificar no dicionário do sistema pode também fazer algumas verificações adicionais.
pam_faildelay.so	Módulo **auth** que permite definir o tempo que decorre desde que o utilizador falha a autenticação até que lhe é apresentada a mensagem de erro.

Linux-PAM – utilização pelas aplicações

A configuração do PAM é independente para cada aplicação, cada aplicação é identificada por um **nome de serviço** fornecido à API pela aplicação. Deve existir na pasta **/etc/pam.d/** um ficheiro com o nome de serviço fornecido, será esse o ficheiro de configuração usado.

Exemplo para o serviço SSH que se identifica à API com o nome **sshd**, ficheiro **/etc/pam.d/sshd**:

```
@include common-auth
account  required     pam_nologin.so
account  required     pam_access.so
@include common-account
session    required     pam_loginuid.so
session    optional     pam_keyinit.so force revoke
@include common-session
session    optional     pam_motd.so  motd=/run/motd.dynamic
session    optional     pam_motd.so noupdate
session    optional     pam_mail.so standard noenv
session    required     pam_limits.so
session    required     pam_env.so user_readenv=1 envfile=/etc/default/locale
@include common-password
```

Como se pode observar grande parte das configurações encontra-se em ficheiros incluídos que são partilhados por vários serviços.

Linux-PAM – exemplo /etc/pam.d/common-auth

```
# /etc/pam.d/common-auth - authentication settings common to all services


auth    [success=2 default=ignore]      pam_unix.so nullok_secure
auth    [success=1 default=ignore]      pam_ldap.so use_first_pass
auth    requisite                       pam_deny.so
auth    required                        pam_permit.so
auth    optional                        pam_cap.so
```

Este ficheiro destina-se a ser incluído nas definições **auth** dos vários serviços através da diretiva **@include common-auth.**

Podemos observar que um utilizador será autenticado com sucesso se a autenticação é bem sucedida através dos ficheiros passwd/shadow (**pam_unix.so**) ou a autenticação é bem sucedida através do **bind** (login) no repositório LDAP (**pam_ldap.so**). Se falhar em ambos será atingido o módulo **pam_deny.so** e a autenticação falha.

A informação de configuração do módulo **pam_ldap.so** é a mesma que é usada pelo módulo **libnss_ldap** do NSS, ou seja o ficheiro **/etc/ldap.conf.**

Administração de Sistemas (ASIST)

Aula Teórico Prática 03

Permissões e direitos de utilizadores.
Áreas de trabalho dos utilizadores (*homes*).
Controlo de quotas.

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Grupos implícitos

As contas de utilizadores existem com o objetivo de identificar diferentes atores perante o sistema e estabelecer que tipos de acesso cada ator tem a diferentes partes do sistema. Ou seja estabelecer um conjunto de permissões (*capabilities*) distintas para cada utilizador.

O simples facto de a conta de utilizador existir pode estabelecer desde logo um conjunto de permissões base, como por exemplo aceder ao sistema (login), aceder para leitura a alguns objetos, ou mesmo criar objetos em pastas temporárias. Estas permissões base advêm de o utilizador pertencer automaticamente a determinados grupos implícitos (***Implicit Groups*** ou ***Special Identities***).

Nos sistemas **Unix** o grupo implícito que se destaca é ***others***. Todos os utilizadores válidos pertencem a este grupo.

Nos sistemas **Windows** existem vários grupos implícitos onde os utilizadores são automaticamente colocados em função de seu estado, por exemplo:

Everyone – todos os utilizadores válidos, incluindo **Guest** e contas de sistema (a conta de utilizador Guest está desativada por omissão, permite o acesso sem password).

Authenticated Users – Todos os utilizadores que se autenticaram através de *username/password*.

Atribuição de permissões

Dependendo do tipo de permissões, elas podem ser associadas aos utilizadores de duas formas:

- **Permissões sobre objetos do sistema**: são associadas ao objeto em causa sob a forma de uma **ACL** (***Access Control List***). A ACL contém uma lista de utilizadores e grupos e para cada um deles as permissões que tem sobre o objeto.
- **Permissões gerais sobre o sistema** (***user rights***): são associadas à conta do utilizador (**ou preferencialmente a uma conta de grupo**) através de um conjunto de atributos (***user rights***). Por exemplo nos sistemas Windows a permissão **Log on locally** permite efetuar o login na consola da máquina.

As permissões, e em particular as permissões gerais sobre o sistema, devem ser atribuídas aos utilizadores indiretamente tornando os utilizadores membros de grupos já existentes para o efeito pretendido, exemplos:

Nos sistemas Linux, os membros do grupo **sudo** possuem a permissão de executar comandos como administrador (***root***). Os membros do grupo **adm** possuem permissão para aceder aos registos (logs) do sistema.

Nos sistemas Windows, os membros do grupo ***Account Operators*** possuem a permissão de gerir contas de utilizadores e grupos.

Permissões sobre objetos do sistema (ACLs)

De acordo com o tipo de sistema de ficheiros e sistema operativo que o está a usar, temos diferentes tipos de ACL associada a cada objeto.

Na realidade o tipo de ACL depende mais do sistema operativo do que do sistema de ficheiros, quando o sistema de ficheiros não contém o tipo de ACL apropriado para o sistema operativo serão realizadas as adaptações (equivalência de permissões) necessárias.

- **ACLs nos sistemas Windows:** derivam das ACL suportadas pelo sistema de ficheiros NTFS (*New Technology File System*), permitem especificar livremente uma lista de utilizadores e grupos e as permissões associadas a cada um.

- **ACLs nos sistemas Unix/Linux:** como base, usam as **ACL POSIX**, mais exatamente do standard POSIX.1e ou POSIX.1-2001 (IEEE Std 1003.1-2001), também designado ***Single UNIX Specification version 3***. Adicionalmente, o sistema operativo Linux e outros da família Unix também suportam ACLs designadas ***extended***, são mais flexíveis do que as anteriores e mais semelhantes às dos sistemas Windows.

POSIX (***Portable Operating System Interface***) é um conjunto de normas IEEE que visa manter alguma compatibilidade entre sistemas operativos de diferentes tipos. As ACL nativas Linux são POSIX, os sistemas Windows proporcionam uma interface POSIX para as ACL NTFS.

ACLs base (POSIX) dos sistema Linux

As ACLs base (***base/minimal ACLs***) dos sistemas Linux apenas permitem a definição de permissões relativas a **3 entidades:**

- O proprietário do objeto (***owner***) – cada objeto tem um único proprietário, inicialmente é o utilizador que o criou. O proprietário de um objeto só pode ser alterado pelo administrador (root).
- O grupo do objeto (***group***) – cada objeto está associado a um grupo de utilizadores, inicialmente é o grupo primário do utilizador que o criou. Um utilizador pode alterar o grupo do objeto se for o proprietário e pertencer ao grupo para o qual quer alterar.
- Grupo implícito ***others*** – todos os utilizadores válidos.

As **permissões efetivas** de um utilizador são o **somatório das permissões** que obtém através das 3 entidades definidas.

Para cada uma das entidades existem 3 permissões: leitura (***read***), escrita (***write***) e execução (***execute***). Embora as designações das permissões sejam razoavelmente claras, o seu significado prático varia ligeiramente com o tipo de objeto a que a ACL está aplicada.

POSIX ACL – read, write and execute permissions

Permissão	Ficheiros	Pastas
read	Permite ler o conteúdo do ficheiro.	Permite listar os nomes dos objetos contidos na pasta. Os atributos e permissões dos objetos só serão acessíveis se tiver acesso à pasta (permissão execute).
write	Permite alterar e eventualmente eliminar o conteúdo do ficheiro.	Permite criar novos objetos na pasta, eliminar objetos existentes e alterar as suas propriedades. Isto só é válido se tiver acesso à pasta (permissão execute).
execute	Permite executar o ficheiro. Se o ficheiro for um ***script*** (programa interpretado), a sua execução exige também a permissão de leitura.	**Acesso à pasta. Para ter acesso a uma pasta é necessário ter a permissão execute sobre todas as pastas desde a raiz do sistema de ficheiros até à pasta pretendida.**

Observações: Para alterar o nome ou outra propriedade de um objeto ou remover o objeto não é necessária a permissão *write* sobre o objeto mas sim sobre a pasta onde o objeto de encontra. Sem permissão *execute* numa pasta todos os objetos nela existentes e subpastas ficam inacessíveis, independentemente de outras permissões.

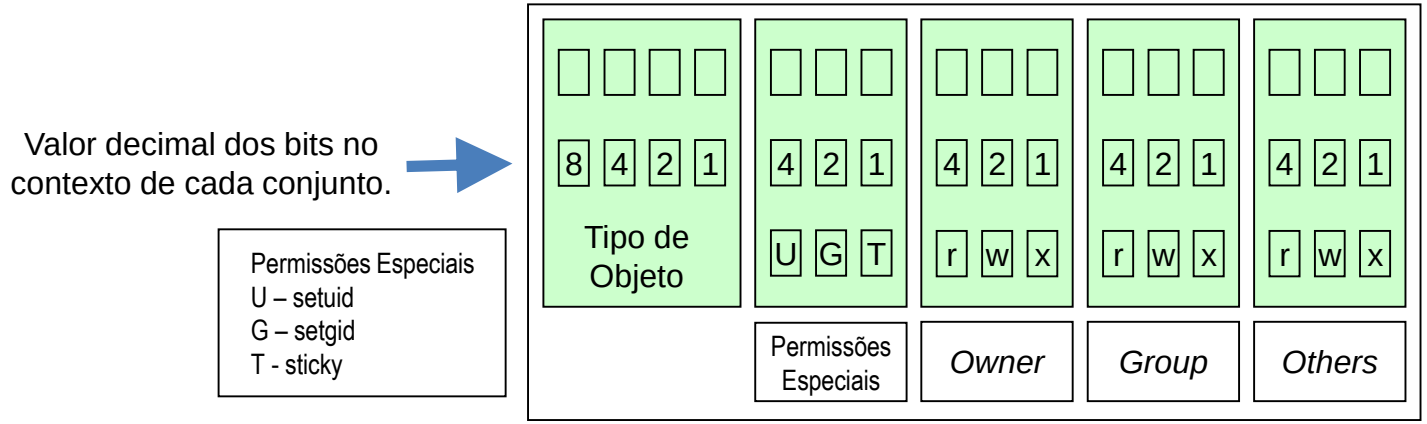
POSIX ACL – setuid, setgid and sticky permissions

As permissões *read*, *write* e execute são definidas para *owner*, *group* e *others* (9 bits). Além destas existem 3 permissões especiais:

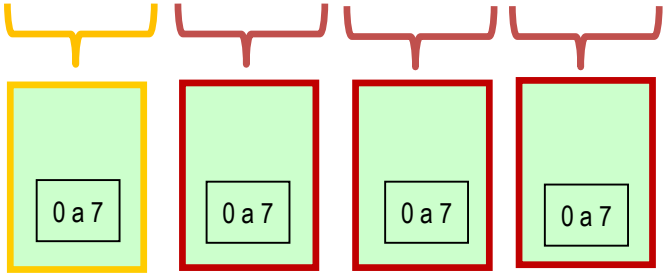
setuid	Se o utilizador possui as permissões necessárias para executar o ficheiro, então o processo criado poderá será executado com as permissões do proprietário do ficheiro, usando a *system-call* **setuid()**. Este expediente é usado em comando como ***passwd, sudo*** e ***ping*** por exemplo. São comandos que têm de realizar operações que não estão acessíveis a utilizadores correntes. Para atribuir esta permissão, o proprietário do ficheiro/comando, tipicamente o utilizador ***root***, tem de confiar totalmente na integridade do comando (só faz o que é suposto).
setgid	Semelhante ao anterior, mas para o grupo. Se o utilizador possui as permissões necessárias para executar o ficheiro, então o processo criado pode ser executado com as permissões do grupo do ficheiro, usando a *system-call* **setgid()**.
sticky	Usado nas pastas temporárias partilhadas nas quais todos os utilizadores podem criar objetos. Faz com que a permissão ***write*** sobre a pasta deixe de ser condição suficiente para remover, renomear ou alterar propriedades de objetos armazenados na pasta, apenas o proprietário do próprio objeto poderá realizar essas operações.

POSIX ACL – Representação octal das permissões

No total existem 12 permissões, estas permissões são armazenadas num conjunto de 16 bits que está associado a cada objeto da seguinte forma:



Cada conjunto de 3 bits de permissões pode ser representada por um símbolo octal (0 a 7), deste modo as permissões podem ser representadas por uma sequência de 4 ou 3 símbolos octais. Tipicamente são usados apenas 3 símbolos omitindo a permissões especiais. Exemplos:



700 – todas as permissões para o proprietário, nenhuma permissão para o grupo ou outros.

644 – permissões read + write para o proprietário, permissão read para o grupo ou outros.

751 – todas as permissões para o proprietário, read e execute para o grupo, execute para outros.

POSIX ACL – permissões iniciais - UMASK

As permissões dos objetos podem ser visualizadas com o comando **ls –l** e alteradas com o comando **chmod.** Relativamente às permissões iniciais de um objeto recém criado dependem da aplicação que cria o objeto e da **UMASK** (*user file-creation mode mask*).

A UMASK é uma máscara de **negação de permissões** em representação octal anteriormente descrita, o seu significado é que os bits existentes na UMASK devem ser **retirados** das permissões dos objetos criados.

Uma das UMASK mais vulgarmente utilizadas é 022 ou 0022 indica que nos objetos criados, as permissões write para grupo e others devem ser sempre retiradas. Uma UMASK que garante a total privacidade sobre os objetos criados é 0077.

Na linha de comandos (Shell) a UMASK pode ser definida usando o comando **umask**, cada processo/aplicação tem uma UMASK independente, quando um processo é criado (*fork*) o processo filho herda a UMASK do processo pai, mas depois pode alterá-la.

Nos scripts de login do sistema (executados no ambiente do utilizador após o login) o administrador inclui habitualmente o comando:

umask 0022

Se o utilizador não alterar esta UMASK, ela será herdada pela Shell e por todas as aplicações lançadas a partir dela.

Permissões Windows (permissões NTFS)

As permissões usadas nos sistemas Windows Server derivam do sistema de ficheiros NTFS, podendo definir permissões para nenhuma entidade ou para um número variável de entidades (*Access Control Entries* – **ACE**).

Para cada ACE (tipicamente um utilizador ou um grupo, mas também pode ser por exemplo um processo), a ACL define um conjunto de permissões. Enquanto nas ACL POSIX existem apenas 3 permissões (*read*, *write* e *execute*) mais 3 permissões especiais, as diferentes permissões que podem estar associadas a um ACE numa ACL Windows são bastante detalhadas e numerosas (***special permissions***).

Para facilitar o seu manuseamento são definidos 6 conjuntos de permissões básicas (***basic permissions***):

- Full Control
- Modify
- Read & Execute
- List Folder Contents (folders only)
- Read
- Write

Windows – basic permissions and special permissions

	Basic permissions					
Special permissions	**Full Control**	**Modify**	**Read and Execute**	**List Folder Contents (folders only)**	**Read**	**Write**
Traverse Folder/Execute File	YES	YES	YES	YES		
List Folder/Read Data	YES	YES	YES	YES	YES	
Read Attributes	YES	YES	YES	YES	YES	
Read Extended Attributes	YES	YES	YES	YES	YES	
Create Files/Write Data	YES	YES				YES
Create Folders/Append Data	YES	YES				YES
Write Attributes	YES	YES				YES
Write Extended Attributes	YES	YES				YES
Delete Subfolders and Files	YES					
Delete	YES	YES				
Read Permissions	YES	YES	YES	YES	YES	YES
Change Permissions	YES					
Take Ownership	YES					
Synchronize	YES	YES	YES	YES	YES	YES

Permissões Windows – herança e negação de permissões

Nos sistemas Windows as permissões são herdadas automaticamente. Se uma pasta possui determinadas permissões, os objetos que são criados na pasta herdam as permissões da pasta (***inherited permissions***). As permissões herdadas não constam da ACL do objeto, mas possuem efeito prático idêntico.

Nas ACL Windows cada permissão pode ser dada (***Allow***) ou retirada (***Deny***) as permissões efetivas (finais) de um utilizador são definidas pelo somatório das permissões dadas através dos vários ACE suprimidas das permissões retiradas através dos vários ACE. **As permissões *Deny* têm precedência sobre as permissões *Allow*.**

No entanto, **as permissões explícitas (definidas na ACL do objeto) têm precedência sobre as permissões herdadas.** Assim se o *Deny* de uma permissão é herdado e na ACL do objeto a mesma permissão é dada (*Allow*) então essa permissão será efetiva.

Precedências: Explicit Deny - Explicit Allow - Inherited Deny - Inherited Allow

Um outro aspeto importante é que as ACL Windows além de poderem estar associada a objetos do sistema de ficheiros como ficheiros e pastas também podem estar associadas a outros tipos de objetos do sistema operativo como por exemplo processos e serviços.

ACLs Windows – DACL e SACL

Nos sistemas Windows Server atuais existem dois tipos da ACL com objetivos diferentes:

DACL (*Discretionary Access Control List*) – destina-se a controlar o acesso ao objeto definindo a permissões de cada ACE. Cada permissão pode ser dada (*Allow*) ou retirada (*Deny*). Para gerir este tipo de ACL basta ter as permissões adequadas sobre o objeto.

SACL (*System Access Control List*) – destina-se a registar os acessos (*audit reports*). Para cada ACE contém uma lista de permissões que quando são aplicadas ficam registadas em log. Para cada ACE e para cada permissão podem ser registas as tentativas de acesso bem sucedidas (*Success*) e/ou falhadas (*Fail*). As SACL só podem ser geridas pelos administradores.

Em Windows todos os objetos possuem um proprietário (***owner***), que pode tratar-se de um utilizador ou um grupo. Inicialmente o proprietário é o utilizador que cria o objeto, mas pode ser alterado por que tiver a permissão ***Take Ownership***, ou possuir a permissão geral (*user right*) ***Take ownership of files or other objects***, como acontece com o grupo **Administrators**.

Independentemente das permissões nas ACL em Windows o proprietário pode sempre alterar as permissões sobre o objeto.

Pasta pessoal (home directory)

O conceito de pasta pessoal é comum à maioria dos sistemas operativos *multiuser*, normalmente a identificação da ***home directory*** é um atributo armazenado na conta de utilizador, cada utilizador deve ter uma pasta pessoal independente.

Para utilizadores correntes deve tratar-se de uma pasta sobre a qual o utilizador tem permissões totais e mais nenhum utilizador tem permissões de alteração. Tipicamente o utilizador é o proprietário da respetiva ***home directory***.

Sendo o proprietário e tendo permissões totais, cabe ao próprio utilizador gerir o seu conteúdo e controlar as permissões de acesso ao mesmo.

Num ambiente de rede a ***home directory*** deve estar disponível através da rede.

Linux: se existe um conjunto de servidores ligados a um sistema de autenticação centralizado, então cada utilizador terá o mesmo UID em todos os sistemas, nesse caso pode usar-se o **NFS** (*Network File System*) para montar as pastas pessoais em todos os servidores. O NFS mantém os proprietários originais dos objetos e as permissões.

Domínio Windows: as pastas pessoais podem ser partilhadas através da rede a partir de um *Windows Server* que pertença ao domínio.

Domínios Windows – *User Profile*

Nos postos de trabalho Windows existe um outro conceito designado ***User Profile***, contudo este não é nem a conta de utilizador nem a ***home directory*** do utilizador.

O *User Profile* é um conceito subjacente aos postos de trabalho Windows, contém configurações de *registry* pessoais do utilizador (ficheiro NTuser.dat) e itens pessoais a colocar no *desktop*, *start menu*, etc..

Em qualquer posto de trabalho existe um ***Default User Profile***, quando um utilizador efetua o login no posto de trabalho, se não tem um *User Profile* pessoal será criado um por cópia do *Default User Profile* local. Alterações de configurações e personalizações do ambiente feitas pelo utilizador ficam guardadas no seu *User Profile* pessoal local. O *User Profile* está normalmente contido numa pasta com o nome do utilizador.

Quando o posto de trabalho Windows é um ***domain member***, isso significa que o utilizador não faz login no posto de trabalho local mas sim no domínio Windows, neste caso diversas opções se colocam relativamente ao funcionamento do *User Profile*.

Num domínio Windows podemos ter *User Profiles* definidos no próprio domínio, para esse efeito é necessário definir na conta de utilizador o atributo *User Profile*. Este atributo deve conter a identificação de uma pasta partilhada na rede onde se encontra o perfil.

Domínios Windows – *User Profile* - cenários

A. O atributo *User Profile* não está definido na conta de utilizador, ou seja não existe. Neste caso pode existir um ***Default User Profile*** de domínio (pasta ***Default User***) na partilha de rede **NETLOGON** do controlador de domínio (a partilha **NETLOGON** é replicada automaticamente entre todos os DC do domínio).

Se no posto de trabalho não existe um *User Profile* pessoal será criado um por cópia do ***Default User Profile*** do controlador de domínio, ou, se este não existir, por cópia do ***Default User Profile*** local existente no posto de trabalho.

Neste caso as alterações realizada ao perfil são meramente locais, persistem no posto de trabalho, mas se o utilizador entrar noutro posto de trabalho terá um novo perfil.

B. O atributo *User Profile* está definido na conta de utilizador, apontando para uma pasta partilhada por um servidor do domínio. Na pasta referida encontra-se um ficheiro **NTuser.dat**, então estamos perante um ***Roaming User Profile***. Sendo um *Roaming Profile*, ao efetuar o *login* o perfil é copiado da partilha de rede para o posto de trabalho, ao efetuar o *logout* o perfil é copiado do posto de trabalho para a partilha de rede. Obviamente que o utilizador terá de ter as permissões necessárias sobre a partilha de rede. Com este tipo de configuração o utilizador tem sempre o mesmo perfil personalizado independentemente do posto de trabalho em que efetua o login.

Domínios Windows – *User Profile* - cenários

C. O atributo *User Profile* está definido na conta de utilizador, apontando para uma pasta partilhada por um servidor do domínio. Na pasta referida encontra-se um ficheiro **NTuser.man (**e não ***Ntuser.dat*)**, então estamos perante um ***Mandatory User Profile***. Ao efetuar o login o perfil é copiado da partilha de rede para o posto de trabalho, ao efetuar o *logout* **o perfil local é eliminado**. Neste caso o atributo *User Profile* pode ser igual para todos os utilizadores e a partilha terá apenas permissão de leitura. Com este tipo de *profile* o utilizador sempre que faz login tem o mesmo perfil (copiado da partilha), mesmo em logins sucessivos no mesmo posto de trabalho.

Adicionalmente, se o nome da pasta do ***Mandatory User Profile*** termina em **.man** será um ***Super-mandatory User Profile***, a diferença é que se a partilha de rede prevista não está disponível o utilizador é impedido de todo de efetuar o login. Com um ***Mandatory User Profile*** normal ou um ***Roaming User Profile***, se a partilha não está disponível é usado o cenário **A.**

As alternativas ***Mandatory User Profile*** e a utilização de um ***Default User Profile*** de domínio têm algumas semelhanças, a diferença é que no segundo caso o perfil só é copiado do servidor se não existir localmente o perfil do utilizador, no primeiro caso é sempre copiado.

Controlo de quotas

Desde o momento em que é atribuída a permissão de escrita aos utilizadores em pastas residentes em sistemas de ficheiros (partições/volumes) controlados pelo administrador torna-se fundamental manter controlo sobre o espaço usado por cada utilizador. É essa a missão do controlo de quotas.

Sem controlo de quotas um utilizador poderia colocar dados na pasta até esgotar o espaço disponível na correspondente partição/volume. Nessa altura os outros utilizadores e o próprio sistema deixariam de poder escrever em outras pastas existentes na mesma partição/volume.

A quota de cada utilizador consiste no somatório do espaço ocupado na partição/volume por todos os objetos que são propriedade desse utilizador. O controlo de quotas permite definir limites de quota, as unidades usadas podem ser várias: Kb, Mb, Gb. No Linux a unidade usada é o bloco cuja dimensão pode varia entre distribuições e/ou *file systems*.

Normalmente são definidos dois limites de quota: ***soft limit*** e ***hard limit***, devendo o segundo ser superior ao primeiro. Quando um utilizador atinge o ***soft limit*** continua a poder usar mais espaço, mas é avisado de que está a exceder o limite de quota. Atingido o ***hard limit***, deixa de poder escrever. Pode também ser definido um ***grace period*** (tempo máximo - normalmente expresso em dias - durante o qual é aceitável exceder o ***soft limit***, ultrapassado este tempo o utilizador deixa de poder usar qualquer espaço acima do ***soft limit***).

Controlo de quotas – Linux e Windows Server

Linux: é possível fazer o controlo de quotas para utilizadores e/ou grupos (a quota de grupo representa o somatório dos espaços ocupados por todos os objetos associados ao grupo). As quotas em Linux são definidas para cada partição, não é de todo possível controlar as quotas de uma pasta individual. Além de quotas e limites de quotas relativas a espaço ocupado, em Linux é também possível definir quotas de *i-nodes*, ou seja o número de objetos.

Windows Server: existem dois sistemas de controlo de quotas independentes nas versões atuais do Windows Server: ***Disk Quotas*** e ***Directory Quotas***.

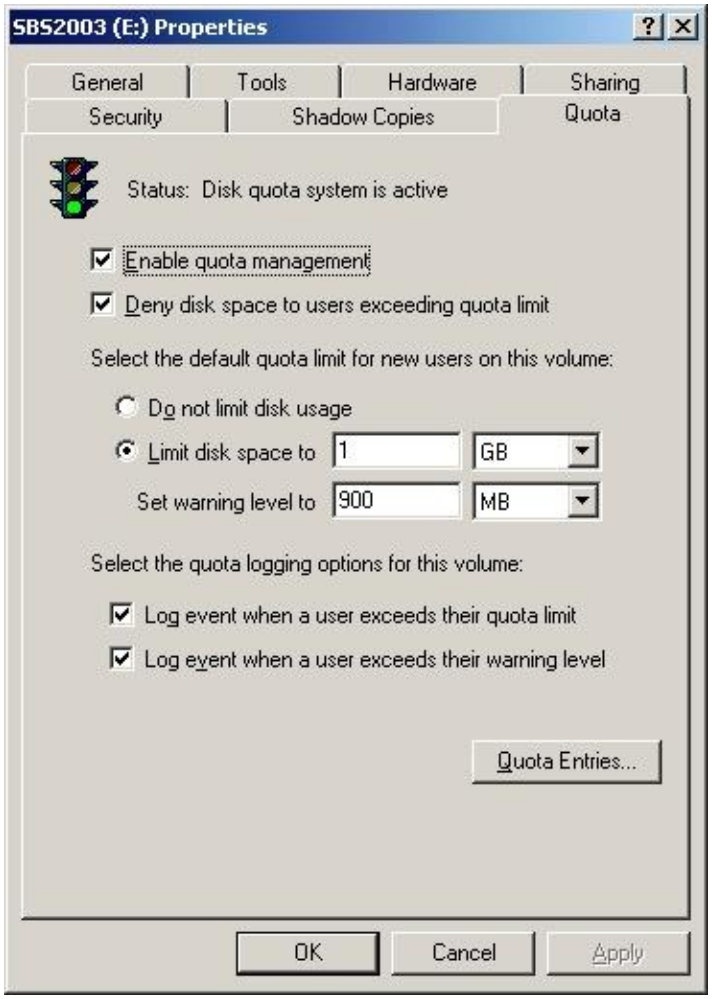
- ***Disk Quotas*** – introduzidas no *Windows 2000*, permitem controlar as quotas de utilizadores individuais na globalidade de um volume que esteja associado a uma letra de drive, novamente, não é de todo possível controlar as quotas de uma pasta individual. Não são suportadas quotas de grupo, no Windows os objetos não estão associados a grupos, apenas a utilizadores (***owner***).
- ***Directory Quotas*** – introduzidas no *Windows Server 2003 R2*, permitem definir limites de quota para uma pasta, no entanto não se trata de uma quota de utilizadores individuais, mas sim de todos os utilizadores. Isto significa que um utilizador pode ocupar toda a quota disponível ficando os outros utilizadores sem nenhum espaço.

Controlo de quotas – Windows Server

Como é evidente as ***Directory Quotas*** não vão de encontro ao objetivo de controlar o espaço ocupado por cada utilizador. Numa perspetiva simplista apenas permitem gerar alertas quando o nível de ocupação de espaço de uma pasta pelo conjunto dos utilizadores atinge um dado limite.

As ***Disk Quotas*** são sempre orientadas por letra de drive (associada a um volume). É possível definir limites gerais aplicados a todos os utilizadores, para considerar o espaço ocupado previamente é necessário a execução do comando **dirquota**. Também é possível aplicar limites de quota utilizador a utilizador (*Quota Entries*).

Uma **partilha de rede** de uma pasta residente num volume com ***Disk Quotas*** também fica sujeita às quotas definidas. No entanto, para poderem visualizar corretamente os valores de quota os utilizadores terão de mapear uma letra de drive para a partilha de rede.



Controlo de quotas – Linux

O controlo de quotas é realizado pelo ***kernel*** durante as operações de manuseamento do sistema de ficheiros, o ***kernel*** regista as alterações realizadas pelo utilizador e em função das variações atualiza o registo de quotas do utilizador ou grupo.

A contabilização de quotas pelo ***kernel*** não é realizado em valor absoluto, apenas são contabilizadas as alterações. Supõe-se que o ponto de partida é um registo de quotas correto.

O primeiro passo para ativar o controlo de quotas é montar a partição com as opções **usrquota** e/ou **grpquota**, para isso é necessário editar o ficheiro **/etc/fstab** e montar novamente a partição (*remount*).

Tendo a partição montada com as opções necessárias, é necessário efetuar a contabilização inicial usando o comando **quotacheck**. Este comando vai percorrer todo o sistema de ficheiros da partição e contabilizar o espaço que está a ser ocupado por cada utilizador e/ou grupo. Na raiz da partição serão criados os ficheiros **aquota.user** e/ou **aquota.group** (ou dependendo da versão **quota.user** e/ou **quota.group**) contendo as quotas contabilizadas.

Após concluir a contabilização e ter criado os referidos ficheiros, é executado o comando **quotaon** que instrui o ***kernel*** para começar a monitorizar todas as operações de escrita e manter os ficheiros atualizados em conformidade. Se necessário recalcular o espaço, deve ser usado o comando **quotaoff** e contabilizado de novo.

Controlo de quotas – Linux

Contudo e a menos que haja uma falha grave no disco, a contabilização inicial de quotas (comando **quotacheck**) nunca mais será necessária. Depois de realizada a primeira vez o controlo é passado ao ***kernel*** que se encarrega de manter o registo atualizado.

Inicialmente todos os utilizadores têm limites de quota zero, o que significa que não têm limite de quota.

Os limites de quota têm de ser definidos utilizador a utilizado e grupo a grupo, por exemplo usando os comandos **edquota** ou **setquota**.

O comando edquota invoca um editor de texto e permite a edição dos valores de limite de quota, o comando setquota recebe os limites diretamente como argumentos na linha de comando.

Em qualquer caso existem sempre quatro limites de quota na seguinte ordem:

Block soft limit – limite soft de espaço em unidades de 1 Kb*

Block hard limit – limite hard de espaço em unidades de 1 Kb*

I-nodes soft limit – limite soft de número de objetos

I-nodes hard limit – limite hard de número de objetos

* Por omissão

Controlo de quotas – Linux

Sintaxes básicas do comando **setquota** para quotas de um utilizador com nome de login **username:**

```
setquota  username <block-softlimit> <block-hardlimit> <inode-softlimit> <inode-hardlimit> <filesystem>
setquota -p protousername username <filesystem>
```

Os quatro limites de cota são argumentos obrigatórios, se não se quiser impor um dos limites, deve-se usar o valor zero no seu lugar. A segunda versão da sintaxe apresentada permite copiar as definições de limites de quotas do utilizador **protousername** para o utilizador **username.**

O comando **quota** permite ao administrador visualizar o estado atual das quotas de um qualquer utilizador. Permite também a um utilizador visualizar o estado atual apenas das suas próprias quotas. Como exemplo (para um administrador a ver a quota do utilizador *i999999*):

```
root@server# quota i999999
Disk quotas for user i999999 (uid 2687):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/sdb1    4228  524288  525000             66       0       0
```

O administrador (*root*) também pode obter um relatório do estado das quotas de todos os utilizadores usando o comando **repquota.**

Login scripts

Dependendo da contexto em que o utilizador efetua a entrada no sistema (*login*) poderão ser executados automaticamente programas com diversas finalidades. Apesar de serem normalmente designados de **login scripts**, podem ser qualquer tipo de programa executável no ambiente em questão.

Linux: um dos atributos das contas de utilizador Unix é o programa inicial que é executado quando o utilizador efetua o *login* num terminal. Uma vez que normalmente se trata se uma Shell (interpretador de comandos) este atributo também é muitas vezes designado de ***Login Shell***. Existem vários interpretadores de comandos alternativos, alguns dos mais habituais são o **bash** (*Bourne-again Shell*) e a **csh** (*C Shell*), cada uma delas executa automaticamente vários scripts após o login do utilizador (ficheiros de arranque da Shell).

Windows: as contas de utilizador contêm o atributo ***Logon Script***. Este atributo pode conter o nome de um ficheiro executável no ambiente Windows, script ou não. Se definido, o ficheiro será executado após o login do utilizador.

No contexto do login num domínio este ficheiro deve encontrar-se na partilha de rede **NETLOGON** pois esta partilha é replicada automaticamente entre todos os DC do domínio.

Linux - ficheiros de arranque da Shell

A Shell é o programa inicial que é normalmente executado quando um utilizador entra no sistema através de um terminal. Depois de concluído o *login*, todas as atividades do utilizador são desencadeadas a partir da Shell, incluindo o lançamento de novas aplicações.

O carregamento da Shell inicial é por isso a altura ideal para executar operações de configuração do ambiente do utilizador e aplicações que se pretenda que sejam automaticamente executadas durante o login.

Cada tipo de Shell usa ficheiros de arranque distintos, a Shell tradicional **/bin/sh** executa os ficheiros **/etc/profile** e de seguida **.profile** na *homedir* do utilizador (**~/.profile**).

A **/bin/sh** é na realidade simulada pela BASH, quando invocada como BASH (**/bin/bash**) usa a seguinte sequência de ficheiros de arranque: **/etc/profile ; ~/.bash_profile ; ~/.bash_login ; ~/.profile.**

A C Shell, **/bin/csh** ou **/bin/csh**, executa a seguinte sequência de ficheiros de arranque: **/etc/csh.cshrc ; /etc/csh.login ; ~/.cshrc ; ~/.login.**

Uma característica a reter é que os ficheiros da área do utilizador são executados em último lugar, por isso podem alterar o que foi feito nos ficheiros de sistema (**/etc/***), embora apenas para esse utilizador.

Linux – exemplo de ficheiro /etc/profile

```
bash-3.00$ cat /etc/profile
# /etc/profile -*- Mode: shell-script -*-
# (c) MandrakeSoft
if ! echo ${PATH} |grep -q /usr/X11R6/bin ; then
    PATH="$PATH:/usr/X11R6/bin"
fi
if [ "$UID" -ge 500 ] && ! echo ${PATH} |grep -q /usr/games ; then
    PATH=$PATH:/usr/games
fi
umask 022
USER=`id -un`
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"
HOSTNAME=`/bin/hostname`
export PATH USER LOGNAME MAIL HOSTNAME

for i in /etc/profile.d/*.sh ; do
        if [ -x $i ]; then
                . $i
        fi
done
unset i
```

Definição da UMASK

Definição da variáveis de ambiente

Variáveis a exportar

Pasta com mais ficheiros a executar

Note-se que este ficheiro apenas é executado no login para utilizadores cuja Shell definida na conta de utilizador é **/bin/bash** ou **/bin/sh.** Para utilizadores da C Shell ações semelhantes terão de ser definidas nos ficheiros **/etc/csh.cshrc** ou **/etc/csh.login.**

Symbolic links

Uma ligação simbólica é um tipo de objeto que contém **um apontador para outro objeto** do sistema de ficheiros. É suportado por alguns tipos de sistemas de ficheiros e sistemas operativos.

As ligações simbólicas são uma ferramenta importante para o administrador, uma vez que permitem que pastas e objetos possam ser acedidos através de locais e nomes de objetos alternativos. Por outras palavras permitem fazer com que objetos aparentem ter localizações e nomes que não são os reais.

Sob o ponto de vista dos utilizadores e aplicações, as ligações simbólicas são transparentes, quando uma aplicação acede a uma ligação simbólica ela aparenta ter as propriedades do objeto apontado (***target***). No entanto, **remover a ligação simbólica não remove o objeto apontado**, apenas a ligação simbólica.

Por exemplo se num sistema Windows existe uma pasta **E:\Docs**, podemos criar uma ligação simbólica **C:\Documents** que aponta para **E:\Docs** (*C:\Documents* **->** *E:\Docs*).

Quando os utilizadores a aplicações acedem a **C:\Documents** na realidade estão a aceder a **E:\Docs**. As propriedades aparentes de **C:\Documents** são as propriedades de **E:\Docs**.

Sistemas Windows - *Symbolic links*

Nos sistemas Windows as ligações simbólicas são suportadas sobre sistemas de ficheiros NTFS a partir do Windows Vista (Workstation) e Windows Server 2008.

Apenas os utilizadores com a permissão geral (*user right*) ***Create Symbolic Links*** podem criar ligações simbólicas, inicialmente apenas o grupo *Administrators* possui essa permissão.

O ***File Explorer*** do Windows não permite criar ligações simbólicas, sob o ponto de vista do *File Explorer* as ligações simbólicas são interpretadas como atalhos (***Shortcuts***). Não devemos no entanto confundir ligações simbólicas com atalhos do *File Explorer*. Os atalhos do *File Explorer* são ficheiros normais que apenas são reconhecidos pelo *File Explorer* e não por outras aplicações.

Para criar ligações simbólicas no Windows deve ser usado o comando **mklink:**

mklink [/D] SYMLINK TARGET

Se o **TARGET** for uma pasta deve ser usada a opção **/D**, caso contrário obtém-se algo incoerente: um ficheiro que aponta para uma pasta.

SYMLINK e **TARGET** podem ou não encontrar-se no mesmo sistema de ficheiros, volume ou partição. **TARGET** também pode ser um objeto residente numa partilha de rede.

Sistemas Unix/Linux - *Symbolic links*

Nos sistemas Unix/Linux as ligações simbólicas podem ser criadas por qualquer utilizador que tenha as permissões necessárias na pasta onde a ligação vai ser criada. Para isso pode usar o comando **ln:**

ln –s TARGET [SYMLINK]

Relativamente ao comando **mklink** do Windows destaca-se que **TARGET** e **SYMLINK** estão em ordem inversa, também não é necessário indicar se o **TARGET** é uma pasta ou não. Se **SYMLINK** for omitido será criado na pasta corrente uma ligação simbólica com o mesmo nome do **TARGET.**

A ligação simbólica tem todas as permissões ativas, mas isso é ilusório, as permissões reais são as do objeto apontado. Novamente, remover a ligação simbólica não tem qualquer efeito sobre o **TARGET.**

Se o **TARGET** deixar de existir (for removido) as ligações simbólicas a ele mantêm-se mas ficam inoperacionais (quebradas).

```
andre@vsrv24:~/TESTE$ ln -s /lib myliblink
andre@vsrv24:~/TESTE$ ln -s /lib2 myliblink2
andre@vsrv24:~/TESTE$ ln -s /etc/passwd
andre@vsrv24:~/TESTE$ ls -l
total 0
lrwxrwxrwx 1 andre profs  4 Set 21 10:08 myliblink -> /lib
lrwxrwxrwx 1 andre profs  5 Set 21 10:08 myliblink2 -> /lib2
lrwxrwxrwx 1 andre profs 11 Set 21  2017 passwd -> /etc/passwd
andre@vsrv24:~/TESTE$
```

Administração de Sistemas (ASIST)

Aula Teórico Prática 04

Gestão da configuração de rede.
Servidores DHCP. Servidores DNS.

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Configuração de interfaces de rede em servidores

A configuração de uma interface de rede pode ser efetuada autonomamente pelo próprio sistema ou recorrer a serviços externos. O serviço externo mais usado pera este efeito é o **DHCP** (*Dynamic Host Configuration Protocol*), no caso da configuração IPv6 também se pode usar **SLAAC** (*Stateless Address Auto Configuration*), eventualmente complementada com **DHCP**.

Como normalmente um servidor presta serviços de rede, deve ter endereços fixos conhecidos dos clientes. Isso exclui a utilização de SLAAC, mas pode ser usado DHCP para atribuir endereços IPv4 e IPv6 fixos.

No entanto, sempre que possível, um servidor deve ser independente de serviços externos. Por essa razão, normalmente opta-se por configurações de rede autónomas em que o sistema possui todos os dados para configurar as suas interfaces de rede. Deste modo, mesmo que os serviços DHCP da rede não estejam operacionais, o servidor pode funcionar normalmente.

Sob o ponto de vista dos protocolos IPv4 e IPv6, para configurar uma interface de rede bastam dois dados: **endereço de nó** e **prefixo de rede** (máscara de rede).

Com estes dois elementos é possível deduzir qual é o **endereço da rede**, e no caso do IPv4, qual é o **endereço de *broadcast***.

Nomes das interfaces de rede

No sistema operativo cada interface de rede (**NIC** – *Network Interface Controller*) é identificada por um nome. Se o servidor possui várias interfaces de rede é fundamental identificar a correspondência entre os nomes e as interfaces físicas, isso poderá não ser trivial, especialmente se o hardware das interfaces é igual.

Linux: tradicionalmente as interfaces de rede Ethernet eram identificadas por **eth** seguido de um sufixo numérico (***eth0***, ***eth1***, ***eth2***, …). Nas atuais versões do *kernel*, as interfaces de rede Ethernet são identificadas por **en** seguido de um sufixo complexo que representa o seu posicionamento sob o ponto de vista do hardware. Por exemplo **ens**… significa que se trata de um interface ligada ao barramento PCI externo, **eno**… significa que se trata de uma interface *on-board*. A restante parte do prefixo também está relacionada com o hardware.

Windows: as interfaces de rede são identificadas com nomes que tentam representar o seu tipo, por exemplo, **Ethernet**. Existindo várias interfaces do mesmo tipo, a primeira mantém o nome e as seguintes recebem um sufixo numérico a partir de 2. Por exemplo num servidor com 3 interfaces Ethernet: **Ethernet**, **Ethernet 2** e **Ethernet 3**. A configuração IPv4 e IPv6 das interfaces de um Windows Server em modo gráfico faz-se facilmente recorrendo ao ***Control Panel***.

Linux - configuração das interfaces de rede

Os comandos **ifconfig** e **ip** podem ser usados para visualizar o estado atual das interfaces de rede e alterar as configurações IP das mesmas. Note-se no entanto que as alterações realizadas por esta via não persistem após um novo ***boot*** da máquina (ao contrário da alterações realizadas no Windows via ***Control Panel***).

As configurações de rede aplicadas durante o ***boot*** da máquina residem em ficheiros de configuração que dependem da distribuição e até da versão; no Ubuntu 16.04 estão no ficheiro **/etc/network/interfaces** e no Ubuntu 18.04 como as interfaces de rede são geridas pelo **Netplan** estão no ficheiro **/etc/netplan/50-cloud-init.yaml**.

O comando **ifconfig** sem argumentos apresenta uma lista das interfaces de rede que estão ativas (*up*) e as respetivas configurações. O comando **ip addr show** apresenta uma lista de todas as interfaces de rede e as respetivas configurações (estejam *up* ou *down*).

O comando **ip** deve ser usado em lugar do comando **ifconfig** que se prevê seja descontinuado no futuro.

Exemplo básico de configuração de uma nova interface com um endereço IPv6 seguido da ativação da interface (admitindo que estava ***down***):

```
ip address add 2001:0db8:85a3::0370:7334/64 dev ens160

ip link set dev ens160 up
```

Default gateway and routing

O endereço de nó e o prefixo da rede são suficientes para conseguir comunicar com outros nós das redes IP a que o servidor está diretamente ligado, no entanto para comunicar com nós de outras redes IP é necessário recorrer a ***routers***.

No cenário mais simples, nas redes IP a que o servidor está ligado existe apenas um *router*, nesse caso basta defini-lo como ***default gateway***. No caso dos sistemas Windows o ***default gateway*** pode ser definido juntamente com as configurações IP. Num sistema Linux terá de ser manualmente adicionado à ***routing table*** do sistema ou definido na configuração da rede.

Como resultado, todos os pacotes destinados a redes IP remotas (a que o servidor não está diretamente ligado) serão transmitidos para o ***default gateway***.

As configurações do ***default gateway*** IPv4 e IPv6 são totalmente independentes, assim se o servidor utiliza IP4 e IPv6 (*dual-stack*) existirá um ***default gateway*** IPv4 e um ***default gateway*** IPv6; podem ser o mesmo, mas os endereços serão certamente diferentes.

Num cenário em que existem vários routers nas redes IP a que o servidor está ligado, a ***routing table*** terá de ser definida em conformidade.

Linux – gestão da *routing table*

O tradicional comando **route** prevê-se que seja descontinuado e deve ser usado o comando **ip [-6] route** em seu lugar. A opção **-6** é usada para gerir a *routing table* IPv6, caso seja omissa atua sobre a *routing table* IPv4.

Podemos visualizar as tabelas através do comando **ip [-6] route show**

Novamente, as alterações de configuração não persistem após um novo ***boot*** da máquina, para isso terão de ser definidas em ficheiros de configuração apropriados dependentes da distribuição.

Exemplos:

- Definição do ***default gateway*** em IPv4 e IPv6 na NIC ens160:

```
ip route add 0.0.0.0/0 via 10.20.0.1 dev ens160
ip -6 route add ::/0 via fd1e:2bae:c6fd:627a::55 dev ens192 protocol static
```

- Adição de uma regra específica em IPv4 e IPv6 na NIC ens160:

```
ip route add 172.16.5.0/24 via 10.20.0.1 dev ens160
ip -6 route add fd1e:2bae::/64 via fd1e:2bae:c6::5 dev ens192 protocol static
```

- Remoção de uma regra em IPv4 e IPv6 na NIC ens160:

```
ip route del 172.16.5.0/24 via 10.20.0.1 dev ens160
ip -6 route del ::/0 via fd1e:2bae:c6fd:627a::55 dev ens192 protocol static
```

Windows – gestão da *routing table*

A gestão da ***routing table*** no que vai além da definição do ***default gateway*** tem de recorrer ao comando **route** na linha de comandos. O comando **route print** apresenta as tabelas atuais (IPv4 e IPv6).

As alterações de configuração são guardadas de forma persistente pelo sistema operativo.

Em função dos endereços fornecidos o comando deduz se se trata de endereços IPv4 ou IPv6, determinando igualmente a interface mais apropriada.

Exemplos (equivalentes aos anteriores em Linux):

- Definição do ***default gateway*** em IPv4 e IPv6:

```
route add 0.0.0.0/0 10.20.0.1
route add ::/0 fd1e:2bae:c6fd:627a::55
```

- Adição de uma regra específica em IPv4 e IPv6:

```
route add 172.16.5.0/24 10.20.0.1
route add fd1e:2bae::/64 fd1e:2bae:c6::5
```

- Remoção de uma regra em IPv4 e IPv6:

```
route delete 172.16.5.0/24
route delete ::/0
```

Caso pretendido pode-se indicar a interface de rede pretendida com a opção **–F**.

Configuração do cliente DNS (resolução de nomes)

Para ser possível a utilização de nomes de máquinas do sistema DNS em lugar de endereços IP é necessário definir quais são **os endereços dos servidores de nomes DNS a utilizar.**

O sistema DNS só resolve nomes qualificados (que incluem o nome de domínio completo), para resolver nomes não qualificados podem ser definidos vários nomes de domínio que serão usados para completar os nomes não qualificados antes de os enviar ao servidor DNS. Num sistema que utiliza DHCP estas informações são normalmente transmitidas por essa via, não sendo usado o DHCP terão de ser definidas manualmente.

Windows: a configuração de cliente DNS é realizada juntamente com as definições dos endereços IP das interfaces de rede. Embora na janela base apenas seja permitida a definição de um ou dois servidores DNS, nas opções avançadas é possível acrescentar mais. Igualmente nas opções avançadas é possível definir um conjunto de nomes de domínio a utilizar para completar nomes não qualificados.

Linux: a configuração do cliente DNS (*resolver*) está tradicionalmente guardada no ficheiro **/etc/resolv.conf**. No entanto na maioria das distribuições atuais este ficheiro é gerado sempre que a máquina arranca, para esse efeito usa-se informação residente em outros ficheiros de configuração (dependem da distribuição). No caso do Ubuntu 16.04 ou 18.04 esta informação deve ser definida no ficheiro de configuração da(s) interface(s) de rede.

Interfaces de rede com vários endereços IP

Uma mesma interface de rede por ter vários endereços IP atribuídos, podendo estes pertencer a redes IP diferentes ou podem pertencer à mesma rede IP.

É possível ter a funcionar sobre uma mesma rede física várias redes IP diferentes. Os nós IP que apesar de estarem ligados à mesma rede física utilizam endereços pertencentes a redes IP diferentes só poderão comunicar entre si através de um *router*. Pode-se evitar a necessidade de recorrer ao *router* se foram atribuídos à interface endereços em cada uma das redes.

Podemos também atribuir a uma mesma interface vários endereços IP pertencentes à mesma rede IP, neste caso o objetivo é normalmente criar vários servidores aparentes (virtuais).

Se o servidor tem vários endereços IP (tipicamente mapeados para nomes DNS distintos), sob o ponto de vista dos clientes vai aparentar tratar-se de um conjunto de vários servidores distintos.

Para que este tipo de configuração se torne interessante tem de ser complementada com uma configuração de serviços de rede que seja dependente do endereço de destino dos pedidos dos clientes. Por exemplo o servidor HTTP Apache permite definir sites virtuais distintos em função desse critério.

Windows: nas opções avançadas de propriedades IPv4 e IPv6 da interface de rede, outros endereços IP podem ser adicionados ao que foi definido na janela base. Neste caso, para endereços pertencentes a diferentes redes não será possível definir um default gateway para cada uma delas, essa configuração terá de ser realizada através do comando **route**.

Linux: através do comando **ip addr add** … podemos ser adicionados vários endereços à interface para além do primeiro (se bem que apenas persistam até ao próximo *boot* do sistema).

Ao adicionar um endereço podemos também definir um nome de interface virtual, por questões de compatibilidade com os *Linux-2.0 net aliases*, o nome deve começar pelo nome da interface física, seguido de dois pontos*, o resto pode ser definido de acordo com as preferências do administrador. Exemplo:

```
root@uvm:~# ip addr add 10.5.10.200/16 broadcast + dev ens160 label ens160:rede1
root@uvm:~# ifconfig ens160:rede1
ens160:rede1 Link encap:Ethernet  Endereço de HW 00:50:56:a7:10:03
      inet end.: 10.5.10.200  Bcast:10.5.255.255  Masc:255.255.0.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Métrica:1
```

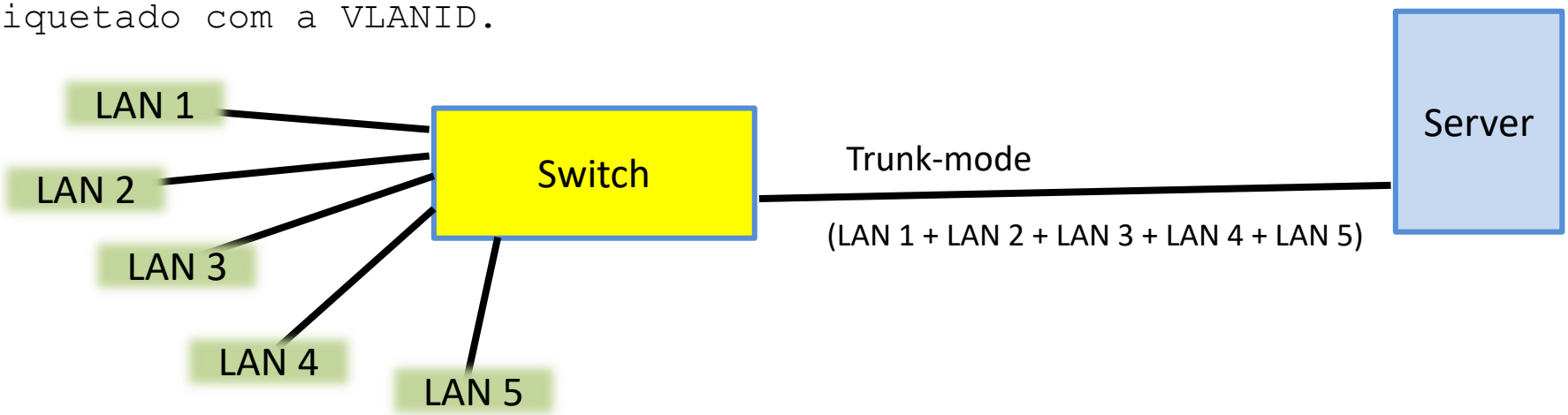
Como se pode observar pelo resultado do comando **ifconfig**, a interface de rede com o nome **ens160:rede1** passou a existir no sistema, esse facto facilita a configuração de serviços de rede pois a maioria deles permite explicitar qual a interface de rede a utilizar.

* Os dois pontos é uma convenção, não uma imposição

Configuração de VLANs

Através da utilização de etiquetas de VLAN IEEE 802.1q (**VLANID**) é possível usar uma única interface de rede para ligar a um conjunto numeroso de redes físicas distintas, esta é a utilização típica de VLANs em servidores.

Para o efeito será necessário um ***switch*** ligado às várias redes físicas e no qual se estabelece um VLANID para cada uma delas. A porta de ligação ao servidor é configurada em ***trunk-mode***, dessa forma o tráfego das várias redes físicas será enviada através dela devidamente etiquetado com a VLANID.



Graças à utilização de etiquetas nas duas extremidades da ligação ao servidor, o tráfego de diferentes redes físicas não se vai misturar. Do lado do servidor, na interface de rede, terá de ser definida uma **interface de VLAN** para cada uma das redes físicas, usando os VLANID estabelecidos no *switch*.

Configuração de VLANs no Windows Server

As versões recentes do Windows Server usam ***NIC Teaming*** para a definição de várias VLANs numa interface de rede usando etiquetas IEEE 802.1q.

A técnica ***NIC Teaming*** é usada para agrupar várias interfaces físicas numa única interface lógica (***NIC Team***), o objetivo é aumentar a performance (***load-balancing***) e garantir a tolerância a falhas (***failover***).

Para definir várias VLANs ligadas à mesma interface física também é usado *NIC Teaming* , mas neste caso o *NIC Team* é constituído por apenas uma interface física.

As interface de VLAN podem depois ser adicionadas ao *NIC Team* definindo o VLANID de cada uma delas e atribuindo nomes apropriados a cada nova interface de VLAN. Note-se que ao adicionar uma interface física a um *NIC Team*, a configuração IP da interface física deixa de ser utilizada, a configuração IP terá de ser definida novamente na interface *NIC Team*, através do ***Control Panel***.

Depois de adicionadas as VLANs ao *NIC Team*, a configuração IP da cada interface de VLAN também é realizada da forma habitual no ***Control Panel***.

A interface *NIC Team* (*primary interface*) não usa etiquetas VLANID (*untagged frames*), as interfaces de VLAN usam etiquetas (*tagged frames*).

Configuração de VLANs em Linux

Para criar interfaces de VLAN sobre uma interface física usa-se o comando **vconfig**:

vconfig add INTERFACE VLANID

O comando cria uma interface de VLAN na interface física INTERFACE onde será usada a etiqueta de VLAN fornecida como segundo argumento (VLANID). A nova interface de VLAN terá o nome **INTERFACE.VLANID** (na realidade a forma de nomear as interfaces de VLAN pode ser estabelecida através do comando **vconfig set_name_type FORMA-DE-NOMEAR** aplicando-se a interfaces adicionadas posteriormente; o valor por omissão da FORMA-DE-NOMEAR é **DEV_PLUS_VID_NO_PAD** que corresponde a **INTERFACE.VLANID**).

Uma vez adicionada a interface de VLAN, as suas configurações IP podem ser definidas através do comando **ip addr add** ….

A interface física (INTERFACE) continua a operar normalmente sem etiquetas VLANID (*untagged frames*) as interfaces de VLAN usam etiquetas (*tagged frames*).

A opção **rem** do **vconfig** remove uma interface de VLAN:

vconfig rem VLAN-INTERFACE

Normalmente: vconfig rem INTERFACE.VLANID

Configuração de VLANs em Linux - exemplo

Neste exemplo criam-se duas interfaces de VLAN sobre a interface física **ens160** com etiquetas **300** e **320**, atribuem-se endereços IPv4 a todas as interfaces e finalmente as interfaces são ativadas:

```
vconfig add ens160 300
vconfig add ens160 320
ip addr add 192.168.5.1/24 broadcast + dev ens160
ip addr add 192.168.6.1/24 broadcast + dev ens160.300
ip addr add 192.168.7.1/24 broadcast + dev ens160.320
ip link set dev ens160 up
ip link set dev ens160.300 up
ip link set dev ens160.320 up
```

Os pacotes emitidos na interface **ens160** não contêm etiquetas de VLAN, os pacotes emitidos na interface **ens160.300** têm a etiqueta de VLAN 300, os pacotes emitidos na interface **ens160.320** têm a etiqueta de VLAN 320.

Na interface **ens160** apenas são recebidos pacotes sem etiqueta, na interface **ens160.300** apenas são recebidos pacotes com etiqueta 300, na interface **ens160.320** apenas são recebidos pacotes com etiqueta 320. Pacotes recebidos com outras etiquetas serão descartados.

VLANs vs. Múltiplos endereços na mesma interface

À primeira vista criar várias interfaces de VLAN sobre a mesma interface física ou atribuir vários endereços IP à mesma interface física podem parecer soluções semelhantes.

Em ambos os casos podemos ter pacotes IP pertencentes a diferentes redes IP a circular sobre a mesma rede física. A diferença está na forma como pacotes de diferentes redes IP são distinguidos uns dos outros.

Não usando VLANs, a distinção é feita através dos endereços IP, portanto no nível 3, isso tem algumas consequências. Por exemplo para pacotes enviados para endereços *multicast* ou *broadcas*t genérico (255.255.255.255) não é possível determinar a que rede pertencem. Isto poderá ter impacto sobre os protocolos que usam este tipo de endereços.

Outro inconveniente é que sem usar VLANs não será possível ligar várias redes a um *switch* porque este opera no nível 2 e não analisa endereços IP.

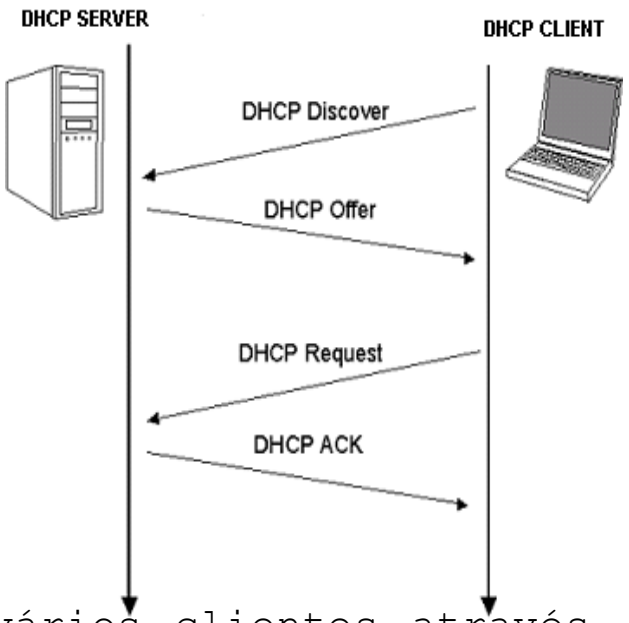
Quando se utilizam VLANs, a distinção é feita através do VLANID transportado nos *frames*, portanto no nível 2. Cada VLAN é para todos os efeitos uma rede física distinta, um *frame* emitido numa VLAN nunca se propaga em circunstancia alguma a uma VLAN diferente.

Nomeadamente, pacotes IP enviados para endereços de *broadcast* e *multicast* nunca passam de uma VLAN para outra VLAN.

Serviço DHCP (DHCP server)

Embora a configuração de interfaces de rede de servidores através de DHCP seja desaconselhada, para postos de trabalho é fundamental. Permite que um utilizador ao ligar o seu posto de trabalho a uma rede que desconhece (cablada ou wireless) obtenha os dados de configuração IP adequados a essa rede. Por exemplo nos sistemas Linux o serviço **ifplugd** deteta quando um cabo de rede é ligado ao equipamento e invoca os serviços necessários à configuração via DHCP da interface de rede.

O serviço DHCP utiliza pacotes UDP, o primeiro pedido do cliente é enviado para o endereço de broadcast genérico **255.255.255.255**, no caso do IPv4, ou para o endereço multicast link local **ff02::1:2**, no caso do IPv6 (**DHCPv6**). A imagem ilustra os vários passos no caso do IPv4, para DHCPv6 o cenário é semelhante embora as mensagens sejam diferentes.



A utilização de broadcast ou multicast link-local implica que na rede física terá de existir pelo menos um servidor DHCP.

Normalmente os servidores DHCP identificam os vários clientes através dos respetivos endereços de nível 2 (MAC *addresses*), a sua principal missão é fornecer a cada cliente endereços IP únicos na rede. Para esse efeito fazem a gestão de gamas de endereços disponíveis na rede.

Além do endereço IP único para cada cliente, o servidor DHCP fornece ou pode fornecer outros parâmetros que são normalmente idênticos para todos os clientes da mesma rede: máscara da rede, *default gateway*, endereços dos servidores DNS e o nome do domínio DNS local (para completar nomes não qualificados). Além destes parâmetros, as respostas DHCP podem ainda conter muitos outros com diversas finalidades.

O serviço DHCP pode ser disponibilizado por diversos tipos de equipamentos de rede como *routers*, *switches Layer 3* e servidores.

Windows Server: o role **DHCP Server** pode ser ativado através da aplicação **Server Manager**, uma vez ativado, a aplicação **DHCP Manager** pode ser usada para o gerir. O serviço deve ser associado às interfaces das redes (**binding**) que se pretende servir. Para cada rede que se pretende servir deve ser adicionado um **Scope** com uma **Pool** de endereços disponíveis e restantes parâmetros a fornecer aos clientes. É também possível definir parâmetros comuns a todos os Scopes. O servidor DHCP dos sistemas Windows atuais suporta tanto DHCPv4 como DHCPv6.

Linux: existem vários servidores DHCP *open source* gratuitos que suportam DHCPv4 e DHCPv6, um dos mais usados é o ***ISC DHCP Server*** disponibilizado pela ISC (Internet Systems Consortium, Inc.). Como é habitual em Linux/Unix, toda a configuração do *ISC DHCP Server* é realizada através de ficheiros de texto.

DHCPv6 (DHCP para clientes IPv6)

No caso do **IPv6** o contexto de aplicação do DHCPv6 é bastante diferente. O IPv6 tem mecanismos próprios para configuração automática dos nós (*Neighbour Discovery Protocol* – **NDP**) que não podem ser contornados. O DHCP pode ser usado ou não dependendo da configuração do *router* IPv6.

Quando um nó IPv6 pretende configurar automaticamente a rede envia um pedido ICMPv6 *Router Solicitation*, para o endereço multicast link-local **ff02::2** (*local routers*). Se existirem routers IPv6 na rede devem responder com a mensagem ICMPv6 ***Router Advertisement***.

O comportamento do cliente é controlado pela resposta do *router*, desde logo a resposta contém o endereço do *router* e também a identificação dos endereços das redes IPv6 que conhece. Com esta informação o cliente fica a conhecer o *router* e a rede a que está ligado, por essa razão os servidores DHCPv6 não necessitam de fornecer aos clientes a identificação da rede nem o *default gateway*.

A resposta contém ainda dois bits que determinam o comportamento do cliente. Se o bit **M** (*Managed address configuration*) tiver o valor 1, então o cliente deve usar DHCPv6 para saber o seu endereço IPv6 e completar a informação que já conhece. Caso contrário o cliente autoatribui-se um endereço sobre a rede que já conhece usando SLAAC.

O bit **O** (*Other configuration*) só é usado para **M=0** (SLAAC). Se **O=1** será usado DHCPv6 para obter parâmetros adicionais necessários (que não incluem o endereço já definido por SLAAC).

IPv6 Router Advertisement Service

A aplicação do DHCPv6 depende das respostas ***Router Advertisement*** do protocolo **NDP** (*Neighbour Discovery Protocol), trata-se* de um serviço que deve ser implementado e configurado nos *routers* IPv6.

Os servidores Windows e Linux também pode operar como *routers* IPv4/IPv6, nessa caso terão de enviar periodicamente para as redes ***Router Advertisements*** e também quando solicitados pelos clientes.

Windows Server: o serviço está integrado no role designado ***Routing and Remote Access*** (ou ***Remote Access*** apenas). Depois de instalado, a ferramenta administrativa ***Routing and Remote Access*** pode ser usada para ativar o *routing* IPv4/IPv6 e configurar o seu funcionamento.

Linux: o *Router Advertisement Daemon* (**radvd**), *open source*, pode assegurar este serviço nos *routers* Linux. No ficheiro de configuração **/etc/radvd.conf** para cada interface de rede onde o router se anuncia pode ser definida o bit **M** (*AdvManagedFlag*) e o bit **O** (*AdvOtherConfigFlag*).

DHCP Relay – O cliente DHCP só consegue contactar o servidor se ele se encontrar na mesma rede (domínio de broadcast). Quando se pretende utilizar num único servidor DHCP para servir um grande número de redes dispersas, isto pode ser inconveniente. O serviço *DHCP relay pode ser* instalado num dispositivo ou servidor de cada rede, e configurado para retransmitir os pedidos para um servidor DHCP remoto.

ISC DHCP Server (Linux) - Exemplo /etc/dhcpd.conf

```
option domain-name "dei.isep.ipp.pt";
option ip-forwarding off;
option netbios-node-type 8;
option netbios-scope "";
max-lease-time 604800;
default-lease-time 86400;

subnet 100.4.0.0 netmask 255.255.0.0 {
option subnet-mask 255.255.0.0;
option routers 100.4.0.1;
option broadcast-address 100.4.255.255;
option domain-name-servers 192.168.62.15;
option netbios-name-servers 192.168.62.15;
option log-servers 192.168.62.37;
range 100.4.10.0 100.4.30.255;
}
```

As declarações **options** correspondem a parâmetros a fornecer aos clientes via DHCP. Para cada rede a servir deverá haver uma declaração **subnet**. As declarações efetuadas fora de um bloco **subnet** aplicam-se a todas as declarações **subnet**, mas podem ser sobrepostas dentro do bloco.

Fault tolerant DHCP Service

Pelo facto de o primeiro contacto com o servidor ser realizado com recurso a *broadcast* (IPv4) ou *multicast* (IPv6), implementar um serviço DHCP tolerante a falhas é relativamente simples.

Existindo vários servidores DHCP na rede (domínio de *broadcast* ou *multicast link-local*), se um deles não estiver disponível para responder a clientes, outros estarão.

Para evitar conflitos é necessário garantir que cada servidor gere uma gama de endereços IP diferente dos restantes. Os endereços atribuídos pelos vários servidores podem pertencer todos à mesma rede IP, ou pertencer a redes IP diferentes, os restantes parâmetros de configuração fornecidos por cada servidor DHCP têm de ser definidos em conformidade. Esta configuração garante também ***load balancing***, tendencialmente o primeiro servidor a responder será aquele que possui menor carga no momento.

A Microsoft recomenda a existência de dois servidores DHCP, devendo o conjunto de endereços disponíveis ser divido entre um servidor primário com 80% dos endereços e um secundário com 20% dos endereços.

No caso de um domínio Windows, recomenda-se que o role DHCP Server seja instalado em DCs do domínio para permitir uma melhor integração com a base de dados *Active Directory*. Neste caso, os postos de trabalho do domínio (*Domain Members*) devem ser configurados para registarem no DNS do domínio o seu nome obtido por DHCP.

Resolução de nomes através do serviço DNS (DNS servers)

Entre outros equipamentos de rede, servidores Windows e servidores Linux podem exercer a função de servidor DNS.

No caso dos domínios Windows o *role* DNS server faz parte dos vários serviços obrigatórios que têm de estar presentes nos DC. Quando um Windows Server é promovido a *Domain Controller*, o *role* DNS Server é instalado automaticamente. Recorde-se que com o *Active Directory* cada domínio Windows corresponde a um domínio DNS.

Em cada domínio DNS existe obrigatoriamente um servidor DNS ***master*** e deve existir pelo menos mais um servidor DNS ***slave***.

O servidor *master* possui a cópia original da base de dados de registos DNS. Os servidores *slave* também possuem uma cópia local da base de dados, mas obtêm-na do *master*. Ambos se designam ***authoritative name servers*** do domínio porque possuem cópias locais da base de dados de registos do domínio. Todas as modificações à base de dados devem ser feitas no *master*. Os *slaves* comparam a data da última alteração da base da dados que se encontra no master (***serial number*** do registo SOA) com a cópia local; se são diferentes solicitam uma transferência total de zona (***AFXR DNS query***) ou apenas das alterações (incremental) relativamente ao *serial number* que possui localmente (***IFXR DNS query***).

Após algum tempo sem conseguir sincronizar-se com o *master*, um servidor *slave* deixar de ser ***authoritative*** para o domínio.

DNS - Zonas

Zonas são conjuntos de registos DNS que pertencem ao mesmo domínio DNS, um servidor DNS pode servir várias zonas. Cada zona possui obrigatoriamente um registo SOA que contém o nome DNS do servidor master da zona, o endereço de email do administrador da zona (com o símbolo **@** substituído por um **ponto**), o ***serial number*** e parâmetros de configuração relativos à sincronização dos *slaves* da zona.

Cada base de dados de zona deve conter também os registos NS relativos à zona (nomes dos servidores DNS) e registos NS relativos a subdomínios.

Zonas diretas – contêm registos usados na resolução direta, ou seja permitem obter endereços a partir de nomes qualificados. Notoriamente contêm registos do tipo A e AAAA, respetivamente para associar nomes a endereços IPv4 e endereços IPv6.

Zonas inversas – permitem a resolução inversa (***reverse resolution***), ou seja obter o nome DNS através do fornecimento do endereço IP. Estas bases de dados de zona correspondem a subdomínios dos domínios especiais **.in-addr.arpa.** e **ip6.arpa.**, respetivamente para endereços IPv4 e IPv6. Cada octeto em notação decimal (no caso do IPv4) ou símbolo hexadecimal (no caso do IPv6) corresponde a um subdomínio. Em ambos os casos o tipo de registo usado para o efeito é o PTR (*pointer*).

Note-se que não existe qualquer correspondência entre zonas diretas e zonas inversas.

Servidores DNS

Linux – O servidor DNS mais popular é o **bind**, *open source*, disponibilizado gratuitamente pela **ISC** (*Internet Systems Consortium, Inc.*). A sua configuração e gestão das bases de dados das zonas no caso de ser um master é realizada através de ficheiros de configuração em formato de texto. Também suporta alterações através de pedidos de clientes (*Dynamic DNS*).

Windows Server – o role ***DNS Server*** pode ser instalado num servidor *Standalone*, nesse caso a administração deverá ser realizada manualmente através da aplicação ***DNS Manager***.

Quando um Windows Server é promovido a controlador de domínio, é aconselhado que seja permitida a instalação automática do role ***DNS Server***, neste caso a gestão do serviço DNS é automaticamente realizada pelos **AD DS** (*Active Directory Domain Services*) e a gestão manual raramente será necessária.

A replicação de bases de dados DNS entre DCs de um domínio Windows não usa o esquema master/slave. Num domínio Windows, normalmente todos os DC são servidores DNS, a base de dados DNS encontra-se no *Active Directory* e é replicada automaticamente entre os vários DC.

Estando o servidor DNS sob o controlo dos AD DS, qualquer máquina adicionada ao domínio é automaticamente adicionada à base de dados DNS.

DNS Server – Linux – Bind version 9

A configuração do servidor bind assenta no ficheiro **named.conf**, normalmente residente na pasta **/etc/** ou **/etc/bind/**. A partir deste ficheiro são normalmente incluídos outros ficheiros de configuração.

Uma das configurações mais importantes é a definição dos ***forwarders***, trata-se de endereços IP de outros servidores DNS para os quais podem ser retransmitidos pedidos de resolução sem estar a recorrer ao processo de resolução normal do DNS. Exemplo:

```
options {
        forwarders { 192.249.249.1; 192.249.249.3; };
};
```

O bind pode ser *slave* de uma ou várias zonas. Declaração exemplo:

```
zone "dei.isep.ipp.pt" IN {
        type slave;
        file "/var/lib/bind/dei.isep.ipp.pt";
        masters { 193.136.62.15; };
};
```

Sendo um *slave* da zona **dei.isep.ipp.pt**, o ficheiro de zona declarado **/var/lib/bind/dei.isep.ipp.pt** é criado e gerido pelo bind e não deve ser editado. Contém uma cópia da base de dados de zona residente no servidor master da zona (com endereço **193.136.62.15** no exemplo).

DNS Server – Linux – Bind version 9

O bind pode ser *master* de uma ou várias zonas. Declaração exemplo:

```
zone "dei.isep.ipp.pt" IN {
        type master;
        file "/etc/bind/direct";
        allow-query { any; };
        allow-transfer { any; };
};
```

Neste caso trata-se do *master* da zona **dei.isep.ipp.pt**, o ficheiro de zona declarado **/etc/bind/direct** tem de ser criado e gerido manualmente.

Se o bind for configurado para suportar *Dynamic DNS*, vai alterar o conteúdo do ficheiro em conformidade, caso contrário o conteúdo só é alterado manualmente pelo administrador.

O ficheiro de zona, neste exemplo **/etc/bind/direct** contém todos os registos DNS que integram a base de dados da zona.

Nos ficheiros de zona todos os nomes devem ser qualificados (**fqdn**) **qualquer nome que não termine em ponto** é tido como não qualificado e o nome do domínio correspondente à zona é **automaticamente acrescentado.**

DNS Server – Linux – Bind version 9 – ficheiro de zona

O ficheiro de zona é um ficheiro de texto em que cada linha corresponde a um **RR** (*Resource Record*), tipicamente o primeiro registo é o **SOA** *(Start Of Authority*). O formato geral de cada linha do ficheiro de zona é:

nome-do-registo ttl class tipo-de-registo dados-do-registo

nome-do-registo – nome DNS do registo. Pode ser omitido, nesse caso a linha deverá começar por um espaço e será usado o mesmo valor do registo anterior.

ttl (*Time To Live*) – tempo máximo de vida em segundos. Quando um cliente DNS obtém o registo deve considerar que passado este tempo o registo é obsoleto e deverá obter um novo. Este elemento é opcional, se não for especificado será o valor do registo anterior ou o valor declarado na diretiva **$TTL**.

class – indica o tipo de nome, para nomes da internet o valor é **IN**. Tal como o anterior pode ser omitido sendo usado o valor do registo anterior.

tipo-de-registo – mnemónica que indica de que tipo de registo DNS se trata.

dados-do-registo – conteúdo do registo, a sua forma depende do tipo de registo.

DNS Server – Linux – Bind version 9 – SOA

O registo **SOA** é normalmente o primeiro registo do ficheiro de zona e contém os seguintes elementos:

nome-master email-administrador (serial refresh retry expiry nx)

nome-master – nome DNS do servidor master da zona.

email-administrador – endereço de correio eletrónico do administrador da zona com o símbolo **@** substituído por um **ponto**.

serial – número de série da zona. Regista a última alteração à base de dados, na forma **yyyymmddss**, onde **yyyymmdd** representa o dia em que foi alterada e **ss** o número da alteração nesse dia (começa por ser zero).

refresh – número de segundos entre verificação de atualizações por parte dos servidores *slave*.

retry – número de segundos que os *slaves* devem aguardar em caso de falha no contacto com o master antes de tentarem novamente.

expiry – número de segundos sem conseguirem contactar o master após o qual os servidores *slave* deixam de ser *authoritative* para a zona.

nx – tempo máximo em segundos de *caching* para respostas NXDOMAIN (*non-existent domain*).

Linux – Bind version 9 – outros registos

Além do SOA outros tipos de registos DNS importantes são:

Tipo (mnemónica)	Conteúdo
A	Endereço IPv4
AAAA	Endereço IPv6
NS	Nome de nó servidor de nomes (correspondente a um registo A e/ou AAAA)
CNAME	Nome de nó (correspondente a um registo A e/ou AAAA)
MX	Prioridade e nome de nó servidor SMTP (correspondente a um registo A e/ou AAAA)
PTR	Nome de nó (correspondente a um registo A ou AAAA)
TXT	Um comentário (também usado na implementação de SPF*)

* SPF (*Sender Policy Framework*) identificam que servidores estão autorizados a enviar mails em nome do domínio

Os registos PTR são usados em zonas inversas, ou seja correspondentes a subdomínios de **in-addr.arpa.** e **ip6.arpa.**, respetivamente para endereços IPv4 e IPv6.

Linux – Bind version 9 – exemplo de ficheiro de zona

```
$TTL 1d
$ORIGIN meu.dominio.pt

@     IN    SOA   nserv.meu.dominio.pt.   admin.meu.dominio.pt  ( 2015091502   1d   15M   2W 1h )

            NS          nserv
nserv       A           192.168.70.2
            AAAA        2001:db8:10::2

@           MX          10          mailserv1
            MX          20          mailserv2.meu.dominio.pt.
server1.meu.dominio.pt.             A           192.168.70.90
mailserv1.meu.dominio.pt.           A           192.168.70.23
                                    TXT         "Servidor SMTP principal"
                                    AAAA        2001:db8:10::1
mailserv2                           A           192.168.70.27
mail                        CNAME   mailserv2
www                         CNAME   server1
ftp                         CNAME   server1
```

A declaração **$ORIGIN** define o domínio correspondente à zona, pode depois ser referenciada através do símbolo @. Todos os registos são da classe IN, com TTL de 1 dia (definido na declaração **$TTL**).

Administração de Sistemas (ASIST)

Aula Teórico Prática 05

Administração de servidores.
Acesso remoto. Gestão de software.
Scripts. Execução programada.
Storage centralizada - SAN - iSCSI.

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Gestão de servidores – acesso remoto

Uma vez que os endereços IP dos servidores estejam definidos, e de preferência registados no DNS, o administrador pode optar por usar protocolos de aplicação apropriados para aceder e administrar o servidor e abandonar a utilização da consola.

Servidores Linux: sem interface gráfica, a forma de acesso mais adequada é a utilização do protocolo **SSH** (Secure Shell). Para o efeito basta instalar um servidor SSH, como o ***OpenSSH Server***. O SSH é uma emulação de terminal efetuada sobre uma ligação TCP protegida por **TLS** (*Transport Layer Security*). Neste tipo de utilização o servidor não é autenticado por certificado de chave pública. Quando o utilizador liga a um servidor, este fornece a sua chave RSA, se for desconhecida, o utilizador é alertado.

Servidores Windows: a forma mais simples de acesso remoto é através do protocolo **RDP** (*Remote Desktop Protocol*). Basta ativar através do *Server Manager* e usar um cliente apropriado (*Remote Desktop Connection* - **RDC** / *Ligação a Ambiente de Trabalho Remoto*). Outra forma de gestão remota dos servidores é através do *Server Manager*. Esta aplicação permite a gestão não apenas do servidor local onde é executada, mas também de outros servidores através da rede.

Gestão de software

Após a instalação inicial do servidor torna-se fundamental dispor de ferramentas que permitam de forma expedita a sua atualização e instalação de novo software necessário.

Linux: para a maioria das distribuições existe um conjunto de repositórios na Internet através dos quais são fornecidas atualizações e acesso a diversos pacotes de software que se pretenda instalar. A aplicação/serviço usada para esse efeito depende da distribuição, no caso da família Debian (na qual se integra o Ubuntu), a gestão de pacotes de software (*package management system*) é controlada através do comando **apt** (*Advanced Package Tool*).

Windows Server: a maioria do software essencial está disponível localmente após a instalação, pode ser gerido através do ***Server Manager*** adicionado/removendo *roles* e *features*. *Role* é um conjunto de pacotes de software que permite desempenhar determinada função. *Features* são pacotes de software que não fazendo parte de um *role* desempenham outras funções que podem estar relacionadas com vários *roles*. É normal a instalação de um *role* obrigar à instalação de algumas *features*. Um dos *role* ´disponíveis é o **WSUS** (*Windows Server Update Services*) que permite disponibilizar localmente um serviço de atualização de software para os equipamentos locais.

Linux - APT (Advanced Package Tool)

Nas distribuições da família Debian o gestor de pacotes de software é o **apt.** A instalação de atualizações pode ser também automatizada através da instalação do pacote **unattended-upgrades.**

Comandos **apt** essenciais:

Comando	Descrição
apt update	Atualiza a lista de pacotes disponíveis descarregando dos vários repositórios as listas de pacotes existentes.
apt install {NOME-DO-PACOTE}	Solicita a instalação do pacote.
apt remove {NOME-DO-PACOTE}	Solicita a remoção do pacote, os ficheiros de configuração alterados pelo administrador não são removidos.
apt purge {NOME-DO-PACOTE}	Solicita a remoção do pacote incluindo os ficheiros de configuração (válido mesmo que o pacote já tenha sido removido).
apt dist-upgrade	Solicita uma atualização geral de todos os pacotes instalados. Pode incluir a remoção de pacotes que deixaram de ser necessários.
apt autoremove	Solicita a remoção de todos os pacotes instalados que já não estão a ser utilizados (pacotes obsoletos que já foram substituídos por outros).

Linux - APT (Advanced Package Tool)

Comando	Descrição
apt list [--installed] [NOME-DO-PACOTE]	Sem opções, lista todos os pacotes existentes. Com a opção --installed, apenas pacotes instalados. É possível especificar um nome de pacote ou um padrão. Utilizando um padrão torna-se simples procurar um pacote que se pretenda. Por exemplo para obter uma lista de todos os pacotes existentes cujo o nome contém a palavra **libnss** pode ser usado o comando: **`apt list *libnss*`**
apt search {STRING}	Permite procurar pacotes através de um STRING; serão procuradas correspondências do STRING no nome e também na descrição detalhada do pacote.
apt show {NOME-DO-PACOTE}	Apresenta todos os detalhes sobre o pacote indicado.

O **apt** usa o **dpkg** (*package manager for Debian*) mas nem todas as possibilidade do *dpkg* estão disponíveis no *apt*. Frequentemente, durante a instalação de um pacote são executados scripts de instalação que solicitam ao utilizador as configurações a aplicar. Se depois da instalação for necessário alterar essas configurações, pode ser usado o comando:

`dpkg-reconfigure {NOME-DO-PACOTE}`

WSUS(*Windows Server Update Services*)

Como qualquer sistema operativo, no Windows Server devemos automatizar as atualizações, a opção recomendada é **descarregar e instalar automaticamente as atualizações**, para manter maior controlo podemos optar por **descarregar atualizações e pedir a confirmação antes de as instalar**.

A instalação de novas aplicações num servidor Windows para além das que estão disponíveis através do *Server Manager* (*roles and features*) deve ser feita com cautela, de preferência deverão ser aplicações da Microsoft cuja compatibilidade seja garantida para a versão do sistema operativo em causa.

Para descarregar pacotes de instalação usando o Internet Explorer, o administrador terá de desativar o modo ESC (**IE ESC – *Internet Explorer Enhanced Security Configuration***) através do Server Manager.

O WSUS Server role pode ser ativado num Windows Server para obter atualizações da Microsoft e disponibilizar as mesmas para as máquinas Windows locais, configurando-o para obter atualizações para os sistemas operativos e linguagens pretendidas. Pode ainda incorporar outras aplicações Microsoft como por exemplo novas versões do IE. É possível controlar as atualizações que serão aplicadas a diferentes grupos de computadores locais e a hora de aplicação. Os computadores locais são configurados através de uma *Group Policy* para obterem as atualizações do servidor WSUS local.

WDS (*Windows Deployment Services*)

O WDS permite administrar um parque informático constituído por um grande número de máquinas Windows. O WDS só pode ser aplicado no contexto de um domínio *Active Directory*, exige também um servidor DHCP e postos de trabalho em que as interfaces de rede suportem PXE (*Preboot eXecution Environment*).

Permite:

- A instalação *unattended* do sistema operativo a partir de um *boot* de rede (PXE). A imagem do disco pode ser distribuída por multicast, efetuando a instalação em muitas máquina simultaneamente.
- O sistema operativo é gerido numa imagem residente no servidor WDS designada **imageX**. A imagem pode ser criada num posto de trabalho e copiada para o servidor ou totalmente criada no servidor.
- No servidor WDS é possível alterar a imagem (**imageX**), nomeadamente instalar software e drivers e aplicar automaticamente as alterações a todos os postos de trabalho (*deploy*).
- Durante o processo de instalação em cada posto de trabalho, são executados procedimentos de instalação do sistema operativo tendo em vista a sua adaptação ao hardware.
- Durante a instalação o nome da máquina é definido corretamente e a máquina pode ser automaticamente inserida no domínio Windows.

Scripts

Atualmente este é um conceito muito abrangente. Os *scripts* começaram por ser *batch files*, ou seja ficheiros de texto com uma sequencia de comandos, o *batch file* pode ser invocado provocando a execução sequencial dos comandos (execução em lote).

Atualmente todas as linguagens de *scripting* incluem também comandos que permitem definir variáveis, estruturas de decisão e repetição assemelhando-se muito a verdadeiros programas. O conceito atual de *script* está fortemente associado ao facto de ser executado por um programa interpretador e não se tratar de um programa compilado.

Para que um *script* (ficheiro de texto), possa ser diretamente executado pelo sistema operativo sem necessidade de invocar explicitamente o programa interpretador é necessário que o sistema operativo o reconheça como tal.

Windows: o tipo de conteúdo de cada ficheiro é identificado através do seu nome, mais exatamente da sua **extensão** (conjunto de 3 carateres no final do nome após o ponto). Para cada extensão o sistema operativo tem definida a forma de lidar com o ficheiro, no caso dos *scripts*, o programa interpretador que deve executar para o processar.

Linux: o tipo de conteúdo de cada ficheiro é identificado através do ***magic number*** do seu conteúdo, o nome é completamente ignorado.

Unix - Magic number

Os sistemas operativos Unix identificam o tipo de conteúdo dos ficheiros através de ***magic numbers***. Um ***magic number*** é um conjunto de bytes com valores fixos existentes em determinadas posições fixas do ficheiro que permite identificar o seu conteúdo.

Na Shell, o comando **file** permite determinar de que tipo é um objeto do sistemas de ficheiros. Depois de verificar que não se trata de uma pasta, uma ligação simbólica ou algum tipo de objeto especial, verifica o seu conteúdo e confronta-o com os ***magic number*** conhecidos.

O *magic number* que identifica um script é constituído pelo caracteres **#!** nas duas primeiras posições do ficheiro. Esta sequência no inicio de um ficheiro é designada **shebang**, imediatamente a seguir, na mesma linha do ficheiro de texto, deve ser identificado o programa interpretador que deve ser usado para processar o conteúdo.

```
#!/bin/bash
--- --
-- -- --
```

```
#!/usr/bin/perl
--- --
-- -- --
```

```
#!/bin/sh
--- --
-- -- --
```

```
#!/bin/csh
--- --
-- -- --
```

```
#!/usr/bin/pyton
--- --
-- -- --
```

Adicionalmente o ficheiro deve ter a permissão *execute* para poder ser diretamente executado pelo sistema operativo. Note-se que as linhas começadas pelo caractere **#** são consideradas comentários pelos interpretadores pelo que a linha do **shebang** não interfere com a execução do script.

```
#!/usr/bin/php
--- --
-- -- --
```

Linux – Shell scripts

Trata-se de scripts que são executados pelos interpretadores de linha de comando habitualmente designados Shell, destacando-se a *Bourne Shell* (sh/bash) e a C Shell (csh/tcsh), entre outras.

Se um ficheiro começa pelo *shebang* mas nessa linha não se identifica o programa interpretador, será usada a *default* Shell, normalmente a **bash** (embora isso depende da distribuição).

Tanto a *Bourne Shell* tradicional como a *C Shell* estão disponíveis em qualquer instalação Linux pelo que constituem uma ferramenta importante que está sempre disponível para o administrador automatizar tarefas, bastando dispor de um editor de texto.

Os Scripts escritos em C Shell utilizam uma sintaxe bastante semelhante à linguagem de programação C, tornam-se por isso interessantes para administradores que trabalham habitualmente com essa linguagem.

Outros tipos de linguagens interpretadas como *Perl* e *Pyton* possuem potencialidades muito maiores, com bibliotecas bastante extensas, mas exigem a familiarização do administrador.

O exemplo apresentado à direita, escrito em C Shell, imprime todas as potências de dois inferiores a 1100.

```
#!/bin/csh
echo "Starting"
@ num = 1
while($num<1100)
 echo $num
 @ num = 2 * $num
end
```

Linux – Shell scripts

O exemplo seguinte, escrito em *bash*, recebe um número como argumento na linha de comando (${1} é o primeiro argumento) e imprime o valor do fatorial desse número.

```
#!/bin/bash
echo "Calculador de fatorial"
if [ -z "${1}" ]
        then
        echo "Deve fornecer o argumento (inteiro positivo)"
        exit 1
fi
NUM=${1}
FACT=1
while [ ${NUM} -gt 1 ]
        do
        FACT=$((${FACT}*${NUM}))
        NUM=$((${NUM}-1))
done
echo "Fatorial de ${1} = ${FACT}"
#### DONE
```

A colocação de chavetas a envolver os nomes das variáveis referenciadas não é obrigatória. Os espaços colocados no contexto dos testes de condições (**if** e **while**) são fundamentais.

Scripts em Windows – WSH (Windows Script Host)

Nos sistemas operativos Windows os scripts são identificados através da sua extensão:

Extensão	Tipo de script
.BAT .CMD	Ficheiros processados pelo interpretador de linha de comandos (cmd.exe).
.JS .JSE	**JScript** – Linguagem interpretada baseada no Microsoft Visual J++ 6.0.
.VBS .VBE	**VBScript** – Linguagem interpretada baseada no Microsoft Visual Basic 6.0.
.WSF	**Windows Script File** – Utiliza XML para suportar diferentes linguagens no mesmo script, por exemplo JScript e VBScript.
.PS1	**Windows PowerShell** shell script

Os sistemas Windows atuais dispõem de uma plataforma de suporte à execução de scripts designada *Windows Script Host* (**WSH**).

Esta plataforma permite às linguagens interpretadas compatíveis, nomeadamente JScript e VBScript, o acesso a um vasto conjunto de funcionalidades através de objetos que podem utilizar. Através do WSH é por exemplo possível utilizar o **WMI** (*Windows Management Instrumentation*) e o **ADSI** (*Active Directory Service Interfaces*).

WSH (Windows Script Host) - Exemplo

Neste exemplo em JScript é usado o WSH para aceder ao WMI e obter uma lista de contas de utilizador. Para cada conta é apresentada sucessivamente uma janela com o nome de utilizador e a data do último login:

```
LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile");
for(e = new Enumerator(LoginProfiles) ; !e.atEnd() ; e.moveNext())
{
   Profile = e.item();
   lastLog="never";
   if(Profile.LastLogon!=null) lastLog=Profile.LastLogon;
   WScript.Echo(Profile.Name + " : " + lastLog);
}
```

O exemplo seguinte, em VBScript faz o mesmo:

```
Set LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile")
   for each Profile in LoginProfiles
      WScript.Echo Profile.Name
      WScript.Echo Profile.LastLogon
   next
```

WSF (Windows Script File) - Exemplo

Um script WSF utiliza XML e pode conter diferentes linguagens:

```
<job>
<script language="VBScript">
  MsgBox "Eu agora estou a executar VBScript, mas a seguir vou executar JScript"
</script>
<script language="JScript">
  WSH.echo("Agora já estou a executar JScript e vou mostrar as contas de utilizador.");

  LoginProfiles = GetObject("winmgmts:").InstancesOf ("Win32_NetworkLoginProfile");
  num=0;
  for(e = new Enumerator(LoginProfiles) ; !e.atEnd() ; e.moveNext())
   {
   Profile = e.item();
   lastLog="never";
   if(Profile.LastLogon!=null) lastLog=Profile.LastLogon;
   WScript.Echo(Profile.Name + " : " + lastLog);
   num++;
   }
  WScript.Echo("Foram apresentadas " + num + " contas de utilizador");
  WScript.quit();
</script>
</job>
```

Windows PowerShell

A PowerShell disponibiliza um conjunto suplementar de comandos designados ***cmdlets***, permitindo administrar todas as funcionalidades do servidor. Foi concebida com o propósito de facilitar as tarefas de administração dos servidores Windows.

A quantidade de cmdlets é imensa, os nomes são orientados pelo verbo da ação pretendida, por exemplo para consultar o estado são usados comandos começados por **Get-**, para alterar comandos começados por **Set-**.

Desde logo, o comando **Get-help** pode ser usado para obter ajuda, por exemplo **Get-help service** permite obter uma listas dos cmdlets cujo nome contém **service**. Ajuda detalhada sobre o cmdlet pode depois ser pedida diretamente, por exemplo **Get-help Start-service**.

A PowerShell também dispõe de *autocomplete* através da tecla TAB.

O desenvolvimento de scripts é muito facilitado pelo *Integrated Scripting Environment* (**ISE**). Trata-se de uma ferramenta em modo gráfico de desenvolvimento de scripts PowerShell com assistência direta disponível, desde logo com informação sobre todos os cmdlets disponíveis.

Os scripts (ficheiros com extensão **.PS1**) são editados no ISE e podem desde logo ser testados no próprio ambiente. O ambiente de desenvolvimento de scripts PowerShell ISE encontra-se disponível nas ***Administrative Tools*** do Windows Server.

Windows PowerShell script – exemplos WMI

```
# WMI query to list all properties and values of the Win32_BIOS class
# http://www.robvanderwoude.com

param( [string]$strComputer = "." )

$colItems = get-wmiobject -class "Win32_BIOS" -namespace "root\CIMV2" -computername $strComputer

foreach ($objItem in $colItems) {
          write-host "Name                          :" $objItem.Name
          write-host "Version                       :" $objItem.Version
          write-host "Manufacturer                  :" $objItem.Manufacturer
          write-host "SMBIOSBIOS Version            :" $objItem.SMBIOSBIOSVersion
          write-host
}

# WMI query to list all properties and values of the Win32_Printer class
# http://www.robvanderwoude.com

$colItems = get-wmiobject -class "Win32_Printer" -namespace "root\CIMV2" -computername $strComputer

foreach ($objItem in $colItems) {
          write-host "Name                          :" $objItem.Name
          write-host "Default                       :" $objItem.Default
          write-host "Network                       :" $objItem.Network
          write-host "Port Name                     :" $objItem.PortName
          write-host "Driver Name                   :" $objItem.DriverName
          write-host "Server Name                   :" $objItem.ServerName
          write-host "Share Name                    :" $objItem.ShareName
          write-host
}
```

Execução programada de tarefas

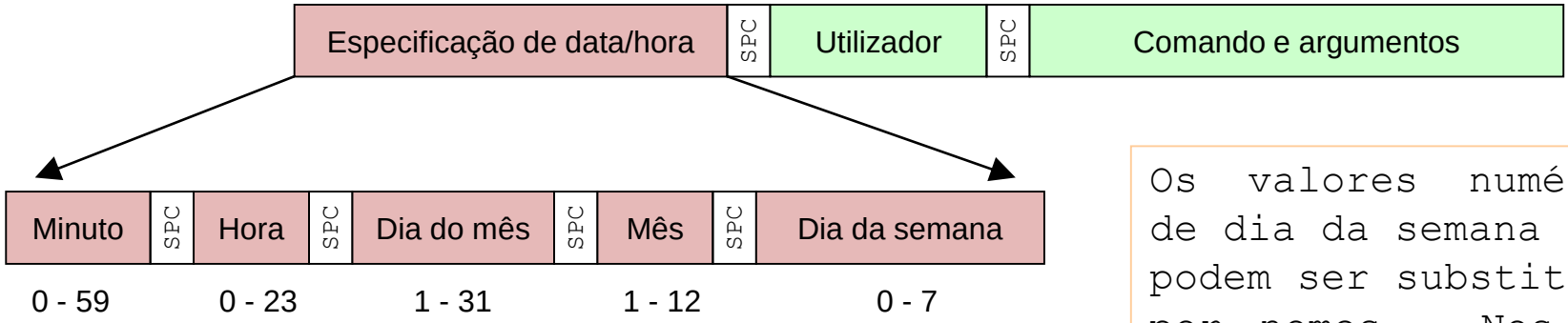
A execução programada de tarefas é uma ferramenta fundamental na administração de servidores, pois permitem ao administrador definir a execução de aplicações e scripts em modo *unattended* em horas apropriadas. Um exemplo típico são as operações de cópia de segurança.

Windows: a aplicação gráfica ***Task Scheduler*** permite programar tarefas. É possível especificar uma grande variedade de parâmetros relativamente à execução de tarefas, desde logo a conta de utilizador que será usada e condições especificas que poderão condicionar a execução da tarefa. Além de uma programação horária ao segundo, as tarefas também podem ser programadas para serem executadas em outros tipos de situações, como por exemplo no *boot* da máquina ou quando a máquina está *idle*. Também é possível definir uma data de expiração da tarefa, a tarefa não será eliminada mas ficará inativa a partir dessa data e não será executada. A granularidade mínima é 1 minuto.

Linux: nos sistemas Unix em geral o programador de tarefas mais usado é o **cron**. Trata-se de um serviço em execução permanente que de minuto a minuto lê o ficheiro de configuração **/etc/crontab** e executa as tarefas aplicáveis nesse minuto. Através das definições neste ficheiro podemos estabelecer horários de execução baseados na hora, minuto, dia do mês e dia da semana. A granularidade mínima é portanto também o minuto.

Linux – Serviço cron

A execução programada de tarefas em sistemas Linux usa habitualmente o serviço cron. Normalmente o administrador define as tarefas no ficheiro de texto **/etc/crontab**. Cada linha representa uma tarefa a ser executada na seguinte forma:



Os valores numéricos de dia da semana e mês podem ser substituídos por nomes. Nos dias da semana o valor **0** e o valor **7** são duas alternativas para especificar **domingo**.

Os valores de data/hora podem ser especificados em intervalos ou por enumeração dos valores separados por virgulas. O símbolo ***** representa a enumeração de todos os valores possíveis para esse campo. Por exemplo para minutos, a enumeração de todos os valores possíveis pode ser feita na forma **0-59** ou *****.

De minuto a minuto o serviço cron lê este ficheiro e compara a data/hora atual com as especificações existentes no ficheiro. Para cada correspondência encontrada executa o comando usando a conta de utilizador especificada.

Linux – exemplo /etc/crontab

```
*/5 * * * * root /usr/bin/mrtg /etc/mrtg/mrtg.cfg
5,35 * * * * root /etc/LinuxHealth >/dev/null 2>&1 &
45 2 * 8 6 root /root/make-backup >/dev/null 2>&1 &
```

As enumerações de valores podem ser associadas a um ***step***, no exemplo acima, nos minutos, ***/5** (equivalente a **0-59/5**) é equivalente a **0,5,10,15,20,25,30,35,40,45,50,55.** Neste caso (como as restantes especificações são *****) a tarefa será executada de cinco em cinco minutos, sempre que o valor data/hora tenha um número de minutos correspondente.

O ficheiro **/etc/crontab** apenas pode ser editado pelo administrador, no entanto o serviço *CRON* também está acessível aos outros utilizadores através de ficheiros de configuração normalmente guardados em **/var/spool/cron/*username*.** Estes devem ser manipulados indiretamente através do comando **crontab -e.**

O formato é idêntico ao do **/etc/crontab**, mas o nome do utilizador (6º campo) é omitido pois está implícito, as tarefas de um utilizador definidas desta forma são sempre executadas usando a conta desse utilizador. O acesso dos utilizadores ao serviço *CRON* pode ser controlado pelo administrador através dos ficheiros **/etc/cron.allow** e **/etc/cron.deny.**

Storage centralizada – SAN e NAS

A disponibilização centralizada através da rede de dispositivos de armazenamento externo (*storage*) pode ser realizada através de duas abordagens distintas, **SAN (*Storage Area Network*)** e **NAS (*Network Attached Storage*).**

No servidor NAS pastas do sistema de ficheiros local são disponibilizadas aos clientes, ao contrário dos discos lógicos a responsabilidade de gerir o sistema de ficheiros cabe ao servidor NAS.

Os sistemas Windows utilizam o protocolo *Server Message Block/Common Internet File System* (**SMB/CIFS**) para **partilhar pastas.** Os sistemas Unix usam o protocolo *Network File System* (**NFS**) para **exportar pastas.**

A terminologia é diferente, mas o conceito é o mesmo.

SAN (Storage Area Network) - iSCSI

O protocolo ***iSCSI*** permite criar uma SAN sobre qualquer infraestrutura de rede IP, não exigindo hardware especial. Servidores Windows e servidores Linux podem ser *servidores de storage* - ***targets*** **-** ou *clientes de storage* - ***initiators*** **-** que usam os discos lógicos.

Os nós intervenientes no *iSCSI* (***targets*** e ***initiators***) são identificados através de nomes especiais, normalmente no formato **IQN (***iSCSI Qualified Name***)**, na forma:

iqn.{YYYY-MM}.{nome-dns-invertido}:{nome}

Onde **{YYYY-MM}** representa o ano e mês em que foi criado o domínio DNS, **{nome-dns-invertido}** representa o nome DNS do domínio invertido e **{nome}** é um nome arbitrário que garante que o conjunto é único.

Por exemplo o nome do nó (no caso de um *target* habitualmente este nome também identifica o disco lógico).

Exemplo de IQN de um target: **iqn.2000-08.pt.ipp.isep.dei:server1-lun0**

Os IQN usados pelos *targets* podem ser descobertos através da função *discovery*, desde que se conheça os seus endereço IP ou nomes DNS.

O acesso dos *initiators* aos *targets* é controlado através de *username* e *password* sendo usado o algoritmo **CHAP**.

iSCSI Initiator – Windows

Os sistemas Windows atuais (incluindo o Windows Server) podem usar discos lógicos disponibilizados por targets, ou seja operar como *initiator*. Para gerir esta funcionalidade deve ser usada a aplicação **iSCSI Initiator** disponível nas **Administrative Tools**.

O target pode ser adicionado através do endereço IP ou nome DNS respetivo, é necessário então ligar ao target (*Connect*) definindo o *username* e *password* para autenticação via CHAP (opções avançadas do *Connect*) tal como foram definidos pelo target.

Se o target for adicionado à lista de favoritos, a ligação será feita automaticamente sempre que a máquina arranca.

Após a ligação ao target, podemos optar por configurar automaticamente todos os discos lógicos (volumes) ou apenas alguns dos disponibilizados no target. Estando configurados os volumes passam a ser novos discos disponíveis como se fossem locais.

Tratando-se de um disco vazio recém criado, através do **Computer Management** devemos colocar o disco on-line e inicializá-lo (criar a tabela de partições) após o que podem ser criadas novas partições para serem formatadas e atribuídas letras de drive.

Para o **Computer Management** tudo se passa como se tratasse de um disco físico local. Como tal todas as operações em discos locais estão disponíveis, incluindo ser convertido para ***dynamic***.

iSCSI Target – Windows

O **Windows Server 2012** pode funcionar como servidor de *storage* iSCSI (ou seja target iSCSI). Para o efeito terão de ser instalados os roles **File Server** e **iSCSI Target Server** (encontram-se no conjunto de roles ***File and Storage Services***). A *feature* **iSNS Server** (*discovery services*) também será instalada. Através do **Server Manager** é então possível criar discos lógicos a disponibilizar pelo target.

Apenas são suportados discos virtuais, ou seja, discos que são armazenados sob a forma de um ficheiro de grande dimensão (correspondente ao tamanho do disco.

Ao criar o disco virtual temos de definir a sua dimensão, mais tarde pode ser expandido se necessário, mas não pode ser encolhido.

Os discos virtuais podem ser criados de várias formas:
Fixed Size significa que será criado imediatamente um ficheiro com a dimensão correspondente ao tamanho do disco;
Dynamically Expanding leva a que o tamanho do ficheiro vá crescendo à medida que vão sendo colocados dados no disco;
Differencing permite criar um novo disco usando como base um disco já existente, o disco já existente é preservado no estado inicial e as alterações relativamente a ele são registadas no novo disco.

Depois de criado o disco virtual pode então ser definido o target correspondente.

iSCSI Target – Linux

No **Ubuntu 18.04** o pacote que permite criar um target iSCSI é o **tgt.** Após a sua instalação a configuração é realizada através do comando **tgtadm.**

O nome de targets (IQN) criados devem corresponder a discos a disponibilizar. Para facilitar a criação é criada uma pasta **conf.d** sob **/etc/tgt** onde podem ser criados vários ficheiros que são sempre incluídos – como se pode verificar em **/etc/tgt/targets.conf.**

Um disco lógico pode ser um disco físico, por exemplo /dev/sdc, também pode ser uma partição, por exemplo /dev/sdb3, ou à semelhança do Windows Server também pode ser um ficheiro que funciona como disco virtual. Ao definir o target são também definidas as credenciais de autenticação através do CHAP. Exemplo:

```
Target iqn.2015-10.pt.ipp.isep.dei:storage.lun0
        IncomingUser root PASSWORD
        OutgoingUser
        Lun 0 Path=/home/storage/lun0.disk,Type=fileio
```

Neste caso o target é um disco virtual armazenado no ficheiro **/home/storage/lun0.disk**, previamente criado com a dimensão pretendida para o disco lógico, podendo ser preenchido inicialmente com quaisquer dados. Posteriormente ele será formatado pelo *initiator* que o utilizar.

iSCSI Initiator – Linux

No **Ubuntu 18.04** o pacote que permite usar discos lógicos disponibilizados por targets iSCSI é o **open-iscsi** (*Open-iSCSI project* http://www.open-iscsi.org/). O primeiro passo é descobrir os nomes dos targets através do comando:

iscsiadm -m discovery -p {SERVER} -t st

Onde **{SERVER}** é o endereço IP ou nome DNS da máquina target. Identificado o nome IQN do disco lógico pretendido, é necessário definir o método de autenticação (CHAP), o username e a password para esse disco lógico.

```
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.authmethod –v CHAP
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.username –v {USERNAME}
iscsiadm -m node -T {IQN} –p {SERVER} –o update –n node.session.auth.password –v {PASSWORD}
iscsiadm -m node -T {IQN} –p {SERVER} -o update -n node.startup -v automatic
```

Estes dados são guardados e serão usados para ligar ao disco lógico {IQN}. Para ligar ao target basta agora usar o comando:

iscsiadm -m node –login -T {IQN} –p {SERVER}

Os targets configurados com os atributos fornecidos ficam guardados dentro da pasta **/etc/iscsi/nodes** de forma persistente, os ficheiros de configuração podem ser editados manualmente. No exemplo acima foi definido **node.startup = automatic**, quando a máquina arrancar a ligação a este target (login) será realizada automaticamente.

iSCSI Initiator – Linux

Por cada disco lógico a que o sistema se liga (login no target), surge no sistema um novo disco, por exemplo se o sistema apenas possuía o disco **/dev/sda** surgirá um novo disco **/dev/sbd.**

O sistema operativo trata este novo disco como se fosse um disco físico local, embora na realidade seja um disco lógico iSCSI. Se é um disco recém criado (no target) deverá ser usado o comando **fdisk** para definir as partições que depois poderão ser formatadas com o comando **mkfs.** Estando formatadas podem então ser montadas e eventualmente registadas no **/etc/fstab** para serem montadas sempre que o sistema arranca.

A identificação de disco local atribuída pode ser um problema quando são usados vários discos lógicos iSCSI. A atribuição de discos locais é feita pela ordem de ligação, manter consistentemente os mapeamentos entre identificadores de discos locais e targets exige mais algum trabalho. Por exemplo se um *target* não está disponível, os identificadores de disco local para os outros **targets** poderão ser diferentes.

Uma vantagem de se utilizar um protocolo standard como o iSCSI é a interoperabilidade, por exemplo *targets* Windows podem ser utilizados por *initiators* Linux e *targets* Linux podem ser usados por *initiators* Windows.

Administração de Sistemas (ASIST)

Aula Teórico Prática 06

Sistemas de ficheiros em rede (NAS) - SMB/CIFS e NFS
Filtragem de pacotes de rede – *Firewall*
Lightweight Directory Access Protocol (LDAP)
Linux – OpenLDAP

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Protocolos NAS – SMB/CIFS

De entre outros destacam-se dois protocolos de sistemas de ficheiros em rede: o **SMB/CIFS** e o **NFS**, originários respetivamente dos sistemas Windows e sistemas Unix. Atualmente quer servidores Windows quer servidores Linux podem usar ambos os protocolos.

SMB/CIFS (*Server Message Block/Common Internet File System*) refere-se a um protocolo desenvolvido ao longo do tempo pela Microsoft e que permite a partilha de sistemas de ficheiros através da rede. Efetivamente o protocolo é o **SMB**, **CIFS** refere-se ao dialeto mais recente suportado pelo **SMB**.

O SMB possui um mecanismo de negociação do dialeto, o que permite suportar clientes e servidores Windows de diferentes versões, nessa negociação os dois intervenientes determinam qual é o dialeto mais recente que é suportado por ambos. No SMB os dialetos são identificados por nomes, o mais antigo é o **PC NETWORK PROGRAM 1.0**, os mais recentes são o **DOS LANMAN 2.1** e o **NT LM 0.12**, este último corresponde ao CIFS.

O acesso a sistemas de ficheiros SMB/CIFS é orientado por sessão de utilizador, o utilizador autentica-se no servidor e obtém uma listas de partilhas disponibilizadas pelo servidor para esse utilizador. Além de pastas do sistema de ficheiros, também impressoras podem ser partilhadas. O acesso às partilhas é condicionado pelas permissões (ACLs) de acordo com o utilizador que se autenticou.

Claro que no contexto de um domínio Windows, depois de efetuar o login no domínio, o utilizador está desde logo autenticado em todos os servidores que são membros do domínio.

As partilhas de pastas via SMB/CIFS são identificadas através do formato **UNC** (*Universal Naming Convention*):

\\{NOME-SERVIDOR}\{NOME-PARTILHA}

Onde **{NOME-SERVIDOR}** representa o servidor que está a disponibilizar a partilha, podendo ser um nome simples NetBIOS, um nome DNS ou um endereço IP. **{NOME-PARTILHA}** é o nome que foi dado à partilha no servidor, por omissão as pastas são partilhadas usando o nome da pasta, no entanto esse nome pode ser alterado e/ou podem ser definidas várias partilhas para uma mesma pasta, cada uma com um nome diferente.

Cada partilha tem uma ACL associada que define as permissões de acesso dos utilizadores e grupos. As ACL de partilhas só possuem três permissões: **Full Control**, **Change** e **Read**, no entanto <u>as permissões definidas para a partilha não se sobrepõem às permissões no sistema de ficheiros NTFS</u>.

Por omissão quando se cria uma partilha é dada a permissão **Read** ao grupo implícito **Everyone**, mas isto não significa que todos os utilizadores tenham acesso, tudo depende das permissões NTFS definidas na pasta que está a ser partilhada. É mesmo uma prática corrente atribuir nas partilhas a permissão **Full Control** ao grupo **Everyone**, deste modo todo o controlo é feito pelas ACLs do NTFS.

Domínios Windows - partilha da *home* e *profile*

As contas de utilizadores Windows definem os atributos ***Profile Path*** e ***Home Folder***. No contexto de um domínio estes atributos têm de corresponder a partilhas de rede disponibilizadas por servidores do domínio, só dessa forma estarão acessíveis em todos os postos de trabalho que efetuam login no domínio.

As pastas de trabalho pessoais dos utilizadores (***Home Folder***) deverão ser colocadas numa pasta partilhada por um dos servidores do domínio, por exemplo \\SERVER\HOMES. O atributo ***Home Folder*** de cada conta de utilizador poderá então assumir a forma **\\SERVER\HOMES\{username}**, onde **{username}** é o nome do utilizador (admitindo que cada pasta têm um nome igual ao nome do utilizador).

A partilha **\\SERVER\HOMES** pode ser definida com a permissão **Full Control** para **Everyone**, as permissões efetivas serão ditadas pelas ACLs do NTFS.

Para o ***Roaming Profile*** (***Profile Path***), estratégia idêntica deve ser implementada, possivelmente usando outro servidor do domínio.

Numa configuração mais simples pode ser usada para ***Profile Path*** uma subpasta de ***Home Folder***, por exemplo:

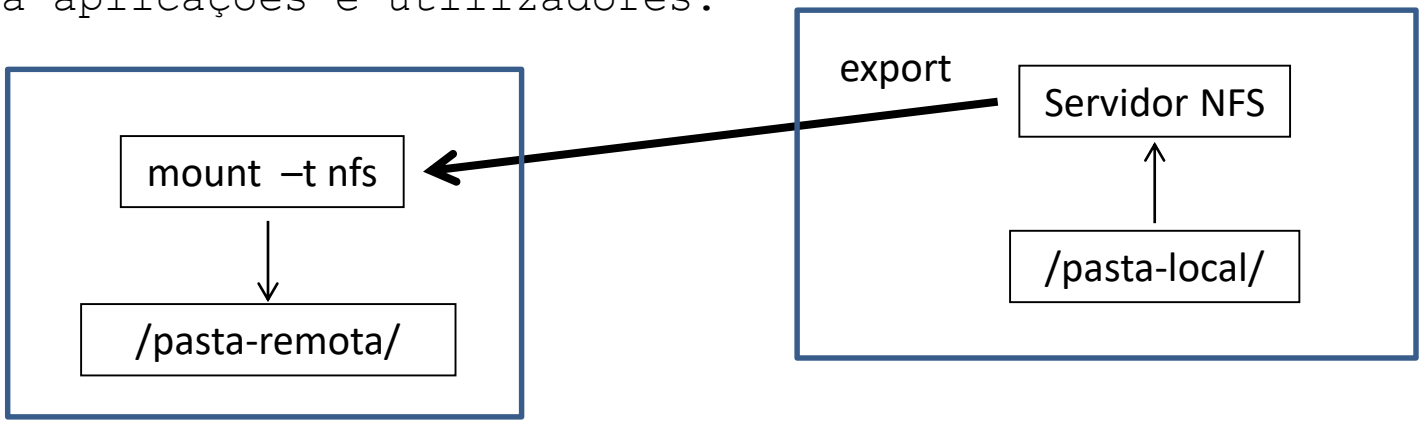
\\SERVER\HOMES\{username}\Profile\{username}

Protocolos NAS – NFS (Network File System)

O protocolo NFS é originário dos sistemas Unix, foi desenvolvido inicialmente pela *Sun Microsystems*. As implementações tradicionais NFSv2 e NFSv3 são totalmente vocacionadas para serem manuseadas exclusivamente pelos administradores dos sistemas, não existe suporte para autenticação de utilizadores.

No NFSv4 são suportadas novas formas de autenticação, incluindo a autenticação de utilizadores individuais.

Focalizando na utilização pelos administradores de sistemas, o NFS permite **exportar pastas locais** de uma máquina que atua como servidor NFS (o verbo exportar é razoavelmente equivalente a partilhar). Uma pasta exportada por NFS pode ser **montada por outra máquina** (cliente NFS) e desta forma fica integrada no sistema de ficheiros e disponível para aplicações e utilizadores.



Unix – NFSv2 e NFSv3 – NFS Server

Nos sistema Unix o servidor NFS é configurado através do ficheiro **/etc/exports**, neste ficheiro definem-se as pastas locais que são exportadas e em que condições.

No NFSv2 e NFSv3 o controlo de acesso às pastas exportadas é realizado através do endereço IP do cliente, assim no ficheiro **/etc/exports**, para cada pasta exportada são enumerados os endereços IP dos clientes aceites. O endereço IP pode ser especificado de várias formas, incluindo um endereço IP, uma rede IP (endereço da rede e prefixo), um nome que seja resolvido para um endereço IP e através de metacaracteres, por exemplo ***** corresponde a qualquer endereço.

Para cada exportação podem ser definidas várias opções, incluindo leitura apenas (**ro**), leitura e escrita (**rw**). Por omissão as exportações usam sempre a opção **root_squash**, isto significa que o utilizador **root** da máquina cliente não terá sobre o sistema de ficheiros exportado as mesmas permissões que o utilizador **root** da máquina servidora possui.

Os restantes utilizadores terão exatamente as mesmas permissões tanto no servidor como no cliente, desde que os UID sejam os mesmos.

Exemplo de ficheiro

/etc/exports:

```
/users/home        190.130.60.0/255.255.255.0(ro)
/users/backups     vsrv2(rw)   *(ro)
/usr/local/dump    190.130.60.10(rw)      190.130.60.123(rw)
```

Unix – NFSv2 e NFSv3 – Cliente NFS

Os clientes NFS autorizados podem montar os sistemas de ficheiros exportados pelo servidor usando o comando **mount:**

mount –t nfs {SERVER}:{PASTA} {PONTO-DE-MONTAGEM}

Onde **{SERVER}** representa o endereço IP ou nome DNS do servidor NFS, **{PASTA}** representa o nome da pasta exportada (tal como consta no ficheiro /etc/exports do servidor) e **{PONTO-DE-MONTAGEM}** representa uma pasta local vazia onde será disponibilizada a pasta remota.

Para tornar a configuração permanente, a montagem deve ser adicionada ao ficheiro **/etc/fstab** para que seja realizada no arranque da máquina.

Todos os atributos do sistema de ficheiros são exportados, incluindo o UID (owner), GID e permissões associados a cada objeto.

Consequentemente as permissões dos utilizadores da máquina cliente sobre o sistema de ficheiros montado são exatamente as mesmas que possuem no servidor, só é necessário garantir que o UID e GID de cada utilizador e grupo é exatamente o mesmo em todas as máquinas.

Garantir que o UID e GID é o mesmo em todas as máquinas é facilmente conseguido se todas as máquinas utilizarem um mesmo repositório de contas de utilizadores e grupos centralizado, por exemplo LDAP.

Autenticação por endereço IP – NFSv2 e NFSv3

A autenticação por endereço IP só pode ser considerada segura se for aplicada com algumas precauções:

- **Tem de haver confiança total no administrador do nó cliente.** Pelo facto de serem usados números de porto reservados (<1024), tanto o cliente como o servidor sabem que estão a dialogar com uma aplicação sob o controlo do administrador do nó. Particularmente quando o servidor recebe um pedido de um cliente verifica que provém de um endereço IP autorizado e de um número de porto reservado.

- **Tem de haver confiança no funcionamento da rede.** O sistema é totalmente vulnerável a ataques de *IP Spoofing*, consequentemente só deve ser usado numa rede a que os utilizadores não têm acesso, tipicamente uma rede onde apenas existem servidores, uma DMZ (*DeMilitarized zone*).

- **Tem de haver confiança no funcionamento da resolução de nomes.** No caso de se usarem nomes de máquinas em lugar de endereços, o sistema poderia ser atacado através do DNS. Se o servidor DNS for comprometido poderá dar respostas falsas e desse modo levar o servidor NFS a permitir os endereços errados. Como medida de segurança, no contexto do NFS, é aconselhável usar sempre endereços IP e não nomes DNS de máquinas.

Utilização do NFS para partilha das *homes*

Num ambiente em que existe um conjunto de servidores Linux, a utilização de um repositório de contas de utilizadores e grupos centralizado, por exemplo, numa base de dados acessível por LDAP, permite que os utilizadores e grupos sejam iguais em todos os servidores.

Consequentemente os utilizadores podem efetuar o login em qualquer dos servidores, o NFSv2/NFSv3 pode então ser usado para igualmente disponibilizar as ***home directories*** em todos os servidores.

As ***home directories*** ficarão alojadas numa pasta de um servidor que é exportada por NFS para todos os servidores onde as *home directories* são necessárias. Como é evidente a pasta contendo as *home directories* deverá ser exportada em ***read-write***. Como o NFS mantém todas as propriedades dos objetos, nomeadamente UID/GID associados e permissões, os utilizadores terão em qualquer dos servidores as permissões corretas sobre a respetiva *home directory*.

Nos servidores que recebem a pasta exportada é necessário ter o cuidado de montar a pasta exportada num local que permita que o atributo *home directory* da conta de utilizador corresponda à efetiva localização da pasta.

Eventualmente se não corresponderem também será simples resolver o problema através de uma ligação simbólica.

Filtragem de pacotes - firewall

Designam-se por ***firewalls*** mecanismos integrados nos nós de rede que se destinam a condicionar a passagem de tráfego de rede.

Num nó intermédio como um ***router*** permite controlar o tráfego que é transferido entre diferentes redes. Num nó final como um servidor ou um posto de trabalho permite condicionar o tráfego que entra (***inbound***) e sai (***outbound***) nesse nó.

No caso de um nó final, todas as atenções se concentram no tráfego ***inbound*** pois assume-se que as ameaças provêm da rede e não do interior do nó.

Existem vários formas de analisar o tráfego de rede correspondendo a diferentes designações para o tipo de firewall:

- A análise individual de cada pacote (designados ***stateless***).
- A análise das transações completas ao nível dos protocolos de transporte ***connection-oriented*** e dos protocolos de aplicação (designados ***stateful***).

Windows Server - Windows Firewall with Advanced Security

As regras *stateless* dividem-se em ***Inbound Rules*** e ***Outbound Rules***, respetivamente aplicadas ao tráfego que provém da rede e tráfego que o nó pretende emitir.

Os pacotes que chegam da rede ou que se pretendem emitir para a rede são confrontados com as listas de regras, respetivamente *Inbound* e *Outbound*.

Cada regra define critérios relativos a quais os pacotes que lhe correspondem e a ação a desencadear em caso de correspondência: ***Allow***, ***Block*** e ***Allow if secure***. No último caso, os critérios para considerar ***secure*** podem ser definidos para cada regra.

O tráfego **inbound** só é permitido se corresponder a uma regra ***Allow*** e não corresponder a nenhuma regra ***Block*** (nas ***Inbound Rules***).

O tráfego ***outbound*** só é bloqueado se corresponder a uma regra ***Block*** (nas ***Outbound Rules***).

Como se conclui a aplicação das regras ***inbound*** é muito mais restritiva dos que a aplicação das regras ***outbound***.

No caso do **inbound**, não havendo nenhuma correspondência o pacote é **bloqueado**. No caso do ***outbound***, não havendo nenhuma correspondência o pacote é **permitido**.

Para cada regra define-se os critérios de correspondência com pacotes, podem ser baseados em:

- **Programs and Services** - aplicações locais às quais os pacotes se destinam (*inbound*) ou nas quais os pacotes têm origem (*outbound*).
- **Protocols and Ports** – protocolo (TCP, UDP, ICMPv4, …) e, quando aplicável, números de porto de origem e destino (TCP e UDP) ou tipo de mensagem ICMP.
- **Scope** – endereços de nó de origem e destino do pacote (IPv4 e/ou IPv6).
- **Local Principals** – utilizadores locais (a quem pertencem as aplicações que vão receber o pacote ou emitiram o pacote).
- **Remote Users** – utilizadores remotos (do *Active Directory*).
- **Remote Computers** – máquinas remotas (do *Active Directory*).

Também é possível condicionar a aplicação das regra ao contexto de rede em que a máquina se encontra (***Profile***):

- **Domain** – se a máquina está ligada a uma rede que pertence ao domínio.
- **Private** – se a máquina está ligada a uma rede que foi classificada como privada.
- **Public** – se a máquina está ligada a uma rede que foi classificada como pública.

Linux – Netfilter e comandos iptables

O **Netfilter** é um sistema de controlo de tráfego integrado no *kernel* Linux, permite entre outras funcionalidades definir regras relativamente aos pacotes de rede que são permitidos.

Para gerir o Netfilter o administrador usa os comandos **iptables** e **ip6tables**, respetivamente para regras relativas ao IPv4 ou IPv6.

Através do iptables podem ser geridas 5 tabelas: **filter**, **nat**, **mangle**, **raw** e **security**.

A tabela **filter** é usada para definir critérios de permissão (**ACCEPT**) ou não permissão (**DROP**) de pacote. A tabela **nat** é usada para definir regras de manipulação de endereços (***Network Address Translation***), de origem (SNAT) e de destino (DNAT),

Ao utilizar o comando **iptables** podemos indicar a tabela que pretendemos manipular através da opção **-t**, por omissão a tabela usada é **filter**.

Em cada tabela existem cadeias de regras (***chains***). Cada tabela possui algumas cadeias pré-definidas (*built-in*), mas podem ser criadas outras cadeias. Cada cadeia pré-definida aplica-se a pacotes em determinadas situações.

Em cada cadeia, as regras (rules) são numeradas a partir de 1 (primeira regra). Cada pacote aplicável à cadeia é confrontado sequencialmente com cada regra da cadeia.

Table	Built-in chains	Objetivos
filter	INPUT	Sequência de regras ACCEPT/DROP aplicadas aos pacotes quando chegam da rede e se destinam ao nó.
	FORWARD	Sequência de regras ACCEPT/DROP aplicadas aos pacotes quando chegam da rede, mas não se destinam ao nó. Utilizada quando o nó opera como **router**.
	OUTPUT	Sequência de regras ACCEPT/DROP aplicadas aos pacotes que vão ser emitidos pelo nó e têm origem no nó.
nat	PREROUTING	Alterações de endereços a aplicar a pacotes que chegam ao nó, (sejam ao não destinados ao nó), antes do pacote ser encaminhado.
	OUTPUT	Alterações de endereços a aplicar a pacotes emitidos pelo nó (têm origem no nó) antes de serem encaminhados.
	POSTROUTING	Alterações de endereços a aplicar a pacotes que vão ser emitidos pelo nó (tenham ou não origem no nó), após o pacote ser encaminhado.

Cada regra de uma cadeia define **critérios de correspondência** com pacotes e a **ação a realizar** (**target**) em caso de correspondência. Não correspondendo com a regra o pacote será confrontado com a regra seguinte da cadeia.

Se um pacote atinge o fim de uma cadeia, será executada sobre ele a ação definida na **política da cadeia** (apenas para cadeias *built-in*).

iptables - targets

Cada regra define a ação a realizar em caso de correspondência de um pacote através das opções **–j target** ou **–g chain.**

Ação	Descrição da ação sobre o pacote
-j ACCEPT	Deixar passar o pacote, termina o processamento da cadeia para o pacote.
-j DROP	Bloqueia (destrói) o pacote, termina o processamento da cadeia para o pacote.
-g {chain}	Processa a cadeia identificada por {chain}. Para além das *built-in* é possível criar novas cadeias de regras. Usando a opção **–g** o pacote será confrontado com as regras da cadeia designada.
-j RETURN	Termina o processamento da cadeia atual e regressa à cadeia de origem para continuar o processamento das regras seguintes. Se a cadeia atual é *built-in* aplica ao pacote a politica da cadeia.
-j SNAT	Alteração do endereço de origem (IP e/ou porto) (através da opção **--to-source**). Apenas é válido na cadeia **POSTROUTING** da tabela **nat**. Termina o processamento da cadeia para o pacote.
-j DNAT	Alteração do endereço de destino (IP e/ou porto) (através da opção **--to-destination**). Apenas é válido nas cadeias **PREROUTING** e **OUTPUT** da tabela **nat**. Termina o processamento da cadeia.
-j LOG	Regista o pacote nos logs do sistema. Não termina o processamento da cadeia, continua a processar o pacote através da regra seguinte da cadeia.

iptables - critérios de correspondência

Cada regra define **critérios de correspondência** com pacotes, os critérios são especificados através de opções. Sendo definidos vários critérios, **a regra apenas será aplicada a pacotes para quais todos os critérios se verificam.** Os critérios podem ser negados se a opção for precedida pelo símbolo **!**.

Critério	Descrição do critério
-i {iface}	O pacote foi recebido através da interface de rede {iface}.
-o {iface}	O pacote vai ser emitido através da interface de rede {iface}.
-p {proto}	O pacote transporta dados do protocolo designado {proto}. O valor de {proto} pode ser por exemplo **tcp**, **udp**, **icmp** ou **icmpv6**. O valor **all** representa qualquer protocolo, é equivalente a não definir este critério.
-s {address}[/{mask}]	O pacote tem como endereço de origem os endereços especificados, se for omitida /{mask} o endereço é interpretado com sendo um único nó, ou seja mask 32 ou 128 bits respetivamente para IPv4 e IPv6.
-d {address}[/{mask}]	Idêntica à opção –s, mas especificando o endereço de destino do pacote.
--dport {port}[:port2]	O número de porto de destino do pacote é {port}, ou está dentro do intervalo [{port}, {port2}] (apenas válido para **–p udp** ou **–p tcp**).
--sport {port}[:port2]	O número de porto de origem do pacote é {port}, ou está dentro do intervalo [{port}, {port2}] (apenas válido para –p udp ou –p tcp).

iptables – definição das cadeias de regras

Comando	Descrição do comando
-N {chain}	Cria uma nova cadeia.
-X {chain}	Elimina uma cadeia anteriormente definida (não aplicável para cadeias build-in).
-F [{chain}]	Elimina todas a regras da cadeia, se a cadeia for omitida elimina todas as regras de todas as cadeias da tabela.
-P {built-in-chain} {target}	Define a política da cadeia, apenas válido para cadeias built-in, os targets válidos são ACCEPT e DROP.
-A {chain} {rulespec}	Acrescenta uma regra (append) à cadeia designada. {rulespec} é uma definição de regra, incluindo critérios e target.
-I {chain} [{num}] {rulespec}	Insere na cadeia uma nova regra na linha {num}, se {num} não é especificado insere na primeira posição.
-R {chain} {num} {rulespec}	Substitui a regra que se encontra na posição {num} da cadeia.
-D {chain} {num}	Elimina a regra que se encontra na posição {num} da cadeia.
-L [{chain}]	Apresenta a lista das regras na cadeia designada ou de todas as cadeias da tabela. A opção **–n** apresenta todos os endereços e portos na forma numérica. A opção **–v** apresenta mais detalhes. A opção **--line-numbers** apresenta a numeração das regras.

iptables - módulos

A opção **–m** permite carregar módulos para suportar funcionalidades adicionais. Por exemplo o módulo **state** (**-m state**) permite definir o critério de correspondência **--state {estados}**, onde *estados* é uma lista separada por virgulas de valores tais como: **ESTABLISHED**, **NEW** e **RELATED**. O critério **ESTABLISHED** aplica-se a tráfego TCP relativo a conexões já estabelecidas, **NEW** a novas conexões TCP. O critério **RELATED** aplica-se a tráfego UDP de resposta, ou seja pacotes que correspondem a respostas de pedidos UDP anteriormente realizados.

Linux – *routing*

Por omissão, o *kernel* Linux rejeita pacotes que não lhe sejam destinados (não encaminha tráfego entre redes). Esse comportamento pode ser alterado modificando as variáveis de sistema **net.ipv4.ip_forward** e **net.ipv6.conf.all.forwarding** para o valor **1**, respetivamente para IPv4 e IPv6. O nó passará a operar como *router*.

Para tornar estas configurações permanentes (aplicadas sempre que o sistema arranca) devem ser definidas no ficheiro de configuração **/etc/sysctl.conf**.

LDAP (*Lightweight Directory Access Protocol*)

O protocolo LDAP foi concebido para permitir um acesso expedito a serviços de informação através da rede, tipicamente consultas e pesquisa de informação.

Sob o ponto de vista do LDAP a informação é guardada em objetos e organizada numa estrutura em árvore (***DIT - Directory Information Tree***). Cada objeto é a instanciação de uma classe, os objetos contêm atributos onde os dados propriamente ditos são armazenados.

A classe (***LDAP objectClass***) define os atributos dos objetos, tais como:

- Quais são os atributos e o seu nome.

- Que tipo de dados um atributo pode guardar (que valores pode assumir).

- Se o atributo é obrigatório ou opcional.

- Se o atributo pode ser repetido várias vezes no objeto ou tem de ser único.

As classes de objetos são normalizadas e não devem ser definidas arbitrariamente, as classes são definidas através do ***schema***.

Existe uma grande variedade de classes disponíveis para os mais diversos tipos de aplicação.

LDAP - classes e objetos

A definição de classes (no *schema*) recorre à herança, ou seja pode definir-se uma nova classe a partir de uma classe existente e acrescentar novos atributos. Existem 3 tipos de classe:

ABSTRACT – é uma classe de topo definida com o objetivo de ser herdade por classes concretas. Não se podem criar objetos com este tipo de classes.

STRUCTURAL – é uma classe com a qual se podem criar objetos, um objeto tem de pertencer a uma classe estrutural, mas não pode pertencer a mais do que uma.

AUXILIARY – é uma classe com o propósito de acrescentar atributos a um objeto que já pertence a uma classe estrutural.

Um objeto tem de pertencer a uma e uma só classe estrutural, mas pode pertencer a várias classes auxiliares.

Exemplo de parte de um *schema*:

```
objectclass ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL
  DESC '2 character iso 3611 assigned country code'
  MUST c
  MAY ( searchGuide $ description ) )
```

LDAP – DN (*Distinguished Name*) e root DN

Cada objeto na DIT é identificado de forma única através do DN. No seio do ramo da DIT em que se encontra um objeto é identificado de forma única através do valor de um dos seus atributos esse é o **RDN** (*Relative Distinguished Name*) do objeto. O DN é constituído pelo RDN seguido do DN do ramo (DN do objeto pai). Os vários elementos do DN e RDN são separados por uma virgula.

A raiz da DIT (topo da hierarquia) tem ela própria um DN que é definido pelo administrador (**root-DN**). Qualquer objeto de uma DIT terá necessariamente incluído no seu DN o root-DN da DIT.

Segundo as normas em vigor, o **root-DN** deve corresponder ao nome DNS do domínio local separado em componentes através do atributo **dc** (*Domain Component*):

dc=sub-domínio,dc=sub-domínio,…,dc=domínio-de-topo

Por exemplo para o domínio DNS *dei.isep.ipp.pt*, o **root-DN** da DIT deve ser: **dc=dei,dc=isep,dc=ipp,dc=pt**

O objeto com **dn: cn=user1,ou=utilizadores,dc=dei,dc=isep,dc=ipp,dc=pt** fica perfeitamente identificado, tem como RDN **cn=user1** e está localizado no ramo **ou=utilizadores,dc=dei,dc=isep,dc=ipp,dc=pt.**

Linux – Servidor OpenLDAP (http://www.openldap.org)

Um dos servidores LDAP mais usado em Linux é o OpenLDAP. Tradicionalmente a configuração é realizada através do ficheiro **slapd.conf**, mas em distribuições recentes como o Ubuntu atual, a configuração é definida numa DIT do próprio servidor com root-DN **cn=config**.

Alguns dos elementos base de configuração são:

- Os **schema** a suportar (classes de objetos).
- **Backend** – a base de dados onde os objetos são armazenados, o mais comum é **bdb** (*Oracle Berkeley DB*), mas também pode usar **SQL**.
- **Suffix** – refere-se ao root-DN da DIT.
- **Conta de administração** – identifica um objeto na DIT que corresponderá ao administrador LDAP, o objeto deve conter o atributo **userPassword**.
- **Atributos a indexar** - para agilizar a pesquisa na árvore, os atributos mais relevantes devem ser indexados, desde logo o atributo **objectClass**. Para contas de utilizadores outros atributos como **cn**, **uid**, **uidNumber** e **gidNumber** devem também ser indexados.

Uma vez definida as configurações base, a conta de administração pode ser usada para gerir os objetos a partir da root-DN definida e, se aplicável, gerir a configuração residente na DIT **cn=config**.

LINUX - LDAP – ficheiros LDIF

Estabelecida a conta de administração, o LDAP pode então ser usado para gerir os objetos nas DIT, para esse efeito podem ser usados os comandos:

ldapadd – adicionar um objeto.

ldapdelete – eliminar um objeto.

ldapmodify – adicionar, eliminar ou alterar um objeto (atributos), incluindo alterar o nome (**DN**). Note-se que alterar o DN pode incluir mover o objeto de um ramo para outro (**moddn**) ou não (**modrdn**).

Estes comandos aceitam como argumento ficheiros em formato **LDIF** (*LDAP Data Interchange Format*) com instruções das ações a realizar usando uma sintaxe específica, por exemplo:

```
dn: cn=user1,dc=example,dc=com
        changetype: modify
        replace: mail
        mail: modme@example.com
        -
        add: title
        title: Grand Poobah
        -
```

Se um ficheiro com este conteúdo for fornecido ao comando **ldapmodify** iria alterar o conteúdo do atributo ***mail*** e acrescentar o atributo ***title*** no objeto com DN **cn=user1,dc=example,dc=com.**
Cada ação definida deve terminar com uma linha constituída pelo símbolo **-.**

Administração de Sistemas (ASIST)

Aula Teórico Prática 07

Utilizadores e grupos no *Active Directory*
Políticas no *Active Directory*
Relações de confiança no *Active Directory*

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Active Directory - LDAP

A estrutura lógica de domínios *Active Directory (domains, domain trees, forests)* é sustentada em bases de dados residentes em cada domínio (replicada entre os vários DC do mesmo domínio).

O serviço de diretoria usado no acesso a essas bases designa-se *Active Directory* (**AD**), que usa como base o protocolo LDAP. Cada domínio AD é identificado por um nome DNS, correspondendo ao **root-DN** da DIT do domínio. Tratando-se de uma DIT LDAP, o domínio está organizado em ramos.

Objetos como utilizadores e grupos estão alojados em ramos da DIT, alguns ramos são definidos pelo sistema, tais como **Builtin**, **Computers** e **Users** (são da classe *Container*). O administrador pode definir novos ramos criando objetos da classe ***Organizational Unit***.

Muitas das ferramentas de gestão do *Active Directory* protegem o utilizador de interagir diretamente com o LDAP, nomeadamente não identificando os objetos através do *distinguished name* (**DN**).

Num domínio AD, dois objetos podem ter o mesmo **RDN** desde que o **DN** seja diferente, ou seja, <u>dois objetos podem ter o mesmo nome desde que estejam em ramos diferentes</u>.

Sob o ponto de vista de administração, a divisão do domínio em ramos tem várias vantagens, mas a necessidade de referir a localização de cada objeto dentro do domínio acaba por ser um inconveniente.

AD – identificação de utilizadores e grupos

Os domínios Windows anteriores ao AD, sob o ponto de vista de nomeação de objetos eram completamente rasos.

Os utilizadores e grupos são identificados na forma **DOMAINNAME\USERNAME** e **DOMAINNAME\GROUPNAME**, onde *DOMAINNAME* é o nome *NetBIOS* do domínio (nome não qualificado do domínio AD).

Esta forma continua a ser suportada nos domínios AD, tanto para grupos como utilizadores, no caso dos utilizadores é mesmo a forma mais usada para identificação no início de sessão (***Down-Level Logon Name***).

No entanto, esta forma de identificar utilizadores e grupos não é suficiente no AD porque dentro do domínio podem existir vários grupos e utilizadores com o mesmo nome.

Para resolver este problema no AD cada conta de grupo possui um nome **Pre-Windows 2000** para ser usado na forma **DOMAINNAME\nome-grupo**. Normalmente este nome é igual ao nome da conta de grupo, no entanto tem de ser único em todo o domínio.

O mesmo se passa no caso das contas dos utilizadores, o nome **Pre-Windows 2000** é habitualmente usado no login na forma **DOMAIN\username** (*Down-Level Logon Name*), por omissão é igual ao nome da conta. Dois utilizadores podem ter contas com o mesmo nome no AD do domínio, mas o *Down-Level Logon Name* de cada um deles terá de ser diferente.

AD – *User principal name* (UPN)

A forma mais atual de identificar um utilizador no AD é na forma UPN, também designado *User Logon Name* porque é usado no *logon*:

username@AD-DNS-DOMAIN

Por omissão o UPN está definido com o nome da conta do utilizador (RDN do objeto) e o nome DNS do domínio AD. Num domínio não podem existir dois UPN iguais, se existirem duas contas de utilizadores com o mesmo nome um deles terá de usar um UPN diferente do nome da conta.

Uma conta de utilizador num domínio é um objeto LDAP identificado por um DN, o atributo **cn** (*Common Name*) é usado no RDN, o seu valor é o nome da conta, por exemplo **cn=specialuser**.

Neste exemplo o UPN será **specialuser@AD-DNS-DOMAIN** e o nome Pre-Windows 2000 será **DOMAIN\specialuser**. Estes serão os valores por omissão, podem ser alterados pelo administrador se for necessário ou conveniente.

Também as contas de grupo são identificadas pelo atributo **cn** (*Common Name*) correspondendo ao nome da conta. Por isso, uma conta de utilizador e uma conta de grupo podem ter o mesmo nome apenas se tiverem em ramos diferentes, mas os nomes Pre-Windows 2000 terão de ser necessariamente diferentes.

AD – grupos

As contas de grupo num domínio AD são bastante diferentes das contas de **grupos locais** que apenas existem na máquina local. Num domínio as contas de grupo são armazenadas na base AD do domínio e são válidas em todo o domínio.

No AD os grupos são objetos LDAP identificados por um DN, dentro da base de dados do domínio (DIT) podem estar localizadas num qualquer ramos, o atributo **nome Pre-Windows 2000** (*DOMAIN\group*) permite identificar o grupo no domínio sem a necessidade de identificar o ramo. Ao contrário do que acontece com um grupo local, os grupos do AD podem ter como membros outros grupos (group *nesting*).

As máquinas que são membros de um domínio podem continuar a trabalhar com utilizadores e grupos locais, no entanto nos *domain controllers* não existem contas locais.

Existem dois tipos de grupo no AD:

Security group – são objetos de segurança, podem ser utilizados para atribuir direitos aos seus membros e podem ser incluídos em ACLs.

Distribution group – destinam-se a criar grupos lógicos de utilizadores e grupos, com outros objetivos. Não permite atribuir diretos aos membros nem podem ser usados em ACLs. Podem servir por exemplo para criar listas de distribuição de correio eletrónico. Note-se que um *security group* também pode ser usado para distribuir correio.

AD – *group scope*

O scope do grupo define a sua abrangência: em termos de membros que pode ter, grupos de que pode ser membro, e permissões ou direitos que lhe podem ser associados.

Local groups (Local scope) – existem apenas localmente na máquina (não estão no AD). Podem incluir membros de toda a floresta e outros domínios de confiança (*trusted domains*). Os direitos atribuídos ao grupo apenas podem ser relacionados com os recursos locais da máquina onde são definidos. Um *local group* não pode ser membro de outros grupos.

Domain local groups (Domain local scope) – existem no domínio AD. Podem incluir como **membros:** utilizadores e grupos globais de toda a floresta e outros domínios de confiança (*trusted domains*), grupos universais da floresta e grupos *domain local* do mesmo domínio. **Podem ser membros de:** grupos *domain local* do mesmo domínio e grupos locais de máquinas do mesmo domínio (excluindo grupos built-in). Os direitos atribuídos ao grupo podem dizer respeito a qualquer objeto ou recurso existente no domínio, mas não recursos de outros domínios.

AD – *Global and Universal groups*

Global groups (Global scope) – existem no domínio AD. **Membros possíveis:** utilizadores e grupos globais do mesmo domínio. No entanto um *global group* pode ser **membro de** grupos universais da mesma floresta, grupos globais do mesmo domínio e grupos locais definidos em máquinas de domínios de confiança. É possível atribuir a um *global group* permissões sobre qualquer domínio da floresta ou outros de confiança.

Universal groups (Universal scope) – existem no domínio AD. **Membros possíveis:** utilizadores da floresta, grupos globais e universais da mesma floresta. **Podem ser membros de:** grupos universais da floresta, grupos *domain local* de domínios de confiança (inclui todos os domínios da floresta) e grupos locais de máquinas de domínios de confiança. Os direitos atribuídos ao grupo podem dizer respeito a qualquer objeto ou recurso existente na floresta e outros *trusted domains*.

Um grupo **universal** pode ser convertido para ***domain local*** ou **global**.

Um grupo **global** pode ser convertido para **universal**.

Um grupo ***domain local*** pode ser convertido para **universal**.

Estas conversões apenas são válidas se não violarem as regras anteriormente enunciadas.

Organizational Units (OUs)

No contexto de um domínio AD objetos como utilizadores, computadores e grupos encontram-se em ramos do LDAP. Embora existam inicialmente definidos no AD alguns ramos como **Builtin**, **Users** e **Computers** que são do tipo ***Container***, podem ser definidos novos ramos através de objetos do tipo *Organizational Unit* (**OU**).

As OU permitem subdividir um domínio em áreas lógicas numa estrutura em árvore (uma OU pode conter outras OUs). Os objetos como utilizadores e grupos podem ser movidos de uma OU para outra OU dentro do mesmo domínio.

Existem várias vantagem em definir OUs. Sob o ponto de vista de organização tornam-se mais evidente para os administradores relações de afinidade entre subconjuntos de utilizadores e grupos estando organizados em OUs. Inicialmente está definida por exemplo a OU **Domain Controllers** que contém objetos do tipo **Computer** correspondentes aos vários DCs do domínio.

As OUs permitem:

- **Delegar poderes de administração** – para uma dada OU o administrador pode delegar a respetiva gestão a determinados utilizadores e grupos, definindo exatamente que poderes são delegados.
- **Definir políticas de grupo** (**GPO** – *Group Policy Objects*) a aplicar à OU e consequentemente a todos os objetos que estão na OU, como por exemplo contas de utilizadores.

Nomeação de gestores de grupos e OUs

No caso de uma OU a forma recomendada para atribuir poderes de gestão a utilizadores ou grupos sobre a OU e objetos nela residentes é através da delegação (***Delegate Control***).

No entanto objetos das classes **Group**, **Computer** e **Organizational Unit** possuem o atributo **managedBy** que identifica o utilizador ou grupo que é responsável pela sua gestão. Esta constitui uma via alternativa de atribuir direitos de administração sobre uma OU, que no entanto parece apresentar alguns problemas em situações específicas.

Políticas

Uma política é um **conjunto de elementos de configuração**, inclui elementos de configuração relativos ao utilizador e relativos ao computador. Os elementos de configuração podem ser tão diversos como entradas no **Start Menu** e **Desktop**, chaves no **registry** (do utilizador e do computador), configurações de firewall e permissões (***user rights***) a atribuir ao utilizador.

Um elemento de configuração pode ou não estar definido numa política:

- Se estiver definido é aplicado ao utilizador/computador **eliminando definições anteriores** (originais, ou resultantes de aplicações anteriores de outras políticas).
- Se não tiver definido não afeta o valor corrente desse elemento.

Aplicação de políticas - Local Policy

As políticas são aplicadas no momento em que o utilizador realiza o login (quando de torna membro do grupo **Authenticated Users**).

No entanto, as políticas afetam não apenas o utilizador, mas também a configuração do computador onde ele efetua o login.

No caso de um login local, usando uma conta local do computador, existe uma única política aplicada designada **Local Policy** que está armazenada localmente no computador e pode ser gerida com a ferramenta **Local Security Policy.**

No caso de um login num domínio, a **Local Policy** definida no computador utilizado é aplicada em primeiro lugar, mas segue-se a aplicação de políticas definidas no *Active Directory* que consequentemente se vão sobrepor à **Local Policy.**

No contexto de um domínio *Active Directory* as políticas estão armazenadas em *Group Policy Objects* (**GPO**) existem alguns GPO definidos, mas podem ser criados outros.

Num domínio os GPO podem ser geridos através da ferramenta **Group Policy Management.** Um GPO é um conjunto de elementos de configuração, para que seja aplicado durante o login dos utilizadores é necessário estabelecer ligações entre objetos apropriados do domínio e o GPO.

Aplicação dos GPO (*Group Policy Objects*)

Quando um utilizador efetua o login no domínio são inicialmente aplicadas as Local Policies, seguem-se as políticas definidas nos GPO do *Active Directory*.

Uma vez definidos os GPO, três tipos de objetos do AD podem ser ligados (*linked*) a eles: **Site**, **Domínio** e **OUs**.

Diferentes Sites, Domínios ou OUs podem ser ligadas ao mesmo GPO, também, um Site, Domínio ou OU pode ser ligado a vários GPO.

A ordem de aplicação dos GPO é notoriamente relevante porque um GPO sobrepõe-se aos GPO aplicados anteriormente (apenas para os elementos de configuração que define).

A ordem de aplicação segue a hierarquia do *Active Directory*: primeiro são aplicados os GPO aos quais o **Site** está ligado, depois os GPO aos quais o **domínio** está ligado e finalmente os GPO aos quais **cada OU** está ligada (seguindo a hierarquia entre diferentes OUs).

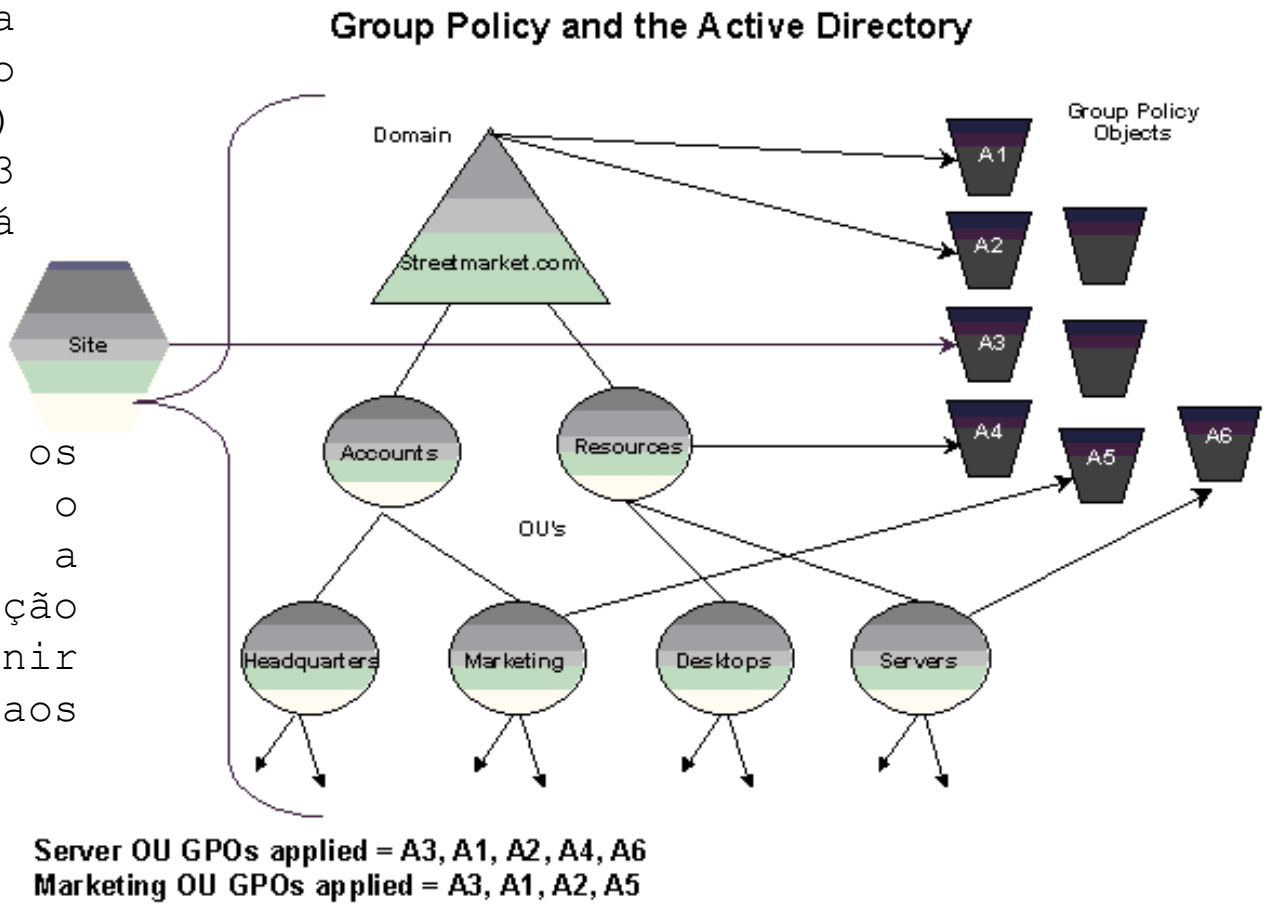
Quando um Site, domínio ou OU está ligado a vários GPO podemos definir exatamente a ordem de aplicação do vários GPO (***link order***).

Aplicação dos GPO - (*Group Policy Objects*)

A figura abaixo (*https://technet.microsoft.com*) apresenta um exemplo de aplicação de políticas no *Active Directory*.

Depois de aplicada a *Local Policy* (não representada na imagem) é aplicado o GPO A3 porque o Site está ligado a ele.

A seguir são aplicadas os GPO A1 e A2 porque o domínio está ligado a eles, a ordem de aplicação é estabelecida ao definir as ligações do domínio aos GPO (*link order*).



Aplicação dos GPO - (*Group Policy Objects*)

Os GPO são aplicadas a objetos do tipo **User** e **Computer**, não a objetos do tipo **Group**. Por outras palavras, os GPO aplicados a um utilizador são estabelecidos apenas pela conta de utilizador (nomeadamente a OU em que está definida) e **não pelos grupos de que ele é membro.**

Concretamente, se um utilizador é membro de um grupo e esse grupo está definido numa OU que está ligada a um GPO, então isso não é motivo para que esse GPO será aplicado ao utilizador durante o login.

Um cenário diferente é aquele em que o utilizador se encontra definido numa OU ligada a um GPO e esse GPO atribui a um grupo determinadas permissões (*user rights*). Neste caso o GPO será aplicado ao utilizador durante o login. Consequentemente, se o utilizador é membro do referido grupo terá as permissões que foram atribuídas ao grupo.

Active Directory Trust Relationships

As relações de confiança entre domínios podem ser estabelecidas nos dois sentidos (***two-way trust***) ou apenas num sentido (***one-way trust***).

Uma relação de confiança *one-way trust* pode ser usada quando temos interesse que um domínio possa aceder aos recursos de outro domínio, mas não o inverso.

As relações de confiança podem ser transitivas (***transitive trust***) ou não transitivas (**non*transitive trust***). Ser transitiva significa que a confiança se estende entre domínios, por exemplo: se o domínio A confia no domínio B (A-->B) e o domínio B confia no domínio C (B-->C) então o domínio A confia mo domínio C (A-->C):

A-->B-->C => A-->C

Numa floresta existe implicitamente uma relação de confiança ***two-way transitive trust*** entre um domínio e o domínio hierarquicamente superior (pai). Por ser transitiva a confiança estende-se a todos os domínios da floresta, assim entre quaisquer dois domínios de uma mesma floresta existe sempre uma ***two-way trust***.

Para além das relações de confiança estabelecidas desta forma entre domínios da mesma floresta podemos definir outras relações de confiança, desde que se possuam as permissões de administração necessárias nos domínios e florestas envolvidos.

Active Directory – External Trust and Forest Trust

Uma **external trust** pode ser estabelecida entre um domínio de uma floresta e outro domínio de outra floresta. Pode ser **one-way** ou **two-way**, mas é **non*transitive***. Pelo facto de ser não transitiva não se propaga a outros domínios das florestas e fica confinada aos dois domínios entre os quais é estabelecida. De acordo com os sentidos em que é estabelecida permite aos utilizadores de um domínio aceder aos recursos do outro domínio.

Uma ***forest trust*** é de certa forma equivalente, mas ao nível de florestas. Pode ser **one-way** ou **two-way**, mas é sempre ***transitive***. Ao ser estabelecida aplica-se a toda a floresta, ou seja todos os domínios da floresta. De acordo com os sentidos em que é estabelecida permite aos utilizadores de qualquer domínio de uma floresta aceder aos recursos de qualquer domínio da outra floresta.

Active Directory – Realm Trust and Shortcut Trust

Uma ***realm trust*** é usada para estabelecer uma relação de confiança entre um domínio *Windows Active Directory* e um *realm Kerberos versão 5* de sistemas não Windows, por exemplo Linux.

A *realm trust* pode ser **one-way** ou **two-way** e pode ser ***transitive*** ou **non*transitive***.

Uma ***shortcut trust*** é sempre ***transitive***, mas pode ser **one-way** ou **two-way**. Esta relação de confiança pode ser definida no seio de uma floresta para melhorar a performance das operações entre árvores diferentes da mesma floresta. A validação da relação de confiança entre dois domínios de árvores distintas é estabelecida recorrendo às relações de confiança através da hierarquia do AD até ao nível da floresta.

Sob o posto de vista de eficiência isso poderia ser penalizador, por exemplo quando um utilizador de um domínio de uma árvore efetua o login num domínio de outra árvore. As ***shortcut trusts*** permitem evitar ir ao nível da floresta para este tipo de operações.

Administração de Sistemas (ASIST)

Aula Teórico Prática 08

Integração Windows – Linux
Software Samba.
Registos de Eventos (*Event Logs*)

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Integração Linux - Windows

A interoperacionalidade entre servidores Windows e servidores Linux através da rede é na atualidade facilitada pelo facto de ambos servidores integrarem cada vez mais protocolos standard como o DNS, LDAP, iSCSI, NFS, SMB/CIFS e Kerberos 5, entre outros.

Apesar deste suporte ao nível de protocolos de aplicação básicos, a forma como eles são usados difere substancialmente entre servidores Windows e Servidores Linux, essencialmente devido as diferenças de conceção entre os sistemas operativos subjacentes.

Desde logo, a maioria dos conceitos de domínios do *Active Directory* dos servidores Windows está ausente nos servidores Linux.

O **Samba** (*https://www.samba.org/*) é um conjunto de componentes de software para Linux que inclui serviços de rede e aplicações clientes. Permite a um servidor Linux simular muitas das funcionalidades dos servidores Windows e constitui a ferramenta base de qualquer integração mais séria entre estes dois tipos de sistemas em ambiente de rede.

Contas de utilizadores e grupos

Enquanto em Linux os utilizadores e grupos são identificados, respetivamente, através de um **UID** e **GID** únicos, em Windows são identificados através de um **SID** (*Security Identifier*) único.

Tendo em mente esta diferença a compatibilização vai exigir o estabelecimento de um mecanismo de mapeamento (***UID <-> SID***) e (***GID <-> SID***).

Além deste aspeto as incompatibilidades são diversas começando pelos nomes de utilizadores e grupos que são muito mais restritivos em Linux do que em Windows.

Uma forma de resolver as incompatibilidades de nomes é criar mapeamentos estáticos entre nomes Windows que não são válidos em Linux e nomes válidos em Linux. Por exemplo, fazendo corresponder ao grupo Windows **Domain Users** um grupo criado em Linux **dom_users**.

Note-se também que a maioria das restrições relativas a nomes de utilizadores e grupos em Linux não existem de facto, são restrições de retro compatibilidade, genericamente os nomes suportados em Windows são diretamente suportados em Linux.

Quer em Linux quer em Windows cada utilizador pertence obrigatoriamente a um grupo primário (compatibilidade POSIX) e opcionalmente a outros grupos. No entanto, enquanto em Linux apenas utilizadores podem ser membros de grupos, no *Active Directory*, grupos podem ser membros de grupos.

Partilha de ficheiros

Sob o ponto de vista de SAN (iSCSI), nunca existem problemas de integração porque os discos lógicos são formatados e utilizados pelo sistema operativo que os recebe como se fossem discos locais.

Ao nível de partilha de ficheiros em rede (NAS), os servidores Windows são orientados para a utilização de SMB/CIFS enquanto os servidores Linux são orientados para a utilização de NFS.

Embora os servidores Windows possam operar como clientes e servidores NFS, a melhor solução de integração consiste em adicionar o suporte de SMB/CIFS aos servidores Linux através do Samba.

O Samba permite partilhar pastas e impressoras através de SMB/CIFS de forma totalmente compatível com as máquinas Windows, as incompatibilidades surgem sob o ponto de vista das caraterísticas dos sistemas de ficheiros subjacentes à partilha.

No Linux as ACL base (Posix) apenas suportam três entidades: **owner**, **group** e **others**. Com apenas três permissões possíveis para cada um deles: **read**, **write** e **execute**.

Desde que o UID (*owner*) associado ao objeto corresponda a um utilizador Windows (*SID*) e o GID (*group*) associado ao objeto corresponda um grupo Windows (*SID*), estas duas entidades são diretamente colocadas na ACL Windows. A entidade Linux **others** é normalmente associada ao SID do grupo Windows **everyone**.

Linux - *Extended Permissions* (ACL)

Nos sistemas Linux atuais as ACL base podem ser expandidas através da utilização dos comandos **getfacl/setfacl**. Continuam a existir apenas as três permissões das ACL base (*rwx*), mas as entidades deixam de estar limitadas a (*owner/group/others*). Além destas três entidades das ACL base podem ser acrescentados outros utilizadores e grupos.

Usando o comando **setfacl** é possível definir para uma pasta a "*Default ACL*", essa ACL será herdada pelos objetos colocados dentro dessa pasta. Também é possível definir uma máscara de permissões efetivas máximas que podem ser atribuídas pelas várias entradas da ACL.

Tanto as partilhas de rede via Samba (SMB/CIFS) como via NFS suportam as *extended permissions* do Linux, relativamente às ACL Windows permitem uma melhor integração através do Samba.

Numa partilha de rede realizada através do Samba as ACL definidas são armazenadas sob a forma de *extended permissions* no Linux, assim deixam de estar limitadas a apenas três entidades.

Note-se que se o Samba estiver a exercer a função de DC de domínio AD as ACL das partilhas serão ACLs nativas Windows e não são armazenadas sob a forma de *extended permissions* no Linux. Por essa razão aconselha-se a que o DC Samba *Active Directory* não exerça a função de **File Server** (servidor de partilhas) e essa função seja transferida para um *domain member*.

Samba – Atributos do MS-DOS

Algumas aplicações ainda utilizam os atributos dos ficheiros herdados do MS-DOS: ***System***, ***Hidden***, ***Archive*** e ***Read-only***. Esses atributos não existem nos sistemas de ficheiros Linux subjacentes às partilhas de rede realizadas através do Samba.

O problema pode ser colmatado através do seu armazenamento na ACL de *extended permissions* do Linux ou através do seu mapeamento para permissões execute de owner, group e others.

Samba – *Partilha de impressoras*

A partilha de impressoras (Print Server) é integralmente suportada pelo Samba, incluindo a instalação automática de drivers nos sistemas operativos Windows clientes (Workstations).

O Samba pode disponibilizar impressoras configuradas localmente através do **CUPS** (*Common UNIX Printing System*) ou do **LPRng**, ou ainda usar diretamente impressoras de rede.

Também é possível criar uma impressora virtual PDF que cria automaticamente ficheiros em formato PDF numa pasta partilhada na rede.

Integração Linux/Windows - Samba

Desde a versão 4 o Samba pode assumir a função *Domain Controller Active Directory*, isso significa que pode fazer praticamente tudo o que o Windows Server faz como DC no *Active Directory*. É mesmo possível ter no mesmo domínio *Domain Controllers* Windows Server 2016 a trabalhar conjuntamente com *Domain Controllers* Samba.

Este será o papel mais avançado que o Samba suporta, no entanto existem muitos cenários de integração nos quais pode ser usado.

Samba – *Standalone Server*

O Samba configurado como ***Standalone Server*** permite disponibilizar partilhas de rede. Não estando integrado em nenhum domínio, será necessária a autenticação do utilizador perante o servidor para aceder às partilhas.

As contas de utilizador válidas serão todas as contas Linux válidas (**NSS**) desde que tenham sido acrescentadas à base de dados de contas Windows do Samba.

O Samba tem de manter uma base de dados sombra de contas de utilizadores e grupos Linux para armazenar atributos que, sendo necessários nas contas Windows, não existem nas contas Linux.

Neste tipo de configuração (***Standalone Server***) o Samba tem de simular contas de utilizadores e grupos Windows a partir das contas Linux.

O único atributo das contas Linux que é diretamente transferido para as contas Windows é o **login-name,** tudo o resto tem de ser definido pelo Samba e armazenado num repositório sombra. Os SID dos utilizadores e grupos são definidos com base respetivamente no UID e GID e são armazenados juntamente com todos os outros atributos das contas Windows.

O Windows realiza a autenticação dos utilizadores através de ***hashes*** designados **NT-password** e **LM-password** (este último apenas é mantido para compatibilidade com clientes antigos). Estes ***hashes*** ficam também guardados nas contas de utilizadores Windows.

A password Linux e a password Windows são totalmente independentes, no entanto o Samba inclui um mecanismo através do qual sempre que o utilizador altera a password através do Windows a respetiva password Linux é alterada da mesma forma.

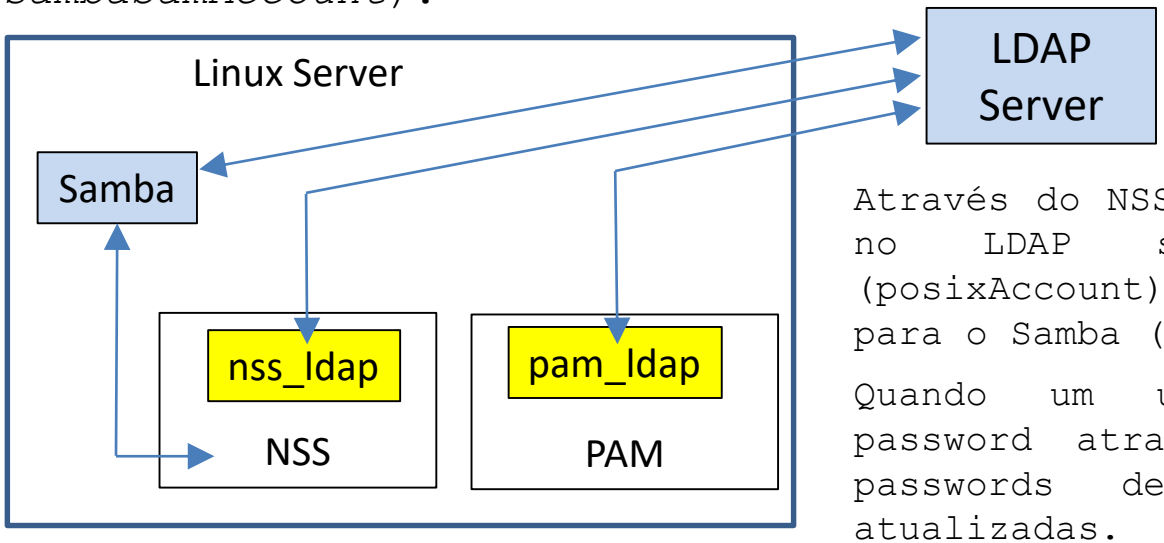
Uma conta de utilizador ou grupo só é válida sob o ponto de vista do Samba/Windows se for um utilizador/grupo válido no Linux e existir no repositório sombra do Samba (contas Windows).

Samba - Utilização do LDAP como repositório

Em qualquer configuração do Samba em que seja necessário gerir contas locais de utilizadores Windows, o *default* é o Samba armazenar esses dados em bases de dados internas (**tdbsam** – *Trivial Data Base Security Account Manager*). Neste caso temos armazenados em locais diferentes a conta Linux (*NSS*) e a conta sombra Windows (*tdbsam*), usar LDAP permite resolver este problema com algumas vantagens.

No LDAP podemos definir objetos que pertencem a várias classes (desde que apenas uma seja estrutural e as restantes auxiliares).

Por exemplo, no caso das contas de utilizadores podem ser usadas as classes LDAP **posixAccount** e **sambaSamAccount.** Isto permite armazenar no mesmo objeto os atributos do Linux (*posixAccount*) e do Windows (*sambaSamAccount*).



Através do NSS, os utilizadores definidos no LDAP são válidos no Linux (posixAccount) e são igualmente válidos para o Samba (sambaSamAccount).

Quando um utilizador altera a sua password através do Windows/Samba as passwords de ambas as contas são atualizadas.

Samba – *Active Directory Domain Member*

O Samba pode ser um ***Domain Member Server*** de um domínio *Active Directory* ou de um domínio Windows NT, independentemente do domínio operar com *Domain Controllers* Windows Server ou Samba.

Isto significa que todos os objetos do domínio, incluindo utilizadores e grupos passarão a ser válidos no sistema Linux e consequentemente no Samba. Deste modo, os utilizadores autenticados num domínio da floresta terão acesso direto às partilhas de rede disponibilizadas pelo Samba sem necessidade de autenticação adicional.

Adaptar as contas de utilizadores e grupos, tal como estão definidas no domínio Windows, ao que o Linux espera que sejam contas de utilizadores e grupos exige algum esforço. Existe um serviço local dedicado a essa tarefa designado **Winbind**.

O Winbind é o responsável por dar um aspeto Linux às contas Windows e tem vários desafios pela frente. O mais importante é realizar o mapeamento entre os SID dos utilizadores e grupos Windows para UID e GID em Linux.

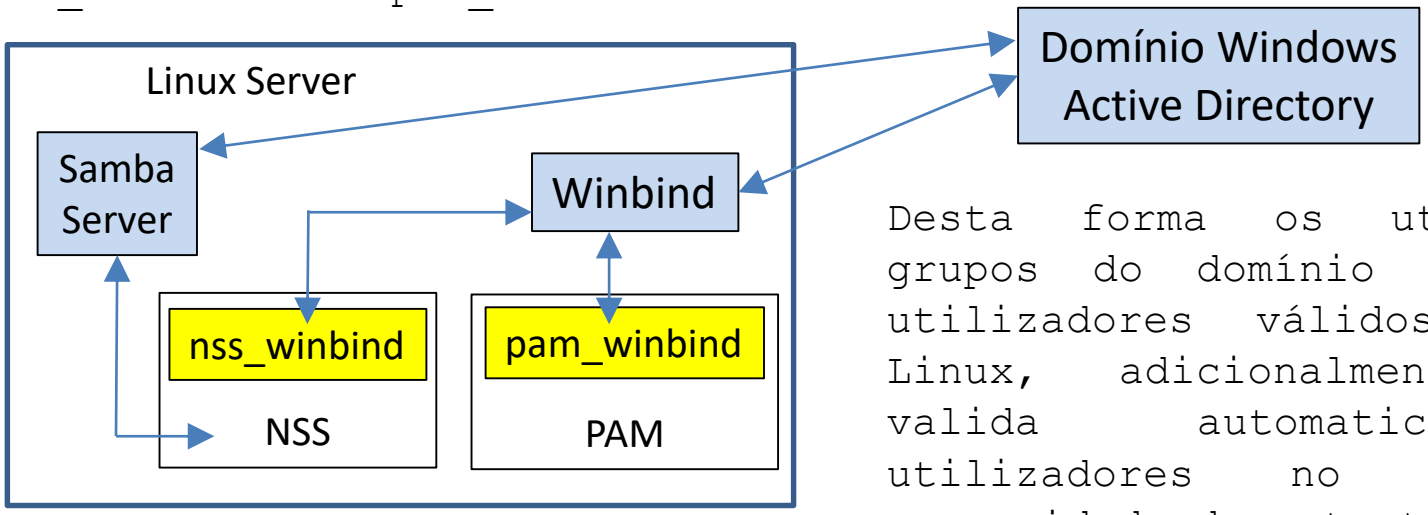
Neste tipo de configuração é atribuída uma gama de UIDs e GIDs para o Winbind gerir, cada vez que se depara com um SID desconhecido associa-lhe de forma <u>persistente</u> um novo UID ou GID das gamas estabelecidas, conforme se trate de um utilizador ou grupo.

Samba *domain member* – *Winbind*

O Winbind mantém os mapeamentos estabelecidos numa base de dados interna, assim quando atribui um UID ou GID a um determinado SID, o utilizador ou grupo correspondente vai ter sempre o mesmo UID ou GID.

No entanto, se existirem vários *domain members* configurados desta forma nada garante que em todos eles o Winbind atribua o mesmo UID ou GID ao mesmo SID. Para resolver este problema é possível configurar o Winbind para guardar os mapeamentos num repositório centralizado LDAP, desta forma garante-se que um utilizador ou grupo Windows terá o mesmo UID ou GID em todos os servidores Linux que são membros do domínio.

A integração dos utilizadores e grupos do Winbind no sistema Linux é feita à custa de módulos NSS e PAM apropriados, nomeadamente o nss_winbind e o pam_winbind:

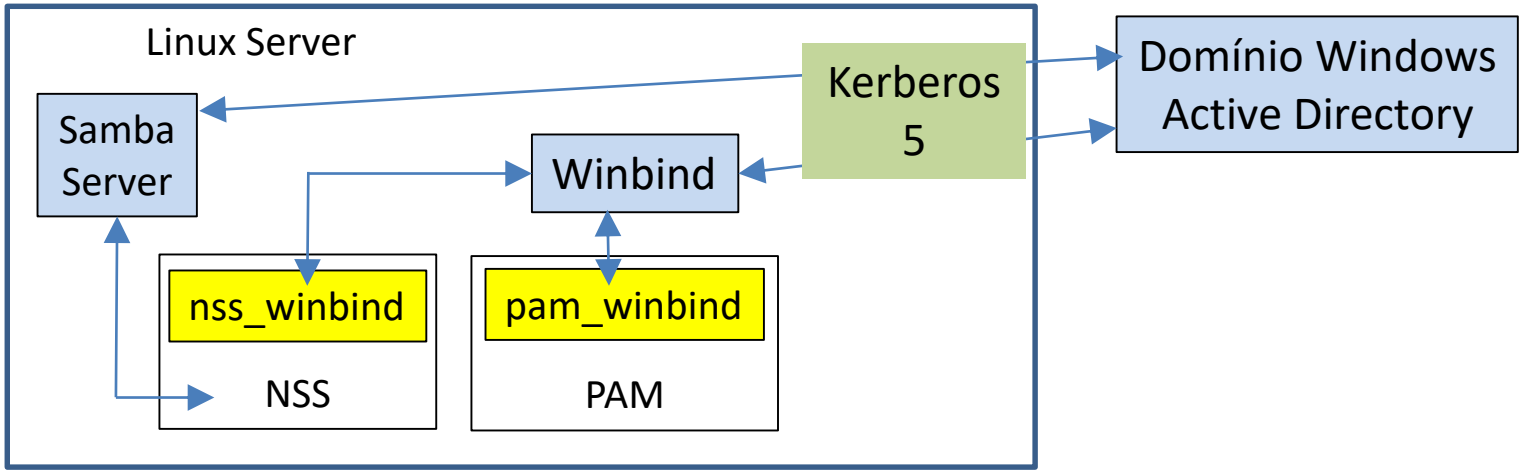


Desta forma os utilizadores e grupos do domínio passam a ser utilizadores válidos no sistema Linux, adicionalmente o Samba valida automaticamente os utilizadores no domínio sem necessidade de autenticação manual.

Samba *domain member* – *Kerberos 5*

O funcionamento do *Active Directory* exige que todas as transações entre membros do domínio sejam protegidas e autenticadas através do *Kerberos 5*, os *Domain Controllers* possuem um servidor *Kerberos 5*, incluindo um **KDC** (*Key Distribution Center*).

Assim, **os *domain members* de um domínio *Active Directory*** necessitam de ter um cliente *Kerberos 5* configurado para usar esse KDC.



Para evitar ataques de ***replay*** as mensagens usadas pelo *Kerberos* transportam ***timestamps***, normalmente uma mensagem torna-se invalida se a diferença horária é superior a 300 segundos.

É portanto fundamental assegurar a sincronização horária entre as máquinas de um domínio *Active Directory*. No caso do Linux, a melhor forma de assegurar isso é utilizar o serviço **NTP** (*Network Time Protocol*).

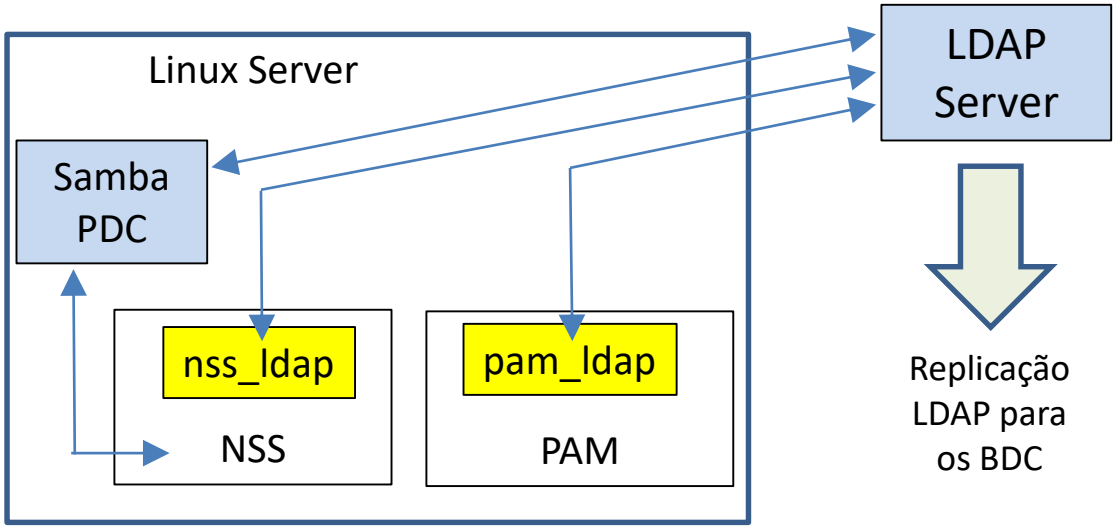
Samba *domain controller NT*

O Samba pode funcionar como **PDC** (*Primary Domain Controller*) de um domínio Windows em modo de compatibilidade NT (pré *Active Directory*).

A configuração é semelhante ao modo de funcionamento ***Standalone Server***, ou seja são usadas as contas locais do Linux apoiadas numa base de dados com informação complementar de atributos Windows.

A solução mais interessante é a utilização de uma base de dados LDAP para armazenar a informação relativamente a contas de forma integrada.

São suportadas todas as funcionalidades presentes nos domínios NT nativos, nomeadamente: Single Sign-On no domínio, home directory, roaming profiles, logon script e políticas de domínio.



Sendo usado o LDAP, também é simples definir BDCs (*Backup Domain Controllers*), para esse feito devem ser usados os mecanismos de replicação do LDAP para garantir a replicação da base de dados LDAP do PDC para as que são usadas pelos vários BDC.

Samba *Active Directory Domain Controller*

A Samba também pode operar como *Domain Controller Active Directory*, atualmente isso exige algumas condições na configuração do servidor Linux.

Devido às especificidades do *Active Directory*, terá de ser usado um servidor LDAP interno do Samba e terá de ser também usado um servidor DNS integrado no Samba. Pode ser usado o servidor *Kerberos* MIT do sistema operativo ou em alternativa um servidor *Kerberos* interno do Samba.

O facto de se usarem servidores integrados no Samba significa que toda a gestão de utilizadores, grupos e DNS terá de ser realizada através do Samba/Windows.

O servidor LDAP integrado é puramente *Active Directory*, não contém atributos de contas Linux. Por exemplo para que os utilizadores e grupos do domínio sejam válidos no Linux é necessário usar o Winbind e configurar o Linux como *domain member* como se tratasse de um domínio Windows *Active Directory* nativo. Não é possível usar diretamente o servidor LDAP integrado.

Num domínio *Active Directory*, devem existir pelo menos dois DC, neste modo de funcionamento o Samba assegura a replicação automática da base de dados *Active Directory* entre todos os *Domain Controllers* do domínio.

Registos de atividades (*Event Logs*)

Os sistemas operativos e aplicações produzem mensagens representativas dos eventos que ocorrem, essas mensagem são guardadas persistentemente em ficheiros de log.

A análise dos ficheiros de log pelos administradores deve ser o primeiro passo quando algo não funciona como pretendido. A inspeção dos ficheiros de log permite ainda detetar situações anómalas que apesar de não afetarem o funcionamento aparente dos sistemas podem ser importantes, por exemplo sob o ponto de vista da segurança.

Se forem usadas ferramentas de monitorização automática dos sistemas, um dos aspetos que deve ser monitorizado são os ficheiros de log.

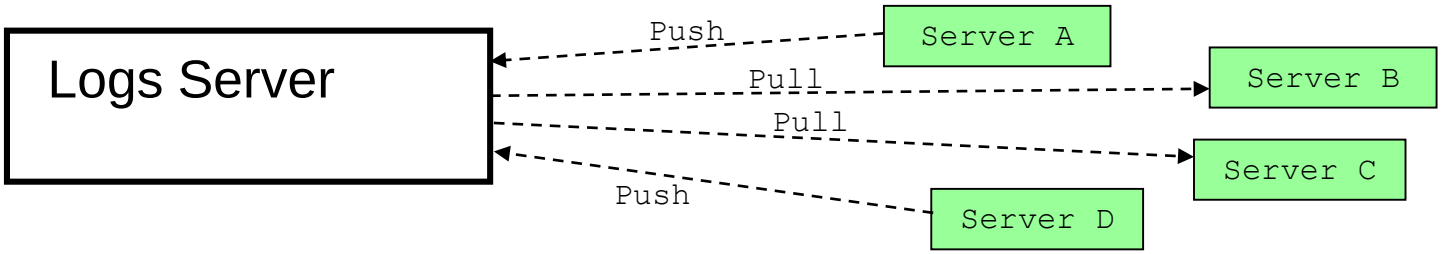
Cada mensagem de log gerada por uma aplicação, serviço ou pelo sistema operativo contém normalmente um **nível de gravidade**, uma identificação da **origem** (quem a produziu – computador, aplicação, serviço, subsistema, etc.) e um **texto descritivo** do evento.

A quantidade de informação de log gerada por cada origem pode ser geralmente configurada nessa origem, tipicamente um serviço ou aplicação. Se um serviço ou aplicação não está a funcionar corretamente podemos muitas vezes configurá-la para aumentar a quantidade de informação de log que ela gera (nível de log), uma vez detetado e resolvido o problema, devemos reduzir novamente o nível de log para o normal para não afetar a sua performance.

Registos de eventos centralizado

Num ambiente de rede distribuído em que existem vários servidores e outros equipamentos, podemos centralizar o registo de eventos. Para esse efeito será escolhido um servidor (servidor de log) que vai obter através da rede os eventos produzidos em todos os outros equipamentos. Desta forma o administrador dispõe de um local único onde pode consultar os eventos relativos a todos os equipamentos da infraestrutura.

A centralização de log pode ser implementada de duas formas **PUSH** (*Source Computer Initiated*) ou **PULL** (*Collector Initiated*).



PULL: O servidor de logs toma a iniciativa de estabelecer uma sessão para transferência de logs, para isso será necessário definir no servidor de logs um a um todos os equipamentos dos quais se pretende obter eventos.

PUSH: O equipamento de origem do evento toma a iniciativa de contactar o servidor de logs. Em cada um dos equipamentos é necessário definir o mesmo servidor de logs correto.

Windows Event Logs

Nos sistemas Windows, os eventos são guardados em ficheiros distintos de acordo com a sua origem. A aplicação ***Event Viewer*** permite consultar e gerir esses registos de eventos.

Os logs estão organizados por temas de acordo com a sua origem (sistema, aplicações, serviços, etc.) existindo um log distinto para cada, nestes logs são registados todos os eventos independentemente da sua gravidade. Também podem ser adicionados novos ficheiros de log.

No ***Event Viewer*** podemos criar ***Custom Views*** de logs, numa ***Custom View*** podemos agregar conjuntos de eventos de vários logs, definir os níveis de gravidade que pretendemos incluir na ***Custom View*** e outros critérios de filtragem.

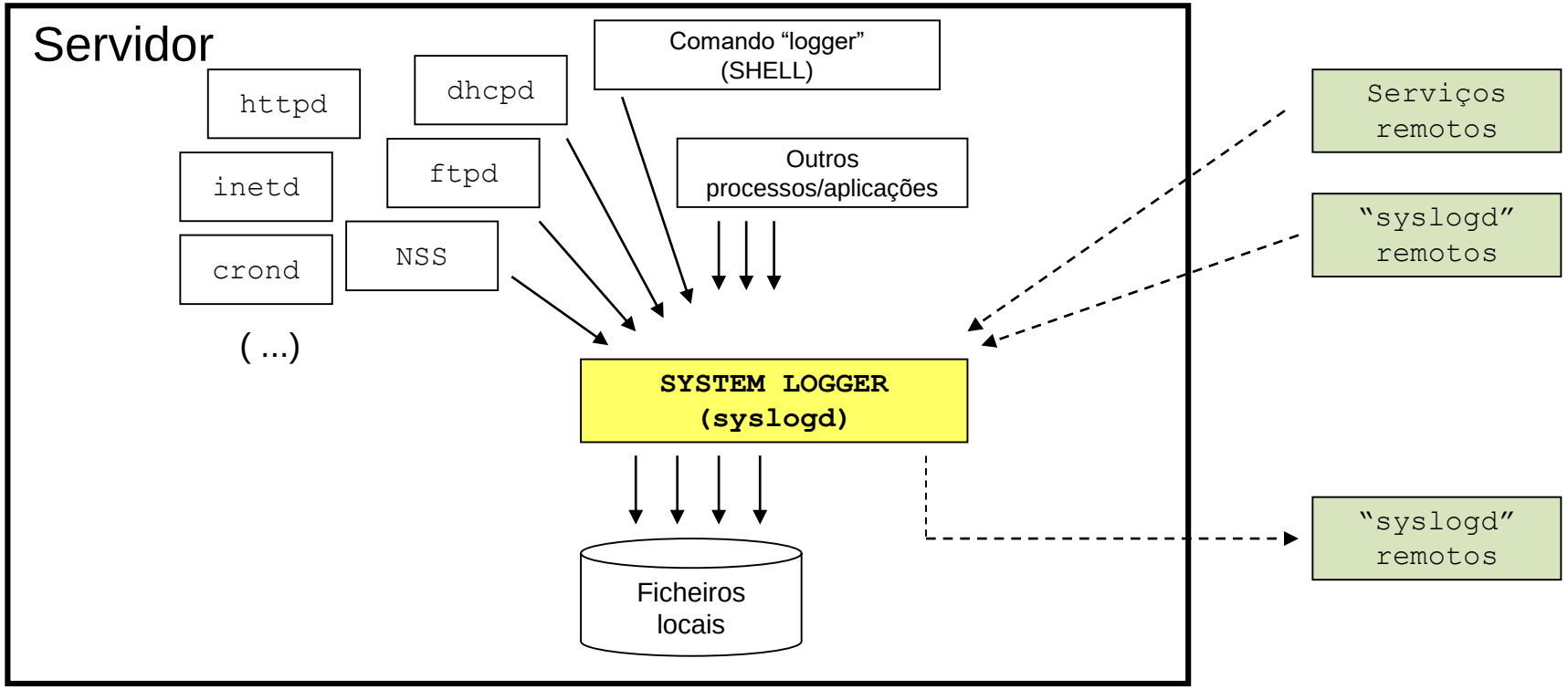
Podemos centralizar os logs através da criação de uma ***Subscription*** na qual definimos as máquinas de onde queremos obter o eventos, o log onde depositar esses eventos, e critérios de filtragem.

Numa ***Subscription*** podemos obter os eventos por iniciativa do servidor que os pretende recolher (*Collector Initiated*) ou por iniciativa das máquinas de origem (*Source Computer Initiated*). Neste último caso a máquina de origem terá de ser configurada para o efeito. A transferência de logs entre máquinas usa o protocolo HTTP/HTTPS dirigido para números de porto especiais.

Linux - *System Logger* (syslog)

O Linux utiliza um sistema *standard* de log conhecido por **Syslog**, através de uma API apropriada, as aplicações e serviços enviam ao Syslog os seus eventos.

De acordo com os atributos definidos na origem o Syslog classifica os eventos e, através das configurações definidas pelo administrador, realiza uma ou várias ações sobre eles. Normalmente uma das ações mais comuns é acrescentar (append) o evento a um ficheiro de texto que é usado como repositório, normalmente na pasta **/var/log/**.



System Logger – Classificação das mensagens

Embora as versões mais avançadas como o **RSYSLOG** (*the rocket-fast system for log processing*) permitam usar uma grande variedade de critérios para classificar os eventos, incluindo através do conteúdo do respetivo texto, a forma mais tradicional de classificação é através dos atributos ***facility*** e ***severity*** definidos na origem.

O atributo ***facility*** serve para identificar o tipo de origem, pode ter um dos valores seguintes: *auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, security* (o mesmo que *auth*), *syslog, user, uucp* e *local0* a *local7*.

O atributo ***severity*** define o nível de gravidade do evento, por ordem crescente de gravidade, os valores possíveis são: debug, info, notice, warning, warn (igual ao anterior), err, error (igual ao anterior), crit, alert, emerg, panic (igual ao anterior).

Estes dois atributos são frequentemente apresentados na forma agrupada: ***facility.severity***

A missão fundamental do Syslog é analisar estes dois atributos e em função do seu valor definir o que fazer com o evento. Algumas ações standard são: acrescentar (append) a um ficheiro local; enviar para um servidor Syslog remoto; enviar diretamente para o terminais de utilizadores ativos; executar um programa externo e passar-lhe a mensagem através do ***stdin***.

Linux System Logger – Configuração

Nas versões tradicionais do Syslog o ficheiro de configuração é o **/etc/syslog.conf** e a classificação dos eventos baseia-se exclusivamente nos atributos ***facility.severity***, o ficheiro contém uma sequência de padrões destes atributos e a ação a desencadear para os eventos que lhe correspondam. Para cada evento, todas as ações definidas para padrões que lhe correspondam serão executadas, não apenas a primeira correspondência.

Exemplo:

```
kern.*                          /var/adm/kernel
mail.*;mail.!=info              -/var/adm/mail
*.alert                         root,admin
mail.=info                      @server2.ipp.pt
mail,news.=info                 *
kern.info;kern.!err             |/root/kern-info
```

Os valores de ***severity*** no padrão fazem correspondência com eventos com nível de gravidade superior ou igual. Para cada padrão é especificada uma ação, se a ação começa por **/** significa que se trata de um ficheiro de logs onde o evento deve ser registado, pode ser precedido do sinal **-** para indicar é necessário executar o **sync** (caso contrário pode não ser imediatamente escrito no ficheiro).

```
*.alert                          root,admin
mail.=info                       @server2.ipp.pt
mail,news.=info                  *
kern.info;kern.!err              |/root/kern-info
```

A ação também pode ser um ou uma lista de utilizadores, nesse caso se os utilizadores estiverem ativos em sessões de terminal recebem uma mensagem com o evento no seu terminal. O símbolo ***** permite o envio da referida mensagem a todos os utilizadores ativos em sessões de terminal. A barra vertical (**|** - *pipe*) identifica um comando externo a ser executado que recebe no ***stdin*** o evento.

O símbolo **@** permite identificar através de um endereço IP ou nome DNS uma máquina para a qual pretendemos enviar o registo de evento. É necessário que na máquina remota esteja em funcionamento um Syslog configurado para aceitar eventos da rede no número de porto UDP 514.

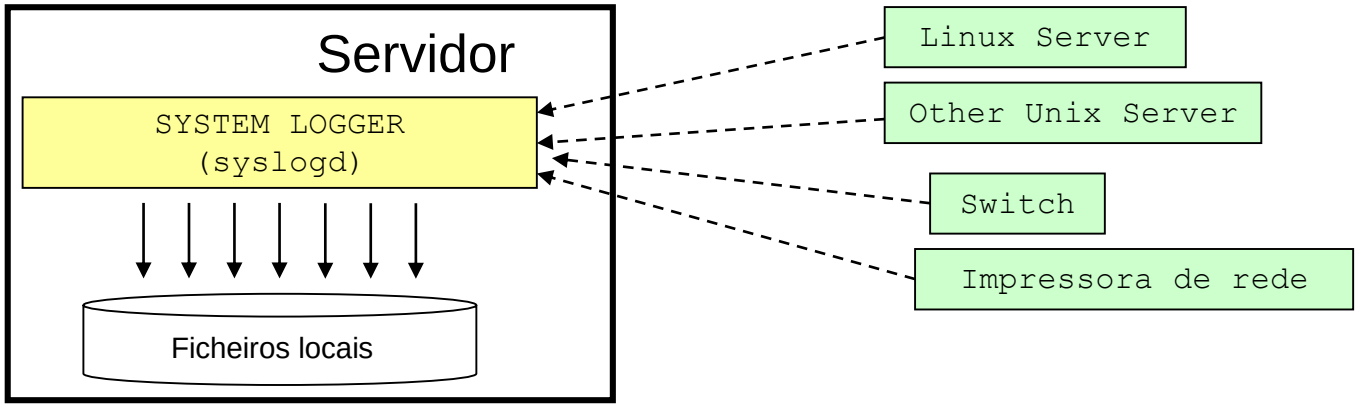
Algumas versões do Syslog também suportam a utilização de TCP e outros números de porto, nesse caso a ação deve ser identificada por @@ e o número de porto pode ser definido, por omissão será o porto 514.

Exemplo:

```
*.*             @@server1.example.net:10514
*.info          @192.168.5.7:23000
*.!=info        @@192.168.5.1
```

Linux System Logger – Centralização

Uma vantagem do Syslog é que se trata de um standard (RFC5424), que entre outros aspetos define como requisitos mínimos o transporte dos eventos através de UDP. Uma grande diversidade de equipamentos de rede simples como impressoras, switches, routers e outros, suportam o protocolo Syslog.



É mesmo possível através de DHCP fornecer aos equipamentos da rede a indicação de qual é o endereço do servidor Syslog que devem usar (atributo **log-servers** do DHCP).

O protocolo de transferência de eventos através da rede do Syslog não dispõe de mecanismos de autenticação e segurança, assim deve ser usado através de redes seguras, de preferência usando TCP e permitindo o acesso apenas para os endereços de origem correspondentes aos equipamentos dos quais se pretende receber os eventos (filtragem).

Embora o protocolo Syslog não seja suportado pela Microsoft nos sistemas Windows, existem implementações disponíveis não Microsoft, incluindo uma versão do **RSYSLOG.**

Linux System Logger – Ficheiros de log

Em última instância, a maioria dos eventos processados pelo Syslog acabam em ficheiros de log. Tratam-se de ficheiros de texto e como tal podem ser consultados usando vários comandos disponíveis no Linux para esse efeito como por exemplo: **cat**, **less**, **tail**, **more**, **grep** e **cut**.

Logrotate

A dimensão de cada ficheiro de log vai crescer sucessivamente, para organizar estes ficheiros pode ser usado o comando ***logrotate***.

Tipicamente o ***logrotate*** é executado diariamente pelo CRON para consolidar os ficheiros de log segundo as configurações definida no ficheiro **/etc/logrotate.conf**. A consolidação consiste numa rotação do ficheiro de log, ou seja transferir o conteúdo atual para um ficheiro de arquivo e esvaziar o ficheiro de log, os ficheiros de arquivo possuem o mesmo nome do log com um sufixo numérico de ordem ou de data.

Através desta consolidação podemos definir para cada ficheiro de log: a periodicidade da rotação, se os ficheiros de arquivo devem ser comprimidos e o número máximo de rotações a partir do qual se começa a eliminar as cópias de arquivo mais antigas.

Em lugar de uma rotação periódica também podemos definir que a rotação será realizada quando a dimensão do log ultrapassa um dado limite. Pode ainda ser configurado um endereço de email para o qual as copias de arquivo que ultrapassam o número máximo de rotações serão enviadas antes de serem eliminadas.

Linux – registo de entradas e saídas

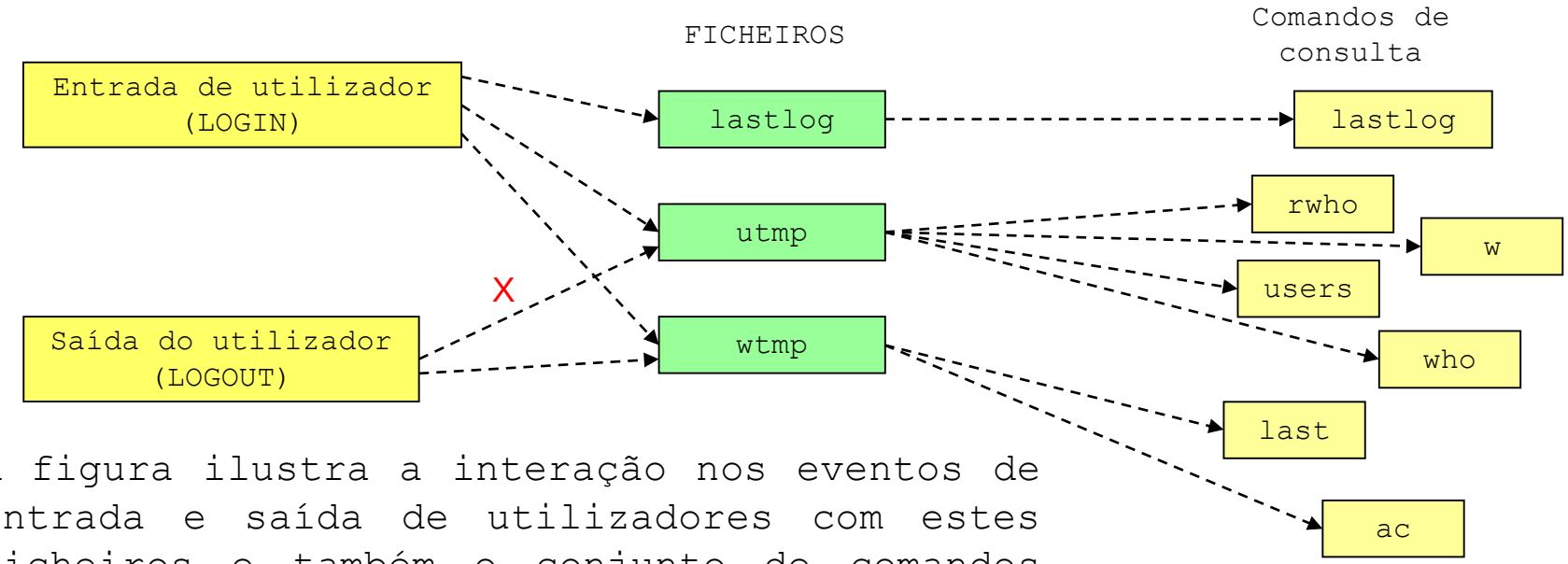
Embora os eventos de entrada (login/logon) e saída (logout/logoff) de utilizadores no sistema sejam registados sob a forma de eventos através do Syslog (incluindo tentativas de acesso falhadas), no Unix existem registos adicionais para este efeito. Os programas que lidam com estes eventos têm a responsabilidade de os registar através de uma API distinta do Syslog. Estes registos são baseados em três ficheiros que não se encontram em formato de texto legível:

- **utmp** (normalmente **/var/run/utmp**) – contém, para cada terminal, o nome do terminal, o nome do utilizador atual, a origem do utilizador (máquina remota/endereço) e a data/hora a que efetuou o login. Quando o utilizador saí, o registo é eliminado.

- **lastlog** (normalmente **/var/log/lastlog**) – contém, para cada UID, a hora e local (terminal/máquina remota/endereço) da última entrada no sistema (login).

- **wtmp** (normalmente **/var/log/wtmp**) – contém o registo histórico de todas as entradas e saídas no sistema.

Linux – registo de entradas e saídas



A figura ilustra a interação nos eventos de entrada e saída de utilizadores com estes ficheiros e também o conjunto de comandos usados para os consultar. Exemplo de resultado do comando **last**:

```
[root@server ~]# last -5
andre    pts/0        beavis.dei.isep. Wed Nov 12 16:35   still logged in
dei      pts/0        beavis.dei.isep. Wed Nov 12 11:46 - 14:12  (02:25)
i090763  pts/4        srv3.dei.isep.ip Wed Nov 12 09:36 - 09:54  (00:17)
i080751  pts/3        srv3.dei.isep.ip Wed Nov 12 08:52 - 10:02  (01:10)
i040478  pts/2        srv2.dei.isep.ip Wed Nov 12 08:51 - 10:03  (01:11)

wtmp begins Sat Nov  1 12:59:40 2004
```

Administração de Sistemas (ASIST)

Aula Teórico Prática 09

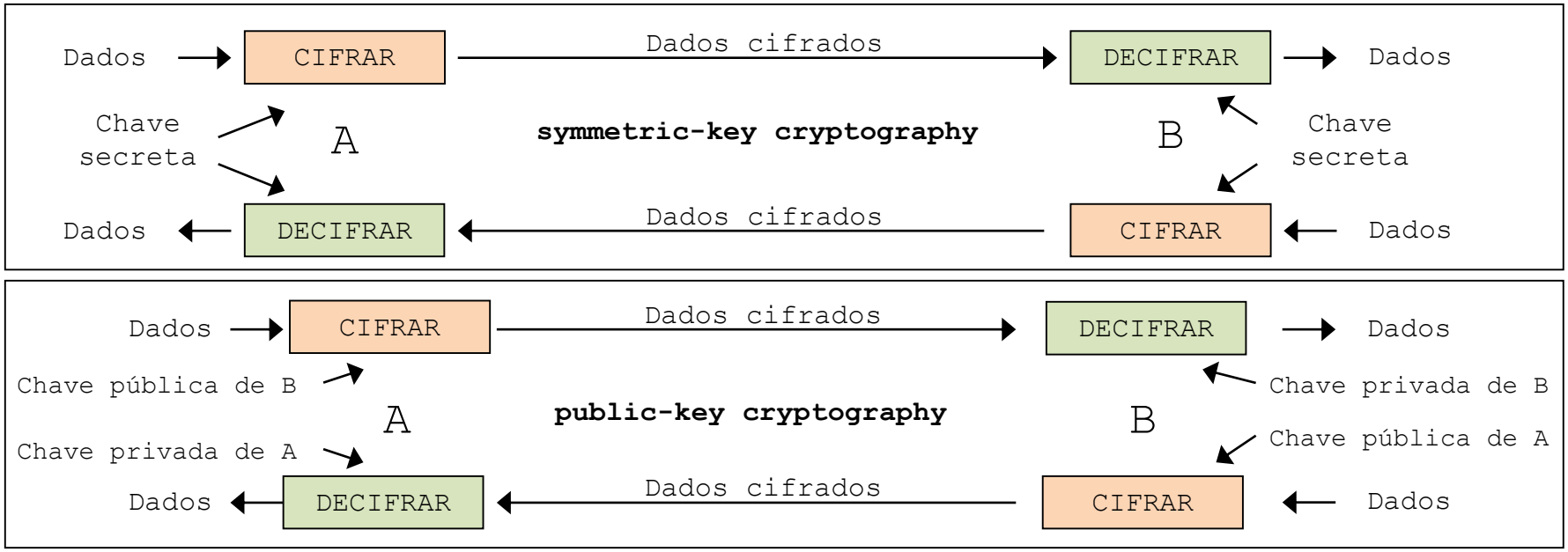
Segurança em rede – Autenticação e privacidade.
Certificados de chave pública.
Cópias de segurança e arquivo.

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Privacidade

O ambiente de rede, em especial em redes WAN, caracteriza-se pela grande dificuldade em controlar o acesso aos dados que nela circulam. A única alternativa para garantir a privacidade é recorrer a algoritmos de cifragem, o acesso aos dados não é impedido, mas os dados serão ilegíveis para entidades ilícitas.

A abordagem tradicional de chave simétrica (***symmetric-key cryptography***) exige que os dois intervenientes possuam uma mesma chave secreta (pré-partilhada). A criptografia de chaves assimétricas (***asymmetric-key***), usa chaves diferentes para cifrar e decifrar e com isso torna a distribuição de chaves muito mais simples. Este é o princípio da chave pública (***public-key cryptography***).



Autenticação - Symmetric-key cryptography

A criptografia torna-se totalmente irrelevante se não for aplicada juntamente com procedimentos de autenticação. Os procedimentos de autenticação em rede destina-se a garantir que as trocas de informação através da rede estão a ser realizadas com a contraparte pretendida e não com um intruso (ataques ***man-in-the-middle***). Sem esta validação, o facto da informação ser cifrada deixa de ser garantia de privacidade.

Symmetric-key cryptography

Se efetivamente temos como pressuposto que apenas as duas entidades lícitas conhecem a chave secreta, então a autenticação está desde logo garantida. Note-se que o problema fica neste caso relegado para um processo anterior de distribuição segura da chave secreta.

A distribuição de chaves secretas pode ser realizada de várias formas:

- **Manual:** o administrador define manualmente a mesma chave nos dois nós de rede.
- **Gerada nos dois nós:** ambos os nós usam um mesmo algoritmo para produzir uma chave a partir de algo que apenas ambos conhecem, por exemplo a password de um utilizador ou o respetivo *hash*. Normalmente no processo de geração da chave nos dois nós é também incluído um elemento aleatório produzido por um deles (*challenge* ou *salt*).

- **Utilização de um canal seguro:** a chave secreta é gerada por um dos nós e transmitida ao outro através de um canal seguro (autenticação e privacidade) pré-existente, por exemplo um canal de comunicação que utiliza criptografia de chave pública. A motivação para este procedimento é o facto de os algoritmos de ***symmetric-key cryptography*** serem muito mais rápidos do que os algoritmos de ***asymmetric-key cryptography***.
- **Utilização de um centro de distribuição de chaves:** um **KDC** (*Key Distribution Center*) é um serviço de rede que funciona como intermediário para o estabelecimento de uma chave secreta entre dois nós. Para isto ser possível é necessário que ambos os nós possam um canal seguro (autenticação e privacidade) estabelecido com o mesmo KDC. Uma das implementações mais divulgadas deste tipo de sistema é o Kerberos, usado nos sistema Linux e no *Active Directory*.

Public-key cryptography

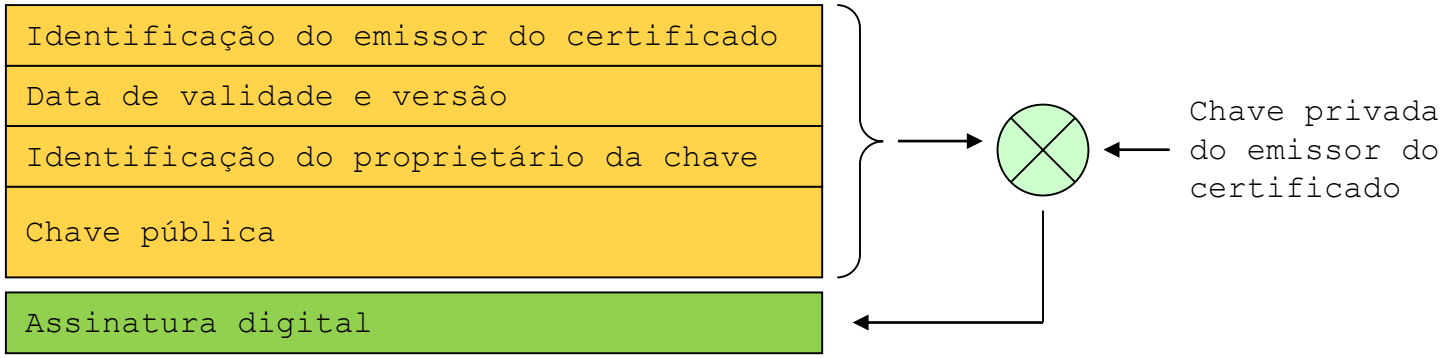
A autenticação não é garantida porque a chave usada para cifrar é pública, logo qualquer um a pode usar.

No entanto como a chave privada correspondente é conhecida apenas pelo próprio, mais ninguém poderá decifrar. Atendendo a que as transações vão ocorrer nos dois sentidos entre os dois nós, se cada nó tiver a certeza absoluta que está a usar a chave pública da contraparte, a autenticação fica garantida.

Certificados de chave pública

Ao usar um algoritmo de chave pública para cifrar uma mensagem temos a garantia que apenas o detentor da chave privada correspondente a poderá decifrar, consequentemente garantir a autenticidade dos nós resume-se a garantir a autenticidade das chaves públicas que usamos.

A autenticidade de uma chave pública pode ser comprovada através de um **certificado de chave pública**. Um *certificado de chave pública* é um bloco de dados que contém: a identificação do proprietário da chave (***Subject***), normalmente será um nome DNS de um nó de rede; a identificação do emissor do certificado (***Issuer***) que será uma entidade certificadora, a chave pública propriamente dita e uma assinatura digital dos elementos anteriores produzida pela entidade certificadora:



O certificado será considerado válido se a entidade certificadora é considerada fidedigna, ou seja se existe confiança na entidade certificadora. Conhecendo a chave pública da entidade certificadora pode facilmente comprovar-se se a assinatura foi produzida por ela.

Exemplo:

Entidade certificadora

Datas de validade

O proprietário da chave pública que se está a certificar, tipicamente o nome DNS do servidor

A chave pública que se está a certificar

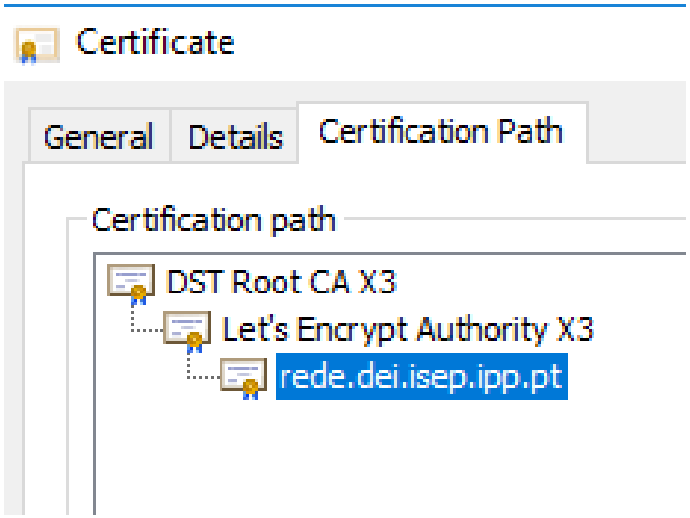
Assinatura digital da entidade certificadora

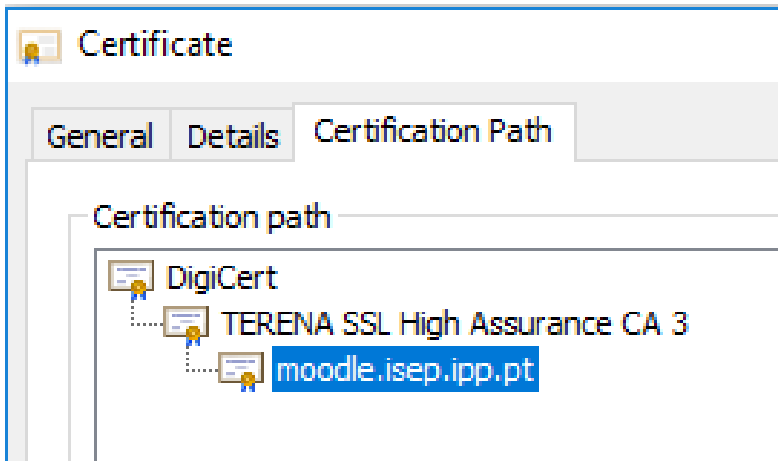
```
[root@server ~]# openssl x509 -in /etc/cert/redecert.pem -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=PT, ST=PORTO, L=PORTO, O=DEI.ISEP, OU=DEI,
      CN=DEICA/emailAddress=dei@dei.isep.ipp.pt
        Validity
            Not Before: Jun 20 11:37:28 2007 GMT
            Not After : Jun 17 11:37:28 2017 GMT
        Subject: C=PT, ST=PORTO, L=PORTO, O=DEI.ISEP, OU=DEI,
      CN=rede.dei.isep.ipp.pt/emailAddress=dei@dei.isep.ipp.pt
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:cc:38:e2:e6:5d:ff:43:6d:ff:43:12:94:03:db:
                    8f:5c:28:61:1a:61:0d:d5:5e:04:dd:20:a9:d8:99:
                    0c:53:e3:c6:23:b0:6c:4d:fe:6b:9e:b8:00:ea:23:
                    1d:55:fa:e7:9a:9b:1b:fa:ef:e0:0d:2c:e0:e1:43:
                    31:2d:b1:37:b5:27:68:01:e0:3d:d8:bf:96:15:bb:
                    25:31:ec:6c:38:5e:2f:17:4c:b5:14:5e:8c:de:1b:
                    14:20:b9:9c:fb:fe:41:5e:ea:68:17:ab:50:a7:9f:
                    6d:93:b3:30:0f:c2:09:2b:7b:43:a4:06:1b:2a:e8:
                    d2:e5:ca:ff:71:e1:69:9c:fb
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        97:e9:b2:9d:f1:ca:16:37:43:21:3a:11:61:03:d7:4b:de:f9:
        06:f9:ee:02:4e:6e:08:25:bc:e3:98:e8:1d:bf:9f:43:5b:cb:
        6c:20:d0:6e:7c:d2:53:f2:29:3b:f9:6e:aa:c3:e0:ab:f8:8f:
        06:83:95:9f:dc:a8:94:bb:a8:50:67:34:de:64:0a:02:29:f0:
        1b:1d:f1:bc:51:09:37:f9:15:23:14:5d:b6:98:85:86:d4:37:
        ff:58:d0:74:24:2f:0d:da:d9:c4:02:89:7f:e6:6c:98:c3:f0:
        8e:42:77:45:3d:a1:ae:a2:8c:20:83:83:c5:2b:cc:58:7c:d7:
        2f:06:32:c6:fa:95:b3:de:5f:48:76:54:08:7a:df:b1:58:a8:
        8d:d9:74:94:57:c1:57:5b:79:32:f1:71:02:77:bb:6f:2a:9f:
        42:94:ef:37:d1:03:da:db:9f:b6:11:8a:4b:d4:e4:34:48:86:
        a0:e3:ce:9e:a3:f4:97:86:37:e1:b9:d4:af:de:a8:e9:ce:2a:
        70:aa:8f:c5:90:d9:71:48:a8:9b:8b:f0:2e:6c:42:7e:c3:bf:
        b6:8e:80:79:2f:ed:81:cb:3a:01:69:ce:c0:e8:9f:fe:ce:0a:
        bf:08:89:4b:68:d0:d5:9b:78:9a:e6:e5:19:92:6c:93:7f:c5:
        a0:89:f9:45
```

Cadeias de certificação

Um certificado de chave pública é considerado fidedigno na medida em que a entidade certificadora (**CA** – *Certification Authority*) que o emitiu é considerada de confiança.

A confiança propaga-se entre as CA, uma CA será considerada de confiança se o seu certificado de chave pública foi emitido por uma outra CA considerada de confiança. Forma-se assim uma **cadeia de certificação**, mas tem de haver uma origem para esta cadeia, na origem está uma **ROOT CA**.





A lista de todas as ROOT CA fidedignas (respetivos certificados de chave pública) é gerida pelo sistema operativo ou aplicação e é atualizada quando se atualiza o sistema operativo ou aplicação.

ROOT CA e certificados auto assinados

Se as ROOT CA são a origem das cadeias de certificação, quem assina o certificado de chave pública de uma ROOT CA? A resposta é, a própria ROOT CA, ou seja é um certificado auto assinado.

Um certificado auto assinado é considerado fidedigno por configuração local do sistema operativo ou aplicação, através da sua colocação na lista de certificados fidedignos. Dessa forma a entidade correspondente e todas a cadeias de certificação com origem nela passam a ser consideradas fidedignas.

Validade dos certificados

Quando um certificado é usado num ambiente de rede (normalmente recebido de um servidor por um cliente) são realizadas várias validações:

- Se está dentro da data de validade.
- Se o campo **CN** do *subject* do certificado corresponde ao **nome DNS do servidor**.
- Se o emissor do certificado está na cadeia de uma CA fidedigna.

Dependendo do tipo de aplicação cliente, em caso de falha numa desta verificações poderá ser perguntado ao utilizador se ainda assim pretende aceitar o certificado e eventualmente mesmo instalar esse certificado com sendo fidedigno (por sua conta e risco).

Revogação de certificados de chave pública

Embora cada certificado contenha uma data de validade, pode em determinadas circunstâncias ser necessário impedir a sua utilização antes de expirado.

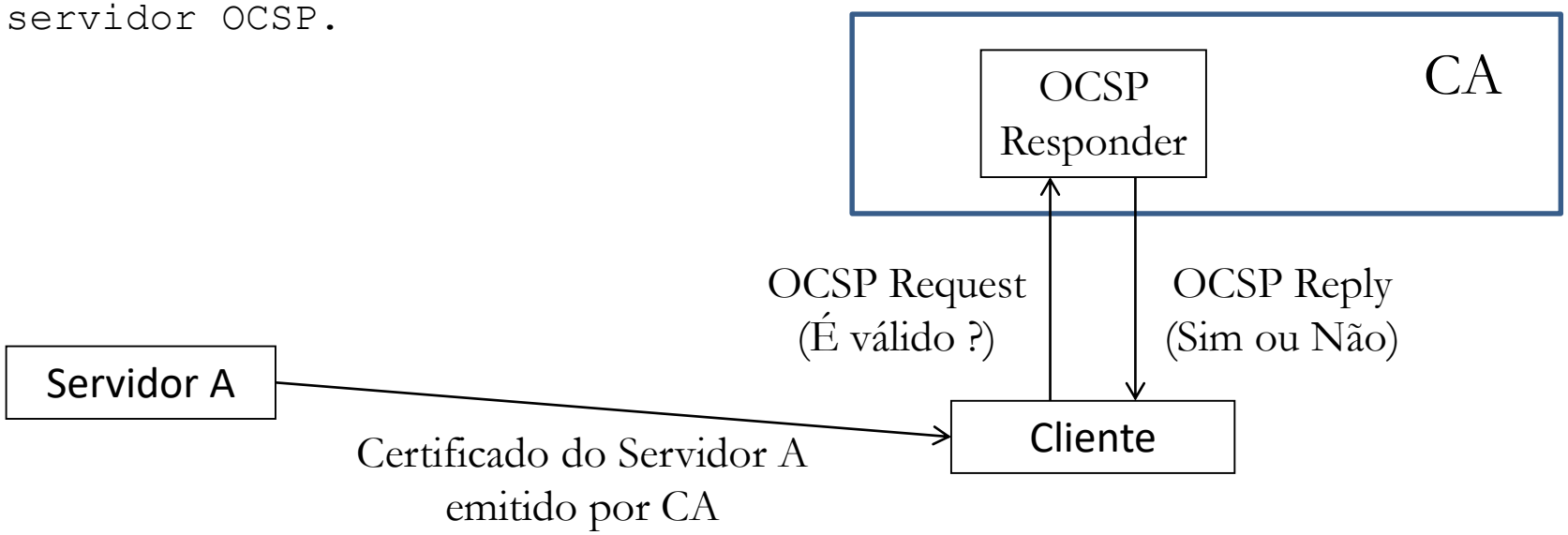
A revogação de certificados é necessária quando se toma conhecimento de que a chave privada correspondente foi comprometida e está na posse de entidades ilícitas. Se não fosse revogado, os utilizadores poderiam continuar a usá-lo e desse forma estar a comunicar com um servidor ilícito criado pelos atacantes.

A verificação da revogação de certificados exige que o utilizador do mesmo realize uma validação adicional através de um contacto direto com a entidade certificadora que o emitiu.

Para que se possa verificar se um certificado foi revogado, o emissor do certificado tem de incluir no mesmo um atributo adicional que identifica através de um nome DNS o serviço de rede disponível para esse efeito.

OCSP (*Online Certificate Status Protocol*)

Embora existam outros mecanismos a forma mais divulgada de se verificar se um certificado foi revogado é através do **OCSP.** Para esse efeito, a CA deverá disponibilizar um servidor OCSP, normalmente designado ***OCSP responder***, adicionalmente deve definir em conformidade um atributo nos certificados que emite onde coloca o nome DNS do seu servidor OCSP.



O grau de exigência das várias aplicações clientes que recebem certificados de servidores varia, algumas podem não aceitar certificados de uma CA se ele não disponibilizar um serviço de verificação de revogação.

Linux – comando openssl

O comando **openssl** fornece uma interface de linha de comando para acesso à biblioteca openssl que suporta SSL v2 e v3 e TLS v1. Entre outras funcionalidades o comando openssl permite gerir certificados de chave pública.

openssl req … – a finalidade principal é gerar um par chave pública/privada e emitir um pedido de certificação da chave pública. O pedido de certificação deve depois ser processado por uma entidade certificadora que emitirá o respetivo certificado. Este comando também pode ser usado para produzir um certificado auto assinado, por exemplo:

```
openssl req -nodes -x509 -keyout pri.key -out pub.crt -days 3650 -newkey rsa:1024
```

Neste exemplo será gerado o ficheiro **pri.key** contendo uma chave privada **RSA de 1024 bits** e o ficheiro **pub.crt** contendo o respetivo certificado de chave pública auto assinado.

openssl x509 … – processamento de certificados, por exemplo é usado pela entidade certificadora para produzir o certificado a partir de um pedido de certificado.

O comando openssl usa o ficheiro de configuração **openssl.cnf**. Algumas operações, nomeadamente de gestão de certificados como autoridade certificadora, exigem o uso deste ficheiro de configuração.

Windows Server – Certificate Authority

O Windows Server pode exercer a função de entidade certificadora (CA) no contexto do *Active Directory* através da ativação do role ***Active Directory Certificate Services*** podendo também instalar-se o ***Online Responder***.

Para operar no contexto do *Active Directory*, a CA deve ser do tipo ***Enterprise CA***. Num servidor que não esteja integrado num domínio AD a CA deve ser do tipo ***Standalone CA***.

A CA criada pode ser inserida numa cadeia de certificação (como ***Subordinate CA***), usando um certificado apropriado emitido por outra CA, ou pode ser uma ***ROOT CA***, neste caso será criado um certificado auto assinado.

Através da aplicação ***Certification Authority*** é então possível gerir a CA, receber pedidos de certificados, emitir os mesmos e revogar certificados.

Também podem ser instalados serviços Web (***Certification Authority Web Enrollment)*** que permite aos utilizadores solicitar certificados à CA através do URL **https://SERVER-DNS-NAME/certsrv**.

Cópias de segurança e arquivo

Existem duas razões principais pelas quais os administradores dos servidores podem criar cópias dos sistemas de ficheiros dos seus servidores.

Cópias de segurança – vulgarmente designadas ***backups***, destinam-se a garantir que caso haja uma falha grave no servidor que leve à perda irremediável dos dados armazenados nos seus sistemas de ficheiros, exista pelo menos uma cópia a partir da qual eles possam ser repostos. A periodicidade com que estas cópias são realizadas vai determinar até que ponto o sistema recuperado vai ser diferente do estado do sistema no momento da falha (**RPO** – *Recovery Point Objective*). Outro aspeto a considerar é que, atendendo aos objetivos, o local de armazenamento da cópia deve ser fisicamente afastado do original, deste modo evita-se que um desastre local grave afete tanto o original como a cópia.

Cópias de arquivo – neste caso o objetivo é libertar espaço nos sistemas de ficheiros do servidor, os dados que já não são necessários são copiados para outro local e removidos do servidor. Ficam ainda assim disponíveis se necessário.

Abordaremos apenas aplicativos existentes por omissão nos sistemas operativos.

Um aspeto que deve estar presente quando se realizam cópias é que elas devem garantir o mesmo nível de privacidade que é assegurado para os dados que se encontram nos sistemas de ficheiros originais.

A privacidade deve ser garantida tanto sob o ponto de vista do local onde as cópias serão armazenadas como do ponto de vista do processo de realização da cópia. Nomeadamente se a cópia é realizada através da rede deve usar protocolos de transferência que incluam mecanismos que deem garantias de autenticação e privacidade.

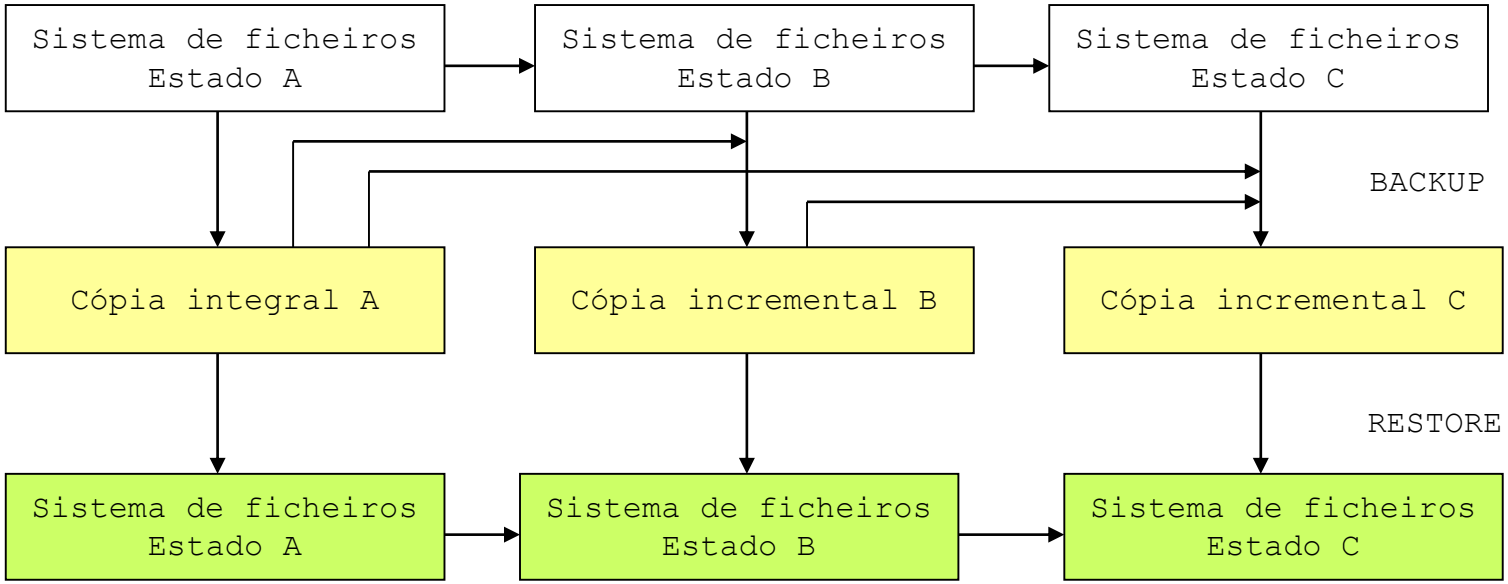
As cópias podem ser realizadas a vários níveis:

- **Disco** – cópia integral do disco, incluindo a tabela de partições, normalmente só pode ser reposta para um disco idêntico completamente vazio.
- **Partição** – cópia integral da partição, a reposição exige uma partição de tamanho idêntico à original.
- **Sistema de ficheiros** – cópia integral do sistema de ficheiros, é de certa forma semelhante à anterior, mas é realizada copiando as estruturas do sistema de ficheiros e não toda a partição.
- **Objetos do sistema de ficheiros** – eventualmente pode contemplar todos os objetos de um sistema de ficheiros, partição ou volume. Ao contrário das anteriores pode ser efetuada sobre um sistema de ficheiros que está em utilização normal porque os objetos são copiados um a um através do sistema operativo.

Cópias incrementais

A cópia de objetos através do sistema operativo acaba por ser a mais utilizada porque é a única que pode ser realizada com o sistema de ficheiros em funcionamento normal, os outros tipos de cópia exigem no mínimo que o sistema de ficheiros esteja em modo *read-only*.

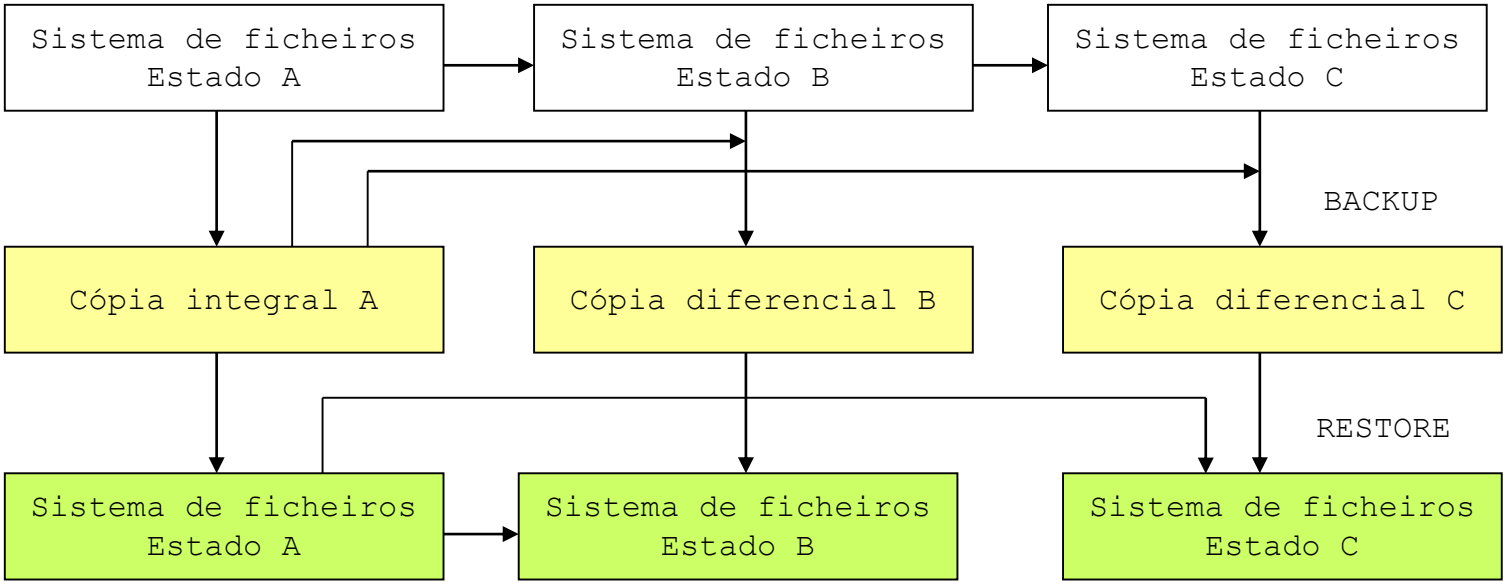
Uma **cópia integral** de todos os objetos existentes num sistema de ficheiros pode ser bastante morosa. Depois de criada uma cópia integral pode recorrer-se a cópias incrementais onde se registam apenas a diferenças relativamente à cópia anterior:



Cópias diferenciais

As cópias incrementais têm dois inconvenientes: é necessário manter tanto a cópia integral como todas as cópias incrementais seguintes; a reposição (*restore*) é morosa porque obriga a aplicar a cópia integral e de seguidas as várias cópias incrementais uma a uma na ordem correta.

As **cópias diferenciais** são ligeiramente diferentes, cada cópia diferencial regista sempre as diferenças do sistema relativamente à cópia integral (e não a cópia anterior).



Apenas é necessário manter a cópia integral e a última cópia diferencial, a reposição torna-se muito mais rápida.

A vantagem das cópias diferenciais é que apenas é necessário manter a cópia integral e a última cópia diferencial, as cópias diferenciais anteriores podem ser eliminadas. A reposição torna-se muito mais rápida porque envolve apenas repor a cópia integral e de seguida a cópia diferencial.

Sejam incrementais ou diferenciais, o número de aplicações sucessivas do processo deve ser sempre limitado, periodicamente deve reiniciar-se o ciclo e realizar uma nova cópia integral.

Uma prática recomendada é realizar todos os fins de semana uma cópia integral e durante a semana realizar cópias incrementais ou diferenciais. A cópia integral, mais morosa, é assim realizada num período de menor atividade.

A realização de cópias de segurança deve ser automatizada através de serviços de execução programada, havendo toda a vantagem em que sejam realizadas em períodos de pouca atividade.

A realização de cópias terá sempre algum impacto no desempenho dos servidores e também será facilitada se não estiverem no momento a ocorrer alterações significativas no sistema de ficheiros.

Windows Server Backup

A aplicação ***Windows Server Backup*** pode ser usada no Windows Server depois de instalada a *feature* com o mesmo nome.

Permite realizar diretamente uma cópia (***backup once***) ou programar a execução de cópias (***backup schedule***). É possível realizar a cópia de drives inteiros (volumes), dados do sistema, todo o servidor (*bare metal recovery*) ou apenas alguns ficheiros e pastas.

O destino da cópia pode ser um drive local ou uma partilha de rede, no local de destino será criada uma pasta como o nome ***WindowsImageBackup*** que contém as cópias realizadas para esse destino e toda a informação associada, incluído o que contém, a máquina e sistema de ficheiros de origem e a data/hora a que foi realizado.

O Windows Server Backup faz a gestão automática de cópias integrais e diferenciais sem controlo direto do administrador. Inicialmente faz uma cópia integral e depois cópias incrementais, periodicamente volta a efetuar uma cópia integral.

Para fazer a reposição deve ser usada a opção ***recover***, e selecionar a origem (drive ou partilha de rede que contém a pasta ***WindowsImageBackup***), as várias cópias disponíveis nesse local serão apresentadas, todos ou parte dos elementos constantes na cópia podem ser restaurados para as localizações originais ou outras localizações.

Linux – comandos de cópia por omissão

No Linux não existe uma aplicação equivalente ao ***Windows Server Backup***, pelo menos não com o mesmo grau de integração com o sistema operativo e simplicidade de utilização.

Existem várias ferramentas de alto nível disponíveis, muitas delas são apenas *front-ends* amigáveis de um conjunto de comandos de cópia que estão disponíveis no sistema operativo.

Com a ajuda de alguns scripts e a execução programada, estes comandos podem ser usados para implementar mecanismos de cópia automática com os mais diversos propósitos.

Comando tar (*Tape Archive*)

O comando **tar** permite criar um ficheiro por vezes designado ***tarball*** que contém partes ou a totalidade de um sistema de ficheiros.

Uma vez criado o tarball (opção **-c**), sempre que necessário, podem ser extraídos (**-x**) todos ou parte dos ficheiros e pastas nele armazenados preservando todas as propriedades dos originais (opção **-p**).

Adicionalmente o tarball pode ser comprimido usando o gzip (opção -z) ou outros algoritmos de compressão. Os nomes dos objetos são guardados no tarball, mas nunca incluem a raiz do sistema de ficheiros, isso permite que os objetos possam ser extraídos para um local diferente do original.

O **tar** suporta cópias incrementais através da opção **–g**, através desta opção pode ser indicado um ficheiro de registo de ***snapshot***, se esse ficheiro não existe será feita uma cópia integral e o ficheiro será criado, exemplo:

```
tar –c –f /backups/backup.00.tar –p –g /backups/backup.snap /etc /var/lib
tar –c –f /backups/backup.01.tar –p –g /backups/backup.snap /etc /var/lib
tar –c –f /backups/backup.02.tar –p –g /backups/backup.snap /etc /var/lib
```

Presumindo que inicialmente o ficheiro **/backups/inc.snap** não existe, então o arquivo **/backups/backup.00.tar** será uma cópia integral com todo o conteúdo das pastas **/etc** e **/var/lib**. Os seguintes serão cópias incrementais. Recorde-se que o tar omite a raiz nos nomes dos objetos, logo serão armazenados com nomes começados por **etc/** e **var/lib/**.

Para recuperação (extração) não é necessário indicar o registo de *snapshot*, mas é necessário usar a opção **–G** para indicar que se trata de uma sequência de cópias incrementais e a recuperação terá de ser feita exatamente na mesma ordem:

```
tar –x –f /backups/backup.00.tar –p –G –C /
tar –x –f /backups/backup.01.tar –p –G –C /
tar –x –f /backups/backup.02.tar –p –G –C /
```

A opção **–C** indica o local para onde se pretende extrair o conteúdo do arquivos, se não for especificada serão extraídos para a pasta de trabalho corrente.

Cópias remotas

Normalmente as cópias serão realizadas para servidores remotos, no caso das cópias de segurança deve ser considerado como um requisito.

A forma mais simples de conseguir isso é simplesmente efetuar a cópia para uma partilha de rede (NAS) com permissão ***read-write***, no Linux será tipicamente uma pasta remota montada por NFS, mas também pode ser por SMB/CIFS.

Uma alternativa é a utilização do comando **rsync**, este comando permite copiar sistemas de ficheiros para servidores remotos através de SSH ou através de um serviço próprio (**rsyncd**) que terá de ser instalado na máquina alvo.

Normalmente a operação será realizada *unattended* pelo que nenhuma password de acesso poderá ser solicitada. Se for usado o serviço *rsyncd* a password pode ser guardada num ficheiro (o username/password do rsyncd é independente dos utilizadores do sistema). Se for usado SSH, é necessário configurar esse serviço para funcionar sem password, a opção mais aconselhada é fazer a autenticação através de um par chave pública/chave privada.

Se o serviço SSH estiver configurado para operar sem passwords também se torna possível enviar o resultado do comando tar através de uma sessão SSH, por exemplo:

```
tar –c –p /etc /var/lib | ssh USER@TARGET-HOST “( cat > /backups/backup.tar )”
```

Cópias parciais

Uma cópia parcial abrange apenas uma parte do sistema. Normalmente o objetivo é salvaguardar dados referentes a um dado subsistema ou serviço, tais como servidores Web, servidores de bases de dados, etc..

A cópia parcial pode ser realizada em Unix com recurso ao comando **tar**, para esse efeito basta localizar todos os ficheiros e pastas associados ao subsistema ou serviço em causa.

O exemplo seguinte cria uma cópia de segurança comprimida para uma instalação particular do servidor Apache:

```
tar –c –z –p -f /backups/apache2.tar.gz /etc/apache2 /var/www  \
/etc/ssl/certs/apache2.pem /etc/ssl/private/apache2.key
```

Neste caso estão a ser contemplados todos os ficheiros de configuração na pasta **/etc/apache2**, as páginas disponibilizadas **/var/www** e ainda as chaves usadas pelo HTTPS.

Muitos serviços dispõe de comandos próprios para realizar cópias de salvaguarda dos seus dados enquanto estão em funcionamento, especialmente no caso das bases de dados.

Exemplos: o serviço **MySQL** dispõe do comando **mysqldump**, o serviço **OpenLDAP** dispõe do comando **slapcat**.

Nestes casos é claramente vantajoso usar estes comandos especialmente concebidos para o efeito e que podem ser usados de forma segura sem necessidade de interromper o serviço.

Linux – sistemas de cópia

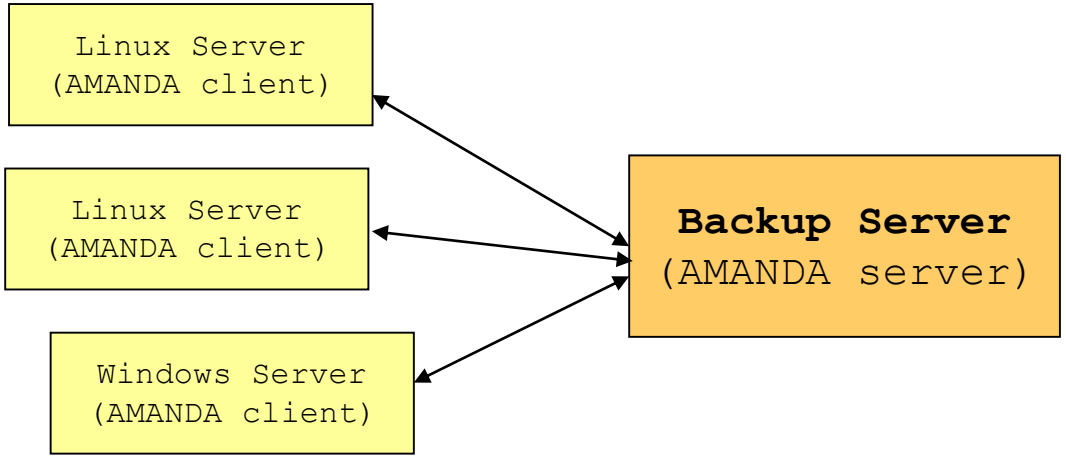
Embora seja possível através de scripts e os comandos anteriormente referidos automatizar as cópias, existe também algum software gratuito de nível mais elevado que deve ser mencionado.

AMANDA (Advanced Maryland Automatic Network Disk Archiver) (http://www.amanda.org/)

O AMANDA é um sistema centralizado de cópias, é constituído por um servidor que corre numa máquina Linux onde as cópias são armazenadas e clientes que realizam as cópias para o servidor.

Os clientes podem ser máquinas Linux, mas existe também um cliente para sistemas Windows. A realização das cópias programadas pode ser desencadeada a partir do servidor AMANDA.

Muitas das funcionalidades do AMANDA são implementadas através de comandos nativos dos sistemas Linux como o tar e ssh.



Administração de Sistemas (ASIST)

Aula Teórico Prática 10

Linux – serviços de rede através do INETD
Linux – controlo de acesso (*libwrap*)
Windows Server – Network Policy and Access Services
Windows Server - Hyper-V
Windows Server - Emergency Management Services
Windows Server – DCDIAG

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Linux – *Standalone Server Daemons*

Para disponibilizar um serviço de rede é necessário manter em execução uma aplicação servidora, permanentemente disponível, para receber pedidos de clientes através do protocolo e número de porto estabelecidos para o efeito.

As aplicações que funcionam em *background* nos sistemas operativos são normalmente conhecida por ***daemons***, muitos deles são aplicações servidoras de rede.

Este tipo de configuração faz todo sentido para serviços de rede com elevado grau de utilização e para os quais é desejável um tempo de resposta o mais curto possível.

É verdade que um serviço pouco solicitado não consome recursos de CPU porque o processo se encontra normalmente ***idle*** a aguardar um contacto de um cliente, mas consome memória.

Para serviços de rede com baixa taxa de utilização e para os quais o tempo de resposta não é crítico, o ***inetd*** oferece uma alternativa que evita a necessidade de um processo em execução para cada serviço de rede.

Linux - *Internet Service Daemon* (inetd)

O serviço de rede ***inetd*** permite com um único processo em execução, substituir vários ***daemons*** durante a fase em que estão ***idle*** a aguardar o contacto de um cliente.

Para conseguir esse objetivo o **inetd** cria vários *sockets* UDP e TCP e associa-os aos números de porto correspondentes aos serviços que disponibiliza.

Assim que é contactado por um cliente num desses *sockets* cria um processo filho (*fork()*) e faz-se substituir (*exec()*) pela aplicação servidora correspondente.

É a aplicação servidora que vai efetivamente dialogar com o cliente e não o **inetd**.

Antes de o processo filho ser substituído pelo processo da aplicação servidora efetiva, o descritor correspondente ao *socket* é duplicado para os descritores 0 (*stdin*), 1 (*stdout*) e 2 (*stderr*).

Isso significa que quando a aplicação servidora lê e escreve nestes descritores, na realidade vai estar a ler do *socket* (*stdin*) e a escrever no *socket* (*stdout* e *stderr*).

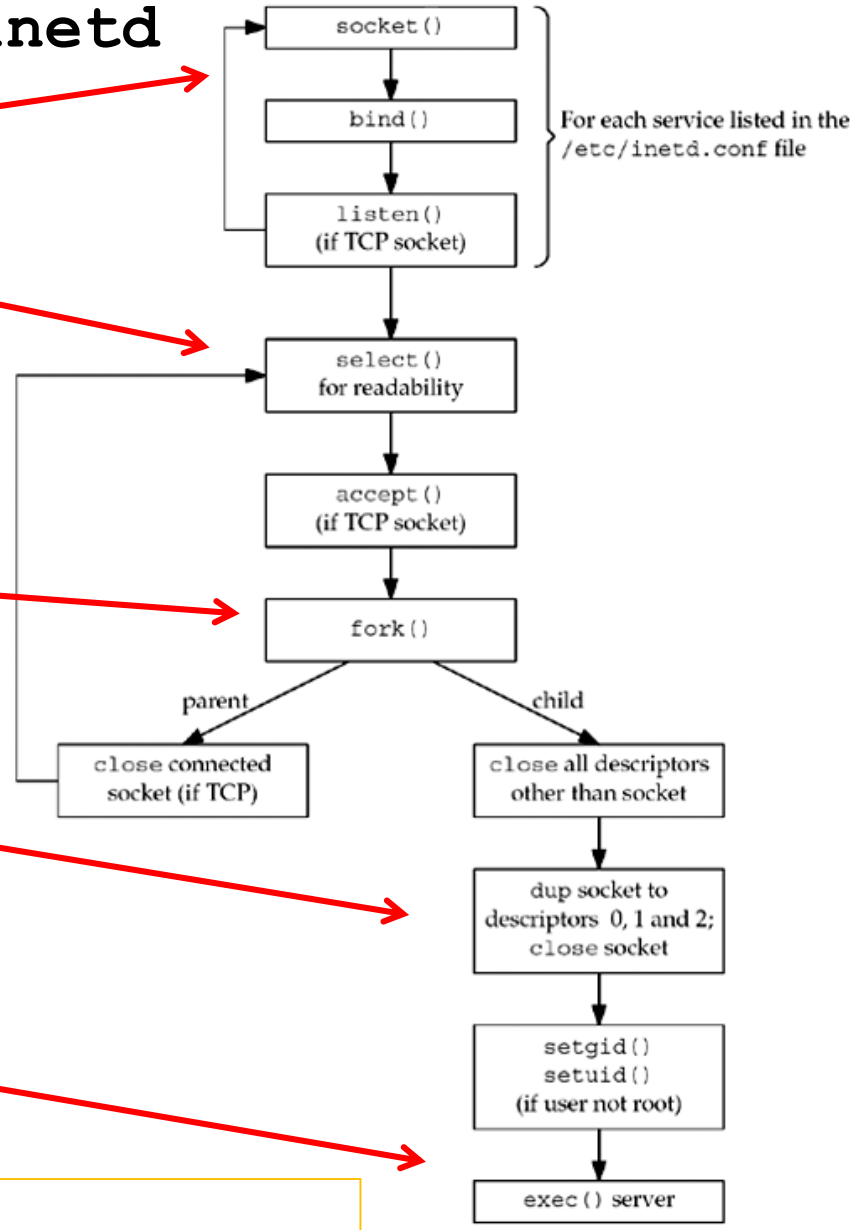
Linux – Funcionamento do inetd

Depois de inicializar os vários *sockets* o INETD recorre à *system-call select* para detetar atividade em qualquer deles.

Quando um cliente contacta um dos *sockets*, no INETD é criado um processo filho

No processo filho os descritores *stdin*, *stdout* e *stderr* são redirecionados para a ligação de rede (*socket*).

O executável é carregado num ambiente em que os *outputs* são enviado pelo *socket* e os *inputs* são lidos do *socket*.



Source:
http://www.masterraghu.com/subjects/np/introduction/unix_network_programming_v1.3/ch13lev1sec5.html

Linux – inetd Configuração (/etc/inetd.conf)

O ficheiro **/etc/inetd.conf** define os serviços prestados, é um ficheiro de texto em que cada linha define um serviço, na forma:

service-name	socket-type	protocol	wait/nowait	username	exec-file	args

service-name – é o nome do serviço definido no ficheiro **/etc/services**. Nesse ficheiro ao nome é associado o protocolo (UDP ou TCP) e o número de porto.

socket-type – *dgram* para o protocolo UDP ou *stream* para o protocolo TCP.

protocol – nome do protocolo de transporte definido no ficheiro **/etc/protocols**, normalmente UDP ou TCP.

wait/nowait – o comportamento ***wait*** obriga o servidor a finalizar o atendimento de um cliente antes de atender o seguinte (comportamento típico para serviços UDP), ***nowait*** permite atender vários clientes em simultâneo (comportamento típico para serviços TCP).

username – nome do utilizador com que o serviço vai ser executado. Serviços que não necessitam de permissões especiais devem ser executados por um utilizador com poucas permissões, com por exemplo *nobody*.

service-name	socket-type	protocol	wait/nowait	username	exec-file	args

exec-file – caminho absoluto do ficheiro executável que vai ser usado para dialogar com cliente segundo o protocolo de aplicação.

args – argumentos a fornecer ao executável, incluindo o arg[0].

Exemplo de ficheiro /etc/inetd.conf:

```
ftp      stream  tcp     nowait   root     /usr/bin/ftpd ftpd -1
daytime  stream  tcp     nowait   root     internal
telnet   stream  tcp     nowait   root     /usr/bin/telnetd telnetd
nrpe     stream  tcp     nowait   nagios   /usr/bin/nrpe  nrpe -c /etc/nrpe.cfg --inetd
tftp     dgram   udp     wait     nobody   /usr/bin/tftpd tftpd -s /tftpboot
```

Neste exemplo podemos observar que o segundo serviço é proporcionado autonomamente pelo INETD (*internal*) sem recorrer a nenhum programa externo.

Os nomes dos serviços (à esquerda) devem estar definidos no sistema (**/etc/services**), cada um corresponde a um número de porto standard para o serviço de rede correspondente.

Linux – Controlo de acesso a serviços de rede

Os sistemas operativos de família Unix incluem uma biblioteca partilhada (*shared library*) que pode ser usada pelas aplicações servidoras para controlo de acesso aos respetivos serviços, trata-se da **libwrap**.

Podemos averiguar se uma aplicação suporta essa biblioteca listando todos os objetos partilhados que o respetivo executável utiliza através do comando **ldd** (*List Dynamic Dependencies*).

Por exemplo para saber se o serviço SSH (**/usr/sbin/sshd**) utiliza a biblioteca **libwrap** podemos usar:

```
root@myserver:~# ldd /usr/sbin/sshd
        linux-vdso.so.1 =>  (0x00007ffccedfa000)
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f22bdfa7000)
        libaudit.so.1 => /lib/x86_64-linux-gnu/libaudit.so.1 (0x00007f22bdd80000)
        libpam.so.0 => /lib/x86_64-linux-gnu/libpam.so.0 (0x00007f22bdb71000)
        libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007f22bd94f000)
        libsystemd.so.0 => /lib/x86_64-linux-gnu/libsystemd.so.0 (0x00007f22bd8ca000)
        (…)
```

Neste caso conclui-se que o serviço SSH utiliza a **libwrap**.

Linux – libwrap - configuração

A **libwrap** usa dois ficheiros para determinar se o acesso a um serviço deve ser permitido ou não:

- **/etc/hosts.allow**
- **/etc/hosts.deny**

Primeiro é consultado o ficheiro **/etc/hosts.allow**, em caso de correspondência, o acesso é imediatamente permitido.

Não havendo uma correspondência em **/etc/hosts.allow**, será consultado o ficheiro **/etc/hosts.deny**, em caso de correspondência, o acesso é imediatamente negado.

Não havendo nenhuma correspondência em nenhum dos ficheiros o acesso é permitido.

Os ficheiros */etc/hosts.allow* e */etc/hosts.deny* são constituídos por sequências de linhas na forma geral:

LISTA-DE-DAEMONS : LISTA-DE-CLIENTES

Para cada tentativa de acesso, havendo uma correspondência numa linha o processamento termina e as linhas seguintes são ignoradas.

Ambas as listas são constituídas por enumerações de valores separadas por espaços ou virgulas. Para que um acesso corresponda a uma linha, terá de ter simultaneamente uma correspondência em ambas as listas da linha.

LISTA-DE-DAEMONS – define uma lista de nomes de processos (argv[0]). Se o serviço de rede é prestado por um processo com um destes nomes, vai corresponder a este critério.

LISTA-DE-CLIENTES – define uma lista de endereços de clientes, pode ser especificada sob a forma de endereços IP, nomes DNS ou padrões. Se o endereço de origem do acesso corresponde a um dos elementos desta lista, vai corresponder a este critério.

Em ambas as listas o valor **ALL** corresponde respetivamente a qualquer *daemon* e qualquer endereço de cliente. Existem várias opções avançadas, por exemplo através do serviço de rede **identd** torna-se possível especificar o nome de utilizador na máquina cliente como critério de acesso a *daemons*.

Linux – suporte de libwrap no inetd

As versões atuais do INETD suportam a libwrap:

```
root@myserver:~# ldd /usr/sbin/inetd
        linux-vdso.so.1 =>  (0x00007ffe831fb000)
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007fcea44cd000)
        libbsd.so.0 => /lib/x86_64-linux-gnu/libbsd.so.0 (0x00007fcea42b8000)
        (…)
```

Se o INETD não suportar a libwrap, ainda assim ela pode ser usada através do **tcpd**, nesse caso o INETD executa o tcpd que por sua vez executa a aplicação servidora.

Exemplos:

```
ftp      stream  tcp     nowait   root     /usr/sbin/tcpd  /usr/bin/ftpd -1
telnet   stream  tcp     nowait   root     /usr/sbin/tcpd  /usr/bin/telnetd
ntalk    dgram   udp     wait     root     /usr/sbin/tcpd  /usr/local/lib/ntalkd
```

Como se pode observar (exemplo ntalk), apesar do seu nome, o tcpd pode ser usado também para serviços de rede UDP.

Windows Server – Network Policy and Access Services

O Windows Server permite a realização de vários tipos de validação relativamente aos acessos através da rede.

O role ***Network Policy Server*** **(NPS)** é uma implementação do serviço RADIUS (*Remote Authentication Dial-In User Service*). O RADIUS é um protocolo de autenticação, autorização e registo (AAA – *Authentication, Authorization, Accounting*), não se destina a interagir diretamente com os utilizadores.

Os clientes do servidor RADIUS são serviços de rede que interagem com os utilizadores e transmitem ao servidor as credências de autenticação juntamente com a identificação do serviço, o servidor RADIUS tem como missões:

1º - validar as credenciais do utilizador (***authentication***).

2º - verificar se o utilizador está autorizado a aceder ao serviço identificado (***authorization***).

3º - caso o acesso seja concedido, informa o serviço de que o acesso deve ser concedido e regista-o (***accounting***).

Windows Server – Network Access Protection (NAP)

No Windows Server o NAP é um conjunto de mecanismos que permite realizar vários tipos de validação relativamente aos acessos através da rede.

A máquina cliente deve possui um agente **NAP** (*NAP-capable computer*) que disponibiliza diretamente ao serviço de rede (*NAP Enforcement Server*) informações sobre o seu estado interno e desta forma o serviço pode determinar se deve ou não autorizar o acesso.

A declaração de estado emitida pelo agente NAP (**SoH** – *Statement of Health*), inclui tipicamente: estado do firewall, configurações de rede, estado das atualizações de segurança e antivírus, entre outras.

A análise do **SoH** pode ser centralizada, nesse caso o serviço de rede envia-o a um NPS (*Network Policy Server*) configurado para suportar NAP, que desta forma funciona como ***NAP health policy server***.

Vários serviços de rede podem operar como *NAP Enforcement Server*, incluído servidores DHCP e servidores de VPN.

Windows Server – *NAP remediation servers*

Quando uma máquina cliente é declarada ***noncompliant*** pelo NAP, o acesso ao *NAP Enforcement Server*, é negado.

Nas configurações NAP do NPS é possível estabelecer um ***remediation server group*** a ser fornecido aos clientes *noncompliant*.

O ***remediation server group*** é uma lista de servidores (endereços IP ou nomes DNS) acessíveis aos clientes *noncompliant* que estes deverão usar para resolver os seus problema relativos ao NAP e tornarem-se ***compliant***, podem ser entre outros:

- Servidores de assinaturas antivírus.
- Servidores de atualizações (WSUS).
- *Domain Controllers* (problemas relacionados com a falta de autenticação)
- *DNS/DHCP Servers* (se o cliente não está a usar o servidor DNS ou DHCP correto).
- *Troubleshooting servers* (URLs) – páginas com instruções para o administrador do cliente tornar o computador *compliant*.

Windows Server - Hyper-V

O Hyper-V é uma plataforma de virtualização que substitui sistemas anteriores equivalentes da Microsoft tais como Microsoft Virtual Server, Microsoft Virtual PC e Windows Virtual PC.

O Hyper-V pode ser ativado nas versões recentes do Windows, nas versões server encontra-se disponível como um role.

A lista de sistemas operativos suportados pelas máquinas virtuais (*guest* OS) inclui inúmeras versões do Windows (server e workstation) e sistemas operativos da família Unix, incluindo o Linux.

Criar uma máquina virtual envolve definir as respetivas características de hardware e o sistema operativo *guest* que se pretende instalar, incluindo: quantidade de memória RAM, discos que vai ter (discos virtuais) e interfaces de rede (associadas a *switches* virtuais definidos no Hyper-V).

Normalmente a instalação do sistema operativo *guest* é realizada a partir de um leitor de CD/DVD virtual associado a um ficheiro com a respetiva imagem (.iso).

Windows Server - Hyper-V – *Switches* virtuais

As interfaces de rede das máquinas virtuais são ligadas a *switches* virtuais definidos no Hyper-V. Um *switch* virtual do Hyper-V pode ser:

- ***External switch*** – está associado a uma interface de rede física do Windows Server onde o Hyper-V está instalado. Permite a comunicação com o exterior através da referida interface física. Isso inclui a comunicação com o próprio servidor, outras máquinas virtuais no mesmo servidor e máquinas virtuais em outros servidores (desde que ligadas a um *external switch*). As máquinas virtuais podem usar diretamente a rede física (*bridge*) ou podem partilhar o acesso à rede definido nas configurações da interface física (NAT). Também é suportada a utilização de VLANs.

- ***Internal switch*** – apenas permite comunicações no interior do servidor, incluindo o próprio servidor e outras máquinas virtuais no mesmo servidor (ligadas ao *switch internal*).

- ***Private switch*** – define uma rede isolada, apenas permite a comunicação entre máquinas virtuais (locais) ligadas ao mesmo *private switch*.

Windows Server - Hyper-V – *Integration Services*

Para garantir uma melhor integração entre o sistema operativo usado na máquina virtual (*guest OS*) e o Hyper-V, existem pacotes de software designados *Integration Services* (**hv_utils**) que podem ser instalados no *guest OS*.

Os *Integration Services* são disponibilizados pelo Hyper-V ao *guest OS* através da opção ***Insert Integration Services Setup Disk*** que irá disponibilizar no leitor de CD/DVD o software adequado para o sistema operativo em causa.

Os *Integration Services* incluem suporte para:

- **Encerramento** (*shutdown*) do *guest OS* através do Hyper-V.
- **Sincronização do relógio** do *guest OS* com o do servidor.
- **Acesso a dados** sobre o estado do *guest OS* pelo Hyper-V.
- ***Heartbeat*** – monitorização do funcionamento do *guest OS* pelo Hyper-V (*watchdog*).
- ***Backup*** (*volume snapshot*) sem necessidade de parar a máquina virtual.
- ***Guest services*** – permitem ao Hyper-V a cópia direta de objetos para os sistemas de ficheiros do *guest OS*.

Windows Server – EMS (*Emergency Management Services)*

O EMS é uma ferramenta de último recurso que pode ser usada para aceder a um Windows Server que por alguma razão se encontra inacessível ao administrador por não estar a funcionar convenientemente.

Para que esta via de acesso esteja disponível tem de ser ativada nas opções de arranque do sistema operativo (ficheiro BOOT.INI do volume de arranque) este ficheiro deve ser editado indiretamente através do comando **bootcfg** (note-se no entanto que este comando assume que o ficheiro é *C:\BOOT.INI* o que pode não ser verdade).

Embora o serviço EMS possa ser configurado para ser acessível pela rede, porque nessas situações a rede pode não estar disponível, geralmente opta-se por utilizar uma porta série, por exemplo:

bootcfg /ems on /port com1 /baud 19200 /id N

Onde **N** representa o número da entrada de arranque no ficheiro C:\BOOT.INI. Neste exemplo, se o servidor ficar inacessível por qualquer outra via pode ser ligado um terminal à porta série COM1 (argumento do comando) para interagir através de uma consola.

Windows Server Domain Controller - dcdiag

O **dcdiag** é uma ferramenta de linha de comando que permite realizar testes ao funcionamento de um *Domain Controller* integrado num domínio *Active Directory*.

Se não forem dados argumentos realiza um conjunto de testes base, mas pode ser usado para realizar testes adicionais orientados para serviços específicos, por exemplo:

dcdiag /test:DNS

A linha de comando anterior permite realizar vários testes à configuração e funcionamento do serviço DNS (fundamental no contexto de um domínio *Active Directory*).

Esta ferramenta constitui um auxiliar precioso para o administrador permitindo identificar rapidamente problemas de configuração e consistência num domínio *Active Directory*.

Administração de Sistemas (ASIST)

Aula Teórico Prática 11

Clustering: failover e load balancing.
Linux – HAProxy
Windows Server - NLB (*Network Load Balancing*)

Baseado em: A. Moreira, 2018/2019, Aulas teórico-práticas de ASIST

Fault tolerance

Fault tolerance (tolerância a falhas) é a propriedade de um sistema que lhe permite continuar a funcionar normalmente mesmo que ocorra uma falha num ou vários dos seus componentes.

Tipicamente a tolerância a falhas implica redundância, ou seja ter vários componentes idênticos que realizam a mesma função. Implica também a existência de mecanismos que detetam os componentes em falha e garantem que o sistema deixa de os utilizar, sem que isso tenha qualquer impacto no desempenho.

Poucos sistemas são totalmente tolerantes a falhas porque para o serem teriam de eliminar todo e qualquer ponto único de falha (**SPoF** – *Single Point of Failure*).

Por exemplo, normalmente consideramos um sistema de discos RAID1 tolerante a falhas, mas isso só é verdade sob o ponto de vista do componente disco, não sob o ponto de vista por exemplo do controlador do disco.

Failover

Uma abordagem para proporcionar tolerância a falhas de um componente de um sistema é o ***failover***.

Na abordagem *failover* existem vários componentes semelhantes (redundância), no entanto apenas um deles está ativo (a ser usado pelo sistema), os restantes estão em *standby*, mas prontos para serem ativados.

Se o componente em funcionamento falha, um dos que se encontra em *standby* é automaticamente ativado por um sistema de controlo para o substituir.

O processo de substituição do componente em falha deve ser automático e rápido para evitar qualquer perturbação no funcionamento global do sistema. O sistema de controlo que efetua a monitorização do estado do componente ativo e desencadeia a sua substituição constitui normalmente um indesejável ponto único de falha.

Se o componente possui estado (*stateful*) é também necessário implementar mecanismos de sincronização que garantam que em qualquer altura os componentes em *standby* tem o mesmo estado do componente que entra em falha (exemplo: **RAID1**).

Load Balancing

Ter componentes em *standby* é sempre um desperdício de recursos. Se for possível o sistema usar em simultâneo os vários componentes redundantes para melhorar a sua performance e simultaneamente garantir a tolerância a falhas essa é a opção ideal. É normalmente conhecida por balanceamento de carga (***load balancing***).

Com *load balancing* todos os componentes redundantes estão em funcionamento permanente, usando um mecanismo de controlo apropriado. O sistema pode por exemplo usar alternadamente cada um dos componentes, se um componente entra em falha deixa de ser usado (*fault tolerance*).

A forma como o mecanismo de controlo define qual dos componentes vai ser usado em cada instante pode ser mais ou menos sofisticada, desde um simples ***round-robin*** até uma monitorização da performance de cada componente e utilização do componente que no momento oferece melhores garantias.

Note-se que a monitorização do estado dos componentes é geralmente obrigatória para detetar falhas. Novamente, o sistema de controlo constitui um ponto único de falha.

Redes - *Fault Tolerance / Load Balancing*

Temos já conhecimentos sobre mecanismos que garantem a tolerância a falhas e o balanceamento de carga nas redes.

Nível 2 (ex.: Ethernet) – caso existam caminhos ou ligações redundantes entre nós intermédios de nível 2 (*switches*), o **STP** (*Spanning Tree Protocol*) implementa *failover*, o **LACP** (*Link Aggregation Control Protocol*) ou outros equivalentes de *Link Aggregation* implementam *load balancing* com *fault tolerance*.

A técnica de *Link Aggregation* também pode ser usada para ligação a nós finais como servidores. No caso dos servidores Windows através de um ***NIC Team***, nos servidores Linux através de um ***bonded NIC***.

Nível 3 (ex.: IPv4/IPv6) – caso existam caminhos redundantes entre nós intermédios de nível 3 (*routers*), o encaminhamento dinâmico apoiado em protocolos de encaminhamento permite implementar *load balancing* com *fault tolerance*. Todos os protocolos de encaminhamento garantem *fault tolerance*, a eficiência do *load balancing* depende essencialmente da métrica utilizada.

Servidores e serviços de rede

Na implementação de serviços de rede redundantes, a estratégia que deve ser adotada é a utilização de servidores redundantes. Isto deriva da necessidade de dentro do possível evitar pontos únicos de falha.

Se implementarmos serviços de rede redundantes usando o mesmo servidor, em caso de falha do servidor ficariam em falha todas as alternativas de prestação do serviço. Seria também necessário lidar com a impossibilidade de ter no mesmo servidor vários serviços redundantes a usar o mesmo número de porto.

No contexto da virtualização de servidores, devemos ainda optar por não só colocar os serviços redundantes em diferentes servidores, mas também de preferência em servidores virtuais residentes em plataformas de virtualização diferentes e independentes.

Novamente a motivação é evitar pontos únicos de falha que afetem todos os serviços alternativos, neste caso se a plataforma de virtualização falhar, falham todos os servidores que lá se encontram e consequentemente todos os serviços alternativos.

Clusters de servidores

Resumindo, na implementação de serviços de rede redundantes devemos evitar que haja a possibilidade de serem todos afetados por uma mesma falha (ponto único de falha). Isso consegue-se implementando-os em máquinas distintas e evitando que haja pontos únicos de falha que possam afetar todas as máquinas.

Um *cluster* de servidores é um conjunto de servidores que trabalham em conjunto com um objetivo comum. Objetivos típicos de um *cluster* de servidores são *load balancing* e *fault tolerance*, ou seja disponibilizar serviços de alta disponibilidade (*High-availability cluster* – HA-*cluster*).

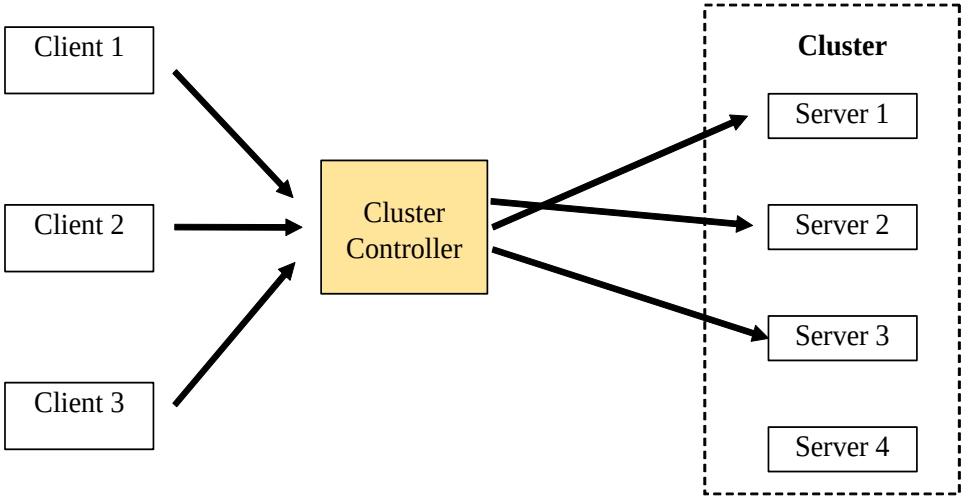
Uma outra caraterística que um *cluster* de servidores deve ter é ser usado como um sistema único, ou seja as solicitações são dirigidas ao *cluster* e não a servidores individuais do *cluster*.

Muitas vezes esta caraterística é implementada através de um controlador do *cluster* que se constitui num ponto único de acesso ao mesmo, consequentemente também constitui um ponto único de falha.

IP único de acesso ao *clusters* de servidores

Embora constitua um ponto único de falha, uma forma de implementar um ponto único de acesso a um *cluster* de servidores é através de um controlador com um endereço IP único.

Os clientes acedem ao *cluster* enviando os pedidos para o endereço IP único do controlador, o controlador redireciona os pedidos para os vários servidores do *cluster* segundo critérios de *load balancing* e *fault tolerance*.

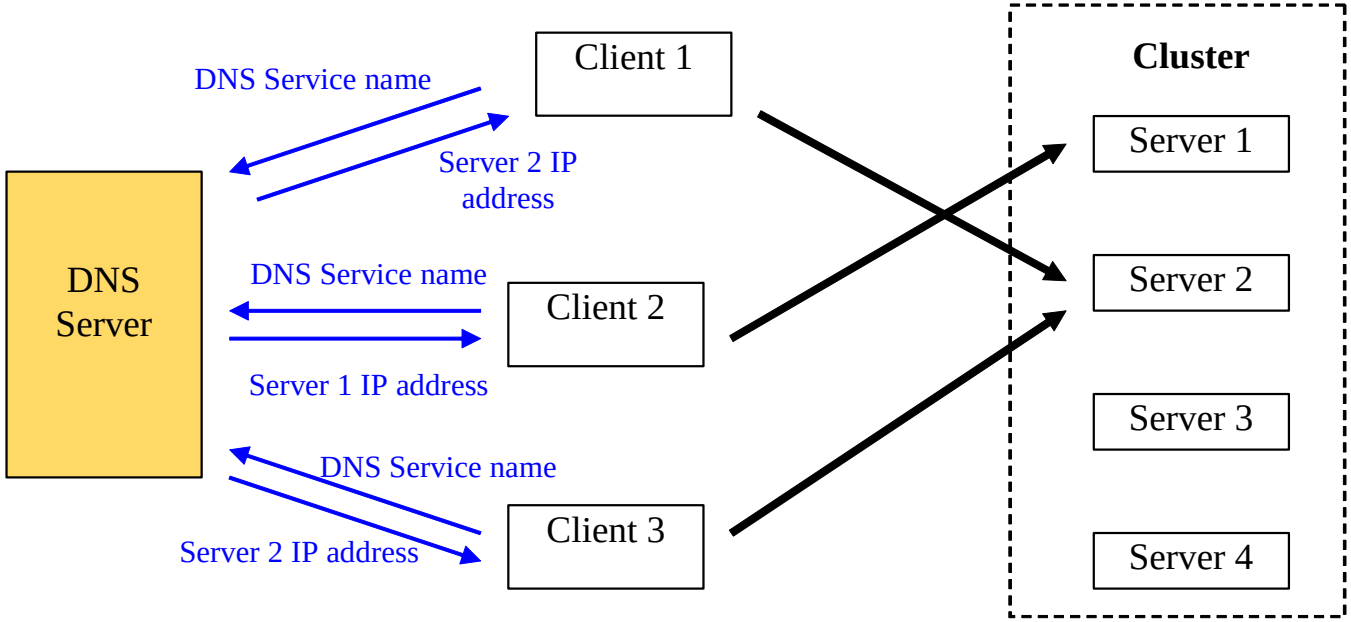


O controlador pode ser implementado através de um router a executar a função **DNAT** (*Destination Network Address Translation*) ou um servidor a funcionar como *proxy*, simplesmente redirecionando os pedidos que recebe.

Para muitos protocolos de aplicação será desejável que, uma vez atribuído um servidor a um cliente, essa atribuição se mantenha (ex.: cookies do HTTP). Caso contrário será necessária uma sincronização mais estrita entre os vários servidores.

Nome único de acesso ao *clusters* de servidores

Uma alternativa a um IP único é a utilização de um nome DNS único, a vantagem é que este sistema assenta sobre o funcionamento do DNS que é por conceção tolerante a falhas.



O controlador do *cluster* é o servidor DNS, é ele que distribui os clientes pelos vários servidores do *cluster* segundo critérios de *load balancing* e *fault tolerance*. Os registos DNS fornecidos pelo servidor DNS aos clientes terão necessariamente de ter um tempo de vida (**TTL**) muito curto pois enquanto não expirarem continuam a ser usados sem nova consulta ao servidor.

Failover e *load balancing* pelos clientes

Uma forma de evitar o ponto único de falha no acesso ao *cluster* é violar uma das características que um *cluster* deve ter e permitir aos clientes o acesso a servidores individuais do *cluster*. Neste caso o controlo passa para o cliente.

O cliente possui uma lista de servidores alternativos e implementa ele próprio o *failover*, ou seja tenta utilizar um deles, se não obtém resposta passa a utilizar outro da lista.

Em complemento ao *failover*, o cliente pode também implementar uma forma de *load balancing* básico fazendo uma rotação da lista (*round-robin*).

Na implementação do *failover* pelo cliente é fundamental definir de forma adequada o tempo máximo (*timeout*) das tentativas de acesso aos servidores.

Este método é muito usado em várias circunstâncias tais como clientes DNS, clientes SMTP, clientes LDAP (**nss_ldap**), etc..

Os servidores DNS podem ser usados para centralizar este processo através de registos **MX** (**SMTP**), **SRV** e registos **A**/**AAAA**.

DNS – Round-robin

No DNS podemos definir registos iguais do tipo NS, MX, SRV, A e AAAA, ou seja que associam o mesmo nome DNS a dados diferentes.

Exemplo:

```
ssh.dei.isep.ipp.pt.   60   A     193.136.62.80
ssh.dei.isep.ipp.pt.   60   A     193.136.62.81
ssh.dei.isep.ipp.pt.   60   A     193.136.62.82
ssh.dei.isep.ipp.pt.   60   A     193.136.62.83
ssh.dei.isep.ipp.pt.   60   A     193.136.62.84
ssh.dei.isep.ipp.pt.   60   A     193.136.62.85
```

Neste exemplo existem 6 registos A todos com o mesmo nome, quando um cliente pede ao DNS a resolução do nome **ssh.dei.isep.ipp.pt** vai obter uma lista de 6 endereços IP.

No entanto, tendo em vista o *load balancing*, o servidor DNS não os fornece na mesma ordem a todos os clientes, vai rodando a lista entre pedidos (*round-robin*).

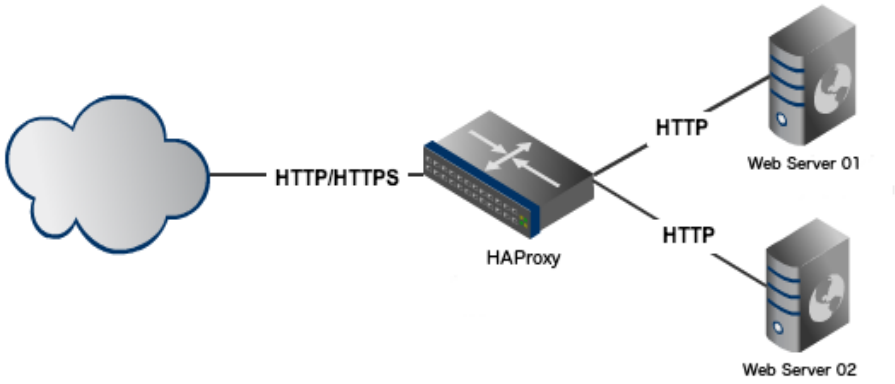
Assumindo que cada cliente tenta em primeiro lugar usar o primeiro endereço da lista fornecida, isso leva a que os clientes se distribuam pelos vários servidores.

Esta funcionalidade deve ser implementada por todos os servidores DNS sempre que existem registos com nomes repetidos.

Linux – HAProxy (The Reliable, High Performance TCP/HTTP Load Balancer)

O **HAProxy** (*http://www.haproxy.org/*) é um controlador de *cluster* para Linux capaz de redirecionar ligações TCP para um *cluster* de servidores incluindo funções de segurança e funcionalidades específicas para o protocolo HTTP.

Entre outras funcionalidades, o HAProxy permite por exemplo receber ligações protegidas por SSL/TLS como HTTPS e transformar as mesmas em ligações não seguras no interior do *cluster*.



Esta possibilidade torna-se útil porque como os clientes acedem ao *cluster* através do nome DNS da máquina onde se encontra o HAProxy, assim evita-se ter de manter certificados de chave pública iguais em todos os servidores do *cluster*.

O HAProxy permite implementar diversas funções de segurança que vão desde a limitação do número de ligações por segundo até à validação das linhas de cabeçalho HTTP antes de serem retransmitidas aos servidores do *cluster*.

Frontends e backends

No HAProxy um **frontend** representa um ponto de acesso para clientes, um **backend** é um conjunto de servidores do *cluster*.

O ficheiro de configuração é constituído por secções contendo parâmetros de configuração, nomeadamente uma secção **global** e diversas secções **defaults**, **frontend**, **backend** e **listen**.

Declaração da secção	
global	Define parâmetros globais relacionados com o funcionamento geral do HAProxy.
defaults [NOME]	Define conjuntos de parâmetros que serão aplicados a todas as secções seguintes, a menos que sejam alterados na secção ou através de outra declaração defaults.
frontend NOME	Define um *frontend*, nomeadamente através do parâmetro **bind** e do parâmetro **use_backend**.
backend NOME	Define um *backend*, nomeadamente através de parâmetros **server**.
listen NOME	Forma de declaração mais expedita de um proxy incluindo na mesma declaração *frontend* e *backend*.

Stickiness

Quando o HAProxy é contactado por um cliente, para o **frontend** correspondente estará associado um **backend**, de entre os vários servidores definidos no **backend** será atribuído um a esse cliente.

Devido ao funcionamento dos protocolos de aplicação, o HAProxy tem de manter essa associação ***cliente<->servidor*** em pedidos subsequentes, isso é designado ***stickiness***.

O mecanismo de *stickiness* é gerido automaticamente através de *stick-tables*, mas há vários parâmetros que permitem o seu controlo.

Parâmetros ***mode tcp*** e ***mode http***

Para cada frontend/backend, são suportados dois modos de operação: **tcp** e **http**. O modo tcp (mode tcp) retransmite a informação diretamente entre o cliente e o servidor sem a analisar. No modo http (mode http) é realizada uma análise e validação das mensagens HTTP antes de serem transmitidas ao servidor.

Parâmetro ***balance ALGORITHM***

O parâmetro **balance** pode ser usado nas secções **backend** e **listen**, serve para definir o modo de balanceamento de carga entre os servidores ativos dos que estão definidos no **backend.**

Existem cerca de nove algoritmos disponíveis, a maioria dos quais é parametrizável. Se não for definido nenhum, será usado o valor **roundrobin**. O algoritmo **roundrobin** é ponderado dinamicamente, isto é, a probabilidade de um servidor ser selecionado depende da sua eficiência anterior.

Outro algoritmo possível é **leastconn**, nesse caso o eleito é o servidor com menor número de ligações ativas no momento. Entre vários servidores com o mesmo número de ligações, é aplicado round-robin.

Note-se que o algoritmo só é usado quando se trata de um cliente desconhecido, ausente da *stick-table* ou quando há necessidade de lhe atribuir um novo servidor porque o anterior deixou de estar disponível.

HAProxy - exemplo de configuração

A figura representa uma secção *listen* do ficheiro de configuração:

```
listen http_proxy
    bind 10.2.0.1:80,10.2.0.1:8080
    stats enable
    mode http
    option httpclose
    option forwardfor
    balance roundrobin
    server S1 192.168.5.10:80 check
    server S2 192.168.5.11:80 check backup
    server S2 192.168.5.12:80 check
```

Disponibiliza informação estatística via HTTP no URL /haproxy?stats

Insere nos cabeçalhos HTTP Connection: close

Insere nos cabeçalhos HTTP X-Forwarded-For: IP-CLIENTE

A opção **check** na declaração server força a verificação periódica do seu estado através de ligações TCP.

Um servidor declarado como **backup** só é usado se nenhum dos outros estiver disponível.

Windows Server – NLB e *clustering*

A *feature* **NLB** (*Network Load Balancing*) do Windows Server permite que o servidor possa ser integrado num *cluster*, designado *NLB Cluster*.

Os clusters NLB de servidores Windows utilizam uma abordagem diferente: todas as máquinas que são membros do *cluster* vão ter o mesmo endereço de nó, que será o endereço do *cluster*.

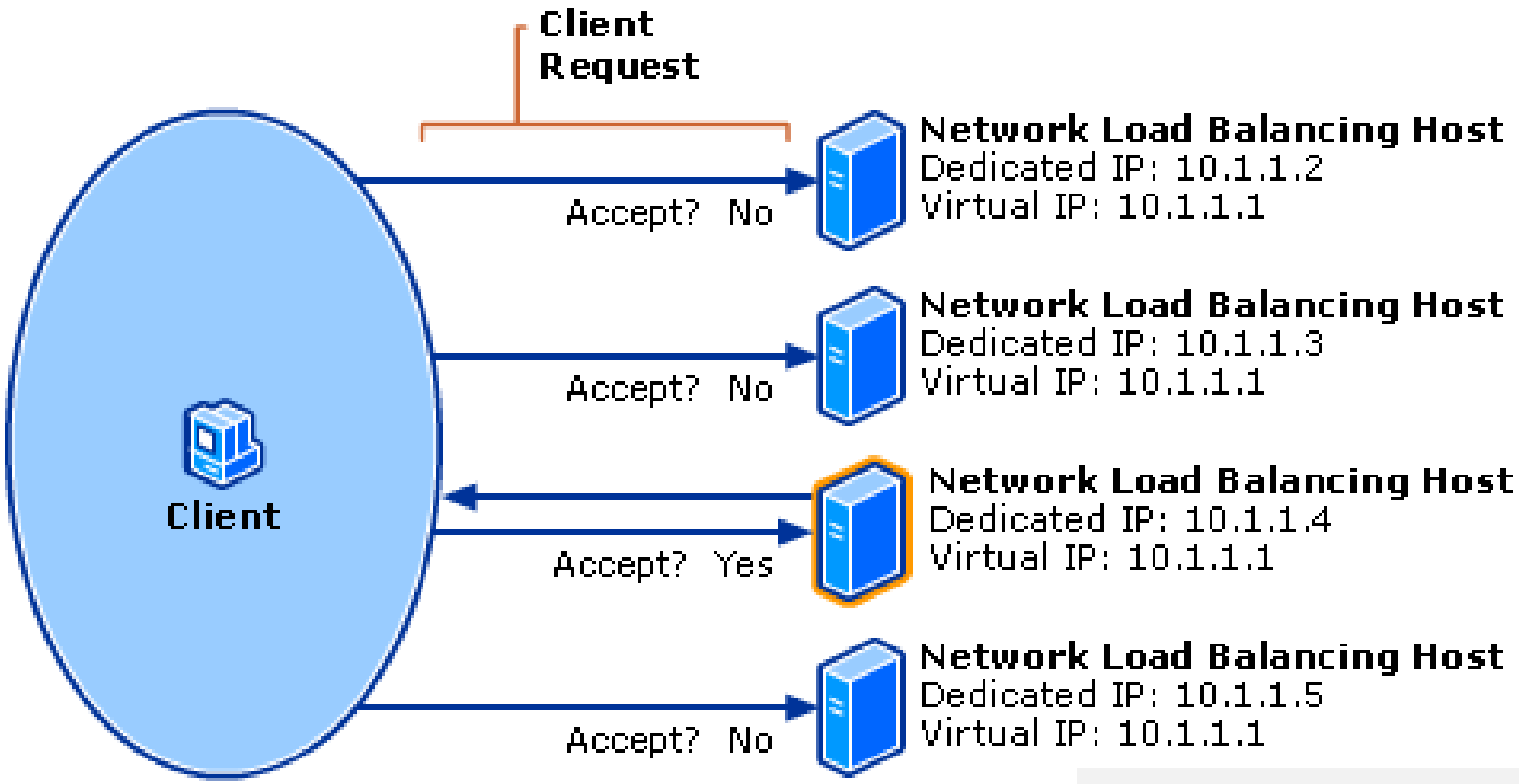
Quando um cliente envia um pedido para o endereço do *cluster*, todas as máquinas que são membros do *cluster* vão receber uma cópia do pedido. Apenas um membro pode responder, isso é estabelecido entre os vários membros do *cluster*.

Se um membro deixa de estar disponível, os pedidos continuam a ser respondidos por outros membros.

A grande vantagem dos *clusters* NLB é a inexistência de um ponto único de falha (SPOF), não existe um controlo centralizado do *cluster*, ele é gerido de forma distribuída por todos os membros. Desde que pelo menos um membro esteja a funcionar, o *cluster* está disponível.

Clusters NLB – Funcionamento da rede

Num *Cluster* NLB todos os membros possuem o mesmo endereço de nó, mas apenas um dos membros responde a cada pedido.



(Fonte: technet.microsoft.com)

Os membros do *cluster* podem manter o seu endereço IP único (*Dedicated IP*), mas todos partilharão um endereço comum entre eles (*Primary / Virtual IP*) que será o endereço do NLB *Cluster*.

Clusters NLB – Unicast nível 2

Estão disponíveis duas formas de garantir que os pedidos dirigidos ao endereço do *cluster* NLB chegam a todos os membros:

- Unicast nível 2 (MAC)
- Multicast no nível 2 (MAC)

A abordagem recomendada pela Microsoft é unicast de nível 2. Nesta configuração, os endereços de nível 2 (MAC/Ethernet) das interfaces de rede de todos os membros do *cluster* são substituídos por um endereço único igual em todos os membros (***inbound destination address***).

Os *frames* emitidos pelos membros do cluster nunca usam este endereço como origem pois isso levaria os comutadores de rede a inclui-lo na sua tabela MAC e os pedidos seguintes deixariam de ser entregues em todos os membros do *cluster*.

Quando um membro do *cluster* emite uma resposta usa como endereço MAC de origem o ***outbound source address*** que é único para cada membro do *cluster*.

Clusters NLB – Multicast nível 2

Esta abordagem é semelhante à anterior, a única diferença é que são utilizados endereços MAC multicast.

Nas interfaces de rede de todos os membros do *cluster* é adicionado um endereços de nível 2 (MAC/Ethernet) multicast, igual em todos os membros (*inbound destination address*). Como é um endereço multicast, o endereço original da interface pode manter-se. Sendo um endereço MAC multicast já não existem problemas com o funcionamento dos comutadores de rede porque eles são sempre obrigados a retransmitir estes *frames* em todas as suas portas.

A tabela anexa apresenta os endereços MAC usados. Enquanto o endereço *inbound* é igual para todos os membros do *cluster*, os endereços de origem (*source outbound*) contêm o valor P que é único para cada membro.

IP Mode	MAC Address	Explanation
Unicast inbound	02-BF-W-X-Y-Z	W-X-Y-Z = IP address Onboard MAC disabled
Multicast inbound	03-BF-W-X-Y-Z	W-X-Y-Z = IP address Onboard MAC enabled
Source outbound	02-P-W-X-Y-Z	W-X-Y-Z = IP address P = Host priority

(Fonte: technet.microsoft.com)

Clusters NLB – modos

A implementação de um *cluster* NLB nunca pode combinar diferentes formas de operação, todos os membros do *cluster* operam da mesma forma, unicast ou multicast.

Seja qual for a opção, o facto de todos os membros do *cluster* usarem o mesmo endereço IP, obriga a que se encontrem todos na mesma rede de nível 2 (domínio de broadcast).

No **modo unicast** o endereço MAC original da interface de rede é desativado e substituído por um endereço na forma 02:bf:…, comum a todos os membros, em consequência da desativação do endereço MAC original o membro deixa de ficar acessível individualmente e fica apenas acessível como *cluster*. Para poder comunicar com individualmente com um membro de um *cluster* unicast é necessário dotar o membro de uma interface de rede adicional, caso contrário para aceder individualmente a um membro é necessário primeiro retirá-lo do *cluster*.

No **modo multicast** o problema anterior não ocorre porque um endereço MAC multicast pode ser adicionado à interface sem desativar o endereço MAC original. Assim a mesma interface de rede pode ser usada para o endereço virtual do *cluster* e para o endereço dedicado do membro.

Clusters NLB – multicast com IGMP

Devido à forma de operarem com os *frames*, garantido que chegam a todos os membros do *cluster*, os *clusters* NLB impedem segmentação pelos comutadores.

Um *cluster* NLB inunda todo o domínio de broadcast com *frames*, o ideal é portanto limitar o máximo possível o domínio de broadcast, ou seja a rede do *cluster* deve ser de preferência uma rede dedicada.

No modo multicast pode ser usado o **IGMP** (*Internet Group Membership Protocol*). Usando a opção ICMP o *multicast inbound destination address* é definido de forma diferente e os membros emitem mensagens IGMP para se registrem em grupos de multicast.

Usar o IGMP apenas tem interesse se no domínio de broadcast existem comutadores de rede que o suportem. Através das mensagens IGMP esses comutadores evitam a propagação de *frames* multicast para locais desnecessários.

Clusters NLB - ARP

Quando um nó localizado na rede do *cluster* (tipicamente o router desse rede) tenta aceder ao endereço IPv4 do *cluster* (*primary or virtual address*) vai tentar obter o endereço MAC correspondente através do **ARP** (*Address Resolution Protocol*). A forma como o *cluster* NLB usa os endereços MAC pode então causar problemas.

Quando os membros do *cluster* respondem, a resposta ARP contém o endereço MAC *destination inbound unicast* ou *multicast* (igual em todos os membros), no entanto a resposta ARP é transportada num *frame* que tem como origem o endereço MAC *source outbound* (único para cada membro).

Este facto pode ser interpretado por alguns nós como uma incoerência e pode levar o nó a ignorar a resposta ARP por motivos de segurança.

Uma solução possível para esta situação é inserir nesses nós um entrada estática na tabela ARP, de forma a mapear o endereço MAC *destination inbound unicast* ou *multicast* para o endereço IPv4 do *cluster* (*primary or virtual address*).

Clusters NLB – Priority

Na implementação atual, um *cluster* NLB pode conter até 32 membros, cada membro tem de usar um número de prioridade diferente de 1 a 32.

O número de prioridade é definido pelo administrador no momento em que adiciona um membro ao *cluster*, o valor 1 significa prioridade mais elevada. Dependendo das regras de portos (*port rules*), o membro de prioridade mais elevada pode receber todos os pedidos dos clientes.

O número de prioridade também identifica o membro no *cluster*, sendo por exemplo colocado nos endereços MAC *source outbound* dos *frames* emitidos por esse membro.

O *cluster* NLB tem como objetivo receber tráfego TCP e UDP destinado a serviços de rede e distribuir esse tráfego pelos seus membros, a forma como essa distribuição é realizada é definida nas *port rules*.

As *port rules* apenas se aplicam a UDP e TCP, por exemplo o tráfego ICMP é sempre enviado a todos os membros do *cluster* (apenas o de prioridade mais elevada responde aos pedidos).

Clusters NLB – *Port rules*

Um *cluster* pode ter vários endereços (*primary or virtual address*), para cada um deles podem ser definidas diferentes *port rules*, no entanto todos os membros de um *cluster* têm de ter as mesmas *port rules*, quando se adiciona um novo membro ele adquire as *port rules* do *cluster*.

Para alterar as *port rules* num membro ele terá de ser retirado do *cluster*, as regras alteradas e novamente adicionado, estas operações podem ser realizadas sem perturbar o funcionamento do *cluster* sob o ponto de vista dos utilizadores.

Para cada endereço do *cluster* podemos definir várias *port rules*, cada uma especifica um intervalo de números de porto UDP e/ou TCP. Os intervalos em diferentes regras aplicadas ao mesmo endereço não se podem sobrepor.

O objetivo de uma *port rules* é definir a forma de lidar com o tráfego correspondente que chega ao *cluster*, existem três alternativas (*filtering modes*):

- ***Multiple Host*** – o tráfego é distribuído pelos membros
- ***Single Host*** – modo de *failover*
- ***Disable*** – o tráfego não é permitido

Clusters NLB – *filtering mode single host*

O modo *default* é *single host*, isto é, aplica-se a qualquer tráfego que não corresponda a nenhuma *port rule*. Isto inclui desde logo tráfego que não pode ser definido numa *port rule*, como por exemplo ICMP.

O *filtering mode* ***single host*** corresponde a um comportamento de ***failover***, todo o tráfego é dirigido para o membro disponível com prioridade mais elevada, se ele fica indisponível passará a ser utilizado o membro disponível com prioridade mais elevada seguinte. Depois de adicionado ao *cluster*, para cada membro é possível definir um valor de prioridade (1 a 32) específico para cada *port rule*.

O *filtering mode* ***multiple host*** permite a distribuição do tráfego por todos os membros disponíveis.

Neste modo é também possível definir para cada membro a percentagem de pedidos que lhe são dirigidos, por omissão os pedidos são distribuídos de forma equitativa.

A definição da percentagem de pedidos atribuídos a cada membro só pode ser realizada nas propriedades do membro depois de adicionado ao *cluster* e não nas propriedades do *cluster*.

Clusters NLB – *multiple host e affinity*

No *filtering mode* ***multiple host*** é fundamental definir a persistência das associações entre clientes e membros do *cluster*. A *affinity* permite garantir que quando um primeiro pedido de um cliente é atribuído a um membro, os pedidos seguintes desse mesmo cliente também serão atribuídos ao mesmo membro. Sem esta garantia muitos protocolos de aplicação como por exemplo os que operam sobre TCP não podem funcionar.

Os modos de *affinity* disponíveis são:

none – todos os pedidos são distribuídos pelos membros, só pode ser usado para serviços sem ligação muito simples. Por exemplo serviços UDP com um pedido e uma resposta.

single – todos os pedidos provenientes do mesmo endereço de origem (cliente) são enviados para o mesmo membro.

network – todos os pedidos provenientes da mesma rede de classe C (prefixo de 24 bits) são enviados para o mesmo membro.

Nos modos ***single*** e ***network*** pode ser especificado um *timeout* em minutos. Se for definido, quando o cliente não comunica durante esse período de tempo, a persistência é eliminada e numa próxima comunicação será atribuído um novo membro ao cliente.

Sincronização dos membros do *cluster*

Como os membros de um *cluster* devem aparentar ser um único servidor para os clientes, os serviços disponibilizados por cada um deles devem ser dotados de mecanismos de sincronização que garantam que todos os membros possuem o mesmo estado.

A *affinity* ou *stickiness*, garante que durante uma sessão de trabalho de um cliente ele vai trabalhar sempre com o mesmo membro, mas isso pode não ser suficiente.

Por exemplo quando um utilizador faz um *upload* de um conteúdo para um site web que está implementado sob a forma de *cluster*, espera que esse conteúdo fique disponível a todos os utilizadores do *site/cluster*.

A sincronização dos membros do *cluster* deve ser implementada caso a caso para cada tipo de serviço disponibilizado, por exemplo podem ser usados sistemas de ficheiros em rede (NAS) para esse efeito, os servidores de bases de dados possuem normalmente mecanismos próprios de replicação.

Sincronização de certificados de chave pública

Desejavelmente, sob o ponto de vista de segurança, muitos protocolos de aplicação usam atualmente SSL/TLS. Nesse contexto o servidor fornece ao cliente um certificado de chave pública a ser validado pelo cliente para efeitos de autenticação do servidor.

O certificado de chave pública contém o nome DNS do servidor, num *cluster*, para que o certificado seja considerado válido pelo cliente, esse nome DNS tem de corresponder ao endereço IP do *cluster* (endereço usado pelos clientes no acesso ao *cluster*).

Consequentemente, todos os membros do *cluster* terão de usar esse mesmo certificado.

No caso do HAProxy a instalação e manutenção do mesmo certificado em todos os membros pode ser evitada. Quando o HAProxy opera em modo HTTP, pode ele próprio fornecer o certificado ao cliente, neste caso a ligação entre o cliente e o HAProxy usa HTTPS enquanto a ligação correspondente do HAProxy ao membro pode usar HTTP.