# Guia de Comandos Essenciais do Linux

**Professores:** Marcos Paulo e Nathan Cirillo

**Instituição:** Universidade Paulista UNIP - Campus Jundiaí

**Parte 1: Navegação e Manipulação de Arquivos e Diretórios**

1. ls - Listar conteúdo de diretórios
2. pwd - Exibir o diretório de trabalho atual
3. cd - Mudar de diretório
4. mkdir - Criar um novo diretório
5. rmdir - Remover um diretório vazio
6. cp - Copiar arquivos e diretórios
7. mv - Mover ou renomear arquivos e diretórios
8. rm - Remover arquivos ou diretórios
9. touch - Criar um arquivo vazio ou atualizar seu timestamp

**Parte 2: Visualização e Busca de Conteúdo em Arquivos**

10. cat - Concatenar e exibir o conteúdo de arquivos

11. head - Exibir o início de um arquivo

12. tail - Exibir o final de um arquivo

13. grep - Buscar por padrões em texto

14. less - Visualizar o conteúdo de arquivos de forma interativa (mais poderoso)

15. more - Visualizar o conteúdo de arquivos de forma paginada (mais simples)

**Parte 3: Monitoramento e Gerenciamento de Processos e Sistema**

16. lscpu - Exibir informações sobre a CPU

17. ps - Exibir informações sobre os processos atuais

18. top - Exibir processos do sistema de forma dinâmica

19. nice/renice - Alterar a prioridade de um processo

20. kill - Enviar um sinal para um processo (terminá-lo)

21. bg - Colocar um processo em segundo plano (background)

22. fg - Trazer um processo para primeiro plano (foreground)

23. nohup - Executar um comando imune a interrupções

24. strace - Rastrear chamadas de sistema e sinais

25. cat /proc/cpuinfo - Exibir informações detalhadas da CPU

26. cat /proc/meminfo - Exibir informações detalhadas da memória

### Parte 1: Navegação e Manipulação de Arquivos e Diretórios

#### 1. ls

* **Funcionalidade:** Lista os arquivos e diretórios dentro do diretório atual ou de um diretório especificado.
* **Sintaxe:** ls [opções] [diretório]
* **Exemplo de Aplicação:** Listar o conteúdo do diretório atual em formato longo, mostrando permissões, proprietário, tamanho e data.  
  $ ls -l
* **Resultado Esperado:**  
  total 24  
  drwxr-xr-x 2 aluno aluno 4096 Ago 12 10:30 Documentos  
  -rw-r--r-- 1 aluno aluno 8980 Ago 11 15:00 relatorio.pdf  
  -rwxr-xr-x 1 aluno aluno 125 Jul 30 09:00 script.sh  
  drwxr-xr-x 2 aluno aluno 4096 Ago 10 18:20 Videos

#### 2. pwd

* **Funcionalidade:** Imprime o caminho completo do diretório de trabalho atual (Print Working Directory).
* **Sintaxe:** pwd
* **Exemplo de Aplicação:**  
  $ pwd
* **Resultado Esperado:**  
  /home/aluno/Documentos

#### 3. cd

* **Funcionalidade:** Altera o diretório de trabalho atual (Change Directory).
* **Sintaxe:** cd [caminho\_do\_diretório]
* **Exemplo de Aplicação:** Mover-se para o diretório Documentos.  
  $ cd Documentos
* **Resultado Esperado:** O prompt de comando pode ser alterado para refletir o novo diretório, e o comando pwd confirmará a mudança. Nenhuma saída é exibida se o comando for bem-sucedido.

#### 4. mkdir

* **Funcionalidade:** Cria um ou mais diretórios (Make Directory).
* **Sintaxe:** mkdir [nome\_do\_diretório]
* **Exemplo de Aplicação:** Criar um diretório chamado Projetos.  
  $ mkdir Projetos
* **Resultado Esperado:** Um novo diretório chamado Projetos é criado no local atual. Nenhuma saída é exibida.

#### 5. rmdir

* **Funcionalidade:** Remove diretórios **vazios**.
* **Sintaxe:** rmdir [nome\_do\_diretório]
* **Exemplo de Aplicação:** Remover o diretório Projetos (assumindo que esteja vazio).  
  $ rmdir Projetos
* **Resultado Esperado:** O diretório é removido. Se o diretório não estiver vazio, um erro será exibido.

#### 6. cp

* **Funcionalidade:** Copia arquivos ou diretórios de uma origem para um destino.
* **Sintaxe:** cp [origem] [destino]
* **Exemplo de Aplicação:** Copiar o arquivo relatorio.txt para relatorio\_backup.txt.  
  $ cp relatorio.txt relatorio\_backup.txt
* **Resultado Esperado:** Um novo arquivo, relatorio\_backup.txt, é criado com o mesmo conteúdo de relatorio.txt.

#### 7. mv

* **Funcionalidade:** Move ou renomeia arquivos e diretórios.
* **Sintaxe:** mv [origem] [destino]
* **Exemplo de Aplicação (Renomear):**  
  $ mv relatorio\_antigo.txt relatorio\_novo.txt
* **Exemplo de Aplicação (Mover):**  
  $ mv relatorio\_novo.txt ./Documentos/
* **Resultado Esperado:** No primeiro caso, o arquivo é renomeado. No segundo, o arquivo é movido para o diretório Documentos.

#### 8. rm

* **Funcionalidade:** Remove (apaga) arquivos. Com a opção -r, remove diretórios e todo o seu conteúdo.
* **Sintaxe:** rm [opções] [arquivo/diretório]
* **Exemplo de Aplicação:** Remover o arquivo arquivo\_temporario.txt.  
  $ rm arquivo\_temporario.txt
* **Resultado Esperado:** O arquivo é permanentemente apagado. **Atenção:** Este comando é irreversível. Use com extremo cuidado.

#### 9. touch

* **Funcionalidade:** Cria um arquivo vazio se ele não existir. Se o arquivo já existir, atualiza sua data de modificação (timestamp) para a data e hora atuais.
* **Sintaxe:** touch [nome\_do\_arquivo]
* **Exemplo de Aplicação:**  
  $ touch novo\_relatorio.docx
* **Resultado Esperado:** Um arquivo vazio chamado novo\_relatorio.docx é criado.

### Parte 2: Visualização e Busca de Conteúdo em Arquivos

#### 10. cat

* **Funcionalidade:** Concatena e exibe o conteúdo de um ou mais arquivos na saída padrão (geralmente o terminal).
* **Sintaxe:** cat [arquivo]
* **Exemplo de Aplicação:** Exibir o conteúdo do arquivo config.txt.  
  $ cat config.txt
* **Resultado Esperado:**  
  USER=admin  
  HOST=192.168.1.1  
  PORT=8080

#### 11. head

* **Funcionalidade:** Exibe as primeiras 10 linhas de um arquivo por padrão.
* **Sintaxe:** head [opções] [arquivo]
* **Exemplo de Aplicação:** Exibir as 5 primeiras linhas de um arquivo de log.  
  $ head -n 5 system.log
* **Resultado Esperado:**  
  [2023-08-12 10:00:01] INFO: System startup initiated.  
  [2023-08-12 10:00:02] DEBUG: Loading kernel modules.  
  [2023-08-12 10:00:03] DEBUG: Mounting filesystems.  
  [2023-08-12 10:00:04] INFO: Network service started.  
  [2023-08-12 10:00:05] WARN: Deprecated module found.

#### 12. tail

* **Funcionalidade:** Exibe as últimas 10 linhas de um arquivo por padrão. Muito útil para ver logs em tempo real com a opção -f.
* **Sintaxe:** tail [opções] [arquivo]
* **Exemplo de Aplicação:** Ver as últimas 3 linhas de um arquivo de log.  
  $ tail -n 3 system.log
* **Resultado Esperado:**  
  [2023-08-12 11:30:15] INFO: User 'aluno' logged in.  
  [2023-08-12 11:32:00] ERROR: Failed to connect to database.  
  [2023-08-12 11:32:05] INFO: Shutting down service.

#### 13. grep

* **Funcionalidade:** Busca por linhas que contêm um determinado padrão (texto ou expressão regular) dentro de arquivos.
* **Sintaxe:** grep [padrão] [arquivo]
* **Exemplo de Aplicação:** Encontrar todas as linhas que contêm a palavra "ERROR" no arquivo system.log.  
  $ grep "ERROR" system.log
* **Resultado Esperado:**  
  [2023-08-12 10:15:45] ERROR: Authentication failed for user 'guest'.  
  [2023-08-12 11:32:00] ERROR: Failed to connect to database.

#### 14. less

* **Funcionalidade:** Permite visualizar o conteúdo de arquivos de forma interativa. É mais poderoso que o more, permitindo rolar para cima e para baixo, buscar texto, etc.
* **Sintaxe:** less [arquivo]
* **Exemplo de Aplicação:**  
  $ less system.log
* **Resultado Esperado:** O terminal é limpo e o conteúdo do arquivo é exibido, começando do topo. O usuário pode usar as setas para navegar, / para buscar e q para sair.

#### 15. more

* **Funcionalidade:** Permite visualizar o conteúdo de arquivos uma página (uma tela) por vez.
* **Sintaxe:** more [arquivo]
* **Exemplo de Aplicação:**  
  $ more system.log
* **Resultado Esperado:** O terminal exibe a primeira "página" do arquivo. O usuário pode pressionar Espaço para avançar para a próxima página ou q para sair.

### Parte 3: Monitoramento e Gerenciamento de Processos e Sistema

#### 16. lscpu

* **Funcionalidade:** Exibe informações sobre a arquitetura da CPU, como número de cores, threads, caches, etc.
* **Sintaxe:** lscpu
* **Exemplo de Aplicação:**  
  $ lscpu
* **Resultado Esperado (Exemplo Abreviado):**  
  Architecture: x86\_64  
  CPU op-mode(s): 32-bit, 64-bit  
  Byte Order: Little Endian  
  CPU(s): 8  
  On-line CPU(s) list: 0-7  
  Thread(s) per core: 2  
  Core(s) per socket: 4  
  Socket(s): 1  
  Vendor ID: GenuineIntel  
  Model name: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz

#### 17. ps

* **Funcionalidade:** Mostra uma "fotografia" dos processos que estão em execução no momento.
* **Sintaxe:** ps [opções]
* **Exemplo de Aplicação:** Listar todos os processos em execução no sistema em formato detalhado.  
  $ ps aux
* **Resultado Esperado (Exemplo Abreviado):**  
  USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND  
  root 1 0.0 0.1 167620 11380 ? Ss 10:00 0:02 /sbin/init  
  root 2 0.0 0.0 0 0 ? S 10:00 0:00 [kthreadd]  
  aluno 1234 0.5 2.5 154231 45012 ? Sl 10:15 0:35 /usr/bin/firefox  
  aluno 1567 0.0 0.2 234567 12345 pts/0 Ss 11:00 0:01 /bin/bash  
  aluno 1890 0.0 0.1 87654 8765 pts/0 R+ 11:05 0:00 ps aux

#### 18. top

* **Funcionalidade:** Fornece uma visão dinâmica e em tempo real dos processos em execução. Mostra uso de CPU, memória e outras estatísticas vitais.
* **Sintaxe:** top
* **Exemplo de Aplicação:**  
  $ top
* **Resultado Esperado:** O terminal é limpo e uma tela interativa é exibida, atualizando-se a cada poucos segundos. Mostra um resumo do sistema no topo e uma lista de processos ordenados por uso de CPU abaixo. Pressione q para sair.

#### 19. nice / renice

* **Funcionalidade:** nice executa um comando com uma prioridade de agendamento alterada. renice altera a prioridade de um processo já em execução. A prioridade vai de -20 (mais alta) a 19 (mais baixa).
* **Sintaxe:** nice -n [valor] [comando] / renice [valor] -p [PID]
* **Exemplo de Aplicação:** Aumentar a prioridade (valor negativo) de um processo com PID 1234.  
  $ renice -5 -p 1234
* **Resultado Esperado:**  
  1234 (process ID) old priority 0, new priority -5

#### 20. kill

* **Funcionalidade:** Envia um sinal a um processo, geralmente para terminá-lo.
* **Sintaxe:** kill [opções] [PID]
* **Exemplo de Aplicação:** Terminar de forma "gentil" o processo com PID 3456 (sinal TERM).  
  $ kill 3456
* **Exemplo de Aplicação (Forçar):** Terminar à força o processo com PID 3456 (sinal KILL).  
  $ kill -9 3456
* **Resultado Esperado:** O processo é terminado. Nenhuma saída é exibida se for bem-sucedido. kill -9 deve ser usado como último recurso.

#### 21. bg

* **Funcionalidade:** Coloca um processo que foi parado (geralmente com Ctrl+Z) para continuar sua execução em segundo plano (background).
* **Sintaxe:** bg %[job\_id]
* **Exemplo de Aplicação:**  
  $ sleep 100 # Executa um comando  
  # Pressione Ctrl+Z para parar  
  [1]+ Stopped sleep 100  
  $ bg %1
* **Resultado Esperado:**  
  [1]+ sleep 100 &  
    
  O processo agora continua executando em segundo plano.

#### 22. fg

* **Funcionalidade:** Traz um processo que está em segundo plano (background) ou parado para o primeiro plano (foreground).
* **Sintaxe:** fg %[job\_id]
* **Exemplo de Aplicação:**  
  $ fg %1
* **Resultado Esperado:**  
  sleep 100  
    
  O comando volta a ocupar o terminal, como se tivesse sido executado normalmente.

#### 23. nohup

* **Funcionalidade:** Executa um comando de forma que ele continue rodando mesmo que o usuário se desconecte do terminal (No Hang Up). A saída é geralmente redirecionada para um arquivo nohup.out.
* **Sintaxe:** nohup [comando] &
* **Exemplo de Aplicação:** Executar um script demorado em segundo plano.  
  $ nohup ./meu\_script\_longo.sh &
* **Resultado Esperado:**  
  [1] 5678  
  nohup: ignoring input and appending output to 'nohup.out'  
    
  O processo continuará executando mesmo após fechar o terminal.

#### 24. strace

* **Funcionalidade:** Uma ferramenta de depuração poderosa que intercepta e registra as chamadas de sistema (interações do processo com o kernel) e os sinais recebidos por um processo.
* **Sintaxe:** strace [comando]
* **Exemplo de Aplicação:** Rastrear as chamadas de sistema do comando ls.  
  $ strace ls
* **Resultado Esperado (Exemplo Abreviado e Intenso):**  
  execve("/bin/ls", ["ls"], 0x7ffc... /\* 55 vars \*/) = 0  
  brk(NULL) = 0x55c...  
  access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)  
  openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3  
  ... (muitas outras linhas) ...  
  write(1, "Documentos\nrelatorio.pdf\n", 26) = 26  
  ...  
  exit\_group(0) = ?  
  +++ exited with 0 +++

#### 25. cat /proc/cpuinfo

* **Funcionalidade:** Não é um comando, mas a leitura de um arquivo virtual do sistema de arquivos /proc que contém informações detalhadas sobre cada processador do sistema.
* **Sintaxe:** cat /proc/cpuinfo
* **Exemplo de Aplicação:**  
  $ cat /proc/cpuinfo
* **Resultado Esperado (Exemplo para um core):**  
  processor : 0  
  vendor\_id : GenuineIntel  
  cpu family : 6  
  model : 158  
  model name : Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz  
  stepping : 10  
  microcode : 0xf0  
  cpu MHz : 2600.000  
  cache size : 12288 KB  
  ... (e muitas outras informações)

#### 26. cat /proc/meminfo

* **Funcionalidade:** Leitura do arquivo virtual /proc/meminfo, que fornece um relatório detalhado sobre o uso da memória RAM e swap do sistema.
* **Sintaxe:** cat /proc/meminfo
* **Exemplo de Aplicação:**  
  $ cat /proc/meminfo
* **Resultado Esperado (Exemplo Abreviado):**  
  MemTotal: 16342580 kB  
  MemFree: 8450232 kB  
  MemAvailable: 12534876 kB  
  Buffers: 345678 kB  
  Cached: 4123456 kB  
  SwapTotal: 8396796 kB  
  SwapFree: 8396796 kB  
  ... (e muitas outras informações)

**Sugestões para aprofundamento:**

* Use o comando man (ex: man ls) para ler o manual completo de qualquer comando.
* Combine comandos usando pipes (|). Ex: ps aux | grep firefox.
* Explore as diferentes opções que cada comando oferece.