

**PET**  
COMPUTAÇÃO

# Curso de Linux

---

# O que é **LINUX**?

---

- **Sistema Operacional**
- **Diferentes *distribuições***
- **Usado no Departamento de Informatica (Dinf)**
  - Recursos
    - Pegar login e senha na Secretaria
    - Página Pessoal
    - 4GB de espaço
      - /home
      - /nobackup

# Mas porque usá-lo?

---

- *Gratuito*
- **Maior controle**
- *Segurança*
- **90% dos computadores do mundo**
  - Google
  - Facebook
  - Weta Digital
- **Melhor ambiente para programação**

# O que são as **distribuições**?

---

- *“Sabores”* do linux



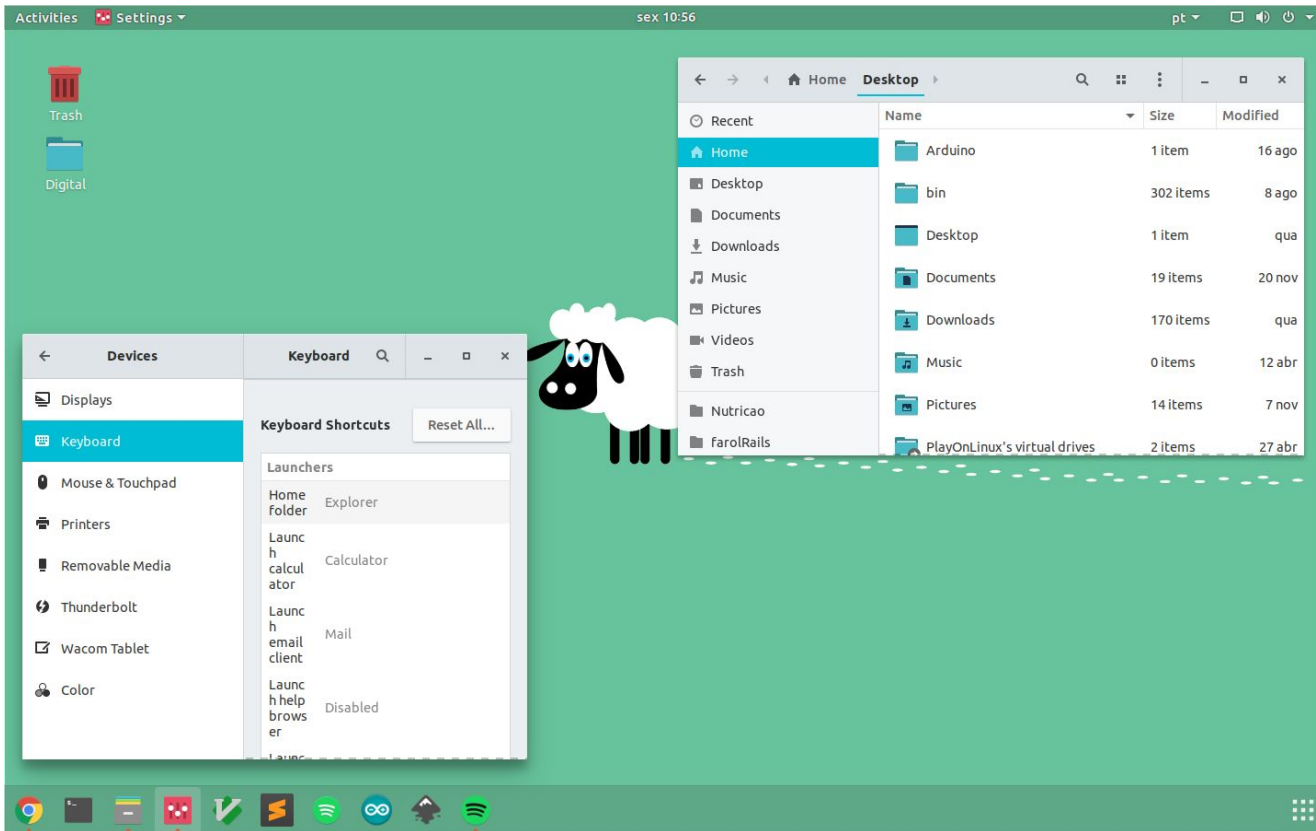
debian



archlinux™

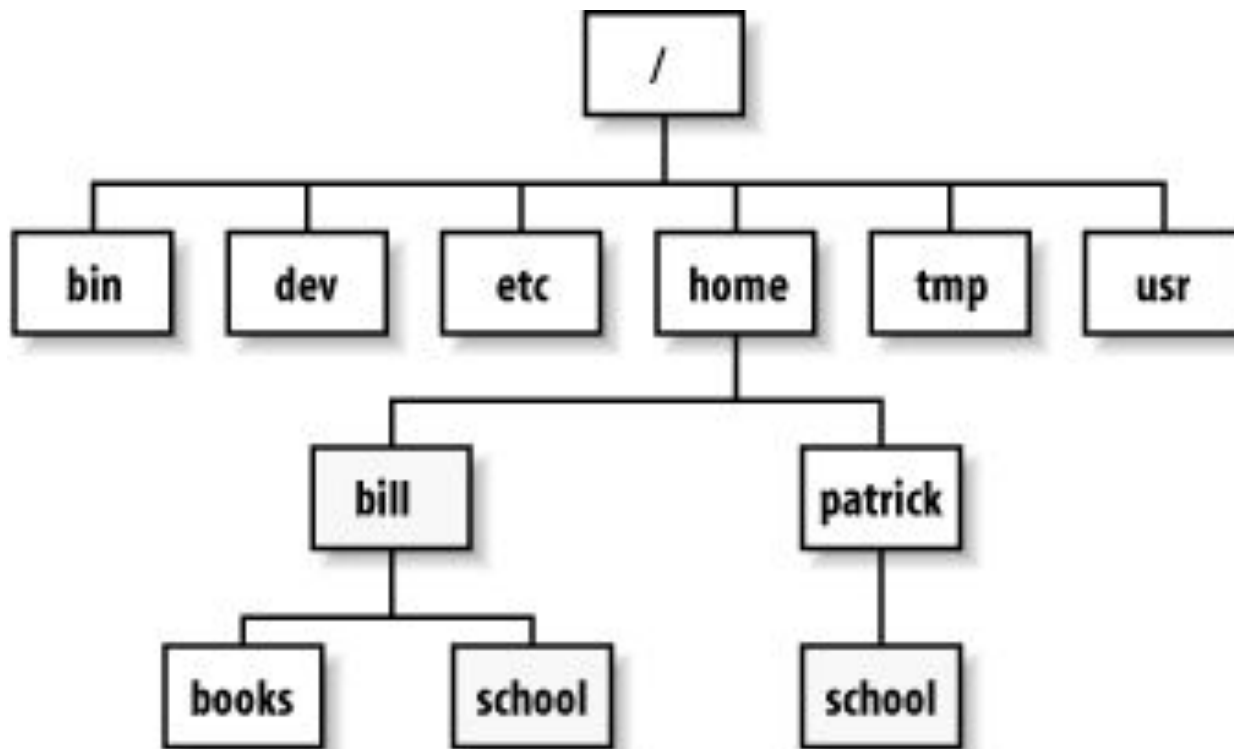
# Começando

- Familiarize-se



# Diretórios

- Usuário sempre atua sob um diretório



# Caminhos

---

## Caminho absoluto

/bin/grep

/boot/grub/locale

## Caminho relativo

~/Documentos

.

..

# Caminhos relativos e absolutos

---

São atalhos para navegação

"/" raiz

"~" home

"." pasta atual

".." pasta acima



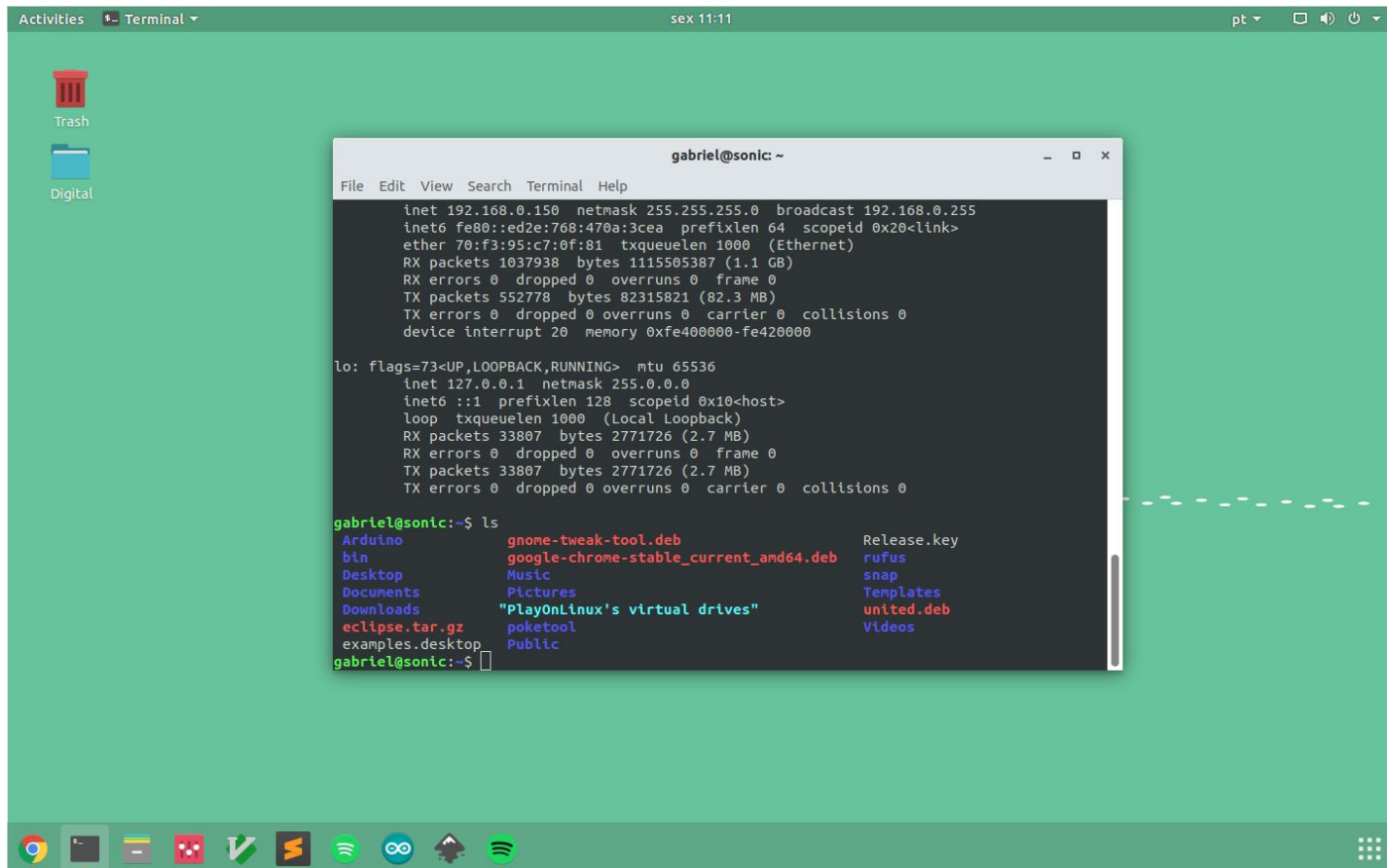
# Programas

---

- **Gerenciadores de pacotes**
  - Você pode instalar programas sem fazer buscas na internet
  - Repositórios:
    - **Apt-get, urpmi, yum**
- **Arquivos de “instalação”**
  - **.deb, .rpm, .tgz**
- **Dependências**

# terminal

- Atalho: CTRL + ALT + T



```
gabriel@sonic: ~  
File Edit View Search Terminal Help  
inet 192.168.0.150 netmask 255.255.255.0 broadcast 192.168.0.255  
inet6 fe80::ed2e:768:470a:3cea prefixlen 64 scopeid 0x20<link>  
ether 70:f3:95:c7:0f:81 txqueuelen 1000 (Ethernet)  
RX packets 1037938 bytes 1115505387 (1.1 GB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 552778 bytes 82315821 (82.3 MB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
device interrupt 20 memory 0xfe400000-fe420000  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 33807 bytes 2771726 (2.7 MB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 33807 bytes 2771726 (2.7 MB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
gabriel@sonic:~$ ls  
Arduino          gnome-tweak-tool.deb          Release.key  
bin              google-chrome-stable_current_amd64.deb  rufus  
Desktop         Music                          snap  
Documents       Pictures                       Templates  
Downloads       "PlayOnLinux's virtual drives"  united.deb  
eclipse.tar.gz  poketool                      Videos  
examples.desktop  Public  
gabriel@sonic:~$
```

# Comandos no terminal

- Estrutura de comandos:

[comando][opções][argumentos]

- Comandos *básicos*:
  - *ls* - *List*
    - *Lista arquivos e diretórios em um diretório*
  - *cd* - *Change Directory*
    - *Move o usuário entre diretórios*
  - *pwd* - *Print Working Directory*
    - *Exibe o diretório atual do usuário*

## Estrutura de um comando:

**[comando] [opções] [argumentos]**

**ls** Lista pastas e arquivos do diretório atual

Opções interessantes:

**ls pasta** - lista conteúdo dentro da pasta especificada

**ls -a** - mostra arquivos ocultos (os que começam com .)

**ls -l** - mostra mais informações (dono, permissões, tamanho)

**ls -1** - exibe um item por linha

**ls -R** - exibe um item por linha

# cd

Vai para pasta especificada  
(**C**hange **d**irectory)

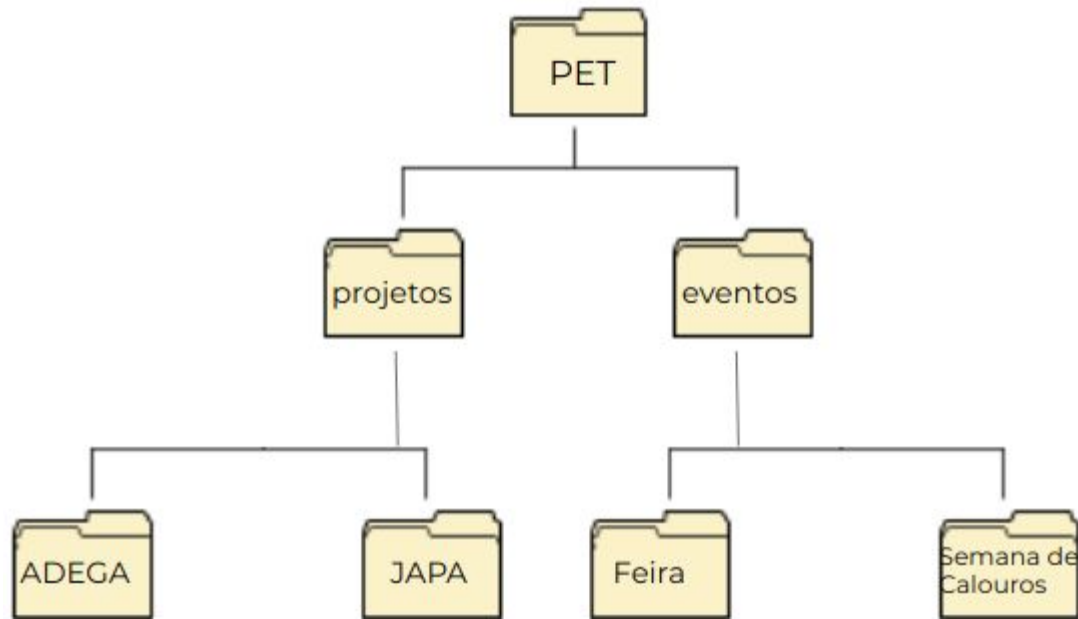
**cd** apenas o comando, vai para a home

**cd -** vai pra pasta anterior ao último cd

**cd ..** vai pra pasta acima

# Diretório de arquivos - praticando caminho relativo

**PROBLEMA EXEMPLO:** Dar “ls” nas pastas sem ter que entrar nelas (sem usar cd).



# Diretório de arquivos - caminho relativo

Estando na pasta PET

Ver o conteúdo da pasta...

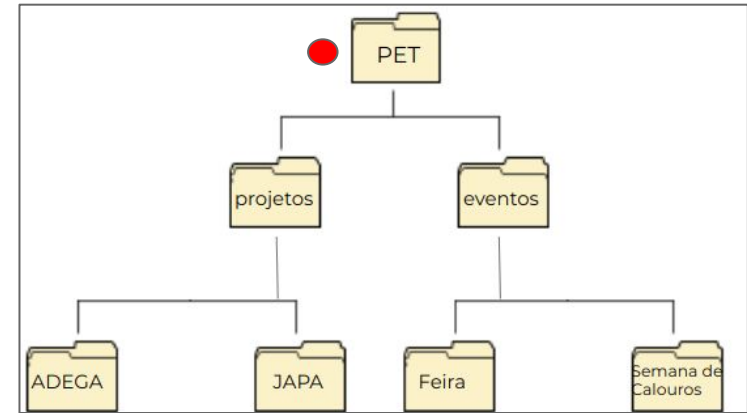
PET: `$ ls`

projetos: `$ ls projetos/`

eventos: `$ ls eventos/`

ADEGA: `$ ls projetos/ADEGA/`

Feira: `$ ls eventos/Feira/`



## Diretório de arquivos - caminho relativo

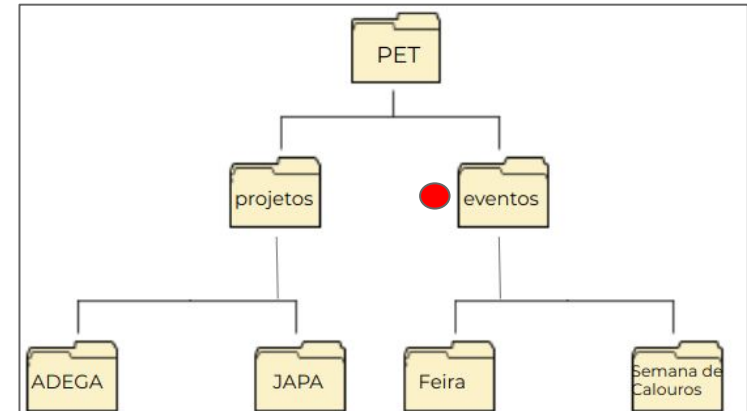
Se você está na pasta eventos  
Ver o conteúdo da pasta...

PET: `$ ls ..`

projetos: `$ ls ../projetos/`

ADEGA: `$ ls ../projetos/ADEGA/`

Feira: `$ ls Feira/`





# Diretório de arquivos - caminho relativo

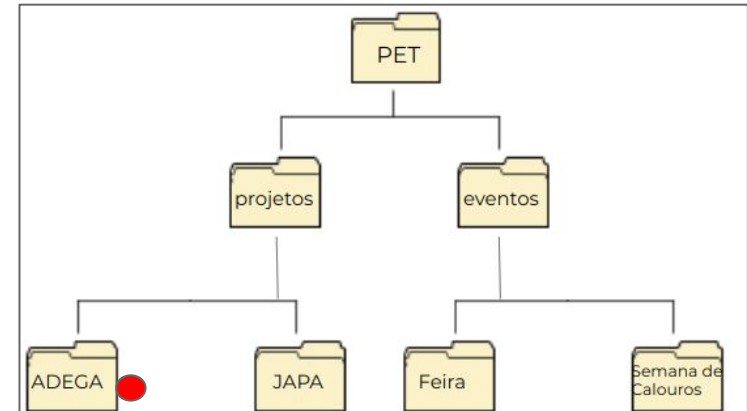
Se você está na pasta ADEGA  
Ver o conteúdo da pasta...

PET: `$ ls ../../..`

projetos: `$ ls ../`

ADEGA: `$ ls`

Feira: `$ ls ../../..../eventos/Feira/`



## cd, pwd - EXEMPLO

---

```
user@machine:~ $ pwd
/home
user@machine:~ $ cd documentos/
oficina/ projetos/
user@machine:~ $ cd documentos/oficina/
user@machine:~ $ pwd
/home/documentos/oficina
user@machine:~ $ cd ../../
user@machine:~ $ pwd
/home
user@machine:~ $ cd -
/home/documentos/oficina
user@machine:~ $ cd -
/home
```

← <tab> <tab>

Mostra as pastas que tem dentro

# Comandos de Gerenciamento de Pastas

---

## **mkdir**

Cria um diretório

## **rmdir**

Remove um diretório vazio

```
user@machine:~$ mkdir Aula
user@machine:~$ mkdir Aula\ de\ Linux
user@machine:~$ rmdir Aula
```

Caractere de escape: "\"

# Importância do caractere de escape: "\"

```
user@machine: mkdir Um\ nome\ com\ espaços
user@machine: ls -n
total 4
drwxrwxr-x 2 1000 1000 4096 Fev 20 23:10 Um nome com espaços
```

O caractere de escape faz o próximo caractere ter seu valor literal e não ser interpretado pelo bash

```
user@machine: mkdir Um nome com espaços
user@machine: ls -n
total 16
drwxrwxr-x 2 1000 1000 4096 Fev 20 23:22 com
drwxrwxr-x 2 1000 1000 4096 Fev 20 23:22 espaços
drwxrwxr-x 2 1000 1000 4096 Fev 20 23:22 nome
drwxrwxr-x 2 1000 1000 4096 Fev 20 23:22 Um
```

# man

---

- **Manuais** sobre programas, utilitários ou funções
  - Acesso fácil e rápido
  - Não depende de conexão com a internet
- **Experimente:**
  - `man pwd`
  - `man ascii`
  - `man man`

# Gerenciamento de Arquivos

- **touch** [nome]: cria um arquivo (ou só atualiza sua data)
- **rm** [nome]: remove um arquivo
- **cp** [arquivo] [destino]: copia um arquivo
- **mv** [arquivo] [destino]: recorta um arquivo
- **mv** [arquivo] [novo nome]: renomeia um arquivo

```
user@machine:~$ touch teste
user@machine:~$ cp teste ..
user@machine:~$ mv ../teste .
user@machine:~$ mv teste testando
```

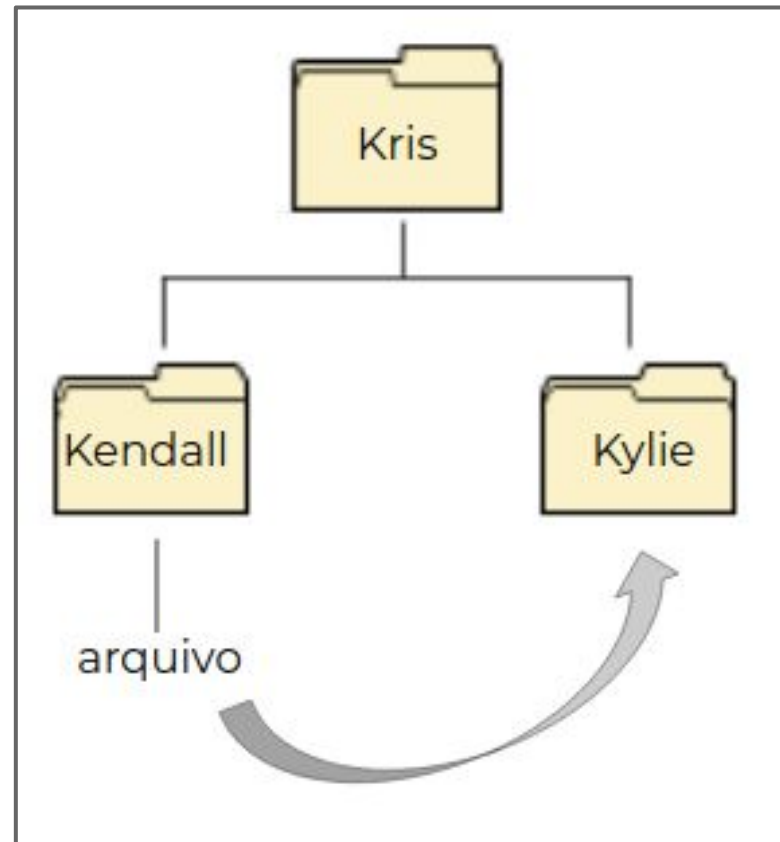
# mv e caminho relativo

**caminho relativo** → descreve o caminho do arquivo a partir da pasta em que se está

## PROBLEMA

## EXEMPLO:

**Mover um arquivo para a pasta irmã**



## mv e caminho relativo

### EXEMPLO: Mover um arquivo para a pasta irmã

Se você está na pasta Kris:

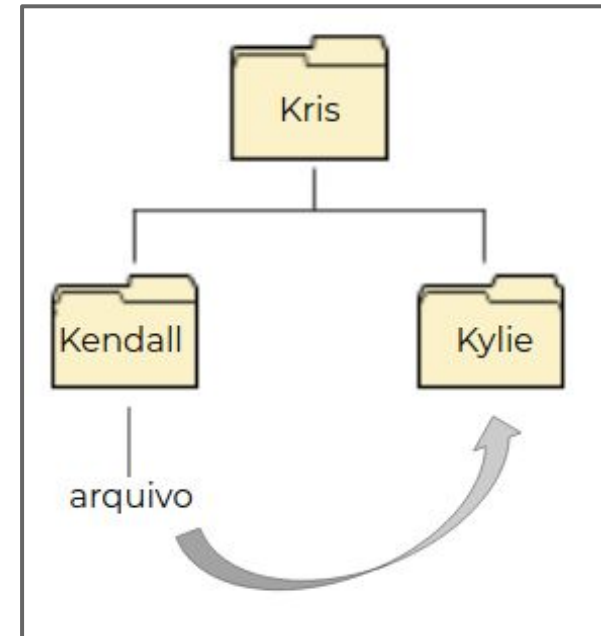
```
$ mv Kendall/arquivo Kylie/
```

Se você está na pasta Kendall:

```
$ mv arquivo ../Kylie/
```

Se você está na pasta Kylie:

```
$ mv ../Kendall/arquivo .
```



Lembre:

“.” a pasta atual

“..” a pasta acima



# Gerenciamento de Arquivos

---

- ❑ file [arquivo]: mostra o tipo do arquivo
- ❑ du [arquivo/diretorio]: mostra o tamanho do arquivo/diretório (disk usage)
  - ❖ Para saber o espaço usado pelos seus diretórios:  
\$ du -shc \* | sort -nr
- ❑ quota -s : Para saber quanto você usa da quota



# Editores de texto

nano, geany, gedit, vim, emacs, sublime

# Entrada e saída de dados

---



**cat** Envia dados para a *saída padrão (o terminal)*

- **cat arquivo:** imprime conteúdo do arquivo
- **cat:** imprime a *entrada padrão (o teclado)* → CTRL + D p/ sair
- **cat arquivo1 arquivo2:** concatena(junta) dados e imprime

## **tac**

- Igual ao cat, porém inverte a ordem das linhas

# Entrada e saída

---

## ▣ echo

Imprime o texto inserido (o argumento)

## Redirecionamento

- ▣ enviar dados para um arquivo, em vez da saída padrão
  - ▣ destrutivo e não-destrutivo: `>` e `>>`
  - ▣ ex: `echo oi > arquivo.txt`
  
- ▣ Enviar dados de um arquivo para a entrada padrão: `<`

# Entrada e saída de dados EXEMPLO

---

```
user@machine:~$ echo "Este é um teste"
Este é um teste
user@machine:~$ echo "Este é outro teste" > teste.txt
user@machine:~$ cat teste.txt
Este é um teste
user@machine:~$ echo "Este é o ultimo teste" > teste.txt
user@machine:~$ cat teste.txt
Este é o ultimo teste
user@machine:~$ echo "Mentira" >> teste.txt
user@machine:~$ cat teste.txt
Este é o ultimo teste
Mentira
```

# Entrada e saída de dados EXEMPLO

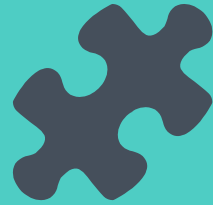
---

```
user@machine: cat > arquivo.txt
escrevendo no arquivo
strogonoff
gelatina
^C
user@machine: cat arquivo.txt
escrevendo no arquivo
strogonoff
gelatina
user@machine: █
```

# Entrada e saída de dados EXEMPLO

```
user@machine: ls
oficina projetos
user@machine: ls > conteudo.txt
user@machine: cat conteudo.txt
conteudo.txt
oficina
projetos
```

```
user@machine: tac conteudo.txt >> conteudo.txt
user@machine: cat conteudo.txt
conteudo.txt
oficina
projetos
projetos
oficina
conteudo.txt
```



# Caracteres coringas

Expressões regulares: representando padrões de busca



# Expressões regulares no Linux (caracteres coringas)

---

- \* faz referência a um nome completo/restante de um arquivo/diretório
- ? faz referência a uma letra naquela posição
- **[a-b]** faz referência a uma faixa de caracteres de um arquivo/diretório.
- **{a..b}** usado com comandos como touch ou mkdir.

# Expressões regulares no Linux (coringas) EXEMPLOS

---

```
user@machine:~ $ ls
oficina projetos
user@machine:~ $ touch teste{1..4}.txt
user@machine:~ $ ls
oficina projetos teste1.txt teste2.txt teste3.txt teste4.txt
user@machine:~ $ ls *.txt
teste1.txt teste2.txt teste3.txt teste4.txt
user@machine:~ $ ls teste?.txt
teste1.txt teste2.txt teste3.txt teste4.txt
user@machine:~ $ ls teste[1-3].txt
teste1.txt teste2.txt teste3.txt
```

# Compactando arquivos (tar com opção -c)

---

- **tar -cf ["nome"] [arquivo/diretório]:** Cria um arquivo .tar contendo os arquivos
- **tar -zcf ["nome"] [arquivo/diretório]:** Cria um arquivo .tar.gz contendo os arquivos

```
user@machine:~$ tar -cf Downloads.tar Downloads/  
user@machine:~$ tar -cf meuArquivoCompactado.tar arquivo.pdf  
user@machine:~$ tar -zcf Downloads.tar.gz Downloads/  
user@machine:~$ tar -zcf meuArquivoCompactado.tar.gz arquivo.pdf
```

# Descompactando arquivos (tar com opção -x)

---

- ▣ **tar -xf ["nome"]**: Descompacta um arquivo .tar no diretório local
- ▣ **tar -zxf ["nome"]**: Descompacta um arquivo .tar.gz no diretório local

```
user@machine:~$ tar -xf Downloads.tar
user@machine:~$ tar -xf meuArquivoCompactado.tar
user@machine:~$ tar -zxf Downloads.tar.gz
user@machine:~$ tar -zxf meuArquivoCompactado.tar.gz
```

# ALTERE SUA SENHA COMANDO PASSWD

```
user@machine:~$ passwd
Current password:
Enter new password:
Retype new password:
user@machine:~$
```

SÓ NÃO VAI ESQUECER, HEIN?!  
NÃO PASSE SUA SENHA PARA **NINGUÉM**



# Permissões

O que cada usuário ou grupo do sistema pode fazer

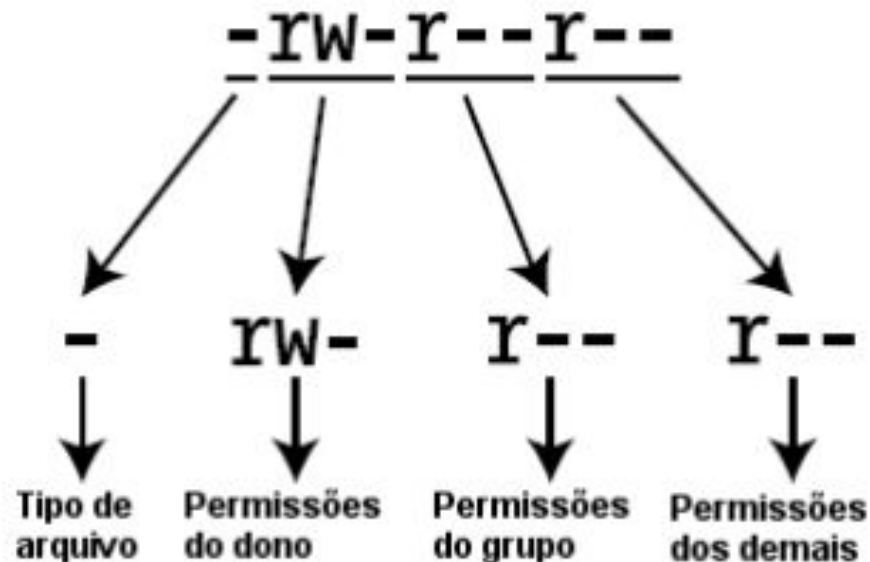
# Permissões

---

- ▣ Três tipos:
  - ▣ **R**ead: apenas leitura
  - ▣ **W**rite: gravação, alteração e deleção
  - ▣ **E**xecute: execução
  
- ▣ Três grupo:
  - ▣ **U**ser: você
  - ▣ **G**roup: a sua rede
  - ▣ **O**thers: o resto do mundo

# Permissões - E como ver?

ls -l



```
$ ls -l
```

```
drwxr-xr-x 2 chapolin pet-user 4096 2011-02-25 16:22 Comandos
-rw-r--r-- 1 chapolin pet-user 392 2011-02-25 14:37 Makefile
```



# Permissões - E como alterar?

chmod

## Notação textual:

```
$ chmod (ugoa)(+ -=)(rwx) arquivos
```

1º argumento - usuarios(user, group, others, all)

2º argumento - tipo de alteração

3º argumento - tipo de permissão

**Notação octal:** não veremos

# Permissões

---

## EXEMPLO

```
$ ls -l
-rw-rw-r-x 1 chapolin pet-user
$ chmod u=rw,g=r,o= Makefile
$ ls -l
-rw-r----- 1 chapolin pet-user
```

# Permissões - utilidade

---

## Página pessoal:

chmod o+rx arquivo.txt  
na pasta ~/public\_html



# Atalhos do teclado

Usando o terminal com agilidade e praticidade ;)

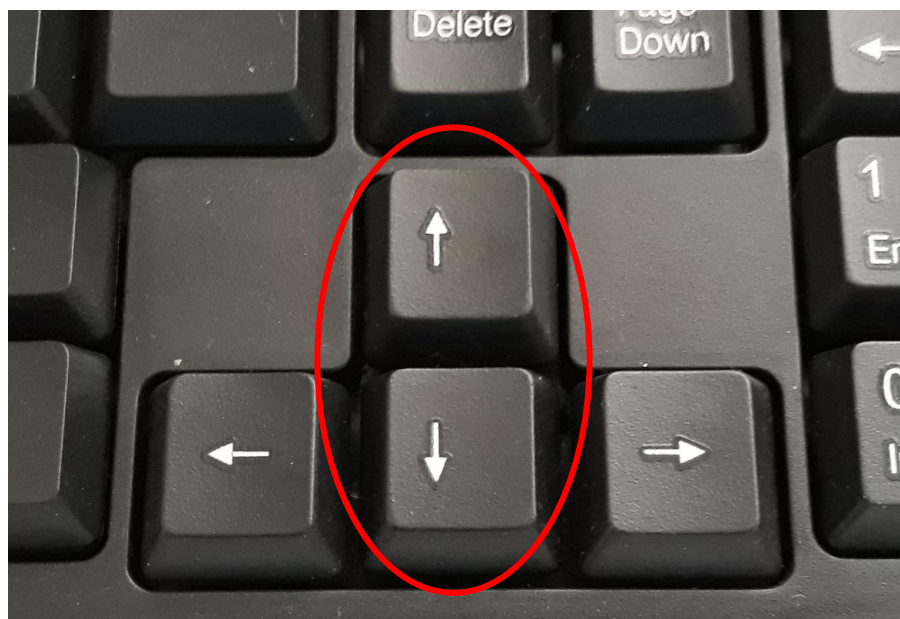
- **ctrl+alt+t** abre o terminal
- **Setas (up/down)** navegar pelos últimos comandos escritos
- **ctrl+shift+ c/v** para copiar e colar no terminal
- **ctrl + a** pula para o início do comando
- **ctrl + e** pula para o final do comando
- **ctrl + u** limpa o que está escrito antes do cursor
- **ctrl + l** limpa a tela (mesmo que o comando “clear”)
- **Tab** preenche o comando/argumento (quando possível)
- **Tab tab** mostra pastas ou arquivos que tem na pasta
- **ctrl + r** pesquisa reversa (busca no seu histórico de comandos)
- **ctrl + c** interrompe o processo

Obs: se o terminal “congelou” talvez você tenha apertado ctrl+s (stop), para sair(quit) faça ctrl+q

## Teclas up e down

Navegue pelo seu histórico de comandos indo pra cima e pra baixo

**Evite ficar digitando comandos que você acabou de dar: procure eles no histórico**



## TECLA TAB (AUTO-COMPLETE)



### PROBLEMA EXEMPLO:

```
user@machine: ls  
Supercalifragilisticexpialidocious Supercão
```

→ queremos ir para a pasta Supercalifragilisticexpialidocious

**Não precisa digitar o nome inteiro**

Apenas “cd Superca <tab>”

```
user@machine: cd Superca  
Supercalifragilisticexpialidocious/ Supercão/  
user@machine: cd Supercalifragilisticexpialidocious/
```



# Protocollo SSH



# Secure Shell

---

- `ssh login@computador`: acessa outro computador pelo terminal

```
user@machine:~$ ssh user@destiny
user@destiny password:          #digitar a senha do pc a ser acessado
```

- `scp [arquivo] login@computador:~/ :` copia um arquivo para outro computador

```
user@machine:~$ scp file.txt user@destiny:~/Trabalhos/Materia
user@destiny password:
teste.txt                                100%    0    0.0KB/s    00:00
```

# Secure Shell

---

- ▣ Principais máquinas do Departamento de Informática
  - ▣ macalan - servidora de uso geral
  - ▣ orval - servidora para processamento, para processos que utilizam muitos recursos

```
ssh seu_login@ssh.inf.ufpr.br
```

# Secure Shell

---

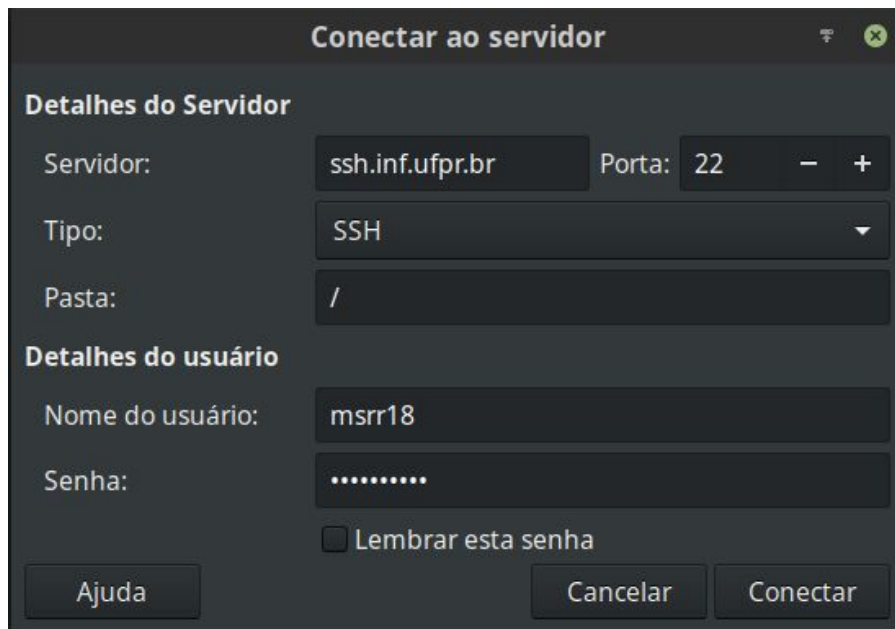
- ▣ Como logar na máquina virtual do departamento:

```
ssh seu_login@macalan.c3sl.ufpr.br
```

- ▣ Por que é importante?  
Única maneira de acessar a sua conta do departamento de modo remoto (utilizando seu computador pessoal)

# Secure Shell por interface gráfica (file system)

- ▣ Abra a Pasta Pessoal pela interface gráfica mesmo
- ▣ Arquivos → Conectar a um servidor...



**Conectar ao servidor**

**Detalhes do Servidor**

Servidor:  Porta:  - +

Tipo:  ▼

Pasta:

**Detalhes do usuário**

Nome do usuário:

Senha:

Lembrar esta senha

Técnica interessante se você está em casa e quer visualizar/navegar pela sua pasta no Dlnf, copiar e mover arquivos

# Secure Shell - Comandos

---

- ▣ **whoami** - mostra o nome do seu usuário no sistema
- ▣ **hostname** - mostra o nome do sistema
- ▣ **who** - mostra quem mais está conectado no sistema
- ▣ **finger** - mostra detalhes sobre os usuários do sistema

# Outros exemplos

# Opção -i do mv, cp, rm e outros

```
user@machine: ls -R
.:
banana  exemplo  outro_arquivo

./banana:
exemplo
user@machine: cat exemplo
user@machine: cat banana/exemplo
se tem essa linha é o original
```

```
user@machine: mv -i exemplo outro_arquivo banana/
mv: sobrescrever 'banana/exemplo'? n
user@machine: ls
banana  exemplo
user@machine: ls banana/
exemplo  outro_arquivo
user@machine: cat banana/exemplo
se tem essa linha é o original
```

```
user@machine: mv exemplo banana/
user@machine: cat banana/exemplo
```

Pede confirmação antes de sobrescrever qualquer arquivo.

# Exercícios



- Faça um amigo
- Você e ele devem estar logados em uma mesma máquina (um de vocês dá ssh na máquina do outro)
- Conversem usando who e write:
  - who
  - write [usuário] [tty/pts]
  - Digite na entrada padrão, CTRL + C / CTRL + D p/ sair
  - Para desativar mensagens, mesg [y | n]



# Exercícios

- Se junte a um colega e utilize o comando SSH para acessar a máquina dele com a conta;
- Utilize o comando **eject** e **eject -t**
- Dê SSH em outra máquina e copie um arquivo para seu computador usando o comando **scp**.



**Edigool \*9\*:**

Cuidado com as palavras, já esta rastreado.

← Responder



# Comandos Avançados

history, grep, find, less, more, head e tail

# History e visualização de dados

---

- ▣ **history**: exibe os últimos comandos executados
- ▣ **head** [arquivo]: exibe as 10 primeiras linhas do arquivo
- ▣ **tail** [arquivo]: exibe as 10 últimas linhas do arquivo

# RELEMBRANDO

## REDIRECIONAMENTO (> e >>)

---

```
user@machine:~$ echo "Este é um teste"
Este é um teste
user@machine:~$ echo "Este é outro teste" > teste.txt
user@machine:~$ cat teste.txt
Este é um teste
user@machine:~$ echo "Este é o ultimo teste" > teste.txt
user@machine:~$ cat teste.txt
Este é o ultimo teste
user@machine:~$ echo "Mentira" >> teste.txt
user@machine:~$ cat teste.txt
Este é o ultimo teste
Mentira
```

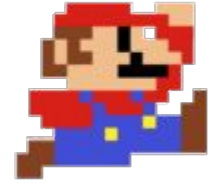
?



**Mas e se eu quero  
redirecionar a saída não  
para um arquivo, mas para  
outro comando?**

# Pipeline

---



- ▣ O caractere | (**pipeline**) faz com que a saída de um comando torne-se a entrada de outro comando

```
user@machine: ls
teste1.txt teste2.txt teste3.txt teste4.txt
user@machine: ls | cat
teste1.txt
teste2.txt
teste3.txt
teste4.txt
user@machine: ls | tac
teste4.txt
teste3.txt
teste2.txt
teste1.txt
```

# Pipeline e xargs

- ▣ Para quando o comando só funciona com argumentos
- ▣ Os dados da entrada padrão servirão de argumento para o comando utilizado com o xargs.

```
user@machine: ls
teste1.txt teste2.txt teste3.txt
user@machine: ls | rm
rm: falta operando
Tente "rm --help" para mais informações.
user@machine: ls | xargs rm
user@machine: ls
user@machine: █
```

→ não foi

→ foi

# Exercício

## origem

```
linha 1
linha 2
linha 3
linha 4
linha 5
linha 6 * ME IMPRIMA JUNTO COM A LINHA 7 *
linha 7 * ME IMPRIMA JUNTO COM A LINHA 6 *
linha 8
linha 9
linha 10
linha 11
linha 12
linha 13
linha 14 * ME IMPRIMA SOZINHO *
linha 15
```

## gabarito1

```
linha 6 * ME IMPRIMA JUNTO COM A LINHA 7 *
linha 7 * ME IMPRIMA JUNTO COM A LINHA 6 *
```

## gabarito2

```
linha 14 * ME IMPRIMA SOZINHO *
```



# Pipeline

---



- ▣ O **pipeline** dispensa o uso de um arquivo auxiliar

Sem pipeline:

```
user@machine: head -7 origem > aux.txt
user@machine: tail -2 aux.txt
linha 6 * ME IMPRIMA JUNTO COM A LINHA 7 *
linha 7 * ME IMPRIMA JUNTO COM A LINHA 6 *
```

Com pipeline:

```
user@machine: head -7 origem | tail -2
linha 6 * ME IMPRIMA JUNTO COM A LINHA 7 *
linha 7 * ME IMPRIMA JUNTO COM A LINHA 6 *
```

# Para visualizar arquivos grandes

- ▣ **less**: visualizador de dados

O comando pode receber os dados por argumento:

```
less arquivo_grande
```

Mas normalmente recebe pelo pipeline:

```
user@machine: history | less
user@machine: finger * | less
user@machine: █
```

→ em bcc/ ou ibm/

(exibe o histórico na interface less e  
exibe lista de usuários na interface less)

# Para visualizar arquivos grandes

---

- ▣ **less**, navegando pela interface:
  - “q” para sair da interface
  - “/padrão” grifa as ocorrências de “padrão”
  - “h” para listar outros comandos de navegação

Opções bacanas:

-N enumera linhas

-S vai para o lado também



# Comandos de busca

find (p/ arquivos)

grep (p/ conteúdo dentro de arquivos)

# Comandos de busca - find

---

- **find** [starting-point] [opção] [argumento]  
Procura por arquivos
- starting-point: por onde começa a procurar (se não especificado assume-se “. ”)
- exemplo de opção:
  - -iname
- argumento: depende da opção usada, pode ser nome, tipo de arquivo, permissões, datas de criação, modificação ou último acesso ao arquivo

# Comandos de busca - grep

---

- ▣ Procura por uma palavra/frase dentro de um arquivo
- ▣ Uma espécie de ctrl+f
  - ▣ **Global regular expression print**
  - ▣ **Globally look for a regular expression and print**
  - ▣ **Global regular expression parser**
- ▣ estrutura:  
**grep** [opção] [palavra] [arquivo]

Dê um man para ver as opções (-w, -n, -i)

# Comandos de busca - grep

---

## Exercício: **Encontrar o nome do colega pelo GRR ou nome**

- Relembrando: comando finger [login]
  - Com esse comando encontrávamos o GRR e nome a partir do login
- \$ finger \*
- \$ finger \* | grep [grr]
- \$ finger \* | grep -B 3 [grr]
- ou
- \$ finger \* | grep -i [nome]

# Vários comandos na mesma linha

- Com o pipeline, você pode dar vários comandos na mesma linha

```
user@machine:~$ history | less      #exibe o historico na interface less
user@machine:~$ ls | grep b | head #exibe os 10 primeiros arquivos/diretorios com a letra 'b'
user@machine:~$ ls | tail -n 5      #exibe as últimas 5 linhas ao invés das 10 últimas
user@machine:~$ history | less | grep ssh | head | tail -n 3 #exibe as últimas 3 linhas das
#primeiras 10 linhas dos últimos comandos que continham a palavra 'ssh' na interface less
user@machine:~$ find -name teste.txt #exibe o(s) diretorio(s) que contem o arquivo 'teste.txt'
user@machine:~$ find -iname TESTE.txt #exibe o(s) diretorio(s) que contem o arquivo
#'teste.txt' ignorando letras maiúsculas e minúsculas
user@machine:~$
```





# Gerenciamento de processos

ps, top, kill

# Gerenciamento de processos

---

- ▣ exibe os processos em execução:
  - ▣ **ps** de forma simplificada
  - ▣ **top** de forma detalhada
  - ▣ **htop** top colorido que suporta comandos com o mouse
  
- ▣ Encerrar o processo
  - ▣ Pelo seu PID: **kill** [id do processo]
  - ▣ Pelo seu nome: **pkill** [nome do processo]
  - ▣ **xkill**: encerra o processo usando o mouse

```
user@machine:~$ ps -ef | grep firefox #'-ef' exibe mais processos
1986 ?        Sl      7:22 /usr/lib/firefox-3.5.3/firefox
user@machine:~$ kill -9 1986 #'-9' previne que o comando seja ignorado
user@machine:~$ pkill firefox
user@machine:~$ xkill
```

# Execução de programas

---

- ▣ comando **&**, executa em segundo plano
- ▣ **ctrl + z**, para o processo
- ▣ **Ctrl + c**, fecha/cancela o processo
- ▣ **jobs**, mostra os processos parados e em segundo plano
- ▣ **fg**, põe o processo no primeiro plano

# Obrigado!

## Dúvidas?



<http://pet.inf.ufpr.br/calouros/#apoio>

pet@inf.ufpr.br