Introductory Course in Linux

Pedro Ojeda-May, Birgitte Brydsö, and Jerry Eriksson

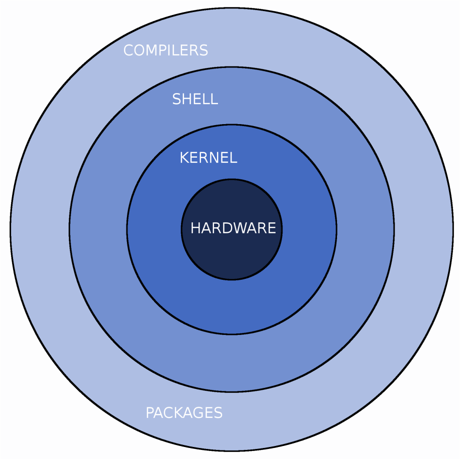
HPC2N,
UmeåUniversity,
901 87, Sweden.

January 21, 2018

Table of contents

1. Linux terminal
2. Pipes
3. File editing
4. Data handling
5. Environment variables
6. Scripting
7. Working with the Prompt

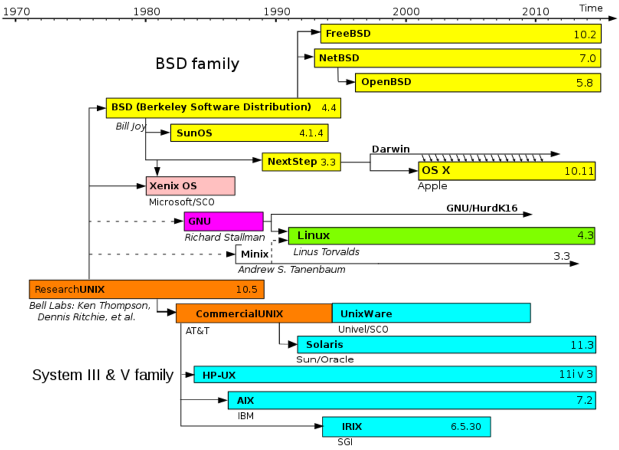
Linux OS



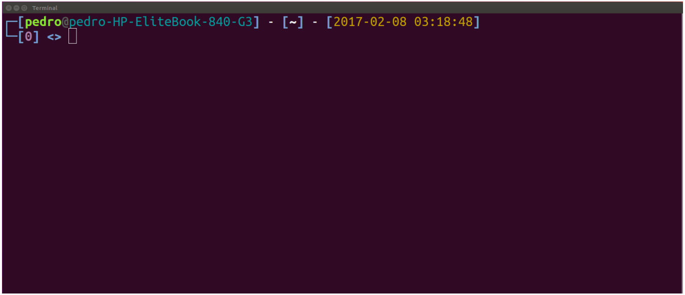
Linux OS

- UNIX-like OS
- used in modern Android smartphones
- the difference between UNIX OS is small

Linux timeline



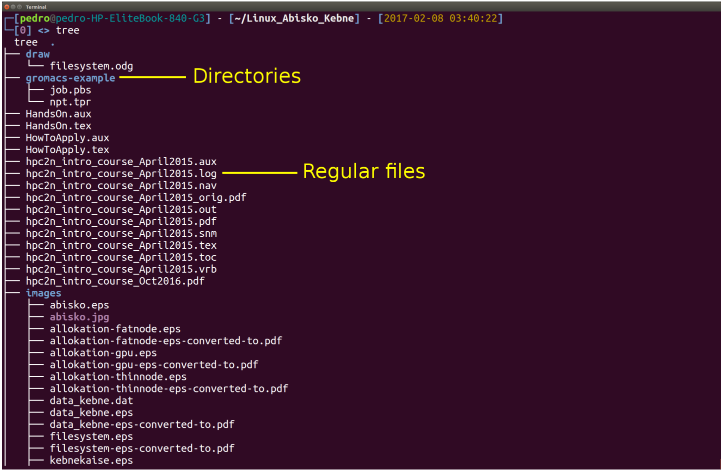
The Linux terminal



```
┌─[pedro@pedro-HP-EliteBook-840-G3] - [~] - [2017-02-08 03:18:48]
└─[0] <> 
```

- on the terminal you can see the so-called Prompt
- here you can control your PC/account or even a remote server

Files organization



```
┌─[pedro@pedro-HP-EliteBook-840-G3] - [~/Linux_Abisko_Kebne] - [2017-02-08 03:40:22]
└─[0] <> tree
  tree  .
├── draw
│   └── filesystem.odg
├── gromacs-example
│   ├── job.pbs
│   └── npt.tpr
├── HandsOn.aux
├── HandsOn.tex
├── HowToApply.aux
├── HowToApply.tex
├── hpc2n_intro_course_April2015.aux
├── hpc2n_intro_course_April2015.log
├── hpc2n_intro_course_April2015.nav
├── hpc2n_intro_course_April2015_orig.pdf
├── hpc2n_intro_course_April2015.out
├── hpc2n_intro_course_April2015.pdf
├── hpc2n_intro_course_April2015.snm
├── hpc2n_intro_course_April2015.tex
├── hpc2n_intro_course_April2015.toc
├── hpc2n_intro_course_April2015.vrb
├── hpc2n_intro_course_Oct2016.pdf
├── images
│   ├── abisko.eps
│   ├── abisko.jpg
│   ├── allokation-fatnode.eps
│   ├── allokation-fatnode-eps-converted-to.pdf
│   ├── allokation-gpu.eps
│   ├── allokation-gpu-eps-converted-to.pdf
│   ├── allokation-thinnode.eps
│   ├── allokation-thinnode-eps-converted-to.pdf
│   ├── data_kebne.dat
│   ├── data_kebne.eps
│   ├── data_kebne-eps-converted-to.pdf
│   ├── filesystem.eps
│   ├── filesystem-eps-converted-to.pdf
│   ├── kebnekaise.eps
```

Directories

Regular files

Useful commands: man

Manual pages.

- **man command: man nano**

```
NANO(1)                    General Commands Manual                   NANO(1)

NAME
       nano - Nano's ANOther editor, an enhanced free Pico clone

SYNOPSIS
       nano [options] [[+line,column] file]...

DESCRIPTION
       nano  is  a small, free and friendly editor which aims to replace Pico,
       the default editor included in the non-free Pine package.   On  top  of
       copying  Pico's  look  and  feel, nano also implements some missing (or
       disabled by default) features in Pico, such as "search and replace" and
       "go to line and column number".
```

Useful commands: ls

List the content of a directory

```
$ls
1CD9

$ls -l
total 24843644
drwxrwxr-x  2 pedro pedro        4096 nov  9 11:17 1CD9

$ls -la
total 24844368
drwxr-xr-x 44 pedro pedro        4096 feb 13 13:19 .
drwxr-xr-x  3 root  root         4096 sep 19 11:05 ..
drwxrwxr-x  2 pedro pedro        4096 nov  9 11:17 1CD9

$ls -lah
total 24G
drwxr-xr-x 44 pedro pedro 4,0K feb 13 13:25 .
drwxr-xr-x  3 root  root  4,0K sep 19 11:05 ..
drwxrwxr-x  2 pedro pedro 4,0K nov  9 11:17 1CD9
```

Useful commands: ls

```
$ls -laht
total 24G
drwxr-xr-x 44 pedro pedro 4,0K feb 13 13:29 .
-rw-------  1 pedro pedro 431K feb 13 13:29 .zsh_history
drwx------  6 pedro pedro 4,0K feb 13 13:28 Linux_Abisko_Kebne

$ls -lahrt
total 24G
-rw-r--r--  1 pedro pedro  655 sep 19 11:05 .profile
```

Useful commands: cd

Change directory.
Useful cases:

- cd directory
  move to "directory"
- cd
  move to $HOME$ directory
- cd -
  move to previous visited directory
- cd ..
  move to upper directory in the hierarchical tree

Useful commands: cp

Copy files.

Useful cases:

- cp text.txt directory/
  copy text.txt file to ”directory”
- cp -r test/ directory/
  copy the directory test into directory/.
  cp overwrites existing files!

Useful commands: touch/mkdir

Create files.
Useful cases:

- touch text.txt
  creates text.txt file
- mkdir test
  creates the directory test

Useful commands: rm

Remove files.
Useful cases:

- rm text.txt
  deletes text.txt file
- rm -rf test/
  deletes the directory test
  deleted files cannot be recovered!

Wild cards

- ?

  it represents a single character
- *

  it represents a string of characters
- $[0-9]\,,[A-B]$

  it represents a range of numbers or characters

Useful commands: grep

This command searches for patterns in text files.
Useful cases:

- grep 'word' file
  it searches for pattern 'word' in file
- grep -rine 'word' home
  pattern word is searched recursively in the directory /*home*

Useful commands: awk

This command finds patterns in a file and can perform arithmetic/string operations.
Useful cases:

- awk '/gold/ {*print*$1}' file
- it searches for pattern 'gold' in file and prints out the first column

Useful commands: ssh

Command for connecting to a remote computer.
Useful cases:

- ssh username@abisko.hpc2n.umu.se
  connecting to abisko machine
- ssh -Xl username abisko.hpc2n.umu.se
  if you want to enable graphical display.

Useful commands: sftp (scp)

Protocol for data transfer.

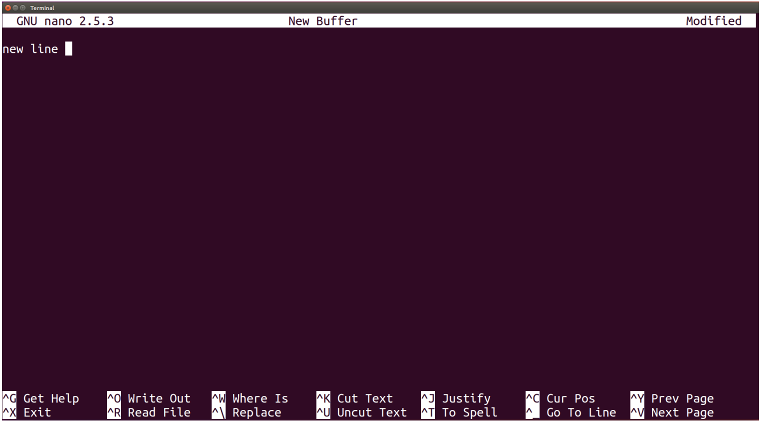
```
$sftp username@abisko.hpc2n.umu.se

$get file

$put file
```

Pipes

- One can use the output of some command as the input for someother command:

```
grep 'string' | wc
```

Editing files



```
  GNU nano 2.5.3                  New Buffer                                    Modified

new line 

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y Prev Page
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  ^V Next Page
```

Compress/decompress files

Compressing files:

```
$gzip file     --->  file.gz
```

Decompressing files:

```
$gunzip file.gz
```

Generating archives

Generate tar-ball:

```
$tar -cvf directory.tar directory
```

Opening tar-ball:

```
$tar -xvf directory.tar
```

Exporting variables

- some programs or libraries require environment variables to work
- they allow the program to follow different schemes without being re-compiled
- some variables such as $\$HOME$ are intrinsic to Linux OS
- we need to export the variables for further use:

```
$export NUMBER_OF_THREADS=6
```

Scripting

- allows to perform complex tasks without user intervention
- all Linux commands can be used in a script including wild cards

Scripting

analysis.sh

```
#!/bin/bash

grep 'ABCD' file.pdb >  file_filtered.pdb

program < file_filtered.pdb > output.dat
```

execute script with ./analysis.sh

Scripting

```
$ls -lah
total 24G
drwxrwxr-x  2 pedro pedro 4,0K nov  9 11:17 1CD9
```

- permissions are set of ”user”, ”group”, or ”others”
- we can change permissions with chmod command

For instance,

```
$chmod u+x analysis.sh
```

```
$execute script with ./analysis.sh
```

Prompt tips

- ctrl+r: Make a reverse search

```
export HISTCONTROL=erasedups
export HISTSIZE=100000
export HISTFILESIZE=100000
shopt -s histappend
```

- ctrl+a: Go to the beginning of the line
- ctrl+e: Go to the end of the line

Linux Cheat Sheet

- https://www.hpc2n.umu.se/documentation/guides/linux-cheat-sheet