Sistemas Operacional Unix

Prof. Celso Maciel da Costa

Histórico

- - Projeto Multics (1965)
  - MIT
  - Bell Laboratories
  - GE
- - Bell se retira do projeto (1969)
  - sistema grande
  - complexo
  - características batch

Histórico

Ken Thompson

. Retornou com idéias interessantes

. Escreveu um programa para rodar no PDP-7

. Pequeno sistema operacional

Histórico

Prmeira versão do UNIX

- - 1969
- - Idéias interessantes para pesquisa e desenvolvimento de software
- - Em linguagem de montagem do PDP-7

Histórico

Unix no PDP 11/20

- 1971
- monoprogramável

Histórico

Unix reescrito em C (Dennis Ric

. 1973

. portável

. claro

. fácil manutenção

Histórico

System lll

. 1981

. suporte da AT&T

System V

. 1983

. suporte da AT&T

Histórico

1989

- System V release 4

- features da versão 4.3 BSD e SunOS

Histórico BSD

1978

- versão 7 -> 3BSD
- memória virtual
- financiamento da defesa americana (DARPA)

Histórico

1979

- 4BSD
- redes: . suporte ao protocolo TCP/IP
. locais: Ethernet
. não locais: NFSNET
- novas funções para terminais
- nova interface (Cshell)
- novos editores: vi
- compiladores Pascal e Lisp

Histórico

Histórico outros sistemas

Xenix (1981)

- segunda importante versão comercial
- Microsoft, para PC
- a partir da versão 7

Aix

- IBM
- Várias plataformas

Histórico

Macintosh a Cray

- inúmeras plataformas
- menor atenção para confiabilidade e facilidades de uso
- ideal: qualquer aplicação Unix em qualquer sistema Unix
- necessidade de padrões

Histórico

Razões de sucesso

- simplicidade

- portabilidade
. escrito em linguagem C
. primeiro não assembler

- código fonte disponível
. auxílio de diversos grupos
. correção local e imediata de erros

Histórico

- Possui uma linguagem de comandos fácil de usar
e que oferece grandes recursos

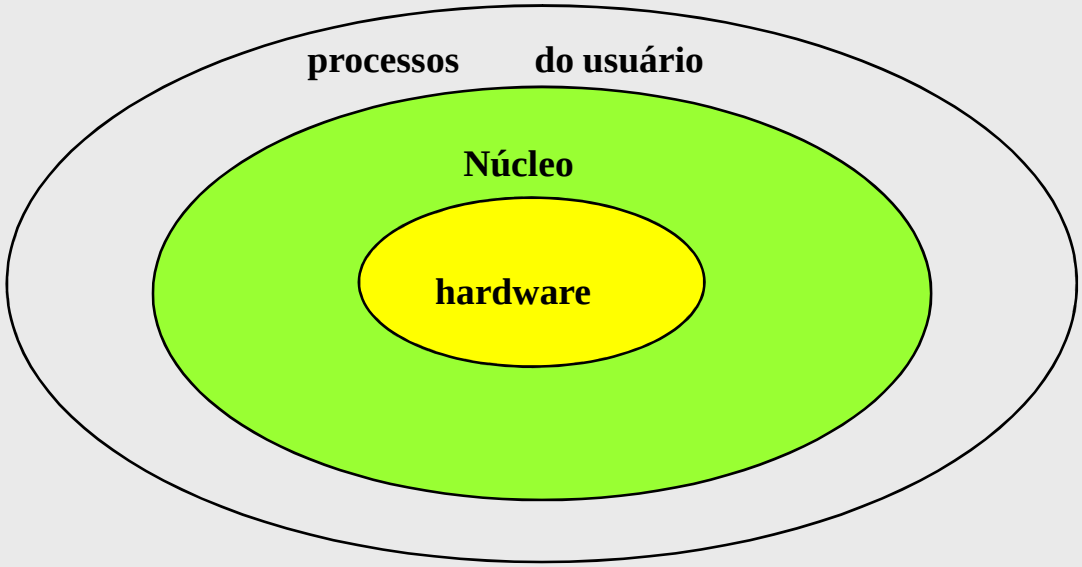
- Possui uma grande quantidade de programas
utilitários

- Oferece um bom compartilhamento de informação
. bom ambiente para trabalho em grupo

- Facilidades para criação de novas ferramentas

A Organização Interna do Unix

Arquitetura do Unix



Arquitetura

Usuário

- pessoas interagindo com o sistema (interpretador de comandos)

- os processos do usuário invocam o núcleo através de chamadas de funções primitivas do sistema

Arquitetura

Núcleo

- Atendimento de Interrupções

- Atendimento de chamadas de sistema

- Gerência de Arquivos

- Gerência de processos

- Gerência de memória

- Gerência de entrada e saída

Perspectiva do Usuário: Shell e Sistema de Arquivos

Shell

- **linguagem de comandos**

- **interface entre o usuário e o sistema operacional**

Arquitetura

Sistema de arquivos

- hierárquico

- cada usuário tem seu próprio diretório

- o usuário pode organizar sub-diretórios

Unix

Acesso ao Sistema

sessões de trabalho

Universidade´Estadual do Rio Grande do Sul
Celso Maciel da Costa

Sessões de Trabalho

- *user* (conta)
  - cada usuário possui uma palavra reservada identificando uma conta, um usuário específico
  - um usuário pode possuir *N* contas, embora normalmente possua apenas uma
  - obrigatório acessar o sistema através de uma conta
  - alterações nas contas através do super-usuário

Sessões de Trabalho

- *password* (senha)
  - palavra reservada associada com cada usuário
  - usuário escolhe uma senha
  - uma função encripta a senha e armazena o resultado em um arquivo
  - a função não permite fazer o caminho inverso
  - pede-se ao usuário a sua senha, sendo a mesma função executada e o resultado comparado com o valor armazenado

Sessões de Trabalho

- super-user
  - nome básico: “root”
  - direito de administrar o sistema - “administrador do sistema”
- grupo
  - diversas contas agrupadas por um determinado nome registrado
  - diversos usuários trabalhando em um mesmo projeto

Sessões de Trabalho

- processo de “login” (abertura de sessão)
  - aparece uma mensagem solicitando a entrada do nome da conta (*username* ou *login*)
  - usuário fornece seu *username* e sua senha
    - mensagem de erro: ‘login incorrect’
    - se ok, usualmente aparece data e hora do último *login*, junto a mensagens de aviso da administração do sistema

Primeiros Contatos

- *prompt*
  - indica que o Unix está esperando que o usuário entre um comando
  - configurável, o usual é o '$' para usuários normais e '#' para o *root*
  - informações adicionais como nome do usuário, nome da máquina, hora ou diretório corrente
  - exemplo:
    ```
    usuario@freud: /home/usuario 16 %
    ```

Primeiros Contatos

- regras para formação de senhas
  - não deve ser curta demais (mínimo de 5 ?)
  - máximo de 8 caracteres
  - usar obrigatoriamente pelo menos um desses:
    - uma ou mais letras maiúsculas
    - um ou mais caracteres especiais, como ‘^’, ‘%’, etc.
    - um ou mais números

Primeiros Contatos

- senhas mais comuns são palavras do dicionário, datas...
- programas especiais para quebra de senhas, *crackers*

- encerramento de sessão
  - `‘logout’`
  - `CTRL-d (^d)`
  - `‘exit’`

Primeiros Comandos

Primeiros Comandos

- **Comando echo:**
  - Função: mostrar uma cadeia de caracteres na tela
  - Sintaxe: echo <cadeia de caracteres>
  - Exemplos:
    - **$ echo ola**
    - **ola**
    - **$ echo Curso de Unix Básico**
    - **Curso de Unix Básico**
    - **$**

Primeiros Comandos

- **Comando: banner**
- **Função: imprimir a cadeia de caracteres ampliada e em letras maiúsculas (máximo 10)**
- **Sintaxe: banner <cadeia de caracteres>**
- **Exemplos:**
- $ banner Ola

Primeiros Comandos

Primeiros Comandos

- **Comando: cal**
- **Função: mostrar na tela o**
- **calendário do mês especificado**
- **Sintaxe: cal [mês] [ano]**

Primeiros Comandos

Exemplo:

$cal 6 1994

```
      June 1994
 S   M  Tu   W  Th   F   S
                1   2   3   4
 5   6   7   8   9  10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29  30
```

- OBS: cal sem argumentos mostra o mês atual

Primeiros Comandos

- **Formato (cont.)**
  - **%m: mes 1 - 12**
  - **%M: minuto - 00 - 59**
  - **%r: tempo em notação AM/PM**
  - **%S: segundos - 00 - 59**
  - **%T: hora em hh:mm:ss**
  - **%w: dia da semana - sun=0**
  - **%W: número da semana no ano**
  - **%y: dois últimos dígitos do ano**

Primeiros Comandos

- **Comando: tty**
- **Função:** imprime na saída padrão a identificação do terminal
- **Sintaxe:** tty
  - Exemplo:
    - **$tty**
    - **/dev/tty0**

Primeiros Comandos

- **Comando who**
- **Função:** mostrar os usuários atualmente usando o sistema
- **Sintaxe:** who [argumentos]
  - Argumentos:
    - **-u : mostra os usuários, a identificação do terminal, a data, a hora em que começaram a usar o sistema, o tempo sem utilização, o process-ID do shell e a linha usada pelo terminal**
    - **am i ou am I: mostra informações do próprio usuário**

Primeiros Comandos

- **Exemplos:**
  - $who -u

```
user1    tty0    Jun  2 15:55   .    17893    (143.54.12.47
user2    tty1    Jun  2 16:41   .    17991    (regulus
user3    tty2    Jun  2 16:42   .    18001    (regulus)
```

- **$who am i**
- **user1 tty2 Jun 2 16:42 (regulus)**

Primeiros Comandos

- **Exemplos (cont.)**
  - **$who**

```
user1    tty0    Jun  2 15:55   (143.54.12.47)
user2    tty1    Jun  2 16:41   (regulus)
user3    tty2    Jun  2 16:42   (regulus)
```

SISTEMA DE ARQUIVOS

Universidade´Estadual do Rio Grande do Sul
Celso Maciel da Costa

Arquivos e Diretórios

Organização

- estrutura em árvore, hierárquica
- tipos de arquivos
  - arquivos comuns
    - contém dados (textos, programas, ...) ou imagens executáveis
  - diretórios
    - formato especial, indicam diretórios e arquivos
  - arquivos especiais
    - servem para mapear dispositivos de I/O

Organização

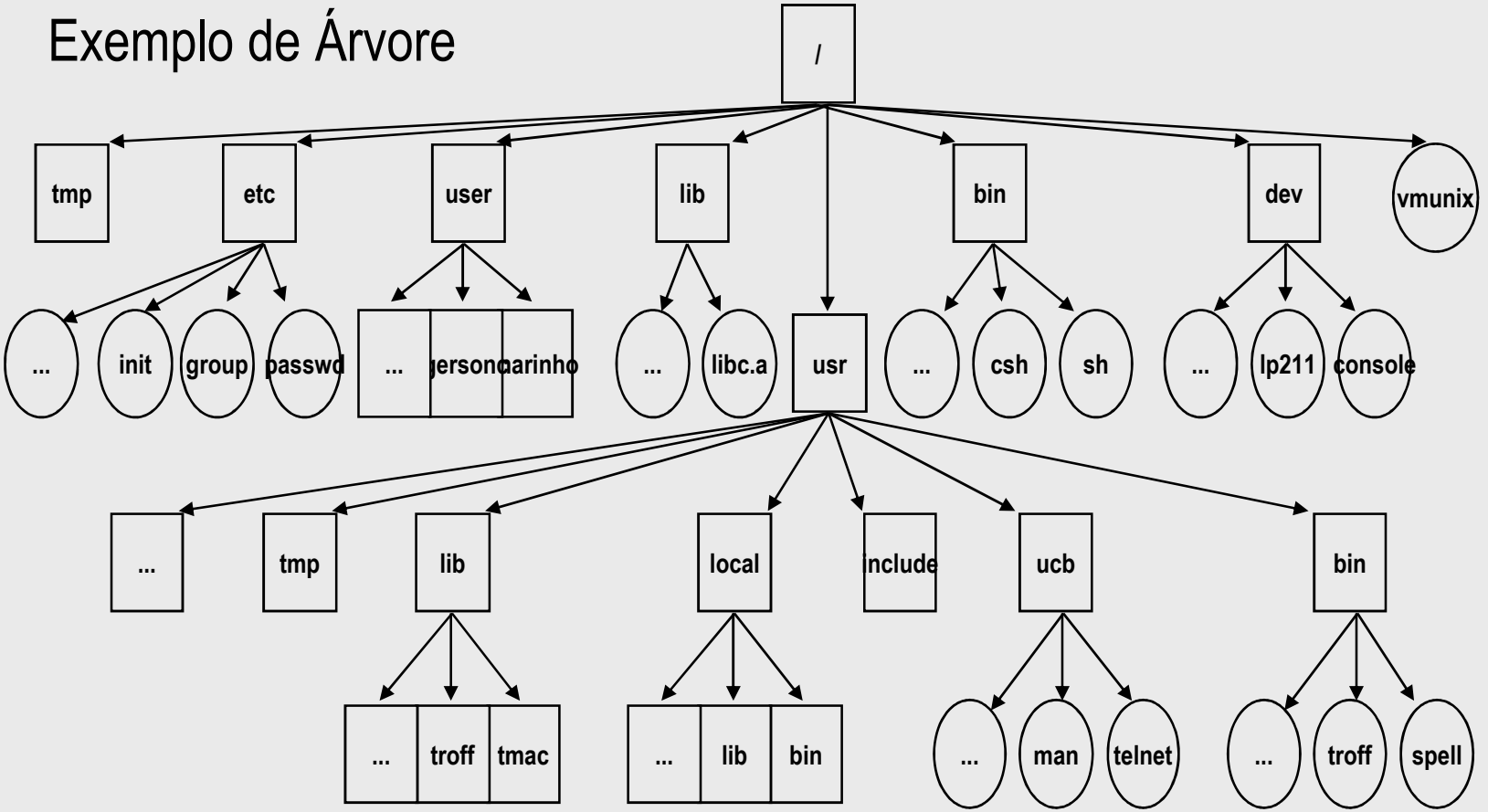
- diretório “HOME”
  - cada usuário possue o seu próprio HOME
  - diretório inicial ao abrir uma sessão
- diretório corrente de trabalho (*current working directory*)
  - usuário “caminha” na árvore de arquivos, alterando diretório corrente
  - atributo de um programa (ex.: *shell*)
  - comando `‘cd’` (*change directory*)

Organização

- **Principais diretórios Unix**
  - **/ - contém a raiz do sistema de arquivos**
  - **/bin - comandos mais usados no sistema**
  - **/dev - dispositivos de entrada e saída**
  - **/etc - arquivos para administrar o sistema**
  - **/lib - bibliotecas do sistema**
  - **/tmp - arquivos temporários do sistema**
  - **/usr - diretórios dos usuários**
  - **/usr/bin - comandos menos usados**
  - **/lost+found - arquivos que o Unix tenta recuperar quando faz um check-up no sistema de arquivos**

Exemplo

- Exemplo de Árvore



Conceitos

- Referências
  - problema como identificar um arquivo?
  - forma completa ou absoluta:
    - diretório raiz: “/”
    - da raiz da árvore ao arquivo ou diretório em questão
    - separador dos elementos: ‘/’
    - exemplo:
      “/usr/waters/wall/numb.doc”

Conceitos

- forma relativa:
  - caminho percorrido a partir do diretório corrente
  - exemplo:
    - diretório corrente: “.” (ex.: “`numb.doc`” = “`./numb.doc`”)
    - diretório pai do corrente: “..” (ex.: “`../rush/xanadu.doc`”)
- comandos e chamadas do sistema usualmente aceitam as duas formas de referência

Conceitos

- nomes de arquivos
  - arquivos escondidos
    - nome começa com ponto (exemplo: “`.profile`”, “`.login`”)
    - mostrados com uma opção especial do listador de diretórios
  - tamanho do nome
    - BSD: até 255 caracteres
    - System V: até 14 caracteres

Comandos de Manipulação de Diretórios

Comandos

- **Comando:** pwd (*print working directory)*
- **Função:** `imprimir o nome do diretório corrente`
- **`Sintaxe:`** `pwd`
  - Exemplo:
    - **$pwd**
    - **/usr/joao/curso**

Comandos

- **Comando:** cd
- **Função:**
  - **troca diretório corrente:**
- **Exemplo:**
  - **se o diretório corrente é** /, **o comando**
  - **$cd usr**
  - **troca o diretório corrente para** /usr

Comandos

- **Comando:** ls
- **Função:** listar conteúdo de diretórios
- **Sintaxe:** ls [-flags] [files...]
  - “l”: lista longa de arquivos
  - “t”: ordena de acordo com data de alteração
  - “d”: imprime apenas o nome do diretório na lista, mas não o seu conteúdo
  - “a”: imprime também arquivos invisíveis
- tipo de arquivo: “-”, “d”, “c”, “b”, “l”, “p”

Comandos

- **Exemplos:**
  - **$ls**
  - **bin export kernel net tmp vol**
  - **dev home lib opt ufsboot**
  - **devices hsfsboot lost+found proc usr**
  - **etc kadb mnt sbin var**

Comandos

Exemplos (cont.):

$ls -l

```
drwxrwxr-x    sys         5632    May 16 09:29 dev
drwxrwxr-x    sys          512    Apr 28 19:47 devices
drwxrwxr-x    sys         2560    Jun  1 16:28 etc
drwxrwxr-x    sys          512    Apr  5 18:49 export
dr-xr-xr-x    root           4    Jun  2 17:47 home
-rw-r--r--    sys       169304    Sep 27  1993 hsfsboot
-rw-r--r--    sys       352832    Apr  5 03:26 kadb
drwxr-xr-x    sys          512    Apr 28 19:26 kernel
```

Comandos

- **Comando:** mkdir [nomedir...]
- **Função:** criação de diretórios: *make directory*
- **Sintaxe:** mkdir [nomedir...]
- **Exemplo:**
  - **$mkdir /usr/curso**

Comandos

- **Comando:** rmdir
- **Função:** remover arquivos e diretórios
- **Sintaxe:** rmdir [-flags] [dir...]
- **Flags:**
  - **p: remove o diretório pai do que se tornou vazio**
- **Exemplo:**
  - $rmdir -p curso/unix
  - **remove o diretório** unix, **se estiver vazio**, **e** curso,
  - **se também se tornar vazio**

Comandos de Manipulação de Arquivos

Comandos

- **Comando:** rm
- **Função:** remoção de arquivos
- **Sintaxe:** rm [-flags] [name...]
  - Flags:
    - **-f : remove todos os arquivos no diretório, sem avisar o usuário**
    - **-i : antes de remover cada arquivo, é pedida uma confirmacão**
    - **-r : remove recursivamente arquivos, diretórios e subdiretórios especificados**
- **Exemplo:**
  - **$ rm -r curso**

Comandos

- **Comando:** cp
- **Função:** copiar arquivos de *sourcefile para target file*
- **Sintaxe:** cp [-flags] sourcefiles... [targetfile | targetdir]
- **Flags:**
  - **-i : solicita confirmação antes de sobrepor em target**
  - **-p : preserva as permissões, datas de modificacão**
  - **-r : faz cópia recursiva de diretórios e subdiretórios**
- **Exemplo:**
  - **$cp myfile newfile**

Comandos

- **Comando**: mv (rename)
- **Função:** movimentação de arquivos
- **Sintaxe:** mv [-flags] sourcefiles... [targetfile | targetdir]
  - Flags:
    - **-f : move sem confirmação**
    - **-i : solicita confirmação antes de mover**
- **Exemplos:**
  - $mv arq1 arq1-save
  - $mv arq1 ../arq/arq1

Comandos

- **Comando: cat**
- **Função:** concatena conteúdo de arquivos: *concatenate*
- **Sintaxe:**cat [files...] [> targetfile]
- **Exemplos:**
  - **$cat filename**
  - **exibe filename no terminal**
  - **$cat filename1 filename2 > filename3**
  - **concatena filename1 e filename2 e escreve o conteúdo em filename3**
  - **cat - > filename**
  - **le do terminal e escreve o conteúdo em filename**

PROTEÇÃO DE ARQUIVOS

Atributos

- atributos (header)
  - conjunto de informações fixas sobre um arquivo
    - exemplos: proprietário, grupo, tamanho, data de criação, tipo
- proprietário: a qual usuário pertence o arquivo
    - *user identifier* - uid
    - na criação do arquivo: assume o uid do processo que cria o arquivo
    - alterável pelo comando *chown* e por função *chown*
- grupo: a que grupo de usuários pertence o arquivo
    - *group identifier* - gid
    - na criação do arquivo: assume o gid do processo que cria o arquivo
    - alterável pelo comando *chgrp* e por função *chown*

Atributos

- modos de proteção
  - atributos de um arquivo que definem as possibilidades de acesso ao mesmo
  - na criação: assume máscara em variável UMASK
  - alterável: pelo comando *chmod* e por função *chmod*
  - 3 níveis
    - proprietário *user, owner*
    - grupo *group*
    - outros *other*
  - 3 tipos de acesso:
    - r leitura *read*
    - w escrita *write*
    - x execução *execution*
  - 3 x 3 = 9 bits

Atributos

- modos de proteção
  - exemplo: "`rwxr-x--x`"
    - `rwx` proprietário pode ler, escrever e executar
    - `r-x` grupo pode ler e executar
    - `--x` outros podem apenas executar
    - para diretórios, o bit `x` assume significado diferente

Atributos

- outros atributos
  - tipo de arquivo
    d - diretório
    b - arquivo especial de bloco
    c - arquivo especial de caracter
    l - arquivo link simbólico
    p - arquivo especial pipe
    - - arquivo comum
  - tamanho (em bytes)
  - data e hora da última modificação
  - data e hora do último acesso

Atributos

- exemplo: saída da execução do comando *ls* com opção *-l*
  - -: arquivo comum
  - rw-: usuário pode ler e escrever
  - r--: grupo pode ler
  - r--: outros usuários podem ler
  - 1: 1 (quantidade de) link
  - waters: nome do usuário
  - pink: nome do grupo
  - 9121: tamanho em bytes
  - mar 3 18:11: mes, dia e hora da última modificação
  - numb.doc: nome do arquivo

```
ls -l
-rw-r--r-- 1 waters pink 9121 mar 3 18:11 numb.doc
```

Comandos

- comando *chown*:
  - função: altera proprietário de um arquivo
  - sintaxe: `chown` **`[-flags] username files...`**
  - opções:
    - -R: atua de modo recursivo sobre os arquivos de diretórios (argumentos `files`)
  - exemplo: **`chown -R paulo fontes`**
    - altera o proprietário dos arquivos do diretório `fontes` (inclusive), de modo recursivo (-R), para `paulo`

Comandos

- comando *chgrp*:
  - função: altera grupo de um arquivo
  - sintaxe: `chgrp` **`[-flags] groupname files...`**
  - opções:
    - -R: atua de modo recursivo sobre os arquivos de diretórios (argumentos `files`)
  - exemplo: `chgrp` **`folha inclusaoFuncion`**
    - altera o grupo do arquivo `inclusaoFuncion` para `folha`

Comandos

- comando *chmod*:
  - função: altera permissões de acesso de um arquivo
  - `sintaxe: chmod` **`[-flags] who[+|-|=]what files...`**
  - opções:
    - -R: atua de modo recursivo sobre os arquivos de diretórios argumentos (files)
    - `who: u=user, g=group, o=other, a=all`
    - `+:` acrescenta permissão
    - `-:` retira permissão
    - `=:` acrescenta permissão indicada (*what*), retirando as outras (so para *who*)
    - `what: r=read, w=write, x=execute`
  - **`Exemplo: chmod -R go+r pub docs *txt`**
    - acrescenta (+) a permissão de leitura (r) para outros usuários (o) e para o grupo (g) em todos os arquivos dos diretórios *pub*, *docs,* e cujo nome termina com *text*

Funções

- função *chown*:
  - função: modifica dono *(owner)* e/ou grupo de um arquivo
  - utilizável a nível de programação
  - sintaxe: **`status =`** `chown`**`(name, dono, grupo)`**
  - `status` é o resultado da operação
    - se >= 0, ok
    - se -1, encontrou um erro
  - `nome` é o nome do arquivo
  - `dono` é a identificação numérica do novo dono do arquivo
  - `grupo` é a identificação numérica do grupo do novo dono

Processos no Unix

Processos

- Unix é multiprogramado e multiusuário
  - execução de diversos programas
  - diversos usuários acessam a máquina ao mesmo tempo
  - cada um dos programas corresponde a um processo em execução

Identificação de Processos

- identificação do processo: *process identifier*
  - denominado *pid*
  - um valor inteiro gerado pelo sistema
  - operações em processos são executadas com base no *pid*
- atributos de um processo
  - *pid*: identificador único do processo
  - *uid*: identificação do proprietário do processo
  - *gid*: identificação do grupo de usuário do processo

Comandos

- comando *ps*:
  - função: imprime o estado dos processos
  - `sintaxe:ps [`*`opções`*`]`
  - o comando *ps* é um dos comandos que mais varia de sistema para sistema
  - opções são dependentes da versão do sistema
  - *default*: processos do próprio usuário associados a um terminal
  - opções básicas (Bell 7):
    - **`-a`**: exibe todos os processos (de todos os usuários) associados a um terminal
    - **`-x`**: mostra não apenas os processos que possuem um terminal de controle associado
    - **`-l`**: lista a informação usando formato longo

Formas de Execução

- há duas formas de se executar um comando:
  - *Foreground*: um comando de cada vez
  - *Background*: executa mais de um comando de cada vez
- *foreground*
  - modo padrão de execução
  - programa pode interagir com o usuário, com entrada de dados
  - pode ser interrompido, parado, etc. através de teclas especiais, como '^C'
- pano de fundo (*background*)
  - sinal '&' no final da linha
  - executa ao mesmo tempo que o *shell*

Morte de Processos

- comando *kill*:
  - função: enviar um “sinal” a um processo (pode matar processos)
  - `sintaxe: kill [-`***número_sinal***`] ***pid1 pid2 pid3***...`
  - exemplos:

```
$ kill -9 164
$ kill 2645 2646
```

  - nem sempre um sinal mata um processo,
  - o sinal “-9” não pode ser interceptado, sempre mata o processo

Comando *nohup*

- comando *nohup*:
  - `sintaxe: nohup` ***comando*** `&`

    permite que um programa continue executando após o encerramento da sessão do usuário
  - exemplo:

```
$ nohup make &
[123]
$ logout
```

Comando *at*

- comando *at*

executa um arquivo de comandos em uma determinada hora

```
sintaxe: at hora [[m] | [dia] [week] [arq]
```

  - **1 a 4 dígitos (1/2 dígitos = hora, 3/4 = horas e minutos)**
  - **abreviação (inglês) (jan = january, dec = december)**
  - **1 a 31, é o dia do mês (*default* é hoje)**
  - **um dia da semana abreviado em inglês (tue = tuesday)**
  - **week uma semana mais após o próximo**
  - **nome de um arquivo de comandos**

Comando *at*

- comando *at*

exemplos

```
at 8am                  hoje às 8 da manhã
at 2130 tue             as 21:30h da próxima terça
at 2pm apr 3            às 2h da tarde do próximo 3 de
  abril
at 1700 wed week        às 17:00h da outra quarta feira
```

Comando *at*

- é possível entrar o comando através da entrada padrão (ao invés de fornecer o nome de um arquivo contendo o(s) comando(s) a ser(em) executado(s))
- exemplos:
  - prepara um aviso na tela para as 23:55h:

```
$ at 2355
echo “Atencao, chega de trabalhar...” >
  /dev/tty
^D
$
```

Execução de Comandos

Comandos

- Tipos de Comandos
  - Comandos Simples
  - Lista de Comandos
  - Comandos em Background
  - Pipe
  - Redirecionamento de Entrada e Saída
  - Construções:
    - if
    - case
    - for
    - while
  - Expressões Aritméticas
  - Entrada e Saída de Dados

Comandos

- Comandos Simples
  - Ë uma sequência de palavras separadas por brancos
  - A primeira palavra especifica o nome do comando
  - As demais palavras palavras são passadas como argumentos
- Ex. cp arq arq1

Comandos

- Lista de Comandos
  - Ë uma lista de um ou mais comandos simples separados por “;” ou “&” ou “&&“ ou ||”, opcionalmente terminados por “;” ou “&”
    - “;”e “&” possuem a mesma precedência
    - “&&”e “||”possuem a mesma precedência, que é maior que “;”e “&”
    - “&&”e “||”indicam execução condicional
      - Com “&&”**lista** é executada somente se o último comando executou com sucesso
      - Com “||”**lista** é executada somente se o último comando não executou com sucesso

Comandos

- Comandos em background
  - O shell cria um processo para executar o comando
  - Indicados pelo caractere “&”
  - O shell volta a ler o terminal
- Ex. gcc prog.c > erros&

Comandos

- Pipe
  - Ë uma sequência de dois ou mais comandos separados pelo caractere “|”
  - Os comandos são executados por processos diferentes
  - A saída do comando à esquerda é conectada a entrada do comando à direita (esquema produtor/consumidor)
  - Ex. ls -l | more

Comandos

- Redirecionamento de Entrada e Saída
  - A entrada e a saída padrão do shell é o terminal
  - Podem ser redirecionadas com o uso dos símbolos:
    - > **palavra**: a saída é redirecionada para **palavra** com eventual criação ou destruição
    - >> **palavra:** a saída é redirecionada para **palavra,** mas se o arquivo já existe, ele não é destruido. A saída é acrescentada ao arquivo.
    - < **palavra:** redireciona a entrada padrão para **palavra**