Administração de Sistemas (ASIST)

Administração de Servidores Linux

Aulas Teórico Práticas - 2016/2017

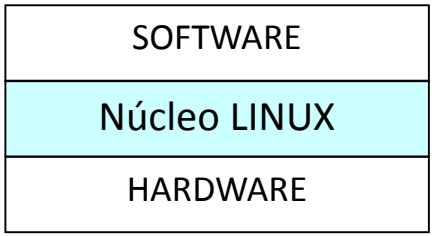
Instalação de distribuições Linux

O núcleo (*kernel*) do sistema operativo Linux implementa as funcionalidades básicas mais importantes, tais com gestão de dispositivos, processos e memória e interage diretamente com o *hardware*.

Pode-se dizer que "**o núcleo é o sistema operativo**". O *kernel* Linux está em permanente desenvolvimento como código aberto, e gratuito.

O núcleo fornece acima de tudo uma plataforma estável para que o *software* de nível mais elevado possa funcionar sem conhecer detalhes sobre o *hardware*.

Para se poder tirar partido das capacidades do núcleo é necessário recorrer a um conjunto extenso de programas auxiliares interage com o *kernel* e permitem obter um sistema funcional.



Alguns destes programas auxiliares têm funções criticas de arranque, como por exemplo carregar o núcleo para a memória e passar-lhe o controlo. Ou até ajudar no processo de instalação do sistema operativo na máquina.

Ao conjunto NÚCLEO + PROGRAMAS dá-se o nome **Distribuição**, as distribuições não são necessariamente grátis.

A maioria das distribuições atuais dispõe de programas de instalação assistida de nível bastante elevado que protegem o utilizador de questões para as quais eventualmente não sabe as respostas.

A solução de instalação expedita quase sem opções, tem a vantagem de trazer o LINUX para os utilizadores correntes, mas para os administradores pode ter inconvenientes.

As decisões que o programa de instalação toma autonomamente, sem perguntar, podem comprometer determinados objetivos específicos.

✓Felizmente em muitos caso é possível optar por um modo "*expert*" no qual o programa de instalação tome menos decisões e faz mais perguntas.

✓Por outro lado a maioria das configurações pode ser alterada após a instalação inicial.

Embora as configurações criadas pelo processo de instalação possam sempre ser alteradas mais tarde, existem alguns pontos em que isso pode ser mais complicado. Um dos mais relevantes é a divisão dos discos em partições, depois de concluída a instalação, a sua alteração é mais complexa.

O ideal é que não haja necessidade de alterar as partições criadas durante a instalação, particularmente as partições que são usadas no arranque do sistema. Tipicamente o administrador deverá optar pelo "particionamento" manual dos discos.

Uma questão fundamental com que se depara um administrador que pretende instalar um servidor Linux 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 localização dos ficheiros de configuração.

Normalmente as distribuições estão disponíveis em duas versões: ***server*** e ***desktop/workstation***. 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 Linux 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 mantidas atualizações durante 5 anos.

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

Discos e partições em Linux

No sistema operativo Linux a maioria dos recursos de *hardware* são identificados por entradas especiais no diretório de sistema "**/dev**". Os vários discos existentes são identificados de acordo com o tipo de interface que utilizam.

Discos IDE (em desuso): **/dev/hda** **/dev/hdb** **/dev/hdc** **/dev/hdd**

Discos SCSI ou SATA : **/dev/sda** **/dev/sdb** **/dev/sdc** **/dev/sdd** **…**

A maioria dos discos (não os CD/DVD e outros discos móveis) está dividida em partições. Trata-se de uma divisão lógica do disco que está definida na **tabela de partições**. Esta tabela encontra-se numa zona inicial do disco designada *Master Boot Record* (MBR).

Cada partição existente num disco é independente das restantes, podendo cada uma conter tipos de dados (formatações) totalmente diferentes, eventualmente associadas a sistemas operativos diferentes. Entre outras informações a tabela de partições contém associado a cada partição um identificador do formato da mesma.

As partições existentes num disco são identificadas por números crescentes desde 1.

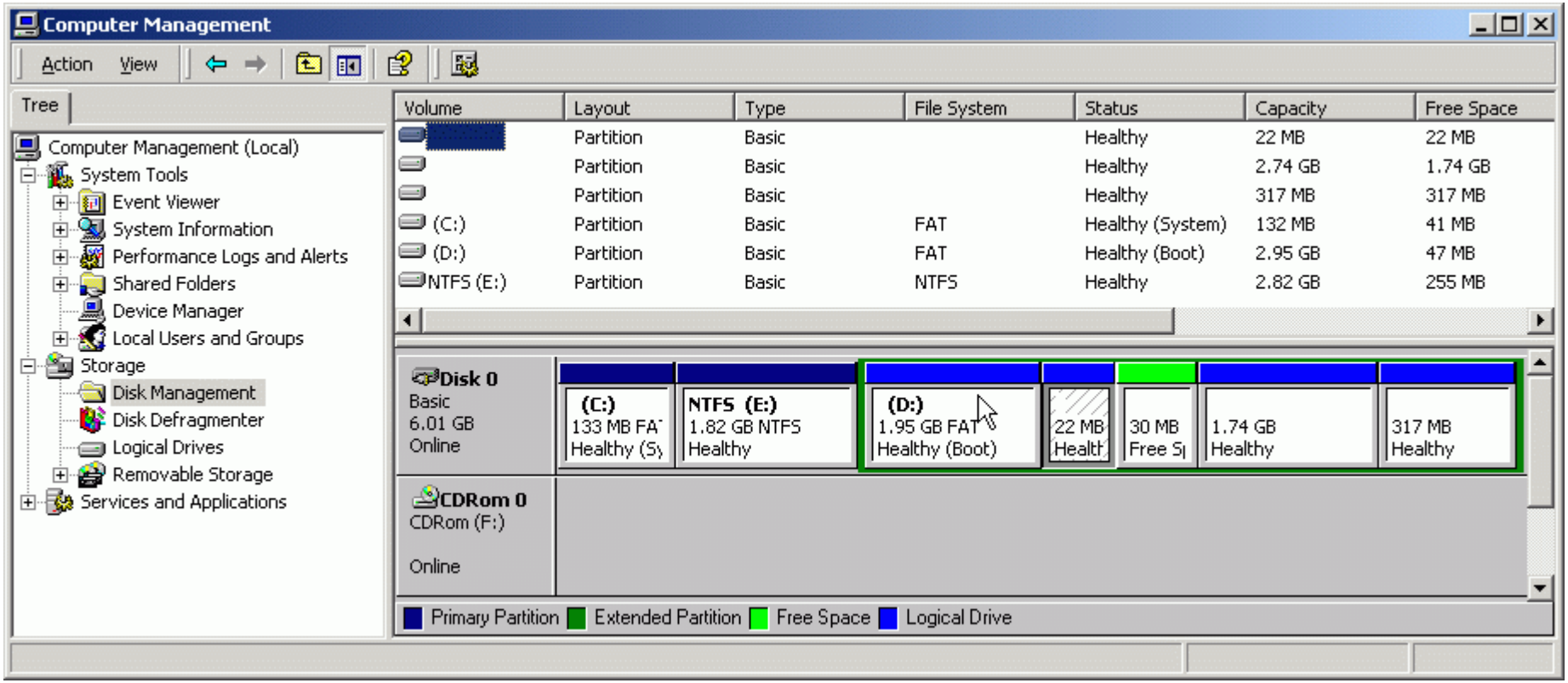
Por exemplo: **/dev/sda2** representa a 2ª partição do primeiro disco (SATA ou SCSI).

Tabelas de partições – comando *fdisk* (Linux)

```
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
```

```
HOST12# fdisk /dev/hda
Command (m for help): p
Disk /dev/hda: 20.0 GB, 20003880960 bytes
255 heads, 63 sectors/track, 2432 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device    Boot      Start     End       Blocks    Id        System
/dev/hda1 *         1         13        104391    83        Linux
/dev/hda2           14        778       6144862+  83        Linux
/dev/hda3           779       1415      5116702+  83        Linux
/dev/hda4           1416      2432      8169052+  5         Extended
/dev/hda5           1416      1925      4096543+  83        Linux
/dev/hda6           1926      1990      522081    82        Linux swap
/dev/hda7           1991      2432      3550333+  83        Linux
```

Tabela de partições – *Computer Management* (Windows)



Uma partição é uma zona contigua de um disco, cujos limites (início e fim) estão definidos na tabela de partições.

A tabela de partições associa a cada partição um identificador do formato do sistema de ficheiros que lá existente para que o sistema operativo saiba como deve interpretar os dados que lá se encontram.

Formatação das partições

Definir uma nova partição (por exemplo recorrendo ao comando *fdisk* em Linux) 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).

Depois de *formatada*, cada partição pode ser usada pelo sistema operativo para guardar dados, tipicamente organizados em diretórios (pastas) e ficheiros. A forma como os dados são organizados e guardados é conhecida por **formato do sistema de ficheiros** (***file system***).

Existem muitos formatos de sistema de ficheiros. Na atualidade nos ambientes Windows da Microsoft usa-se o **NTFS**, embora ainda se use também **FAT**. No sistema operativo Linux o mais usado é o **EXT4**. Uma vez que as partições são totalmente independentes podemos ter no mesmo disco partições com diversos tipos de sistemas de ficheiros.

O sistema de ficheiros guarda vária informação 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.

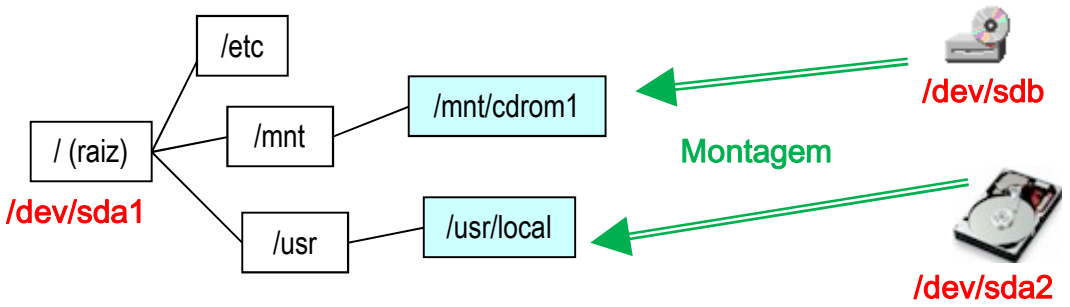
Acesso aos sistemas de ficheiros

A forma lógica como os sistemas operativos facultam acesso aos sistemas de ficheiros residentes nas partições dos dispositivos de armazenamento existentes varia. Nos sistemas DOS/Windows usam-se as **letras de drive**. As letras **A:** e **B:** foram tradicionalmente reservadas para os discos flexíveis (em desuso) e por isso as partições existentes 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, vão ser integrados no sistema de ficheiros de raiz através de uma operação designada por montagem.

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.



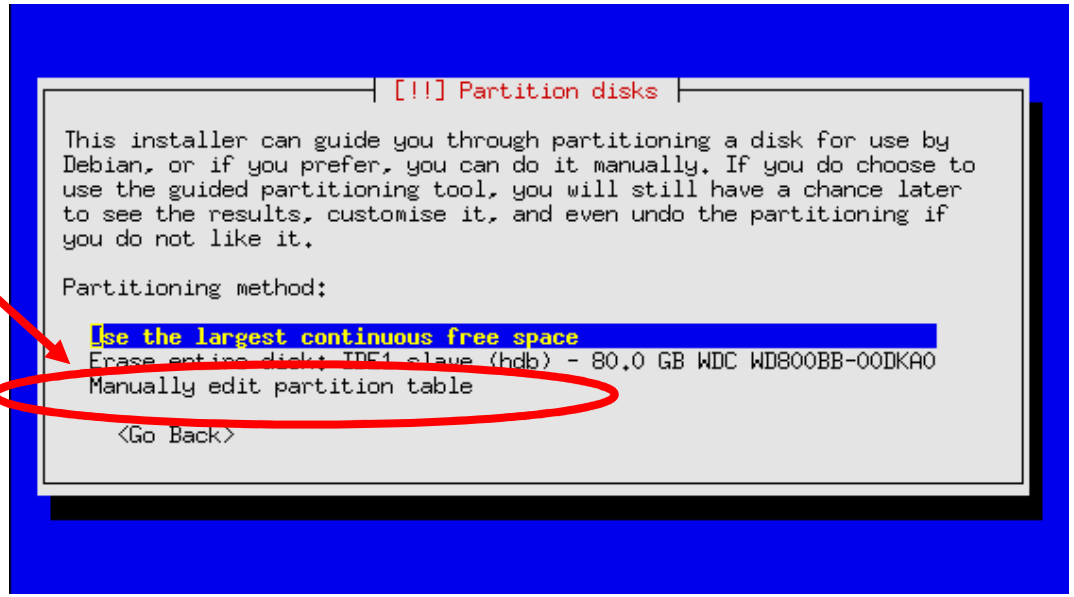
Planeamento das partições e sistemas de ficheiros

Os programas de instalação, encarregam-se de analisar os discos existentes, e em função disso definir as partições e características dos sistemas de ficheiros a criar. Na maioria dos casos tomam as decisões corretas, mas falta-lhes o conhecimento sobre os propósitos exatos do sistema.

Um administrador que inicia a instalação de um servidor Linux tem objetivos específicos para esse servidor, a esses objetivos corresponde um conjunto de requisitos, nomeadamente quanto às partições necessárias.

Felizmente na maioria dos casos é dada oportunidade de intervir durante a instalação no sentido de fazer escolhas relativas às partições a criar.

No exemplo ao lado, através da alteração direta da tabela de partições. Em outros casos são dados várias vários modelos possíveis a escolher.



Capacidade das partições

Sendo complexo alterar as partições depois de concluída a instalação, o ideal é que desde logo elas sejam corretamente dimensionadas para os objetivos pretendidos.

Numa primeira análise podemos afirmar que o aproveitamento ideal do disco é obtido com uma partição única. Isso é verdade porque as partições são limites estáticos que não podem ser ultrapassados sob o ponto de vista de capacidade. Se uma partição de um disco ficar cheia deixa de ser possível escrever, mesmo que no mesmo disco ainda exista espaço livre ou uma partição pouco utilizada.

Sob o ponto de vista de segurança estes limites são vantajosos pois permitem atribuir a diferentes partes do sistema espaços próprios no disco, impedindo perturbações entre elas.

Por exemplo, se as áreas de armazenamento pessoal dos utilizadores (*homes*) se encontrarem numa partição diferente do sistema (raiz), então se um utilizador ocupar espaço abusivamente e encher a partição *homes*, isso não vai afetar a partição de sistema.

Embora afete os outros utilizadores pois a partição fica cheia e deixarão de poder escrever nas suas áreas de armazenamento pessoais.

Capacidade das partições e cotas

Para evitar que o uso abusivo de espaço em disco por parte de um utilizador perturbe os restantes seria necessário criar uma partição para cada utilizador. **Claramente essa solução não é praticável.**

A alternativa é o sistema de **cotas** em que o sistema operativo mantém um controlo constante sobre o espaço que cada utilizador (cota de utilizador) ou grupo de utilizadores (cota de grupo) está a usar **numa dada partição**.

AS COTAS SÓ PODEM SER DEFINIDAS AO NÍVEL DE PARTIÇÃO

O sistema de cotas permite limitar o espaço que cada utilizador ou grupo ocupa **em cada partição**. Não é possível definir cotas:

- num disco inteiro (a menos que o disco tenha apenas uma partição)
- num conjunto de partições (é necessário definir cotas uma a uma em cada uma das partições)
- numa pasta isolada

O facto de as cotas apenas funcionarem ao nível de partição condiciona a forma como o disco deve ser dividido em partições.

Não é boa ideia ter as áreas dos utilizadores na partição de raiz e estabelecer cotas nessa partição pois quando ficar sem cota o utilizador vai ter dificuldades em usar o sistema por não conseguir escrever na partição de raiz.

Muitas vezes é necessário atribuir não uma mas várias cotas de espaço em disco independentes, por exemplo:

- Cota área de pessoal do utilizador
- Cota de área WEB
- Cota para correio eletrónico
- Cota na partição de sistema

A única forma de se conseguir este objetivo é preparar uma partição diferente para armazenamento de dados relativo a cada uma das cotas.

Arranque do sistema Linux (*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 tipicamente 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 o **/sbin/init** deste programa inicial vão surgir todos os processos em execução no sistema.

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

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 atualmente 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.

Capacidade da partição RAIZ

A partição RAIZ destina-se ser a base do sistema de ficheiros, durante a fase inicial de arranque do sistema operativo é tudo de que o núcleo dispõe, só mais tarde é possível montar outras partições.

A partição RAIZ deve ter capacidade para conter todo o sistema operativo base, incluindo elementos cujas dimensões podem crescer ao longo da vida do sistema tais como ficheiros de configuração, dados e registos.

Na prática, atendendo às capacidades atuais dos dispositivos de armazenamento, num servidor baseado em discos correntes, não faz qualquer sentido usar uma partição de raiz de tamanho inferior a 8 Gbytes.

Aumentar o espaço disponível no sistema é relativamente simples pois basta adicionar (montar) novas partições, eventualmente residentes em novos discos. Este procedimento pode depois ser rematado com a utilização de ligações simbólicas para produzir a estrutura de ficheiros necessária.

Capacidade das partições SWAP

A **partição SWAP** funciona como memória virtual, 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.

Por outras palavras a quantidade de memória central de um sistema servidor deve ser tal que a memória virtual apenas seja usada muito esporadicamente. Por esta razão 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**.

Desde que exista espaço em disco livre (não atribuído a nenhuma partição), é sempre possível acrescentar novas partições SWAP pois o sistema operativo tem capacidade de usar simultaneamente várias destas partições, eventualmente em discos distintos.

Tipos de sistema de ficheiros

O sistema operativo LINUX suporta uma grande variedade de sistemas de ficheiros, atualmente o mais usado é o ext4 (*fourth extended filesystem*).

A maioria dos programas de instalação de distribuições LINUX atuais assume o ext4 como valor por omissão.

Tal como acontece para as partições do disco, é normalmente possível intervir durante a instalação para alterar o tipo de sistema de ficheiros. Algumas das alternativas atuais ao ext4 que têm vantagens interessantes, são o *ReiserFS* e o XFS.

Todos estes sistemas de ficheiros recentes são do tipo *journaling*, isso significa que as alterações realizadas são guardadas num registo e apenas são transferidas para o sistema de ficheiros em momentos oportunos.

Na prática isto significa que após uma falha grave não há necessidade de verificar a integridade da totalidade do sistema de ficheiros, tal como acontecia por exemplo com o ext2.

RAIZ inicial e módulos do *kernel*

Dada a enorme variedade de capacidades do núcleo Linux, por uma questão de economia de recursos não é desejável incluir o suporte integral de todas elas. O *kernel* Linux é totalmente modular, ao compilar, para a maioria das funcionalidades, existem 3 opções: "incluir no kernel"; "suportar através de módulo externo"; "não suportar".

Na primeira opção o *kernel* vai suportar autonomamente a funcionalidade. Na segunda opção reconhece a funcionalidade, mas para a suportar necessita de recorrer a um módulo externo KLM (*Kernel Loadable Module*).

Consequentemente, o facto de um tipo de sistema de ficheiros ser suportado pelo Linux não significa que o *kernel* o consiga usar como raiz inicial. Se o suporte exige o carregamento do KLM correspondente isso constitui um problema porque ainda não existe sistema de ficheiros a partir do qual se possa carregar o ficheiro que contém o KLM.

Os núcleos incluídos nas distribuições actuais suportam ext3 e ext4 sem recurso a módulos, contudo o mesmo não se aplica necessariamente aos sistemas de ficheiros *ReiserFS* e *xfs*.

É claro que a utilização de um sistema de ficheiros raiz inicial em *ramdrive* resolve este tipo de problemas porque os KLM podem ser colocados no *ramdrive*.

Formatação de sistemas de ficheiros

Em Linux a formatação dos sistema de ficheiros recorre a programas externos com o nome *mkfs* seguido do tipo de sistema de ficheiros, por exemplo *mkfs.ext3* ou *mkfs.ext4*. O comando *mke2fs* é a versão geral que invoca diretamente essas aplicações:

mke2fs [-c | -l filename] [-b block-size] [-f fragment-size] [-g blocks-per-group] [-G number-of-groups] [-i bytes-per-inode] [-I inode-size] [-j] [-J journal-options] [-K] [-N number-of-inodes] [-n] [-m reserved-blocks-percentage] [-o creator-os] [-O feature[,...]] [-q] [-r fs-revision-level] [-E extended-options] [-v] [-F] [-L volume-label] [-M last-mounted-directory] [-S] **[-t fs-type] [-T usage-type]** [-U UUID] [-V] **device** [blocks-count]

A opção **–t fs-type** indica o tipo de sistema de ficheiros a criar, por exemplo *ext3* ou *ext4*. O parâmetro ***device*** identifica a partição onde se pretende criar o sistema de ficheiros, por exemplo **/dev/sdd2**. A maioria das restante opções serão normalmente omitidas, de entre elas destaque para a opção **-T *usage-type*** em que **usage-type** é um dos valores possíveis presentes no ficheiro de configuração **/etc/mke2fs.conf**, por exemplo *largefile*. Na realidade cada um destes valores representa um conjunto de "afinações" dos outros parâmetros, por exemplo quanto ao tamanho dos blocos (*block-size*) em que valores elevados permitem melhor performance com um pequeno desaproveitamento de espaço.

Tabela de sistemas de ficheiros - /etc/fstab

Quando o *kernel* 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 na seguinte forma:

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**

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*.

Tabela de sistemas de ficheiros - /etc/fstab - 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 da interface de rede

Num sistema servidor as interfaces de rede devem ser configuradas estaticamente

Utilizar configuração dinâmica, 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 estática da interface de rede exige a colaboração do administrador da rede que deverá fornecer os vários elementos:

- Endereço IP estático + mascara da rede + endereço do encaminhador da rede

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

A única desvantagem da configuração estática 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.

Formas de instalação do LINUX

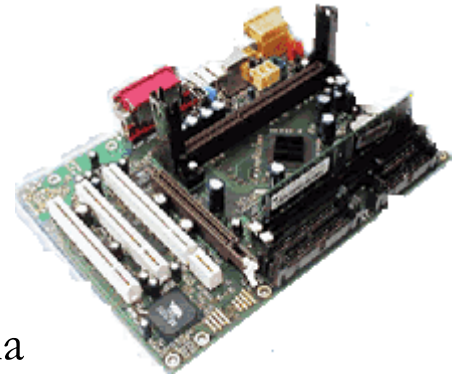
As distribuições LINUX disponibilizam várias formas de instalação, a maioria delas baseada no próprio LINUX.

O arranque do sistema de instalação (LINUX) é normalmente realizado através de um CD/DVD. Na realidade qualquer forma de arranque do LINUX poderá ser usada, por exemplo diretamente do MS-DOS/MS-WINDOWS ou através de uma ROM.

As distribuições atuais são compostas por um grande volume de software (atualmente muitas ultrapassam as duas dezenas de *CDs*). Contudo, na prática a maioria das instalações recorre apenas a uma fração deste software.

A "**instalação de rede**" usa apenas um CD (ou até menos do que isso …) para colocar o sistema de instalação em funcionamento e de seguida usa a rede (ligada à *Internet*) para obter o *software* em repositórios (*mirrors*) criados para este efeito. Desde que se possua uma ligação à *Internet* com capacidades mínimas, esta é sem dúvida uma alternativa muito prática, os mesmos repositórios serão mais tarde usados para manter o servidor atualizado ou instalar novo pacotes.

Instalação de Linux - *hardware*



O variedade de *hardware* existente é o responsável pela maioria das dificuldades na instalação de distribuições Linux.

Se o *hardware* é muito recente devemos ter o cuidado de usar uma distribuição com núcleo o mais recente possível para haver alguma garantia que esse *hardware* é suportado.

O procedimento correto é escolher o *hardware* para o *software* que vamos usar, assim antes de adquirir o *hardware* temos de verificar se a distribuição Linux que vamos usar suporta esse *hardware*.

A aquisição de *hardware* "especial" de elevado custo pode terminar numa desilusão quando se chega à conclusão que não há sistemas operativos capazes de tirar partido dele. Os controladores de disco são uma fonte de problemas pois se não são suportados pelo núcleo do sistema de instalação, esta torna-se de todo impossível.

Com a virtualização de servidores estes tipo de dificuldades não existe porque a máquina virtual simula *hardware* de uso corrente facilmente reconhecido pelo sistema operativo.

Bases de dados de utilizadores - /etc/passwd

A forma tradicional de um sistema operativo UNIX guardar as definições relativas a utilizadores é através do ficheiro de texto **/etc/passwd**.

```
bash-3.00$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
nobody:x:65534:65534:Nobody:/:/bin/sh
sshd:x:77:77:system user for openssh:/var/empty:/bin/true
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
```

Como se pode observar cada linha constitui um registo com vários campos separados pelo símbolo ":" na seguinte ordem:

- Nome curto (*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 de trabalho ou pasta inicial (*home*). Pasta corrente após o *login*.
- Programa inicial. Tipicamente uma *shell* executada após o *login*.

Bases de dados de utilizadores - /etc/shadow

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**, se são iguais a autenticação é válida.

O sistema operativo exige que o ficheiro **/etc/passwd** seja legível por todos os utilizadores consequentemente conclui-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 *passwords* usando o *hash* da *password*, 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 se obtém o *hash* da *password* que está no **/etc/passwd**. Em lugar de *passwords* completamente aleatórias o atacante poderia ainda usar um gerador de *passwords* prováveis, nesse caso o ataque designa-se de dicionário.

Para evitar esta vulnerabilidade os *hash* das *passwords* no **/etc/passwd** podem ser substituídos pelo símbolo **x** que significa o *hash* das *password* se encontra no ficheiro auxiliar /**etc/shadow**, ao contrário do anterior este ficheiro não é legível para todos os utilizadores.

Bases de dados de grupos - /etc/group

Os grupos de utilizadores facilitam a administração dos sistemas pois alterando propriedades tais como permissões de um grupo essa alteração aplica-se a todos os membros do grupo.

Em Unix cada utilizador pertence sempre a pelo menos um grupo, designado **grupo primário do utilizador**. O grupo primário de cada utilizador está registado no ficheiros **/etc/passwd** através do correspondente GID.

À semelhança dos utilizadores, a forma tradicional de um sistema operativo Unix guardar as definições de grupos é através do ficheiro de texto **/etc/group**.

```
-bash-3.00$ cat /etc/group
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
```

Como se pode observar o formato é idêntico ao do **/etc/passwd**. O 1º campo é o nome de grupo e o 3º é o GID ambos estes campos têm valores únicos que identificam o grupo.

O 2º campo foi historicamente usado para definir uma *password* para o grupo, mas atualmente não é suportado.

Lista de membros de um grupo

```
-bash-3.00$ cat /etc/group
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
```

O 4º campo do ficheiro **/etc/group** contém uma lista (separada por virgulas e sem espaços) dos nomes dos utilizadores que são membros do grupo e têm um grupo primário diferente desse.

Resumindo

Um utilizador pode ser membro de um grupo por duas vias:

- Porque é o seu grupo primário (GID no 4º campo do **/etc/passwd**)
- Porque o seu nome está na lista de membros (nome no 4º campo do **/etc/group**)

Gestão de utilizadores e grupos locais

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 aprovada e realizarem algumas validações e operações acessórias.

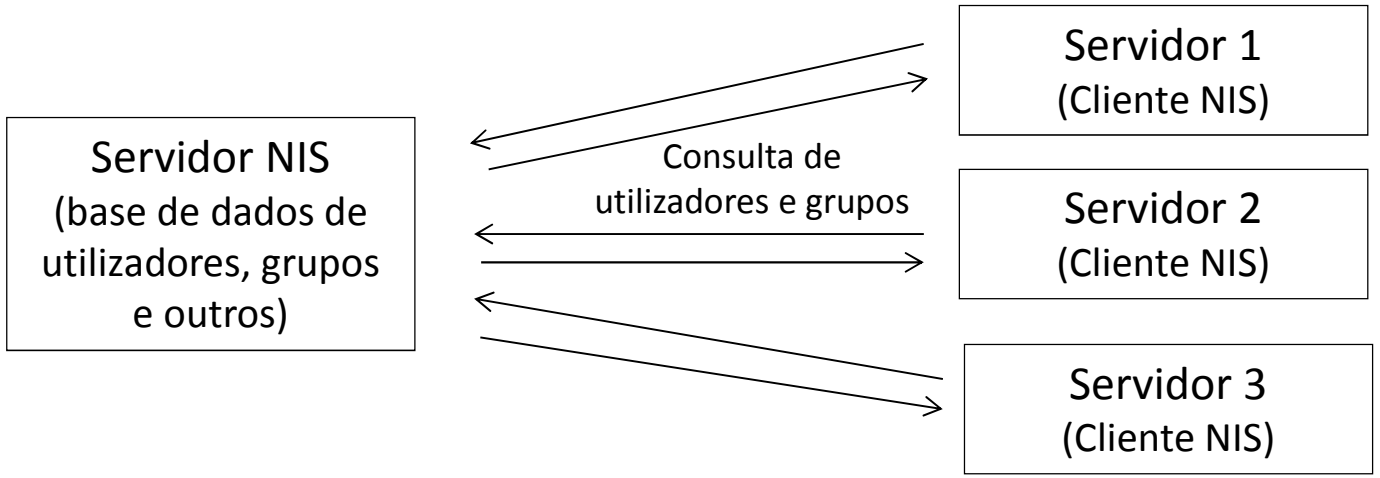
A maioria das distribuições atuais dispõe de programas de gestão de utilizadores em modo gráfico que permitem realizar a maioria das tarefas básicas.

A terceira alternativa consiste na manipulação direta dos ficheiros, ou seja usar um editor de texto para alterar os ficheiros **/etc/passwd** e **/etc/group** esta forma de atuar envolve algum risco pois há necessidade de evitar qualquer tipo de incoerência ou sobreposição de identificadores. 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".

Diversificação dos repositórios de utilizadores e grupos

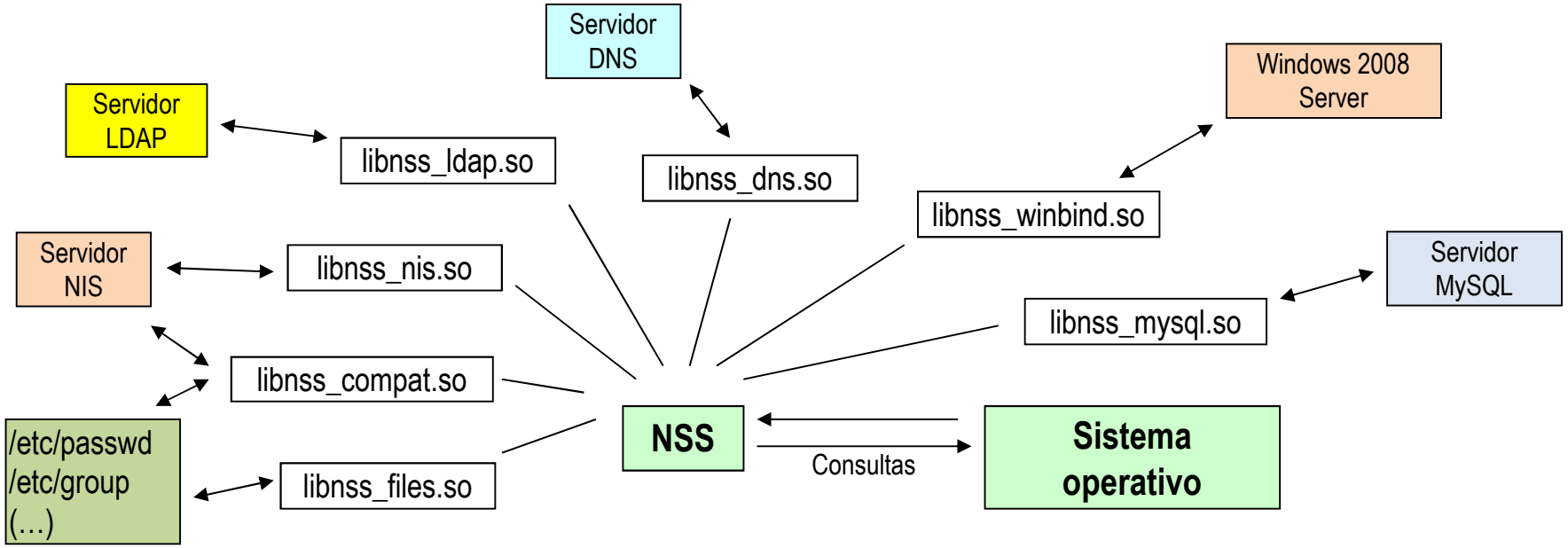
Nos sistemas **Unix** desde cedo se sentiu a necessidade de diversificar e centralizar os repositórios de informação de configuração do sistema (entre outros, bases de dados de utilizadores e grupos), essa necessidade levou ao desenvolvimento de sistemas alternativos, um dos mais importantes é o sistema 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 Linux desenvolveu e flexibilizou este conceito através do serviço NSS, trata-se de um sistema modular em que através da adição de um novo módulo (biblioteca dinâmica) se acrescenta o suporte para um novo tipo de repositório.



Nem todos estes módulos são vocacionados para 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.

Name Service Switch – gestão de utilizadores e grupos

Quando os utilizadores não estão definidos nos ficheiros locais (/etc/passwd; …), a maioria dos comandos usados para administração de utilizadores locais deixa de poder ser usada. A sua administração passa a ser realizada nos repositórios através de ferramentas específicas para cada tipo de repositório, tipicamente trata-se de gerir registos numa base de dados.

A maioria dos módulos do NSS necessita de parâmetros específicos de configuração, esta configuração é específica de cada módulo e não é da responsabilidade do NSS.

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**, entre outros elementos neste ficheiro 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

Name Service Switch – configuração

O serviço NSS é configurado através do ficheiro **/etc/nsswitch.conf**, o principal papel é definir quais os módulos que vão ser usados e a ordem pela qual vão ser usados, ou seja a ordem pela qual os vários repositórios disponíveis vão ser usados.

Para cada tipo de informação define-se 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***)

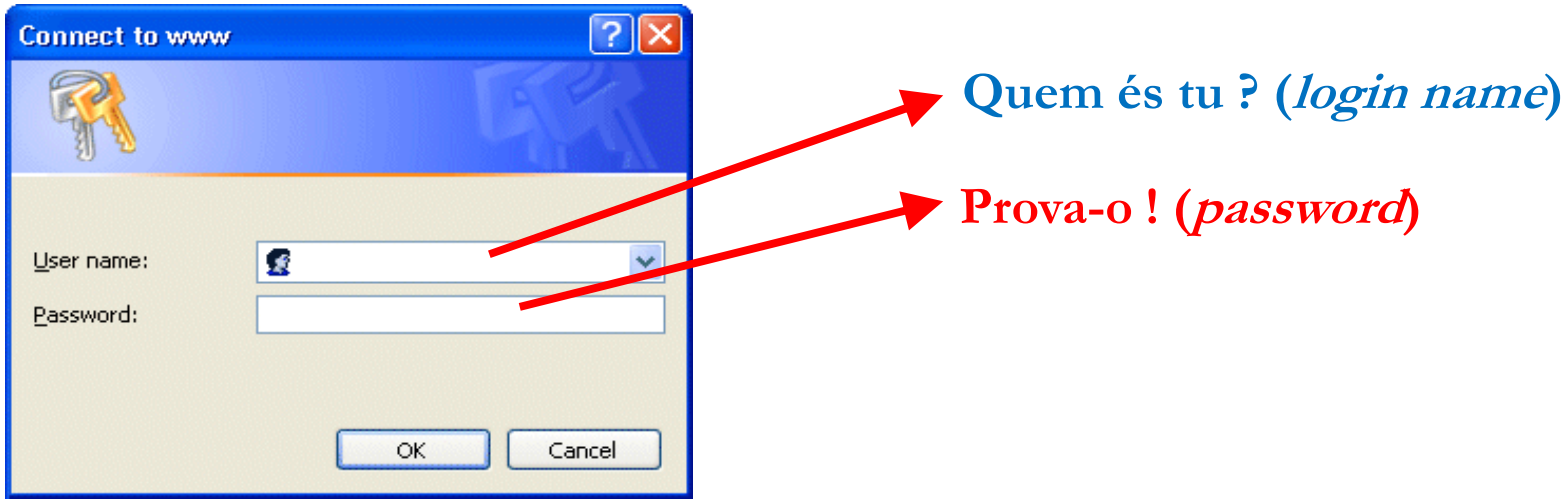
3º Servidores Windows (***winbind***)

4º Servidores NIS (***nis***)

Será usado o primeiro que for encontrado terminando ai a pesquisa

Autenticação dos utilizadores

Apesar das suas fraquezas, o mecanismo de autenticação de utilizadores mais usado continua a ser a associação de uma palavra secreta (*password*) ao nome de utilizador.



Devido ao seu carácter secreto, o manuseamento das *passwords* dos utilizadores tem de ser tratado com cuidados especiais. Como as *passwords* são normalmente armazenadas juntamente com as restantes definições dos utilizadores, a diversificação dos tipos de repositório suportado veio complicar a autenticação.

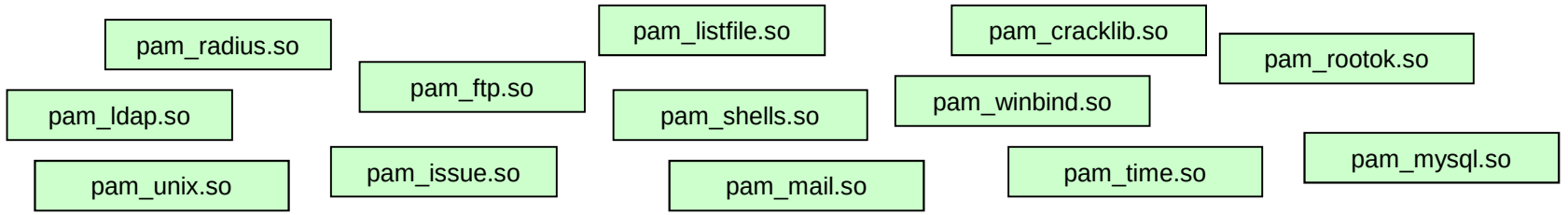
Como resposta a estes e outros desafios foi criado um serviço destinado exclusivamente aos procedimentos de autenticação e *login*, o *Pluggable Authentication Module* (PAM).

Pluggable Authentication Module (PAM)

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 base de dados (PAP – *Password Authentication Protocol*). Mesmo usando algoritmos *hash* recentes como o MD5 ou o SHA, o melhor é manter os *hash* secretos, foi com esse objetivo que esta informação foi transferida do ficheiro **/etc/passwd** para o ficheiro **/etc/shadow**.

Além da necessidade da manter confidencial o *hash*, o facto de o sistema não ter acesso à *password* (apenas o *hash*), impede a implementação de mecanismos de autenticação mais sofisticados como o CHAP (*Challenge Handshake Authentication Protocol*). O sistema PAM introduziu uma grande flexibilidade permitindo a utilização de diversos mecanismos e suportando diversos tipos de repositório.

Tal como o NSS trata-se de um sistema modular, cada modulo implementa determinada funcionalidade relacionada com o processo de autenticação e *login*.:



PAM – tarefas (auth; account; password; session)

O sistema PAM define 4 **tipos de tarefa** relacionadas com a autenticação e *login*:

Autenticação (**auth**) – trata-se da autenticação propriamente dita, tipicamente através de *username/password*

Acesso (**account**) - consiste em validar o acesso. Esta tarefa pode ser invocada antes da autenticação, nesse caso aplica-se a todos as tentativas de acesso, por exemplo restringir o horário de acesso ou endereços de origem. Pode também ser aplicada depois da autenticação nesse caso aplica-se ao utilizador que se autenticou, por exemplo verificando se a respetiva password expirou.

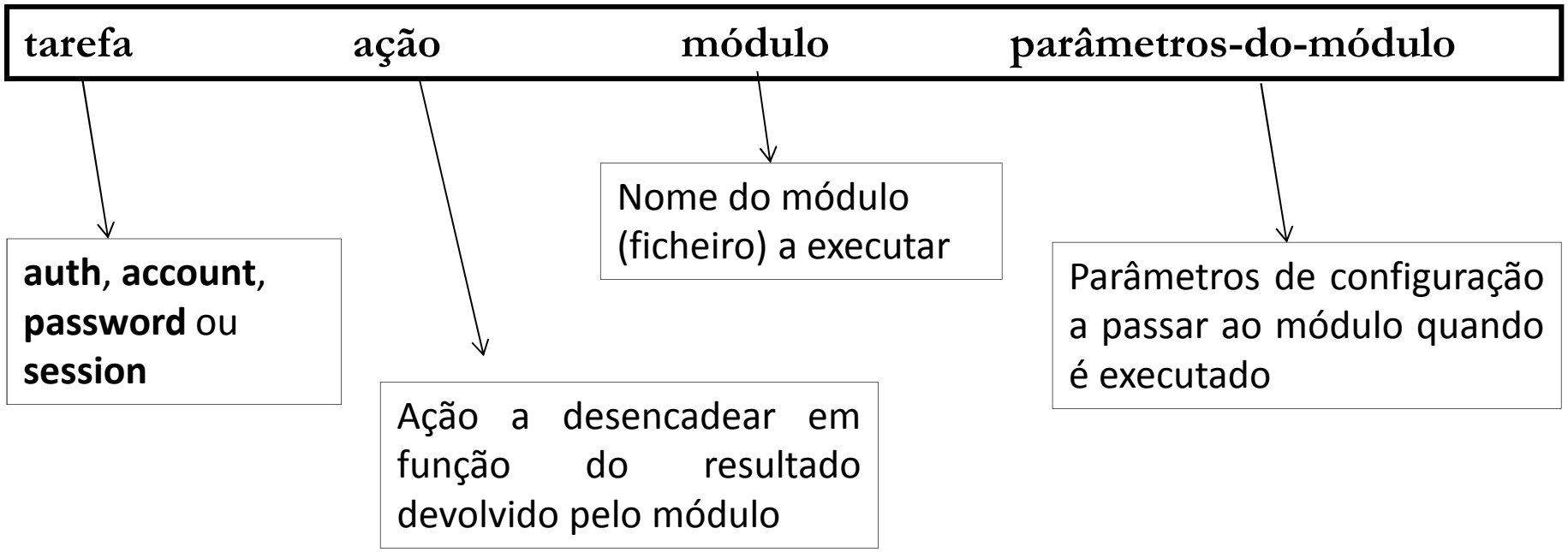
Password (**password**) – tarefa de alteração da *password* do utilizador.

Sessão (**session**) – tarefas de preparação da sessão de trabalho do utilizador que se autenticou, tipicamente inclui a definição do ambiente de trabalho e registo da entrada, pode ser novamente invocada no *logout*, por exemplo para registar a saída do utilizador.

Alguns módulos PAM podem ser usados em vários tipos de tarefa, outros são específicos para algumas apenas.

PAM – configuração

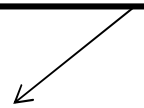
Configurar o sistema PAM consiste em definir a sequência de módulos a utilizar em cada tipo de tarefa, isso é conseguido com ficheiros de texto em que cada linha representa a invocação de um módulo no contexto de uma tarefa, na forma:



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.

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.

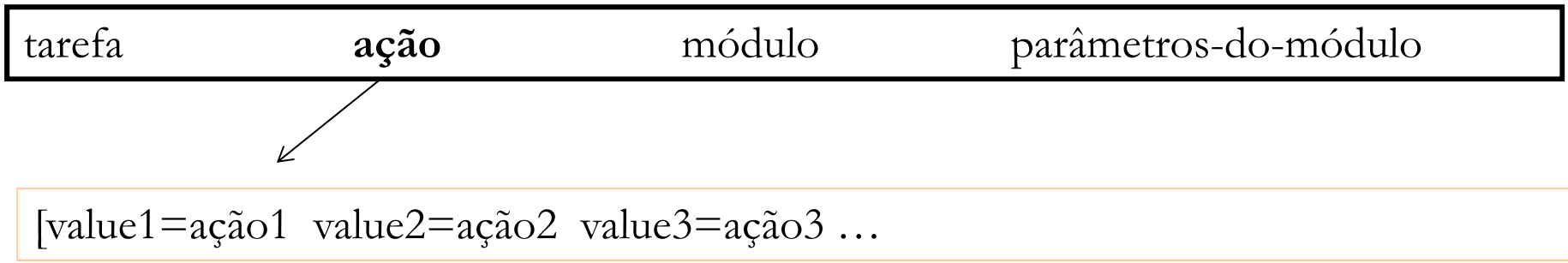
requisite - em caso de sucesso continua o processamento dos módulos seguintes, em caso de falha termina de imediato resultando em falha.

sufficient – em caso de sucesso, 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.

PAM – configuração - ação

Sintaxes mais complexas da ação a realizar em função do resultado devolvido pelo módulo podem ser especificadas:



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, mas 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 seguintes e continua o processamento. (*NN* é um número inteiro superior a zero)

PAM – Alguns módulos comuns

`pam_unix`	Pode ser usado em qualquer das tarefas, implementa as funcionalidades tradicionais do UNIX pré-PAM.
`pam_deny`	Pode ser usado em qualquer das tarefas, devolve sempre falha. Usado para testes ou para finalizar uma sequencia.
`pam_env`	Módulo **auth** e **session** que permite manipular as variáveis de ambiente segundo um ficheiro de configuração, geralmente "/etc/security/pam_env.conf".
`pam_mail`	Módulo **auth** e **account** que faz a verificação de mail no inicio da sessão.
`pam_ldap`	Modulo de interface com repositórios LDAP, suporta as funcionalidades **auth**, **account** e **password**.
`pam_issue`	Módulo **auth** que permite apresentar uma mensagem antes da autenticação do utilizador.
`pam_motd`	Módulo **session** que apresenta uma mensagem após a entrada no sistema (***message of the day***).
`pam_nologin`	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`	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`	Módulo **password** que verifica a solidez da nova "password" que o utilizador está a digitar, além de verificar a no dicionário do sistema, pode também fazer algumas verificações adicionais.

PAM – configuração - ação

Exemplo de aplicação em que o processamento vai devolver falha se não for possível a autenticação nem nos ficheiros locais (**pam_unix.so**) nem nos servidores LDAP (**pam_ldap.so**):

```
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
```

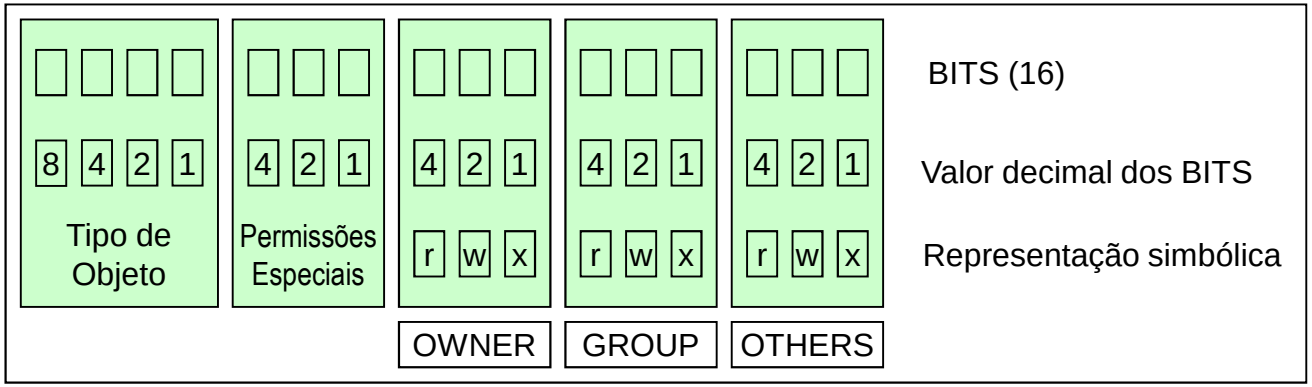
Existem as seguintes equivalências entre as formas de especificar a ação:

```
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]
```

Área de trabalho do utilizador e permissões

A privacidade da área de trabalho do utilizador consegue-se retirando todas as permissões aos outros utilizadores e grupos, ou seja 700.

Determinadas aplicações podem exigir outros tipos de permissão, por exemplo para o "apache" poder divulgar uma página Web numa pasta no interior da HOME, a HOME necessita de ter a permissão "x" para "others".



As permissões podem ser facilmente alteradas com o comando **chmod**, mas no que diz respeito aos novos ficheiros e pastas criados depois disso, as permissões com que são criadas são determinadas pela UMASK. Trata-se de uma máscara de negação de bits de permissão, ou seja indica os bits que devem ser desativados.

Por exemplo a UMASK 0022, que é bastante usada, indica que os bits de valor 2 (permissão de escrita) devem ser desativados para o "grupo" e para "outros".

O valor da UMASK pode ser alterado com o comando **umask**, mas não se mantém de umas sessões para as outras, terá de ser definido no inicio de cada sessão.

Ficheiros de arranque da SHELL

Uma vez que a SHELL é usada como base para o lançamento de aplicações, é o local ideal para configurar o ambiente que vai depois ser herdado por todos os processos lançados. Isto é particularmente válido para a SHELL inicial ou de *login*.

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

A SHELL **/bin/sh** é na atualidade 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/tcsh** 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.

Ficheiro /etc/profile típico

```
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

Cotas de utilizadores e grupos

As cotas servem para controlar e limitar o espaço que cada utilizador ou grupo usa numa determinada partição. Além do espaço ocupado também é possível limitar o número de objetos.

O objetivo das cotas é garantir uma partilha equitativa de um recurso que é limitado, evitando que o abuso de alguns comprometa o trabalho dos restantes.

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

O controlo de cotas pelo núcleo não é realizado em valor absoluto.

Supõe-se que o ponto de partida é um registo de cotas correto.

As cotas de utilizador dizem respeito aos recursos ocupados por cada utilizador, as cotas de grupo referem-se ao somatório dos recursos ocupados por todos os membros de cada grupo.

Cotas – montagem da partição

O primeiro passo para ativar a contabilização de cotas numa dada partição é alterar as opções de montagem no ficheiro **/etc/fstab**, incluindo as opções **usrquota** e/ou **grpquota**.

Os programas de suporte de cotas verificam o **/etc/fstab** para saberem quais as partições onde existe suporte de cotas. Por exemplo durante o arranque do sistema é executado o comando **quotaon –a**.

O comando **quotaon** informa o núcleo de que deve começar a controlar as cotas numa determinada partição. Esta operação pressupõe que a contabilização de cotas nessa partição já existe e está atualizada.

A opção **-a** faz com que o comando ative as cotas em todas as partições referidas no **/etc/fstab** que têm as opções **usrquota** ou **grpquota**.

O comando **quotaoff** desativa o controlo de cotas pelo núcleo numa dada partição.

Cotas – contabilização inicial

O núcleo regista variações de cota e atualiza o registo absoluto em função dessas variações. A validade inicial do registo absoluto não é da responsabilidade do núcleo.

O comando **quotacheck** tem como missão efetuar a contabilização absoluta das cotas, normalmente só é necessário executar uma vez, a menos que uma falha grave danifique o registo existente nos ficheiros **(a)quota.user** e/ou **(a)quota.group** localizados na raiz da partição correspondente.

O comando **quotacheck** deve ser executado com precaução, as cotas devem estar desativadas (**quotaoff**) e não devem ocorrer alterações na partição durante o processo. Por esta razão o comando **quotacheck** tenta "remontar" a partição em modo "leitura-apenas".

Após a execução do comando **quotacheck** os ficheiros **(a)quota.user** e/ou **(a)quota.group** estão reparados e atualizados podendo então ser executado o comando **quotaon** para que o núcleo se encarregue de os manter.

Cotas – definição de valores

O valor da cota máxima têm de ser definido utilizador a utilizador (cotas de utilizador) ou grupo a grupo (cotas de grupo).

Tanto para as cotas de espaço (blocos de 1024 octetos) como de objetos (i-nodes) são definidos dois limites: **soft** e um **hard** mais elevado. O limite **soft** pode ser excedido por um período de tempo máximo conhecido por **grace period**.

Os valores de cota definidos como zero serão considerados sem limite.

Os vários limites de cota podem ser alterados com o comando **setquota** e o comando **edquota**, este último interage através de um editor de texto.

```
-bash-3.00$ /usr/sbin/setquota
setquota: Bad number of arguments.
setquota: Usage:
  setquota [-u|-g] [-F quotaformat] <user|group>
        <block-softlimit> <block-hardlimit> <inode-softlimit> <inode-hardlimit> -a|<filesystem>...
  setquota [-u|-g] [-F quotaformat] <-p protouser|protogroup> <user|group> -a|<filesystem>...
  setquota [-u|-g] [-F quotaformat] -b -a|<filesystem>...
  setquota [-u|-g] [-F quotaformat] -t <blockgrace> <inodegrace> -a|<filesystem>...
  setquota [-u|-g] [-F quotaformat] <user|group> -T <blockgrace> <inodegrace> -a|<filesystem>...
Bugs to: mvw@planets.elm.net, jack@suse.cz
```

Cotas – apresentação do estado

Cada utilizador pode visualizar a sua situação de cotas com o comando **quota**, ao administrador este comando permite visualizar as cotas de qualquer utilizador.

```
qfel% quota -a
Disk quotas for user hjw (uid 1379):

Filesystem   blocks   quota     limit  grace  files  quota limit grace
/netscratch5     0    52428800 5248800  0      204800  204800
/netscratch6     0    52428800 5248800  0      204800  204800
/netscratch7     0    52428800 5248800  0      204800  204800
/netscratch8     8748 52428800 5248800  0      204800  204800
```

O comando **repquota** permite ao administrador obter o estado geral das cotas no sistema de ficheiros.

```
[root@redhat6 /home]# repquota -a
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
root      --    2088       0       0              148     0     0
ftp       --       4       0       0                1     0     0
nobody    --       4       0       0                1     0     0
odagiri   --   10004       0       0               13     0     0
devl      --      60    5000       0               16     0     0

                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
root      --    2096       0       0              150     0     0
odagiri   +-   10004   10000   12000   none       13     0     0
suzuki    --      28     500       0                7     0     0
sato      --      32    1000    1000                9     0     0
[root@redhat6 /home]#
```

Interfaces de rede

Designa-se por **interface de rede** o conjunto de componentes de hardware que permitem a ligação física de um computador a um dado tipo de infraestrutura de rede. Relativamente ao MR-OSI, a interface de rede implementa funcionalidades do nível 1 e nível 2.




Quando o núcleo LINUX *deteta* uma interface de rede associa-lhe um identificador apropriado. Uma vez que podem existir vários dispositivos do mesmo tipo, os identificadores têm um sufixo numérico que começa em zero.

Por exemplo as interfaces do tipo ETHERNET serão identificadas por eth0; eth1; eth2; …, na ordem em que são detetadas.

Para que o núcleo possa detetar uma interface de rede (ou qualquer hardware) necessita que o respetivo suporte (*driver*) esteja incluído no núcleo, ou seja adicionado através de um módulo.

Administração de interfaces de rede – comando *ifconfig*

O comando *ifconfig* é um dos meios diretos de que o administrador dispõe para manusear as interfaces de rede de um sistema LINUX.

```
-bash-3.00$ /sbin/ifconfig --help
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <endereço>[/<tam_prefixo>]]  [del <endereço>[/<tam_prefixo>]]
  [[-]broadcast [<endereço>]]  [[-]pointopoint [<endereço>]]
  [netmask <endereço>]  [dstaddr <endereço>]  [tunnel <endereço>]
  [outfill <NN>] [keepalive <NN>] [hw <HW> <endereço>]  [metric <NN>]
  [mtu <NN>] [[-]trailers]  [[-]arp]  [[-]allmulti]  [multicast]
  [[-]promisc] [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <tipo>]
  [txqueuelen <NN>] [[-]dynamic]  [up|down] ...
```

O campo **<AF>** (*Address Family*) serve para identificar a pilha/família de protocolos o valor por omissão é **inet** que representa o IPv4. Outros valores suportados são por exemplo o **inet6** e o **ipx**. A forma e a aplicação de muitos dos restantes parâmetros depende de **AF**.

Administração de encaminhamento – comando *route*

O comando *route* permite manipular a tabela de encaminhamento (*routing*) do núcleo do sistema operativo. É com base na tabela de encaminhamento que o núcleo sabe como fazer chegar os pacotes ao destino correto.

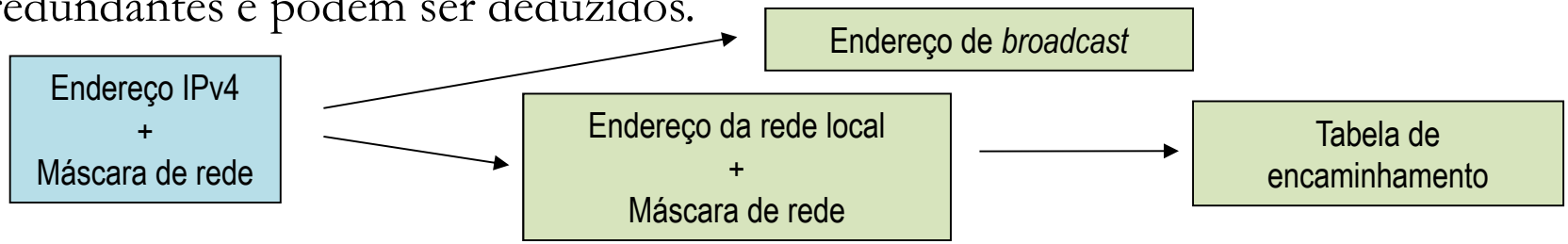
```
/sbin/route [ add | del ] [ -net | -host ]
        [ REDE/IP-DESTINO ] [ netmask MÁSCARA-DE-REDE ]
        [ gw GATEWAY ] [ [dev] INTERFACE ]
```

Usado sem argumentos o comando *route* apresenta a tabela de encaminhamento:

```
[root@server ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.16.2     0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
172.16.16.4     0.0.0.0         255.255.255.255 UH    0      0        0 ppp2
193.136.62.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.62.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0.3
172.18.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth1.7
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth1.6
172.22.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth1
172.23.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth1.5
172.24.0.0      192.168.62.7    255.252.0.0     UG    0      0        0 eth0.3
172.28.0.0      192.168.62.8    255.252.0.0     UG    0      0        0 eth0.3
0.0.0.0         193.136.62.1    0.0.0.0         UG    0      0        0 eth0
```

Configuração IPv4 de uma interface de rede

Para configurar uma interface de rede IPv4 apenas são necessários dois elementos: o endereço IP local e a respetiva mascara de rede. Os outros elementos são redundantes e podem ser deduzidos.



Esta informação não é suficiente para um funcionamento normal de um sistema em rede, mas sob o ponto de vista de informação associada a uma interface de rede é a necessária.

```
-bash-3.00$ /sbin/ifconfig eth0 192.168.111.150 netmask 255.255.255.0
-bash-3.00$ /sbin/route add -net 192.168.111.0 netmask 255.255.255.0 eth0
-bash-3.00$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  Endereço de HW 00:0C:29:FE:E7:F8
          inet end.: 192.168.111.150 Bcast:192.168.111.255  Masc:255.255.255.0
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:243766872 errors:0 dropped:0 overruns:0 frame:0
          TX packets:249142107 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:3379540387 (3.1 GiB)  TX bytes:1843450578 (1.7 GiB)
```

Comando *ip*

O comando **ip** junta as funcionalidades de vários comandos relacionados com a administração de rede, de entre eles o comando *ifconfig* e o comando *route*. Acrescenta ainda várias outras funcionalidades bastante importantes.

```
/sbin/ip [ ... ] [ link | addr | route | rule | neigh | tunnel |
        maddr | mroute | monitor | xfrm ]  ...
```

Usando o comando **ip**, o exemplo anterior seria:

```
# /sbin/ip addr add 192.168.111.150/24 broadcast + dev eth0
# /sbin/ip route add 192.168.111.0/24 eth0
# /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:0C:29:FE:E7:F8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.111.150/24 brd 192.168.111.255 scope global eth0
```

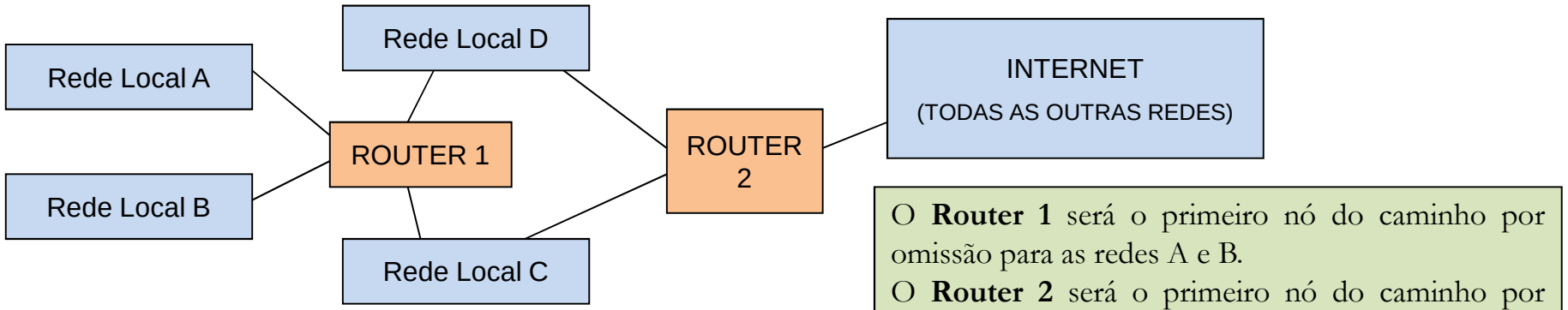
```
# /sbin/ip route show
172.16.16.11 dev ppp9  proto kernel  scope link  src 172.16.16.1
172.16.16.10 dev ppp8  proto kernel  scope link  src 172.16.16.1
193.136.62.0/24 dev eth0  proto kernel  scope link  src 193.136.62.11
192.168.62.0/24 dev eth0.3  proto kernel  scope link  src 192.168.62.11
172.18.0.0/16 dev eth1.7  proto kernel  scope link  src 172.18.0.1
default via 193.136.62.1 dev eth0
```

Caminho por omissão (*default route*)

Só é possível enviar pacotes IPv4 para destinos (redes) que constem na tabela de encaminhamento.

É claro que não é possível manter entradas na tabela de encaminhamento para todas as redes IPv4 existentes a nível global.

Se isso fosse possível rapidamente se iria verificar que com a exceção das redes locais, todas as outras seriam representadas por linhas idênticas na tabela de encaminhamento pois o primeiro salto para chegar a qualquer dessas redes é sempre o mesmo: o *router* local de ligação à *internet*. Esse é o *default router*.



Múltiplos endereços IP na mesma interface de rede

Até ao *kernel* 2.4 o suporte de vários endereços sobre a mesma interface física era conhecido por **ip alias**, para o efeito aos nomes das interfaces era acrescentado um sufixo na forma "**:n**", por exemplo "**eth0:0**", "**eth0:1**", são **alias** da interface "**eth0**", cada um com o seu próprio endereço IP diferente do da interface "**eth0**".

Nos núcleos mais recentes, com o comando "**ip addr add …**" é possível atribuir vários endereços a uma mesma interface. Para manter a compatibilidade com o comando **ifconfig** é prática corrente manter a mesma terminologia usando a opção **label** do comando **ip**.

```
[root@server]# /sbin/ip address add 192.168.199.35/24 brd + dev eth0
[root@server]# /sbin/ip address add 192.168.199.37/24 brd + dev eth0 label eth0:0
[root@server]# /sbin/ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:80:c8:f8:4a:51 brd ff:ff:ff:ff:ff:ff
    inet 192.168.199.35/24 brd 192.168.199.255 scope global eth0
    inet 192.168.199.37/24 brd 192.168.199.255 scope global secondary eth0:0
[root@server]# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:80:C8:F8:4A:51
          inet addr:192.168.199.35  Bcast:192.168.199.255  Mask:255.255.255.0
eth0:0    Link encap:Ethernet  HWaddr 00:80:C8:F8:4A:51
          inet addr:10.10.20.10  Bcast:10.10.20.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Suporte de VLAN

Ao contrário de um *alias* que é definido no nível de rede (Ex.: IPv4), uma VLAN é definida no nível de ligação lógica, por isso existe uma separação real tal como se tratassem de meios físicos separados.

Para se conseguir esta separação a norma IEEE 802.1q define a forma de etiquetar as tramas de nível 2 para que não se misturem entre si. O identificador de VLAN é um número de 0 a 4095, os equipamentos de rede devem tratar estas tramas etiquetadas (*labeled* ou *tagged*) de tal forma que não se misturem identificadores de valores diferentes.

O comando **vconfig** é usado para criar uma VLAN numa interface física existente:

```
[root@server]# /sbin/vconfig add eth0 9
```

VLANID

Após a execução do comando acima torna-se válida a interface **eth0.9**, todas as tramas emitidas através de ela transportam a etiqueta IEEE 802.1q com o valor "9". De igual modo, nesta interface apenas serão recebidas tramas com etiqueta IEEE 802.1q contendo o valor "9".

Para remover uma VLAN usa-se o mesmo comando:

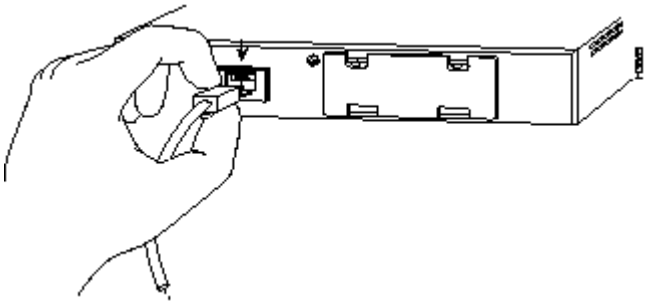
```
# /sbin/vconfig rem eth0.9
```

Configuração automática da rede

Não sendo muito aconselhável em servidores, a configuração automática é fundamental nos postos de trabalho.

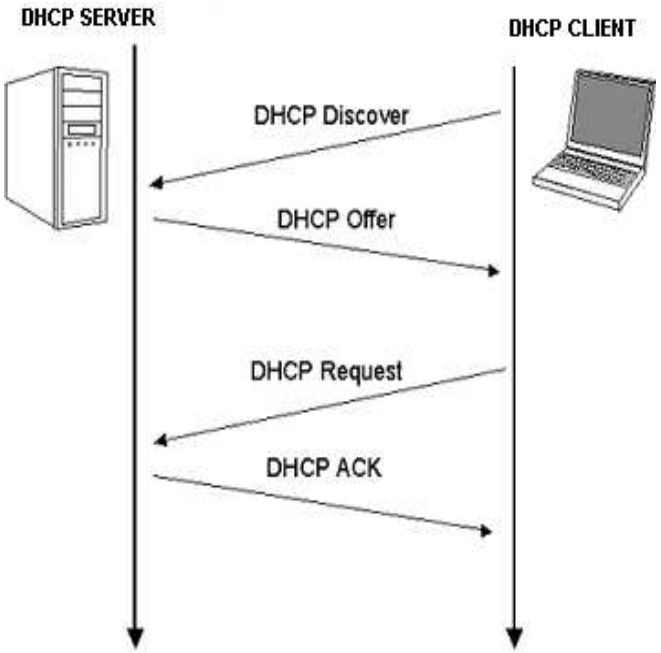
A configuração automática da rede IPv4 faz-se com recurso a servidores DHCP existentes na rede, em LINUX o comando/serviço *dhclient* permite configurar por este meio todas ou algumas das interfaces de rede. Depois de obter o endereço para uma interface, o serviço mantém-se em execução para controlar o aluguer (*lease*) do endereço, procedendo aos refrescamentos necessários junto do servidor DHCP.

O serviço **ifplugd** é particularmente útil em postos de trabalho com várias interfaces de rede, este serviço monitoriza constantemente o estado das ligações físicas.



Quando um cabo é ligado encarrega-se de ativar a respetiva interface, quando um cabo é desligado remove a respetiva interface de rede no sistema operativo.

Servidor **DHCP**



O servidor DHCP (*dhcpd*) recebe pedidos de clientes enviados sob a forma de *datagramas* UDP que inicialmente são emitidos para o endereço *255.255.255.255*.

A missão do servidor é observar o endereço físico (MAC) de origem responder com todos os dados necessários para o cliente poder configurar a rede.

Embora o servidor DHCP possa ser configurado com endereços MAC estáticos, torna-se particularmente útil o facto de poder gerir autonomamente uma gama de endereços IP. Para que a atribuição dinâmica de endereços seja viável tem de ser limitada no tempo, daqui surge o conceito de aluguer (*lease*) que não existia no BOOTP onde as atribuições eram "vitalícias".

Além do endereço IP, o servidor envia muito mais informação ao cliente, tais como nome do domínio e endereços dos servidores de nomes, essa informação é pré-configurada no servidor, normalmente no ficheiro **/etc/dhcpd.conf**.

Servidor DHCP em Linux – exemplo de configuração

```
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 10.4.0.0 netmask 255.255.0.0 {
option subnet-mask 255.255.0.0;
option routers 10.4.0.1;
option broadcast-address 10.4.255.255;
option domain-name-servers 192.168.62.15;
option netbios-name-servers 192.168.62.15;
option log-servers 192.168.62.37;
range 10.4.10.0 10.4.30.255;
}
```

Gama de endereços a atribuir dinamicamente aos clientes

Servidor DHCP em Linux – exemplo de configuração

Exemplo com endereços IP estáticos atribuídos a determinados nós de acordo com o seu endereço *ethernet*.

```
group {
option subnet-mask 255.255.0.0;
option routers 10.4.0.1;
option broadcast-address 10.4.255.255;
option domain-name-servers 192.168.62.15;
option netbios-name-servers 192.168.62.15;
option netbios-dd-server 192.168.62.15;
option log-servers 192.168.62.37;
host hp_super { fixed-address 10.4.2.45; hardware ethernet 00:11:85:d2:df:5f; }
host lex1 { fixed-address 10.4.2.46; hardware ethernet 00:04:00:d6:f8:a9; }
host vproj01 { fixed-address 10.4.2.63; hardware ethernet a4:ee:57:ea:fa:db; }
}
```

Internet Daemon – INETD

Grande parte dos serviços de um sistema LINUX são assegurados por processos servidores autónomos, esta metodologia assegura a mais elevada disponibilidade e performance.

Além dos serviços de elevado grau de utilização e com necessidade de elevada performance, acumulam-se muitas vezes uma grande variedade de serviços menores que apesar da reduzida utilização têm de estar disponíveis em permanência.

O *Internet Daemon* foi desenvolvido para substituir vários processos servidores durante a fase em que estão à espera de ser contactados por clientes. Quando o contacto ocorre o *Internet Daemon* invoca o programa apropriado para prestar os serviço.

O *Internet Daemon* é configurado através do ficheiro **/etc/inetd.conf**, para cada serviço é necessário especificar o protocolo a usar (TCP ou UDP), o número de porto (ou identificador do serviço definido em **/etc/services**) e o programa que implementa o serviço.

INETD – configuração (/etc/inetd.conf)

O ficheiro **/etc/inetd.conf** define os serviços prestados pelo INETD, 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**, neste 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 – comportamento. ***wait*** obriga o servidor a finalizar o atendimento de um cliente antes de atender o seguinte, é o comportamento típico para serviços UDP. ***nowait*** permite atender vários clientes em simultâneo, é o comportamento típico para serviços TCP.

username – *login name* com que o serviço vai ser executado, para serviços que não necessitam de acessos especiais ao sistema o mais seguro é o utilizador *nobody*.

INETD – configuração (/etc/inetd.conf)

service-name	socket-type	protocol	wait/nowait	username	exec-file	args

exec-file – caminho desde a raiz até ao ficheiro executável que vai atender o cliente

args – argumentos a fornecer ao executável, incluindo o arg[0].

Exemplo:

```
ftp       stream  tcp   nowait  root    /usr/bin/ftpd      ftpd -l
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 sem recorrer a nenhum programa externo.

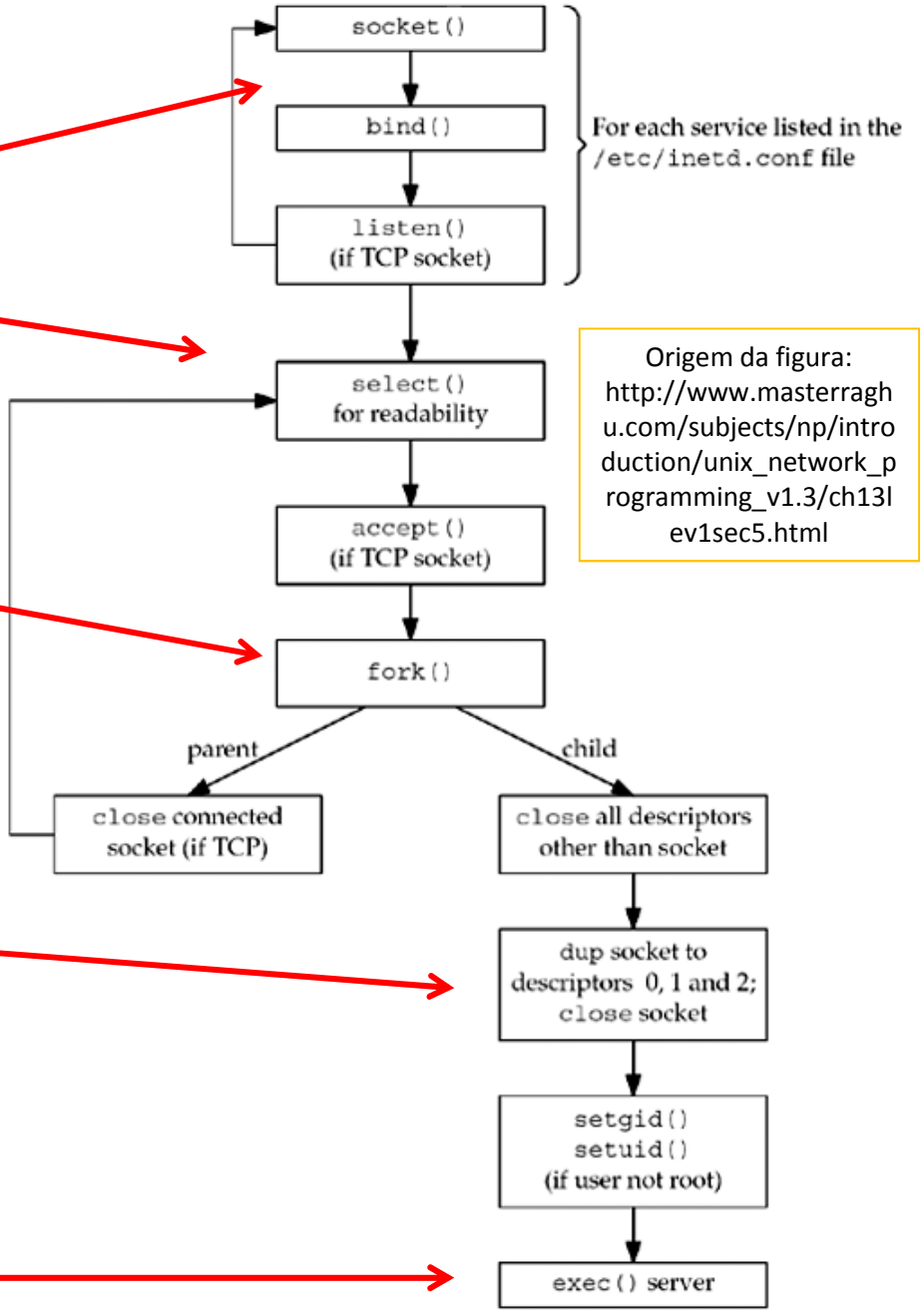
INETD - funcionamento

Depois de inicializar os vários *sockets* o INETD recorre à *system-call select* para detetar atividade eu qualquer deles.

Quando um cliente contacta o 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*.



Origem da figura: http://www.masterraghu.com/subjects/np/introduction/unix_network_programming_v1.3/ch13lev1sec5.html

Resolução de nomes DNS - cliente

A resolução de nomes de máquinas (obtenção do respetivo endereço IP) pode começar por recorrer ao ficheiro **/etc/hosts**, mas na maioria dos casos acaba por necessitar do serviço DNS.

A resolução de nomes por DNS consiste no envio do nome qualificado a um servidor de nomes. Os parâmetros para configurar o sistema DNS cliente são apenas os endereços dos servidores de nomes.

Uma vez que podem surgir nomes não qualificados para serem resolvidos, será necessário adicionar-lhes um nome de domínio antes de os enviar ao servidor. Para esse efeito pode ser definido o domínio local e uma lista de domínios de pesquisa.

Em LINUX o ficheiro de configuração do cliente DNS é o **/etc/resolv.conf**.

```
[root@server ~]# cat /etc/resolv.conf
domain dei.isep.ipp.pt
search isep.ipp.pt
nameserver 192.168.62.15
nameserver 193.136.62.3
```

De referir que nas distribuições atuais o ficheiro **/etc/resolv.conf** é tipicamente gerado de forma dinâmica pelos serviços de rede, nesse caso a sua edição direta é inútil, a gestão deste parâmetros deve ser realizada no serviço de rede.

Filtragem – comando *iptables*

O comando *iptables* permite administrar a filtragem do tráfego de rede realizada pelo núcleo. Existem 4 tabelas: **filter**; **nat**; **mangle** e **raw**, destas as mais usadas são as 2 primeiras, por omissão a tabela usada é **filter**. Algumas ações mais importantes são:

`iptables -F`	`(FLUSH) apagar todas as regras da cadeia`
`iptables -A`	`(ADD/APPEND)adicionar uma regra ao fim da cadeia`
`iptables -D`	`(DELETE) eliminar uma regra`
`iptables -I`	`(INSERT) inserir uma regra`
`iptables -R`	`(REPLACE) substituir uma regra`
`iptables -L`	`(LIST) listar regras existentes na cadeia`
`iptables -P`	`(POLICY) definir política da cadeia`

Cada tabela têm determinadas cadeias de regras pré definidas, mas outras podem ser criadas. Para as duas principais tabelas:

filter : INPUT; FORWARD e OUTPUT.

nat : PREROUTING; OUTPUT; POSTROUTING

Em cada cadeia pode ser definida uma sequência de regras com uma numeração implícita com início no valor 1. A numeração é importante para algumas operações como a eliminação de regras ou inserção de regras.

Filtragem – tabelas e cadeias

A tabela *filter* é usada para implementar a filtragem de pacotes, as 3 cadeias pré definidas aplicam-se a diferentes tipos de tráfego de acordo com a respetiva origem e destino:

INPUT – tráfego destinado ao servidor (endereço de destino igual ao do servidor)

OUTPUT – tráfego originado no próprio servidor

FORWARD – tráfego em transito, quando o servidor opera como *router*.

A tabela *nat* é usada para aplicar traduções de endereços (NAT), as 3 cadeias definem o tipo de tráfego e o momento é que a tradução é aplicada:

PREROUTING – traduções a aplicar no momento em que o pacote chega ao servidor, independentemente de ser ou não destinado ao servidor.

OUTPUT – traduções a aplicar na emissão de um pacote emitido pelo próprio servidor.

POSTROUTING – traduções a aplicar na emissão de um pacote que foi encaminhado, ou seja não tem origem no servidor.

IPTABLES – ação por omissão

Para cada cadeia de regras da tabela *filter* é necessário definir o comportamento por omissão, ou seja, o que se vai passar se **para um dado pacote a cadeia for processada até ao final sem que nenhuma das regras de verifique**.

De entre as quatro ações possíveis (ACCEPT; DROP; QUEUE e RETURN) o que deve ser adotado como ação por omissão (política da cadeia) é DROP.

```
iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -P FORWARD DROP
```

Isto significa que se um dado pacote não é explicitamente permitido por uma das regras da cadeia será recusado (DROP).

Esta metodologia é a mais indicada sob o ponto de vista de segurança, mas também poderá revelar-se a mais trabalhosa.

IPTABLES - regras

Existe uma grande variedade de critérios que se podem usar numa regra, algumas mais comuns são:

	Correspondência com pacotes em que:
[!] –p protocolo	o protocolo é o indicado (tcp, udp, icmp,…)
[!] -s endereço[/mascara]	o endereço de origem é da rede indicada
[!] –d endereço[/mascara]	o endereço de destino é da rede indicada
[!] –i interface	a interface de entrada é a indicada (eth0, …)
[!] –o interface	a interface de saída é a indicada
[!] –-dport porto1[:porto2]	o número de porto de destino está nesta gama
[!] --sport porto1[:porto2]	o número de porto de origem está nesta gama
	o símbolo ! significa a negação da regra

Para que um pacote corresponda à regra tem de corresponder simultaneamente a todos os critérios definidos nessa regra. No contexto de cada regra é ainda definido o que fazer aos pacotes que correspondam, normalmente:

-j AÇÃO	executar uma ação imediata, tipicamente ACCEPT ou DROP
-g CADEIA	passar o pacote para uma outra cadeia de processamento

Exemplo:

```
iptables –F FORWARD
iptables –P FORWARD DROP
iptables –A FORWARD –i eth0 –s 195.20.10.23/32 –o eth2 –p tcp –j ACCEPT
iptables –A FORWARD –i eth0 –d 193.136.0.0/16 –p tcp –dport 8080:8081 –j ACCEPT
```

Scripts

Designa-se por *script* um pequeno programa interpretado contido num simples ficheiro de texto diretamente executável.

Em outros sistemas não Unix a parte final do nome de um ficheiro (extensão) é usada para indicar o seu conteúdo, no caso particular de um *script* a extensão poderá indicar a linguagem usada e por isso qual o interpretador a usar.

Em Linux/Unix usa-se o conteúdo dos primeiros bytes do ficheiro para o classificar (*magic number*), independentemente do nome/extensão do ficheiro.

No caso de ficheiros de texto contendo programas interpretados (*scripts*) a primeira linha identifica o interpretador que deve ser usado, além disso sendo um *script*, deverá ter a permissão de execução (x) ativa.

Shebang - #!

O *shebang* (de *hash+bang*) é a sequência "#!" colocada na primeira coluna da primeira linha dos *scripts* Unix. Os caracteres "#!", ou seja 0x2321 são o *magic number* que identifica o ficheiro como sendo um *script*.

A parte restante da linha do *shebang* serve para identificar o programa interpretador que deve ser usado para executar o *script* contido no ficheiro.

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

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

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

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

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

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

Como nas linguagens interpretadas as linhas começadas por "#" são consideradas comentários, o *shebang* não interfere com o *script* propriamente dito.

"Shell Scripts"

O objetivo dos programas classificados como *shell* é fornecer um meio de interação com o sistema através de comandos, no entanto esses comandos podem agrupados em sequência num *script* constituindo assim um verdadeiro programa interpretado. Para se tirar todo o proveito dos *shell scripts* existem comandos de controlo tais como a decisão e os ciclos de repetição.

Os *shell scripts* são uma ferramenta importante para o administrador pois permitem automatizar muitas tarefas e estão sempre acessíveis, basta um editor de texto.

Muitos preferem usar a CSHELL devido à semelhança da sua sintaxe com a linguagem de programação C.

Uma vez que nos sistemas LINUX atuais tanto a *shell* tradicional (**sh/bash**) como a C *shell* (**csh/tcsh**) estão sempre disponíveis a opção por uma ou outra é uma escolha quase pessoal. Normalmente a opção recai sobre a *shell* que usa diariamente em modo interativo.

Outras linguagens como por exemplo Perl e Pyton possuem potencialidades muito maiores, com bibliotecas bastante extensas, mas exigem a familiarização do administrador.

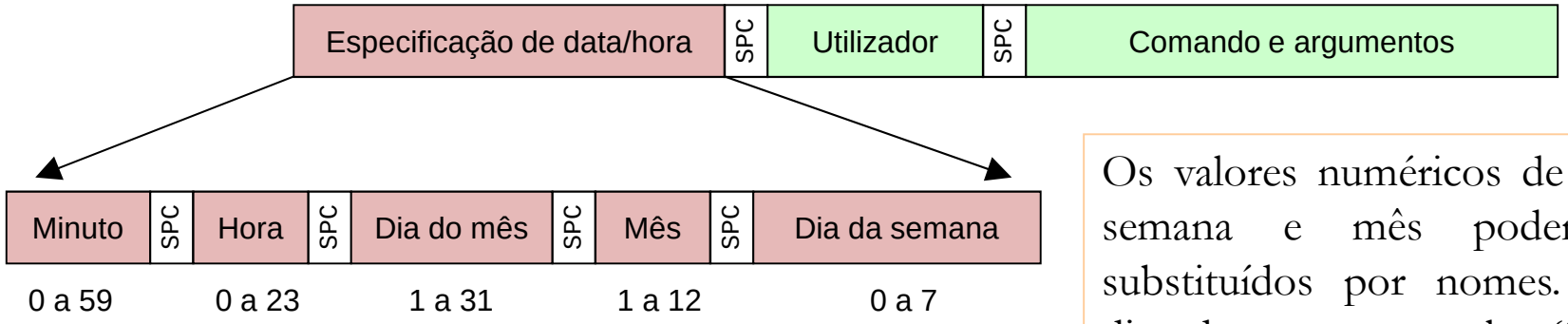
Shell Scripts – exemplo simples em BASH

```
#!/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
```

Execução programada - serviço CRON

A possibilidade de programar a execução de comandos, por exemplo *scripts* tem enormes vantagens para o administrador. Permite definir a execução periódica de tarefas, por exemplo em horários apropriados a operações de manutenção.

Nos sistema LINUX o serviço CRON (*crond*) é o mais usado para atingir estes objetivos, o ficheiro **/etc/crontab** é o principal ficheiro de configuração. As linhas de comando do ficheiro de configuração apresentam a seguinte forma:



Os valores de data/hora podem ser especificados em intervalos ou por enumeração. O símbolo "*" representa o intervalo de todos os valores possíveis para esse campo.

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".

Serviço CRON – exemplos "/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 gamas de valores podem ser associadas a um valor de "passo" na gama de valores, no exemplo acima "*/5" ("0-59/5") nos minutos significa percorrer os minutos de 5 em 5, ou seja é equivalente a "0,5,10,15,20,25,30,35,40,45,50,55".

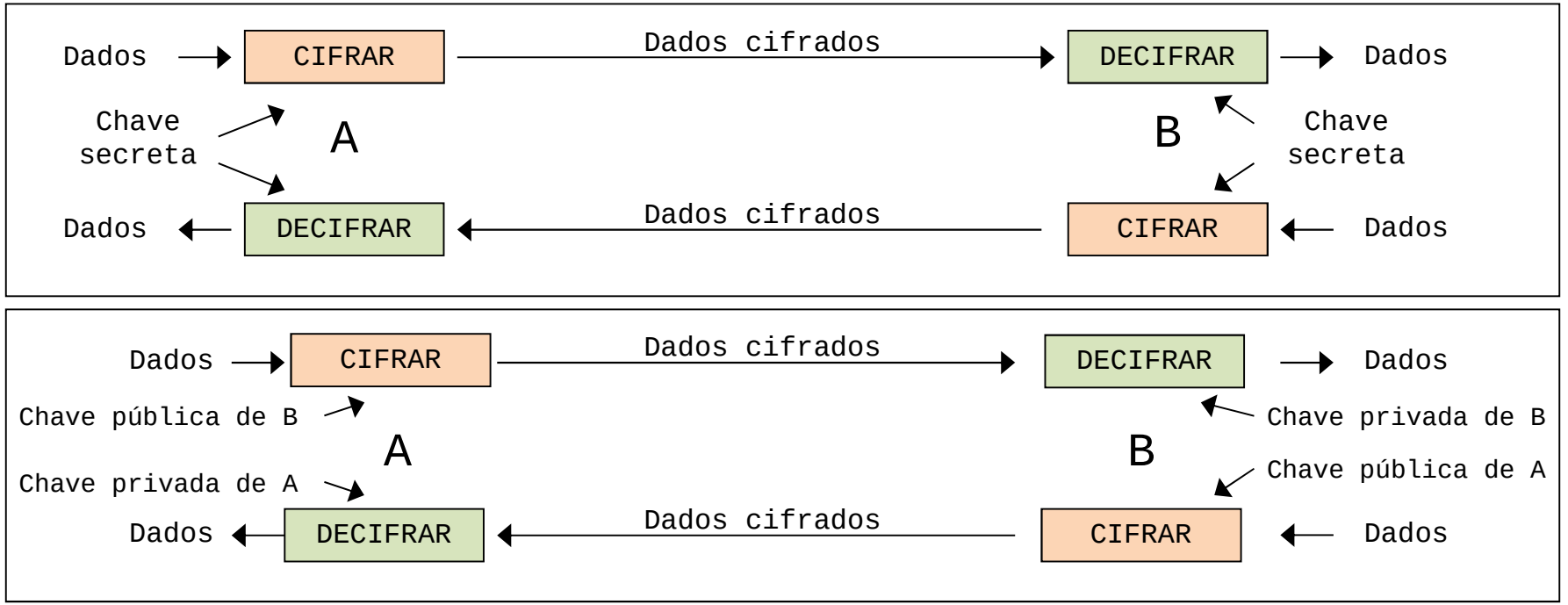
O ficheiro **/etc/crontab** apenas pode ser usado pelo administrador, mas o serviço *crond* também está acessível aos outros utilizadores através de ficheiros de configuração normalmente guardados em **/var/spool/cron/**, estes devem ser manipulados indiretamente através do comando **crontab –e**.

O formato é idêntico ao do **/etc/crontab**, mas o nome do utilizador (6º campo) é omitido pois está implícito.

O acesso dos utilizadores ao serviço CRON pode ser controlado pelo administrador através dos ficheiros **/etc/cron.allow** e **/etc/cron.deny**.

Segurança em rede

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 é recorrer a algoritmos de cifragem. A abordagem tradicional (simétrica) exige que os dois intervenientes possuam uma chave secreta (pré-partilhada). A criptografia de chave pública, mais recente, usa chaves diferentes para cifrar e decifrar e com isso torna a distribuição de chaves muito mais simples.



Autenticação em rede

Antes de haver preocupações com a confidencialidade dos dados (recorrendo à criptografia), temos de nos assegurar que estamos a transacionar dados com a entidade correta. Essa é a missão da autenticação.

Geralmente a autenticação está associada à cifragem

Criptografia simétrica: a autenticação está assegurada à partida pois apenas duas entidades possuem a chave secreta. Note-se que este facto apenas delega o problema da autenticação para a fase anterior de distribuição de chaves. Tipicamente a distribuição de chaves secretas recorre a um outro segredo pré-partilhado, como por exemplo a *password* do utilizador.

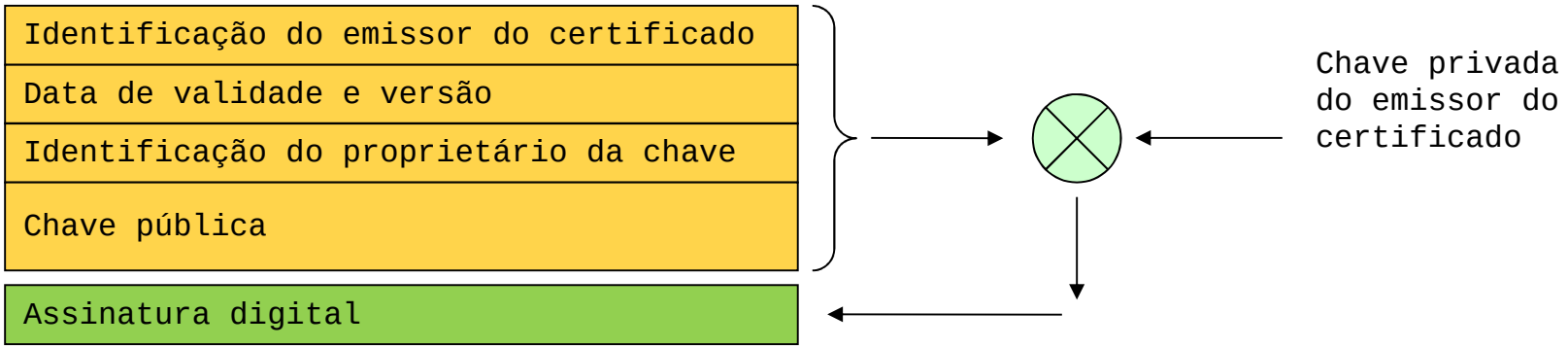
Criptografia de chave pública: a autenticação não está assegurada porque as chaves usadas para cifrar são públicas. Se a chave é pública, qualquer um a pode usar.

No entanto, uma vez que a chave privada é conhecida por apenas uma entidade, esse facto pode ser usado para garantir a autenticação. Basta garantir a autenticidade das chaves públicas pois apenas o detentor da respetiva chave privada poderá decifrar.

Certificados de chave pública

A criptografia de chave pública simplificou muito a distribuição de chaves, mas esta simplificação eliminou a autenticação que estava implícita na criptografia simétrica. Contudo garantindo a autenticidade das chaves públicas o problema fica resolvido.

Um certificado de chave pública contém a identificação do proprietário da chave pública (***Subject***), a identificação do emissor do certificado (***Issuer***), a chave pública e assinatura digital do conjunto realizada pelo emissor do certificado.



Os clientes aceitam o certificado com base na confiança sobre o emissor do certificado, data de validade do mesmo e identificação do proprietário da chave.

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

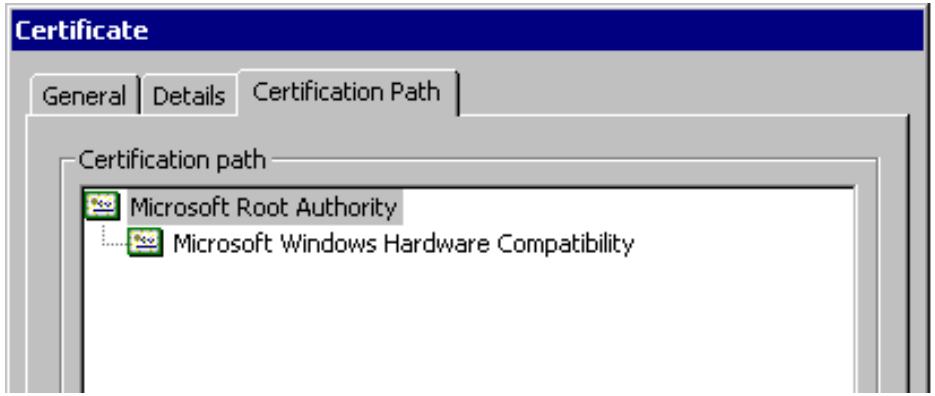
```
[root@server ~]# openssl x509 -in /etc/cert/redecert.pem -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=PT, ST=PORTO, L=PORTO, O=DEI.ISEP, OU=DEI,
    CN=DEICA/emailAddress=dei@dei.isep.ipp.pt
        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
```

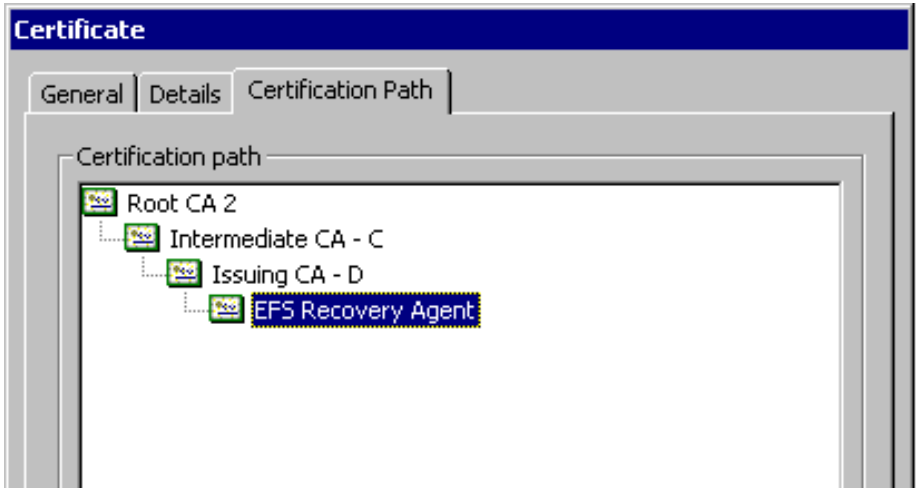
Cadeia ou caminho de certificação

Um certificado de chave pública só é aceite na medida em que a entidade certificadora (CA – *Certification Authority*) que o emitiu é considerada de confiança.

Desde logo uma **entidade certificadora** será considerada de confiança se o seu certificado de chave pública foi emitido por uma outra entidade certificadora considerada de confiança. Forma-se assim uma cadeia de certificação, mas tem de haver uma origem, na origem está uma ROOT CA.

Exemplos de cadeia ou caminhos de certificação:





"ROOT CA" e certificados auto assinados

Uma vez que o caminho de certificação tem de ter um final (sendo normalmente essa entidade designada ROOT CA).

A questão que se coloca é: **quem emite o certificado da ROOT CA ?**

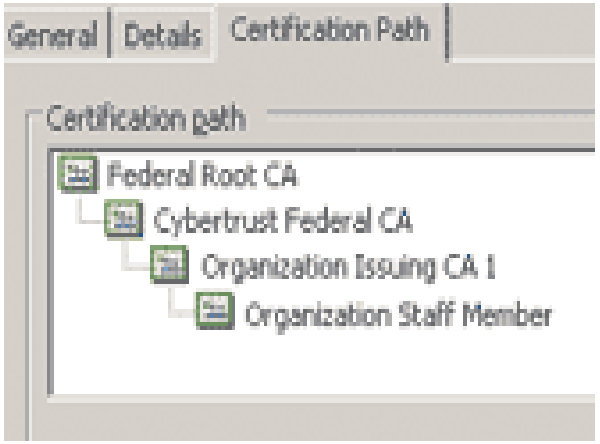
Resposta: **a própria ROOT CA** , ou seja é um certificado auto assinado.

Um certificado auto assinado "**não certifica nada**" porque estamos a confiar numa entidade para comprovar a sua própria idoneidade.

Este problema resolve-se através da instalação administrativa dos certificados auto assinados das ROOT CA que dessa forma passam a ser consideradas autoridades certificadores fidedignas por opção do administrador.

Sob o ponto de vista de um sistema operativo ou aplicação que recebe certificados no contexto de comunicações seguras (por exemplo um *browser* a usar HTTPS), a confiança num certificado existe quando seguindo a cadeia de certificação encontra um certificado (eventualmente de uma ROOT CA) que está instalado localmente e classificado com entidade certificadora fidedigna.

Verificação de certificados



Nos sistemas operativos e aplicações são incluídos um conjunto de certificados de CA fidedignas a nível mundial. Os serviços de rede públicos que usam certificados (por exemplo com TLS) possuem normalmente certificados pertencentes às cadeias de certificação destas CA.

Quando uma aplicação de rede (tipicamente um cliente) recebe um certificado (normalmente do servidor) realiza algumas verificações:

- 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 de rede, 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.

Revogação de certificados

Teoricamente em certificado seria válido até ao fim da sua data de validade, contudo sob o ponto de vista de segurança isso não é o ideal.

O problema ocorre quando a **chave privada** de uma entidade A é comprometida (capturada por um atacante X), nestas circunstâncias temos um problema grave:

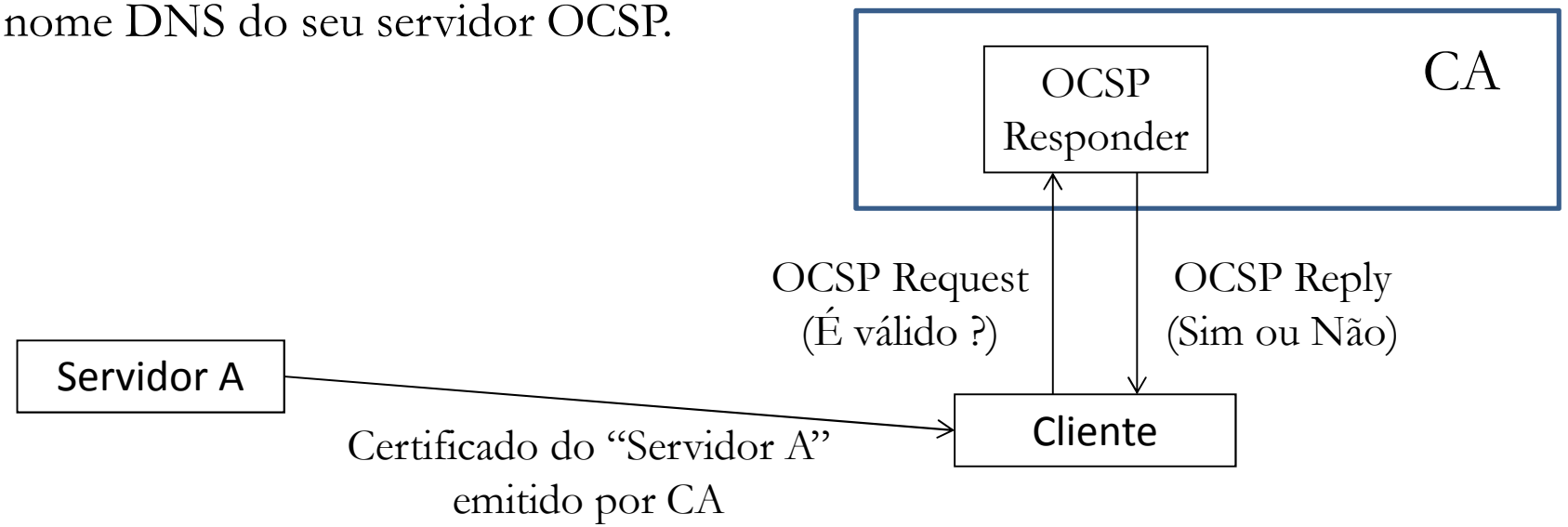
Quando uma outra entidade B quiser enviar informação confidencial para A vai usar a chave pública que se encontra no certificado de A (que ainda está dentro da validade) uma vez que X possui a chave privada da A vai conseguir ler a informação.

Para se conseguir resolver este tipo de problema as CA têm de ter a possibilidade de revogar um certificado que tenham emitido apesar de estar ainda dentro da data de validade. **Isso só pode ser conseguido se quem vai utilizar o certificado contactar diretamente a CA emissora.**

Esta passa a constituir uma **4ª verificação** a adicionar às 3 anteriormente referidas.

OCSP (*Online Certificate Status Protocol*)

O contacto da CA emissora para verificar a validade de um certificado pode ser realizado de varias formas, um dos protocolos mais divulgados é o OCSP. Cada CA deverá disponibilizar um servidor OCSP, normalmente designado *OCSP responder* adicionalmente deve definir um atributo nos certificados que emite onde coloca o nome DNS do seu servidor OCSP.



De certa forma este tipo de verificação viola os princípios que levaram ao desenvolvimento dos certificados – o facto de não ser necessário contactar a CA para verificar que o certificado é válido. Nem todos os clientes realizam este tipo de validação sobre os certificados que recebem.

Comando *openssl* (LINUX)

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 uma chave e emitir um pedido de certificação da mesma. 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 seria gerado o ficheiro **pri.key** contendo uma chave privada RSA de 1024 bits e o ficheiro **pub.crt** contendo o respetivo certificado auto assinado.

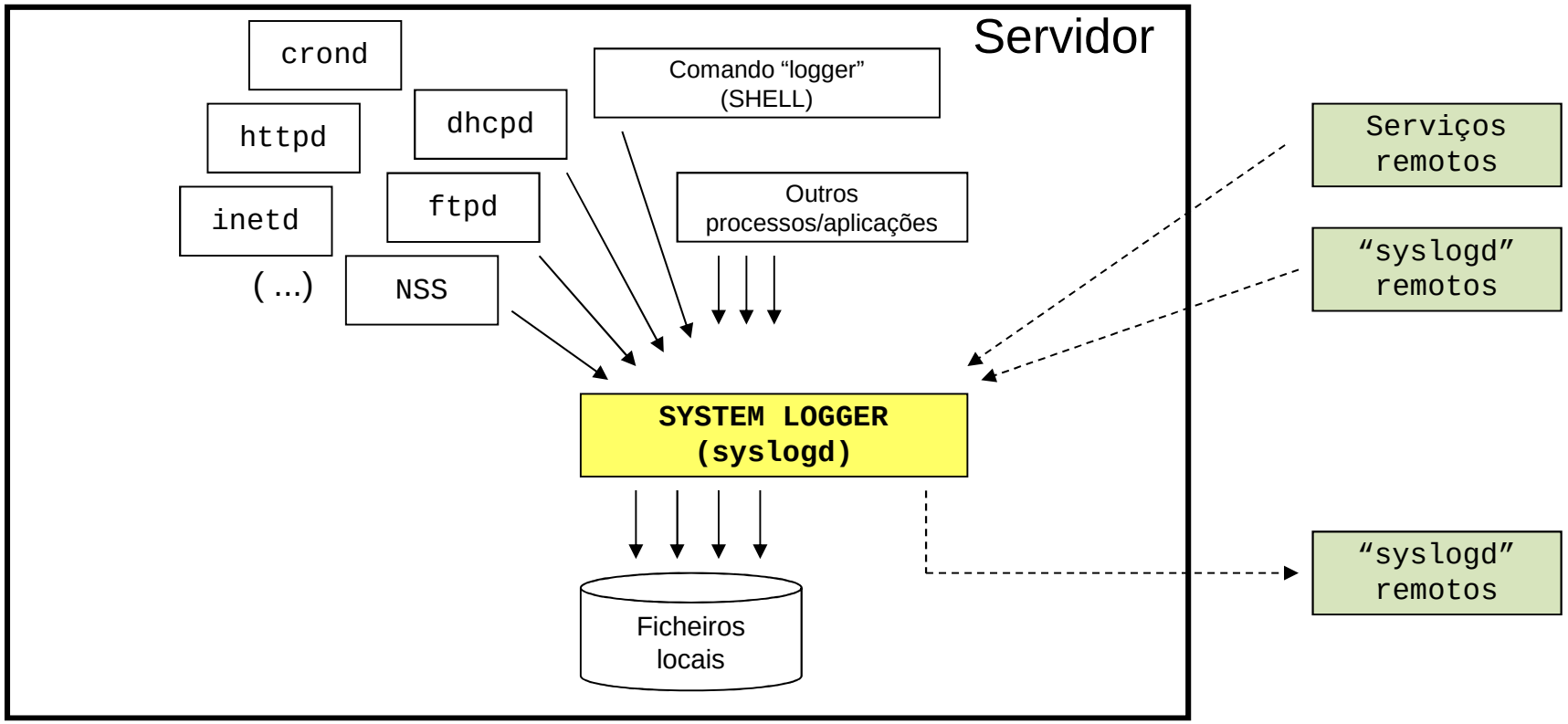
"**openssl x509 …**" – processamento de certificados, por exemplo é usado pela entidade certificadora para produzir o certificado a partir de um pedido de certificado.

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.

Registo de atividades – *System Logger* (*syslog*)

Embora cada componente/serviço do sistema possa criar registos de atividade próprios, o ideal é que exista um serviço central único para gerir esses registos.

O serviço *syslog* recebe mensagens dos clientes (normalmente uma linha de texto que representa um evento) e de acordo com a configuração vai encaminhar essa mensagem para vários destinos possíveis.



System Logger – Classificação das mensagens

Quando o *syslogd* recebe uma mensagem recebe também dois elementos que permitem a sua classificação rápida:

Serviço de origem (*facility*) que pode assumir os seguintes valores: *auth*, *authpriv*, *cron*, *daemon*, *kern*, *lpr*, *mail*, *mark*, *news*, *security* (o mesmo que *auth*), *syslog*, *user*, *uucp* e *local0* até *local7*.

Gravidade ou Prioridade (*severity*, *priority* ou *level*) que pode ter os seguintes valores, por ordem crescente de gravidade: *debug*, *info*, *notice*, *warning*, *warn* (igual ao anterior), *err*, *error* (igual ao anterior), *crit*, *alert*, *emerg*, *panic* (igual ao anterior).

Estes dois elementos podem ser agrupados na forma "{*facility*}.{*severity*}" e é com base neste conjunto que o *syslog* determina o que fazer com cada mensagem.

O ficheiro de configuração do *syslog* é habitualmente o **/etc/syslog.conf**. Neste ficheiro determina-se, em função do valor **{*facility*}.{*severity*}**, o que fazer à mensagem. Algumas ações possíveis são: acrescentar a um ficheiro local; enviar para um servidor *syslog* remoto; enviar diretamente para o terminal dos utilizadores ativos; executar um programa externo e passar-lhe a mensagem através do *stdin*.

syslog – configuração

A configuração do serviço *syslog* resume-se a especificar padrões para o par **{facility}.{severity}** e associar a cada uma da regras uma ação que definem o que fazer com a mensagens. Cada linha do ficheiro é constituída por dois campos separados por um ou mais espaços: uma mascara **{facility}.{severity}** e uma **ação**.

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
```

O nível de gravidade (*severity*) no padrão faz correspondência com mensagens desse nível de gravidade ou superior (>=). No caso de existirem várias regras (linhas) aplicáveis a uma dada mensagem, todas elas serão aplicadas sucessivamente, o processamento de uma mensagem não termina na primeira correspondência.

syslog – registo centralizado

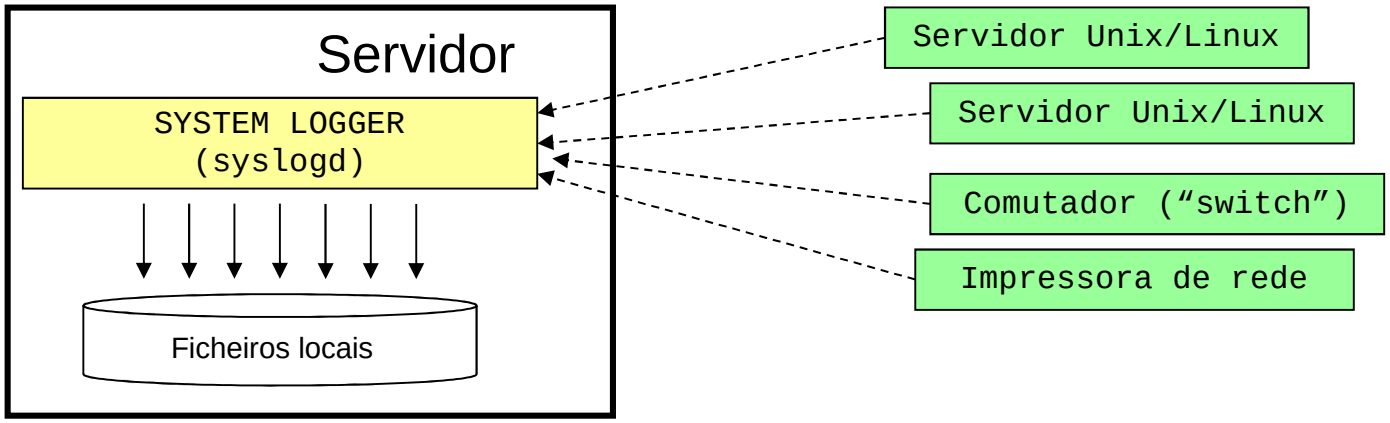
O serviço *syslog* permite a receção de mensagens remotas através da rede, usando pacotes UDP dirigidos ao porto 514. Normalmente esta possibilidade está desativada por motivos de segurança, a forma de ativar a escuta no porto 514 depende da versão do serviço, nas versões mais tradicionais bastava fornecer o argumento "-r" no arranque. Muitos dispositivos de rede tais como impressoras, comutadores, *routers* e *access-points* podem ser configurados para enviar mensagens *syslog* para um dado endereço/máquina. Essa informação de configuração pode mesmo ser fornecida na resposta do servidor DHCP ("option log-servers …").

O administrador de um *cluster* de servidores Linux/Unix pode por esta via centralizar os registos numa única máquina, colocando em todas as restantes um ficheiro **/etc/syslog.conf** semelhante a:

```
*.*                    @servidor.domínio
```

Ou enviando apenas uma parte das mensagens, realizando assim uma filtragem local em cada máquina e enviando para o registo central apenas as mais importantes.

syslogd – consulta de registos



Na maioria dos casos as mensagens que o *syslog* recebe acabam por ser acrescentadas a um ficheiro de texto identificado no **/etc/syslog.conf**. Na maioria dos sistema Linux adota-se o diretório **/var/log/** para colocação destes ficheiros.

Tratando-se de ficheiros de texto, a sua consulta é direta, recorrendo-se a um vasto conjunto de comandos disponíveis para esse efeito, alguns dos mais importantes são:

"cat"; "less"; "tail"; "more"; "grep"; "cut"; "sed"

Registo de entradas e saídas - *wtmp*

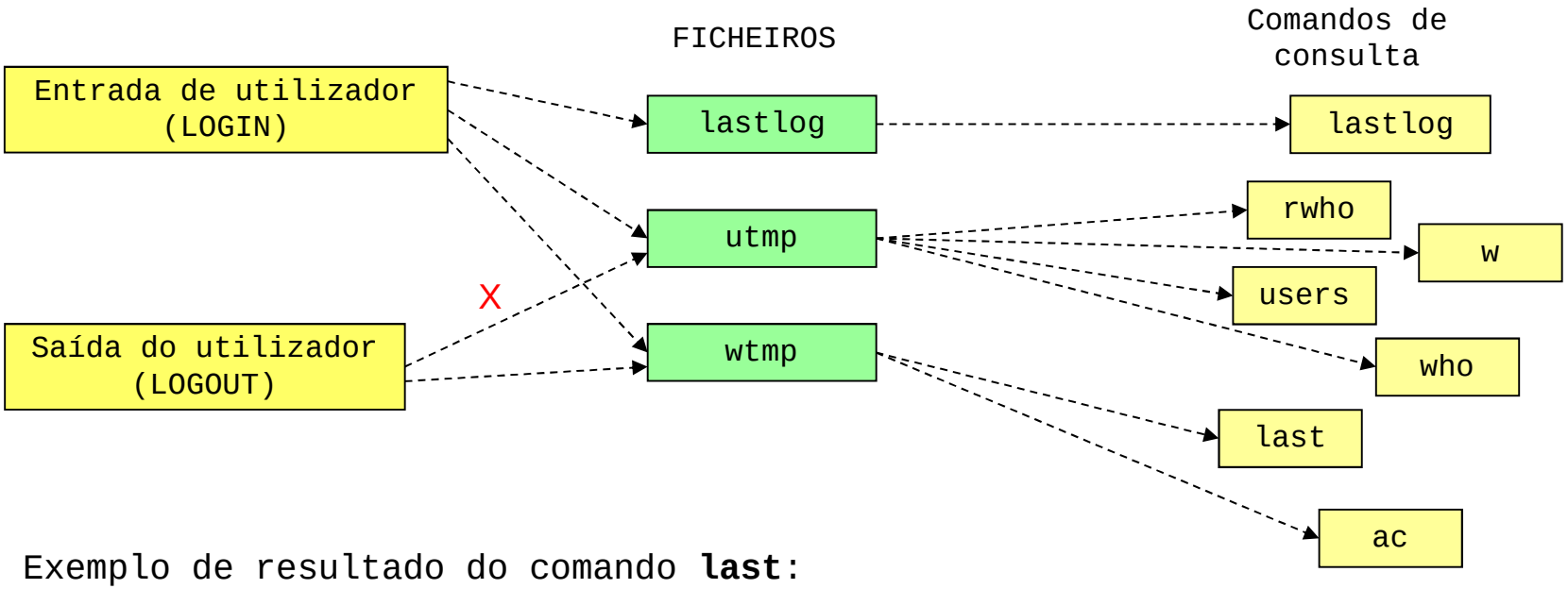
Dada a sua importância, existe um sistema independente para registar as entradas e saídas dos utilizadores no sistema. Cabe aos programas que controlam essas entradas e saídas recorrer a este registo para o manter atualizado, ele é constituído por três ficheiros:

•**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.

Consulta do registo de entradas e saídas



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
```

Servidores 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 dos *schema* (normalmente sob a forma de ficheiros de texto). Existe uma grande variedade de classes disponíveis para os mais diversos tipos de aplicação.

LDAP – classes e objetos

A definição de classes recorre à herança, ou seja pode definir-se uma superclasse a partir de uma ou várias classes existentes 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:

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

DN – *Distingushed* Name

Cada objeto na DIT é identificado de forma única através do DN. O DN identifica localmente o objeto através do valor de um ou vários do seus atributos (RDN – *Relative Distinguished Name*) juntamente com o DN do objeto pai (na DIT). Os vários elementos do DN são separados por uma virgula.

O topo da DIT (raiz) tem ele próprio um DN definido pelo administrador, deve representar o nome DNS do domínio através dos componentes do nome DNS guardados em atributos do tipo **dc** (*domain component*), na forma:

dc=meu-dominio,dc=dominio-acima,…,dc=dominio-topo

Por exemplo se "**dc=dei,dc=isep,dc=ipp,dc=pt"** é o DN da raiz

, o objeto com DN "**cn=user1,ou=utilizadores,dc=dei,dc=isep,dc=ipp,dc=pt"**

fica perfeitamente identificado, tem o RDN "**cn=user1**" e está localizado no ramo "**ou=utilizadores,dc=dei,dc=isep,dc=ipp,dc=pt**", ou seja tem como pai o objeto "**ou=utilizadores,dc=dei,dc=isep,dc=ipp,dc=pt**".

Servidor *OpenLDAP*

http://www.openldap.org

Um dos servidores LDAP mais usado em Linux é o *OpenLDAP*, o seu ficheiro principal de configuração é o **slapd.conf**, alguns dos aspetos mais importantes da configuração são:

	Exemplos:
Definição dos *schema* a utilizar.	include /etc/ldap/schema/core.schema include /etc/ldap/schema/cosine.schema include /etc/ldap/schema/nis.schema include /etc/ldap/schema/inetorgperson.schema
Definição da base de dados onde os dados vão ser efetivamente armazenados	backend bdb
Definição do DN da raiz da DIT e do administrador. O *hash* pode ser gerado com comando *slappasswd*.	suffix "dc=teste,dc=ipp,dc=pt" updatedn "cn=admin,dc=teste,dc=ipp,dc=pt“ rootdn "cn=admin,dc=teste,dc=ipp,dc=pt" rootpw {SSHA}R+DxrTU6YZFu9H/fB8Vpj
Atributos a indexar para facilitar a pesquisa.	index objectClass eq index uidNumber eq

Servidor *OpenLDAP*

Uma vez configurados os elementos base, o servidor pode entrar em funcionamento e as restantes tarefas de administração já podem ser realizadas através do protocolo LDAP. Para esse efeito existe um conjunto de comandos disponíveis em Linux:

ldapadd – acrescentar um objeto

ldapdelete – eliminar um objeto

ldapmodify – acrescentar um objeto, editar um objeto ou eliminar um objeto

ldapmodrdn – para alterar o DN de um objeto

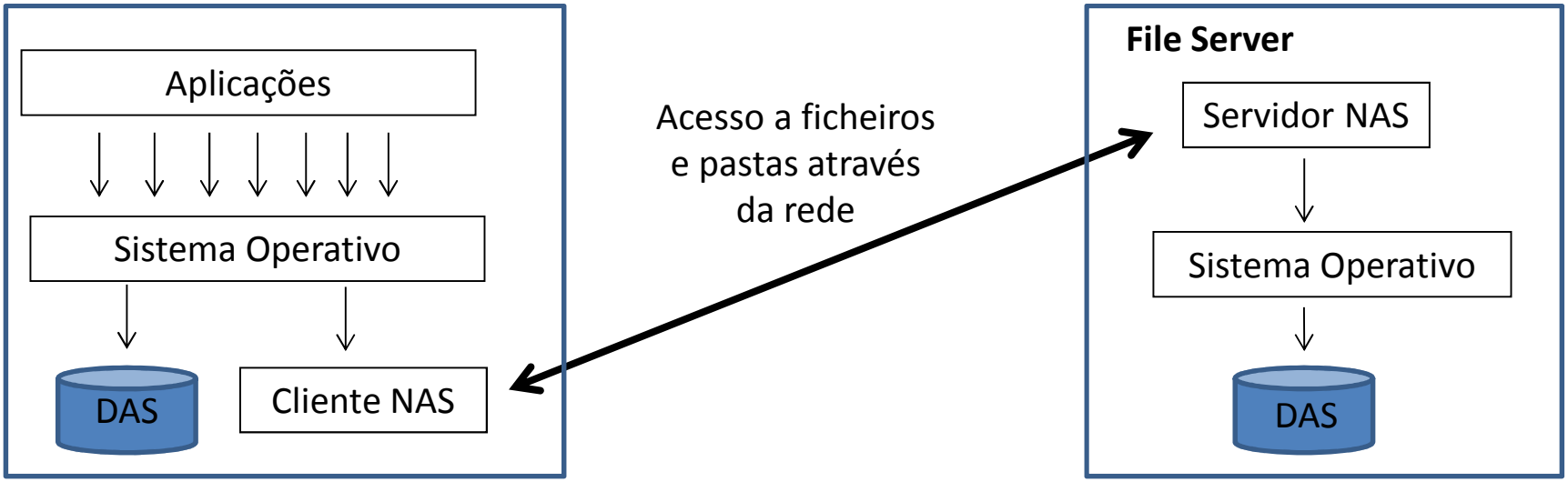
Estes comandos aceitam como argumento ficheiros em formato LDIF (*LDAP Data Interchange Format*) com instruções das ações a realizar, por exemplo:

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

Um ficheiro com este conteúdo 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"

Network Attached Storage (NAS)

Os sistemas de ficheiros em rede, permitem partilhar de forma transparente através da rede diretórios (pastas). Para esse efeito são usados protocolos específicos segundo o modelo cliente-servidor, o servidor muitas vezes designado *file server* responde aos pedidos de acesso remotos facultando acesso aos seus ficheiros locais. O cliente está integrado no sistema operativo de tal forma que não exista diferença aparente entre aceder a ficheiros locais ou a ficheiros remotos.



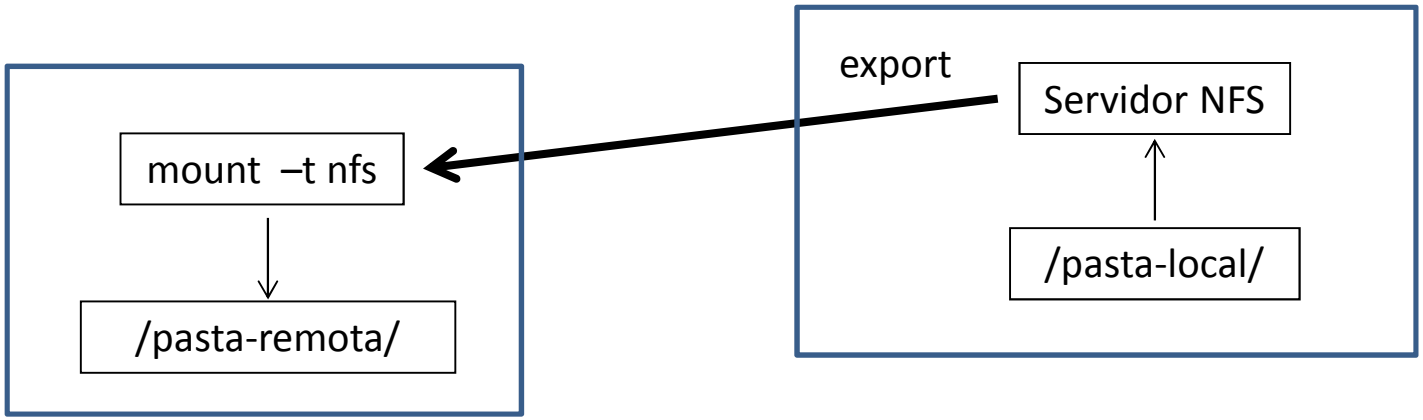
Este tipo de utilização designa-se NAS, por oposição a DAS (*Direct Attached Storage*) em que o sistema de ficheiros reside num disco local.

Sistemas de ficheiros de rede - NFS

Existem muitos protocolos de partilha de ficheiros (NAS). Do "mundo" Windows destaca-se o **SMB** ou **CIFS**, designações alternativas para o mesmo conjunto de protocolos, respetivamente *Server Message Block* e *Common Internet File System*.

No "mundo" Unix, o mais divulgado é o **NFS** (*Network File System*), atualmente a maioria dos sistemas operativos suporta qualquer um destes sistemas de ficheiros.

O NFS foi desenvolvido no contexto dos sistemas operativos Unix e adapta-se particularmente bem à filosofia de montagem administrativa de sistemas de ficheiros. O **NFS Server** normalmente está integrado no *kernel* Linux e **exporta** diretórios locais. Os clientes NFS realizam a montagem de diretórios remotos que ficam assim acessíveis aos utilizadores no sistema de ficheiros local.



NFS – autenticação por endereço de rede

O protocolo NFS tradicional realiza a autenticação com base apenas no endereço de origem do cliente, isto é atualmente válido para o NFSv2 e NFSv3. Atualmente está disponível o NFSv4 que permite a autenticação de utilizadores individuais através de GSS-API com *Kerberos* 5.

No NFSv2 e NFSv3 o servidor é configurado para permitir aos nós com determinados endereços o acesso sem qualquer autenticação adicional, isto pressupõe uma total confiança nos administradores desses nós e uma total confiança na rede.

O ficheiro **/etc/exports** define quais os diretórios locais a exportar e para que endereços de rede a exportação é permitida, adicionalmente podem ser definidas opções de exportação, como por exemplo *ro* (*read-only*). Exemplo:

```
/users/home          190.130.60.0/255.255.255.0(ro,async,no_subtree_check)
/users/backups       vsrv1(ro)    vsrv2(ro)
/usr/local/dump      190.130.60.10(rw)        190.130.60.123(rw)
```

NFS – autenticação por endereço de rede

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

- **Tem de haver confiança total no administrador do nó cliente.** Devido à utilização de números de porto reservados, apenas o administrador do nó cliente pode realizar a montagem do diretório exportado. Depois de realizar a montagem o administrador controla através das permissões que utilizadores têm acesso ao diretório montado.
- **Tem de haver confiança no funcionamento da rede ao nível de endereços.** 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, a DMZ (*demilitarized zone*).
- **Tem de haver confiança no funcionamento da rede ao nível da resolução de nomes.** No caso de se usarem nomes de máquinas em lugar de endereços no /etc/exports 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.

SAN (*Storage Area Network*)

A designação SAN refere-se a um tipo de rede dedicada exclusivamente a interligar dispositivos de armazenamento com os equipamentos que utilizam esses dispositivos de armazenamento, são redes de dimensão muito reduzida e elevada performance.

Existem tecnologias de rede desenvolvidas especificamente para este efeito, é o caso do *Fibre Channel*. Atualmente o *iSCSI* apresenta-se como uma alternativa interessante sob o ponto de vista de custo pelo facto de poder operar sobre qualquer rede IP e não exigir *hardware* específico como acontece com o *Fibre Channel*.

O funcionamento das SAN difere substancialmente do das NAS. Uma NAS permite a **partilha de sistemas de ficheiros** através da rede, uma SAN permite a **partilha de discos lógicos** através da rede.

Apesar de aparentemente semelhantes os conceitos são diferentes:

NAS – o servidor disponibiliza acesso a um sistema de ficheiros local. Esse sistema de ficheiros é gerido pelo servidor, por exemplo é o servidor que formata o sistema de ficheiros antes de o poder disponibilizar.

SAN – o servidor disponibiliza um disco local ou uma partição, o cliente recebe-o como sendo um disco lógico e vai usar esse disco lógico como se fosse um disco local, formatando-o com o sistema de ficheiros que bem entender.

iSCSI (*Internet Small Computer System Interface*)

O *iSCSI* opera diretamente sobre a pilha TCP/IP, isso permite-lhe usar as mais diversas plataformas de *hardware* de rede, tipicamente utiliza-se *ethernet* a 1 ou 10 Gbps como base.

No *iSCSI* usa-se uma terminologia específica para identificar os dois componentes principais do sistema:

target - é o servidor *iSCSI*, ou seja a máquina de disponibiliza os discos lógicos aos clientes.

initiator – é o cliente *iSCSI*, ou seja a máquina que vai utilizar os discos lógicos disponibilizados pelo *target*.

A maioria dos sistemas operativos disponibilizam *software* capaz de exercer as funções de *target* e *initiator iSCSI*. Para performances mais elevadas existem equipamentos de *hardware* dedicados.

Existem atualmente no mercado diversos equipamentos vulgarmente designados NAS que têm a capacidade de operar em vários modos, podendo funcionar como servidores de discos lógicos (SAN) ou como servidores de ficheiros (NAS) suportando ainda vários protocolos em cada um dos modos.

IQN – *iSCSI Qualified Name*

Tanto os *initiators* como os *targets* são identificados por nomes únicos no iSCSI, o formato mais usado é o IQN:

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. **{nome}** é um nome arbitrário que garante que o conjunto é único, por exemplo o nome do nó (no caso de um *target* habitualmente também identifica o disco lógico).

Exemplo: **iqn.2000-08.pt.ipp.isep.dei:server1**

Os identificadores usados pelos *target* podem ser descobertos através da função *discovery*, por exemplo em Linux:

```
root@server1:~# iscsiadm -m discovery -t st -p 192.168.0.101
192.168.0.101:3260,1 iqn.2001-04.com.example:storage.lun1
root@server1:~#
```

O acesso ao *target* pode ser controlado por *username*/*password* sendo usado um algoritmo do tipo CHAP. Depois de efetuado o *login* os discos lógicos (LUN) disponibilizados pelo *target* passarão a estar disponíveis com se fossem discos locais.

Servidores DNS - bind

O ***bind*** é o servidor DNS mais divulgado em ambiente Linux. O ficheiro principal de configuração é o **named.conf**, habitualmente **/etc/bind/named.conf** entre outras declarações, neste ficheiro são declaradas as zonas, ou seja domínios DNS geridos, por exemplo:

```
zone “meu.dominio.pt" {
        type master;
        file "/etc/bind/zones/meu.dominio.pt";
};
```

No exemplo a declaração de zona é do tipo ***master***, ou seja trata-se do servidor principal que detém a base de dados da zona, por oposição ao tipo ***slave*** que seria um servidor secundário que se mantem atualizado através da consulta do *master*. Quer num caso que no outro seriam servidores ***authoritative*** para o domínio em questão, ou seja são servidores que têm na sua posse uma cópia de todos os registos do domínio.

No caso acima, tratando-se do *master* da zona obtém os registos de um ficheiro local, neste caso lendo o ficheiro **/etc/bind/zones/meu.dominio.pt**.

BIND – ficheiros de zona

Cada ficheiro de zona contém os registos DNS correspondentes ao domínio, é organizado em linhas com o seguinte formato geral:

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

nome-do-registo – nome DNS do registos, é sempre implicitamente qualificado, se for usado um nome simples será automaticamente acrescentado o nome do domínio correspondente à zona. Pode ser omitido, nesse caso 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 em **$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.

BIND – ficheiro de zona – registo SOA

O registo SOA (*Start Of Authority*) é normalmente o primeiro registo do ficheiro de zona, contém os seguintes elementos:

nome-do-nameserver email-do-administrador { serial refresh retry expiry nx }

nome-do-nameserver – nome DNS do servidor master

email-do-administrador – endereço de correio eletrónico do administrador da zona com o simbolo @ substituido por um ponto.

serial – número de séria 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 em zero).

refresh – número de segundos entre 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*.

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

BIND – ficheiro de zona – registos

Alguns dos registos mais usados são:

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

Os registos **PTR** são usados na resolução inversa (obter nomes DNS a partir de endereços IPv4 ou IPv6), normalmente num ficheiro de zona separada, o domínio **in-addr.arpa.** é usado para registar endereços IPv4 e o domínio **ip6.arpa.** é usado para registar endereços IPv6.

BIND – ficheiro de zona – exemplo

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

Cópias de arquivo e segurança

Em vários contextos há a necessidade de copiar para sistemas de armazenamento externo os dados residentes nos servidores (sistema de ficheiros).

Arquivo – quando existe um volume significativo de informação que deixou de ser necessária para o funcionamento atual do sistema, mas tem de ser preservada, essa informação pode ser copiada para dispositivos externos e posteriormente removida do sistema libertando espaço.

Segurança – qualquer sistema está sujeito a perdas de dados, por erro dos utilizadores, por falha do *hardware/software* ou por ações maliciosas. Cabe ao administrador manter um calendário de cópias de segurança adequado para que estas estejam suficientemente atualizadas para resolver as situações de perdas de dados que venham a ocorrer no sistema.

Sob o ponto de vista de segurança os sistemas redundantes podem ser considerados uma alternativa a esta metodologia. Não são mais do que o levar deste conceito ao extremo em que as atualizações das cópias são realizadas de forma totalmente sincronizada com as alterações que ocorrem no original.

Tipos de cópias de segurança

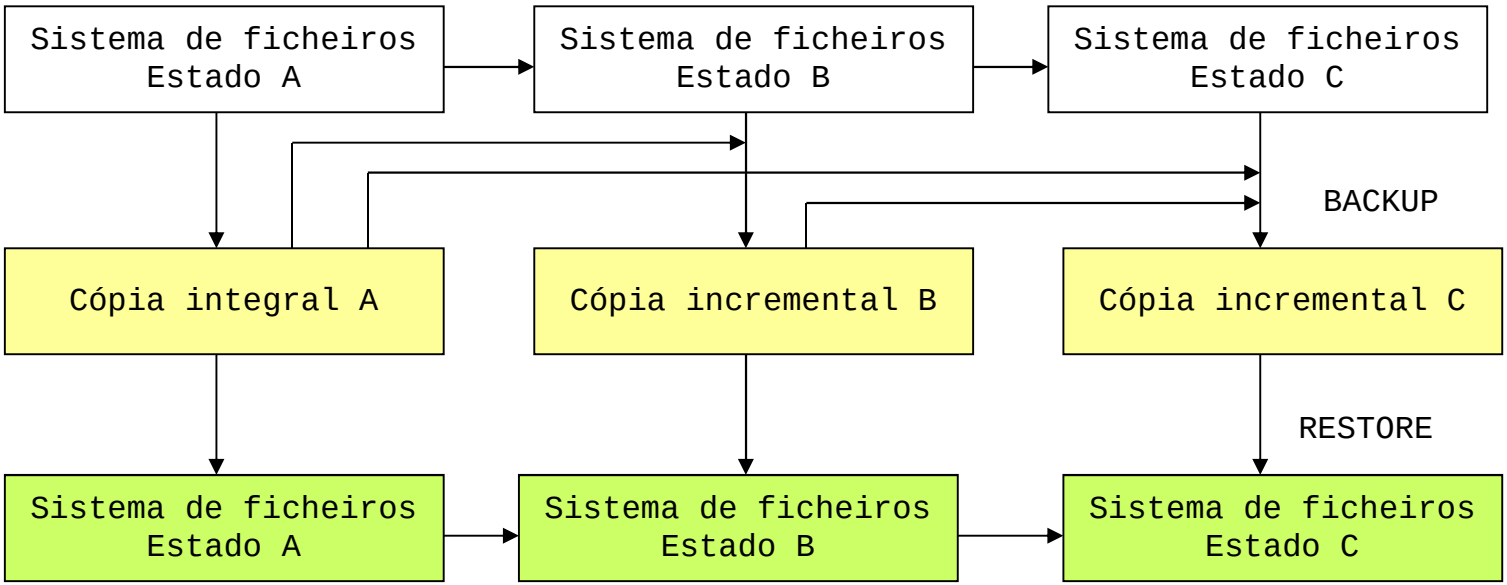
Cópias integrais de discos ou partições – uma cópia deste tipo permite recuperar de forma expedita um sistema totalmente destruído (recuperação usando novo hardware), consiste na cópia integral, do disco ou da partição sector a sector, como tal é independente dos tipos de sistemas de ficheiros que residem no disco, tudo, incluindo a tabela de partições é copiado. Tem contudo alguns inconvenientes importantes: exige que o disco não esteja a ser usado quando se realiza o *backup* e exige que o disco para o qual se realiza a recuperação (*restore*) seja exatamente igual ao disco original. Em Unix o comando **dd** permite realizar cópias deste tipo.

Cópias integrais de sistemas de ficheiros – este tipo de cópia percorre todas as entradas do sistema de ficheiros de uma partição e copia-as, bem como todas as propriedades de cada objeto. É aconselhável que o sistema de ficheiros não esteja a ser usado ou pelo menos esteja a ser pouco usado no momento da cópia. Em Linux os comandos **dump** e **restore** permitem realizar este tipo de cópias i-node a i-node, por esta razão o sistema de ficheiros deve estar "desmontado", estes comandos não trabalham objeto a objeto. O comando ***tar*** também permite realizar este tipo de cópias, mas trabalha diretamente com objetos (ficheiros, diretórios, etc).

Cópias incrementais

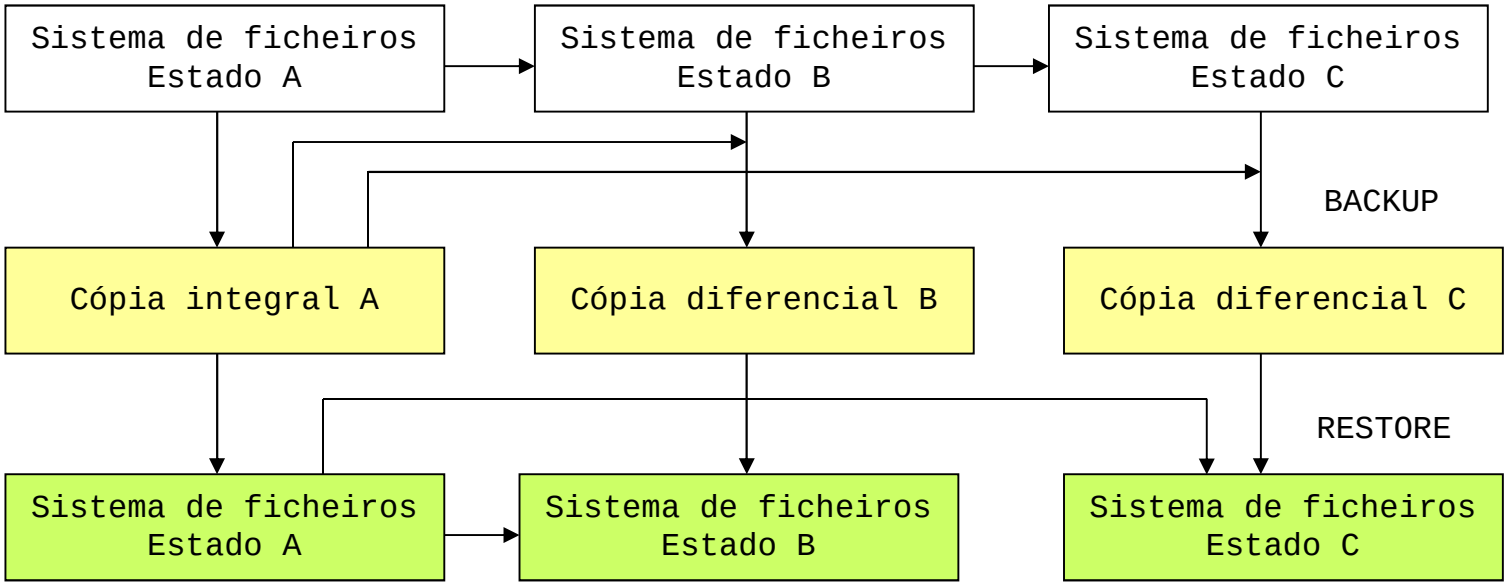
As cópias integrais (*full backup*) de sistemas de ficheiros são operações demoradas (dependendo do tamanho do sistema de ficheiros e do tipo de suporte para a cópia), depois de criada uma cópia integral, é possível recorrer a cópias incrementais.

A cópia incremental consiste em registar as diferenças entre o estado atual do sistema de ficheiros e o estado que está registado na cópia integral ou incremental anterior. O ponto de partida é sempre uma cópia integral.



Cópias diferenciais

A cópia diferencial é semelhante à incremental, no entanto registam-se as diferenças relativamente à cópia integral, ignorando cópias diferenciais anteriores



Ao contrário do que acontece com as cópias incrementais, para realizar o restauro basta a cópia integral e a última cópia diferencial.

Nas cópias incrementais o restauro exige a cópia integral e todas as cópias incrementais até à mais recente, como consequência o restauro é mais moroso.

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.

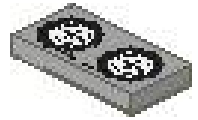
A cópia parcial pode ser realizada em Unix com recurso ao comando **tar**, para esse efeito é necessário localizar todos os ficheiros e diretórios associados ao subsistema ou serviço em causa.

Exemplo de produção de uma cópia de segurança comprimida para uma instalação particular do servidor Apache:

```
tar -cvzf apache2.tar.gz /etc/apache2/* /var/www/cgi-bin /var/www/htdocs \
/etc/sysconfig/apache2 /etc/init.d/apache2
```

Muitos serviços dispõe de comandos próprios para realizar cópias de salvaguarda, especialmente no caso das bases de dados. Por exemplo 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.

Sistemas de suporte para as cópias



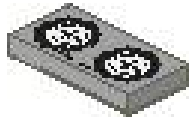
Tradicionalmente as cópias de arquivo e segurança usam como suporte unidades de fita de acesso sequencial que se caracterizam por um baixo custo para grandes capacidades. Tratando-se de meios amovíveis tem a vantagem de proporcionar uma capacidade ilimitada. Note-se que a necessidade de troca manual das unidades de fita é um inconveniente importante, a menos é claro que se disponha de um sistema robotizado para esse efeito. A operação com unidades de fita tem algumas particularidades, mas tanto os comandos **dump**/**restore** como o comando **tar** suportam unidades de fita.

Atualmente, em muitas aplicações, torna-se praticável a utilização de discos que atualmente têm custos relativamente baixos e capacidades muito elevadas.

Seja qual for o tipo de suporte, há uma necessidade absoluta de garantir a separação física entre o sistema original e as respetivas cópias.

As cópias do sistema contêm frequentemente informação reservada, este facto deve estar presente quando se faz o seu manuseamento. Por exemplo, quando as cópias são realizadas através de uma rede.

Software de cópia



Os comandos Unix mencionados (**dump**/**restore** e **tar**) em combinação com o serviço *crond* e algumas linhas de programação em *shell* permitem implementar sistemas de cópia de arquivo e segurança totalmente eficazes.

Existem contudo programas mais fáceis de utilizar, muitas vezes estes programas recorrem internamente aos comandos **dump**/**restore** e **tar**.

Um dos exemplos *open source* mais divulgados é o Amanda (*http://www.amanda.org/*), trata-se de uma arquitetura cliente/servidor em rede que consiste num servidor onde são criadas as cópias de arquivo/segurança e de clientes que são usados nas máquinas cujos dados se pretende copiar: