

# Trabajando en GNU/Linux

Dr. Luis Gerardo de la Fraga

Departamento de Computación  
Cinvestav

Correo-e: [fraga@cs.cinvestav.mx](mailto:fraga@cs.cinvestav.mx)

Página WEB: <http://cs.cinvestav.mx/~fraga>

25 de octubre, 2016

## Contenido

1. Motivación
2. ¿Por qué usar GNU/Linux?
3. La lista de herramientas
4. Método de trabajo en GNU/Linux
5. Descripción de cada una de las herramientas
6. Conclusiones

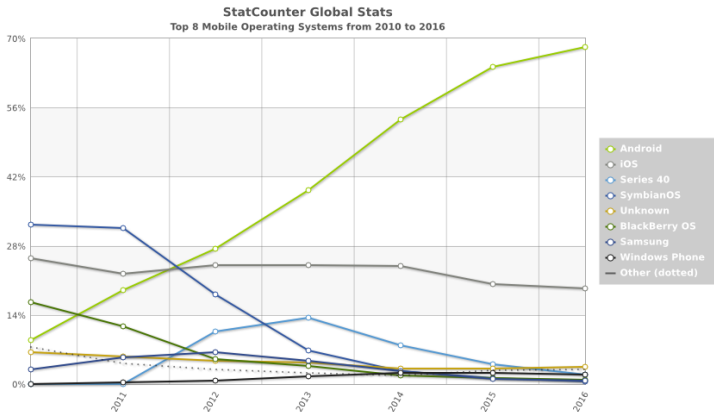
## Motivación

- ▶ Mostrar las herramientas de trabajo diario en GNU/Linux
- ▶ La lista de herramientas está basada en mi experiencia personal
- ▶ Yo he trabajado más de 23 años (desde 1993) con software libre y quiero compartirles esta experiencia de trabajo

## ¿Por qué trabajar en GNU/Linux?

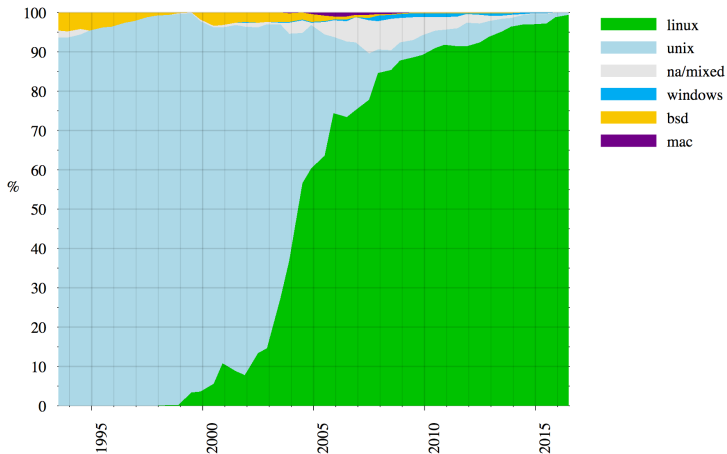
1. Es software libre, no se requiere pagar licencias de uso
2. Está disponible en Internet
3. Las herramientas que se verán son el resultado del trabajo de muchos años por una comunidad de programadores, lo que asegura que son de muy alta calidad
4. El **reto** es aprender a usarlo





<http://gs.statcounter.com>

# SOs en supercomputadoras de 1993 a 2016 de acuerdo con TOP500



By Benedikt.Seidl - Eigenes Werk (own work) / source top500.org,  
Public Domain, <https://commons.wikimedia.org/w/index.php?curid=5932978>  
[http://en.wikipedia.org/wiki/Usage\\_share\\_of\\_operating\\_systems](http://en.wikipedia.org/wiki/Usage_share_of_operating_systems)

## Las herramientas que veremos

Herramienta	Uso
L <sup>A</sup> T <sub>E</sub> X	Formateador de texto para crear documentos, presentaciones
gnuplot	Crear gráficas
xfig	Para crear figuras
ImageMagick	Para editar imágenes
latex2html	Para crear páginas WEB con ecuaciones
python	Programación de alto nivel
maxima	Manipulación simbólica
octave	Para algebra lineal. Usa la misma sintaxis que Matlab



## Método de trabajo en GNU/Linux (1/2)

- ▶ Excepto **xfig** todas las demás herramientas no tienen una interfaz gráfica
- ▶ El concepto de trabajo es distinto a las interfaces gráficas: no se tiene inmediatamente el resultado de la manipulación
- ▶ El concepto aquí es:
  - ▶ Usar cualquier editor de texto,
  - ▶ generar un script, que es un archivo con comandos de cada herramienta
  - ▶ utilizar los comandos para resolver nuestro problema
- ▶ Los editores clásicos son **emacs** y **vi**

## Método de trabajo en GNU/Linux (1/2)

- ▶ Desventaja: La curva de aprendizaje es más lenta y se realiza un mejor trabajo cuando más comandos se conocen de cada herramienta
- ▶ Ventaja: Se requiere mucho menos recursos computacionales de los que necesita una herramienta con interfaz gráfica
- ▶ Ventaja: Es un sistema operativo muy seguro

Un archivo básico de L<sup>A</sup>T<sub>E</sub>X:

```
\documentclass{article}

\begin{document}
\Huge
Este es un archivo de prueba
\end{document}
```

Si este script lo guardamos como el archivo `doc.tex`, un documento PDF se genera como:

```
pdflatex doc.tex
```

Este es un archivo de prueba

1

## En $\LaTeX$ lo que importa es la estructura del documento

Tipos de archivo	Formato
letter	ninguno
article	secciones (section)
report	partes, secciones (part)
book	capítulos, partes, secciones

También se pueden usar `\subsection{}` y `\subsubsection{}`  
beamer se usa para crear presentaciones.

Se pueden usar listas tipo `enumerate` e `itemize`:

```
\begin{enumerate}
```

```
\item Primero
```

1. Primero

```
\item Segundo
```

2. Segundo

```
\item Tercero
```

3. Tercero

```
\end{enumerate}
```

```
\begin{itemize}
```

▶ Primero

```
\item Primero
```

▶ Segundo

```
\item Segundo
```

▶ Tercero

```
\item Tercero
```

```
\end{itemize}
```

- ▶ Se pueden escribir negritas: `{\bf palabras en negrita}`  
**palabras en negrita**
- ▶ Palabras en itálicas: `{\em palabras así}`  
*palabras así*

Uno de los mejores aspectos de  $\text{\LaTeX}$  es la notación matemática:

▶ Una ecuación cuadrática:  $\$a x^2 + b x + c = 0\$$

▶ Una ecuación cuadrática  $ax^2 + bx + c = 0$

▶ Una matriz:

```
\[  
\begin{bmatrix}  
a_{11} & a_{12} \\ a_{21} & a_{22} \\ \end{bmatrix}  
\]
```

▶ Una matriz:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$



▶ `\int_{x=0}^2 e^x dx`

$$\int_{x=0}^2 e^x dx$$

▶ `\sum_{i=0}^{10} a_i x_i`

$$\sum_{i=0}^{10} a_i x_i$$

▶ `f(x) = \alpha \beta x ( \Delta x ) + 6`

$$f(x) = \alpha\beta x(\Delta x) + 6$$

▶ `x = r \cos(\theta); \ ; y= r \sin(\theta)`

$$x = r \cos(\theta); y = r \sin(\theta)$$

## El uso de etiquetas es algo muy poderoso en $\text{\LaTeX}$

- ▶ Se ponen etiquetas con `\label{eq:uno}` y se refieren con `\ref{eq:uno}`
- ▶ Se pueden poner etiquetas a:
  - ▶ Ecuaciones
  - ▶ Figuras
  - ▶ Secciones
  - ▶ Capítulos
- ▶ A números de página con `\pageref{etiqueta}`

- ▶ La parte del documento entre el `\documentclass` y el `\begin{document}` se le llama **preámbulo**
- ▶ Algunos de los paquetes útiles son:

```
\documentclass{article}
\usepackage[spanish]{babel}
\usepackage[utf-8]{inputenc}
\usepackage{graphicx}
\usepackage{amsmath}
\usepackage{url}
\usepackage{moreverb}
\usepackage{multicol}
```

```
\decimalpoint
```

```
\graphicspath{{Figs/}}
```

## gnuplot

- ▶ Este es un programa para generar gráficas
- ▶ Se puede usar de forma iterativa,
- ▶ pero es mejor usarlo a través de un script
- ▶ Página del proyecto: <http://gnuplot.info>

```
$ gnuplot
```

```
G N U P L O T
```

```
Version 4.6 patchlevel 5      last modified February 2014
```

```
Build System: Darwin x86_64
```

```
Copyright (C) 1986-1993, 1998, 2004, 2007-2014
```

```
Thomas Williams, Colin Kelley and many others
```

```
gnuplot home:      http://www.gnuplot.info
```

```
faq, bugs, etc:   type "help FAQ"
```

```
immediate help:   type "help" (plot window: hit 'h')
```

```
Terminal type set to 'aqua'
```

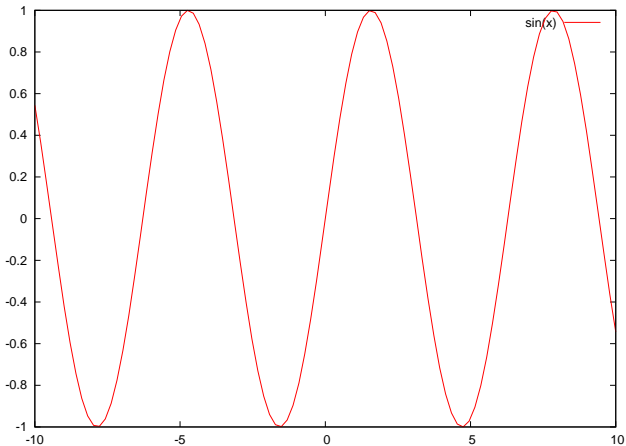
```
gnuplot> set term x11
```

```
Terminal type set to 'x11'
```

```
Options are 'nopersist'
```

```
gnuplot> plot sin(x)
```

```
gnuplot>
```



```
\begin{center}  
  \includegraphics[width=0.8\textwidth]{sin.pdf}  
\end{center}
```

## ¿Cómo graficar un círculo?

- ▶ La ecuación del círculo  $x^2 + y^2 = r^2$ , donde  $r$  es su radio, no es una función
- ▶ Para usar una función tenemos que usar su ecuación paramétrica:

$$x = r \cos \theta$$

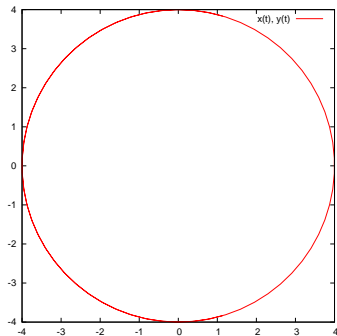
$$y = r \sin \theta$$

```
set term post eps color
set output "circulo.eps"
```

$$x(t) = 4*\cos(t)$$

$$y(t) = 4*\sin(t)$$

```
set size ratio -1
set parametric
plot x(t), y(t) w l lw 2
```

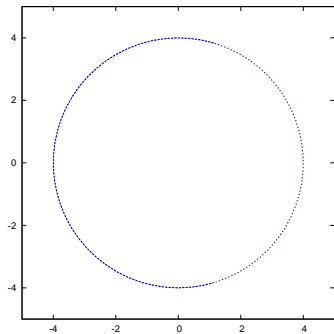




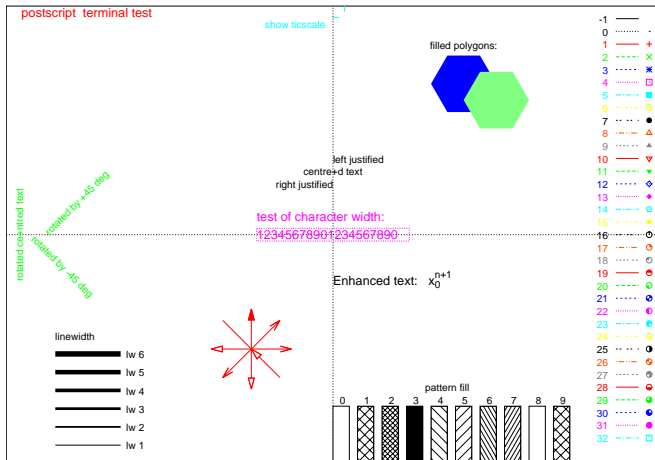
```
set term post eps color
set output "circulo2.eps"
```

```
x(t) = 4*cos(t)
y(t) = 4*sin(t)
```

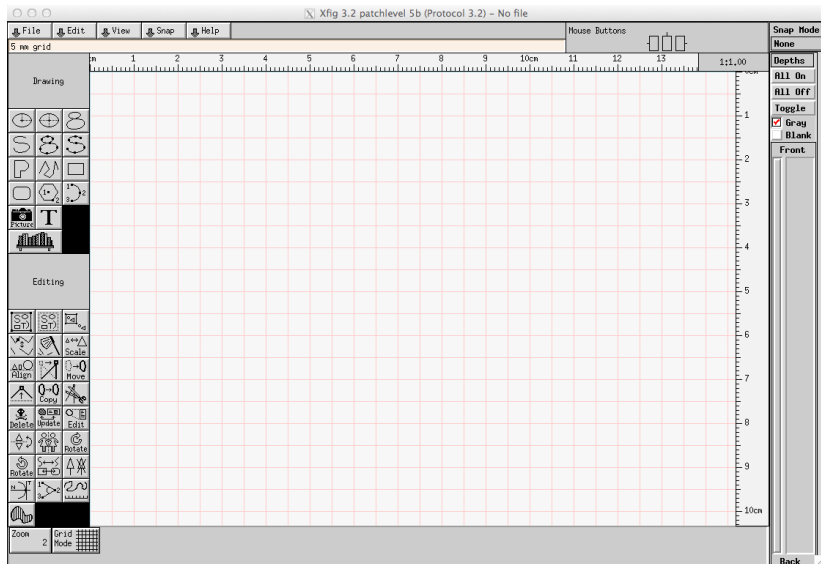
```
set size ratio -1
set parametric
unset key
set xrange [-5:5]
set yrange [-5:5]
plot x(t), y(t) w l lw 3 lt 3
```

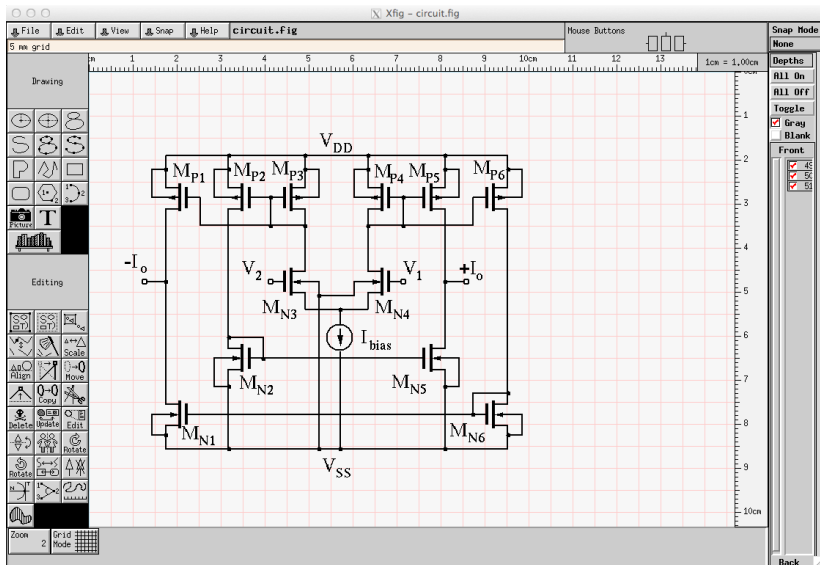


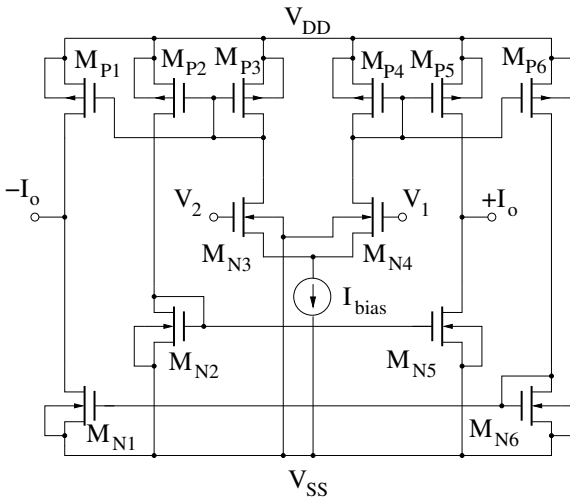
```
set term post eps color
set output "test.eps"
test
```



# Xfig, un programa para dibujar



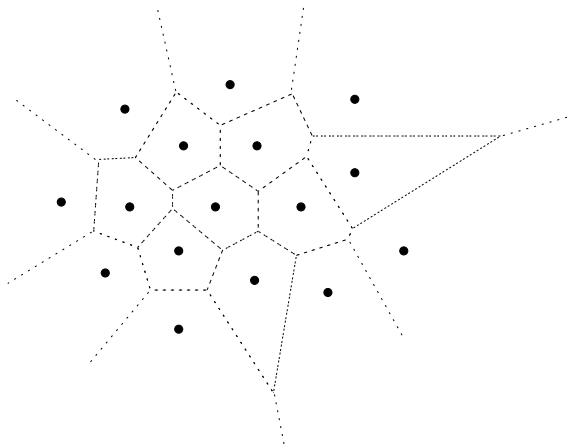


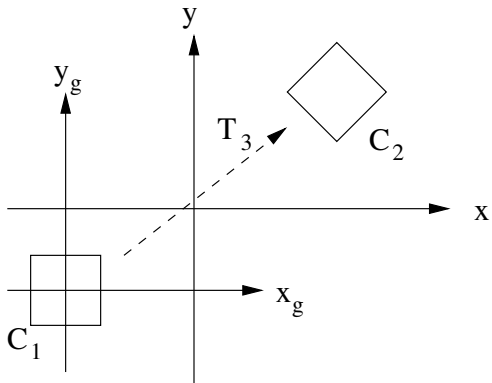


```

\begin{center}
\includegraphics[width=\textwidth]{circuit.pdf}
\end{center}

```





## ImageMagick

- ▶ Es un conjunto de programas para crear, modificar y desplegar imágenes
- ▶ Consta de los programas: display, identify, convert, montage y varios más.
- ▶ **display** sirve para desplegar imágenes
- ▶ **identify** sirve para identificar los archivos de imágenes
- ▶ **convert** para transformar imágenes



```
display marilyn.png
```



```
identify marilyn.jpg
marilyn.jpg JPEG 970x1221 970x1221+0+0 8-bit sRGB \
          91.1KB 0.000u 0:00.000
```

fraga@delta:~

```

Figs fraga$ display marilyn.jpg &
[1] 85294
Figs fraga$ █
  
```

gnuplot\_x11

fraga@delta:~

bash

ImageMagick: marilyn.jpg

Magnify 3X

```

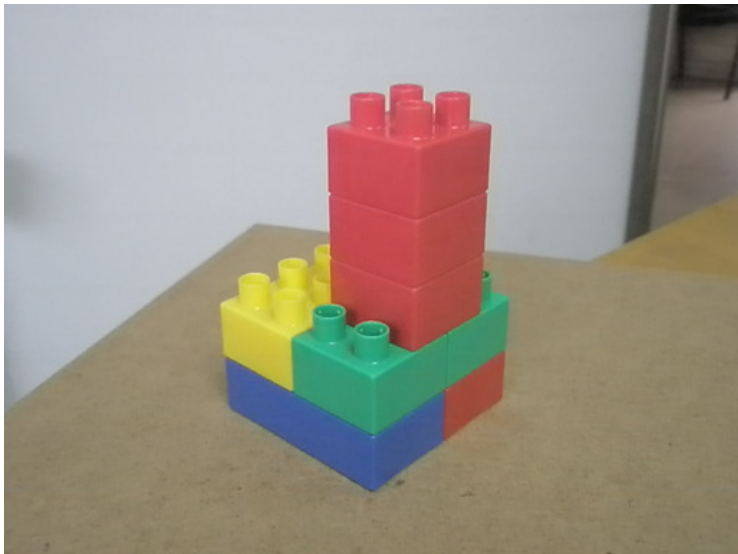
220,240: (111, 72, 14, 255)
#6F480E
srgba(111,72,14,1)
  
```

ImageMagick

- File
- Edit
- View
- Transform
- Enhance
- Effects
- FX
  - Solarize...
  - Image Edit
  - Miscellany
  - Help
- FX
  - Solarize...
  - Image Edit
  - Miscellany
  - Help
  - Oil Paint...
  - Charcoal Draw...
- Image Edit
  - Image Edit
  - Miscellany
  - Help
  - Oil Paint...
  - Charcoal Draw...
- Miscellany
  - Miscellany
  - Help
  - Oil Paint...
  - Charcoal Draw...
- Help
  - Help
  - Oil Paint...
  - Charcoal Draw...

1.2014

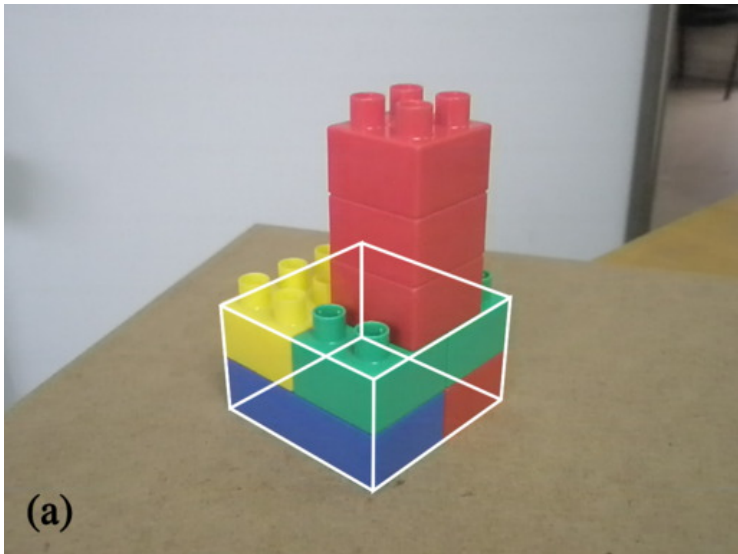
2.2014



## Un script del shell:

image.sh

```
# 704.331 408.317
# 691.428 550.065
# 513.35 676.969
# 514.152 521.65
# 497.054 331.66
# 497.414 462.708
# 314.454 562.35
# 301.286 419.149
# 0 768
cp a.png b.png
mogrify -fill none -stroke white -strokewidth 4 \
  -draw 'polyline 704,408 691,550 513,677 514,522 704,408 \
  polyline 497,332 497,463 314,562 301,419 497,332 \
  line 704,408 497,332 line 691,550 497,463 \
  line 513,677 314,562 line 514,522 301,419' \
  -stroke black -fill black -strokewidth 1 -font Times-Bold -pointsize
  -draw 'text 30,723 "a"' b.png
```



## latex2html

- ▶ Es un script en perl para convertir archivos en L<sup>A</sup>T<sub>E</sub>X a html
- ▶ Las opciones del script se ven con:

```
latex2html -h
```

## Lenguajes de alto nivel

- ▶ Perl
- ▶ Python
- ▶ Php
- ▶ Ruby



## ¿Porqué aprender un lenguaje de alto nivel?

- ▶ C es un lenguaje de nivel medio
- ▶ Los leguajes de alto nivel son interpretados
- ▶ No se definen tipos de datos
- ▶ Tienen una máquina de tiempo de ejecución (para uso de memoria con un recolector de basura)
- ▶ Es fácil construir programas rápidamente
- ▶ Son los lenguajes más usados

## Prototipado rápido

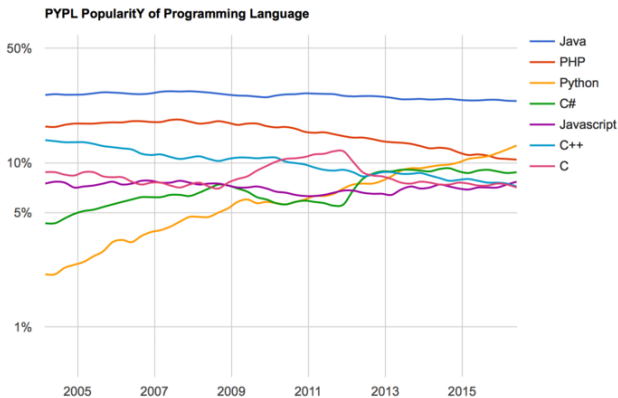
- ▶ Son ideales para realizar programas que procesan texto y generan texto
- ▶ Para generación dinámica de páginas WEB
- ▶ Si se tiene una idea, puede obtenerse un programa funcional en minutos u horas

## Índice de Popularidad de los Lenguajes de Programación

PYPL PopularitY of Programming Language index

<http://pypl.github.io/PYPL.html>

Posición Jul 2015	Cambio Jul 2016	Lenguaje programación	Porción	Tendencia
1		Java	23.9 %	-0.5 %
2	↑	Python	12.8 %	+2.1 %
3	↓	PHP	10.5 %	-0.8 %
4		C#	8.8 %	-0.5 %
5	↑↑	Javascript	7.7 %	+0.6 %
6	↓	C++	7.2 %	-0.4 %
7	↓	C	7.1 %	-0.1 %
8		C-Objetivo	4.7 %	-0.6 %
9	↑	R	3.1 %	+0.5 %
10	↑	Swift	3.0 %	+0.4 %



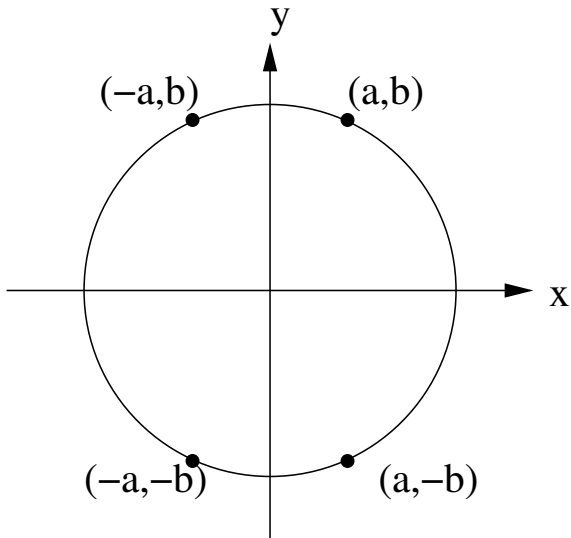
PYPL Popularity of Programming Language index <https://sites.google.com/site/pydata/pypl/PyPL-Popularity-of-Programming-Language>

<http://www.raspberrypi.org/>



Se puede comprar en [www.newark.com.mx](http://www.newark.com.mx)

## Problema: trazar un círculo



## Programa en python

```
import math

R2 = 17*17
x=0
while x<=17 :
    y = math.floor( math.sqrt(R2 - x*x) + 0.5 )
    print x, y
    x += 1
```

```
python p.py > p.txt
```

## Script para gnuplot:

```
set term post eps color
set output "c.eps"

set xrange [-1:18]
set yrange [-1:18]

x(t) = 17*cos(t)
y(t) = 17*sin(t)

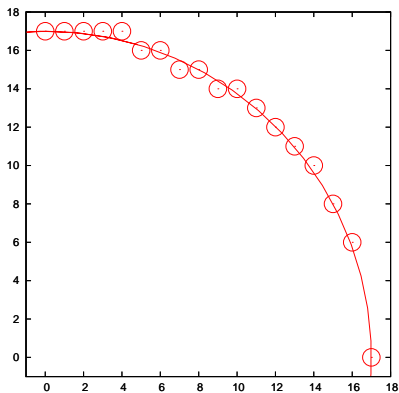
unset key
set size ratio -1
set multiplot
plot 'p.txt' w p pt 6 ps 3.5

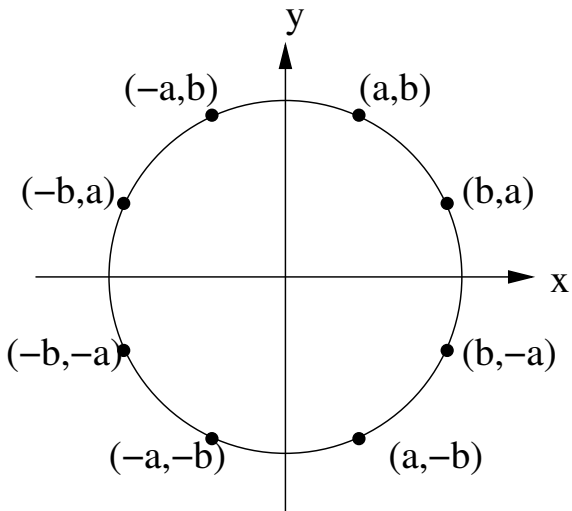
set parametric
plot x(t), y(t)

unset multiplot
```



## La gráfica hecha con gnuplot:





## Maxima

- ▶ Maxima es un programa que usa manipulación simbólica
- ▶ <http://maxima.sourceforge.net>

## Archivo m.max:

```
m.max  
poli: (a+b);  
  
f: poli*poli;  
  
e: expand(f);
```

Lo ejecutamos como:

```
maxima -b m.max
```

La salida:

```
maxima -b m.max
Maxima 5.30.0 http://maxima.sourceforge.net
using Lisp SBCL 1.1.16
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) batch("m.max")

read and interpret file: m.max
(%i2) poli:b+a
(%o2)
      b + a
(%i3) f:poli*poli
(%o3)
      2
      (b + a)
(%i4) e:expand(f)
(%o4)
      2      2
      b  + 2 a b + a
(%o4)      m.max
```

## Otro ejemplo:

$$\begin{bmatrix} \frac{xm}{sx} & 0 & -\frac{xm}{sx} \\ 0 & \frac{ym}{yx} & -\frac{ym}{sy} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b0 & b1 & b2 \\ b3 & b4 & b5 \\ b6 & b7 & b8 \end{bmatrix} \begin{bmatrix} \frac{sx}{xm} & 0 & 1 \\ 0 & \frac{sy}{ym} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

```
B: matrix([b0, b1, b2], [b3, b4, b5], [b6, b7, b8]);
```

```
Ti: matrix ( [ xm/sx, 0, -xm/sx], [0, ym/sy, -ym/sy], [0, 0, 1]
```

```
T: matrix ( [ sx/xm, 0, 1], [0, sy/ym, 1], [0, 0, 1] );
```

```
R: Ti . B . T;
```

```
/**
```

```
C: Ti^^(-1);
```

```
A: Ti.T;
```

```
**/
```

## Salida:

(%i5) R:Ti . B . T

$$\begin{bmatrix} b0 - b6 & ] & [ b1 \text{ sy xm} & b7 \text{ sy xm} & ] \\ [ & ] & [ \text{-----} & - \text{-----} & ] \\ [ b3 \text{ sx ym} & b6 \text{ sx ym} & ] & [ \text{sx ym} & \text{sx ym} & ] \\ [ \text{-----} & - \text{-----} & ] & [ & & ] \end{bmatrix}$$

(%o5) Col 1 =  $\begin{bmatrix} \text{sy xm} & \text{sy xm} & ] \\ [ & & ] \\ [ b6 \text{ sx} & & ] \\ [ \text{-----} & & ] \\ [ \text{xm} & & ] \end{bmatrix}$  Col 2 =  $\begin{bmatrix} b4 - b7 & ] \\ [ & ] \\ [ b7 \text{ sy} & ] \\ [ \text{-----} & ] \\ [ \text{ym} & ] \end{bmatrix}$

$$\begin{bmatrix} (b2 + b1 + b0) \text{ xm} & (b8 + b7 + b6) \text{ xm} & ] \\ [ \text{-----} & - \text{-----} & ] \\ [ \text{sx} & \text{sx} & ] \\ [ & & ] \\ \text{Col 3} = [ (b5 + b4 + b3) \text{ ym} & (b8 + b7 + b6) \text{ ym} & ] \\ [ \text{-----} & - \text{-----} & ] \\ [ \text{sy} & \text{sy} & ] \\ [ & & ] \\ [ b8 + b7 + b6 & & ] \end{bmatrix}$$

## GNU Octave

- ▶ Octave es un lenguaje de programación de alto nivel
- ▶ Su propósito general es para cálculo numérico
- ▶ <https://www.gnu.org/software/octave>



## Un script para octave:

```
A = [1 1 1;  
     2 1 1;  
     0 1 3];
```

```
b = [0; 1; 1];
```

```
x = A \ b
```

```
A * x
```

- ▶ Se ejecuta como:  
octave -qf p.oct
- ▶ La salida:

```
x =
```

```
1  
-2  
1
```

```
ans =
```

```
0  
1  
1
```

## Conclusiones

- ▶ Se mostraron varias herramientas para trabajo de uso diario el GNU/Linux
- ▶  $\text{\LaTeX}$ , para editar documentos
- ▶ **gnuplot**, para realizar gráficas
- ▶ **xfig**, para realizar figuras
- ▶ **ImageMagick**, para editar imágenes
- ▶ **python**, para programación en alto nivel y análisis numérico
- ▶ **maxima**, para cálculo simbólico
- ▶ **octave**, para análisis numérico

## Trabajo futuro:

1. Instalación de alguna distribución de GNU/Linux (Debian, Ubuntu o Fedora)
2. Administración de un sistema GNU/Linux: instalación de nuevos paquetes, manejo de respaldos, manejo de dispositivos de entrada/salida.
3. Un taller para usar  $\text{\LaTeX}$ , gnuplot y xfig
4. Un taller para usar python

¡Gracias!