Laboratório de Sistemas Operacionais (ECOS11A)

Prof Otávio Gomes (otavio.gomes@unifei.edu.br)

Roteiro 01

Terminal Linux – Comandos básicos

1) Ambientação

Neste roteiro de laboratório, vamos explorar uma variedade de comandos fundamentais do terminal Linux, essenciais para qualquer usuário do sistema. Esses comandos desempenham papéis importantes na navegação pelo sistema de arquivos, na manipulação e visualização de arquivos, na administração do sistema e na automação de tarefas.

- **Navegação no Sistema de Arquivos:**
  pwd: Exibe o diretório atual.
  cd: Navega entre os diretórios.
  ls: Lista o conteúdo do diretório atual.
  man: Exibe o manual de um comando específico.

- **Manipulação e Visualização de Arquivos:**
  cat: Concatena e exibe o conteúdo de arquivos.
  head: Exibe as primeiras linhas de um arquivo.
  tail: Exibe as últimas linhas de um arquivo.
  wc: Conta o número de linhas, palavras e caracteres em um arquivo.
  touch: Cria um arquivo vazio.
  tac: Concatena e exibe o conteúdo de arquivos de forma reversa.

- **Administração do Sistema:**
  free: Exibe informações sobre a memória do sistema.
  cal: Exibe um calendário.
  sort: Ordena as linhas de um arquivo.
  tr: Traduz ou deleta caracteres.
  paste: Combina linhas de arquivos de texto.
  diff: Compara dois arquivos linha por linha.
  cmp: Compara dois arquivos byte por byte.

- **Automatização de Tarefas:**
  grep: Procura por padrões em arquivos ou saída de comando.
  cut: Extrai seções de linhas de arquivos ou saída de comando.

1.1) Comando man (Manual)

O comando **man** é uma abreviação de "manual" e é usado para acessar a documentação detalhada dos comandos do Linux. Ele fornece informações completas sobre a sintaxe, opções, funcionalidades e exemplos de uso de cada comando.

Uso Básico: Para acessar o manual de um comando específico, você simplesmente digita ***man*** seguido pelo nome do comando desejado.
Exemplo: **man ls**

Navegação no Manual: O manual é exibido em uma interface de texto paginada. Você pode navegar pelo manual usando as teclas de seta para cima

e para baixo para rolar, e as teclas de página para cima e para baixo para avançar ou retroceder páginas.

Busca no Manual: Para pesquisar palavras-chave específicas dentro do manual, você pode pressionar / seguido da palavra-chave e depois Enter. Isso irá realçar todas as ocorrências da palavra-chave.

1.2) Comando help

O comando **help** é uma funcionalidade embutida em muitos comandos do ***shell bash***, que fornece uma breve descrição das opções disponíveis e da sintaxe básica do comando.

Uso Básico: Para acessar a ajuda embutida de um comando, você simplesmente digita o nome do comando seguido por --help.
Exemplo: **ls –help**

Ajuda Rápida: O help geralmente fornece uma saída mais curta e simplificada em comparação com o **man**, ideal para uma visão geral rápida do comando. Ele geralmente lista as opções disponíveis e uma breve descrição de cada uma delas.

Limitações: É importante notar que nem todos os comandos suportam a opção ***--help***. Isso depende de como o comando foi implementado e se o desenvolvedor optou por incluir essa funcionalidade.

1.3) Comando cd

No terminal, o comando **cd** é usado para mudar o diretório atual. Junto com o comando cd**,** você pode definir diferentes instruções para navegar pelos diretórios de forma eficiente.

Navegar para um diretório específico: cd <caminho_do_diretório>

Substitua <*caminho_do_diretório*> pelo caminho absoluto ou relativo do diretório para o qual você deseja navegar.

Voltar para o diretório pai: cd **..**

Este comando irá mover o diretório atual para o diretório pai.

Navegar para o diretório anterior (bash): cd **-**

Este comando irá mover o diretório atual para o diretório anterior que você estava antes.

Navegar para um diretório usando atalhos (bash): cd **~**<usuário>

Substitua <usuário> pelo nome de usuário do diretório para o qual você deseja navegar. Isso é útil quando você deseja acessar o diretório home de outro usuário.

Navegar para um diretório usando variáveis de ambiente: cd **$VARIAVEL**

Substitua $VARIAVEL pelo nome de uma variável de ambiente que contém o caminho do diretório para o qual você deseja navegar.

1.4) Variáveis de ambiente

As variáveis de ambiente são variáveis associadas ao sistema operacional ou à sessão de um usuário que armazenam informações que podem ser acessadas pelos processos em execução.

Elas são amplamente utilizadas no Linux e em outros sistemas Unix-like para configurar o comportamento do sistema e dos programas. Existem dois tipos principais de variáveis de ambiente:

- **Variáveis de ambiente do sistema:** São variáveis definidas globalmente para todo o sistema operacional. Elas são definidas no momento da inicialização do sistema e permanecem disponíveis durante toda a sessão do usuário. Exemplos comuns incluem variáveis como ***PATH***, que especifica onde o sistema deve procurar por comandos executáveis, e ***HOME***, que indica o diretório home do usuário.
- **Variáveis de ambiente do usuário:** São variáveis definidas para uma sessão de usuário específica. Elas podem ser definidas manualmente pelo usuário ou configuradas por scripts de inicialização do shell. Essas variáveis permanecem disponíveis apenas durante a sessão de login do usuário. Exemplos incluem variáveis para especificar o editor de texto padrão, o diretório de trabalho padrão e as preferências de exibição do terminal.

Para utilizar variáveis de ambiente no Linux, você pode seguir estes passos:

- **Visualizar variáveis de ambiente existentes:** Para visualizar todas as variáveis de ambiente disponíveis, você pode usar o comando **env** ou **printenv** no terminal.
- **Atribuir valor a uma variável de ambiente:** Para atribuir um valor a uma variável de ambiente, você pode usar a sintaxe VARIAVEL=valor. Por exemplo, para definir a variável NOME com o valor "João". Após atribuir um valor a uma variável de ambiente, ela estará disponível para uso durante a sessão de usuário atual.
- **Acessar o valor de uma variável de ambiente:** Para acessar o valor de uma variável de ambiente, você pode usar a sintaxe $VARIAVEL. Por exemplo, para imprimir o valor da variável NOME, você pode fazer: **echo $NOME**
- **Persistir as variáveis de ambiente:** Para fazer com que as variáveis de ambiente persistam entre sessões de login, você pode definir as variáveis em um arquivo de inicialização do shell, como **~/.bashrc** ou **~/.bash_profile**. Isso fará com que as variáveis sejam configuradas automaticamente toda vez que você iniciar uma nova sessão de terminal.

1.5) Opções de Shell

Além do bash, que é o shell padrão mais comumente utilizado em sistemas Linux, existem várias outras opções de shell disponíveis. Algumas das opções mais populares incluem:

- **Bourne Shell (sh):** O Bourne Shell é um dos shells originais do Unix. Ele foi desenvolvido por Stephen Bourne na década de 1970 e é amplamente utilizado como o shell padrão em muitos sistemas Unix e Linux. O bash é uma extensão do Bourne Shell, mantendo compatibilidade com a maioria dos scripts escritos para o Bourne Shell.

- **C Shell (csh):** O C Shell é outro shell clássico do Unix, desenvolvido na década de 1970 na Universidade da Califórnia, Berkeley. Ele possui uma sintaxe semelhante à linguagem de programação C e oferece recursos como histórico de comandos e expansão de variáveis. O **tcsh** é uma versão melhorada do C Shell, com recursos adicionais.

- **Korn Shell (ksh):** O Korn Shell é um shell derivado do Bourne Shell, desenvolvido por David Korn na década de 1980. Ele oferece muitos recursos avançados, incluindo manipulação de strings, expressões regulares e construções de controle de fluxo mais poderosas do que o Bourne Shell padrão.

- **Z Shell (zsh):** O Z Shell é um shell de propósito geral que visa combinar recursos do bash, ksh e tcsh. Ele oferece uma série de recursos avançados, como preenchimento de comandos, temas e plug-ins, tornando-o popular entre os usuários avançados do Unix e Linux.

2) Navegação no sistema de arquivos

a) Utilize o comando pwd (Print Working Directory):para exibir o diretório atual.

- Abra o terminal.
- Digite **pwd** e pressione **Enter**.
- Observe o caminho completo do diretório atual que será exibido como saída.

b) Utilize o comando ls (List) para listar o conteúdo do diretório atual.

- Digite **ls** e pressione **Enter**.
- Observe os arquivos e diretórios listados como saída do comando.

c) Utilize o comando cd (Change Directory) para navegar entre os diretórios.

- Escolha um dos diretórios listados pelo comando ls.
- Digite **cd *<nome_do_diretório>*** e pressione **Enter**.
- Utilize novamente o comando **pwd** para confirmar se você mudou para o diretório desejado.

d) Utilize o comando man para exibir o manual de um comando específico.

- Escolha um comando (por exemplo, ls).
- Digite man <nome_do_comando> e pressione Enter.
- Leia o manual para entender o funcionamento do comando e suas opções.

3) Manipulação e Visualização de Arquivos

a) Utilize o comando touch para criar um novo arquivo vazio.

- Abra o terminal.
- Digite touch arquivo.txt e pressione Enter.
- Verifique se o arquivo foi criado com sucesso.

b) Utilize o comando cat para exibir o conteúdo do arquivo criado anteriormente.

- Digite cat arquivo.txt e pressione Enter.
- Observe o conteúdo do arquivo sendo exibido como saída do comando.

c) Utilize o comando head para exibir as primeiras linhas do arquivo.

- Digite head arquivo.txt e pressione Enter.
- Observe as primeiras linhas do arquivo sendo exibidas como saída do comando.

d) Utilize o comando tail para exibir as últimas linhas do arquivo.

- Digite tail arquivo.txt e pressione Enter.
- Observe as últimas linhas do arquivo sendo exibidas como saída do comando.

e) Utilize o comando wc para contar o número de linhas, palavras e caracteres no arquivo.

- Digite wc arquivo.txt e pressione Enter.
- Observe a contagem de linhas, palavras e caracteres sendo exibida como saída do comando.

f) Utilize o comando tac para concatenar e exibir o conteúdo do arquivo em ordem reversa.

- Digite tac arquivo.txt e pressione Enter.

- Observe o conteúdo do arquivo sendo exibido em ordem reversa como saída do comando.

4) Administração do Sistema

a) Utilize o comando free para exibir informações sobre a memória do sistema.

- Abra o terminal.
- Digite free e pressione Enter.
- Observe as informações sobre a memória física e virtual do sistema sendo exibidas como saída do comando.

b) Utilize o comando cal para exibir um calendário do mês atual.

- Digite cal e pressione Enter.
- Observe o calendário do mês atual sendo exibido como saída do comando.

c) Utilize o comando sort para ordenar as linhas de um arquivo.

- Crie um arquivo de texto com algumas linhas desordenadas.
- Digite sort arquivo.txt e pressione Enter.
- Observe as linhas do arquivo sendo ordenadas alfabeticamente como saída do comando.

d) Utilize o comando tr para traduzir ou deletar caracteres em um arquivo.

- Crie um arquivo de texto com algumas palavras.
- Digite tr 'a' 'A' < arquivo.txt e pressione Enter.
- Observe as letras minúsculas 'a' sendo traduzidas para maiúsculas 'A' como saída do comando.

e) Utilize o comando paste para combinar as linhas de dois arquivos de texto.

- Crie dois arquivos de texto com o mesmo número de linhas.

- Digite paste arquivo1.txt arquivo2.txt e pressione Enter.
- Observe as linhas dos arquivos sendo combinadas lado a lado como saída do comando.

f) Utilize o comando diff para comparar dois arquivos linha por linha.

- Crie dois arquivos de texto com conteúdos diferentes.
- Digite diff arquivo1.txt arquivo2.txt e pressione Enter.
- Observe as diferenças entre os arquivos sendo exibidas como saída do comando.

g) Utilize o comando cmp para comparar dois arquivos byte por byte.

- Crie dois arquivos de texto com conteúdos diferentes.
- Digite cmp arquivo1.txt arquivo2.txt e pressione Enter.
- Observe se há diferenças entre os arquivos. Se não houver, não será exibida nenhuma saída.

5) **Prática 01** – Analise o resultado de cada sequência.

$ clear <Enter>

$ free > arq01.txt <Enter>

$ cat arq01.txt <Enter>

$ cal > arq01.txt <Enter>

$ cat arq01.txt <Enter>

$ cal >> arq01.txt <Enter>

$ cat arq01.txt <Enter>

Prof Otávio Gomes (otavio.gomes@unifei.edu.br)

6) Prática 02 – Revisão dos comandos

Atividade 1: Explorando o Sistema de Arquivos

Utilize o comando pwd para descobrir em qual diretório você está atualmente.

Utilize o comando ls para listar o conteúdo do diretório atual.

Utilize o comando cd para navegar para um diretório de sua escolha.

Utilize o comando mkdir para criar um novo diretório dentro do diretório atual.

Utilize o comando touch para criar um novo arquivo vazio dentro do diretório atual.

Atividade 2: Manipulação de Arquivos e Diretórios

Utilize o comando cp para copiar um arquivo para um novo local.

Utilize o comando mv para mover um arquivo para um novo local.

Utilize o comando rm para excluir um arquivo.

Utilize o comando rmdir para excluir um diretório vazio.

Atividade 3: Visualização e Manipulação de Arquivos

Utilize o comando cat para visualizar o conteúdo de um arquivo.

Utilize os comandos head e tail para visualizar as primeiras e últimas linhas de um arquivo, respectivamente.

Utilize o comando wc para contar o número de linhas, palavras e caracteres de um arquivo.

Atividade 4: Criação e Execução de Shell Scripts

Utilize um editor de texto (como o nano ou vi) para criar um novo arquivo com extensão .sh. Dentro do arquivo, escreva um script simples que exiba uma mensagem na tela, por exemplo:

```
#!/bin/bash
echo "Olá, mundo!"
```

Salve o arquivo e dê permissão de execução utilizando o comando

```
chmod +x <nome_do_arquivo.sh>.
```

Execute o script digitando ./<nome_do_arquivo.sh> e pressione Enter.

7) **Prática 03** – Execução de scripts

Leia, entenda e, depois, execute o script *ECOS11A_Rot01.1.sh* fornecido pelo professor.

Atenção: Utilizando seu Windows, você pode ter desenvolvido seu script para resolver algum problema, mas quando foi executá-lo no Linux, foi agraciado com o erro: */bin/bash^M: bad interpreter: No such file or directory*.

O problema ocorre devido as diferenças no registro da quebra de linha feito pelo Windows e pelo Linux. Para resolver, vamos utilizar uma expressão regular e o comando SED:

sed -i -e 's/\r$//' meu_script.sh

Explicando rapidamente o que este comando faz:

sed: nome do programa que estamos utilizando (sed = Stream Editor);

-e <expressão regular>: Adiciona um comando a ser executado no script. Pode ser utilizado varias vezes, para adicionar multiplos comandos;

-i: Salva as alterações do arquivo. Gera um backup automatico se você fornecer uma extensão;

‘s/\r$//’: Remove os caracteres de Carriage Return (\r) que estão no arquivo, deixando apenas os Line Feed (\n). O que vai resolver o problema.

O caractere Carriage Return (\r) é um caractere de controle que faz o cursor retroceder ao início da linha atual em muitos sistemas de computador. Aqui estão algumas implementações em diferentes sistemas operacionais:

- Unix e Linux: o caractere \r é interpretado como um retorno de carro, que faz o cursor retroceder ao início da linha atual, mas não move o cursor para a próxima linha. Isso é usado principalmente em arquivos de texto e scripts para controlar a formatação do texto.
- Windows: Em sistemas Windows, o retorno de carro (\r) é normalmente seguido por uma quebra de linha (\n) para indicar o final de uma linha de texto. Juntos, \r\n são usados para representar uma nova linha em arquivos de texto.
- macOS (macOS X e posteriores): Em sistemas macOS X e posteriores, o padrão Unix é seguido, onde \n é usado para representar uma nova linha em arquivos de texto.

8) **Prática 04** – Execução de scripts

Leia, entenda e, depois, execute o script *ECOS11A_Rot01.2.sh* fornecido pelo professor.

9) **Prática 05** – Execução de scripts

Leia, entenda e, depois, execute o script *ECOS11A_Rot01.3.sh* fornecido pelo professor.