GERENCIAMENTO DE USUÁRIOS

O Linux é um sistema multiusuário e possui um poderoso sistema de gerenciamento de usuários, grupos e permissões.

Para inserir um usuário, podemos usar o comando *useradd* com os parâmetros desejados:

useradd -g users -s /bin/bash -d /home/user user

No comando acima, o parâmetro *-g* informa de que grupo o usuário fará parte, o *-s* informa qual será o interpretador de comandos e o *-d* o diretório pessoal do usuário. As informações sobre os usuários ficam armazenadas no arquivo */etc/passwd*. Já as senhas serão guardadas criptografadas no arquivo */etc/shadow*, que não tem permissão de leitura para nenhum usuário. Isso evita os ataques de quebra de senha por força bruta. Para definir uma senha para o usuário, usa-se o comando *passwd*:

passwd user

A qualquer momento é possível modificar as propriedades do usuário com o comando *usermod*. Por exemplo, caso desejemos mudar o grupo que o usuário faz parte, criamos antes um novo grupo e depois modificamos o grupo do usuário:

groupadd devel
usermod -g devel user

Podemos ainda adicionar um usuário a outros grupos além do grupo principal:

usermod -a -G web joao

O *-a* do comando acima serve para adicionar. Sem ele, o usuário perderia todos os outros grupos e ficaria apenas no grupo *web*. Para remover o usuário usamos o comando *userdel*, que acompanhado pelo parâmetro *-r* remove junto o diretório pessoal:

userdel -r user

GERENCIAMENTO DE PERMISSÕES

Cada arquivo ou diretório possui um conjunto de permissões representado por caracteres que são mostrados no início de cada linha da listagem longa:

-rwxr-xr-x. 1 joao users 22K Jan 10 2011 tab.gif

Detalhando:
d --- --- --- : Mostra se é arquivo comum (-), diretório (d), block (b), character (c), pipe (p) ou link (l).
- rwx --- --- : Permissões do proprietário do arquivo.
- --- rwx --- : Permissões do grupo do arquivo.
- --- --- rwx : Permissões para todos os demais usuários.

Para alterar as permissões usamos o comando *chmod*. Veja os exemplos:

chmod u+x script.sh
chmod g+w script.sh
chmod o+r script.sh

No primeiro exemplo, damos a permissão de execução para o proprietário do arquivo. No segundo, damos permissão de escrita para todos os usuários que fizerem parte do grupo do arquivo. Por último, damos permissão de leitura para os outros usuários do sistema.

Poderíamos ter combinado tudo em um só comando:

chmod u+x,g+w,o+r script.sh

Podemos também especificar as permissões no formato clássico, representando as permissões por três números:

chmod -R 775 script.sh

Os três números indicam, respectivamente, as permissões de acesso para o dono, grupo e para os outros. Cada número representa a soma das permissões desejadas, sendo que:

4 – Leitura (r)
2 – Gravação (w)
1 – Execução ou listagem (x)

Você soma estes números para ter o valor referente ao conjunto de permissões que deseja.

SUID, SGID e STICKY BIT

SUID e SGID em executáveis

Normalmente os processos rodam sob permissões do usuário que o iniciou. O SUID e/ou SGID usado em executáveis faz com que o processo iniciado rode com as permissões do usuário/grupo proprietário do arquivo.

Para habilitar o SUID em um arquivo, usamos o *chmod*:

chmod u+s shutdown

Isso fará com que qualquer usuário consiga executar o comando *shutdown* como se fosse o usuário root, que é o proprietário do arquivo. Para definir o SGID, o comando é parecido:

chmod g+s arquivo

SGID em diretórios

Outro uso bastante interessante do SGID é em diretórios. Um diretório com SGID habilitado faz com que todos os arquivos criados dentro desse diretório pertençam ao mesmo grupo do diretório e não ao grupo do usuário que o criou. Isso é especialmente útil quando estamos trabalhando em diretórios colaborativos.

A sintaxe é a mesma:

chmod g+s diretorio

Sticky bit

Normalmente os usuários que tem permissão de escrita em um diretório podem apagar qualquer arquivo de diretório mesmo que não tenham essa permissão no arquivo. O uso do Sticky bit no diretório faz com que somente o dono dos arquivos consigam removê-los.

Veja o exemplo de como habilitar o sticky bit em um diretório:

chmod o+t diretorio

Há também a possibilidade de usarmos a sintaxe clássica do *chmod* para definirmos o SUID, SGID e Sticky bit. Para isso, adicionamos um número a esquerda do conjunto já conhecido de permissões. Por exemplo:

chmod 6755 arquivo

O 6 é o valor que representa a configuração de SUID, SGID e Sticky bit. Já os demais números são as permissões clássicas estudadas anteriormente. Para formar o 6, temos:

4 – SUID
2 – SUID
1 – Sticky bit

Novamente, você deve somar estes números para ter o valor referente ao conjunto de permissões que deseja.

FILE ACLs

Uma vez que o Linux não permite que usuários comuns troquem o dono de um arquivo, caso um usuário queira dar acesso a outro usuários (fora do seu grupo) para um arquivo qualquer ele deverá usar as permissões de “outros”. Isso dará acesso não só para o usuário desejado mas também para todos os outros usuários do sistema.

Para evitar situações como essa, podemos usar as ACLs de arquivos. Para isso, devemos montar a partição desejada com suporte a ACLs:

mount -o remount,acl /dados1

É recomendado colocar a opção *acl* no */etc/fstab*, tornando a configuração persistente. Agora que */dados1* tem suporte a ACL de arquivos, usamos o comando *setfacl* para determinar as permissões específicas:

setfacl -m u:joao:rwx arq1.txt

No exemplo acima, o *-m* indica que vamos modificar a acl, o *u* indica que as permissões serão para um usuário, no caso o *joao*, e *rwx* indica o que o *joao* poderá fazer com o *arq1.tx*t.
Para verificar as ACLs de um arquivo, usamos o comando *getfacl*:

getfacl arq1.txt

Há ainda a possibilidade de definirmos que ACLs os arquivos receberão quando criados dentro de um diretório específico. Para isso, devemos criar no diretório uma ACL default:

setfacl -m d:u:joao:rwx diretorio

No comando acima, o *d* indica que essa é uma ACL default,. Isso só faz sentido quando usado em diretórios. Para remover uma ACL, usamos o parâmetro *-x* em vez do *-m*. Nesse caso, não precisamos informar as permissões:

setfacl -x u:joao arq1.txt

QUOTA

O sistema de quotas embutido no kernel do Linux permite controlarmos o uso por usuários e grupos em um determinado sistema de arquivos. As limitações podem ser feitas por “blocks” ou por “inodes” e podemos definir limites “soft” e “hard”.
Devemos antes montar a partição desejada com suporte a quotas:

mount -o remount,usrquota,grpquota /dados1

É recomendado colocar as opções *usrquota* e *grpquota* no */etc/fstab*, tornando a configuração persistente. Depois disso, devemos inicializar o banco de dados de quotas na partição e ativar as quotas:

quotacheck -cugm /dados1
quotaon /dados1

Agora, podemos definir a quota de um determinado usuário:

edquota -u joao

O comando acima abrirá uma sessão de *vi*, com as configurações de quota para o usuário joao:

Disk quotas for user joao (uid 501):

Filesystem	*blocks*	*soft*	*hard*	*inodes*	*soft*	*hard*
/dev/sda8	*0*	*0*	*0*	*0*	*0*	*0*

No arquivo aberto, *blocks* indica os blocos ocupados atualmente pelo usuário. Abaixo de *soft* e *hard* podemos definir os limites de blocos para o usuário. *0* indica sem limites. A mesma lógica se aplica para *inodes*.
Para alterarmos a quota de um grupo usamos o mesmo *edquota*, agora com o parâmetro *-g*:

edquota -g users

O tempo que o usuário poderá exceder o *soft limit* é chamado de “grace preriod”. Esse tempo pode ser definido com o comando abaixo:

edquota -t

Tempos diferentes de “grace period” podem ser definidos para *blocks* e *inodes*.
Para um relatório completo da situação das quotas podemos usar o comando *repquota*:

repquota -a