Universidade Federal de Itajubá – UNIFEI
Curso de Engenharia de Computação
ECO 012 – Laboratório de Técnicas de Programação
Prof. Edmilson Marmo Moreira & Prof. Isaías Lima

Introdução ao Sistema Operacional UNIX

"O livro é o mais alegre de todos os companheiros. Recebe-nos sempre com a mesma bondade, instruindo-nos na juventude, consolando-nos na velhice."

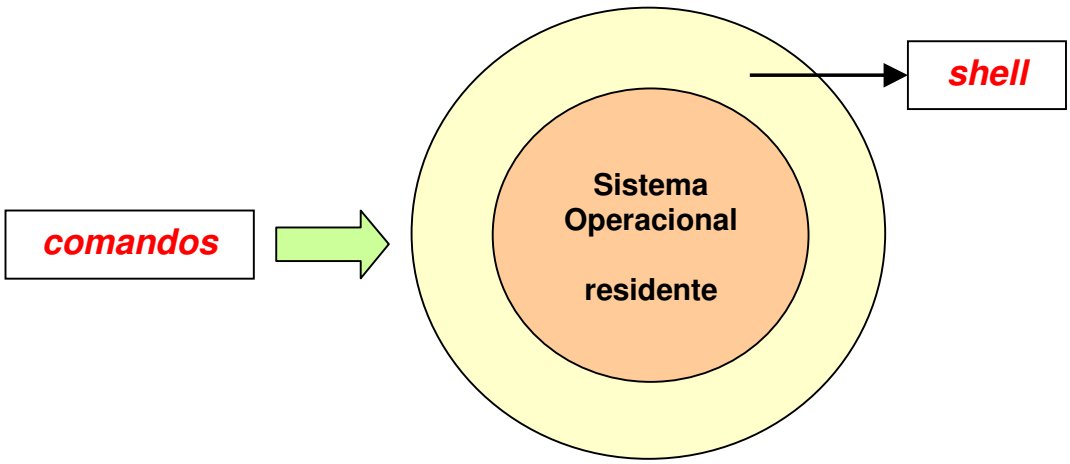
Samuel Smilles

1 Introdução

O UNIX foi desenvolvido, nos anos 70, pelos *Laboratórios Bell*, uma divisão da AT&T. Sendo, quase todo, resultado do trabalho de duas pessoas, Ken Thompson e Dennis Ritchie.

O UNIX é um sistema operacional simples, elegante e fácil de usar, e se tornou um importante padrão que influenciou o desenvolvimento de outros sistemas operacionais como o MS-DOS e OS/2. O UNIX é hoje o sistema operacional mais adequado e mais utilizado em servidores na Internet.

Os comandos do UNIX são processados por uma **cápsula** (***shell***) localizada entre o usuário e o sistema operacional residente.



Este *shell* interpreta os comandos do usuário e converte-os em chamadas do sistema operacional. A cápsula realmente não é parte do sistema

operacional e pode ser modificada pelo programador. Programadores profissionais podem escolher um *shell* técnico. Os iniciantes talvez prefiram escolher comandos, a partir de um menu, ou pela indicação de ícones.

O interpretador de comandos, independente do sistema operacional, foi a grande inovação do UNIX. Existem mais de uma cápsula de uso habitual, e cada fornecedor de UNIX oferece cápsulas alternativas à escolha do usuário.

A cápsula padrão, às vezes chamada de **Bourne Shell** (**sh**), foi desenvolvida nos Laboratórios Bell. Uma segunda cápsula também muito utilizada é a **capsula C** (**C Shell – csh**) que se relaciona com a linguagem de programação C. Uma terceira cápsula é a **Korn Shell** (**ksh**), que vem com o UNIX fornecido pela IBM. Os comandos básicos são os mesmos, variando geralmente nos comandos usados para escrever programas de cápsulas – os comandos ***shell scripts***.

Diferentemente do Windows, o UNIX é um sistema operacional multiusuário, com discos rígidos de alta velocidade, por isso, a distinção entre módulos residentes e transientes é quase transparente para o usuário.

As principais vantagens do UNIX são:

- Portabilidade (conversão para rodar em várias máquinas);
- Padronização (segue um esquema pré-definido);
- Sistema de arquivos hierárquicos e generalidade (utilizado por praticamente qualquer tipo de aplicação).

Além destas vantagens, os sistemas UNIX tendem a ser distribuídos (os recursos são espalhados em várias máquinas) e múltiprocessados (mais de um processador interagindo), além de suportar aplicações em tempo real.

A arquitetura apresenta um componente central (o núcleo ou *kernel*) que interage com o *hardware*, ao mesmo tempo em que oferece serviços através de chamadas do sistema para o *shell*, de utilitários do próprio UNIX e de aplicações do usuário.

O núcleo é o coração do sistema operacional e suas funções básicas são:

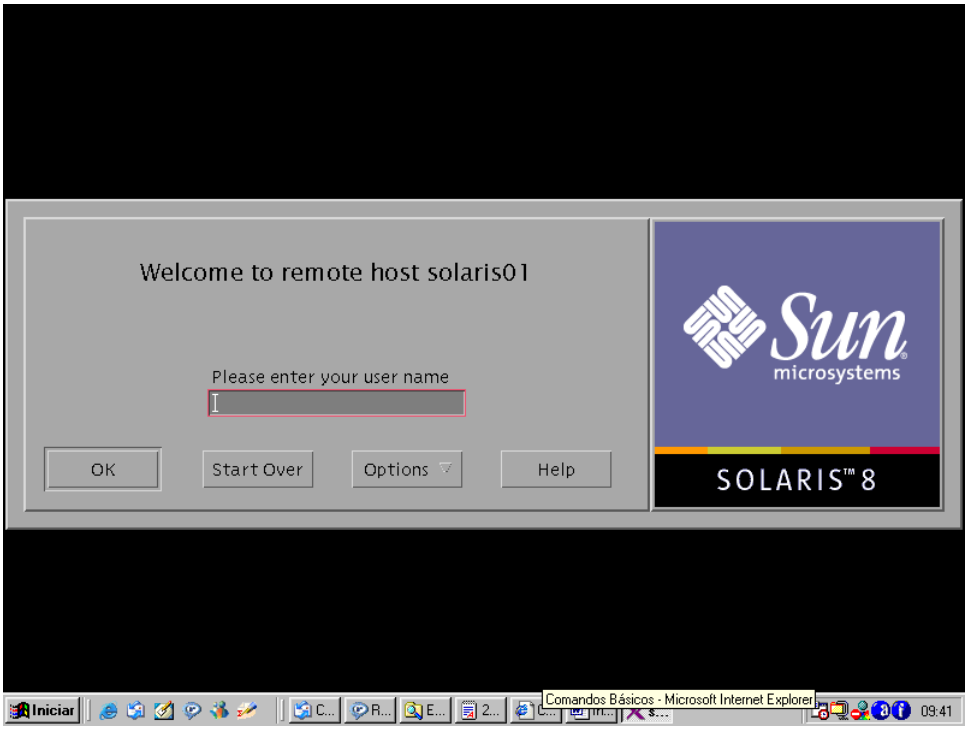
- Interface com *hardware*: serviços de acesso para o *shell*, para utilitários e aplicativos do usuário.
- Gerenciamento de usuários.
- Gerenciamento de arquivos e segurança.
- Serviços de rede.
- Contabilidade do sistema.
- Gerenciamento de erros.
- Gerenciamento de processos.
- Controle de interrupção e erros.
- Serviços de entrada e saída (E/S)

Esta introdução aos comandos e principais utilitários do UNIX é apresentada como um manual de instruções.

2 O Início da Sessão (*Logon*)

O administrador do sistema é, geralmente, o responsável pelos procedimentos de inicialização, como o *boot* do sistema, e o acerto de data e hora, e, portanto, o usuário do UNIX pode ignorar essas tarefas.

Toda sessão no UNIX começa com uma solicitação de **nome de login** e uma **senha**.



Em alguns sistemas, espera-se que você escolha sua senha na primeira vez que entra no sistema. Use uma combinação de até 8 caracteres do teclado.
A estrutura geral de um comando UNIX é:

```
$ comando [-opções] [argumentos...]
```

Os comandos são separados por um ou mais espaços e os campos entre colchetes são opcionais. Os nomes de comandos são geralmente abreviados (`cp` para *copy*, `mv` para *move*, etc).

As opções são usualmente precedidas por um sinal de subtração para distingui-las dos argumentos e são, em sua maioria, indicadas por uma letra minúscula, podendo-se codificar mais de uma. Os argumentos consistem em um ou mais nomes de arquivos.

3 Principais comandos do UNIX

Comandos básicos de usuário

1) **who:** este comando verifica quais são os usuários que estão logados ao sistema.

Sintaxe: `who [am i]`

Argumentos: `am i`: indica com que conta o usuário está logado.

Exemplos:
```
$ who
$ who am i
```

2) **passwd:** comando utilizado para trocar a senha do usuário.

Sintaxe: `passwd [-fs] [username]`

Opções: `-f`: troca o nome completo associado ao usuário.
`-s`: troca o *shell* associado ao usuário.

Exemplo:
```
$ passwd
```

3) **date:** utilitário utilizado para mostrar a data e a hora do sistema.

Sintaxe: `date [+formato]`

Opções: Metacaracteres para formato:

`%D`	Data como MM/DD/AA
`%a`	Dia da semana abreviado (Sun a Sat)
`%h`	Mês abreviado (Jan a Dec)
`%j`	Dia do ano (001 a 365, ou 366 nos anos bissextos)
`%w`	Dia da semana (Domingo=0, Segunda=1,...)
`%m`	Mês do ano (01 a 12)
`%d`	Dia do mês (01 a 31)
`%y`	Os dois últimos dígitos do ano
`%T`	Hora como HH:MM:SS
`%r`	Hora como HH:MM:SS (AM/PM)
`%H`	A hora (00 a 23)
`%M`	O minuto (00 a 59)
`%S`	O segundo (00 a 59)
`%n`	Insere um caractere de avanço de linha
`%t`	Insere um caractere *tab*

Exemplos:
```
$ date
$ date +%D%T
```

4) **cal:** Imprime o calendário para o ano todo ou para um determinado mês do ano.

Sintaxe: `cal [mês] ano`

Argumento: `mês`: indica o mês do ano que se deseja o calendário.

Exemplos:
```
$ cal 2006
$ cal 03 2006
```

5) **write:** inicia conversa interativa com um usuário ou envia um arquivo.

Sintaxe: `write usuario [<arquivo]`

Argumento: `<arquivo`: indica o mês do ano que se deseja o calendário.

Exemplo:
```
$ write edmilson
```

6) **mesg:** permite ou proíbe mensagens.

Sintaxe: `mesg [-ny]`

Opções: `-n`: impede o recebimento de mensagens no terminal.
`-y`: permite o recebimento de mensagens no terminal.

Exemplo:
```
$ mesg -n
```

7) **man:** comando utilizado para exibir o manual do usuário.

Sintaxe: `man [-a...] [seção] comando`

Opções: `-a`: mostra todas as páginas de manual que casam c/ comando.
`...` existem diversas opções para este comando relacionados ao formato de saída (verificar o manual *on-line* para demais opções).

Exemplos:
```
$ man cal
$ man -a date
```

Comandos do sistema de arquivos

1) **pwd:** exibe a rota completa para o diretório corrente.

Sintaxe: `pwd`

Exemplo:
```
$ pwd
```

2) **ls:** lista o conteúdo do diretório e informações sobre os arquivos.

Sintaxe: `ls [-ltasdrucipFR] [nome-do-diretório]`

Opção:
`-l`: lista em formato longo.
`-t`: lista pela ordem cronológica em função da hora da última modificação.
`-a`: lista todas as inserções, inclusive arquivos que comecem com ponto.
`-s`: informa o número de blocos de disco ocupados por cada arquivo.
`-d`: lista nome do diretório, em lugar de seu conteúdo.
`-r`: inverte a ordem da classificação na listagem.
`-u`: lista pela ordem cronológica em função da hora de último acesso.
`-c`: lista pela ordem cronológica em função da hora da última modificação do *inode*.
`-i`: exibe o número do *inode* na primeira coluna.
`-p`: acrescenta uma barra ao nome de cada arquivo de diretório.
`-F`: exibe os nomes de diretório com uma barra (/) no final e os nomes de arquivos executáveis com um asterisco (*) no fim.
`-R`: lista recursivamente os subdiretórios.

Exemplos:
```
$ ls -l
$ ls -la
```

3) **cat:** exibe o conteúdo de um arquivo (ou da entrada padrão, se o arquivo for omitido).

Sintaxe: `cat [-nbvef] [arquivo...]`

Opções:
`-n`: numera as linhas do resultado.
`-b`: quando usado com –n não numera linhas vazias.
`-v`: permite a exibição de caracteres não imprimíveis.

`-e`: quando usado com –v, marca o final de cada linha com $.
`-t`: quando usado com –v, mostra os *tabs* com ^I.

Exemplos:
```
$ cat > Teste
Um
Dois
Três
^D
$ cat -n Teste
```

4) **cp:** faz a cópia de um ou mais arquivos.

Sintaxe: `cp [-i] arquivo... destino`

Opção: `-i`: usa um modo interativo no qual mensagens aparecem antes da cópia prosseguir.

Exemplo:
```
$ cp Teste Teste.backup
```

5) **ln:** cria uma ligação para um arquivo.

Sintaxe: `ln nome pseudônimo`

Exemplo:
```
$ ln Teste Teste.link
```

6) **file:** informa o tipo de um ou mais arquivos.

Sintaxe: `file [-f`*`lista`*`] arquivo...`

Opção: `-f`*`lista`*: informa o tipo dos arquivos cujos nomes estão no arquivo *lista.*

Exemplo:
```
$ file Teste
$ file Teste.backup Teste.link
```

7) **grep:** procura um texto em um arquivo.

Sintaxe: `grep [-vcln] modelo [arquivo...]`

Opção: `-v`: exibe todas as linhas, exceto aquelas contendo *modelo.*
`-c`: informa somente o número de linhas contendo *modelo*.
`-l`: informa somente os nomes dos arquivos contendo *modelo*.
`-n`: precede cada linha com o número da linha no arquivo pesquisado.

Exemplos:
```
$ grep dois Teste
$ grep -vn dois Teste
```

8) **du:** determina a utilização do disco.

Sintaxe: `du [-sa] [diretório]`

Opção: `-s`: informa apenas o número total de blocos.
`-a`: informa também o tamanho de cada arquivo.

Argumento: uso do disco para o sistema de arquivos abaixo de `diretório`.

Exemplo:
```
$ du
```

9) **head:** exibe o início de um arquivo-texto.

Sintaxe: `head [-`*`quant`*`] arquivo...`

Opção: `-`*`quant`*: exibe as primeiras *`quant`* linhas de `arquivo`.

Exemplos:
```
$ head Teste
$ head -3 Teste
```

10) **tail:** exibe a última parte de um arquivo-texto.

Sintaxe: `tail [-`*`quant`* `r] arquivo...`

Opções: `-`*`quant`*: exibe as últimas *`quant`* linhas de `arquivo`.
`-r`: exibe linhas na ordem inversa.

Exemplos:
```
$ tail Teste
```

```
$ tail -3r Teste
```

11) **wc:** conta linhas, palavras e caracteres de um ou mais arquivos.

Sintaxe: `wc [-lwc] arquivo...`

Opções:
`-l`: exibe o número de linhas.
`-w`: exibe o número de palavras.
`-c`: exibe o número de caracteres.

Exemplos:
```
$ wc Teste
$ wc -l Teste
```

12) **sort:** classifica ou combina arquivos.

Sintaxe: `sort [-bdfructc][+pos][-pos][-o resultado] arquivo...`

Opções:
`-b`: Ignora *tabs* e espaços no início das linhas nas comparações.
`-d`: ordena considerando apenas letras, dígitos e espaços.
`-f`: ignora distinção entre letras maiúsculas e minúsculas.
`-r`: inverte a seqüência de ordenação.
`-u`: elimina linhas duplicadas no resultado ordenado.
`-c`: verifica apenas se o arquivo de entrada já está classificado.
`-tc`: especifica separador de campo como sendo o caractere *c.*

Exemplos:
```
$ sort Teste
$ sort -c Teste
```

13) **rm:** elimina arquivos ou diretórios.

Sintaxe: `rm [-ifr] arquivo`

Opções:
`-i`: pede confirmação interativamente para eliminar cada arquivo.

`-f`: força a remoção dos arquivos para os quais não há permissão.
`-r`: recursivamente elimina o conteúdo todo do diretório, assim como o próprio arquivo de diretório.

Exemplo: `$ rm -i Teste*`

14) **mv:** altera o nome ou desloca arquivos ou diretórios.

Sintaxe: `mv [-ifr] arquivo... destino`

Opções: `-i`: pede confirmação interativamente sempre que o destino existir.
`-f`: não pede confirmação para substituir arquivos protegidos contra gravação.

Exemplo: `$ mv Teste Teste2`

15) **mkdir:** cria um diretório.

Sintaxe: `mkdir rota`

Exemplo: `$ mkdir programasC`

15) **rmdir:** remove um diretório.

Sintaxe: `rmdir rota`

Exemplo: `$ rmdir programasC`

16) **chmod:** altera o modo de permissão para arquivos e diretórios.

Sintaxe: `chmod [`*`quem`*`] `*`operação permissão`*` arquivo...`

Argumentos: *quem*: informa a classe do usuário:

- **u**: usuário (proprietário individual do arquivo).
- **g**: proprietário grupal do arquivo.
- **o**: usuários classificados como "outros".
- **a**: todos os usuários do sistema.

operação: representa a operação a ser executada:

- **+**: acrescenta as permissões especificadas.

- **–**: retira as permissões indicadas.
- **=**: atribui as permissões indicadas.

permissão: usa as mesmas abreviações verificadas na listagem longa de um arquivo e/ou diretório:

- **r**: permissão de leitura.
- **w**: permissão de escrita.
- **x**: permissão de escrita.

Exemplo:

```
$ chmod a+x,g-w Teste
```

Bibliografia

ARTHUR, L. J. *UNIX Programação Shell.* 2a. edição. Rio de Janeiro : LTC – Livros Técnicos e Científicos, 1991.

DAVIS, W. S. *Sistemas Operacionais – Uma visão sistemática.* Rio de Janeiro : Campus, 1991.

THOMAS, R. & YATES, J. *UNIX Total.* São Paulo: McGraw-Hill, 1989.