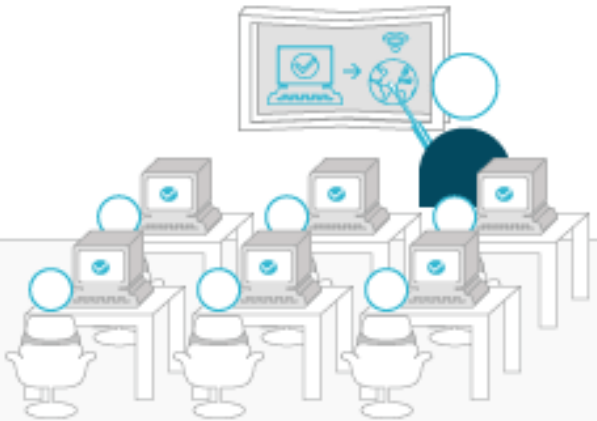
Título do Curso:

Administração de Sistemas GNU/Linux

Conteúdo do curso

ADMINISTRAÇÃO DE SISTEMAS GNU/LINUX

- Tópico 1: Introdução ao sistema operacional GNU/Linux.
- Tópico 2: Introdução ao Shell e comandos básicos.
- Tópico 3: Manipulação de conteúdos com comandos no Shell.
- Tópico 4: Comandos para gerenciamento do sistema e do Hardware.
- Tópico 5: Editor de Texto VI.
- Tópico 6: Administração de usuários e grupos.
- **Tópico 7: Gerenciamento de permissões.**
- Tópico 8: Gerenciamento de processos.
- Tópico 9: Sistemas de arquivos e particionamento.
- Tópico 10: Expressões regulares.
- Tópico 11: Introdução ao Shell Script.
- Tópico 12: Gerenciamento de Pacotes.
- Tópico 13: Agendamento de tarefas (cron) e Backup.



DGP

Tecnologia da Informação

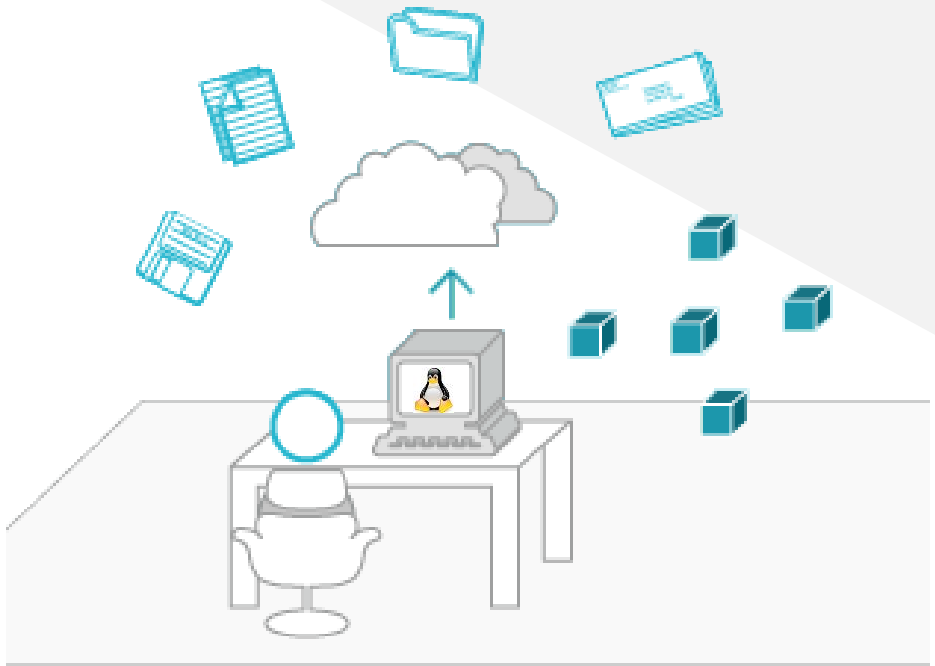
Tópico 7

Gerenciamento de permissões.

Gerenciamento de

Permissões de acesso.

- Neste slide serão abordados conceitos e comandos que nos auxiliam no gerenciamento das permissões de acesso a arquivos e diretórios:
  - Conceitos sobre permissões de acesso;
  - Permissões padrão (*file mode*);
  - Permissões avançadas (*special modes*).



Conceitos sobre permissões de acesso

Tópico 7: Gerenciamento de permissões.

Conceitos sobre Permissões de acesso

- As permissões de acesso a arquivos e diretórios em um sistema operacional possibilitam definir quem pode acessar determinado recurso e o nível de privilégio sobre o item acessado.

- Desta forma é possível prover segurança em arquivos essenciais para o funcionamento do sistema, evitando que um usuário mal intencionado ou que um software instável/vulnerável modifique o conteúdo destes arquivos.

- Resumidamente, as permissões são atributos de arquivos que definem se o arquivo pode ser lido (**R**ead), escrito/gravado (**W**rite) e/ou executado (e**X**ecute). Estes atributos podem ser visualizados com o comando "**ls -l**".

Conceitos sobre Permissões de acesso

- No Linux, as permissões são sequências de 12 bits, sendo que os três bits iniciais são destinados a atribuição de “permissões especiais” ou “estendidas” (*special modes*), e os 9 bits restantes (*file mode*), determinam as permissões para três classes de usuários, sendo:
  - Usuário proprietário/dono do arquivo;
  - Grupo proprietário do arquivo;
  - Outros usuários do sistema (ou seja, todos os demais usuários).

- Os 9 bits definidos pelo *file mode* (utilizados e visualizados com mais frequência), possuem o formato “rwx rwx rwx”, sendo que a cada três bits (ou sequência “rwx”), determinamos o nível de permissões do “usuário proprietário”, grupo e demais usuários respectivamente.

Conceitos sobre Permissões de acesso

- O mesmo tipo de permissão é aplicada em diretórios e arquivos, porém, temos algumas diferenças ao aplicar o mesmo nível de privilégio em um arquivo ou diretório:

Permissão	Atuação sobre o arquivo	Atuação sobre o diretório
(r) Read	Permite visualizar o conteúdo do arquivo	Permite visualizar o conteúdo do diretório
(w) Write	Permite modificar e apagar o arquivo	Permite criar/apagar arquivos/subdiretórios
(x) eXecute	Permite que o mesmo seja executado	Permite acessar o diretório (com o comando "cd" por exemplo)

Permissões padrão (*file mode*)

Tópico 7: Gerenciamento de permissões.

Modificando as permissões com o “chmod”

- chmod → Altera as permissões de acesso a um arquivo ou diretório:
  - Ex.: chmod [permissões] [arquivo]
  - OBS.: Com o “**chmod**” as permissões podem ser definidas de duas formas:
    - Modo Octal (também conhecido como modo numérico):

Permissão	Modo Octal
(r) Read	Valor = 4
(w) Write	Valor = 2
(x) eXecute	Valor = 1

    - Modo Literal (também conhecido como modo textual):

Usuário	Modo Literal
Usuário Proprietário	Valor = u
Grupo Proprietário	Valor = g
Outros	Valor = o
TODOS	Valor = a

Entendendo o modo OCTAL Comando “chmod”

- Para compreender os números utilizados no modo OCTAL do comando “**chmod**”, observe a conversão de binário para decimal na tabela a seguir, bem como o nível de permissões e sua representação no Linux:

Permissões	Binário	Decimal	Descrição / Privilégio
---	000	0	Nenhuma Permissão
--x	001	1	Execução
-w-	010	2	Gravação
-wx	011	3	Gravação e Execução
r--	100	4	Leitura
r-x	101	5	Leitura e Execução
rw-	110	6	Leitura e Gravação
rwx	111	7	Leitura, Gravação e Execução

Modificando as permissões com o “chmod”

- chmod → COMO UTILIZAR???
  - Ex.: chmod [permissões] [arquivo]
    - **Modo Octal:**

```
[root@mail directory]# ls -l
total 0
-rw-r--r-- 1 root root 0 Abr 17 11:02 arquivo
```

• Permissão padrão ao criar o arquivo;

```
[root@mail directory]# chmod 777 arquivo
[root@mail directory]# ls -l
total 0
-rwxrwxrwx 1 root root 0 Abr 17 11:02 arquivo
```

• Aplicando permissão total para todos;

```
[root@mail directory]# chmod 000 arquivo
[root@mail directory]# ls -l
total 0
---------- 1 root root 0 Abr 17 11:02 arquivo
```

• Restringindo o acesso ao arquivo;

```
[root@mail directory]# chmod 644 arquivo
[root@mail directory]# ls -l
total 0
-rw-r--r-- 1 root root 0 Abr 17 11:02 arquivo
```

• Aplicando permissões de leitura para todos e escrita apenas para o dono.

Modificando as permissões com o “chmod”

- chmod → COMO UTILIZAR???
  - Ex.: chmod [permissões] [arquivo]
    - **Modo Literal:**

```
[root@mail directory]# ls -l
total 0
-rw-r--r-- 1 root root 0 Abr 17 11:02 arquivo
```

• Permissão padrão ao criar o arquivo;

```
[root@mail directory]# chmod a+x arquivo
[root@mail directory]# ls -l
total 0
-rwxr-xr-x 1 root root 0 Abr 17 11:02 arquivo
```

• Aplicando permissão de execução para todos;

```
[root@mail directory]# chmod o-x arquivo
[root@mail directory]# ls -l
total 0
-rwxr-xr-- 1 root root 0 Abr 17 11:02 arquivo
```

• Removendo permissão de execução de “outros”;

```
[root@mail directory]# chmod g=rw arquivo
[root@mail directory]# ls -l
total 0
-rwxrw-r-- 1 root root 0 Abr 17 11:02 arquivo
```

• Definindo que o grupo terá permissões de leitura e escrita.

Modificando os proprietários dos arquivos

- chgrp → Modifica o grupo proprietário do arquivo.
  - Ex.: chgrp [grupo] [arquivo]

```
[root@mail directory]# useradd aluno
[root@mail directory]# groupadd grupo-alunos
[root@mail directory]# chgrp grupo-alunos arquivo
[root@mail directory]# ls -l
total 0
-rwxrw-r-- 1 root grupo-alunos 0 Abr 17 11:02 arquivo
```

  - Na imagem acima, temos:
    - A criação do usuário “aluno”;
    - Criação do grupo “grupo-alunos”;
    - Alteração do grupo proprietário do arquivo com o comando “chgrp”, onde foi definido que o grupo “grupo-alunos” será o proprietário do “arquivo”;

Modificando os proprietários dos arquivos

- chown → Altera o usuário proprietário (*owner*) de um arquivo:
  - Ex.: chown [owner] [arquivo]
    - No exemplo abaixo, o comando "**chown**" define o usuário "**aluno**" como proprietário do "**arquivo**":

```
[root@mail directory]# chown aluno arquivo
[root@mail directory]# ls -l
total 0
-rwxrw-r-- 1 aluno grupo-alunos 0 Abr 17 11:02 arquivo
```

  - OBS.: Através do comando "**chown**" podemos modificar em uma única linha de comando o usuário e o grupo proprietário de um arquivo, conforme exemplo a seguir:

```
[root@mail directory]# chown root.root arquivo
[root@mail directory]# ls -l
total 0
-rwxrw-r-- 1 root root 0 Abr 17 11:02 arquivo
```

Permissão padrão – User Mask

- UMASK (User Mask): Permite definir a permissão padrão para os novos arquivos e diretórios a ser criados no sistema Linux.

- Por padrão, temos uma "**umask**" definida para o usuário "**root**" e outra "**umask**" definida para todos os demais usuários.

Máscara (umask) padrão do usuário **root**:

```
[root@mail directory]# umask
0022
```

Máscara (umask) padrão de um usuário **comum**:

```
[chico@mail ~]$ umask
0002
```

- Apesar do sistema definir a permissão padrão para o "**root**" e demais usuários, podemos visualizá-la e modificá-la através do comando "**umask**".

Permissão padrão – User Mask

- umask → Exibe ou define a máscara de permissões padrão para a criação de novos arquivos e diretórios.
  - Ex.: umask <máscara de permissões>

- ***OK, porém, qual será a permissão padrão se a minha umask = 0022??***
  - Simples, basta fazer uma conta:
    - Caso seja um arquivo, subtraímos a "**umask**" do número "666";

      666 _
      0022
      0644

    - Caso seja um diretório, subtraímos a "**umask**" do número "777";

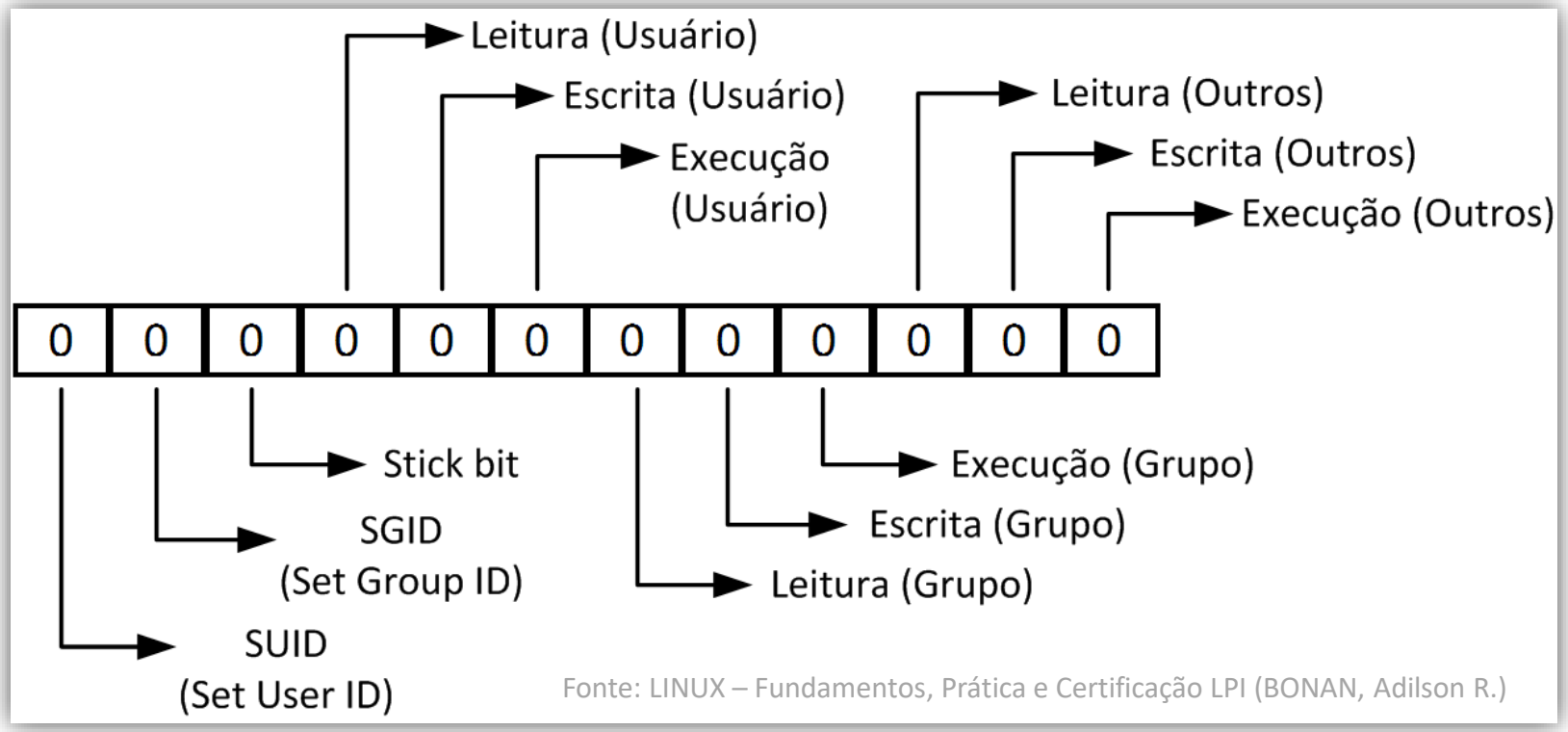
      777 _
      0022
      0755

Permissões avançadas (*special modes*)

Tópico 7: Gerenciamento de permissões.

Permissões avançadas (*special modes*)

- Conforme descrito, temos 12 bits para definir as permissões de acesso, sendo que os 3 primeiros definem as permissões avançadas e os 9 bits restantes (*file mode*), definem as permissões básicas.



Fonte: LINUX – Fundamentos, Prática e Certificação LPI (BONAN, Adilson R.)

Permissões avançadas (*special modes*)

- Portanto, temos 3 tipos de permissões especiais, sendo:
  - **SUID (Set User ID):** Não tem efeitos sobre diretórios, utilizada apenas em arquivos executáveis, fazendo com que o mesmo seja executado com os privilégios de seu usuário proprietário.
    - OBS.: CUIDADO ao utilizar este tipo de permissão, pois a grande maioria dos executáveis possuem o "**root**" como proprietário.
  - **SGID (Set Group ID):** Parecido com o SUID, porém aplicado ao grupo, fazendo com que um arquivo executável utilize os privilégios do grupo proprietário do arquivo.
  - **Sticky Bit:** Utilizado para controlar o acesso a determinados arquivos, de forma que apenas o proprietário do arquivo ou diretório consiga removê-lo do sistema. (Temos como exemplo o "**/tmp**").

Permissões avançadas (*special modes*)

- Para definir as permissões especiais, podemos utilizar os modos “octal” e “literal” da mesma forma, porém:
  - Ao invés de 3, utilizamos 4 números no modo OCTAL;

```
[root@localhost directory]# chmod -v 7644 arquivo
modo de `arquivo' mudado para 7644 (rwSr-Sr-T)
```

  - Ao invés de “**rwx**”, utilizamos “**sst**” no modo LITERAL;

```
[root@localhost directory]# chmod -v +sst teste
modo de `teste' mudado para 7644 (rwSr-Sr-T)
```

Permissões avançadas (*special modes*)

- OBSERVAÇÕES:
  - As permissões especiais são exibidas no “espaço” destinado a permissão de execução do usuário (SUID), grupo (SGID) e outros (Stick Bit);
  - Caso o arquivo/diretório tenha permissões de execução, a permissão especial será exibida com letras minúsculas (sst), caso o arquivo/diretório não tenha permissões de execução, a permissão especial será exibida em “CAIXA ALTA” (SST), conforme exemplo abaixo:

```
[root@localhost directory]# ls -l
total 0
-rwxr-xr-x 1 root root 0 Abr 18 13:32 script.sh
[root@localhost directory]# chmod -v +sst script.sh
modo de `script.sh' mudado para 7755 (rwsr-sr-t)
```

```
[root@localhost directory]# chmod -v 7644 arquivo
modo de `arquivo' mudado para 7644 (rwSr-Sr-T)
```

No próximo slide...

- Tópico 8: Gerenciamento de processos.

Referências

- BONAN, Adilson Rodrigues. **LINUX – Fundamentos, Prática & Certificação LPI**. Editora: Alta Books. RJ. 2010;

- PEREIRA, Guilherme Rodrigues. **Slides para aula expositiva**. Centro Universitário UNA.

- SILVA, Gleydson Mazioli. **Guia Foca GNU/Linux**. Disponível em: https://guiafoca.org/

DGP

Tecnologia da Informação

Obrigado!

Guilherme Rodrigues