Linux and Network Administration

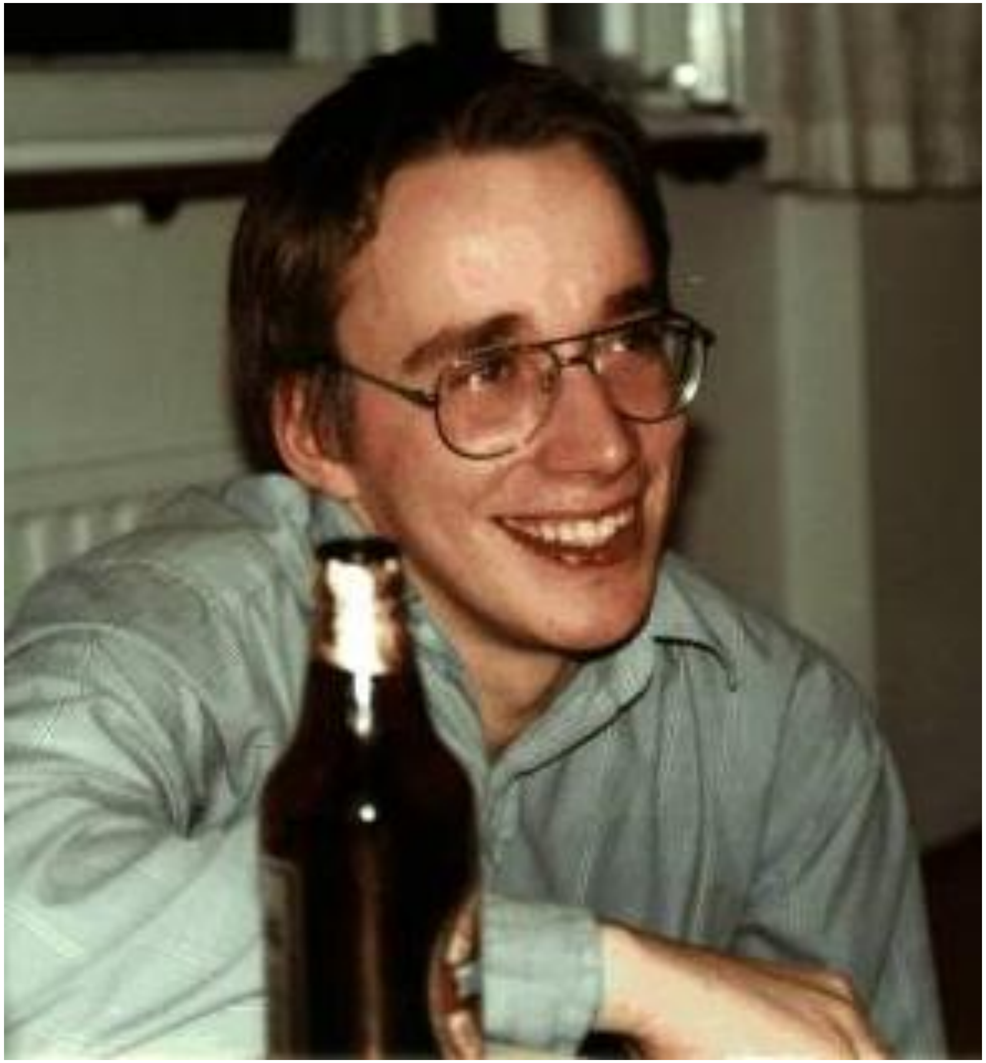
Lorenzo Bracciale

Marco Bonola

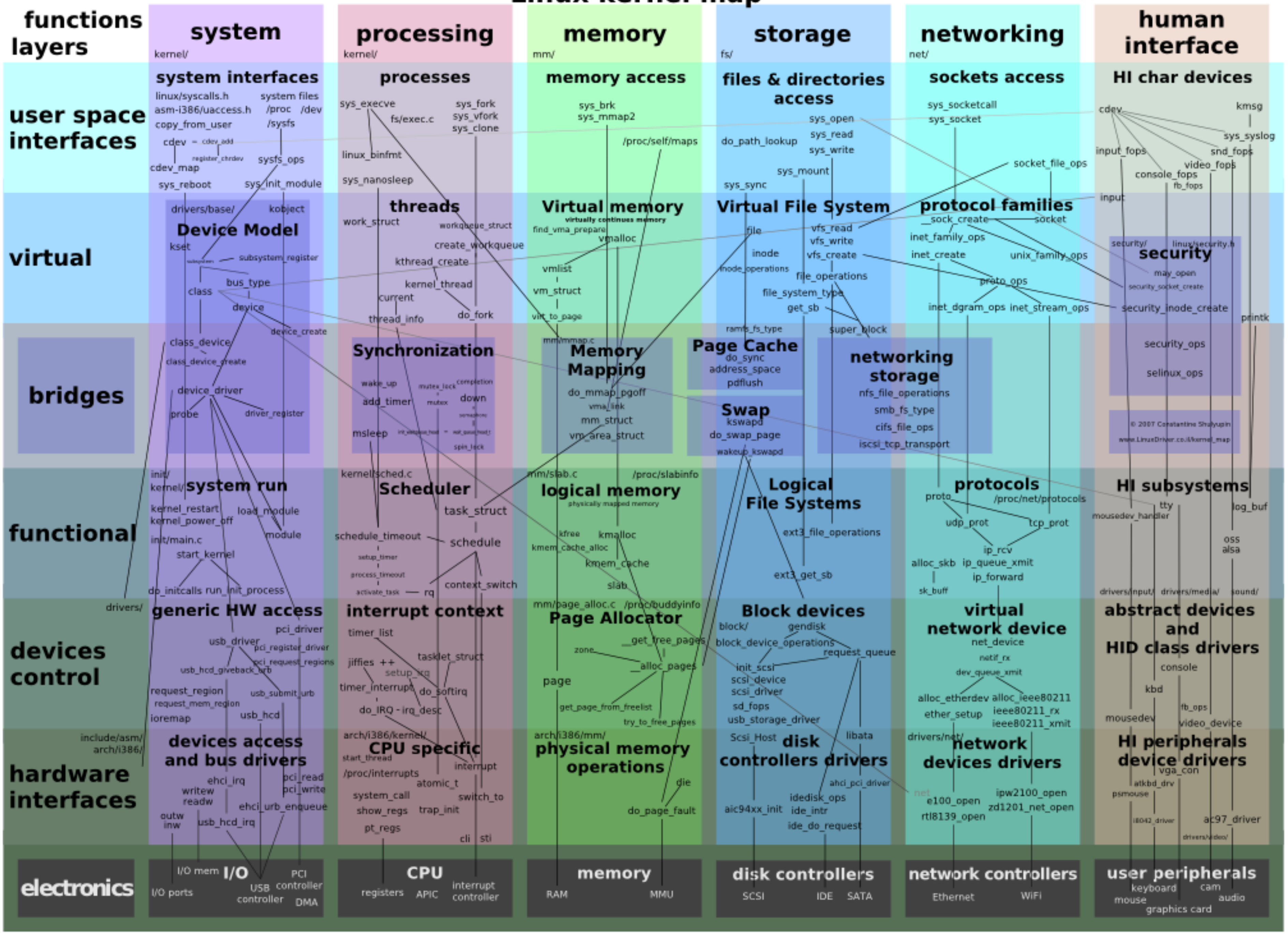
Outline

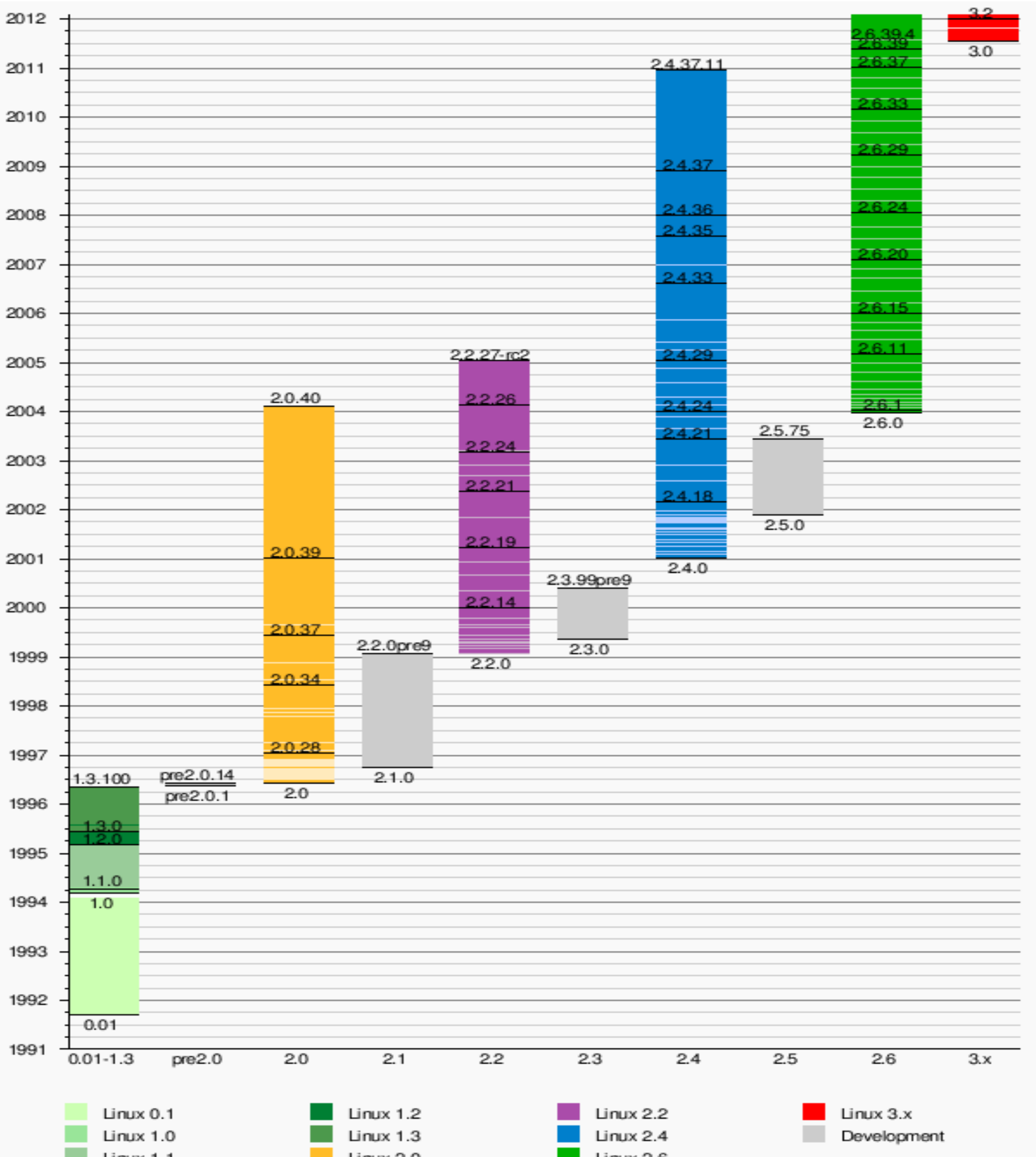
- *What is Linux?*

Who is this guy?



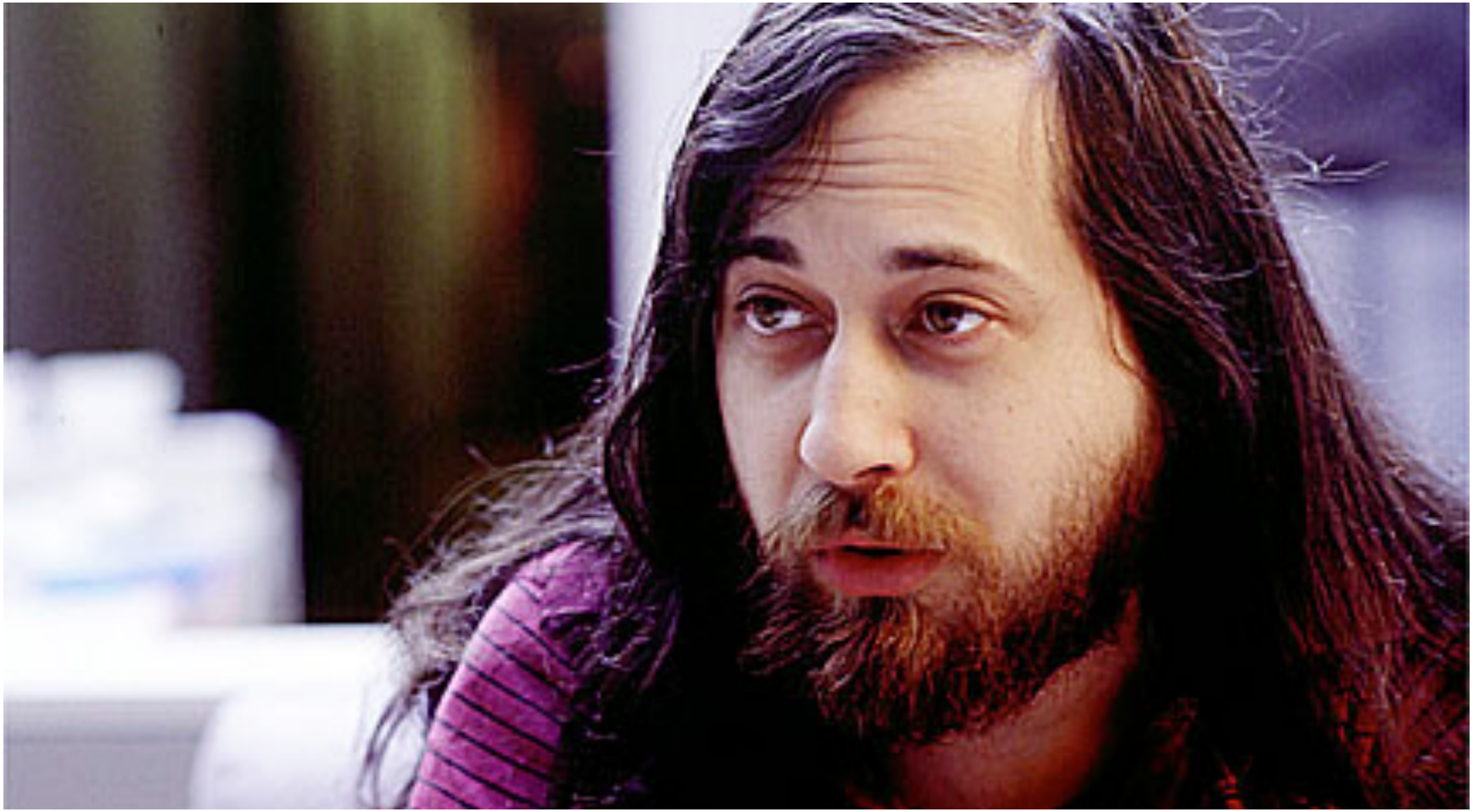
Linux kernel map

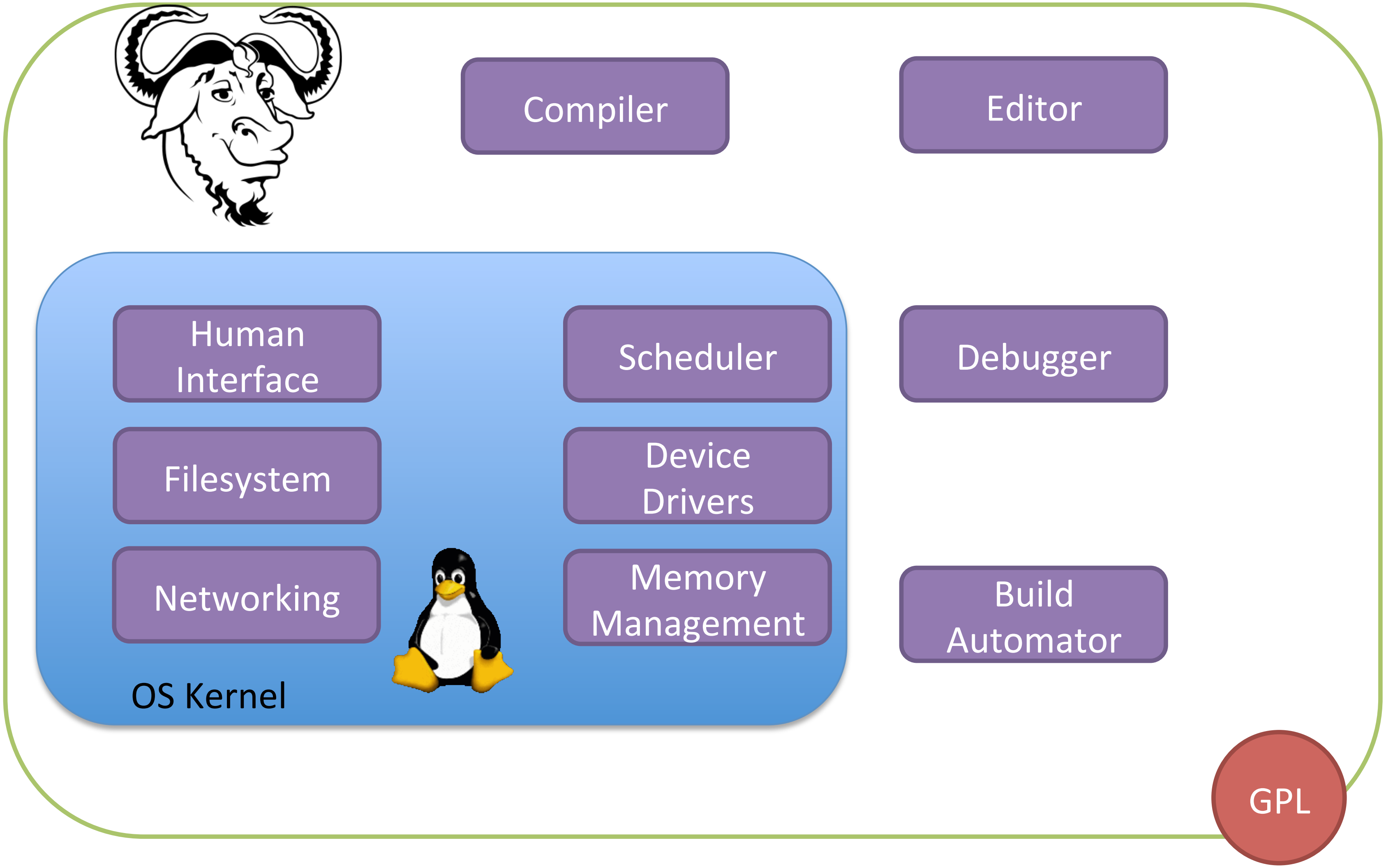


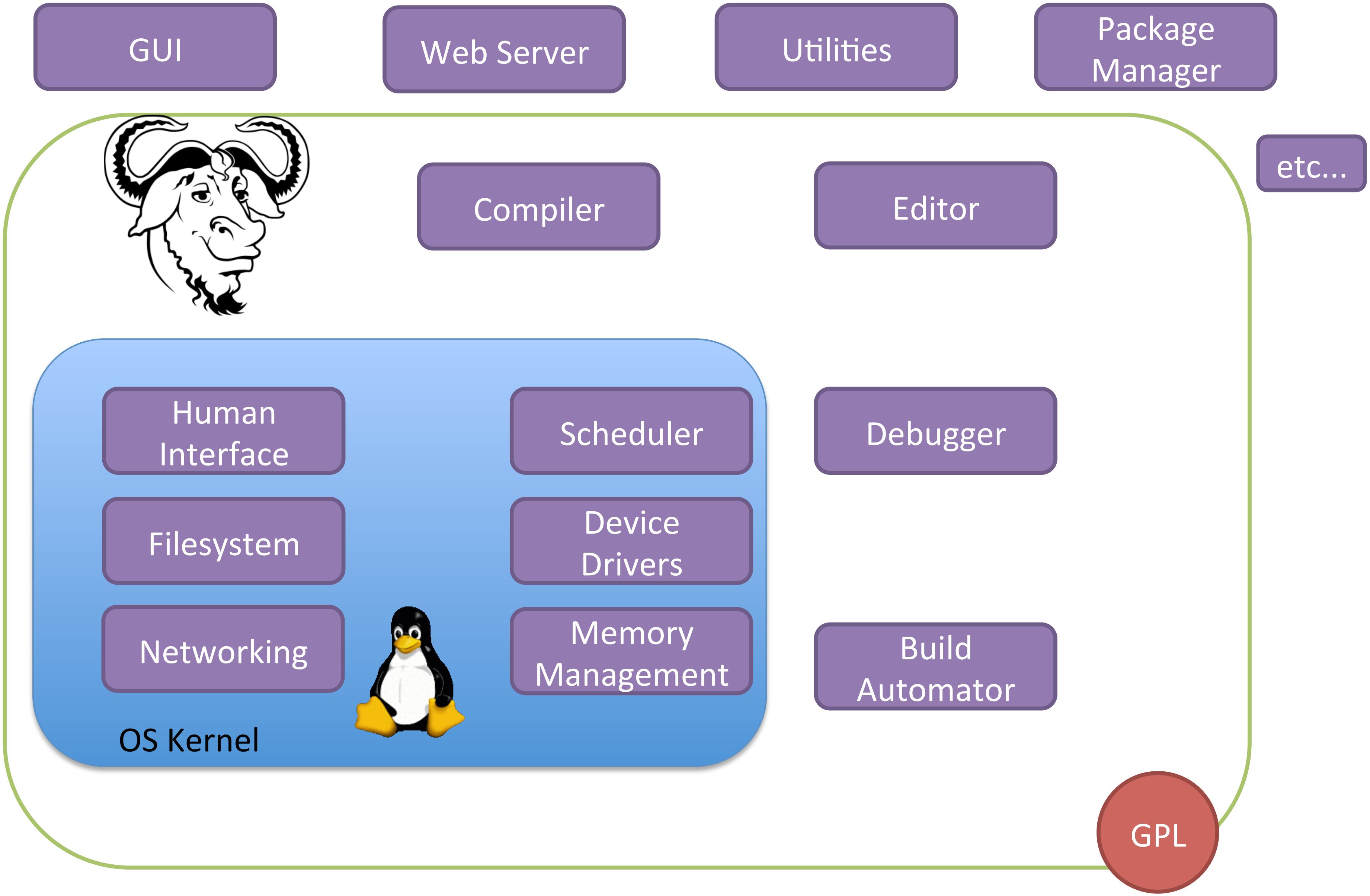


Updated 26/01/2012.

Who is this guy?




Compiler
Editor
Human Interface
Scheduler
Debugger
Filesystem
Device Drivers
Networking
Memory Management
Build Automator
OS Kernel
GPL



GUI

Web Server

Utilities

Package Manager

etc...

Compiler

Editor

Human Interface

Scheduler

Filesystem

Device Drivers

Networking

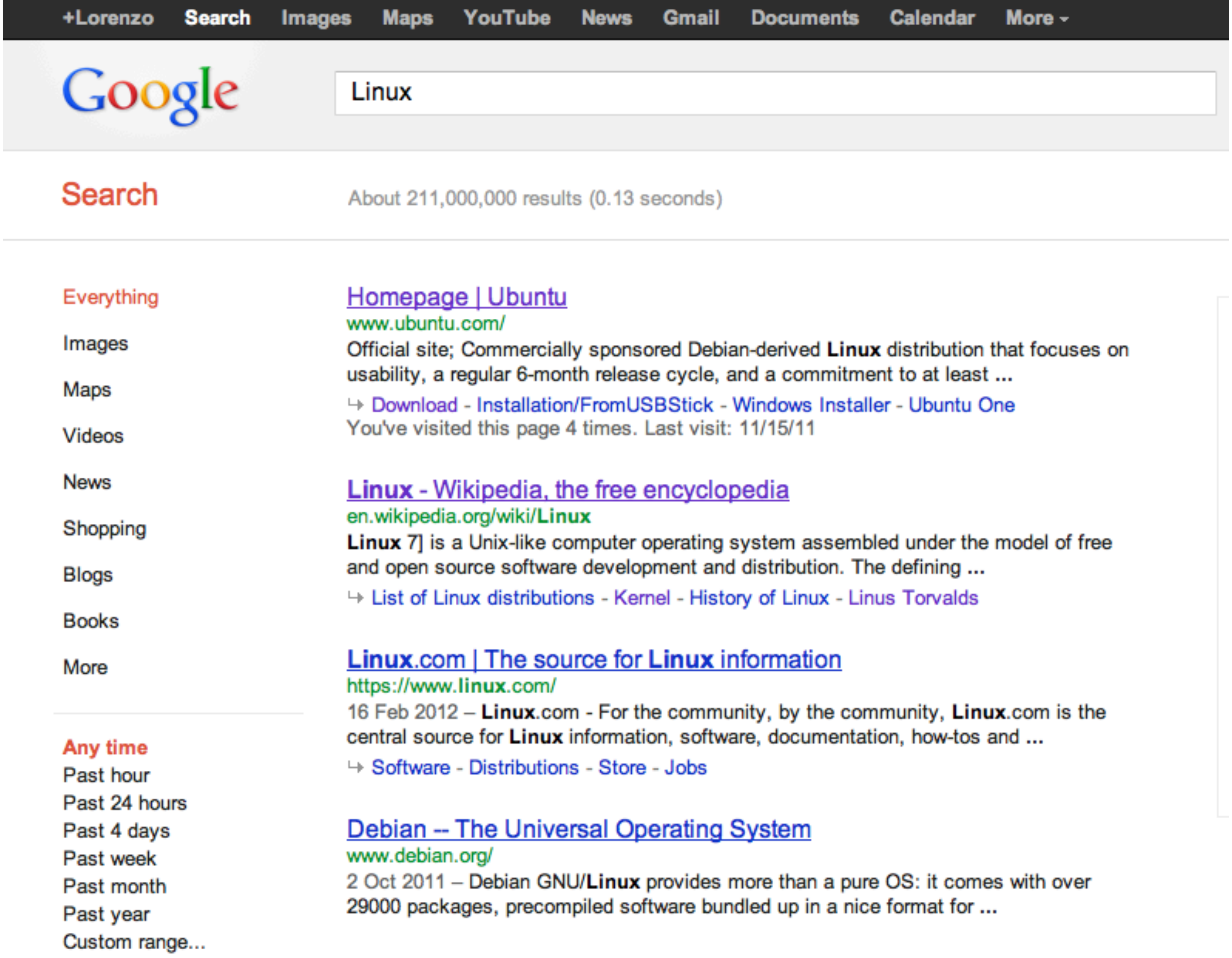
Memory Management

OS Kernel

Debugger

Build Automator

GPL



Linux

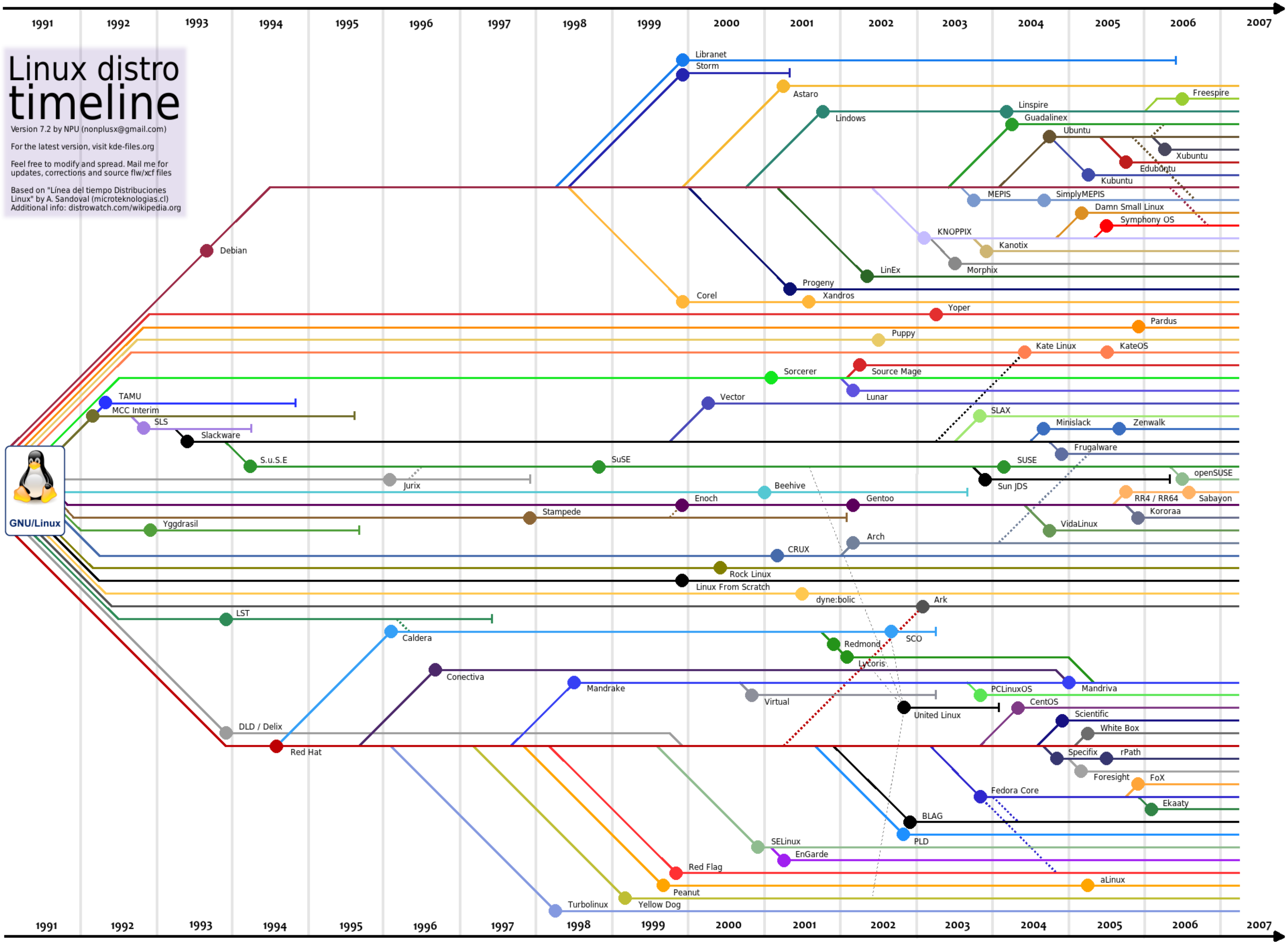
About 211,000,000 results (0.13 seconds)

Homepage | Ubuntu
www.ubuntu.com/
Official site; Commercially sponsored Debian-derived **Linux** distribution that focuses on usability, a regular 6-month release cycle, and a commitment to at least **...**
↳ Download - Installation/FromUSBStick - Windows Installer - Ubuntu One
You've visited this page 4 times. Last visit: 11/15/11

Linux - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Linux**
Linux 7] is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining **...**
↳ List of Linux distributions - Kernel - History of Linux - Linus Torvalds

Linux.com | The source for **Linux** information
https://www.**linux**.com/
16 Feb 2012 – **Linux**.com - For the community, by the community, **Linux**.com is the central source for **Linux** information, software, documentation, how-tos and **...**
↳ Software - Distributions - Store - Jobs

Debian -- The Universal Operating System
www.debian.org/
2 Oct 2011 – Debian GNU/**Linux** provides more than a pure OS: it comes with over 29000 packages, precompiled software bundled up in a nice format for **...**



Linux distro timeline

Version 7.2 by NPU (nonplusx@gmail.com)

For the latest version, visit kde-files.org

Feel free to modify and spread. Mail me for updates, corrections and source flw/xcf files

Based on "Línea del tiempo Distribuciones Linux" by A. Sandoval (microteknologias.cl)
Additional info: distrowatch.com/wikipedia.org

Outline

- What is Linux?
- *Shell*

Users

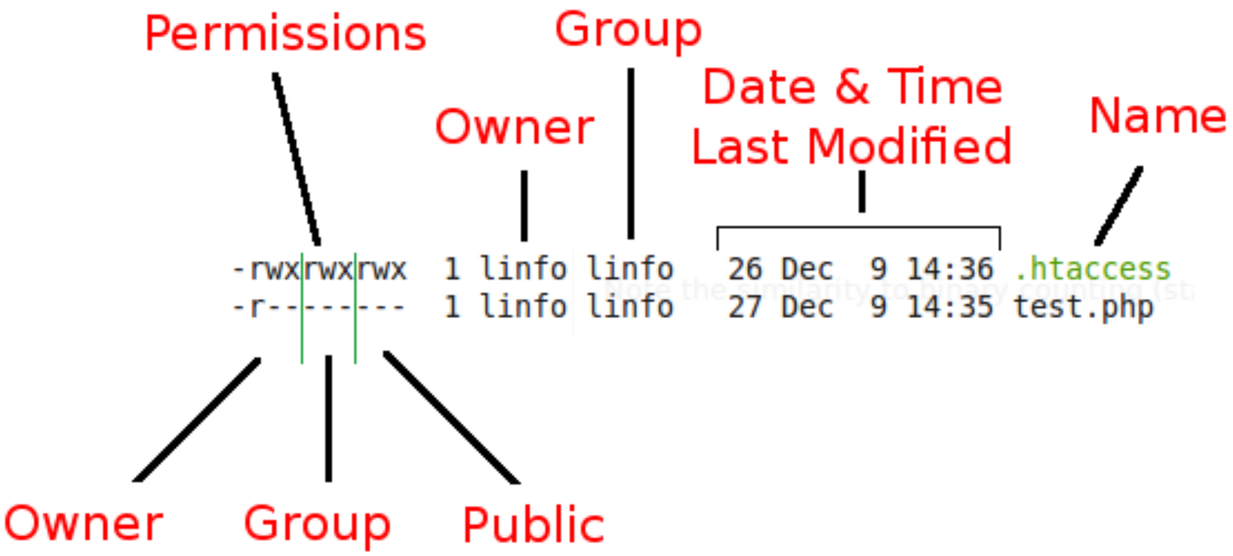
- Whoami?
  - Every user has an ID (UID) and belongs to one or more groups
  - Every groups has an ID (GID)
- See /etc/passwd and /etc/groups
- Related commands: adduser, userdel, su, sudo, whoami, who, last

Handling files

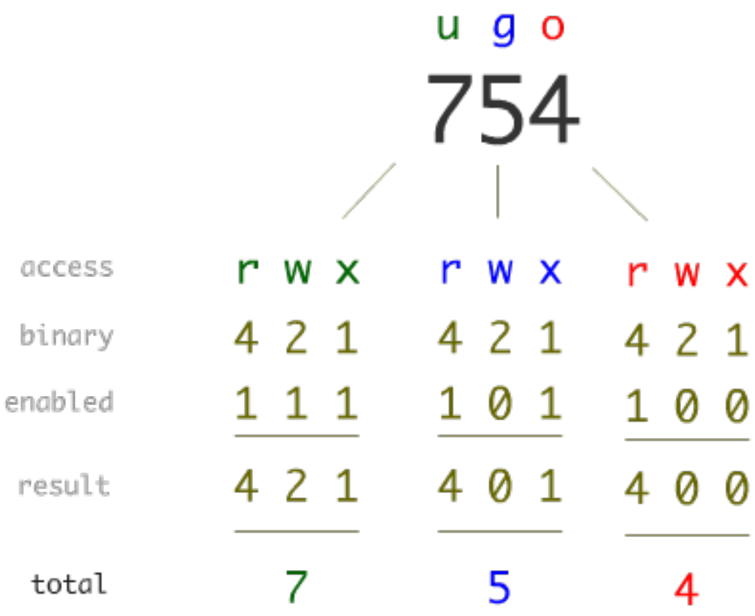
Name	Action
ls	list
cd	change directory
pwd	print working directory
cp/mv/rm	copy/move/remove
cat	concatenate
tail/head	view the first/last lines of a file
mkdir/rmdir	create/remove a directory
find/locate	search for a file/directory
grep	search inside files
ln	Link

MAN

File/Dir Permissions



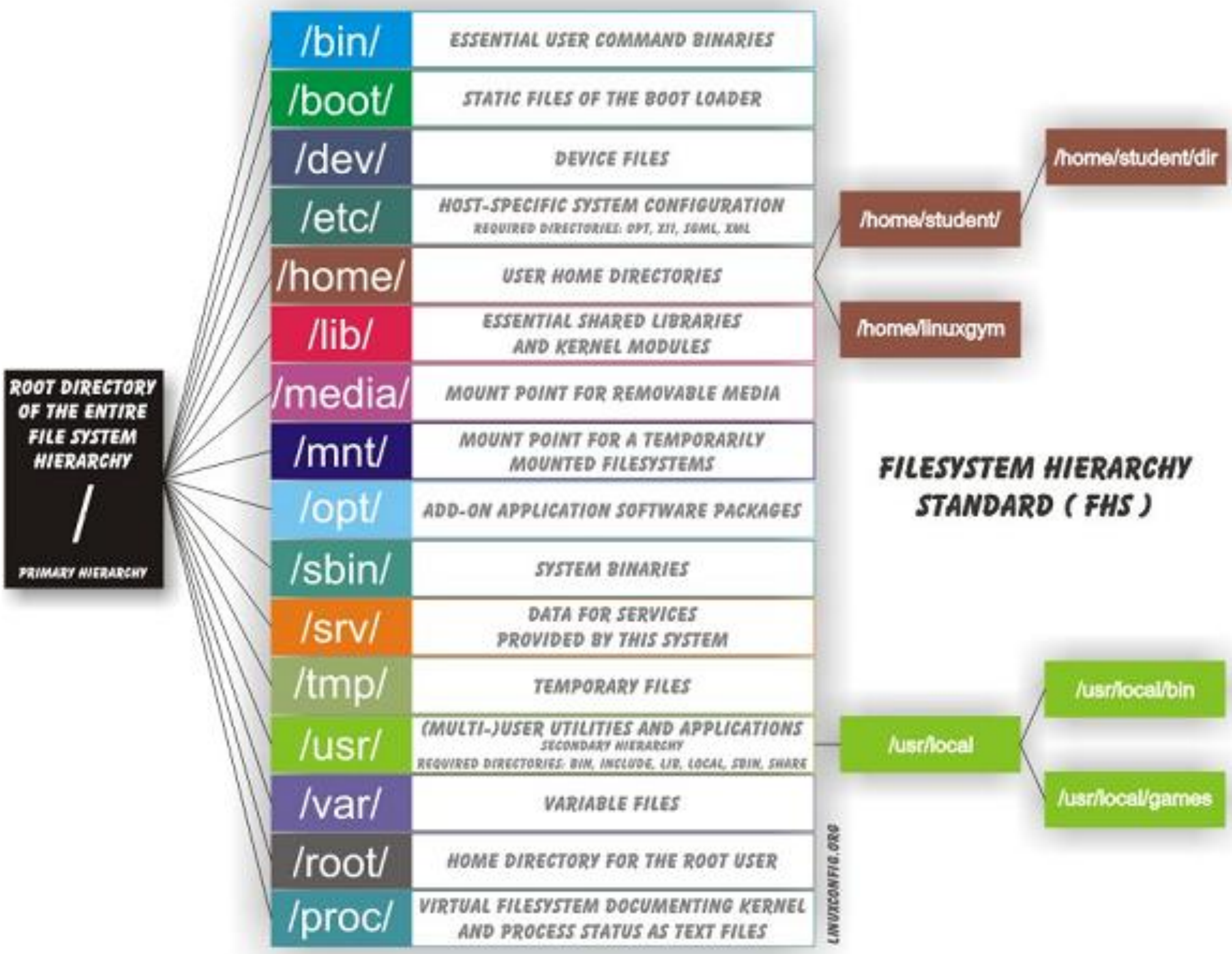
```
-rwxrwxrwx  1 linfo linfo   26 Dec  9 14:36 .htaccess
-r--------  1 linfo linfo   27 Dec  9 14:35 test.php
```



access	r	w	x	r	w	x	r	w	x
binary	4	2	1	4	2	1	4	2	1
enabled	1	1	1	1	0	1	1	0	0
result	4	2	1	4	0	1	4	0	0
total	7			5			4		

try with ls !

Linux Directory Structure



ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY

/

PRIMARY HIERARCHY

/bin/	ESSENTIAL USER COMMAND BINARIES
/boot/	STATIC FILES OF THE BOOT LOADER
/dev/	DEVICE FILES
/etc/	HOST-SPECIFIC SYSTEM CONFIGURATION REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/	USER HOME DIRECTORIES
/lib/	ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/	MOUNT POINT FOR REMOVABLE MEDIA
/mnt/	MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/	ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/	SYSTEM BINARIES
/srv/	DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/	TEMPORARY FILES
/usr/	(MULTI-)USER UTILITIES AND APPLICATIONS SECONDARY HIERARCHY REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/	VARIABLE FILES
/root/	HOME DIRECTORY FOR THE ROOT USER
/proc/	VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES

FILESYSTEM HIERARCHY STANDARD (FHS)

LINUXCONFIG.ORG

Proc FS

- easy way to view kernel and information about currently running processes.
- alternative to *sysctl*

Example: tell to the kernel to do NOT respond to ping

- sysctl -a | grep net.ipv4.icmp_echo_ignore_all
- sysctl -w net.ipv4.icmp_echo_ignore_all=1

or

- cat /proc/sys/net/ipv4/icmp_echo_ignore_all
- echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

Dev FS

- Access to physical devices (sound card, ram, hard drive, serial/parallel interface …) and “presudo” device (/dev/null, /dev/zero, /dev/random)
- Device manager: udev (daemon that speaks with kernet via netlink socket)

Example: Create 1 Gigaof random data

- dd if=/dev/random of=/home/myhome/randomdata bs=1M count=1024

Russian Roulette

```
dd if=/dev/urandom
of=/dev/kmem
bs=1 count=1
seek=$RANDOM
```

Adding pieces: mount

- mount -t type device dir (umount)
- in /dev/fstab information for startup mounting operations
- Files that contain filesystem can be mounted (-o loop)
  - It associates a file with a loop dev node (e.g. /dev/loop1) and …
  - mount the loop dev node to a mounting point

```
ninux@ale:~$ mount
/dev/sda1 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
```

Installing new software

- Dependencies problem
  - but also compiler version, available services…
- Gnu Build System
  - Autoconf, Automake, Libtools

```
./configure
make
make install
```

- On debian:
  - apt-get install foo
  - apt-cache search foo
  - apt-get update
  - apt-file search filename.txt

Archiving, Compression, Decompresson

- Tar: archive file/dir in one file .tar (no compression)
- Useful in combination with compression algorithm (most used: gunzip, bunzip2)
- Archive + gunzip:
  - tar cfvz nameofarchive.tar.gz target_dir
  - (For bunzip2 substitute z with j)
- Decompress
  - tar xvfz nameofarchive.tar.gz
- File extension helps but when in doubt use the "file" command
- Tar useful for logs (text files contains high redundancy)
  - See /var/logs
  - Lot of utilities: zcat, zless

Shell

What shell am I using?
echo $SHELL

- Basically a shell:
  - Allows executing programs
  - Allows set/get variable
  - Allows programming
- Accepts commands (executable programs)
  - absolute or relative path
  - commands in PATH
  - Which NAME_OF_COMMAND
- Variable:
  - **Shell variable** (local to a particular instance of the shell)
  - to list *set*, to set *VAR=VALUE*, to get *echo*
  - **Environment variable** (inherited by any program you start)
  - to list: *env,* to set *export* or *setenv*, to get *printenv* or *echo*

```
ninux@ale:~$ export PIPPO="pluto"
ninux@ale:~$ printenv PIPPO
pluto
```

```
ninux@ale:~$ PLUTO=”ciao"
ninux@ale:~$ echo $PLUTO
ciao
```

IO Channels

- When open every file…
  - stdin, stdout and stderr (FD 0, 1, 2)
- We can **redirect** the channels using major/ minor chars: < , << , >>, >
  - echo "hello world" > myfile
  - Set the stdout of echo to myfile
- n>&m allows to redirect FD n to FD m
  - program 2>&1 myfile
  - All output and error of a program to myfile

Pipelines

- Pipes allow separate processes to communicate without having been designed explicitly to work together.

- Example:
  - ls | grep x
  - Meaning : take the output of ls and give it as the input of grep

Shell tricks!

- ESC + . → repeat the last parameter
- CTRL + A → go to first char
- CTRL + E → go to last char
- CTRL + K → delete any char from the current position to the end of the line

- More? *man getline*

Processes

- Every process has an ID (PID)

- ps –aux → list processes
  - top or htop to see them in realtime
- Kill send signals to process (SIGTERM, SIGKILL)
- nice: set the niceness (useful for process realtime or cpu intensive)

Foreground, Background and Screen

- Some programs (e.g. tcpdump) inhibits you to give more commands on the same shell without interrupting the program
  - mycommand & → Put the command in background
  - fg → put the last command in foreground
  - CTRL+Z stop a program (to resume, fg)

- How keep a program running when we disconnect from the shell?

Foreground, Background and Screen

- Several solutions like *nohup* , *disown* , but the most confortable is *screen*
- *Example: create a named screen called pippo*
  - *screen –S pippo*
  - *top*
  - *C-a d* detach
  - *screen –r pippo (re-attach)*
  - *C-a c* → create
  - *C-a n (or p)* → next (or previous)

Editor

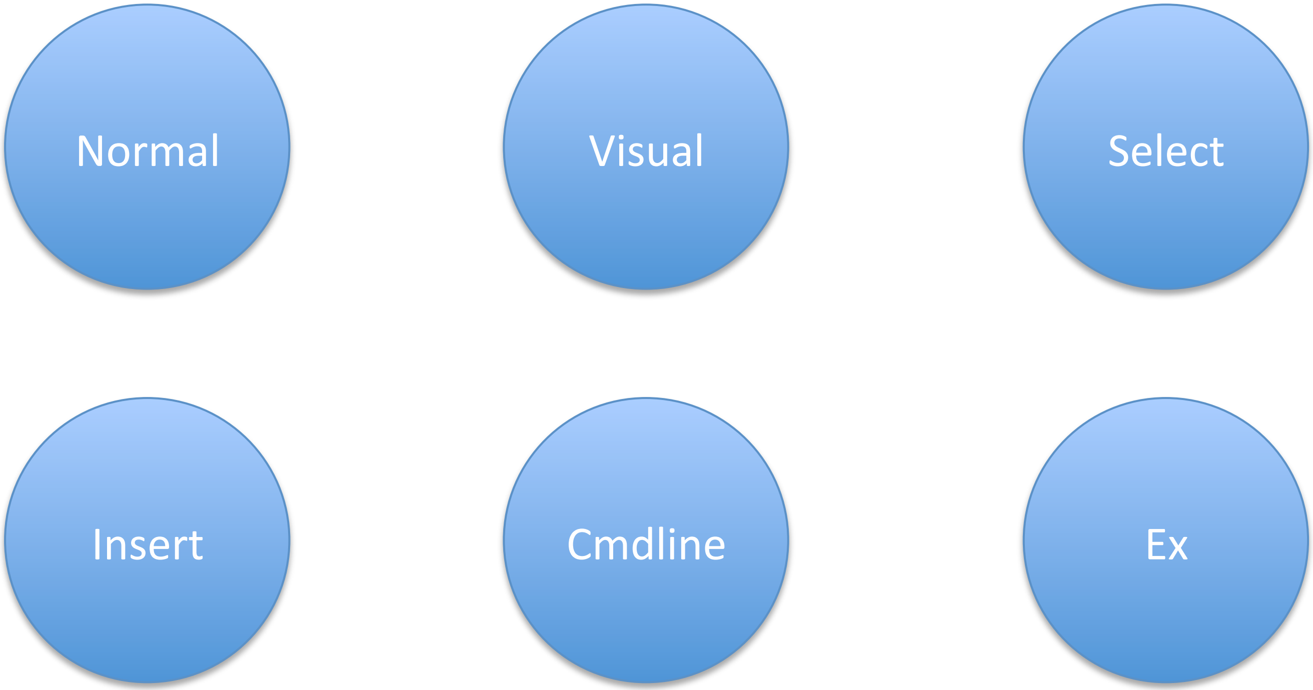
Editor war

From Wikipedia, the free encyclopedia

For a type of conflict between wiki edito

http://www.viemu.com/a-why-vi-vim.html

Vim Modes



Normal

Visual

Select

Insert

Cmdline

Ex

http://vimdoc.sourceforge.net/htmldoc/usr_toc.html
vimtutor

+6 additional modes!

Vim: Basic commands

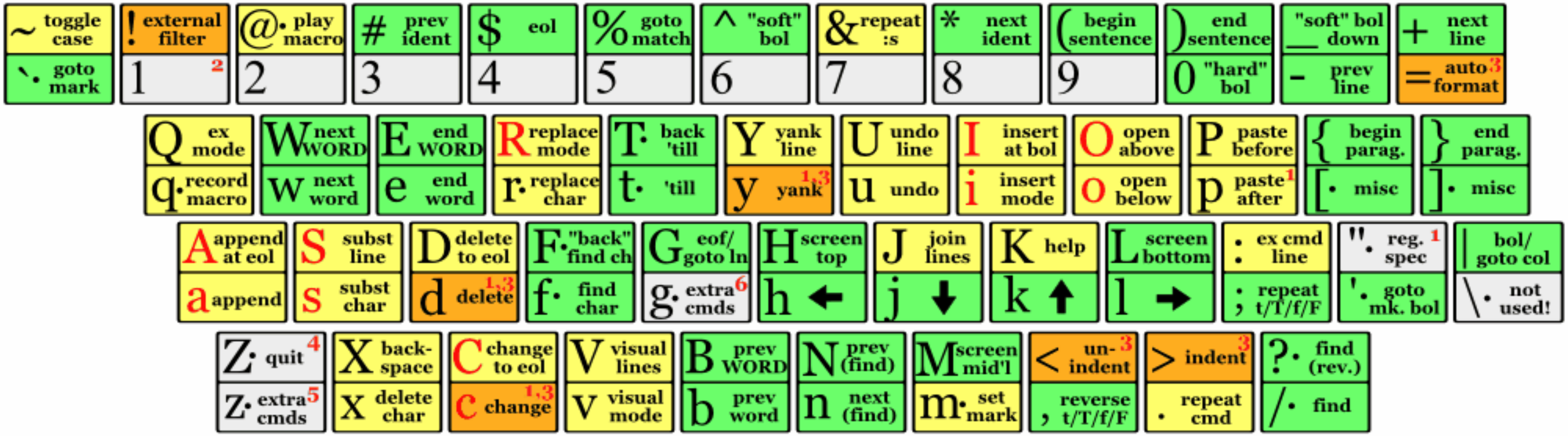
<Esc>	Enter command mode
i	Enter insert mode
:w	Save File
:q	Exit vim without saving
x	Delete the character under the cursor
dw	Delete the current word
dd	Delete the current line
d$	Delete everything Right of the cursor
yy	Yank the current line onto the clipboard
p	Paste the clipboard
u	Undo
:redo	Redo
G	Jump to bottom of file
/text	Search for the **text**from the cursor
>	Indent

set number
syntax on
Visual mode: markers
substitute all:
:%s/AAAA/BBBB/g

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc normal mode



~ toggle case
` · goto mark
! external filter
1 [2]
@· play macro
2
# prev ident
3
$ eol
4
% goto match
5
^ "soft" bol
6
& repeat :s
7
* next ident
8
(begin sentence
9
) end sentence
0 "hard" bol
_ "soft" bol down
- prev line
+ next line
= auto [3] format

Q ex mode
q· record macro
W next WORD
w next word
E end WORD
e end word
R replace mode
r· replace char
T· back 'till
t· 'till
Y yank line
y yank [1,3]
U undo line
u undo
I insert at bol
i insert mode
O open above
o open below
P paste before
p paste after [1]
{ begin parag.
[· misc
} end parag.
]· misc

A append at eol
a append
S subst line
s subst char
D delete to eol
d delete [1,3]
F· "back" find ch
f· find char
G eof/ goto ln
g· extra cmds [6]
H screen top
h ←
J join lines
j ↓
K help
k ↑
L screen bottom
l →
: ex cmd line
; repeat t/T/f/F
"· reg. spec [1]
'· goto mk. bol
| bol/ goto col
\· not used!

Z· quit [4]
z· extra cmds [5]
X back-space
x delete char
C change to eol
c change [1,3]
V visual lines
v visual mode
B prev WORD
b prev word
N prev (find)
n next (find)
M screen mid'l
m· set mark
< un-indent [3]
, reverse t/T/f/F
> indent [3]
. repeat cmd
?· find (rev.)
/· find

🟩 **motion** moves the cursor, or defines the range for an operator

🟨 **command** direct action command, if red, it enters insert mode

🟧 **operator** requires a motion afterwards, operates between cursor & destination

⬜ **extra** special functions, requires extra input

q· commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: quux(foo, bar, baz);
WORDs: quux(foo, bar, baz);

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top, zb: bottom, zz: center

(6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

Exercise

- Using vim write this file:

```
#!/bin/bash
echo Hello World
```

Then make it executable and launch your first script!

more on bash scripting: http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Bash scripting

- Variables:
  - modify the previous script in this way:

```
#!/bin/bash
STR="Hello World!”
echo $STR
```

Bash scripting

- If and arguments

```
#!/bin/bash
if [  $1!=“pippo” ]; then
        echo usage: $0 pippo
        exit
fi
echo You Win!
```

Bash scripting

- loop (for, while, until) and commands:
  - modify the previous script in this way:

```
#!/bin/bash -x
for i in $( ls ); do
          echo item: $i
done
```