Getting around in Linux/Unix

Alan M. Durham
Computer Science Department
University of São Paulo, Brazil
alan@ime.usp.br

Logging in: everyone is a user

- Unix/linux is not a operating system for a *personal* computer
- many users/many accounts
- to provide security each user is given an account name
- each account has a password
- a note on passwords
  - many programs to break passwords
  - bad passwords
    - any word in english or portuguese
    - any date
    - name of any person
  - hint: use a combination of letters (upper and lowercase) and numbers
  - maybe intermingling a word with algarisms of a date/phonenumber

Files and directories

- directory ~ folder
- all files of all disks in a single directory tree, no letters
- all addresses start with “/”, the root directory
- an address is called a “path”
  - /home/alan/courses/introLinux/presentation.ppt
  - /usr/bin/perl
- relative path vs. absolute path
  - when we are “in” a directory we can use local names without the complete path
    - courses/introLinux/presentation.ppt
- when a user logs in he goes to his ***home*** directory
  - /home/*userName* in most Linux systems
  - /home/userName is the users ***current*** directory

Permissions: protecting data

- if we can write a path to everyone’s files, how do we protect data?
- each file/directory has a set of permissions
  - reading
  - writing
  - executing
- three groups of people
  - the owner
  - the group
  - the world
- for ***each*** file and directory we can set or reset each type of permission for each group

Nome	Permissões Dono			Permissões Grupo			Permissões Resto		
Nota.xls		R	W					R	
Projeto.tex		R	W		R	W			
DiretorioPub	X	R	W	X	R	W	X	R	W
ImpostoRenda		R							
Prog1	X	R	W	X	R		X		

The shell

- Windows and Icons are fine, but cumbersome to use
- when we know the system, easier to type commands
- Linux can work in text mode
- program that “collects” user commands is called ***The Shell***
- in graphical mode, we can have a window with the shell
- when we enter the system in text mode, shell is installed automatically
- many different shell programs, user can choose the preffered one
- we will use *bash, (bourne again shell)*

Some basic shell commands

- basic structure

  <command>   [-options] ...
- ***ls:*** listing existing files in current directory
  - short listing: ***ls***
  - list all files: ***ls -a***
  - detailed list: : ***ls -l***
  - all of the above: ***ls -la***
- ***cd:*** changing the current directory
  - to a subdirectory: ***cd cursos/ibi5011***
  - giving complete path: ***cd /home/aluno1/ibi5011***
  - to the next directory above: ***cd ..***
- ***echo:*** printing something in the screen
  - ***echo "alan"***

Some basic shell commands

- ***cat:*** inspecting a file
  - ***cat /home/alan/ibi5011/arquivoMisterio1***
  - ***cat /home/alan/ibi5011/arquivoMisterio2***
- ***more:*** inspecting a file, a page at a time (space bar shows next page)
  - ***more /home/alan/ibi5011/arquivoMisterio2***
- ***cp:*** copying a file
  - ***cp /home/alan/ibi5011/arquivoMisterio2 meuArquivoMisterio***
  - ***cp meuArquivoMisterio meuArquivoMisterioBak***
- ***mkdir:*** creating a new directory
  - ***mkdir meusExemplos***
  - ***mkdir /home/eu/temp***

Some basic shell commands

- ***rmdir:*** removing a directory (cannod be reversed!!!!)
  - rmdir /home/eu/temp
- ***rm:*** removing a file (cannot be undone!!!!)
  - ***rm***

Exercise

- look at the files in /home/alan/basicUnix/exercise1
- create a directory ***exercise1*** in your account
- copy to the new directory, the files in the old directory that contain a date

Exercício

- ***invesigue os arquivos do diretório /home/alan/ibi5011/exercise1***
- ***crie um diretorio “exercicio1” em sua conta***
- ***crie um diretorio “temp” em sua conta***
- ***copie neste novo diretório todos os arquivos do primeiro diretório que contém alguma data***

Some basic commands

- the bash shell has automatic completion, just press <TAB>
- completion is used for command names and for file names
  - try: more /home/alan/bas<tab>/exe<tab>1/<tab>1
- pressing <tab> twice gives you all options
  - more /home/alan/ibi5011/exercise1/<tab><tab>

Input and output

- **in Unix(Linux) the *standard output* of a command or program is the terminal**
- **the *standard input* of a program is the keyboard**
- **we can *redirect* the output of a program to a file**
- **just put “>” after the command and type a file name.**
  - **using *echo* to create a file:**

    ```
    echo “Alan: Rua do Matao, 1010, Cid Universitaria, 3091-6299” > alanAddress
    ```
  - **saving long output to see later**

    ```
    ls -l /usr/bin
    ls -l /usr/bin > temp/saida1
    more saida1
    ```

Dealing with permissions

- to change the permissions on a file you use the command ***chmod***

  chmod <who><operation><permission>
- who:
  - u: owner
  - g: group
  - o: rest of the world
  - a: all
- operation
  - “+” : add permission
  - “-” : remove permission
- permission:
  - “x” : execute
  - “r”: read
  - “w” : write

Examples (guess what they do)

- chmod u-r misteryFile1.txt
- chmod a+w misteryFile2.txt
- chmod g-r misteryFile3.txt

Exercise

- create a file with your name in our home directory, writing a small poem in it
- check the file's protection
- protect the file, preventing anyone from reading it
- create a new directory named *temp*
- create inside this directory a copy of all "mistery" files
- "go into" the new directory, making it your current directory
- try the command
  - cat misteryFile.txt misteryFile2.txt > outFile
- what happened?

Exercício

- crie um arquivo com o seu nome no seu diretório, colocando nele um verso qualquer.
- veja qual a proteção do arquivo.
- proteja o seu arquivo, impedindo que o resto do mundo possa lê-lo
- crie um novo diretório na sua conta com nome “temporario”.
- faça neste novo diretorio uma copia de todos os aquivos com nome “arquivoMisterio...””
- faça do novo diretorio seu diretório corrente
- experimente o comando:
  - cat arquivoMisterio1.txt arquivoMisterio2.txt > saida
- o que aconteceu?

Exercício

- remova seu diretorio “temporario”
- foi fácil?

Looking for help: man, apropos

- most programs in Linux/Unix offer an on-line manual
- man <program name>
- try to type: “man ls” and find some new option
- also unix provides a command finder
- typing
  - apropos <word>
- will show all manual entries that contain that word
- try
  - apropos browser

Wildcards in unix commands

- In unix you can use special character to help you
- "*" means any sequence of character.
- "?" means any character
- Examples
  - ls *.ppt - lists all files that end with ".ppt"
  - ls a*.fasta -lists all files where the name start with an "a" and end with a ".fasta"
  - cat *.fast? - lists the contents of all files that end with "fast" followed by any character.
- Go to the directory where you put the mistery files
  - Try: ls *.txt
  - Try: ls m*
  - Try : ls *2.txt
  - Try: ls misteryFile?.txt

Other unix commands

- where am I?
  - **pwd : prints your current absolute path**
- who is around?
  - **who: prints other users in your machine**
- where is that file?
  - **find** <path> **-name** <name>
  - find . “*.ppt”
- what is the name of that file?
  - **grep** <patern> <files>
  - grep “forces” *
- Counting words and lines in a file
  - **wc <file name>:**
  - wc misteryFile1.txt

Using find and grep with wildcards

- we can use “wildcard”characters to make searches more general
- “*” is the main one, means any set of characthers
- ex:
  - find /home/alan -name “*.ppt” : finds all powerpoint files in my account
  - grep human *.txt : look for the word “human” in all the files in my directory.

Exercise

- look which of the mistery files contain the word “forces” using grep
- find all the files with names ending in “.txt” in your directory using find

Some more stuff

- how big is my file?
  - wc <file_name>
- how big is each txt file of mine?
  - wc *.txt
- You can joing commands together using a “pipe”
  - a cute use of it: finding how many files there are in a directory

    ```
    ls /bio/home/ | wc
    ```

  - You can use grep and ls together

    ```
    ls * | grep ppt
    ```

Exercise

- count how many lines in your mistery files have the word “forces”

Even more stuff: usint multitasking

- Unix/Linux actually runs many programs at the same time
- You can make many programs run at the same time youself, there are some commands you can use for it
- stop the current program for a while
  - C-z
- That last program? Run it in the background
  - bg
- Bring the last program back
  - fg
- Which programs did I ask you to run?
  - jobs

You can use bg and fg with any running program, using its number:

```
emacs
C-z
bg
emacs
C-z
jobs
bg %1
fg %2
```

A modal editor: emacs

- modes for perl, C, java, html, diretories, latex, etc.
- the most used modal editor in Linux/Unix is ***Emacs***
- emacs works both on graphical mode and text mode
- emacs is configurable (if you know lisp programming)
- emacs has modes for most programming languages
  - C, C++
- a modal editor is a text editor capable of working in “modes”
  - Java
  - Pascal
  - Perl
  - Python
  - Fortran, etc
- with emacs in a graphical environment you can work with the mouse or use only the keyboard
- emacs also runs on windows

Using emacs

- to start emacs just “call it” typing

  emacs
- basic editing in emacs is very intuitive
  - use arrows, “pg up”and “pg down”to move cursor
  - use del key to delete
  - back key to delete backwards
  - typing insert text at the cursor position
- to edit an existing file type

  emacs <name of the file>
- Exercise
  - edit the file arquivoMistério1.txt, changing the date in the first line to today’s date
  - add at the end of the file a line with your name.

Using Emacs: keyboard commands

- there are some keyboard commands you need to know
- we use the folowing abreviations
  - “C” is the “Control” key
  - “M” is the “Esc”key
  - “-” between two letters mean both have to be pressed simoutaniously
- Some basic commands
  - C-x, C-s - save the file
  - C-x, C-c - exit Emacs
- Exercise:
  - save the new contents of you file
  - load file “arquivoMisterio2.txt”, remove its first line, and save the result

Using Emacs: the minibuffer

- if you look at your screen you see a solid bar in the bottom of your page
- underneath this bar is the “minibuffer”
- the “minibuffer” is used for thecommunication between you and Emacs
  - emacs prints messages there
  - you type text that emacs needs to perform a command
  - you can type commands here
- Exercise:
  - try to save your file again, what does appear in the minibuffer?

Using Emacs: commands that use the minbuffer

- C-x C-w "save as" - you type the new name in the minbuffer
- C-x C-f load a new file in Emacs
- C-s : search for a string
  - this search is incremental and goes as you search
  - typine C-s again will search for the next occurency of the same string
  - to go back to the editing, just press any arrow key
  - after you go back, typing C-s twice resumes the search
- if you use <TAB> in the minibuffer, Emacs tries to complete the string for you
  - this does NOT work with search (C-s)
  - if there is more than one completion, emacs splits your editor in two, and list all options
  - choosing an option makes the choice window go away\

Exercício

- chame o emacs para sua cópia do arquivo arquivoMisterio1.txt
- salve o arquivo com o nome meuArquivoMisterio1.txt
- procure as tres primeiras ocorrëncas da palavra “ponto” e substiua por “indice”
- salve o resultado
- sem sair do Emacs, carrege o arquivo arquivoMisterio4.txt
- digite “Sao Paulo, 12 de marco”.
- sem salver, tente sair do Emacs
  - O que acontece?

Some more Emacs Commands

- C-x 2 : splits the screen in two
- C-x o : move the cursor from one screen to the other
- all commands apply to the screen where the cursor is
- it is like having 2 Emacs running at the same time
- windows are independent

Exercise:

- load file “arquivoMisterio4.txt”
- split the sreen in two.
- type “Hello World!”
  - What happens?
- page down untill the end of the file
- load the file “arquivoMisterio3.txt”
  - now you have two files at the same ime
- go to the window that is showing file “arquivoMisterio4.txt” and save the contents under the name “result2.txt”

Some Emacs Commands

- M-x shell ==> opens a shell
- M-x goto-line ==> to to a specific line number
- M-x replace-string => replaces text
- M-x query-replace => substitute text, asking each time
- C-x s => save the file
- C-x w => “save as”
- C-x k ==> closes the “buffer”, i.e. The file you are working in
- C-x C-f ===> reads a file
- C-k ==> eliminates a line
- C-a ===> move cursor to beginning of line
- C-e ==> move cursor to end of line
- C-y ==> paste back what you deleted with C-k
- C-d ==> deletes next character
- M-d ==> deletes next WORD.