Administração de Sistemas (ASIST)

Aula Teórico Prática 03

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

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

Grupos implícitos

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

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

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

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

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

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

Atribuição de permissões

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

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

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

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

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

Permissões sobre objetos do sistema (ACLs)

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

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

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

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

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

ACLs base (POSIX) dos sistema Linux

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

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

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

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

POSIX ACL – read, write and execute permissions

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

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

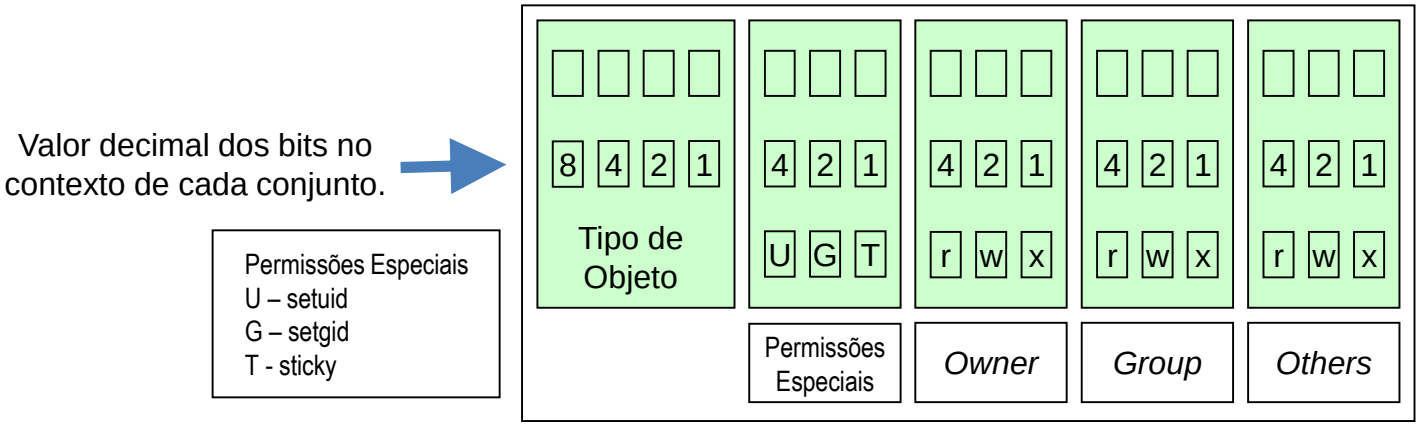
POSIX ACL – setuid, setgid and sticky permissions

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

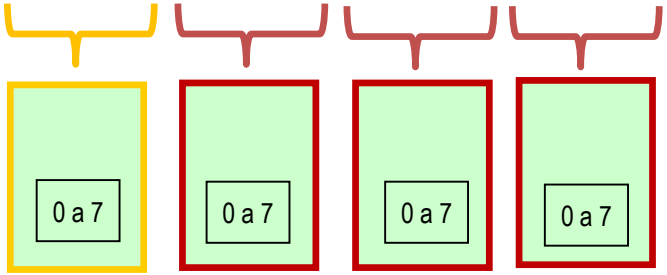
setuid	Se o utilizador possui as permissões necessárias para executar o ficheiro, então o processo criado poderá será executado com as permissões do proprietário do ficheiro, usando a *system-call* **setuid()**. Este expediente é usado em comando como ***passwd, sudo*** e ***ping*** por exemplo. São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

POSIX ACL – Representação octal das permissões

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



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



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

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

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

POSIX ACL – permissões iniciais - UMASK

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

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

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

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

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

umask 0022

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

Permissões Windows (permissões NTFS)

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

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

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

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

Windows – basic permissions and special permissions

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

Permissões Windows – herança e negação de permissões

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

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

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

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

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

ACLs Windows – DACL e SACL

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

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

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

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

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

Pasta pessoal (home directory)

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

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

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

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

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

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

Domínios Windows – *User Profile*

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

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

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

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

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

Domínios Windows – *User Profile* - cenários

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

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

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

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

Domínios Windows – *User Profile* - cenários

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

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

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

Controlo de quotas

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

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

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

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

Controlo de quotas – Linux e Windows Server

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

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

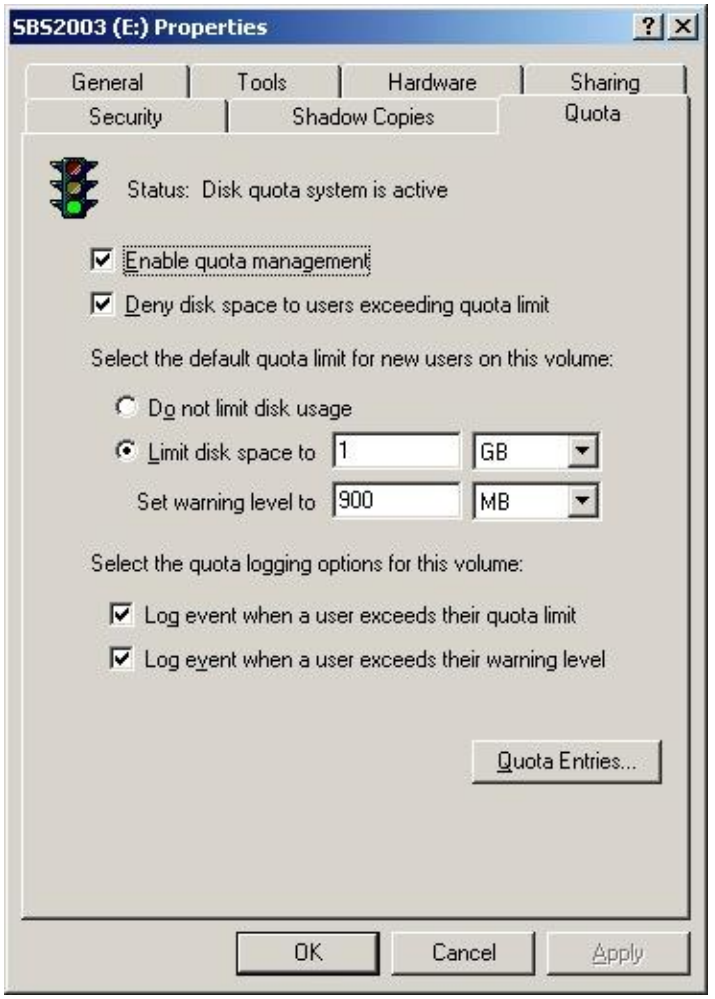
- ***Disk Quotas*** – introduzidas no *Windows 2000*, permitem controlar as quotas de utilizadores individuais na globalidade de um volume que esteja associado a uma letra de drive, novamente, não é de todo possível controlar as quotas de uma pasta individual. Não são suportadas quotas de grupo, no Windows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

Controlo de quotas – Windows Server

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

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

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



Controlo de quotas – Linux

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

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

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

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

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

Controlo de quotas – Linux

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

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

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

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

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

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

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

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

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

* Por omissão

Controlo de quotas – Linux

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

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

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

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

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

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

Login scripts

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

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

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

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

Linux - ficheiros de arranque da Shell

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

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

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

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

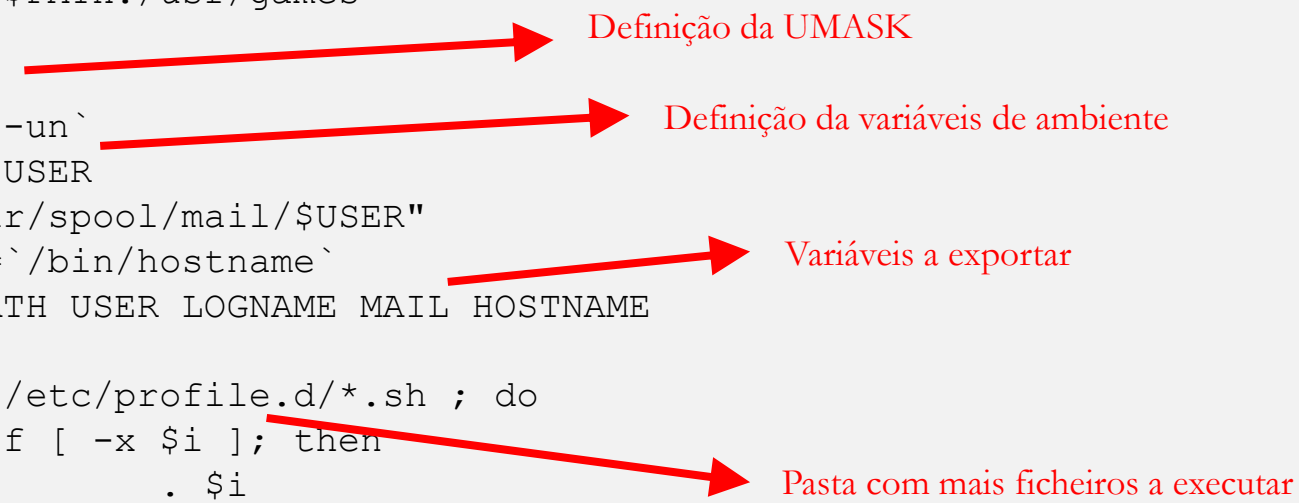
A C Shell, **/bin/csh** ou **/bin/csh**, executa a seguinte sequência de ficheiros de arranque: **/etc/csh.cshrc ; /etc/csh.login ; ~/.cshrc ; ~/.login.**

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

Linux – exemplo de ficheiro /etc/profile

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

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



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

Symbolic links

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

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

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

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

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

Sistemas Windows - *Symbolic links*

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

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

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

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

mklink [/D] SYMLINK TARGET

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

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

Sistemas Unix/Linux - *Symbolic links*

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

ln –s TARGET [SYMLINK]

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

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

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

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