



Instituto Superior de Engenharia de Coimbra  
Departamento de Engenharia Informática e de Sistemas

## Sistemas Operativos

2020/21

### (alguns) Comandos Unix

Sistemas Operativos – 20/21

João Durães

1

#### Unix

- **Sistema de ficheiros Unix**
- Aspecto central do Unix
  - Muitos recursos são vistos como um ficheiro / pseudo-ficheiro
  - Existe uma organização (estrutura) típica. Muitos serviços e recursos dependem dessa estrutura
- A directoria raiz “root” é a directoria base / ponto de partida
- A começar a partir da raiz existe um conjunto de directorias e sub-directorias com um propósito bem definido

Sistemas Operativos – 20/21

João Durães

2

## Unix

/

- Início – directória raiz (não confundir com o utilizador “root”)

/bin

- Comandos e outros ficheiros executáveis

/dev

- Dispositivos (device drivers)

/etc

- Bibliotecas e ficheiros de configuração

## Unix

/lib

- Bibliotecas de âmbito geral e do sistema

/boot

- Ficheiros de arranque da máquina e kernel

/home

- Directórias pessoais (*homedirs*) dos utilizadores

/mnt

/media

- Local standard para montar dispositivos (pens, cdrom etc.). Em Unix os dispositivos aparecem como directórias e não como sistemas de ficheiros independentes (ver comandos *mount* e *umount*)

## Unix

### /opt

- Software opcional, específico a uma determinada instalação

### /proc

- Pseudo-ficheiros que descrevem processos em execução e outros aspectos dinâmicos (run-time) do sistema

### /tmp

- Ficheiros temporários do sistema e dos utilizadores

## Unix

### /usr

- Ficheiros de carácter genérico, tipicamente de âmbito do sistema e não de utilizadores específicos

### /var

- Ficheiros que podem variar muito de tamanho (ex., ficheiros log). Normalmente mapeado num dispositivo diferente e com formato de sistema de ficheiros específico para melhor suportar ficheiros de tamanho muito variável

### /sbin

- Ficheiros de execução restrita (root)

### /srv

- Programas de natureza servidor. Exemplos: servidores de mail, web, ftp, etc.

## Comandos Unix

- Os comandos Unix têm a forma geral
  - **command opções alvo(s)**
- As opções tomam a forma típica -x
  - O símbolo – deve mesmo ser escrito
    - X É uma ou mais letras que identificam a(s) opções pretendidas e podem ter parâmetros
    - Pode-se especificar mais do que uma opção: -x -y
    - Podem-se combinar opções: -xyz em vez de -x -y -z
    - Algumas opções têm parâmetros. Ex., -u jose (user jose)
      - As opções com parâmetros não devem ser combinadas
- Alvo identifica o alvo do comando. Pode ser qualquer coisa, dependendo do comando (ficheiro, directória, user, etc.)
  - Pode haver mais do que um alvo, dependendo do comando

## Comandos Unix

- Com muito poucas excepções, os comandos Unix são simplesmente programas que são executados de forma independente da shell. Desta forma é mais fácil expandir o conjunto de comandos disponíveis.
- Exemplos
  - **ls**
    - Mostra o conteúdo de directórias
    - É um programa localizado em /bin
  - **echo**
    - Imprime mensagens no standard output
    - É um programa localizado em /bin
  - **cd**
    - Muda a directória actual (de trabalho)
    - É um comando interno à shell (nunca poderia ser um programa externo, pelas razões descritas na aula)
  - **set**
    - Define variáveis de ambiente (conceito descrito na aula)
    - É um comando interno à shell, pelas mesmas razões do comando cd

## Comandos Unix

- Comandos internos são executados pela shell (fazem parte do seu algoritmo)
- Comandos externos são programas “normais”, localizados algures no conjunto de directorias onde o sistema procura os programas executáveis. Esse conjunto de directorias é definido pela variável de ambiente PATH
- **Variáveis de ambiente:**
  - Definem aspectos do funcionamento do ambiente do utilizador e do próprio sistema
  - Pares chave-valor
    - *chave* é o nome da variável
    - *valor* é o conteúdo da variável. Não tem tipo de dados associado
    - *chave* e *valor* são simplesmente strings

## Comandos Unix

- **Para obter o valor de uma variável:**
  - Usar `$nome-da-variável` num contexto ou comando que use esse valor.
    - Por exemplo, para imprimir esse valor, usando a variável PATH
    - `echo $PATH`
    - (*echo é um dos comandos brevemente vistos atrás*)
- **Para criar variáveis / mudar o valor de variáveis**
  - Comandos set, export, let
  - Exemplo: `set MyVar=MyValue`
    - Nota: sem o \$ (o \$ é usado apenas para **obter** o valor da variável)

## Comandos Unix

- Melhor comando de todos: **man**

- **man**

- Este comando apresenta informação acerca da maior parte dos assuntos relevantes ao Unix:
    - Ajuda sobre comandos
    - Ajuda nas funções biblioteca da linguagem C
    - Ajuda acerca de componentes e configuração do sistema

- Como usar

- ***man [options] assunto***

Sistemas Operativos – 20/21

João Durães

11

## Comandos Unix

- As ***man pages*** (base de dados para o comando ***man***) estão organizadas em secções

- 1 Programas executáveis e comandos shell
  - 2 Funções sistema Unix (kernel)
  - 3 Funções de biblioteca C
  - 4 Ficheiros especiais (tipicamente em /dev)
  - 5 Formatos de ficheiros especiais (ex., /etc/passwd)
  - 6 Jogos
  - 7 Assuntos variados (ex., outras packages de software)
  - 8 Comandos de administração de sistema
  - 9 Rotinas do kernel (não standard)

- Se o assunto procurado aparecer em mais do que uma secção, o ***man*** apresenta a primeira. Para forçar uma secção:

- ***man número-de-section-number assunto***

Sistemas Operativos – 20/21

João Durães

12

## Comandos Unix básicos, por assunto

### ▪ Operações

- Login, logout,
- Ver utilizadores, ver informação sobre o utilizador
- Mudar a password
- Executar comandos privilegiados

### ▪ Comandos

- logout, exit
- pico, nano, echo
- passwd
- who, whoami,
- su, sudo
- reboot, restart
- apt-get, apt
- useradd, userdel
- ps, kill, top, df

## Comandos Unix básicos

### ▪ logout

- Fecha a sessão do utilizador
  - A máquina continua a correr
  - Outras sessões (desse ou outro utilizador) continuam abertas

### ▪ exit

- Fecha a sessão da shell num terminal:
  - Fecha a sessão, retornando para a sessão anterior
  - Se essa fosse a sessão inicial, então o terminal é encerrado
  - Outros terminais (sessões noutros terminais) não são afectados

## Comandos Unix básicos

### ■ echo

- Apresenta informação no standard output (“ecrã”)
  - A informação pode ser: valores fixos, resultado de execução de outros comandos, valores de variáveis, etc.
- Exemplo: **echo hello Word**

### ■ Algumas opções habituais

- -n
  - Não muda de linha
- -e
  - Permite interpretação de sequências *escape*

## Comandos Unix básicos

### ■ pico e nano

- Editores de texto
- Executam dentro da consola (modo consola / não gráfico)
- Muito fáceis de usar e consomem poucos recursos
- Exemplo: pico myfile              (^o + ^x → grava e sai)

### ■ Outros editores de texto (interface gráfica)

- **mousepad** (vem com o Xubuntu). Actualmente instável
- **notepadqq** (tem que ser instalado manualmente) – relativamente bom e simples

## Comandos Unix básicos

- **passwd** – actualiza informação de utilizador
  - Pode mudar a password e outros detalhes do utilizador
  - Pode ser usado para modificar outros utilizadores, mediante privilégios de root
- **passwd [-k] [-l] [-u [-f]] [-d] [-e] [-n mindays] [-x maxdays] [-w warndays] [-i inactivedays] [-S] [--stdin] [username]**
- Algumas opções (há mais – ver páginas man)
  - **-l**
    - Bloqueia a conta
  - **-n**
    - Define a duração (tempo) da password

## Comandos Unix básicos

- **who** – mostra utilizadores logados
  - Apresenta informação acerca dos utilizadores actualmente logados
- **who [OPTION]... [ FILE | ARG1 ARG2 ]**
- Algumas opções habituais
  - **-u, --userslist**
    - Utilizadores logados
  - **-q, --countall**
    - Nomes e número de utilizadores logados
  - **-m**
    - Apresenta hostname e username
  - **-q, --count**
    - Todos os usernames e número de utilizadores logados

## Comandos Unix básicos

- **whoami** – apresenta identificação do utilizador
  - Apresenta o *username* associado ao **effective user ID** actual.  
Faz o mesmo que *id -un*.
- **whoami [OPTION]...**
- Não tem opções excepto *-help* e *-version*
- Acerca de “Effective ID” -> ?
  - O utilizadores podem assumir temporariamente a identificação de outros utilizadores. Este comando apresenta o ID que está actualmente em uso (ou seja, o **effective user ID**)

## Comandos Unix básicos

- **su** – executa uma nova shell com outro user / grupo ID
  - Executa uma nova shell com o ID do utilizador ou de grupo especificado
  - É pedida a password desse utilizador
  - A nova shell é executada tal como se fosse lançada por esse outro utilizador
  - Se o utilizador não for especificado, +e assumido o root
  - O comando **exit** terminará esse nova shell e fará retornar à shell anterior (a partir da qual foi executado o comando su), recuperando-se o ID de utilizador original
- **su [OPTION]... [-] [USER [ARG]...]**

## Comandos Unix básicos

- **su** - executa uma nova shell com outro user / grupo ID
- Algumas opções habituais
  - **-c, --command=COMMAND**
    - Executa um comando no contexto na nova shell e retorna imediatamente à shell inicial. Útil para executar um comando no contexto de outro utilizador (em vez de obter uma sessão)
    - Exemplo: *su -c comando-qualquer*
  - **-s, --shell=SHELL**
    - Especifica a shell a executar (em vez de usar a default)
  - **-m, --preserve-environment**
    - Mantém as variáveis da shell de origem no contexto da nova shell

## Comandos Unix básicos

- **sudo, sudoedit** - executa comandos como outro utilizador
  - Permite executar comandos em nome de outro utilizador.
  - Por omissão o outro utilizador é o root
  - Não implica a execução de uma nova shell
  - É usado o ficheiro /etc/sudoers para verificar se o comando pode ser utilizado por quem o invoca (pede a password)
- ***sudo options command [options for the command]***
- Algumas opções habituais
  - **-i**
    - Invoca uma shell e permite o uso interativo em vez de apenas executar um comando
  - **-u user**
    - Especifica o utilizador a usar (em vez de usar por omissão o *root*)

## Comandos Unix básicos

- **reboot** – reinicia a máquina
  - *Causa o reinício do sistema (reboot)*
- **reboot [OPTION]...**
- Algumas opções habituais
  - **-f, --force**
    - *Força o reinício mesmo em situações em que o sistema sugere que essa acção não tenha lugar neste instante*

## Comandos Unix básicos

- **shutdown** – Encerra o sistema
  - Faz com que a máquina encerre (não volta a reiniciar). Os utilizadores logados são notificados com uma mensagem. Permite agendar o shutdown especificando um intervalo de tempo antes do sistema efectivamente encerrar. Permite cancelar um ordem de shutdown anterior.
- **shutdown [OPTION]... TIME [MESSAGE]**
- Algumas opções habituais
  - **-r**
    - Faz com que o sistema reinicie (reboot) em vez de apenas encerrar.
  - **-c**
    - Cancela uma ordem anterior de shutdown

## Comandos Unix básicos

- **apt-get** – Gestão de software instalado
  - Utilitário de linha de comandos para gerir software instalado
- **apt-get [options] [-o config=string] [-c=cfgfile] command [pkg]**
  
- Algumas opções habituais
  - Update
    - Actualiza a informação acerca do software disponível nos repositórios remotos
  - Upgrade
    - Actualiza o software instalado
  - Install
    - Instala o software indicado

## Comandos Unix básicos

- **useradd** – Cria ou actualiza um utilizador
  - Permite especificar username, homedir, shell, etc.
- **useradd [options] username**
  
- Este comando é muito detalhado e tem muitas opções
  - -> Consultar a página *man* deste comando
  
- **userdel [options] username** - remove um utilizador
  - Remove um utilizador
  
- Ambos os comandos exigem privilégios de root (ou sudo)
  
- **A informação dos utilizadores está em /etc/passwd e /etc/shadow**

## Comandos Unix básicos

- **Alguns outros comandos úteis**
- **ps**
  - Apresenta os processos em execução
- **kill [signal] process**
  - Envia um sinal a processos
  - Os processos são identificados pelo seu PID (Process ID) que é um valor numérico
  - Útil para terminar processos (sinal 9)
- **top**
  - Apresenta a utilização de memória e processador
- **df**
  - Apresenta a utilização (ocupação) de disco

## Comandos para gerir ficheiros e directórios

- **Operações básicas habituais**
  - Mudar de directório, listar conteúdo de directório
  - Copiar, apagar, mover, comparar, encontrar ficheiros
  - Mudar permissões e posse de ficheiros
- **Comandos**
  - cd, pwd
  - ls, cp, rm, mv
  - mkdir, rmdir
  - whereis, find
  - chmod, chown, touch

## Comandos para gerir ficheiros e directórios

- **ls** – Apresenta conteúdo de directória
  - Apresenta a lista de ficheiros em directória(s). Por omissão os ficheiros são listados por ordem alfabética
- **ls [OPTION]... [FILE]...**
- Algumas opções habituais
  - -a, --all
    - Apresenta também os ficheiros começados por “.”
  - -d, --directory
    - Lista a directória em si, e não o seu conteúdo (útil para ver as propriedades da directória)
  - -l
    - Usa o formato longo (detalhado) para a apresentação
- Existem muitas opções úteis -> consultar a página *man*

## Comandos para gerir ficheiros e directórios

- **ls**
  - Exemplo de output
  - **drwxr-xr-x 2 joao joao 4096 Set 19 20:41 Documents**
- A primeira letra indica o tipo de ficheiro
  - d → Directória
  - - → Ficheiro regular
  - p → named pipe (mecanismo de comunicação)
  - s → socket (mecanismo de comunicação em rede)
  - l → link simbólico (espécie de “shortcut”)
  - b → Device driver do tipo bloco (exemplo, driver de disco)
  - c → Device driver do tipo carácter (exemplo, porta série)

## Comandos para gerir ficheiros e directórios

### ■ ls

#### ■ Caracterização dos ficheiros

- Três tipos de permissões
  - r → read
  - w → write
  - x → execute
- A letra está persente: a permissão é dada
- Está um “-” no lugar da letra: a permissão não é dada”
- Também podem ser descritas como um dígito octal, derivado da representação binária: 1 -> permissão dada, 0 -> permissão não dada, pela ordem read, write, execute
- As permissões são dadas em três conjuntos: para o **owner**, o **group**, e os **others**
- Outras permissões
  - t → **Eliminação restringida – só o donopode apagar o ficheiro.**
  - S → (com execute) = bit setuid / setgid ligado -> o ficheiro executa em nome do dono (em vez de ser como habitualmente em nome de quem o invoca) (usado pelo comando sudo)

## Comandos para gerir ficheiros e directórios

### ■ ls

- Exemplo de output
- **drwxr-xr-x 2 joao joao 4096 Set 19 20:41 Documents**

### ■ Os outros campos são, por ordem

- Número de links para o ficheiro
  - O mesmo ficheiro pode aparecer simultaneamente com vários nomes em vários pontos do sistema de ficheiros. Cada ponto é uma ligação para o ficheiro
- Dono do ficheiro
- Grupo dono do ficheiro
- Tamanho do ficheiro
- Timestamp do ficheiro
- Nome do ficheiro

**Os ficheiros são absolutamente centrais à forma de operação do Unix**

## Comandos para gerir ficheiros e directórios

- **cd** - Muda a directória actual (“de trabalho”)
- **cd directória-pretendida**
  - A directória pode ser relativa à directória actual, ou pode ser absoluta (relativa à raiz do sistema)
  - Exemplos
    - abc/ola/ → directória *ola* dentro de *abc* dentro da directória actual
    - ../otradir → um nível “acima”, e então, *otradir* (relativamente à directória actual)
    - /tmp/games → *games* dentro de *tmp* a partir a da directória raiz
    - ~ → significa sempre a homedir do utilizador actual
    - cd em nada muda para a homedir do utilizador actual
- “caminho” → directória e nome de um ficheiro, geralmente a partir da directória raiz
  - Exemplo: */etc/passwd* é o caminho e nome do ficheiro *passwd*

## Comandos para gerir ficheiros e directórios

- **cp** – copia ficheiros e directórios
  - Copia SOURCE para DEST, ou vários SOURCE(s) para DIRECTORY.
- **cp [OPTION]... [-T] SOURCE DEST**
- **cp [OPTION]... SOURCE... DIRECTORY**
- **cp [OPTION]... -t DIRECTORY SOURCE...**
- Algumas opções habituais
  - -u, --update
    - Copia apenas quando a origem SOURCE é mais recente que o destino DEST
  - -R, -r, --recursive
    - Copia directórios de forma recursiva (directórios dentro de outras directórios)

## Comandos para gerir ficheiros e directórios

- **mv** - move (renomeia) ficheiros
  - renomeia ou move SOURCE para DEST
- **mv [OPTION]... [-T] SOURCE DEST**
- **mv [OPTION]... SOURCE... DIRECTORY**
- **mv [OPTION]... -t DIRECTORY SOURCE...**
- Algumas opções habituais
  - **-f, --force**
    - Não pergunta nada antes de escrever por cima de DEST
  - **-i, --interactive**
    - Pergunta sempre antes de escrever por cima
  - **-u, --update**
    - Move apenas quando SOURCE é mais recente que DEST
  - **-n, --no-clobber**
    - Não escreve por cima quando DEST já existe

## Comandos para gerir ficheiros e directórios

- **rm** – apaga ficheiros e directórios
- **rm [OPTION]... [FILE]...**
- Algumas opções habituais
  - **-f, --force**
    - Executa sempre sem perguntar nada a utilizador
  - **-i**
    - Pergunta confirmação antes de cada eliminação que vai fazer
  - **-r, -R, --recursive**
    - Remove conteúdo de directórios de forma recursiva (directórios dentro de directórios)
  - **-d, --dir**
    - Remove também as directórios que estão/ficaram vazias

## Comandos para gerir ficheiros e directórios

- **mkdir** – cria directórios
  - Cria directórios DIRECTORY(ies), se ainda não existirem
- **mkdir [OPTION]... DIRECTORY...**
  
- Algumas opções habituais
  - -p, --parents
    - Cria também as directórias de suporte se não existirem e forem especificadas
      - Exemplo
        - mkdir abc/ola
        - Falha se abc não existir.
      - mkdir -p abc/ola
      - Se abc não existir, cria-a primeiro, e depois ola dentro de abc.

## Comandos para gerir ficheiros e directórios

- **rmdir** – apaga directórios
  - Elimina directórios DIRECTORY(ies), se estiverem vazias
- **rmdir [OPTION]... DIRECTORY...**
  
- Algumas opções habituais
  - --ignore-fail-on-non-empty
    - Ignora os casos em que as directórias não estão vazias
  - -p, --parents
    - Remove também as directórias base da directória removida
    - Exemplo: **rmdir -p a/b/c** faz o mesmo que **rmdir a/b/c a/b a**

## Comandos para gerir ficheiros e directórios

- **whereis** – identifica os locais onde existem o binário, o código fonte, ou páginas de manual de um determinado comando
  - Este comando normalmente identifica apenas um conjunto restrito de ficheiros (tipicamente comandos) e procurando apenas num conjunto específico de locais onde existem normalmente o código fonte, o executável (“binário”), ou páginas de manual.
  - Para uma pesquisa mais geral, deve-se usar o comando **find**
- **whereis [-bmsu] [-BMS directory... -f] filename...**
- Algumas opções habituais
  - **-b**, **-s** **-M dir**
    - Pesquisa apenas por binários (b) código fonte (s), ou numa directória especificada (M)

## Comandos para gerir ficheiros e directórios

- **find** – procura ficheiros
  - Procura ficheiros de qualquer tipo em qualquer local no sistema de ficheiros
- **find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]**
- Este comando é extremamente poderoso. Permite especificar com grande detalhe as características do(s) ficheiro(s) a procurar, os locais onde procurar, e que acções devem ser tomadas quando os ficheiros são encontrados
- Exemplos
  - **find / -name foo.txt -type f -print**
    - Procura ficheiro dado o seu nome (-name) do tipo ficheiro regular (f) começando na directória raiz (/) e se encontrar esse(s) ficheiro(s), apresenta-o(s) na consola (-printf)

## Comandos para gerir ficheiros e directórios

### ▪ Mais exemplos

- **find / -name foo.txt -type f -print**
  - Procura ficheiro foo.txt (-name) do tipo regular (f) começando na raiz (/) e apresenta-os (-printf)
- **find /opt /usr /var -name foo.txt -type f**
  - Procura ficheiro regular foo.txt nas directórias /usr e /var
- **find . -type f -not -name "\*.html"**
  - Procura ficheiros sem extensão "html" a começar na dir. actual (.)
- **find /usr/local -name "\*.html" -type f -exec chmod 644 {} \;**
  - Procura ficheiros e executa a acção chmod sobre eles (chmod 644 {}). {} significa cada ficheiro encontrado (neste caso, como argumento para chmod). \; indica o fim da acção.
- **find . -type f -name "\*.mp3" -exec cp {} /tmp/Music \;**
  - Encontra e copia ficheiros de música (\*.mp3) para /tmp/Music
- **find . -mtime -7 -type f -exec rm {} \;**
  - Procura ficheiros (-type f) modificados (-mtime) nos últimos 7 dias (-7) e apaga-os (-exec rm {} \;);

## Comandos para gerir ficheiros e directórios

### ▪ **chmod** – modifica o modo (permissões) de ficheiros

- Modifica os bits de permissão r/w/x de ficheiros. O modo pode ser especificado como representação octal dos seus bits correspondentes, ou uma representação simbólica r=read, w=write, x=execute e o=owner, g=group, o=other e ainda - = remover permissão, + = adicionar permissão.

▪ **chmod [OPTION]... MODE[,MODE]... FILE...**

▪ **chmod [OPTION]... OCTAL-MODE FILE...**

▪ **chmod [OPTION]... --reference=RFILE FILE...**

### ▪ As permissões podem ser dadas em **octal** or com letras que simbolizam as permissões e o alvo da permissão (dono, grupo, restantes)

- Utilizadores afectados: u (user), g (group), o (others)

- Operações r (read), w (write), x (execute)

- (Tal como visto no contexto do comando ls)

- + -> adicionar, - -> remover

## Comandos para gerir ficheiros e directórios

- **touch** – modifica timestamp de ficheiros
  - Actualiza o tempo (timestamp) do ultimo acesso/modificação de ficheiros para coincidir com a hora actual do sistema.
  - Se o ficheiro alvo FILE não existir, será criado (com zero bytes de tamanho) excepto se tiverem sido especificados os argumentos -c ou -h.
- **touch [OPTION]... FILE...**
- Algumas opções habituais
  - -a
    - Modifica apenas o timestamp relativo ao último acesso
  - -m
    - Modifica apenas o timestamp relativo à última modificação
  - -c, --no-create
    - Não criar o ficheiro (caso não exista)
  - -d, --date=STRING
    - Utiliza a STRING indicada como hora actual em vez da hora do sistema

## Apresentação e manipulação de texto

- Apresentação e manipulação de texto (de ficheiros de texto)
  - Útil e muito importante nas operações de gestão da máquina:
    - Extracção de dados concretos a partir do output de outros comandos
    - Extracção de dados com base em ficheiros de configuração (que normalmente são de texto com uma estrutura regular. Ex.: /etc/passwd)
- Comandos
  - **cat**      -> Mostra conteúdo de ficheiros
  - **cut**      -> Extrai uma coluna (campo)
  - **grep**      -> Extrai linhas específicas com base num filtro (expr. regular)
  - **head**      -> Mostra primeiras linhas de um ficheiro
  - **tail**      -> Mostra últimas linhas de um ficheiro
  - **more**      -> Mostra ficheiro página a página. Alternativa melhor: **less**
  - **sort**      -> mostra conteúdo ordenado por um certo critério
  - **uniq**      -> mostra conteúdo filtrando repetições
  - **wc**      -> apresenta número de caracteres/palavras/linhas

## Apresentação e manipulação de texto

- **cat [OPTION]... [FILE]...**
  - Mostra (e concatena se forem vários) o conteúdo do ficheiro na saída standard
- **cat [OPTION]... [FILE]...**
- Algumas opções habituais
  - **-b, --number-nonblank**
    - Numera as linhas (que não sejam vazias)
  - **-n, --number**
    - Numera todas as linhas
  - **-s, --squeeze-blank**
    - Não apresenta linhas vazias repetidas
- Se o ficheiro não for especificado, obtém o texto da entrada standard

## Apresentação e manipulação de texto

- **cut - remove sections from each line of files**
  - Mostra na saída standard uma parte (campo / coluna) de cada linha do ficheiro FILE (ou entrada de dados standard)
- **cut OPTION... [FILE]...**
- Algumas opções habituais
  - **-c, --characters=LIST**
    - Especifica a parte a mostrar pelo número de caracteres
  - **-d, --delimiter=DELIM**
    - Especifica qual o carácter que separa um campo do seguinte (delimitador). Exemplo **-d " "** identifica o espaço como delimitador.
  - **-f, --fields=LIST**
    - Especifica a parte a mostrar através do número do campo (primeiro = 1). Usado geralmente com **-d** para identificar qual é o separador de campo. Mostra também as linhas que não tenham sequer o campo especificado (excepto se se indicar a opção **-s**)

## Apresentação e manipulação de texto

- **grep** – filtra as linhas com base num padrão
  - Apresenta na saída standard as linhas recebidas (do ficheiro FILE ou da entrada standard) que cumpram um determinado padrão.
  - O padrão é normalmente especificado como uma expressão regular. Permite seleccionar linhas com base em critérios bastante complexos e poderosos.
    - Um exemplo genérico: *selecionar as linhas que começam por uma determinada palavra, têm um certo conjunto de letras, que se pode repetir, mas não mais do que um determinado número de vezes, e que contém esta ou aquela palavra em alternativa, e que terminam com uma determinada letra dentro de um certo conjunto*
- **grep [OPTIONS] PATTERN [FILE...]**

Este comando é bastante poderoso e as suas funcionalidades mais complexas exigem vários exercícios para praticar. A seguir são dados alguns breves e simples exemplos

## Apresentação e manipulação de texto

- Alguns exemplos (muito) simples do uso (isolado) de grep**
- **grep '^fred'**
    - Linhas começadas por fred
  - **grep '[FG]oo'**
    - Linhas contendo Foo ou Goo
  - **grep '[0-9][0-9][0-9] '**
    - Linhas contendo três algarismos seguidos
  - **grep -v bubble**
    - Linhas que não contém a palavra bubble
  - **grep "^[a-zA-Z]"**
    - Linhas que começam por uma letra (minúscula ou maiúscula)

## Apresentação e manipulação de texto

- **head** – apresenta a parte inicial de ficheiro ou standard input
  - Apresenta na saída standard as primeiras 10 linhas do texto de cada ficheiro FILE ou entrada standard
- **head [OPTION]... [FILE]...**
- **Algumas opções habituais**
  - -n, --lines=[-]K
    - Apresenta as primeiras K linhas em vez das primeiras 10. Com o sinal ‘-’, apresenta todas menos as primeiras K linhas

## Apresentação e manipulação de texto

- **tail** - apresenta a parte final de ficheiro ou entrada standard
  - Apresenta na saída standard as últimas 10 linhas do texto de cada ficheiro FILE ou entrada standard
- **tail [OPTION]... [FILE]...**
- **Algumas opções habituais**
  - -n, --lines=K
    - Apresenta as últimas K linhas em vez das últimas 10. Com –n +K apresenta as linhas a partir da k-ésima

## Apresentação e manipulação de texto

- **more** - Apresenta conteúdo uma página de cada vez
  - O conteúdo pode ser o ficheiro ou a entrada standard
- **more [-dflpcsu] [-num] [+pattern] [+linenum] [file ...]**
- Algumas opções habituais
  - -num
    - Indica que deve apresentar *num* linhas de cada vez.
- Alternativa: comando **less**
  - O comando *less* é menos pesado para o sistema (não necessita de ler todo o ficheiro) e permite “andar” para cima e para baixo no conteúdo apresentado

## Apresentação e manipulação de texto

- **sort – apresenta conteúdo depois de ordenado**
  - Envia para a saída standard o input depois de ordenado por um determinado critério. O conteúdo original (ficheiro FILE ou entrada standard) não é modificado..
- **sort [OPTION]... [FILE]...**
- Algumas opções habituais
  - -d, --dictionary-order
    - Ordem alfabética considerando apenas espaços e caracteres alfanuméricos
    - Caso dos números: “10” parece antes de “9” (“1” < “9”)
  - -f, --ignore-case
    - Ignora diferenças entre minúsculas e maiúsculas
  - -n, --general-numeric-sort
    - Ordem numérica (ex.: “9” aparece antes de “10”)
  - -k num,
    - Usa coluna *num* para o critério de ordenação (primeira = 1)
  - -r, --reverse
    - Inverte a ordem

## Apresentação e manipulação de texto

- **uniq – filtra, omitindo, linhas repetidas**
  - Envia para a saída standard o conteúdo dado (INPUT) omitindo linhas repetidas.
- **uniq [OPTION]... [INPUT]**
  
- Algumas opções habituais
  - -d, --repeated
    - Apresenta apenas as linhas repetidas
  - -u, --unique
    - Apresenta apenas as linhas únicas (que não se repetem)
  - -s, --skip-chars=N
    - Não considera para a comparação de repetição os primeiros N caracteres em cada linha

## Apresentação e manipulação de texto

- **wc – apresenta a contagem de caracteres, palavras e linhas**
  - Apresenta na saída standard o número de caracteres, de palavras e de linhas do conteúdo dado (ficheiro FILE ou entrada standard)
- **wc [OPTION]... [FILE]...**
  
- Algumas opções habituais
  - -c, --bytes
    - Apresenta o número de bytes
  - -m, --chars
    - Apresenta o número de caracteres
  - -l, --lines
    - Apresenta o número de linhas
  - -w, --words
    - Apresenta o número de palavras

## Redireccionamento

- A entrada e a saída dos comandos (e programas em geral) pode ser redireccionada de uns para os outros e para ficheiros.
- Isto significa que a saída de um comando pode ser vir para alimentar a entrada de outro comando, ou seja, um comando pode operar sobre os resultados do comando anterior, acrescentando funcionalidade ao que o anterior já tinha feito.
- Este processo não está limitado apenas a dois comandos: pode ser usado para ligar um número arbitrário de comandos numa longa cadeia, construindo um processamento de texto bastante complexo, um passo (comando) simples de cada vez
  - Mas quando se redirecciona para um ficheiro, a cadeia termina ai
- Requisitos para se poder usar esta característica:
  - Os comandos devem ir buscar os seus dados à entrada de dados standard ("teclado")
  - Os comandos devem enviar o resultado para a saída standard ("ecrã")
  - Normalmente é isso que acontece.

## Redireccionamento

- Redireccionamento de/para ficheiro: > >> <
- **cmd > fich**
  - A saída (standard) do comando *cmd* é enviada para ficheiro *fich* en vez de ir para o ecrã
- **cmd 2> fich**
  - A saída (standard) de erro do comando *cmd* é enviada para ficheiro *fich* en vez de ir para o ecrã
- **>> em vez de >** acrescenta ao ficheiro (em vez de escrever por cima)
- **cmd < fich**
  - O comando cmd vai buscar os dados ao fich em vez de ao "teclado" (entrada standard)
- **cmd < fich1 > fich2 2> fich3**
  - <> e >> podem ser usados em simultâneo

## Redireccionamento

- Redireccionamento de comando para outro comando: **pipe**
- **cmd1 | cmd2**
  - A saída do comando cm1 é enviada para a entrada do comando cm2
  - Pode-se usar uma cadeia de comandos (programas) com comprimento arbitrário
    - **cmd1 | cmd2 | cmd3 | cmd4 .... Etc.**
  - O limite é apenas o da memória (para conter todos os comandos a correr em simultâneo)
- Para serem compatíveis com redireccionamento os comandos (programas em geral) apenas precisam de usar as entrada e saída standard, o que normalmente se traduz em não fazer nada de especial (mais tarde será visto como isto funciona a nível de programação em C)

## Redireccionamento

- Todos os comandos para processar texto tem o seguinte em comum
  - Lêem os dados de trabalho da entrada standard (por omissão)
  - Envio o resultado para a saída standard
- Logo são intrinsecamente compatíveis com o redireccionamento.
- Na maior parte dos casos, estes comandos são mesmo usados como parte de uma cadeia de comandos ligada por redireccionamento
- Exemplo muito simples
  - **cat contactos | grep manuel | wc -l**
    - Conta o número de vezes que uma pessoa com o nome “manuel” aparece no ficheiro *contactos*
- É normal encadear seis, sete ou mais comandos numa única operação