UNIX Básico

em 10 lições

Alexandre Botão

“Release” 4.1 @ 2000.01.17

As dez lições

+ Primeira Lição : Muito Prazer ! UNIX !

  (o que é, para que serve, etc ...)

+ Segunda Lição : Antes de mais nada ...

  (conceitos teóricos)

+ Terceira Lição : Entrando, brincando e saindo

  (entrar em sessão, ..., sair de sessão)

+ Quarta Lição : Diretórios

  (criar, passear, ...)

+ Quinta Lição : Arquivos

  (copiar, remover, ...)

As dez lições

+ Sexta Lição : Segurança

  (proteger arquivos, diretórios, ...)

+ Sétima Lição : Editando texto

  (o editores “ed” e “vi”)

+ Oitava Lição : O shell

  (o processador de comandos)

+ Nona Lição : Utilitários úteis

  (“ferramentas” diárias)

+ Décima Lição : miscelanea

  (faltou dizer ...)

OVERVIEW (ou “objetivos”) ! ! !

apresentação do UNIX
de onde vem, a que veio, pra onde vai,
por que é famoso

+ Conceitos gerais

multiusuário, multitarefa, host-based, cluster,
client-server, distributed, consolidation,
tcp/ip, ping, telnet, ftp, nfs, dns, nis, ...

+ conceitos UNIX

usuário, super-usuário, grupo, filesystem,
arquivo, diretório, arq. especial, link,
C, C++, java, tcl, tk, motif, x-windows, gtk+,
shell (sh, csh, ksh, bash), perl, python, ...

+ UNIX versus ...

ms-dos (...)
ms-windows 3.1 (r.i.p.)
ms-windows 95/98 (tem seu nicho)
ms-windows NT/2000 (em 2010 ...)
novell-netware (ainda ?)
ibm-os/2 (r.i.p.)

+ “Sabores” de UNIX

IBM : aix/risc

HP : hp-ux/risc

SUN : sunos, solaris/sparc
solaris/intel

DEC : ultrix/vax, OSF/1(dunix)-risc

SCO : sco-xenix/intel
sco-unix/intel
unixware/intel

BSD : FreeBSD
OpenBSD
NetBSD

LINUX ! : redhat, mandrake, debian, suse, caldera, slackware, trinux, corel, stormix, stampede, turbolinux, extreme linux / beowulf project, winlinux, dragonlinux, embedix, yellowdog, blacklab, gentoo, ubuntu, fedora ...

MAIS ! : xinu, minix, SGI/irix, rhapsody, gnu hurd, next, openstep, macosx, openBSD (security above all) , netBSD (h/w indepencence), BSDi (commercial version), freeBSD (x86 optimized)

links	url's (http://www.)
linux international	li.org
linux standard base	linuxbase.org
linux online	linux.org
linux today	linuxtoday.org
linux care	linuxcare.com
linux weekly news	lwn.net
linux resources	linuxresources.com
blacklab linux	blacklablinux.com
caldera linux	calderasystems.com
redhat linux	redhat.com
suse linux	suse.com
trinux	trinux.org
slackware linux	slackware.com
turbo linux	turbolinux.com
yellowdog linux	yellowdoglinux.com
slashdot	slashdot.org
opensource	opensource.org

+ UNIX e o hardware

memória, disco, cpu, terminal, micro

+ UNIX e a mulher

o operador
o administrador
o programador

+ Geografia e geologia

/ , /bin , /etc , /usr , /tmp , /lib , /dev , /var , /opt
/home (ou /u ou /users ou /seilá)

+ Entrando e saindo do UNIX

login, senha, prompt, exit

+ Arquivos

“criar”, copiar, renomear, deletar, mover,
listar, status, olhar,
imprimir (enfileirar, consultar, cancelar)

+ Diretórios

criar, renomear, deletar, entrar, sair, passear
o "pai" (..)

+ comunicação simples

mensagens instantâneas (write)
como inibi-las e desinibi-las

+ correio eletrônico “universal” (mail)

preparar e enviar carta
checar e ler cartas

+ proteção de arquivos

quem pode: dono, grupo, outros
o que pode: ler, gravar, executar

ls -l
chmod
umask (proteção default)

+ proteção de diretórios

quem pode: dono, grupo, outros
o que pode: ler, gravar, pesquisar

ler um diretório = saber seu conteúdo

gravar num diretório = poder criar, renomear ou deletar “membros” dele

pesquisar :“passar por” (pathname)
ou “entrar” (cd)

+ programação shell (curso à parte – baratinho ...)

expansão de nomes

redirecionamento de i/o
pipes
background

echo
variáveis de ambiente
set
read
for-do-done
if-then-else-fi
case-in-esac
while-do-done
break
continue

+ processos

ps, kill, background

+ utilitários úteis
wc, sort, grep, find, tar, cpio, awk, etc ...

Primeira Lição : Muito Prazer ! UNIX !

+ O QUE ele é ?

+ ONDE ele nasceu ?

+ QUANDO ele nasceu ?

+ POR QUE usá-lo ?

+ PARA QUE usá-lo ?

+ COMO usá-lo ?

“ se voce é um usuário ,
todos os UNIX são IGUAIS ... ”

“ se voce é um programador C ,
os UNIX são SUTILMENTE diferentes ... ”

“ se voce é um administrador ,
os UNIX são COMPLETAMENTE DIFERENTES ! ”

O QUE ele é ?

- é um sistema operacional (e nao um "ambiente").

- é multiusuário, multitarefa, multicpu,
  multiplataforma, multilingüe.

- é "escrito" em linguagem "C", o que o torna
  "rapidamente disponível" em CHIPS novos.

- os "multi-..." :

  -usuário : muitas pessoas compartilham os recursos
  do computador (cpu, memória, disco, ...)

  -tarefa : muitos programas "rodam" concorrentemente
  (veja “time-sharing”)

  -cpu : o UNIX "roda" em máquinas com muitas CPU's

  -plataforma : o UNIX está disponível em máquinas
  de diversos fabricantes, de "tamanhos"
  (capacidade) e preços bem variados.

ONDE ele nasceu ?

- nasceu nos "Bell Labs." da AT&T
  (ou BTL - Bell Telephone Laboratories)

- desenvolvido originalmente por Ken Thompson
  (com grande participação de Dennis Ritchie,
  que é co-autor da linguagem de programação "C",
  junto com Brian Kernighan)

- "cresceu" (recebeu contribuições técnicas) em várias
  universidades americanas, principalmente na UCB
  (University of California at BERKELEY)

- hoje é conhecido como UNIX da AT&T

- ou ainda, como UNIX SYSTEM V (leia-se "system five")

- subsiste em "sabores" dos "grandes fornecedores" :

  AIX (IBM) , HP-UX (HP) , SOLARIS (SUN) , OSF/1 (DEC) ,

  SCO-UNIX (SCO) , UNIXWARE (NOVELL) , UNIX (AT&T) , ...

- porem TODOS são ditos COMPATÍVEIS com o

  UNIX SYSTEM V (que é o PADRÃO da indústria)

QUANDO ele nasceu ?

1970 - nasce a versao 1 num PDP-7 da Bell Labs.
em assembly, mono-usuário.
- aprox/e 10 anos antes de DOS e NETWARE
- aprox/e 20 anos antes do WINDOWS 3.1

1971 - já com muitos recursos de programação e de editoração, o UNIX versão 3 é "migrado" para um PDP-11, multi-usuário.

1972 - Dennis Ritchie "re-escreve" o UNIX em "C".

1973 - UNIX versão 7 começa a ir para as universidades e centros de pesquisa dos EUA.

1977 - AT&T libera o UNIX versão 7 para fins comerciais.

1980 - UNIX versão 7 chega ao Brasil, nas universidades.

1981 - AT&T libera o UNIX System III.

1983 - AT&T libera o UNIX System V (Padrão da Indústria).

1986 - surgem os primeiros UNIX "nacionais".
("compatíveis" com a versão 7 ou o system III)

1987 - vários fabricantes "adotam" o UNIX (IBM, HP, DEC, CRAY, PHILIPS, OLIVETTI, ...)

1990 - "sabores" (IBM, HP, SUN, ...) chegam ao Brasil

POR QUE usá-lo ?

- é "aberto" (pai dos "open systems") !

  - não é "proprietário" e conta com muita bibliografia.

- é "padrão" (da indústria) !

  - oferece independencia de fabricante e preserva os investimentos em treinamento e desenvolvimento.

- é "sólido" (não só pela idade) !

  - alem da segurança lógica de dados e usuários, seu ambiente nao é fértil para "virus".

- é versátil !

  - suporta várias modalidades de processamento, de apresentação, de aplicações, em vários modelos de computadores, dos mais variados fabricantes.

- tem portabilidade (pela padronização)
- tem escalabilidade (de 386 a CRAY, pela portabilidade)
- tem conectividade (pelos BNU e pelo TCP/IP)

- é ótimo para interoperabilidade (por tudo acima) !

PARA QUE usá-lo ?

- para processamento centralizado ou cooperativo
  ou distribuído (dados, aplicações, processamento)
  ou departamental ou pessoal.

- para aplicações comerciais, científicas, de tempo real,
  CAE/CAD/CAM, transações on-line, inteligência artificial,
  automação de escritório (planilhas eletrônicas, editores de
  texto, bancos de dados, correio eletrônico), multimídia,
  editoração eletrônica, orientação a objeto, ...

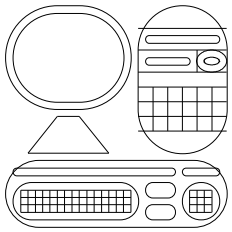
- em modo “caractere” ou “gráfico” (X-Windows).

- em estações pessoais, servidores, Intel ou Risc,
  mono- e multi-cpu, supercomputadores, ...

- com chips Intel, Motorola, Sparc, Mips, Ibm, Hp, Dec, ...

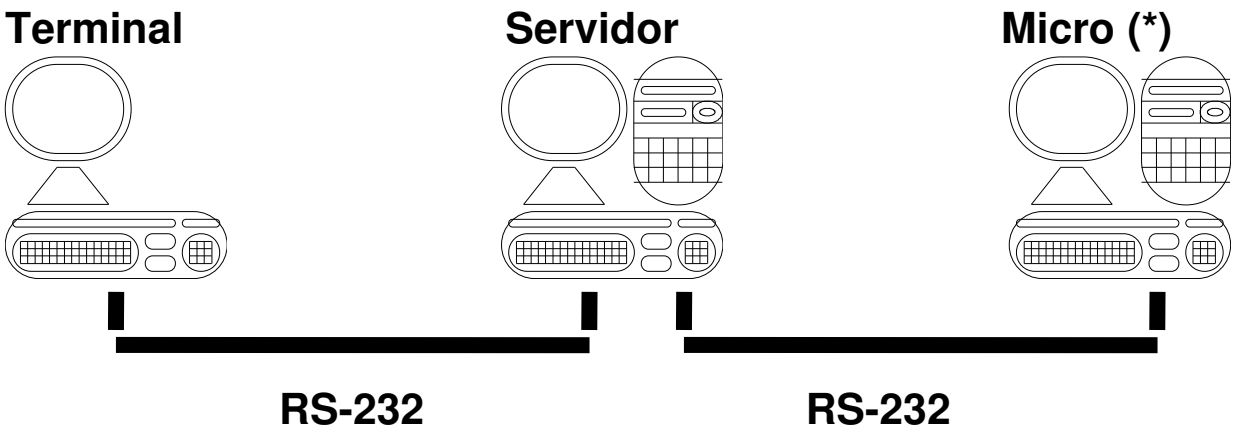
COMO usá-lo ?

“mono-usuário”



+ as aplicações são executadas na sua estação pessoal de trabalho, e os dados ficam no seu disco.

host-based, rede de TP serial

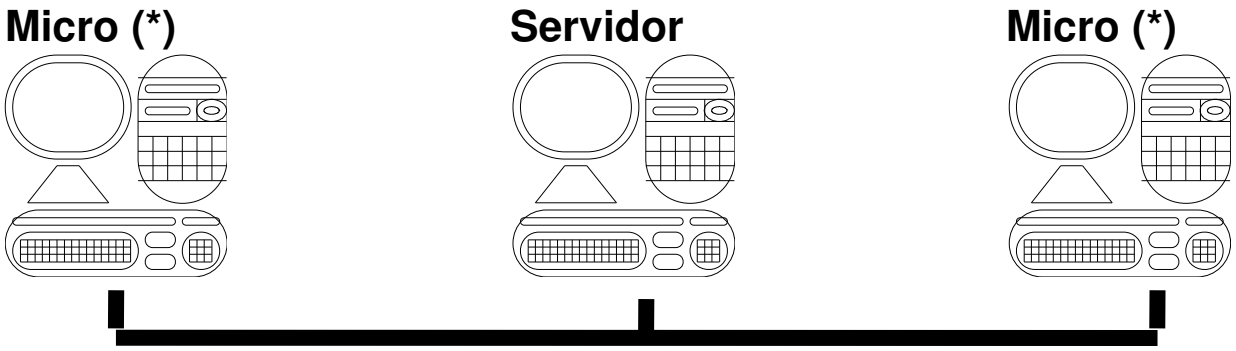


+ as aplicações rodam na cpu do servidor, os dados ficam no disco do servidor.

+ as conexões podem ser “diretas” ou “remotas” (modem).
+ os micros estão com “emuladores” de terminal serial.

COMO usá-lo ?

host-based (ou até client-server), rede TCP/IP



Ethernet (coaxial, par trançado), Token-ring

host-based

+ os micros estão com TCP/IP e o emulador “telnet”.

+ as aplicações rodam na cpu do servidor, os dados ficam no disco do servidor.

client-server

+ a parte “client” da aplicação roda na estação, enquanto a parte “server” roda no servidor.

- **os micros tem TCP/IP, mas o “telnet” não é necessário.**

COMO usá-lo ?

networked, client-server, distributed

Rede local (LAN) ou metropolitana (WAN)

+ na filosofia client-server, a parte “client” da aplicação roda na estação, enquanto a parte “server” roda no servidor.

+ os dados podem ser “distribuídos” entre servidores.

+ as conexões metropolitanas, podem usar RENPAC, redes privadas, satélite, ... em geral com X.25

Segunda Lição : Antes de mais nada ...

conceitos UNIX

usuário

é voce !

voce será reconhecido de duas formas :

uma identificação nominal, chamada USERNAME ou LOGNAME (nome-de-login) usada ao entrar em sessão.

uma identificação numérica, chamada USER ID (UID), que será "carimbada" nos "seus" arquivos e diretórios.

em geral, quem "escolhe" seu USERNAME é o "administrador" do sistema, e o seu USER ID é "resolvido" pelo UNIX quando voce é "cadastrado".

grupo

no UNIX, todo usuário pertence a, ao menos, um "grupo" !

o conceito de "groupware" foi introduzido pelo UNIX e incorporado a muitos outros sistemas operacionais.

seu objetivo é "agrupar" usuários de áreas e tarefas "afins", que compartilham dados e programas "comuns".

por exemplo: os funcionarios do RH, da diretoria, etc ...

Segunda Lição : Antes de mais nada ...

conceitos UNIX

o "super-usuário" (root)

o USERNAME “root” (e o USER ID 0) são especiais.

este usuário “pode TUDO” !

só o administrador deve usar esta identificação !

o administrador

é um usuário mais experiente, que fez o curso de “administração UNIX”, em geral do suporte ...

é o encarregado de cadastrar usuários novos, gerenciar as filas de impressão, o espaço livre em disco, as redes, etc ... alem de atuar em caso de problemas.

file system

no UNIX, os discos (winchesters) são organizados em “volumes lógicos” chamados “file-systems”.

diretório

é a “estrutura” de um file-system.
diretórios guardam arquivos, ou outros (sub-) diretórios.

Segunda Lição : Antes de mais nada ...

conceitos UNIX

arquivo

o UNIX trata arquivos como “conjuntos de bytes”, sem se importar com seu conteúdo.

shell

é o processador de comandos do UNIX.

é executado quando voce “entra em sessão”, para receber e executar seus comandos.

minúsculas

no UNIX, nomes de arquivos e comandos são digitados em MINÚSCULAS (minúsculas) !

sua identificação (e senha) tambem !

o UNIX difere MAIÚSCULAS de minúsculas !

setas

para corrigir erros de digitação, use a tecla [BackSpace] (⇦) que fica geralmente acima do [Enter], ou pressione [Ctrl H] (control H).
NÃO use as setas (← → ↑ ↓) !

Segunda Lição : Antes de mais nada ...

conceitos UNIX

login

para “usar” o UNIX (“dar” uns comandos), voce precisa “entrar em sessão” ou “abrir uma sessão” ou ainda, “fazer login” ou “se logar” ...

num terminal (ou equivalente - micro com emulador serial, ou com TCP/IP) voce deve encontrar a “tela de login”, que é o “convite” do UNIX à identificação do usuário.

terminal

quando seu “meio de acesso” ao UNIX é um terminal serial, basta ligá-lo, e a tela de “Login” já aparece ...

agora, se o seu caso é um micro com emulador serial, voce precisa “rodar” (executar) o emulador ...

no caso de micro com TCP/IP, antes de executar o emulador “telnet”, voce precisa saber o endereço TCP/IP do seu servidor ...

janelas (x-windows)

alem do “modo caractere”, o UNIX tem windows ! seu “modo gráfico” é chamado X-WINDOWS.

Terceira lição : Entrando, brincando e saindo ...

Entrando

1) o convite (ele pede a identificação) ...

obs.: o cursor será representado por □

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : □

2) a identificação (voce digita e tecla [Enter])...

obs.: a tecla [Enter] será representada por ↵

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
□

Terceira lição : Entrando, brincando e saindo ...

Entrando (cont.)

3) ele pede a senha (“password”) ...

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
Password : ☐

4) voce digita (ela não "aparece") e tecla [Enter] ...

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
Password :↵
☐

Terceira lição : Entrando, brincando e saindo ...

Entrando (cont.)

o que pode sair errado ?

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
Password :↵

Login Incorrect !

Login : □

- verifique o estado de "Caps Lock" ...
(deveria estar apagada, indicando "minúsculas")

- Digite sem pressa e com atenção.

- Não use as "setas" para corrigir um possível erro.
(se voce achar que errou, é melhor recomeçar,
teclando [Enter] até aparecer o convite "Login:")

Terceira lição : Entrando, brincando e saindo ...

Entrando (cont.)

Parabens ! Voce entrou “em sessão” !

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
Password : ↵

(opcionalmente, mensagens de saudações, “mensagem do dia”, notificação de correio eletronico, etc ...)

$ □

- o caractere “$” (dólar) é o “prompt”.

- ele significa “estou pronto” (para receber comando).

- a partir de agora, o UNIX está “às suas ordens”.

- voce está “em sessão”, ou “Logado”.
(voce “entrou em sessão”, ou “fez login”)

Terceira lição : Entrando, brincando e saindo

Brincando ...

+ olhando as horas ...

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : *cremilda* ↵
Password : ↵

$ *date* ↵
wed may 25 01:20:30 BST 1995
$ □

+ olhando os usuários “logados” ...

$ *who* ↵
maria tty23 Jun 28 10:34
cremilda tty45 Jun 28 12:05
roxana tty6 Jun 28 09:01
deboratty9 Jun 28 15:45
veronica tty8 Jun 28 09:27
$ □

Terceira lição : Entrando, brincando e saindo ...

Brincando ...

+ mudando a sua senha (ele pede a senha atual)

```
$ passwd ↵
old password : □
```

+ voce digita a senha atual (ela não aparece) e [Enter].
+ ele pede a senha nova ...

```
$ passwd ↵
old password : ↵
new password : □
```

+ voce digita a senha nova (ela não aparece) e [Enter].
+ ele pede para voce re-digitar a senha nova ...

```
$ passwd ↵
old password : ↵
new password : ↵
retype new password : □
```

+ voce re-digita a senha nova e [Enter].

```
$ passwd ↵
old password : ↵
new password : ↵
retype new password : ↵
$ □
```

Terceira lição : Entrando, brincando e saindo ...

Brincando ...

+ “quem sou eu” ou “quem abandonou o terminal” ...

```
$ who am i ↵
cremilda          tty45      Jun  28  12:05
$ □
```

+ produzindo mensagens no seu próprio terminal ...

```
$ echo o rato roeu a roupa da rogeria ↵
o rato roeu a roupa da rogeria
$ □
```

+ enviando mensagens a outro usuário ...

```
$ echo me liga na hora do lanche | write debora ↵
$ □
```

Terceira lição : Entrando, brincando e saindo ...

Brincando ...

+ enviando mensagens “grandes” a outro usuário ...

```
$ write  debora ↵
bom dia ! ↵
quer almocar comigo hoje ? ↵
sairei ao meio dia. ↵
um abraco ! ↵
[Ctrl-D]
$ □
```

+ habilitando e desabilitando a recepção de mensagens ...

```
$ mesg ↵
is  n
$ mesg  y ↵
$ mesg ↵
is  y
$ □
```

Terceira lição : Entrando, brincando e saindo ...

Saindo ...

$ ↵
$ exit ↵

Identificação do Sistema (ou equivalente: servidor, local, ...)

Login : □

+ não deixe sua sessão “aberta” quando se ausentar !

(no almoço, em reunião, ao ir para casa, banheiro, ...)

+ alem do comando “exit” voce pode “sair de sessão” pressionando [Ctrl] [d] (control-d).

+ voce “saiu de sessão”, ou “fechou a sessão”, ou “fez logoff” (ou “logout”).

+ Seção amnésia ...

quem sou eu ?

who am i

onde estou ?

pwd

que horas são ?

date

quem estão aí ?

who

que máquina é esta ?

hostname

socorro !

man

(*) : navegar é preciso ...

file-system = organização do volume

[boot], superbloco, i-list, dados, [swap]

diretório = estrutura do file-system

associa nome <==> status, dados

+ A barra (/) !

+ Diretórios

passear
criar
deletar
renomear
“entrar” e “sair”
o próprio (.)
o "pai" (..)

Quarta lição : Diretórios

considerações

+ todo disco rígido tem, no mínimo, um file-system.

+ o file-system é estruturado em “diretórios”.

+ o “primeiro” diretorio de um file-system é chamado de diretório-raiz, ou ainda “a raiz”.

+ “a partir dele” ou “dele para baixo” existem outros diretórios.

+ alguns são importantes para o UNIX, outros são importantes para alguns sistemas aplicativos, ou bancos de dados, etc ...

+ quando voce “entra em sessão”, voce “cai” ou “é colocado” (posicionado) no “seu diretório” !

+ o “seu diretório” é a “sua casa” no UNIX.

+ ele é chamado “diretorio-HOME”.

+ é “escolhido” (ou resolvido) por voce, ou, em geral, pelo administrador, na hora do seu “cadastramento”.

Quarta lição : Diretórios

+ em qual diretório voce está ? ...

```
$ pwd ↵
/curso/cremilda
$ □
```

+ “pathname” : é o “nome completo” de um arquivo ou diretório.

+ o comando “pwd” (print working directory) infoma o “diretório atual” ou “diretório corrente”.

+ o “diretório de trabalho” é o diretório onde voce “está” ...

Quarta lição : Diretórios

+ “mudar” de diretório (passear)

```
$ cd / ↵
$ pwd ↵
/
$ cd ↵
$ pwd ↵
/curso/cremilda
$ □
```

+ o comando “cd” (change directory) “muda” o seu diretório atual.

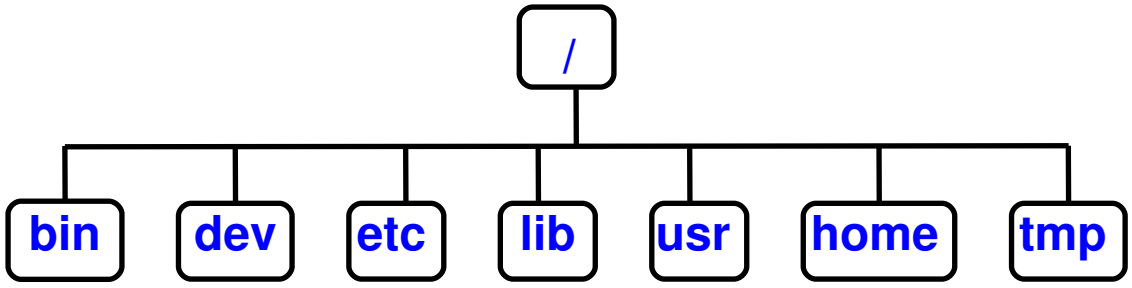
+ o “novo” diretório é passado como parametro ao cd.

+ entre o “cd” e o parametro (o diretório-destino) deve haver, no mínimo, um “espaço”.

+ usando o cd “puro” (sem parametro), ele “volta” ao seu diretório-HOME.

Quarta lição : Diretórios

alguns diretórios do UNIX



+ o diretório “/” (a raiz, ou “root directory”) é o HOME do usuário “root”.

+ ele é o “início lógico” ou “o topo” do disco.

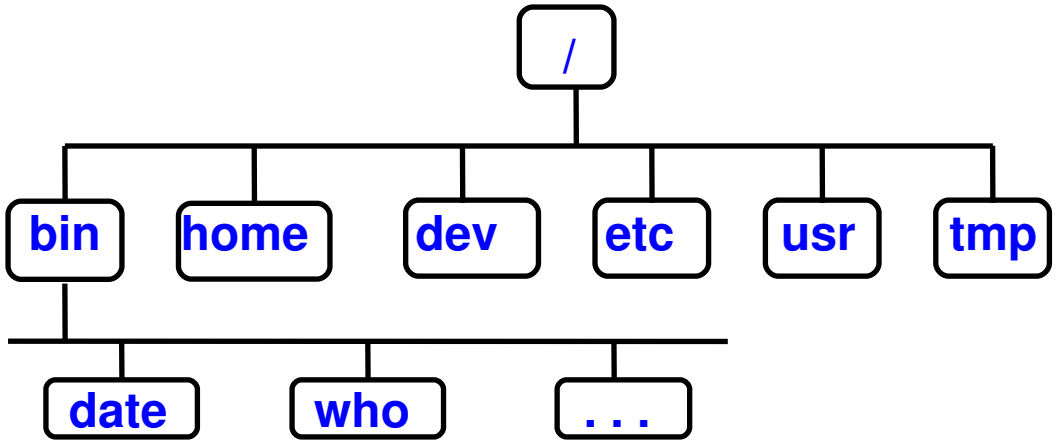
+ ele contem os diretórios “oficiais” do UNIX e alguns arquivos usados na hora do “boot”.

+ arquivos como “/boot” (que “carrega”o UNIX na memória) e “/unix” (o próprio “objeto” do UNIX) ficam na raiz.

“/bin”	programas principais do sistema
“/dev”	arquivos especiais (periféricos)
“/etc”	programas, arquivos, diretórios de administração
“/lib”	bibliotecas de funções (linguagem “C”)
“/usr”	programas, arquivos, diretórios de usuário
“/home”	diretórios HOME dos usuários
“/tmp”	arquivos temporários

Quarta lição : Diretórios

alguns diretórios do UNIX



+ o diretório “/bin” guarda os programas do sistema ...

  (como, por exemplo, “/bin/who” que é executado
  quando voce “dá” o comando “who” , ou o programa
  “date” que informa a data e hora atuais)

+ ele é um diretório “oficial” do UNIX, e não deve ser
  usado para guardar programas “de usuário”.

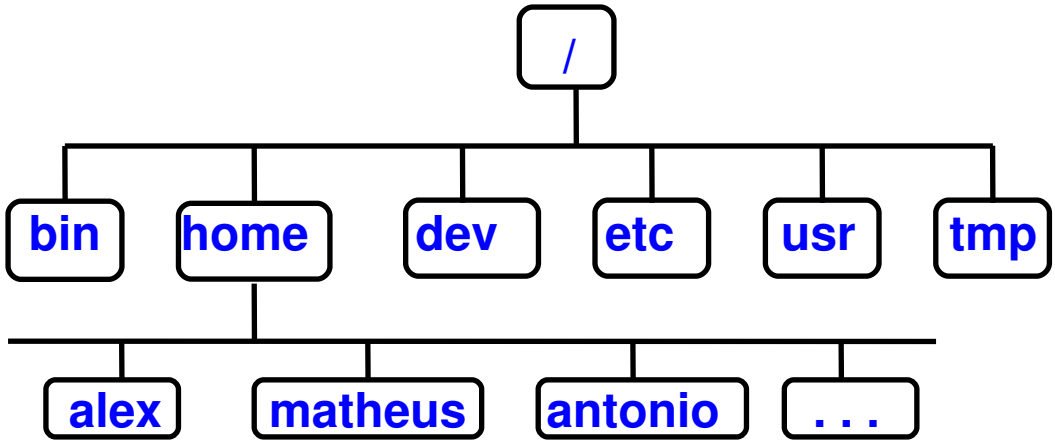
  + primeiro, por que existem diretórios específicos
    para isto.

  + segundo, por que em “upgrades” ou re-instalações
    o seu conteúdo, em geral, é “detonado” ...

+ falando no seu conteúdo, caso ele seja “perdido”
  (ou “danificado”), os usuários terão “problemas” ...

Quarta lição : Diretórios

alguns diretórios do UNIX



+ “abaixo” do diretório “/home” ficam os diretórios-HOME dos usuários ...

+ ele é um diretório “coletivo” ...

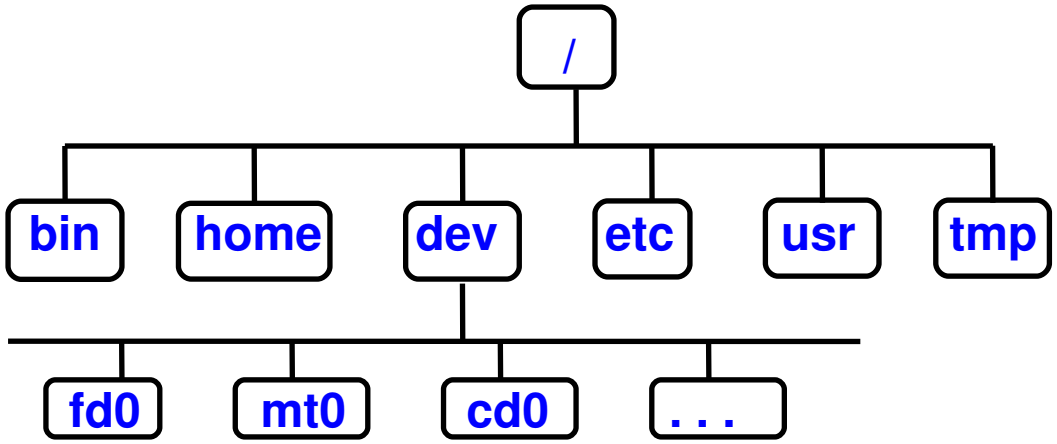
+ originalmente, os “lares” dos usuários ficavam no diretório “/usr”.

+ modernamente, os vários “sabores” de UNIX usam nomes “parecidos” para designar o “pai” dos diretórios-HOME ...

  “ / users “
  “ / home “
  “ / u “
  “ / usr “ (modernamente, não é recomendável)

Quarta lição : Diretórios

alguns diretórios do UNIX

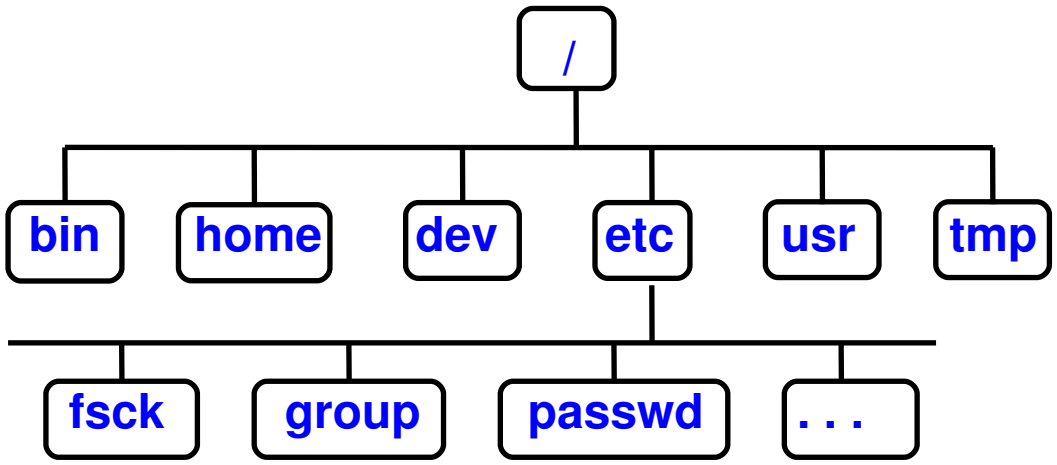


+ no diretório “/dev” ficam os arquivos “especiais”.

+ são arquivos “associados” aos periféricos do computador (diskette, unidade de fita, cd-rom, ...)

Quarta lição : Diretórios

alguns diretórios do UNIX



+ o diretório "/etc" guarda programas e arquivos relativos à administração do UNIX.

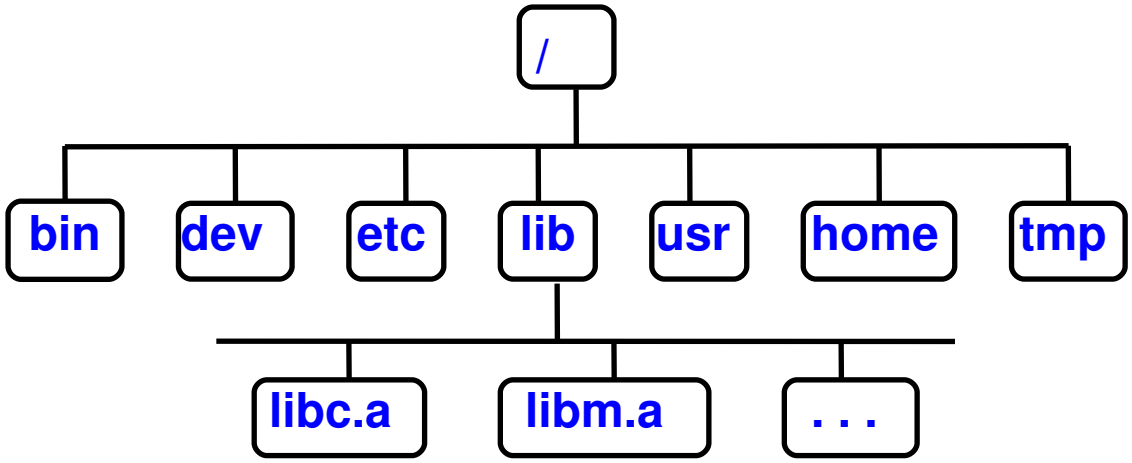
+ o programa "fsck" (file-system check) verifica e corrige (se possível) a integridade de um file-system.

+ o arquivo "group" contem as descrições dos grupos de usuários.

+ o arquivo "passwd" é o cadastro de usuários, contendo :
  + identificação (USERNAME ou LOGNAME)
  + USER ID (a identidade "numérica" interna do usuário)
  + GROUP ID (o grupo "primário" do usuário)
  + um campo "comentário" (ramal, ...)
  + o diretório HOME do usuário
  + o programa executado quando o usuário "se logar", que, em geral, é o shell ("/bin/sh") ou equivalente.

Quarta lição : Diretórios

alguns diretórios do UNIX

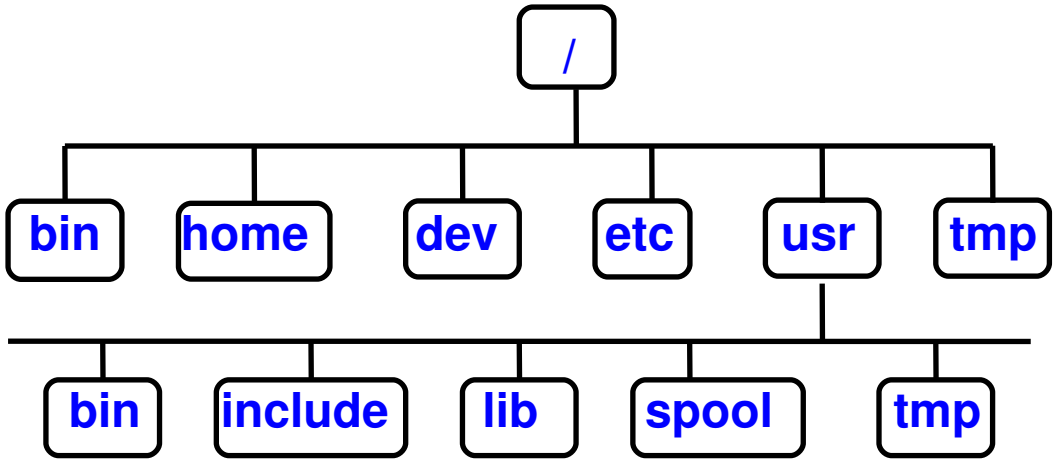


+ no diretório “/lib” ficam “bibliotecas” de funções usadas em linguagem “C”.

+ opcionalmente, guarda arquivos de dados do sistema.

Quarta lição : Diretórios

alguns diretórios do UNIX



+ o diretório “/usr” contem diretórios, programas e arquivos relativos ao dia-a-dia dos usuários.

+ por exemplo, no diretório “/usr/bin” ficam outros programas que estão mais relacionados ao usuário do que ao sistema operacional (estes ficam no “/bin”).

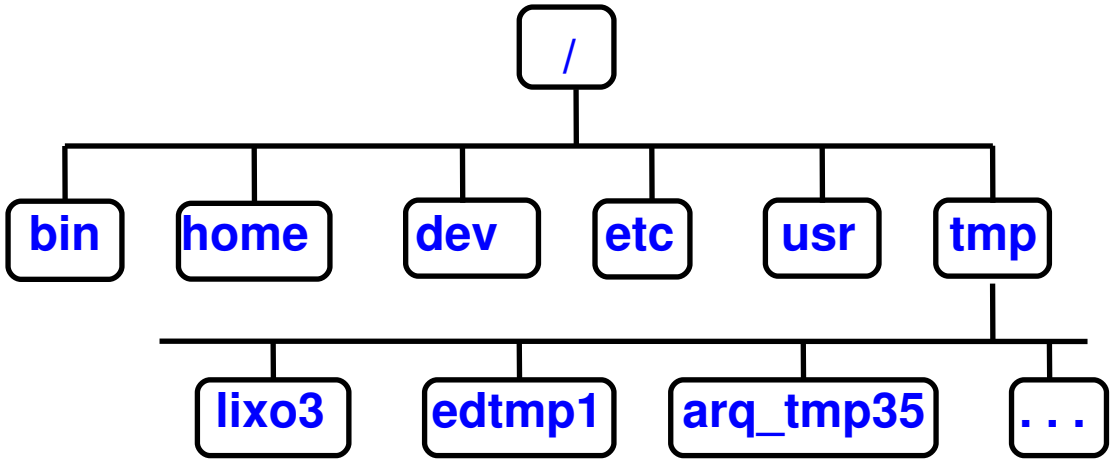
+ no diretório “/usr/spool” ficam programas, arquivos e diretórios relativos ao sistema de gerenciamento de filas de impressão.

+ os diretórios “/usr/lib” e “/usr/tmp” são análogos aos correspondentes “/lib” e “/tmp” ...

+ o diretório “/usr/include” guarda arquivos usados em programação “C”.

Quarta lição : Diretórios

alguns diretórios do UNIX



+ o diretório “/tmp” é um diretório publico, usado por vários programas para armazenar arquivos “temporários”.

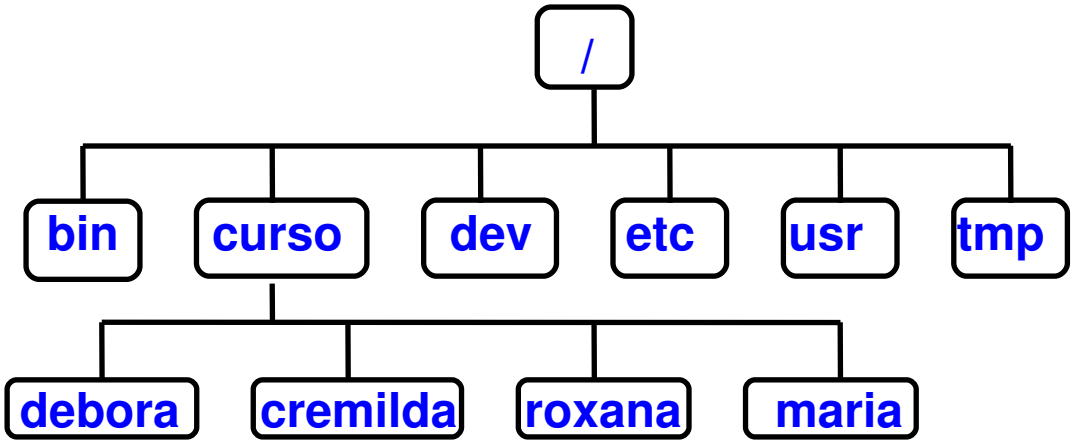
+ tradicionalmente, este diretório é “limpo” quando o UNIX está “entrando no ar”.

+ qualquer usuário pode criar arquivos e diretórios nele.

+ não é bom “entulhá-lo” com qualquer coisa (ou melhor, depois de “usar”, remova ...)

Quarta lição : Diretórios

alguns diretórios do UNIX



+ diretórios “coletivos” são muito usados, com o objetivo de “agrupar” usuários “afins” ...

+ sua localização e nome são de livre escolha ...

(desde que voce não os “misture” com os diretórios “oficiais”do UNIX)

+ alem do exemplo (alunos de um curso), podemos citar “funcionários de um departamento” (financeiro, RH, jurídico, suporte, ...)

Quarta lição : Diretórios

criar

```
$ mkdir  cartas ↵
$ ls ↵
cartas
$ mkdir  testes  lixo ↵
$ ls ↵
cartas   lixo   testes
$ □
```

+ o comando “mkdir” (make directory) cria o diretório cujo nome foi passado como parametro.

+ o comando “ls” (list status) mostra o conteúdo de um diretório (os nomes dos arquivos e diretórios nele).

+ usado “puro” (sem parametros), o comando “ls” mostra o conteúdo do diretório atual.

Quarta lição : Diretórios

remover

```
$ ls ↵
cartas   lixo   testes
$ rmdir  lixo ↵
$ ls ↵
cartas   testes
$ □
```

+ o comando “rmdir” (remove directory) remove o diretório cujo nome foi passado como parametro.

+ o diretório deve estar “vazio”.

renomear

```
$ ls ↵
cartas   testes
$ mv  testes  provas ↵
$ ls ↵
cartas   provas
$ □
```

+ o comando “mv” (move) muda o nome do diretório passado como primeiro parametro para o (novo) nome passado como segundo parametro.

Quarta lição : Diretórios

entrar

```
$ ls ↵
cartas   provas
$ cd  provas ↵
$ pwd ↵
/curso/cremilda/provas
$ □
```

+ para “entrar” num diretório usa-se o comando “cd”.

sair

```
$ pwd ↵
/curso/cremilda/provas
$ cd  . . ↵
$ pwd ↵
/curso/cremilda
$ □
```

+ para “sair” de um diretório usa-se o comando “cd”, mudando-se para o diretório “. .” (diretório “anterior”).

+ o diretório “. .” tambem é chamado de “pai” do diretório atual (aquele que “está acima” ou “vem antes”).

Quarta lição : Diretórios

irmãos

```
$ ls ↵
cartas   provas
$ cd  provas ↵
$ pwd ↵
/curso/cremilda/provas
$ cd  . . / cartas ↵
$ pwd ↵
/curso/cremilda/cartas
$ □
```

+ para “mudar” do diretório atual para um diretório “irmão” (filho do mesmo diretório-pai, ou que está no mesmo nível) usa-se o comando “cd” com a referência “. . /”).

+ a referência “. . /” significa “subir” ao diretório-pai (acima, ou anterior) e depois “descer” ao diretório após a barra.

+ os pontos e a barra (“. . /”) só aparecem “espacejados” por uma questão de legibilidade ...

+ NÃO DEVE haver “espaços” entre os pontos, a barra, e o nome.

Quinta lição : Arquivos

Considerações

+ no UNIX, um arquivo “comum” (uma carta, um programa executável, um banco de dados, etc ...) é considerado um conjunto de bytes, sem mais detalhes.

+ conceitos como “tamanho de registro”, “fator de bloco”, ou “marca de fim de arquivo” não são suportados ...

+ se algum arquivo tiver uma “estrutura” especial, a sua implementação, manutenção, suporte e tambem o “bom funcionamento” - tudo é responsabilidade dos programas que vão usar tal arquivo.

+ o único tipo de arquivo com uma “estrutura” definida e mantida pelo UNIX é o diretório (que é “organizado” em registros de tamanho fixo, armazenando um nome e um número).

+ arquivos especiais são aqueles associados a periféricos, não possuem estrutura, nem ocupam espaço em disco.

+ todos os arquivos possuem “atributos” (o “status”) que são armazenados e mantidos pelo UNIX, como tamanho, dono (e grupo tambem), tipo (arquivo, diretório, especial), proteção (as permissões de acesso), datas (de última leitura e de última gravação), etc ...

Quinta lição : Arquivos

criar

```
$ ls ↵
cartas   provas
$ pwd ↵
/curso/cremilda
$ cat > compras ↵
morango ↵
creme chantili ↵
Ctrl-D
$ ls ↵
cartas   compras   provas
$ □
```

+ o comando “cat” é um “lê-e-escreve” ...

+ neste exemplo, ele está sendo usado para ler “do teclado” e escrever num arquivo chamado “compras”.

+ para finalizar a digitação, usa-se “Ctrl-D” (control-D).

+ como o arquivo não existia, ele é criado.

+ caso o arquivo já exista, seu conteúdo prévio é “perdido”.

Quinta lição : Arquivos

olhar

```
$ ls ↵
cartas   compras   provas
$ cat < compras ↵
morango
creme chantili
$ □
```

+ desta vez, o comando “cat” está sendo usado para ler “do arquivo compras” e escrever “no vídeo”.

+ é possível “combinar” os dois exemplos para fazer uma cópia de um arquivo.

```
$ ls ↵
cartas   compras   provas
$ cat < compras > feira ↵
$ ls ↵
cartas   compras   feira   provas
$ □
```

+ neste caso, o “cat” leu do arquivo “compras” e escreveu no arquivo “feira”.

Quinta lição : Arquivos

copiar

```
$ ls ↵
cartas  compras  feira  provas
$ cp  feira  xpto ↵
$ ls ↵
cartas  compras  feira  provas  xpto
$ □
```

+ o comando “cp” (copy) na sua forma mais simples, faz uma cópia do arquivo “origem” (primeiro parametro) para o arquivo “destino” (segundo parametro).

+ ele tambem pode ser usado para copiar vários arquivos para um diretório “destino” (cujo nome deve ser o último parametro da linha de comando).

```
$ ls ↵
cartas  compras  feira  provas  xpto
$ cp  feira  xpto  compras  cartas ↵
$ cd  cartas ↵
$ ls ↵
compras  feira  xpto
$ □
```

Quinta lição : Arquivos

renomear

```
$ ls ↵
cartas  compras  feira  provas  xpto
$ mv  xpto  zeca ↵
$ ls ↵
cartas  compras  feira  provas  zeca
$ □
```

+ para renomear um arquivo, tambem usa-se o “mv”.
  (análogo ao que foi feito em “diretórios”, lembra ?)

deletar

```
$ ls ↵
cartas  compras  feira  provas  zeca
$ rm  zeca ↵
$ ls ↵
cartas  compras  feira  provas
$ □
```

+ o comando “rm” (remove) deleta o arquivo cujo nome foi passado como parametro.

Quinta lição : Arquivos

mover

```
$ ls ↵
cartas  compras  feira  provas
$ mv  feira  provas ↵
$ ls ↵
cartas  compras  provas
$ cd  provas ↵
$ ls ↵
feira
$ □
```

+ o ato de “mover” um arquivo para um novo diretório NÃO causa duplicação dos dados, como no caso do “cp”.

+ o arquivo é “transferido” para o diretório-destino.

+ o nome do diretório-destino é o último parametro da linha de comando.

+ assim como o comando “cp”, o “mv” pode mover vários arquivos para um diretório, e a sintaxe é análoga à do comando “cp” ... (lembra ?)

mv arquivo1 arquivo2 ... arquivoN Dir.-Destino

Quinta lição : Arquivos

listar

```
$ ls ↵
cartas   compras   provas
$ ls  provas ↵
feira
$ ls  cartas ↵
compras   feira   xpto
$ □
```

+ o comando “ls” (list status) tambem pode ser usado com parametros.

+ caso o nome passado seja de um diretório, o “ls” mostra o conteúdo (os arquivos e diretórios) deste.

+ o comando “ls” (assim como a esmagadora maioria dos comandos UNIX) tambem aceita parametros especiais, chamados “flags” que são usados para “variar” o seu comportamento e o seu resultado (seu “output”).

Quinta lição : Arquivos

listar

```
$ ls ↵
cartas   compras   provas
$ ls  -F ↵
cartas/  compras   provas/
$ □
```

+ a flag "-F" (o "F" é MAIÚSCULO !) faz com que o "ls" dê uma "dica" sobre a natureza dos nomes mostrados :

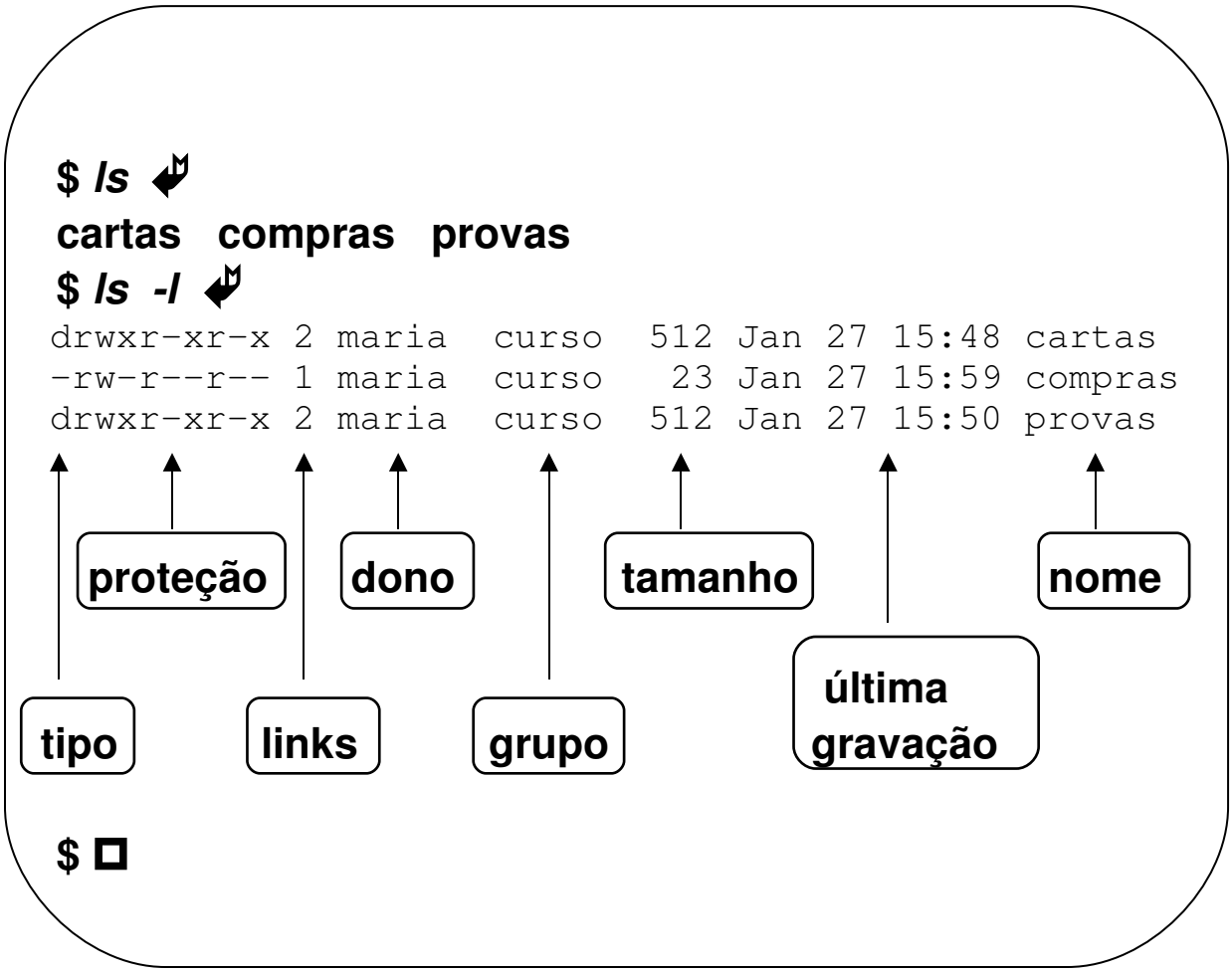
barra	**- o nome ao lado é de um diretório**
asterisco	**- o nome é de um arquivo "executável"**
"nada"	**- trata-se de um arquivo "comum".**

```
$ ls ↵
cartas   compras   provas
$ ls  -l ↵
drwxr-xr-x 2 maria  curso  512 Jan 27 15:48 cartas
-rw-r--r-- 1 maria  curso   23 Jan 27 15:59 compras
drwxr-xr-x 2 maria  curso  512 Jan 27 15:50 provas
$ □
```

+ a flag "-l" (o "L" é minúsculo) faz com que o "ls" mostre informações detalhadas ("tipo", proteção, dono, grupo, tamanho, e data de última gravação) de status.

Quinta lição : Arquivos

listar



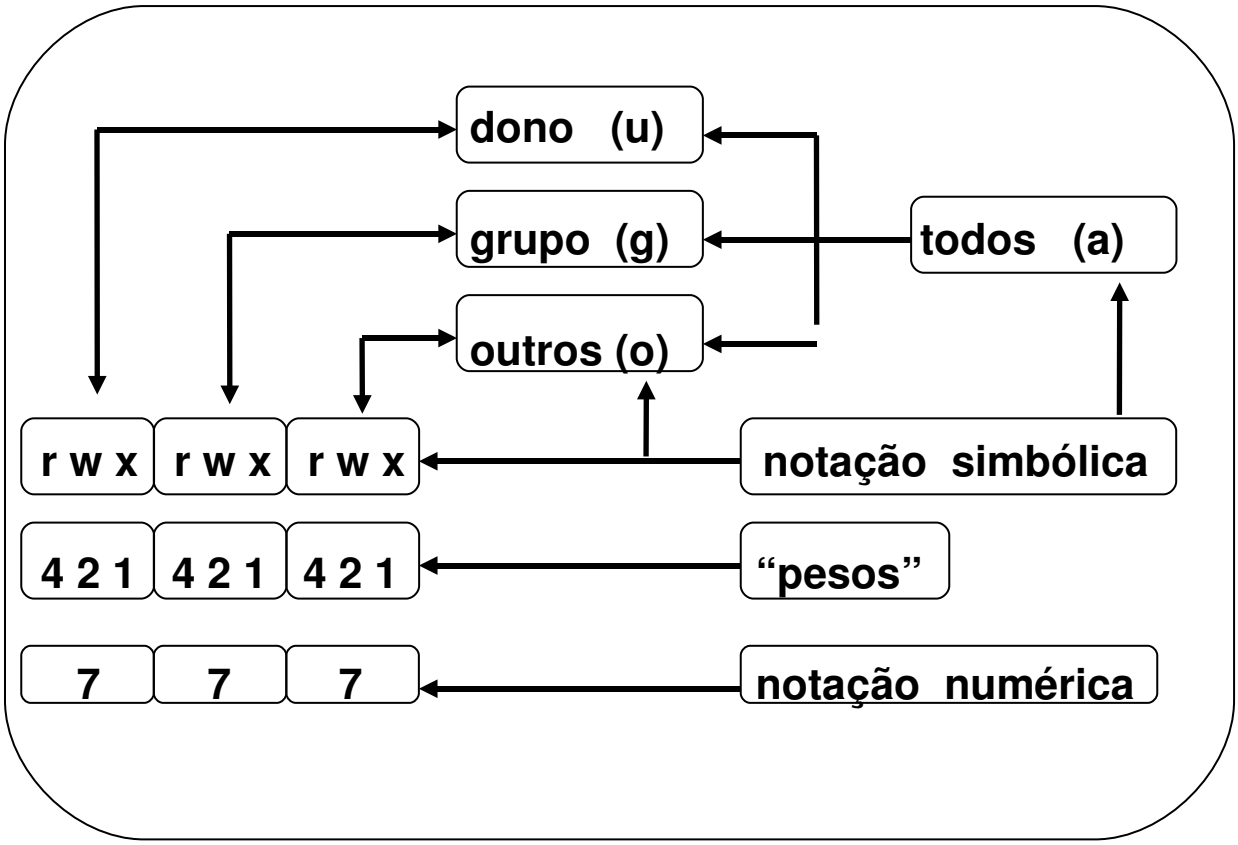
+ o tipo : “d” é diretório ; “-” é arquivo.

+ no diretório “/dev” são encontrados arquivos com os tipos “c”, “b”, “p” e “s” - estes são arquivos especiais.

+ outro arquivo especial é o do tipo “l” (“L” minúsculo), que pode ser encontrado na raiz.

Quinta lição : Arquivos

listar



+ no UNIX, um arquivo pode ser lido (“r” de read), gravado (“w” de write), ou executado (“x” de execute).

+ podem acessar um arquivo : o seu dono (“u” de user), membros do seu grupo (“g” de group), ou outros usuários (“o” de others).

+ as permissões podem ser interpretadas numericamente ou simbolicamente.

Quinta lição : Arquivos

listar

+ a proteção : 3 grupos de 3 letras ...

  + as 3 letras :

    “r” (read) permissão de leitura
    “w” (write) permissão de gravação
    “x” (execute) permissão de execução
    “-” (hífen) respectiva permissão negada

  + os 3 grupos :

    primeiro relativo ao dono do arquivo
    segundo relativo ao grupo do arquivo
    terceiro aplica-se aos “outros”

+ o número de “links” (ou contador de referências) :

  + o UNIX permite que um arquivo possua mais de um nome ...

  + para arquivos, o número de links diz quantos nomes um arquivo tem.

  + para diretórios que não possuem “diretórios-filhos”, o número de links é sempre 2.

  + se algum diretório tem “links” maior que 2, a diferença (links - 2) é o número de diretórios-filhos.

Quinta lição : Arquivos

listar

+ o dono :

  + todo arquivo ou diretório possui um único dono.

  + o USER ID deste usuário fica “carimbado” nos arquivos e diretórios por ele criados.

  + o “nome” mostrado pelo “ls” é o seu LOGNAME (sua identificação de login).

+ o grupo : no “status” de todo arquivo ou diretório, fica tambem registrado um GROUP ID que, geralmente (não obrigatoriamente) é o grupo primário do seu dono.

+ o tamanho : em bytes, inclusive separadores-de-linha no caso de arquivos-texto.

+ a data de última gravação :

  + para um arquivo, indica a última vez que ele foi “atualizado”.

  + para um diretório, indica a ultima vez que um de seus “filhos” (arquivo ou diretório) foi criado, deletado, renomeado, ou movido).

Sexta lição : Segurança

Considerações

+ um arquivo possui um único dono, e seu USER ID fica "carimbado" no status do arquivo.

+ no status tambem fica "carimbado" um GROUP ID (que, geralmente, é o grupo primário do seu dono).

+ quem não é o dono do arquivo nem é do seu grupo, são os “outros”.

+ quando um usuário tenta acessar um arquivo (para ler, gravar ou executar) :

  + o UNIX checa se o USER ID do usuário é o mesmo do arquivo (trata-se do dono).

    + neste caso o UNIX consulta o primeiro conjunto de permisões (“u”).

  + senão, o UNIX checa se o GROUP ID do usuário é o mesmo do arquivo (trata-se de alguem “do grupo”).

    + neste caso o UNIX consulta o segundo conjunto de permisões (“g”).

  + se nenhum teste deu certo, o UNIX consulta o terceiro conjunto de permissões (“o”) porque, com relação a este arquivo, o usuário faz parte dos “outros”.

Sexta lição : Segurança

Conclusões

+ podem (tentar) acessar um arquivo (ou um diretório, ou um arquivo especial) :

  + o dono
  + membros do seu grupo
  + os outros

+ os tipos de acesso a um arquivo são :

  + leitura
  + gravação
  + execução

+ os tipos de acesso a um diretório são :

  + leitura (para saber seu conteúdo)
  + gravação (para criar, deletar, renomear membros)
  + pesquisa (para "entrar nele", ou “passar por ele”)

+ os tipos de acesso a um arquivo especial são :

  + leitura
  + gravação
  + (faz sentido ?)

Sexta lição : Segurança

+ como consultar as permissões de um arquivo ?

```
$ ls -F ↵
cartas/  compras  provas/
$ ls -l compras ↵
-rw-r--r-- 1 maria  curso   23 Jan 27 15:59 compras
$ □
```

+ como consultar as permissões de um diretório ?

```
$ ls -F ↵
cartas/  compras  provas/
$ ls -l provas ↵
-rw-r--r-- 1 maria  curso   23 Jan 27 16:02 feira
$ ls -l -d provas ↵
drwxr-xr-x 2 maria  curso  512 Jan 27 16:08 provas
$ □
```

+ com a flag “-d”, o “ls” mostra o status do diretório, e não do seu conteúdo.

Sexta lição : Segurança

+ para mudar as permissões de um arquivo ou diretório, usa-se o comando “chmod” (change mode) ...

chmod *permissões* nome

+ as permissões podem ser passadas na forma numérica ou na forma simbólica.

+ a forma numérica é composta de tres dígitos (um para cada conjunto de permissões - dono, grupo, outros), sendo que cada dígito é o resultado da soma dos pesos das permissões “ligadas”.

+ supondo que a permissão desejada é :

```
rw-  r--  ---
```

+ para compor a notação numérica :

rw-	r--	---
4 2 1	4 2 1	4 2 1
6	4	0

+ o comando seria :

chmod 640 nome

Sexta lição : Segurança

+ a notação simbólica tem a forma :

+ “quem” : uma ou mais letras, indicando o conjunto ao qual serão aplicadas as permissões.

“u” (user) - dono
“g” (group) - grupo
“o” (others) - outros
“a” (all) - todos

+ “pode” : apenas uma letra, indicando como a permissão será aplicada.

“+” (tambem pode) - a permissão é concedida, e as demais permanecem inalteradas.

“-” (não pode) - a permissão é negada, e as demais ficam inalteradas.

“=” (só pode) - a permissão é concedida, e as demais são todas revogadas.

+ “o que” : uma ou mais letras, indicando o tipo de acesso

“r”(read) - leitura.
“w” (write) - gravação.
“x” (execute) - execução.

Sexta lição : Segurança

+ supondo que a permissão desejada é :

```
rw-  r--  ---
```

+ para compor a notação simbólica :

```
rw-  r--  ---
```

```
u=rw   g=r    o-rwx
```

+ o comando seria :

chmod u=rw,g=r,o-rwx nome

+ NÃO devem haver “espaços” nas permissões !

Sexta lição : Segurança

+ considerando a permissão :

```
rw-  r--  ---
```

+ para “adicionar” aos “outros” o direito de leitura :

```
rw-  r--  r--
```

+ compondo a notação simbólica ...

```
rw-  r--  r--

          o+r
```

+ o comando seria :

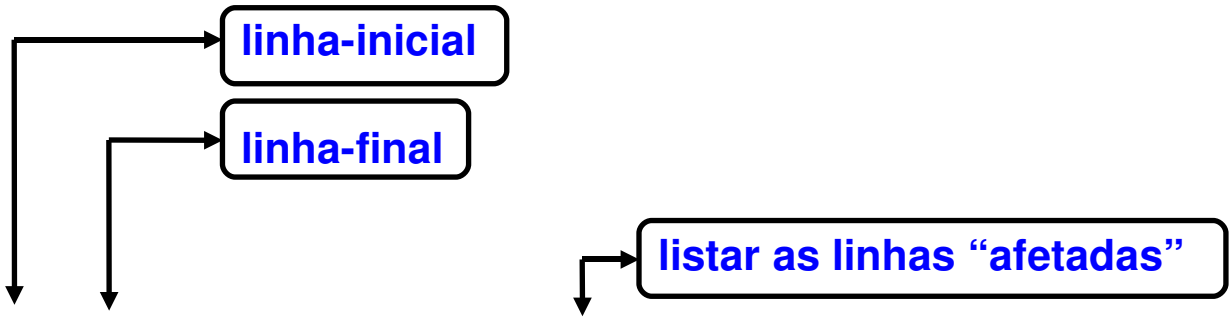
chmod o+r nome

Sétima lição : Editando texto

o editor “ed”

+ é um editor “de linha” (e não “de tela”) e deve ser visto como “último recurso” (para edições “de emergência”), nos casos em que voce não tem o seu editor “favorito”.

+ a forma-geral dos seus comandos é :



[i [, f]] [comando] [p]

+ os colchetes indicam partes opcionais da sintaxe.
+ as linhas são :

números inteiros

$ (última linha)

. (ponto, significa “a linha atual”)

.+N (N linhas a partir da atual)

.-N (N linhas antes da atual)

+ uma coisa muito chata é que o “ed” não tem “prompt” !

Sétima lição : Editando texto

o editor “ed”

+ comandos mais usados :

1,$p	**lista todo o texto**
1,$n	**idem, numerando as linhas**
1	**“vai” para a primeira do texto**
$	**“vai” para a última linha do texto**
w	**salva o arquivo**
q	**termina o editor**
/texto/	**procura (para frente) a linha que contem “texto”**
?texto?	**procura (para trás) a linha que contem “texto”**
i	**insere texto antes da linha atual (termina com uma linha contendo apenas “.”)**
a	**acrescenta (append) texto após a linha atual. (tambem termina com “.”)**
d	**deleta linha**
c	**muda linha (deleta e entra em append)**
s/velho/novo/p	**substitui “velho” por “novo”.**

Sétima lição : Editando texto

o editor “ed”

```
$ ed compras ↵
23
1,$p ↵
morango
creme chantili
q ↵
$ □
```

```
$ ed compras ↵
23
1,$p ↵
morango
creme chantili
1i ↵
banana ↵
. ↵
1,$p ↵
banana
morango
creme chantili
w ↵
30
q ↵
$ □
```

Sétima lição : Editando texto

o editor “vi”

+ tem 2 "modos de operação" :

+ modo "entrada de texto"
+ modo "comandos" (inicial, e default)

+ entrar :

$ vi nome-do-arquivo [enter]

+ sair (abandonando incondicionalmente) :

: q ! [enter]

+ sair (salvando incondicionalmente) :

: x ! [enter]

+ entrar com texto (terminam com ESC) :

i (insere texto a partir do cursor)
I “i maiúsc.” (insere texto no início da linha atual)
o (abre uma linha nova abaixo da atual)
O (abre uma linha nova acima da atual)
A (acrescenta texto ao final da linha atual)

+ salvar sem sair :

: w ! [enter]

Sétima lição : Editando texto

o editor “vi”

+ movimentar-se pelo texto :

k (ou seta p/ cima)

j (ou seta p/ baixo)

l (“L minúsc.” ou espaço ou seta p/ a direita)

h (ou "backspace" ou seta p/ a esquerda)

^F (ou "page down")

^B (ou "page up")

w (próxima palavra)

b (palavra anterior)

0 (início da linha)

$ (fim da linha)

1G (inicio do arquivo)

G (fim do arquivo)

#G (ir para a linha #)

^G (informar o número da linha atual)

Sétima lição : Editando texto

o editor “vi”

+ procurar texto :

/texto [enter] (procura para frente)

?texto [enter] (procura para trás)

/ (repete a pesquisa)

? (idem, para trás)

+ substituir texto (mesmo sem procurar) :

(*) estes comandos deletam caracteres e entram em modos de "inserção" - precisam terminar com ESC !

s deleta o caractere sob o cursor,
e passa ao modo "inserir texto" (comando "i")

#s substitue # caracteres a partir do cursor
pelos caracteres que serão digitados

cw deleta até o fim da palavra atual,
e passa ao modo "i"

c$ deleta até o fim da linha atual,
e passa ao modo "i"

Sétima lição : Editando texto

o editor “vi”

+ deletar texto :

(*) nestes comandos, o texto deletado é copiado para um "buffer", e pode ser recuperado posteriormente.

x (deleta o caractere sob o cursor)

#x (deleta # caracteres a partir do cursor)

dw (deleta até o fim da palavra)

dd (deleta a linha atual)

#dd (deleta # linhas a partir da atual)

+ copiar texto :

yy copia a linha atual para o buffer

#yy copia # linhas a partir da atual para o buffer

P insere o conteúdo do buffer antes da linha atual

p insere o conteúdo do buffer antes da próxima linha

Sétima lição : Editando texto

o editor “vi”

+ miscelânea :

^l [Ctrl-L] (refaz a tela)

. [ponto] (re-executa o último comando)

u (desfaz o último comando)

U (restaura a linha atual)

Oitava lição : o shell

expansão de nomes

+ o shell reconhece alguns carecteres especias como "coletivos" para nomes de arquivos e diretórios ...

+ ao passar nomes de arquivos ou diretórios como parametros para comandos (tipo "ls", "rm", ...) estes caracteres ajudam a formar "padrões de busca" que farão com que o comando atue num conjunto de nomes ...

? - equivale a "qualquer caractere".

exemplos :

"??????" nomes com 6 caracteres quaisquer

"doc??" nomes começando por "doc", seguido por 2 caracteres quaisquer

"???.txt" nomes começando por 3 caracteres quaisquer seguidos de ".txt"

"arq?.bak" nomes que começam com "arq", seguido de 1 caractere qualquer, e terminam com ".bak"

+ experimente estes comandos :

ls /bin/????

ls -ld /???

Oitava lição : o shell

expansão de nomes

*** (asterisco) - “qualquer número de qualquer caractere”.**

exemplos:

“*”	**todos os nomes (de arquivos e diretórios) do diretório atual**
“/bin/*	**todos os nomes no diretório “/bin”**
“doc*”	**nomes que começam por “doc” (seguido de “qualquer coisa”)**
“*.bak”	**nomes que terminam com “.bak” (e começam com qualquer coisa)**
“xi*.txt”	**nomes que começam com “xi” e terminam com “.txt”**

+ experimente os comandos :

ls -ld c*

ls -ld /bin/c* /bin/rm* /bin/l*

ls -ld /lib/*.a

Oitava lição : o shell

expansão de nomes

+ alem do "?" que significa "qualquer caractere", o shell tambem pode reconhecer "alguns caracteres", numa notação chamada *"Classe de caracteres"* ...

[C1 C2 ... Cn] - equivale a "um dos caracteres entre os colchetes".

[C1 - C2] - "qualquer caractere entre C1 e C2" (inclusive)

+ exemplos :

"[dt]*"	**nomes começando com "d" ou "t"**
"*.[ach]"	**nomes terminando com "." seguido de "a" ou "c" ou "h"**
"arq[0-9]*"	**nomes começando com "arq", seguido de um dígito (de 0 a 9)**
"*[fx][17][4-8]"	**nomes terminando com "f" ou "x", seguido de 2 dígitos : o primeiro sendo 1 ou 7, e o segundo entre 4 e 8**

+ experimente :

```
ls -ld /[b-e]??

ls -ld /bin/?[dis]*
```

Oitava lição : o shell

redirecionamento de output

+ qualquer comando que produz “output” (tipo “who”, “ls”, “date, ...) escreve o seu output no vídeo ...

+ o shell permite “capturar” este output, “desviando-o” para (“dentro de”) um arquivo ...

+ se o arquivo não existe, ele é criado pelo shell.

+ se ele já existe, o seu conteúdo prévio é “perdido” ! (redirecionamento de output é “destrutivo” !)

```
$ date ↵
wed  may 25  01:20:30  BST  1995
$ ls ↵
cartas  compras  provas
$ date > horas ↵
$ ls ↵
cartas  compras  horas  provas
$ cat horas ↵
wed  may 25  01:20:35  BST  1995
$ □
```

+ não faz sentido redirecionar o output de um comando que não produz output (como o “rm”) ...

+ desvio múltiplo (“ ... > arq1 > arq2 ”) tambem não dá ...

Oitava lição : o shell

redirecionamento de input

+ qualquer comando que pede “input” do usuário (“write”, “cat”, “mail”, ...) recebe este input “do teclado”, e ele deve terminar com “^D” ([Ctrl-D] , control-D).

+ o shell permite redirecionar este input, usando o conteúdo de um arquivo como substituto do teclado.

```
$ ls ↵
cartas  compras  horas  provas
$ cat < compras ↵
banana
morango
creme chantili
$ write debora < compras ↵
$ □
```

+ não faz sentido redirecionar o input de comandos que não pedem input (“date”, “ls”, “who”, ...)

+ redirecionamento múltiplo (“ ... < arq1 < arq2 ”) não é assim que se faz, mas é perfeitamente possível ...

+ redirecionamento mixto (“ ... < arq1 > arq2 ”) é normal.

Oitava lição : o shell

pipes

+ “pipe” (ou “duto”) é o mecanismo que o shell oferece para desviar o output de um programa para o input de outro.

+ num pipe, dois programas são executados, um deles é do tipo “produtor de output” (“ls, “who”, ...) enquanto o outro é do tipo “consumidor de input” (“write”, “sort”, ...)

+ o output do primeiro programa (que normalmente iria para o vídeo) é passado (pelo shell) ao segundo programa (que iria esperar seu input “do teclado”) ...

```
$ ls ↵
cartas  compras  horas  provas
$ ls | write debora ↵
$ □
```

+ com pipe, é possível fornecer mais de um arquivo como input para um programa ...

```
$ ls ↵
cartas  compras  horas  provas
$ cat horas compras | write debora ↵
$ □
```

Oitava lição : o shell

descritores de arquivo

+ sempre que um programa precisa manipular um arquivo, o usuário passa o nome do arquivo ao programa.

+ porem, internamente o UNIX manipula arquivos através de estruturas chamadas “descritores de arquivo”.

+ todo programa já nasce conhecendo 3 descritores de arquivo, que são preparados pelo shell e passados ao programa na hora da execução :

  0 chamado “entrada padrão” (“standard input” ou “stdin”) e associado ao teclado.

  1 chamado “saída padrão” (“standard output” ou “stdout”) e associado ao vídeo.

  2 chamado “saída padrão de erros” (“standard error” ou “stderr”) e tambem associado ao vídeo.

+ o redirecionamento de output (“ > ”) se aplica à saida padrão (o vídeo)

+ o redirecionamento de input (“ < ”) se aplica à entrada padrão (o teclado)

+ para redirecionar a saída padrão de erro, é preciso usar o descritor no comando (“ ... 2 > arquivo ”)

Oitava lição : o shell

mensagens de erro

+ quando um programa não consegue atingir seu objetivo, ele escreve uma mensagem de erro no vídeo.

+ o redirecionamento de output (" > ") não consegue capturar estas mensagens, pois elas não estão sendo enviadas para a "saída padrão", e sim para a "saída padrão de erros".

+ para capturar mensagens de erro dos comandos, é preciso redirecionar a saída padrão de erros (" 2 > ").

```
$ ls -l /bin/bosque ↵
ls : can't find /bin/bosque
$ ls -l /bin/bosque 2 > bronca ↵
$ ls ↵
bronca cartas compras horas provas
$ cat bronca ↵
ls : can't find /bin/bosque
$ □
```

Oitava lição : o shell

processos

+ um processo é um programa em execução.

+ comandos, em geral, são programas, que são executados como processos.

+ assim como os arquivos, os processos possuem um dono, que quase sempre é o usuário que executou o programa.

+ processos são identificados pelo UNIX com um número chamado PROCESS ID (PID).

+ processos (comandos) em geral, estão “associados” a um terminal.

+ existem processos “administrativos” (spool, tcp/ip, etc ...) cujo dono é o “root” e não possuem terminal associado.

+ quando o programa termina, o processo “morre”.

+ para consultar informações sobre processos “ativos” (em execução) usa-se o comando “ps” (process status).

+ processos podem ser interrompidos durante a execução.

+ para interromper (cancelar) um comando executando no seu terminal, pressione “Ctrl-C”.

+ para cancelar um processo qualquer (que seja “seu”), use o comano “kill”.

Oitava lição : o shell

ps

```
$ ps ↵
PID    TTY    TIME    CMD
1234   tty5   00:01   sh
5678   tty5   00:00   ps
$ ■
```

kill

+ o comando “kill” envia “sinais” a um processo.

+ “sinais” são mensagens identificadas por números, que correspondem a “eventos”.

+ por exemplo: quando voce pressiona “Ctrl-C” em um terminal, o UNIX envia o sinal 2 (“interrupt” ou “SIGINT”) ao processo que estiver sendo executado no terminal naquele momento.

+ ao receber um sinal, um processo geralmente “morre”, a menos que ele esteja programado para “capturar” o sinal (exemplo: o shell não morre com “Ctrl-C”).

+ há um sinal que não pode ser “capturado” : o número 9 (“SIGKILL”).

+ para cancelar (terminar) um processo “infalivelmente”, usa-se o comando “kill” com o sinal 9.

kill -9 *PID*

Oitava lição : o shell

background

+ ao executar um comando, o “prompt” não reaparece (e o seu terminal fica “bloqueado”) até que o comando acabe (o shell, normalmente, “espera” o processo terminar).

+ é possivel fazer com que um comando seja executado “livre” do terminal (ótimo para comandos “demorados”).

+ este modo de execução é chamado “background”, ou “retaguarda” (o shell não espera o processo terminar).

+ para executar um comando “em background”, é preciso redirecionar seu “input / output padrão”, uma vez que ele não terá acesso ao terminal.

+ para executar um comando “em background”, coloque o caractere “&” ao final da linha de comando ...

```
comando  <  arqent  >  arqsai  2 >  arqerr  &
```

+ o shell informa o PID do processo “background” logo após o comando ser transmitido (e em seguida coloca o prompt, disponibilizando o terminal).

+ o PID deve ser anotado para consultas (via “ps”) ou para cancelamento (via “kill”).

+ quando voce “sai de sessão”, seus comandos que estão em background são cancelados ! (a menos que voce use o comando “nohup” ...)

Oitava lição : o shell

nohup

+ o comando “nohup” serve para “preservar” os comandos em background quando vode sai de sessão.

+ para isto, coloque o comando “nohup” no início da linha de comando (não esqueça o “&” no final).

nohup comando < arqent > arqsai 2 > arqerr &

+ o shell informa o PID do processo “background” logo após o comando ser transmitido (bem abaixo da linha de comando enviada).

+ anote este PID para uso futuro (com “ps” ou “kill”) ...

+ em seguida reaparece o prompt, uma vez que o comando será executado em background, e o shell não precisa ficar esperando o processo terminar para ler mais comandos.

+ este comando então não será cancelado quando voce sair de sessão ...

+ é recomendável “acompanhar” o progresso de comandos em background, para prevenir “loops” e “deadlocks”.

+ isto se faz com o “ps” (a coluna “time”), ou inspecionando o tamanho (e eventualmente o conteúdo) de seus arquivos de output ...

Oitava lição : o shell

programação (introdução)

scripts
echo
`substituição de comando` (crase)
variáveis de ambiente
parametros da linha de comando
read

for-do-done
break
continue
if-then-else-fi
test
while-do-done
case-in-esac

Nona lição : utilitários úteis

more

+ o comando “more” lê da entrada padrão e escreve na saída padrão, “parando” a cada tela cheia.

+ é usado para examinar arquivos grandes.

```
more   arquivo
```

ou

```
more   <   arquivo
```

+ tambem é usado (via “pipe”) para “paginar” grandes outputs de comandos.

```
comando   |   more
```

+ outro “paginador de output” análogo ao more é o “pg”.

sort

+ sua função é “sortar” (ordenar, classificar) as linhas que recebe da entrada padrão, escrevendo-as na saída padrão.

+ pode ser usado com “nome de arquivo” ou “ < ” ou “ | ” ...

+ experimente “ who | sort ” ...

Nona lição : utilitários úteis

wc

+ o resultado (o output) do comando “wc” (word count) é uma contagem de linhas, palavras e caraceres que foram lidos.

+ pode ser usado com “nome de arquivo” ou “ < ” ou “ | ” ...

+ seu comportamento “default” é mostrar os 3 valores.

+ usando-se as flags “-l”, “-w” ou “-c” ele mostra apenas (respectivamente) linhas, palavras ou caracteres.

grep

+ o comando grep (global regular expression print) pesquisa texto em arquivos (ou na entrada padrão) e escreve as linhas encontradas na saída padrão.

grep “texto” arquivo

comando | grep “texto”

+ usando a flag “-i” a pesquisa não diferencia maiúsculas de minúsculas.

Nona lição : utilitários úteis

find

+ este comando “procura” arquivos.

+ o critério de busca pode ser nome, dono, grupo, tamanho, data, permissões, ...

+ a busca é feita a partir de um ou mais diretórios.

+ se algum dos diretórios possuir sub-diretórios, estes tambem são pesquisados, recursivmente, “ad nauseam”.

+ quando um arquivo é encontrado (satisfaz o critério de busca) a atitude default do “find” é escrever o nome deste arquivo na saída padrão.

+ outra atitude possível é executar um comando, passando o nome do arquivo encontrado como parametro para este comando.

```
find  /usr  -name  ‘ *.bak ’  -print

find  /etc  -name  ‘ fs* ’  -exec  ls  -ld  { }  \ ;

find  /tmp  -name  ‘ *.Lix ’  -exec  rm  { }  \ ;
```

Nona lição : utilitários úteis

correio eletrônico

+ o comando “mail” envia mensagens para a caixa-postal de outro usuário, e tambem é usado para examinar a correspondencia que chega.

+ ao entrar em sessão, voce é notificado (através do famoso “you have mail”) se houver correspondencia.

preparar e enviar carta

+ o comando mail recebe como parametros a lista de usuários “destinatários”, e lê a carta da entrada padrão.

+ pressionando “Ctrl-D” o texto da carta é encerrado e a carta é enviada.

checar e ler cartas

+ usado “puro” (sem parametros) ele examina sua caixa postal e mostra sua correspondencia.

+ neste modo “interativo”, o prompt do mail é um “&”.

+ pressionando [Enter] ele mostra a próxima carta.

+ pressionando “Ctrl-D” ele termina.

Décima lição : miscelanea

spool

mandar para a fila (default) de impressão

lp arquivo
comando | lp

mandar para uma fila de impressão qualquer

lp –d nomedafiladeimpressão arquivo
comando | lp –d nomedafiladeimpressão

checar a fila de impressão

lpstat –t

cancelar impressão

cancel jobname

umask

checar

umask

(*) a mascara é mostrada em octal ...

interpretar

“a mascara é o inverso das permissões” ou então

“a mascara mostra o que deve ser negado”

ex.: mascara = 027

o inverso é 750 (vejamos ...)

```
027 (8) = 000 010 111 (2)
750 (8) = 111 101 000 (2)
750 (8) = rwx r-x --- (t)

(*) diretórios serão criados com estas
    permissões, porem arquivos criados
    terão os bits “x” desligados ...
```

mudar

umask mascara

o arquivo “.profile”

→ é um shell script “executado” quando voce entra em sessão ...

exemplo (simplificado) :

$ cat $HOME/.profile

```
PS1=”$PWD > ”
export PS1

PATH=/usr/bin:/etc:$HOME/bin:/usr/bin/X11:/sbin:.
export PATH

set –o vi

stty erase "^H" kill "^U" intr "^C" eof "^D"

date
uname –a

alias l='ls –a'
alias ll='ls –la'
alias lc='ls –CFa'
alias cls=clear
alias dir=ll
alias del=rm
alias md=mkdir
alias rd=rmdir
```

$ O