TREINAMENTO LINUX

Curso EAD LINUX
Apostila com os comandos das vídeoaulas e exercícios de fixação

Utilização dos comandos Linux
Apostila criada por Grimaldo Oliveira
MAIO/2013

Informações

CURSO: LINUX – Teoria e Prática	**DIREITOS:** Uso da apostila deve ser autorizado. Enviar e-mail para solicitação.
WEB SITE DE ACESSO AO CURSO: www.aprendavirtual.com	
DATA CRIAÇÃO: 08/04/2013	**REVISÃO:** 3.1
DATA ÚLTIMA MODIFICAÇÃO: 09/02/2014	**VERSÃO:** 2.5
AUTOR Grimaldo Lopes de Oliveira	**EMAIL:** *grimaldo_lopes@hotmail.com*

SOBRE O PROFESSOR

Grimaldo é baiano e soteropolitano. Fez graduação em Estatística pela Universidade Federal da Bahia e logo em seguida uma especialização na Área de Mineração de Dados/BI na Faculdade Visconde de Cairu. Após esta formação, "mergulhou" na área. Trabalha com ferramentas script desde 2003, iniciando projetos em uma empresa de telefonia celular, era responsável por toda a automação de shell script do sistema operacional UNIX (idêntico ao linux). Atualmente trabalha com shell script e banco de dados na PRODEB. Também é mestre em Gestão e Tecnologia Aplicadas à Educação (Gestec), pela Universidade do Estado da Bahia e editor do blog BIcomVatapa.blogspot.com.

Conheça Grimaldo:

Blog: www.bicomvatapa.blogspot.com
Site: www.aprendavirtual.com - (Aulas EAD)
Facebook:www.facebook.com/groups/bicomvatapa/
Perfil:br.linkedin.com/in/grimaldo

Sumário

Introdução

"A persistência é o caminho do êxito."

Charlie Chaplin (1895-1976)

1. Apresentação

O curso de Shell Script, foi idealizado para permitir um rápido aprendizado prático do aluno, através de uma interação com o professor por vídeoaulas pelos mais diversos comandos do LINUX. Durante o curso o aluno terá que assistir as vídeoaulas e praticar diretamente no seu computador. Não será necessário instalar nenhuma distribuição Linux, será utilizado um emulador linux o **CYGWIN**. O aluno terá um prazo máximo de curso de 3 meses, mas poderá solicitar uma única prorrogação pelo tempo que desejar, contudo o aluno será avisado por e-mail sobre o término do curso. Durante todo o curso, o aluno poderá retirar dúvidas com o professor e colegas, através do fórum de dúvidas ou por e-mail.

1.1 Sobre este documento

O objetivo deste documento é fornecer ao usuário, os passos necessários para que o aluno aprenda a trabalhar com shell script na sua plenitude, onde detalhes mais específicos sobre cada comandos serão explicados, através de uma linguagem direta, facilitando a construção de programas na linguagem.

1.2 Recursos necessários

Para acesso ao curso é necessário que o aluno tenha uma internet de rápido acesso para assistir aos vídeos, além de um leitor pdf da apostila do curso, devem ser utilizados os navegadores de internet Firefox(Mozila) e Google Chrome, com às versões mais atuais para acesso ao site de aulas EAD.

1.3 Atualizações deste documento

Futuras modificações poderão ocorrer no conteúdo deste documento em decorrência de possíveis ajustes na documentação do curso, seja elas oriundas do professor ou devido a atualizações pertinentes que possam ser demandadas pelos alunos, dentro de critérios lógicos que não afetarão os objetivos para o qual este documento foi criado.

Comandos Shell Básico

"Não se pode gerenciar aquilo que não se mede."

Lord Kelvin (1824-1907)

Os comandos abaixo estão nas vídeoaulas do curso, acompanhe os vídeos e re-execute os comandos para fixação

1. Primeiros lista de comandos

1- Criar o diretório curso_linux
- **mkdir curso_linux**

2- Entrar no diretório curso_linux
- **cd curso_linux**

3- Verificar o diretório que está logado
- **pwd**

4- Criar o primeiro arquivo, vamos entrar no editor VI
- **vi paises.txt**

5- Baixe o arquivo do site de aulas e abra o arquivo no seu desktop ou notebook, marque e copie os dados no editor VI, o arquivo contém os paises que tem moeda no mundo.
Arquivo: paises.txt

6- Agora clique com botão direito e cole no linux, para salvar o arquivo clique na sequência as teclas:
- aperte a tecla **ESC** e depois digite : e depois digite em sequencia **wq**

7- Saia do diretorio e volte para o raiz
- **cd ..**

8 – Agora vamos renomear o diretório
- **mv curso_linux arquivo_linux**

9 – Agora entre novamente no diretório e veja que o arquivo paises.txt está lá.
- **ls –l**

10 - Agora entre no diretório que foi alterado.
- **cd arquivo_linux**

2. Segunda lista de comandos

1- Vamos copiar o arquivo paises.txt para paises_copia.txt

- **cp paises.txt paises_copia.txt**

2- Vamos trocar o nome do arquivo paises_copia para country.txt

- **mv paises_copia.txt country.txt**

3- Vamos ver uma sequência de comandos **ls**, suas variações de parâmetros:

- **ls –a** → lista os arquivos e as entradas implícitas de diretórios;
- **ls –la** → você pode combinar parâmetros sem problema, união dos comandos;
- **ls –A** → lista os arquvios sem as entradas implícitas dos diretórios;
- **ls -c** → lista os arquivos em colunas;
- **ls -m** → lista os arquivos separados por virgula;
- **ls -s** → lista os arquivos com as informações sem blocos ao lado.

4- Criando lik´s simbólicos pelo comando **LN**, muito útil quando você quer criar um atalho para não digitar um nome tão extenso, no exemplo o arquivo recebeu um apelido.

- **ln -s retira_duplicados_paises.txt sem_duplicados.txt**

3. Terceira lista de comandos

1- Vamos listar o conteúdo de um arquivo, paises.txt

- **cat paises.txt**

2- Podemos combinar comandos, listar o conteúdo e contar quantidade de palavras

- **cat paises.txt | wc -l**

3- Podemos salvar o resultado de um comando em um novo arquivo, são os direcionamentos do linux

- **cat paises.txt > grava_paises.txt**

4- Vamos listar para ver o resultado

- **ls -l**

5- Podemos ordenar as linhas de um arquivo, basta executar o comando sort

- **sort paises.txt**

6- Vamos utilizar os dados ordenados e criar um novo arquivo, utilize o > para redirecionar a saída para um novo arquivo e depois liste este novo arquivo.

- **sort paises.txt > paises_ordenado.txt**

- **cat paises_ordenado.txt**

4. Quarta lista de comandos

1- Lembre-se, caso você não esteja no diretório do curso, digite:
- **cd arquivos_linux**

2- Vamos aprender a exibir as informações de parte de um arquivo, comando CUT, mostrará apenas as informações do arquivo grava_paises.txt da posição 8 até 12.
- **cat grava_paises.txt | cut –c8-12**

3- Vamos realizar vários exemplos do comando CUT:

- **cut–c6-** → Exibe os dados da coluna 6 do arquivo em diante
- **cat grava_paises.txt | cut -c6- > so_paises.txt** → Grava o resultado no arquivo so_paises

4- Vamos trabalhar com as permissões dos arquivos:

- **-chmod 444 so_paises.txt** → O dono, o grupo e outros usuários só terão acesso de leitura
- **chmod 311 so_paises.txt** → O dono terá acesso de escrita e execução, os demais apenas de execução
- **chmod u+wr so_paises.txt** → Outra forma de trabalhar com chmod é informando diretamente a permissão que deseja, no exemplo o arquivo so_paises, terá permissão de escrita e leitura apenas
- **chmod u+wr,o+xr so_paises.txt** → Agora o dono do arquivo e qualquer outro usuário terão as permissões de dono=escrita e leitura e outros=execução e leitura

5- Vamos trabalhar com as permissões de diretórios, primeiro vamos para o diretório raiz:
- **cd ...**

6- Vamos agora mudar as permissões do diretório arquivos_linux
- **chmod 755 arquivos_linux**
- **ls -l**

Obs: Note que o arquivo tem as mesmas características de um arquivo, permanece as definições no caso, dono(permissão leitura,gravação e execução), grupo(permissão apenas de leitura e execução) e outros usuários (leitura e execução). 7-RWX 5-R-X 5-R-X, ou seja,

7- Vamos alterar o grupo de um arquivo, pois a permissão no linux quando você cria um

arquivo ou diretório é criada ao grupo que o usuário pertence, portanto você pode alterar para um outro grupo de um um outro usuário sem problemas, no caso vamos alterar do grupo **none** para **root**.

- **cd arquivos_linux**
- **chgrp root grava_paises.txt**
- **ls -l**

5. Quinta lista de comandos

1- Vamos continuar com mais comandos de manipulação de arquivos, o próximo comando é o **DIFF**. Vamos ver se dois arquivos diferem e quais são estas diferenças:

Obs.: Edite o arquivo paises.txt e elimine a primeira linha (basta apagar) e salve (lembre-se da sequencia de comandos (**ESC : wq**)

- **diff paises.txt paises_ordenados.txt**

Obs.: Note que foram apresentados as diferenças abaixo, veja as definições do que é exibido.

1d0
<
243a243
> FEGANISTAO

Caso tenha aparecido a informação acima, o **1d0** significa que no primeiro arquivo necessita apagar um espaço em branco, a informação **243a243** significa que deve-se acrescentar a palavra **FEGANISTAO** ao primeiro arquivo.

2- Agora vamos trabalhar com um dos comandos mais utilizados no Linux, o **GREP** e o **EGREP,** que realizam pesquisas de string´s e de arquivos no linux.

- **grep BRASIL paises.txt** → Busca a palavra brasil no arquivo paises.txt
- **egrep –w 'BRASIL|ALEMANHA' paises.txt** → Busca duas palavras ao mesmo tempo no arquivo paises.txt
- **grep –c "BRASIL" paises.txt** → Conta quantas vezes aparece a palavra **BRASIL** dentro do arquivo
- **grep -n 'ALE' paises.txt** → Exibe as linhas onde existem a string **ALE**, ou seja, se uma parte da palavra contiver a string

ALE, então exibirá

- **grep -i -n "ale" paises.txt** → Exibe as linhas e pesquisa ignorando se a palavra está escrita em maiúsculo ou minúsculo, você pode combinar parâmetros do comando sem problema
- **grep -r 'BRA' /home** → Pesquisa recursivamente em todos os diretórios a partir do **/home**, quais arquivos possuem a string 'BRA';
- **history | grep cut** → Pesquisa se o comando **CUT** foi digitado no sistema atevés do **HISTORY** do linux;
- **ls -R /home | grep paises** → Combinação de comandos, primeiro pesquisa recursivamente todos os diretórios a partir do **/home** e lista, em seguida através do **grep** só exibe aqueles cujo arquivo possui no nome a string **paises**;
- **ls -l | grep --color paises** → Caso você deseje colorir o resultado da pesquisa, basta acrescentar o parâmetro **--color**;

3- Utilizaremos o comando **FILE** para identificar os tipos de arquivos que possuímos no diretório, isto será útil quando da necessidade de uso destes arquivos em sistemas que irão ler um tipo especifico.
- **file p*.***

4- Vamos utilizar o comando **UNIQ** que retira linhas duplicadas de um arquivo. Antes vamos fazer a junção de dois arquivos idênticos e em seguida retirar as linhas duplicadas.

- **cat paises.txt > juncao_paises.txt**
- **cat paises.txt >> juncao_paises.txt**
- **sort juncao_paises.txt > sort_juncao.txt**
- **uniq sort_juncao.txt > retira_duplicados_paises.txt**
- **cat retira_duplicados_paises.txt**

5- Caso você deseje exibir um arquivo quebrando em uma específica coluna, utilize o comando **FOLD**.

- **fold -w 10 nacoes.txt**

6. Sexta lista de comandos

1- O comando **DD** é bastante interessante, pois poderemos fazer conversões de arquivos e guardar no formato ISO (Imagem de Disco), para restaurar posteriormente, vamos ver funcionando.

- **dd if=paises.txt of=imagem_paises.img**

2- Agora vamos restaurar a imagem gravada, lembre-se imagem de disco são geralmente utilizados para backup, depois liste o arquivo e confira os dados.

- dd if=imagem_paises.img of=paises_restaura.txt
- cat paises_restaura.txt

3- Agora vamos trabalhar com o comando **JOIN** para união de arquivos, os mesmos devem estar na mesma estrutura, devem ser arquivos com layout´s iguais, caso contrário não funcionará.

- join paises.txt paises_ordenado.txt > uniao_paises_join.txt
- cat uniao_paises_join.txt

Obs.: Perceba que o primeiro campo dentro do arquivo paises e paises_ordenados são eles que permitem a união, ou seja, uma chave de relacionamento.

4- Vamos aprender a compactar um arquivo no linux, existem várias maneiras, mas vamos ao primeiro comando, o comando **TAR**.

- **tar –cvf paises.tar pais*.***	→ Compactar todos os arquivos que tem a string inicial “pais”. O parâmetro -**c** informa para compactar, -**v** exibe os detalhes da operação, -**f** permite especificar o nome do arquivo a ser compactado
- **tar -tf paises.tar**	→ O comando exibe os arquivos que estão compactados, ou seja, mostra o conteúdo do arquivo
- **rm paises_ordenado.txt**	→ Vamos descompactar um arquivo, para isso elimine o arquivo paises_ordenado.txt
- **tar -xvf paises.tar paises_ordenado.txt**	→ Agora vamos descompactar, o parâmetro **–X** faz esta função.

5- Vamos utilizar para compactação o comando **GZIP**.

- **gzip -v paises_ordenado.txt** → Compacta e exibe a taxa de compressão

- **gzip -l paises_ordenado.txt** → Lista as informações da compressão

- **gzip -d paises_ordenado.txt** → Descompacta o arquivo

6- Vamos exibir o conteúdo de um arquivo comprimido sem descomprimir, comando **ZCAT**, vamos primeiro comprimir um arquivo, depois exibir o conteúdo.

- **gzip paises.txt**
- **zcat paises.txt.gz**

7- Vamos ver a diferença entre arquivos com o comando **SDIFF**.

- **sdiff sort_juncao.txt so_paises.txt**

Obs: Note que na saída há exibição de dois diferentes caracteres:

| → Indica que as linhas são diferentes, linha a linha.
< → Indica que a linha só existe no primeiro arquivo.
> →Indica que a linha só existe no segundo arquivo (nosso exemplo não houve nenhuma).

- **sdiff sort_juncao.txt so_paises.txt –o sdiff_juncao.txt**

Obs.: Você pode gravar em um arquivo separando a saída da comparação, com o parâmetro **–o**, ao final aperte **<ENTER>** e escolha uma das opções, por exemplo digite **l** e gravará apenas a saída do arquivo da esquerda.

8- Agora vamos aprender a dividir um arquivo em várias partes, comando **SPLIT**.

- **split --lines=30 paises_ordenado.txt**

Obs: Por default, os arquivos saem com o prefix **x** à frente, para colocar seu prefixo vejo o comando abaixo:

- **split --lines=30 paises_ordenado.txt nacoes**

Obs.: Para juntar novamente o arquivo use o comando **CAT** com o direcionador **>**

- **cat nacoesaa nacoes ab nacoesac > nacoes.txt** (fiz a união de 3 arquivos)

9- O comando a seguir exibe as últimas linhas de um arquivo, comando **TAIL.**

- tail –n10 nacoes.txt

10- O comando a seguir exibe agora as primeiras linhas, comando **HEAD.**

- head –n5 nacoes.txt

Obs.: Agora vamos unir os comandos **TAIL** e **HEAD** e exibir um intervalo, no exemplo abaixo, todas as linhas do arquivo nacoes.txt da linha 5 até a 8

- head -n8 nacoes.txt | tail -n3

11- Alterando data e hora de um arquivo, caso ele não exista, será criado vazio, comando **TOUCH.**

- touch -t 201305020405.20 nacoes.txt

Obs.: o formato é AAAMMDDHHMM.SS

12- Agora veremos um dos comandos mais poderosos do linux, o **FIND**.

- **find / -name *.txt** → Busca todos os arquivos/diretórios que tem a extensão .txt a partir do diretório raiz do linux
- **find /usr -name *.txt** → Pesquisa todos os arquivos/diretórios com extensão .txt a partir do diretório usr
- **find /home -perm 755** → Exibe todos os arquivos/diretórios a partir do diretório /home que tem a permissão concedida como 755
- **find /home -perm /u=w,g+w** → Exibe todos os arquivos/diretórios a partir do diretório /home que tem a permissão no arquivo para usuário como escrita(w) **OU** com a permissão de escrita no grupo
- **find /home -name xaa -delete** → Você pode utilizar o commando **find** para pesquisar e eliminar um arquivo, no caso eliminaremos o arquivo **xaa**, criado com o uso do commando **split** no exercício anterior
- **find /home *.* -mtime 1** → Arquivos/diretórios modificados entre 24 e 48 horas atrás
- **find /home *.* -mtime +1** → Arquivos/diretórios modificados a mais de 48 horas
- **find /home *.* -mtime -1** → Arquivos/diretórios modificados a menos que 1 dias ou 24 horas
- **find / -size +2048 -print** → Exibe todos os arquivos com mais de 2 megas de tamanho

- **find / -group root** → Pesquisa todos os arquivos que pertencem ao grupo root

7. Sétima lista de comandos

1- Agora chegou a hora dos comandos internos do linux, o primeiro é o **MAN**, que exibe os parâmetros e explica um comando do linux, é equivalente ao HELP de qualquer programa.

- **man ls**

2- Agora existe o comando **HELP** que ao digitar exibe uma lista dos comandos internos do linux, note que alguns comandos vistos aqui, ficaram de foram, isso porque são programas (TAR, GZIP, LS, dentre outros).

- **help**
- **help ls**

3- Agora vamos exibir data e hora do sistema.

- **date**

4- Agora vamos ao comando que exibe quanto de espaço há no disco.

- **df**
- **Tente também variações (df –h ; df -a ; df -T; df -l)**

5- Agora vamos ao comando que exibe quanto de espaço há no disco utilizado por arquivos e diretórios (pastas).

- **cd /home | du**
- **du –h** (espaço utilizado em Kbyte)

Obs.: **entre** no **diretório arquivos_linux** da sua máquina e digite o comando

- **du –h *.txt**

- **du –h *.txt | sort –n** (coloca em ordem de tamanho do menor para o maior)

6- Agora utilizaremos o comando **ECHO,** você vai ouvir falar muito dele.

- **echo "BRASIL"** → Exibe a mensagem BRASIL no terminal ou

		seja na tela de comando
- **echo 'qual a sua cidade' > /tmp/log_curso.txt**	→	Grava a mensagem no arquivo **log_curso.txt** no diretório **/tmp**, este diretório comumente é utilizado pelo linux para guardar arquivos temporários
- **echo 'qual a sua cidade' >> /tmp/log_curso.txt**	→	Grava a mensagem no arquivo **log_curso.txt** no diretório **/tmp**, sobrescrevendo o conteúdo já gravado anteriormente
- **echo "Hoje é dia de: $(date)"**	→	Mensagem do dia de hoje que exibe a data de hoje pela variável interna **$date**
- **echo $PATH ; echo $HOSTNAME; echo $HISTSIZE**	→	Exibe o conteúdo de variáveis internas, o ; concatena os comandos

8. Oitava lista de comandos

1- Iniciaremos com o comando **TEST** que nos será muito útil quando fizermos programas com shell script.

- **test –e nacoes.txt; echo $?** (verifica se o arquivo existe)
- **test –d ~/arquivos_linux/; echo $?** (verifica se o diretório existe)
- **test -s nacoes.txt; echo $?** (verifica se o arquivo existe e se não está vazio)

2- Os comandos abaixo mostram os usuários do sistema, dependerá da distribuição do linux.

- **users; who; whoami** (deve aparecer o seu usuário apenas)

3- Lista os últimos comandos digitados, comando **HISTORY**.

- **history** (o limite são de 500 linhas por default, mas pode ser alterado)

9. Nona lista de comandos

SED , veremos muitos exemplos do uso do SED, estrutura básica do SED é:

Sed ‘expressão’ arquivo

É um editor de texto não interativo, onde trabalha com várias entradas string seguindo um conjunto de regras específicas.

Obs.: É possível filtrar a saída de outro comando, em vez de filtrar um arquivo, usando pipe (|). Usarei saídas do comando echo para explicar o sed.

1- Primeiro iremos trocar a palavra professor na saída do comando **echo**, note trocou apenas a primeiro string “**professor**” que encontrou.

- **echo "Venha aprender linux com professor e aprenda de forma simples com ele o professor" | sed 's/professor/grimaldo/'**

2- Para trocar todas as string´s acrescente o parâmetro /g.

- echo "Venha aprender linux com professor e aprenda de forma simples com ele o professor" | sed 's/professor/grimaldo/g'

3- Agora existem alguns caracteres especiais que temos que tomar precausões, como por exemplo o **/**, veja o que fazer para funionar, colocar a junção**/**.

- echo "o arquivo /arquivo_linux possui informações importantes do curso EAD" | sed 's/\/arquivo_linux/\/curso_ead/'

4- Vamos utilizar os chamados retrovisores, onde quebraremos uma linha em várias partes.

- echo -e "primeira linha\nsegunda linha\nterceira linha" | sed -r 's/^([a-z]+).*/\1/g'

5- Vamos substituir uma palavra a partir da terceira, independente de estar em maiúscula ou minúscula.

- echo " o PAIS do FUTURO" | sed -r 's/([a-zA-Z0-9]+)/brasil/3g'

6- Agora vamos mostrar quatro usuários do sistema, eles estão no diretório **/etc/password**. Caso retire o 4 e coloque **$** mostrará até o último usuário, ou seja, até a última linha do arquivo password.

- sed -r -n '1,4 s/([^:]+).*/\1/p' /etc/passwd

7- Podemos remover as linhas de um arquivo com **SED**. Vamos remover as 50 primeiras linhas do arquivo nacoes.txt.

- cat nacoes.txt | sed '1,50 d'

8- Agora vamos mostrar as 10 primeiras linhas do arquivo.

- cat nacoes.txt | sed '10 q'

9- Substituindo todos os espaços em branco por quebra de linha.

- sed 's/ /\n/g' nacoes.txt

10- O sed pode substituir diversos comandos como o **CAT**, basta digitar a linha abaixo:

- sed : nacoes.txt

Para saber tudo sobre SED, acesso o link de Aurelio Marinho Vargas, o site mais completo sobre o comando SED. **http://aurelio.net/sed/sed-HOWTO/**

10. Décima lista de comandos

AWK , linguagem de programação para realizar manipulação de dados em textos e arquivos:

awk '<padrao-acao>' [arquivo_1] [arquivo_2]... [arquivo_n]

Algumas variáveis definidas no AWK:

FS -> separador de campos
NR -> número das linhas
NF -> número de campos na linha

1- Primeiro iremos fazer algo simples, como entender a separação de campos no comando **AWK**, os campos são definidos pelo simbolos **$**, então no exemplo do arquivo nacoes.txt temos 2 campos, **codigo e país**, que para o AWK serão **$1** e **$2**, então vamos imprimir apenas o segundo campo.

- awk '{print $2}' nacoes.txt

2- Vamos utilizar o comando para pesquisar no arquivo **nacoes.txt,** todos os paises que começam com a string 'GA'.

- awk '/GA/ {print}' nacoes.txt

3- Agora vamos mostrar o uso do AWK com o comando ls, vamos trabalhar com a saída do comando ls, você verá que há muitos campos com um simples **ls**.

- ls -ltr --| awk '{print $1}'; ls -ltr --| awk '{print $9}'

4- Podemos criar uma condição para exibição de determinados conteúdos dos campos, veremos com mais profundidade quando estudarmos shell script programação, mas vamos ver como funciona, vamos pesquisar pelo tamanho do arquivo.

- ls -ltr | awk '{if ($5 == "1787") print ($0)}'

5- Agora, vamos fazer ao contrário e exibir aqueles arquivos com tamanho diferentes de 1787.

- ls -ltr | awk '{if ($5 != "1787") print ($0)}'

6- Vamos agora fazer algumas impressões com as variáveis do AWK.

- ls -l /etc |awk '{print $1 FS $8}'
Obs.: Faça a experiência sem o FS
-ls -l /etc |awk '{print $1 $8}'

7- Vamos exibir os países cuja a quantidade de caracteres seja 6.

- sed : nacoes.txt | awk 'length($2) == 6'

Obs.: Note que apareceram campos com mais de 6 carasteres no nome do país, pois para o awk a separação de campos etsá no espaço entre eles, como no país: CHINA, REPUBLICA POPULAR. Note o espaço entre CHINA, REPUBLICA

8- Acrescentando string entre os campos.

- cat nacoes.txt | awk '{print "Codigo: " $1 " Pais: " $2}

9- Vamos contar qual a maior linha do arquivo **nacoes.txt**.

- awk '{ if (length($0) > max) max = length($0)}; END { print max}' nacoes.txt

10- Número de linhas do arquvio nacoes.txt, igual ao commando **wc –l**.

- awk 'END { print NR }' nacoes.txt

Shell Script - Programação

"Não se pode gerenciar aquilo que não se mede."

Lord Kelvin (1824-1907)

11. Shell Script – Programação

A programação Shell script foi criada para facilitar e permitir que os diversos comandos vistos no curso até aqui, possam ser encadeados no formato de programa, ou seja, você poderá além de digitar na linha de comando, comando a comando, poderá criar uma sequência e colocar todos os comandos em um arquivo, que geralmente possui como extensão .sh.

A programação shell script não é compilada e sim interpretada, ou seja, enquanto o programa é executado, caso haja algum erro na linha o programa irá parar. Você pode criar inúmeros programas, chamados de shell script´s, para realizar diversas tarefas como: realizar as tarefas diárias de um servidor, fazer pesquisas em arquivos textos ou diretórios, fazer cópias de seguranças, dentre outros.

Para que os comandos sejam entendidos, devemos escolher um interpretador para a execução, no caso do nosso curso, instalamos o interpretador **BASH** que é muito conhecido. Sempre que criarmos um programa shell script, deveremos colocar no início do programa a linha (**#!/bin/bash**) e a partir da segunda linha, digitar o programa que desejamos. Após salvar o programa devemos torná-lo "**executável**", utilizaremos o comando **chmod** que vimos no curso.

Vamos fazer nosso primero programa:

- Digite:

- vi primeiro.sh

```
#!/bin/bash
echo " meu primeiro programa"
```

- Clique com as teclas **ESC : w q**
- Vamos torná-lo executável: **chmod 755 primeiro.sh**

- Execute através do comando: **./primeiro.sh**

Parabéns!!! Você criou seu primeiro programa em linux. Agora vamos aprender o que é indispensável para que você começe a programar:

11.1 Entendendo o que é uma variável

As variáveis são utilizadas para armazenar dados e o interpretador bash reconhece uma variável pelo simbolo "$" , por exemplo: a variável **cidade**, armazenará um conteúdo e a exibição do seu conteúdo deverá **sempre** ser unida ao símbolo **$**.

- Digite:

- cidade= "Salvador"
- echo $cidade

- Digite:

- var1="Estou online com o usuário $USER"
- echo $var1

- Digite:

- var2='Estou online com o usuário $USER'
- echo $var2

- Digite:

- var3="O diretório que estou online é `pwd`"
- echo $var3

Note, utilizamos variáveis do sistema ($user) para serem exibidas de forma diferente. A variável **$user** dentro de aspas duplas, o conteúdo foi impresso, a mesma variável destino de apóstrofos(') não foi exibida, e a colocação de um comando dentro de crases (`) o conteúdo foi impresso.

Você pode interagir com o script e permitir que se digite na medida que o programa é executado.

- echo "Digite um numero qualquer: " ; read numero

Vamos ver o uso do comando backstick ou crase(`), vamos guardar o resultado do comando em uma variável.

- data=`date`
- echo $data

Mais um exemplo com a listagem de um diretório.

- listagem=`ls`
- echo " Lista dos arquivos no diretório específico" $listagem

11.2 Comandos Condicionais

Antes de verificarmos os comandos condicionais, vamos conhecer os operadores relacionais que poderemos utilizar nos comandos condicionais:

- eq → Igual
-ne → Diferente
-gt → Maior
-lt → Menor
-o → Ou
-d → Se for um diretório
-e → Se existir
-z → Se estiver vazio
-f → Se conter texto
-o → Se o usuário for o dono
-r → Se o arquivo pode ser lido
-w → Se o arquivo pode ser alterado
-x → Se o arquivo pode ser executado

1- Vamos trabalhar com o comando condicional **IF**, preste atenção no vídeo para entender como você irá criar o programa abaixo, ok!

Primeiro vamos criar o arquivo que conterá o programa:

- vi pergunta-if.sh

Copie e cole o programa abaixo, pois a digitação no **VI** é complicada, no vídeo explico melhor.

```
#!/bin/bash
echo -n 'Qual país foi 5 vezes campeã do mundo no futebol masculino? '
read resposta
if test "Brasil" = "$resposta"
then
```

```
echo " Incrivel voce acertou, parabens"
else
echo " A  $resposta nem pensar voce errou :-)"
fi
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 pergunta-if.sh
- ./pergunta-if.sh

Obs: Digite um pais qualquer como **Bolivia** e veja o resultado, depois execute novamente e digite **Brasil**.

Outro Exemplo com o comando **IF**, veja abaixo:

```
- vi curso.sh
#!/bin/bash
if [ -e $curso ]
then
  echo 'A variável $curso foi criada'
else
  echo 'A variável $curso não foi criada'
fi
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 curso.sh
- ./curso.sh

2- Vamos trabalhar com o comando condicional **CASE**, ele faz a mesma coisa do comando **IF**, só que bem mais organizado e com mais facilidade para manutenção futuras do script. Preste atenção no vídeo para entender como você irá criar o programa abaixo, ok!

```
- vi case.sh
 #!/bin/bash
case "$1" in

 'pwd' )
```

```
            echo "Mostra o diretorio"
            pwd
            ;;
 'ls' )
            echo "lista o diretorio"
            ls
            ;;
*)
            echo "Opção invalida!"
            echo "As opções válidas são: pwd, ls"
            ;;
esac
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 case.sh
- ./case.sh ls

11.3 Comandos de Repetição

1- Vamos trabalhar com o comando **FOR**, preste atenção no vídeo para entender como você irá criar o programa abaixo, ok!

Primeiro vamos criar o arquivo que conterá o programa:

- vi for-dir.sh

Vamo criar um programa que lista um diretório com o comando **FOR**

```
#!/bin/bash
for lista in `ls`
do
echo "Arquivo $lista"
done
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 for-dir.sh

- ./for-dir.sh

2- Vamos trabalhar com o comando **WHILE**, preste atenção no vídeo para entender como você irá criar o programa abaixo, ok!
Primeiro vamos criar o arquivo que conterá o programa:

- vi while-soma.sh

```
#!/bin/bash
while :
do
read -p "Digite dois numeros com espaco ( digite - 1 que sai ) : " a b
        if [ $a == -1 ]
        then
                break
        fi
        ans=$(( a + b ))
        echo $ans
done
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 while-soma.sh
- ./while-soma.sh

3- Vamos trabalhar com o comando **UNTIL**, outro comando de repetição, muito utilizado, preste atenção no vídeo para entender como você irá criar o programa abaixo, ok!
Primeiro vamos criar o arquivo que conterá o programa:

- vi until.sh

```
#!/bin/bash
count=0
until [ $count = "20" ]; do
  echo " estamos na posicao $count"
  if [ $count = 10 ]; then
    echo "chegamos a metade da contagem, termina em 20"
  fi
  count=`expr $count + 1`
done
```

echo "FIM!!!"

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.

- chmod 775 until.sh
- ./until.sh

11.4 Variáveis Especiais

Importante que você saiba que no linux, existem variáveis especiais que guardam valores das execuções dos script´s e também são utilizados para a passagem de parâmetros via script shell, vejamos quais são.

$0	Nome do script que está sendo executado
$1-$9	Parâmetros passados à linha de comando
$#	Número de parâmetros passados
$?	Valor de retorno do último comando ou de todo o shell script executado. (o comando "exit 1" retorna o valor 1)
$$	Número do PID (Process ID)

- Digite:

- ls -ltr
- echo ?$

- Digite:

- ls -zd
- echo ?$

Vamos fechar esta etapa com um programa completo, utilizaremos funções, acompanhe.

- vi manutencao.sh

```
#!/bin/bash
# Exemplo do uso de funcoes no shell script
#
Inicio() {
  echo "Exemplo uso de funcoes no shell script"
  echo "----------------------------------------"
```

```
 echo "Opções:"
 echo
 echo "0. Diretorio corrente"
 echo "1. Listar os arquivos do diretorio"
 echo "2. Fazer a copia dos arquivos paises para o diretorio backup do dia"
 echo "3. Criar o diretório backup do dia"
 echo "4. Apagar um arquivo especifico"
 echo "5. Sair do programa"
 echo
 echo -n "Escolha a opcao desejada? "
 read opcao
 case $opcao in
   0) Corrente ;;
   1) Listar ;;
   2) Copiar ;;
   3) Criar ;;
   4) Apagar ;;
   5) exit ;;
   *) "Opção desconhecida." ; echo ; Inicio ;;
 esac
}
Corrente () {
 pwd
 echo -n "Aperte Enter "
 read entrada
 Inicio
}
Listar() {
 ls -ltr
 Inicio
}

Copiar() {
 echo -n "Qual o nome do arquivo que você deseja copiar? "
 read arquivo
 echo -n " Lembre-se, voce criou o diretorio do dia? [S/N]"
 read cria
 if [ $cria = "S" ]; then
    cp $arquivo $diret/$arquivo
 fi
  Inicio
}
Criar() {
```

```
  echo -n "Digite o nome do diretorio? "
  read diret
  mkdir $diret
  Inicio
}

Apagar() {
  echo -n "Digite o nome do arquivo para apagar? "
  read arqapa
  rm $arqapa
  Inicio
}
Inicio
```

Salve agora digitando **:qw**
Caso você tente executar o programa o mesmo não funcionará, você deverá dar permissão de execução ao arquivo.
- chmod 775 manutencao.sh
- ./manutencao.sh

11.5 Programas curtos

Os programas abaixo podem ser utilizados em linha de comando ou colocados dentro de um arquivo .sh, são programas curtos e prontos para serem utilizados, basta você substituir as variáveis indicadas no código por um conteúdo específico. Este código facilitam algumas ações do seu dia a dia, Vejamos:

Loop de 1 à 10

```
for i in 1 2 3 4 5 6 7 8 9 10; do echo $i; done
for i in $(seq 10); do echo $i; done
for ((i=1;i<=10;i++)); do echo $i; done
i=1 ; while [ $i -le 10 ]; do echo $i ; i=$((i+1)) ; done
i=1 ; until [ $i -gt 10 ]; do echo $i ; i=$((i+1)) ; done
```

Loop nas linhas de um arquivo ou saída de comando

```
cat /etc/passwd | while read LINHA; do echo "$LINHA"; done
grep 'root' /etc/passwd | while read LINHA; do echo "$LINHA"; done
while read LINHA; do echo "$LINHA"; done < /etc/passwd
```

while read LINHA; do echo "$LINHA"; done < <(grep 'root' /etc/passwd)

Curingas nos itens do comando case

case "$user" in root|analista|curso) echo "Oi $user";; *) echo "Não te conheço";; esac
case "$var" in ?) echo '1 letra';; ??) echo '2 letras';; ??*) echo 'mais de 2';; esac
case "$i" in [0-9]) echo '1 dígito';; [0-9][0-9]) echo '2 dígitos';; esac

Condicionais com o IF

if [-f "$arquivo"]; then echo 'Arquivo encontrado'; fi
if [! -d "$dir"]; then echo 'Diretório não encontrado'; fi
if [$i -gt 5]; then echo 'Maior que 5'; else echo 'Menor que 5'; fi
if [$i -ge 5 -a $i -le 10]; then echo 'Entre 5 e 10, incluindo'; fi
if [$i -eq 5]; then echo '=5'; elif [$i -gt 5]; then echo '>5'; else echo '<5'; fi
if ["$USER" = 'root']; then echo 'Oi root'; fi
if grep -qs 'root' /etc/passwd; then echo 'Usuário encontrado'; fi

Condicionais com o E (&&) e OU (||)

[-f "$arquivo"] && echo 'Arquivo encontrado'
[-d "$dir"] || echo 'Diretório não encontrado'
grep -qs 'root' /etc/passwd && echo 'Usuário encontrado'
cd "$dir" && rm "$arquivo" && touch "$arquivo" && echo 'feito!'
["$1"] && param=$1 || param='valor padrão'
["$1"] && param=${1:-valor padrão}
["$1"] || { echo "Uso: $0 parâmetro" ; exit 1 ; }

11.6 Schedulle de um programa

Agora que criamos alguns programas, vamos aprender a agendar no linux, você pode agendar o programa para ser executado diariamente, mensalmente ou de hora em hora, minuto a minuto, dentre outras formas vai depender do seu uso.

Programa : CRONTAB

O crontab tem o seguinte formato:

[minutos] [horas] [dias do mês] [mês] [dias da semana] [usuário] [comando]

O preenchimento de cada campo é feito da seguinte maneira:

- **Minutos:** informe números de 0 a 59;
- **Horas:** informe números de 0 a 23;
- **Dias do mês:** informe números de 0 a 31;
- **Mês:** informe números de 1 a 12;
- **Dias da semana:** informe números de 0 a 7;
- **Usuário:** é o usuário que vai executar o comando (não é necessário especificá-lo se o arquivo do próprio usuário for usado);
- **Comando:** a tarefa que deve ser executada.

Ex:
10 08 10,13 * * echo "primeiro agendamento" >/home/saida.txt

Neste exemplo, a frase "primeiro agendamento" é inserida no arquivo saida.txt, dentro do diretório /home/, às 10 horas e 08 minutos, nos dias 10 e 13, em todos os meses e em todos os dias da semana

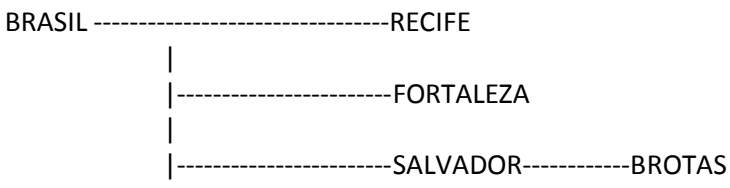
12. Exercícios Finais

Vamos agora ver se o curso foi proveitoso, faça os exercícios abaixo e envie por email para que o professor possa corrigir.~

12.1 Primeira Bateria de Exercícios

OBSERVAÇÃO: COLOQUEI AS RESPOSTAS NO ARQUIVO RESP-EXERCICIO-1, QUE ESTÁ COM SENHA, APENAS VOCÊ SÓ TERÁ ACESSO SE ENVIAR E-MAIL PARA O PROFESSOR SOLICITANDO A RESPOSTA APÓS ENVIAR A TENTATIVA DA RESOLUÇÃO.

1) Criar a seguinte árvore no diretório no seu diretório home/aluno



```
                                        |
                                        |-----PELOURINHO-------- SE
                                                                 |----AXE
                                                                 |----DANCA
```

1.1) Volte ao diretorio BRASIL estando no diretório AXE;

1.2) Apague o diretório BRASIL/SALVADOR/BROTAS;

1.3) Crie um arquivo em BRASIL/SALVADOR/PELOURINHO/DANCA/alegria.txt e digite no arquivo 5 nomes de praias;

1.4) Copie este arquivo para o diretório BRASIL/RECIFE;

1.5) Crie um arquivo chamado estudo.txt dentro do diretório BRASIL/FORTALEZA com os dados do comando ls -al do diretório /etc;

1.6) Criar um arquivo chamado permissao.doc dentro do diretório BRASIL/FORTALEZA com permissão apenas de leitura para os elementos usuário, dono e grupo;

1.7) Copie o arquivo permissao.doc para o diretório /BRASIL/SALVADOR. Terminada esta operação renomeie o arquivo para nova_permissao.doc.

1.8) Qual a capacidade de seu HD ? Quantos por cento está sendo usado do seu HD ?

1.9) Qual é o total (bytes, kbytes e Mbytes) utilizado pelo diretório /BRASIL ?

2) Quais foram os arquivos acessados nos últimos 10 dias ?

2.1) Localize o arquivo alegria.txt a partir do diretório /home.

2.2) Mude a data/hora do arquivo BRASIL/SALVADOR/PELOURINHO/DANCA/alegria.txt

2.3) Qual é o tamanho em MBytes de:

/bin
/dev
/home
/proc

/sbin
/boot
/etc
/lib
/root
/tmp

12.2 Segunda Bateria de Exercícios

3) Baixe o arquivo exercício-2 que está no site de aulas e grave no diretório BRASIL/FORTALEZA e execute os caomandos abaixo:

OBSERVAÇÃO: COLOQUEI AS RESPOSTAS NO ARQUIVO RESP-EXERCICIO-2, QUE ESTÁ COM SENHA, APENAS VOCÊ SÓ TERÁ ACESSO SE ENVIAR E-MAIL PARA O PROFESSOR SOLICITANDO A RESPOSTA APÓS ENVIAR A TENTATIVA DA RESOLUÇÃO.

3.1) Mostre na tela, o conteúdo do arquivo.

3.2) Mostre na tela o conteúdo do arquivo e localize a palavra "CABO".

3.3) Mostre o início do arquivo, antes de começar a listagem de produtos.

3.4) Mostre os últimos 23 produtos desta lista

3.5) Mostre a lista de produtos completa, usando o comando "tail"

3.6) Mostre a lista de produtos da linha 26 até a 35.

3.7) Ordene esta lista em ordem de código.

3.8) Ordene e lista pelo nome dos produtos.

3.9) Ordene a lista pelo preço: os produtos mais caros no início

3.10) Mostre somente os produtos da marca AIPTEK.

3.11) Mostre somente as câmeras digitais que NÃO são da marca SONY.

3.12) Quantas DVD possui a listagem?

3.13) Qual o código e o preço do produto mais caro?

3.14) Qual o preço do cabo de impressora USB?

3.15) Quantos produtos existem na listagem?

3.16) Crie um arquivo "especial.txt" com os 10 produtos mais caros.

3.17) Crie um arquivo "liquidar.txt" com o código dos próximos 10 produtos mais caros.

3.18) Quantos aparelhos EPSON existem? Grave eles no arquivo "epson.txt"

3.19) Crie o arquivo "barato-caro.txt", com os 5 primeiros e os 5 últimos produtos.

3.20) Junte as listas "especial.txt" e "epson.txt" no arquivo "bem-barato.txt"

3.21) Crie um novo arquivo, parecido com o original MAS neste, a lista com os produtos deverá estar numerada e com os produtos em ordem alfabética reversa - as linhas do início do arquivo não deverão ser numeradas, apenas terão numeração os produtos.

3.22) Crie um arquivo com a lista das diferentes marcas de HUB da lista.

12.3 Terceira Bateria de Exercícios

OBSERVAÇÃO: COLOQUEI AS RESPOSTAS NO ARQUIVO RESP-EXERCICIO-3, QUE ESTÁ COM SENHA, APENAS VOCÊ SÓ TERÁ ACESSO SE ENVIAR E-MAIL PARA O PROFESSOR SOLICITANDO A RESPOSTA APÓS ENVIAR A TENTATIVA DA RESOLUÇÃO.

4) Escreva um script que mostre a data de hoje e depois todos os arquivos dentro do seu diretórios, onde você guardou os arquivos do curso.

5) Escreva um script que faça o teste se um específico arquivo existe ou não. Você permitirá que o usuário digite o nome do arquivo como um parâmetro de entrada.

6) Faça um script que imprima quantos processos estão atualmente em execução na sua máquina. Tente criar através do uso dos comandos **wc** e **ps**.

7) Faça um scritp que renomeia todos os arquivos ".txt" para ".doc" do diretório que você gerou os arquivos do curso.

8) Crie um script para listar todos os arquivos de um determinado diretório (passado por parâmetro o diretório que deseja) mas no seguinte formato:
1: arquivo-1
2: arquivo-2
…..

9) Crie um script que permita informar quantas linhas e palavras existe em um determinado arquivpo (passado por parâmetro o nome do arquivo).
Obs.: Lembre-se do comando **WC**

10) Faça um script que receba um número qualquer através de parâmetro e imprima na tela uma contagem regressiva até chegar a zero, não tão rapidamente, cada número a cada segundo (use o comando sleep para esperar).

11) Crie um script que seja capaz de gerar backlup de arquivos no sistema- O programa deve receber um nome de diretório a ser compactado e também um nome de diretório onde o arquivo com o backup será será guardado. O nome do arquivo de backup deve serguir o formato "BKP-AAAA-MM-DD.tar.gz", onde AAAA é o ano, MM o mês e DD o dia.
Obs: Use o comando **TAR** para fazer a criação do backup. Se preocupe em criar testes para determinar se os diretórios de origem e destino realmente existem.

12) Escreva um script capaz de ir adicionando linhas a um arquivo chamado atividades_do_dia.txt. O programa deve ser executado passando na linha de comando uma atividade que você irá realizar naquele dia ou nos próximos dias. Antes da colocação da atividade informar a data/hora do dia (comando date pode ajudar) que a atividade foi inserida no arquivo, por exemplo:

23/03/2013 12:40 Almoçar com o presidente da empresa

13) Para criar este script, utilize o comando **GREP**, procure os arquivos criado com extensão **.txt** redirecionando a saída para o arquivo root.txt. Depois, busque todos os arquivos com extensão **.doc**, redirecionando a saída para o arquivo root.doc. Finalmente, junte os arquivos root.txt e root.txt em um único arquivo juntos_root.txt

Considerações Importantes

"O usuário terá a habilidade de priorizar as fontes e os temas e escolher deliberadamente o que ele quer saber. Será uma atividade que a próxima geração já vai aprender a fazer nas escolas."

Pierre Lévy (1956)

13. Considerações Importantes

- Importante que você saiba que você pode retirar dúvidas com o professor no momento que desejar, para isso entre em contato via e-mail, o mesmo está na folha de rosto da apostila ou através do fórum dos alunos;
- Lembre-se que uma internet de banda larga ajudará na visualização dos vídeos, quando temos lentidão no acesso da internet isso influenciará na aprendizagem rápida do curso;
- Importante que você utilize o shell script diariamente, ou um espaço de tempo de uma aula para outra pequeno, pois isso facilitará seu entendimento;
- Lembre-se o curso tem um custo baixo, para permitir que mais colegas possam realizar o curso e retire dúvidas com os professores e colegas, não compartilhe seu usuário e senha, pois prejudica uma cadeia de profissionais que trabalharam no curso.
- Qualquer dificuldade não hesite e entre em contato com o professor do sistema, passe um e-mail.

13.1 Futuras atualizações

Toda necessidade de inclusão de novos comandos e exercícios, devem ser solicitados ao professor, lembre-se você pode melhorar e muito o curso informando problemas no acesso e sobre algum erro encontrado e identificado na apostila.

Dicas importante: **http://www.hugoazevedo.eti.br/html/shell_script_exerc.html**

Apêndice

Glossário de Siglas e Termos

"Não há saber mais ou saber menos: Há saberes diferentes.."

Hodding Carter (1907-1972)

I. Glossário de Siglas e Termos

A seguir estão disponíveis em ordem alfabética, a relação de siglas e termos frequentemente utilizados durante o curso de Shell Script.

- A –

ALUNO – É o indivíduo que recebe formação e instrução de um ou vários professores ou mestres para adquirir ou ampliar seus conhecimentos.

- E –

EAD – É uma modalidade de educação mediada por tecnologias em que alunos e professores estão separados espacial e/ou temporalmente, ou seja, não estão fisicamente presentes em um ambiente presencial de ensino-aprendizagem.

- L –

LINUX– é um termo utilizado para se referir a sistemas operativos (português europeu) ou sistemas operacionais (português brasileiro) que utilizem o núcleo Linux. O núcleo Linux foi desenvolvido pelo programador finlandês Linus Torvalds, inspirado no sistema Minix.

- P –

PROFESSOR - É uma pessoa que ensina uma ciência, arte, técnica ou outro conhecimento.

PROGRAMAÇÂO - É um método padronizado para comunicar instruções para um computador.1 É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.

- S –

SHELL SCRIPT - É uma linguagem de script usada em vários sistemas operativos (operacionais), com diferentes dialetos, dependendo do interpretador de comandos utilizado.

- U -

UNIX – É um sistema operativo (ou sistema operacional) portátil (ou portável), multitarefa e multiutilizador (ou multiusuário) originalmente criado por Ken Thompson, Dennis Ritchie, Douglas McIlroy e Peter Weiner.