Comandos Básicos do Unix

Roland Teodorowitsch

Programação Orientada a Objetos - ECo - Curso de Engenharia de Computação - PUCRS

7 de março de 2023

Frases

"With a PC, I always felt limited by the software available.
On Unix, I am limited only by my knowledge." (Peter J. Schoenster)

"UNIX is simple and coherent, but it takes a genius (or at any rate, a programmer)
to understand and appreciate its simplicity." (Dennis Ritchie)

"Pretty much everything on the web uses those two things: C and UNIX.
The browsers are written in C.
The UNIX kernel — that pretty much the entire Internet runs on — is written in C." (Rob Pike)

Principais características

- Multiusuário e multitarefa
- Grande variedade de ferramentas, incluindo interpretadores de comandos programáveis e interfaces gráficas
- Flexibilidade e simplicidade
- Orientado a arquivos e fluxos
- Eficiência
- Portabilidade
- Padronização
- Configurabilidade

Histórico

- Em 1969, Ken Thompson, na Bell Laboratories/AT&T, escreve um SO em *assembly* para o DEC PDP7
- O nome Unix surgiu como um trocadilho para outro SO muito popular na época: o Multics (*Multiplexed Information and Computing Service*)
- Originalmente, o Unix foi chamado de "Unics" (*UNiplexed Information and Computing System*)
- Em 1973, Ken Thompson e Denis Ritchie, reescreveram o Unix em C (10.000 linhas, das quais 80 % eram em C)
- Em 1975, o seu código foi foi cedido para universidades, quando começou a ser bastante utilizado
- Veja: *Unix History Project*
- Outros destaques: System V, GNU, MINIX, Linux, etc.

Sessões

- Pode-se usar o Unix através de um gerenciador de janelas (comum, mas opcional) ou diretamente de um terminal (*shell*)
- Iniciando uma sessão
  - Acesso via *user name* (único) e senha
  - O *shell* indica que está apto a receber comandos através de um *prompt* ($)
- Encerrando uma sessão (no *shell*)
  - O comando `exit` encerra uma sessão
  - Também pode-se usar Ctrl + D
  - `logout` pode ser usado para terminais onde houve "*login*"

Formato dos comandos

- Comandos geralmente são mnemônicos (por exemplo, `ls` ou `cp`) e são seguidos de opções e argumentos
- Podem aparecer opções de dois tipos:
  - menos e letra: `-a` (formato original)
  - menos-menos e palavra: `--all` (formato GNU)
- Argumentos podem, por exemplo, ser arquivos e diretórios sobre os quais o comando será executado
- "Filosofia":
  - Se um comando funciona sobre um arquivo e o arquivo não é especificado, então o usuário deve fornecer o conteúdo do arquivo pelo teclado
  - Por exemplo:
    ```
    cat arquivo.txt
    cat
    ```

Para saber mais...

- Há páginas de ajuda para comandos, chamadas de sistema, arquivos de configuração, etc.
- O comando `man` é usado para acessar as páginas de ajuda:

`man man`	(mostra a página de ajuda do comando `man`)
`man ls`	(mostra a página de ajuda do comando `help`)
`man cat`	(mostra a página de ajuda do comando `cat`)
`man sleep`	(mostram a página de ajuda do comando `sleep`)
`man 1 sleep`	(idem)
`man 3 sleep`	(mostra a página de ajuda da chamada `sleep`)
`man 5 passwd`	(mostra a página de ajuda para o arquivo `/etc/passwd`)
`man fopen`	(mostra a página de ajuda para a chamada `fopen`)
`man 3 fopen`	(idem)

- Alguns comandos aceitam também a opção `--help`

`ls --help` (faz com que o próprio comando `ls` mostre informações sobre suas opções)

Comandos para data e hora

`cal`	(mostra o calendário do mês atual)
`cal 2022`	(mostra o calendário do ano 2022)
`cal 12 2022`	(mostra o calendário do mês 12 do ano 2022)
`ncal`	(mostra o calendário do mês atual em formato alternativo)
`ncal 2022`	(mostra o calendário do ano 2022 em formato alernativo)
`ncal 12 2022`	(mostra o calendário do mês 12 do ano 2022 em formato alternativo)
`date`	(mostra a data e hora corrente)
`date '+%d/%m/%Y - %H:%I:%M'`	(mostra a data e hora corrente no formato especificado)

Verificando quem está logado

`who`	(mostra usuários acessando o sistema)
`who am i`	(mostra acesso do usuário corrrente)
`finger`	(mostra informações sobre usuários acessando o sistema – nem sempre está disponível)
`finger roland`	(mostra informações sobre um usuário específico – nem sempre está disponível)
`last`	(mostra últimos acessos de usuários ao sistema)

Manipulação de diretórios

`pwd`	mostra diretório corrente()
`cd`	(vai para o diretório base do usuário – diretório *home*)
`cd ~`	(idem)
`cd Downloads`	(entra no diretório `Downloads`)
`cd -`	(retorna para o diretório visitado anteriormente)
`cd ..`	(vai para o diretório superior – diretório-pai)
`ls`	(mostra arquivos no diretório corrente)
`ls -l`	(mostra detalhes dos arquivos no diretório corrente)
`ls -l -a`	(mostra detalhes de todos os arquivos no diretório corrente – incluindo arquivos ocultos)
`ls -la`	(idem)
`ls -l --all`	(idem)
`ls /tmp`	(vai para o diretório `/tmp`)
`mkdir ~/dir`	(cria o diretório `dir` no diretório base)
`rmdir ~/dir`	(remove o diretório `dir` – que deve estar vazio – no diretório base)

Movimentação, cópia e remoção de arquivos e diretórios

`mv arqOrigem arqDestino`	(troca o nome do arquivo de `arqOrigem` para `arqDestino`)
`mv -i arqOrigem arqDestino`	(idem, porém pedindo confirmação se for preciso sobrescrever `arqDestino`)
`cp arqOrigem arqDestino`	(copia o arquivo `arqOrigem` sobre `arqDestino`)
`cp -i arqOrigem arqDestino`	(idem, porém pedindo confirmação se for preciso sobrescrever `arqDestino`)
`cp -r dir destino`	(copia o diretório `dir` e todos os seus conteúdos para o destino especificado)
`rm arq`	(remove o arquivo `arq`, sem confirmação)
`rm -i arq`	(remove o arquivo `arq`, pedindo uma confirmação)
`rm -r dir`	(remove o diretório `dir` e todos os seus conteúdos)

Visualização de arquivos

```
cd ~ ; for i in `seq 1 1 50` ; do echo linha $i ; done > arq.txt
```

(cria o arquivo `arq.txt` com 50 linhas para teste)

`cat arq.txt`	(mostra o arquivo `arq.txt`)
`more arq.txt`	(mostra o arquivo `arq.txt`, de forma paginada e controlada do início para o fim)
`less arq.txt`	(funciona como o comando `more`, podendo voltar em direção ao início do arquivos)
`head arq.txt`	(mostra o início de um arquivo, ou seja, suas 10 primeiras linhas)
`head -5 arq.txt`	(mostra as 5 primeiras linhas de um arquivo)
`tail arq.txt`	(mostra o final de um arquivo, ou seja, suas 10 últimas linhas)
`tail -5 arq.txt`	(mostra as 5 últimas linhas de um arquivo)

Comandos diversos

`echo "Oi!"`	(mostra a mensagem "Oi!")
`echo $PATH`	(mostra o conteúdo da variável de ambiente `PATH`)
`tty`	(mostra o nome do terminal que está conectado à saída padrão)
`ls -l arq.txt`	(mostra informações completas sobre o arquivo `arq.txt`)
`touch arq.txt`	(atualiza a data e hora de alteração do arquivo `arq.txt`
`ls -l arq.txt`	(mostra informações completas sobre o arquivo `arq.txt`, que agora tem nova data e hora)
`touch novo_arq.txt`	(cria um arquivo `novo_arq.txt`, sem conteúdo)

Permissões de acesso a arquivos (1/2)

`touch arq.txt`	(altera data/hora ou cria o arquivo arq.txt)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`chmod a+x arq.txt`	(define permissões de execução para dono, grupo e outros no arquivo arq.txt)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`chmod go-rwx arq.txt`	(remove permissões de leitura, escrita e execução de grupo e outros)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`chmod 751 arq.txt`	(define as permissões do arquivo para rwx r-x --r)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`chown roland.users arq.txt`	(troca dono e grupo do arquivo – só parar root/administrador)

Permissões de acesso a arquivos (2/2)

`rm arq.txt`	(remove o arquivo arq.txt)
`umask 137`	(define a máscara de permissões para criação de novos arquivos como u=rwx,g=rx,o=rx)
`umask -S`	(mostra a máscara de permissões para criação de novos arquivos)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`rm arq.txt`	(remove o arquivo arq.txt)
`umask 333`	(define a máscara de permissões para criação de novos arquivos como u=r,g=r,o=r)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)
`rm arq.txt`	(remove o arquivo arq.txt)
`umask 337`	(define a máscara de permissões para criação de novos arquivos como u=r,g=r,o=)
`ls -l arq.txt`	(mostra as informações do arquivo arq.txt)

expr

- É utilizado para fazer cálculos simples
- Deve-se usar sempre "\" antes de símbolos (exceto "+" e "-")
- Exemplos:

`expr 14 + \( 10 \* 4 \)`	(retorna 54, ou seja, o resultado de $14 + 10 \times 4$)
`expr 5 \> 1`	(retorna 1, pois 5 é maior do que 1)
`expr 5 \< 1`	(retorna 0, pois 5 NÃO é menor do que 1)

Entrada

- No Unix, a entrada para um comando pode ser obtida de várias formas:
  - Do próprio terminal:
    `sort`
  - De um arquivo aberto pelo próprio comando:
    `sort /etc/passwd`
  - De um arquivo aberto pelo *shell* e passado para o comando por redirecionamento da entrada:
    `sort < /etc/passwd`
- Portanto, em vez de esperar que o usuário digite o que deve ser processado, pode-se especificar o que deve ser processado a partir de um arquivo

Saída

- A saída dos programas pode ser redirecionada para um arquivo com “>”:
  `ls > arquivo.txt`
- Neste caso, o arquivo será criado (ou recriado) sem nenhum conteúdo e a saída do comando será salva neste arquivo
- Para acrescentar no final do arquivo, sem recriá-lo, usa-se “>>”:
  `ls /etc >> arquivo.txt`
- Existem 2 tipos de saídas
  - Saída padrão: gerada por `printf(...)` ou `fprintf(stdout,...)`
  - Saída de erro: gerada por `fprintf(stderr,...)`
- Operadores de redirecionamento:
  - “>” e “>>”: redirecionam apenas a saída padrão
  - “&>”: redireciona as duas saídas
  - “1>”: redireciona apenas a saída padrão
  - “2>”: redireciona apenas a saída de erro

Testes (1/2)

- gedit aplic.c

```
#include <stdio.h>

int main() {
   fprintf(stdout,"informacao\n");
   fprintf(stderr,"erro\n");
   return 0;
}
```

- cc aplic.c -o aplic
- ./aplic

Testes (2/2)

- `./aplic > saida.txt`
- `cat saida.txt`
- `./aplic >> saida.txt`
- `cat saida.txt`
- `rm saida.txt`
- `./aplic > /dev/null`
- `./aplic &> /dev/null`
- `./aplic 1> /dev/null`
- `./aplic 2> /dev/null`

Questões

1. Qual a diferença entre os dois comandos a seguir?
   ```
   expr 5 > 1
   expr 5 \> 1
   ```
2. Qual a diferença entre a execução dos seguintes comandos?
   ```
   cat arq.txt | sort
   sort arq.txt
   sort < arq.txt
   ```