Bash

O poder da linha de comando!

Tchelinux.org - 2017

Sobre o palestrante

- Membro do Tchelinux desde 2009
- Servidor público federal desde 1999
- 24 anos de TI, passando por várias áreas,
- Inclusive **SysAdmin** entre 2001 e 2006 (o que atende ao foco desta palestra)
- A graduação ficou no meio do caminho (faltou $),
- Mas os cursos em algumas tecnologias continuaram…
- Cursando Téc. em Informática p/ Internet, IFSul Santana do Livramento
- Estudar, sempre! (mesmo que seja noutra área, não pare nunca…)

Agenda

1. **Bash, o que é afinal?**
2. **Primeiros comandos**
3. **Oriente-se: Prompt, /home, pastas e PATH**
4. **Recursos e funcionalidades**
5. **Escrevendo scripts shell**
6. **Agendando um script** (resumidamente)

1ª hora: Estes Slides

2ª parte: Scripting, direto no Terminal

Parte 1
O que é o Bash?

O que é o Bash?

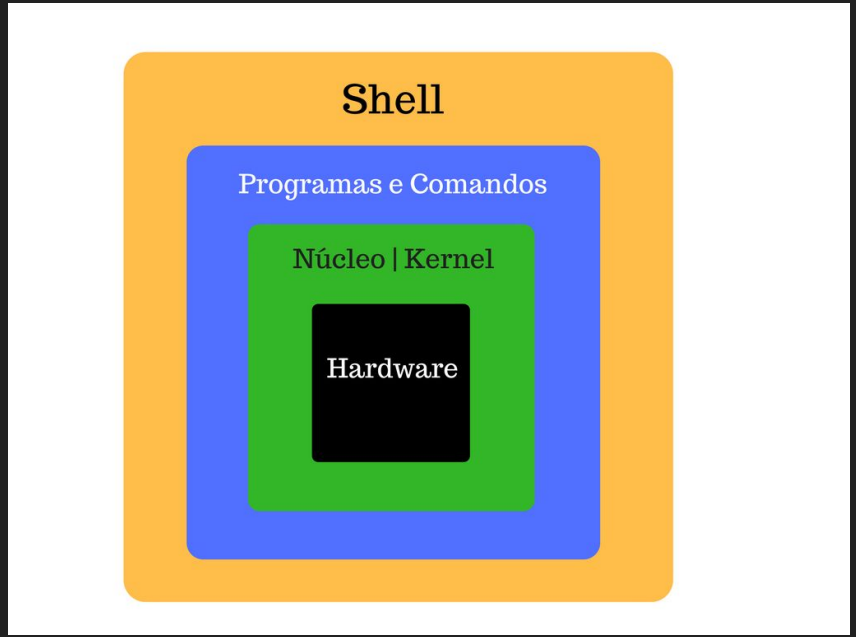
- **B**ourne-**a**gain **sh**ell
- Shell - Interpretador de comandos
- Criado por Brian Fox (Projeto GNU) em 1989
- Shell mais utilizado em distribuições Linux

O que é Shell?

O shell é o "prompt" da linha de comando do Unix e Linux, que recebe os comandos digitados pelo usuário e os executa.

O shell é aquele que aparece logo após digitar-se a senha do usuário e entrar na tela preta.

Ou na interface gráfica, ao clicar no ícone do Terminal, Xterm ou Console.



Principais Shells

Bourne Shell

– Shell padrão do UNIX.

– Escrito por Stephen Bourne da Bell Labs, também chamado de Standard Shell

– Representação para UNIX sh

Principais Shells

Bourne-Again Shell

– Shell padrão do Linux

– 100 % compatível com o Bourne Shell, com comandos adicionais para Korn Shell e C Shell.

– Criado por Brian Fox (Projeto GNU) em 1989

– Representação para Linux é **bash**

Principais Shells

Korn Shell

– Desenvolvido por David Korn da Bell Labs da AT&T.

– Atualização do Bourne Shell, com maior número de comandos

– Representação para o UNIX é **ksh**

Principais Shells

C Shell

– Desenvolvido por Bill Joy da Berkeley University

– Mais usado nos BSD

Parte 2
Primeiros comandos

Comandos ls, pwd e cd...

$ ls	(lista arquivos e pastas de um diretório/pasta)
$ ls -l	
$ pwd	(mostra a pasta/diretório atual)
$ cd Downloads	(muda de pasta/diretório)
$ cd ..	(muda para o diretório/pasta pai ou superior)

Mais alguns comandos...

$ cat arquivo (mostra o conteúdo do arquivo)

$ head arquivo (mostra as 10 primeiras linhas de do arquivo)

$ tail arquivo (mostra as 10 últimas linhas de do arquivo)

$ cp arquivo1 arquivo2 (copia um arquivo)

$ mv arquivo1 arquivo2 (renomeia um arquivo, ou move, se mudar a pasta)

Parte 3
Oriente-se:
Prompt, /home, pastas e o PATH

AVISO - O prompt do ROOT

Prompt da linha de comando de um usuário normal: $

meunome@meuhost:~$

Prompt da linha de comando de um ROOT: # (ou sudo)

meunome@meuhost:~# (muita atenção, evite usar este prompt)

meunome@meuhost:~$ sudo comando (o sudo equivale ao #)

/home/meunome

meunome@meuhost:~$ (aqui o “~” representa a pasta home do usuario)

meunome@meuhost:~$ pwd (“pwd” mostra a pasta onde estamos)

/home/meunome

Pastas e o PATH

PATH = caminho

Há muitas pastas no seu host, mas apenas algumas são pesquisadas quando digitamos um comando.

Estas pastas constam da variável PATH

$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin

Parte 4
Recursos e Funcionalidades

Recursos e Funcionalidades

- Para linha de comando
- Para scripting

Recursos para linha de comando

- Auto-complementação
- Comandos embutidos/encadeados (já em linguagem de script)
- Suporte a variáveis
- Controle de fluxo
- Controle de loops
- Redirecionamento de Saídas
- Pedindo ajuda

Recurso de Auto-complementação

Ao digitar o nome do arquivo, com um TAB o shell completa automaticamente:

$ cat arq<TAB>

$ cat arquivo

E com caracteres especiais também:

$ cat arq* (lista todos arquivos que iniciam com “arq”)

(há opções de máscaras de busca, mas * é a mais utilizada)

Comandos embutidos/encadeados

Seu primeiro “tubo”, teste o seguinte na pasta Home do seu usuario:

$ ls | grep Down

Downloads (ele listará apenas esta pasta)

Use o “|” (pipe, “tubo” em português) para conectar comandos.

Variáveis

$ NOME=Sandro

$ echo $NOME

$ NOME=”${NOME} Custodio”

$ echo $NOME

$ unset NOME

$ echo $NOME

Controle de Fluxo

```
If CONDICAO

then

        comando

else

        comando

fi
```

```
case "$1" in

1)  echo "Texto 1"
    ;;
2)  echo  "Texto 2"
    ;;
*) echo "Outras opções..."
   ;;
esac
```

Controle de Loops

```
while TESTE
do
    comandos
done
```

```
for i in 1 2 3 4
do
    comandos
done
```

```
for i in $( ls ); do
    echo item: $i
done
```

Redirecionamento de Saídas

#	descrição
0	Standard Input / Entrada Padrão
1	Standard Output / Saída Padrão
2	Output Error / Saída de Erro

```
$ ls arq > saida_normal.txt 2> saida_de_erro.txt
```

Pedindo ajuda

Documentação do próprio sistema:

$ comando --help (ajuda resumida)

$ info arquivo (mostra o tipo de arquivo)

$ man comando (manual oficial do comando, acostume-se com o “man”)

Parte 5
Escrevendo scripts shell

Scripting

- Recursos da linha de comando: Comandos encadeados, variáveis, fluxo, loops, etc.
- Editando o script (1º script e editores disponíveis)
- Executando o script (chmod e caminho)
- Parâmetros
- Funções
- Parâmetros em funções
- Agendamento/automação

Escolhendo o Editor

O vi e nano são os editores padrão que vem instalados em todas as distros. Se estiver instalado, o vim é outra opção.

vi - editor original dos Unix existe até o hoje no Linux, é muito peculiar e exige um curso rápido para que possa ser minimamente utilizado. Os comandos de menu são acionados por “:”. ($ vi nome_do_script)

nano - edição intuitiva, melhor para iniciantes, com comandos de menu baseados em combinações do Control (^tecla) ($ nano nome_do_script)

vim - vi improved, basicamente, o vi melhorado ($ vim nome_do_script)

Estrutura de um script

#!/bin/bash (1ª linha, o interpretador de comandos)

este é um comentário

comando1

comando2

Executando o script

$ nome_do_script

nome_do_script: command not found

Para o erro acima, Indique a pasta local com “./”:

$ ./nome_do_script

Executando o script

$ ./nome_do_script

bash: ./nome_do_script: Permission denied

Solução para o erro acima:

$ chmod +x nome_do_script

Parâmetros

$ ./meuscript 1500 “Pedro Alvares Cabral”

Script:

```
#!/bin/bash

echo O Brasil foi descoberto em $1 por $2.
```

Saída: O Brasil foi descoberto em 1500 por Pedro Alvares Cabral.

Funções

Código mais limpo

Reaproveitamento de código

Melhor organização e manutenção do código

```
function bla {
  comandos
}
```

Parâmetros em Funções

```
function mostra {
  echo $2 $3 $1
}
mostra 1500 “O Brasil foi descoberto por” “Pedro Alvares Cabral em”
```

Ponderações...

- Comandos do Linux podem ser usados em scripts
  - ls, cd, mkdir, rmdir, ps, df, more, less, cat, date, cut, grep, sed, awk, etc.
- Praticamente todos os comandos não-interativos podem ser usados
- Encadeamento de comandos através de pipes
- Outros scripts e programas podem ser chamados por scripts
- Agendamento/automação (crontab)

Parte 6
Agendamento/automação

Agendamento/automação (bem resumidamente)

O que é a crontab? CRON é o Serviço de Agendamento de execução de scripts e programas. Vem de cronológico, de “tempo”.

man crontab

crontab -l

crontab -e

Dúvidas?

Sugiro que fiquem para a segunda parte:

Scripting, fazendo scripts.

Sandro Custódio

sandrocustodio.blogspot.com
motofronteira.blogspot.com
www.linkedin.com/in/sandro-custodio

sandrocustodio@gmail.com
+55 55 99935 4273

Agradecimento a Leonardo Vaz, colega do Tchelinux, que iniciou a edição desta apresentação e ajudou na pesquisa de material.