FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS
DEPARTAMENTO DE BIOTECNOLOGÍA Y TECNOLOGÍA ALIMENTARIA
UNIVERSIDAD ARGENTINA DE LA EMPRESA

# Bioinformática

## ANÁLISIS COMPUTACIONAL DE SECUENCIAS

**Dr. Lucas L. Maldonado (PhD)**

**Lic. Biotechnologist and Molecular Biologist**

**Bioinformatics and genomics specialist**

CONICET
Fac. de Medicina - UBA
Fac. de Ciencias Exactas y Naturales – UBA

lucamaldonado@uade.edu.ar
lmaldonado@fmed.uba.ar
lucas.l.maldonado@gmail.com.ar

# CRONOGRAMA TENTATIVO – 1era parte

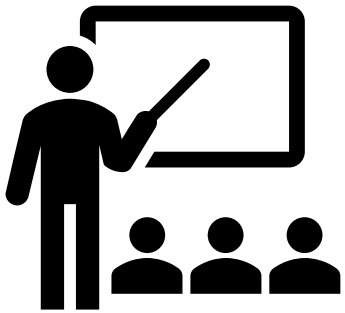
| Clase | Fecha | Temas/Bibliografía/Actividades/Recursos/Evaluaciones |
|---|---|---|
| 1 | 15-mar | Introducción a la bioinformática. Presentación de los contenidos y temas de la materia y estructura de las clases. Introducción a Linux para Bioinformáticos y formatos de archivos.TP. 1:Linux parte1, Familiarizarse con Ubuntu y la Terminal de Linux. Utilizar comandos comunes en Bioinformática, por ejemplo, GREP. Edición de secuencias con el formato adecuado para utilizar los comandos de EMBOSS en la Terminal y conocer formatos de archivos comunes en BI |
| 2 | 22-mar | Introducción al diseño y manejo de bases de datos de biología molecular. Motores de búsqueda, ENTREZ, Google. Base de datos de genomas particulares. Bases de datos (SQL o PostrgeSQL) y esquema (Chado). TP. 2:Usar bases de datos vía web. Ejercicios usando GenBank y Gquery. |
| 3 | 29-mar | Alineamientos de secuencias I:Introducción a los conceptos de homología, ortólogos, parálogos. Algoritmos para alineamiento de secuencias apareado (Global y Local). Introducción al “Dot plot” y Programación dinámica. Matrices de amino ácidos PAM y Blossum. TP. 3:Ejercicios de distintos tipo de alineamiento desde la línea de comandos. |
| 4 | 05-abr | Alineamientos de secuencias II: Algoritmos para alineamiento de secuencias apareado (Global y Local). Introducción al “Dot plot” y Programación dinámica. BLAST. Alineamientos Globales vs Locales: cuándo y por qué usa uno? Parámetros de evaluación (E value, Score, etc.). TP. 4: Ejercicios de uso de programas: BLAST |
| 5 | 12-abr | Alineamiento de secuencias III: Matrices de puntaje, parte 2: Alineamientos multiple de secuancias (MSA). Análisis y clasificación de proteínas. Métodos y algoritmos de MSA. TP. 5: Ejercicios ONLINE y por linea de comandos |
| 6 | 19-abr | Introducción a los conceptos de Super-familias y sub-familias. Identificacion de Dominios y Patrones en secuencias de proteínas. Perfiles. Métodos de identificación y representación de patrones. Bases de Datos Secundarias. (PFAM, PROSITE). TP. 6: Ejercicios MSA y búsquedas de patrones en secuencias |
| 7 | 26-abr | Bioinformática estructural: Estructura de proteínas. Métodos de análisis de estructuras proteicas. Conceptos de dinámica molecular.<br>TP 7: Modelado por homología |
| 8 | 03-may | Puesta en común de TPs |
| 9 | 10-may | EXAMEN PARCIAL I |

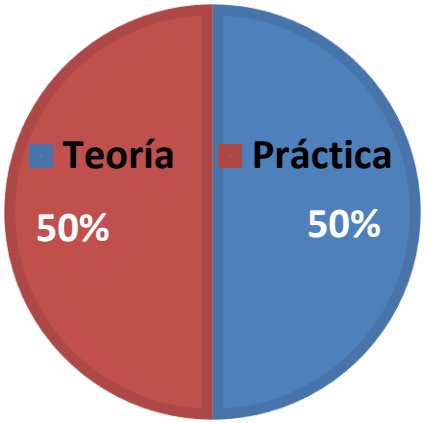
# CRONOGRAMA TENTATIVO – 2da parte

| Clase | Fecha | Temas/Bibliografía/Actividades/Recursos/Evaluaciones |
|---|---|---|
| 10 | 17-may | NGS: Tecnologías de secuenciación. Datos de secuenciación. Tipos de formato de datos y control de Calidad de los datos. |
| 11 | 24-may | Ensamblado de Genomas y métodos de estudio de secuencias genómicas y transcriptómicas de NGS. Algoritmos de ensamblado y de mapeo de secuencias de NGS en bioinformática. TP. 8: Análisis de calidad en secuencias NGS. |
| 12 | 31-may | Identificación de Genes Bacterianos y Eucariotas: Anotación estructural. Algoritmos y métodos de anotación.TP. 8: Parte III – Ensamblado de genomas y calidad del ensamblado. |
| 13 | 07-jun | Anotación Funcional de Genes: Algoritmos y métodos de anotación. Interproscan y Blast2GO.” TP. 9: Anotación estructural de genes (getOrf). |
| 14 | 14-jun | Genómica comparativa: Tipos y métodos de estudio en Genómica Comparativa. Búsqueda de variantes genéticas (SNPs / INDels), TP.10 Realizar la identificación de variantes alélicas en datos de NGS |
| 15 | 21-jun | Puesta en común de TPs |
| 16 | 28-jun | EXAMEN PARCIAL II |
| 17 | 05-jul | Recuperatorio - Final Adelantado |
| EXAMEN FINAL | 19-jul | Final Regular |

# Organización de la materia

## CONTENIDO CLASES TEORICO-PRÁCTICAS






### Parte I

- Informe TP final I: Entrega por grupo, hasta 2 alumnos.
- Parcial Teórico-práctico
  - Parte teórica (Escrito)
  - parte práctica (Ejercicios en computadora)

### Parte II

- Informe TP final II: Entrega por grupo, hasta 2 alumnos.
- Parcial Teórico-práctico
  - Parte teórica (Escrito)
  - parte práctica (Ejercicios en computadora)

**Para aprobar el Final se debe aprobar la parte teórica y la parte práctica (>=4)**

**Recuperatorio de parcial I o II: Se recupera uno solo de los parciales**

- Misma modalidad que los parciales

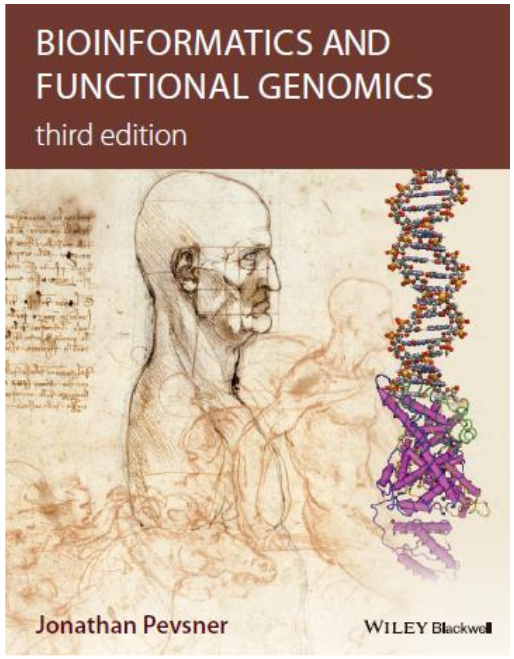
**Final anticipado Teórico y práctico**

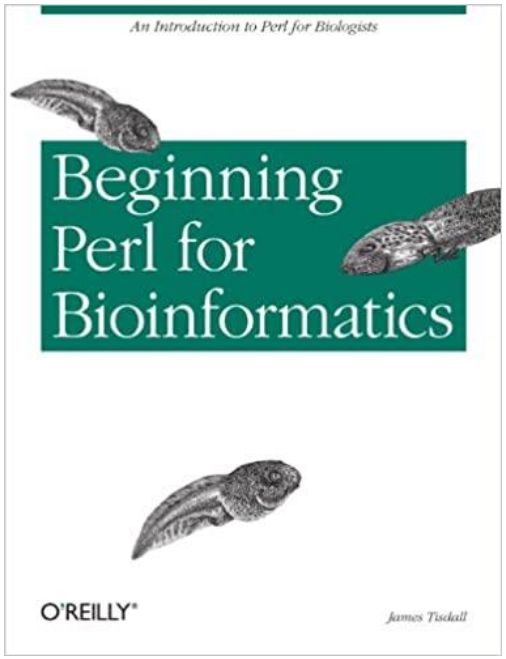
- Parte teórica (Escrito)
- Parte práctica (Escrito – Interpretación de resultados)
- Para acceder a los finales deben estar aprobados los TPs y los parciales I y II

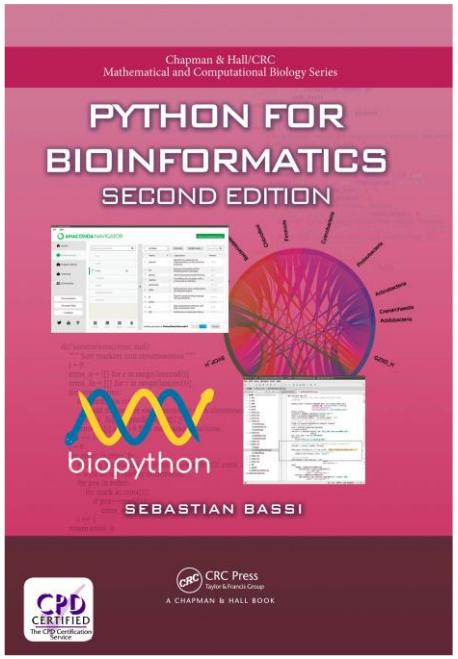
**Para aprobar el Final se debe aprobar la parte teórica y la parte práctica (>=4)**

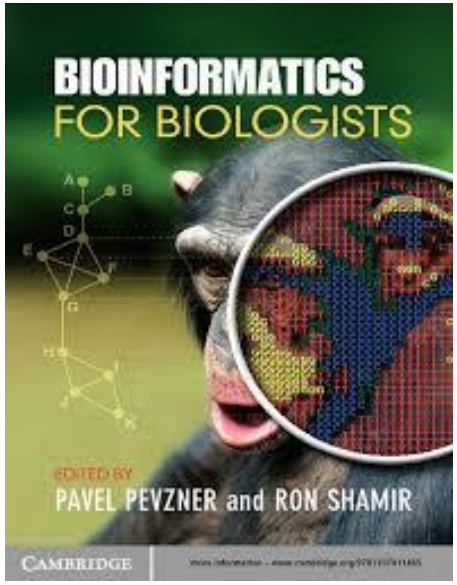
**Para aprobar la materia de deben tener aprobados los 2 parciales, los 2 informes TP y el Examen Final**

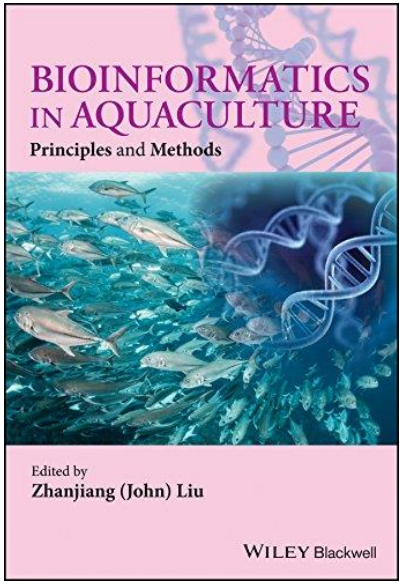
# Lecturas recomendadas



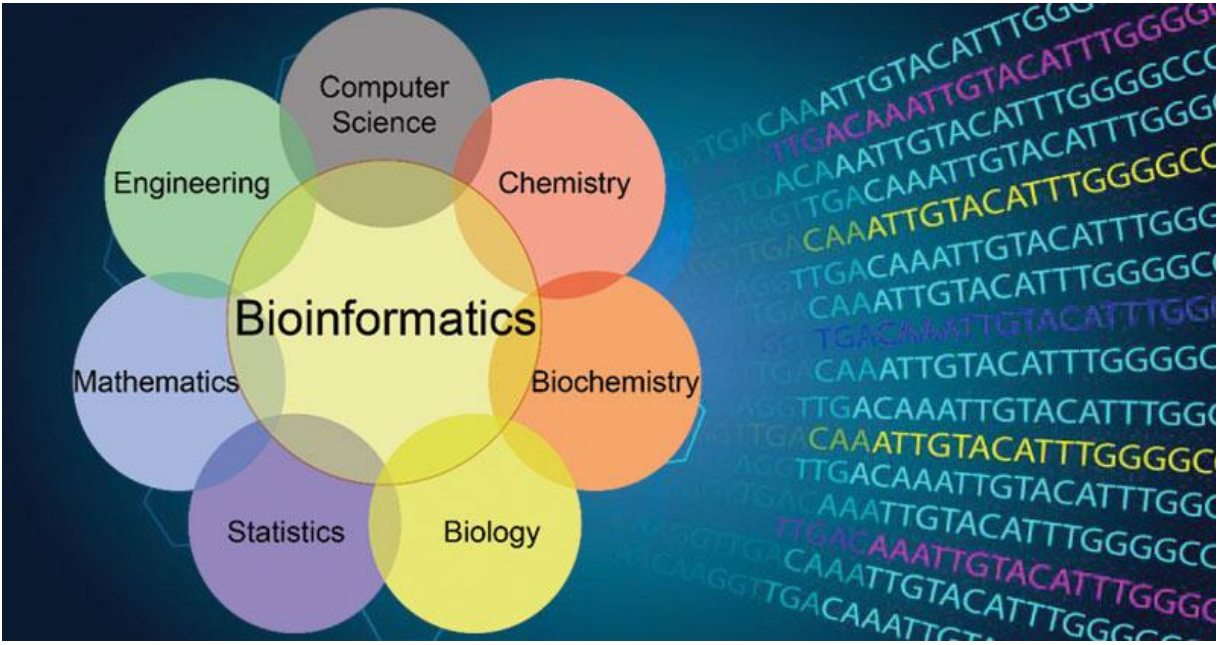
# Objetivo de la materia

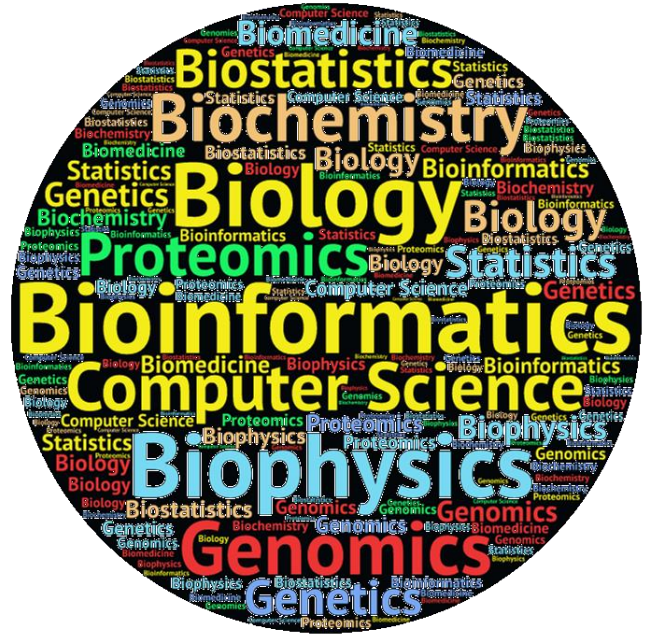
1. **Familiarizar a los estudiantes en temas de la bioinformática (con sus herramientas, algoritmos y fundamentos)**
2. **Orientar en los usos y aplicaciones de la Bioinformática**
3. **Introducir al sistema operativo LINUX**
4. **Aprender a trabajar por Línea de comandos**
5. **Aprender a utilizar herramientas Bioinformáticas**

# ¿Que es la Bioinformática?

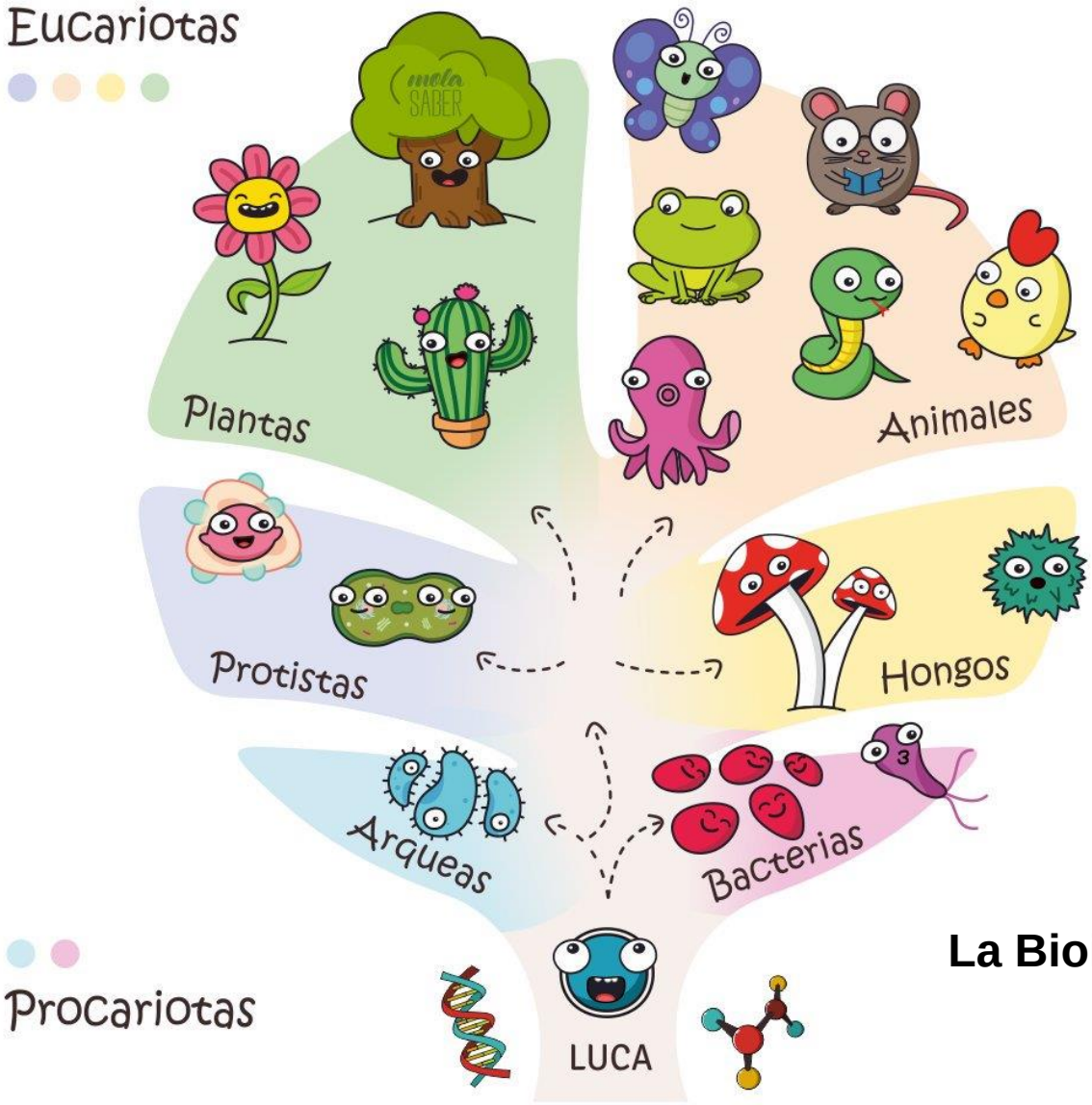
La **bioinformática** es el desarrollo, diseño y la aplicación de tecnologías computacionales a la gestión, administración y análisis de datos biológicos.

- Biología molecular computacional
- Biología de sistemas computacionales
- Biología Computacional

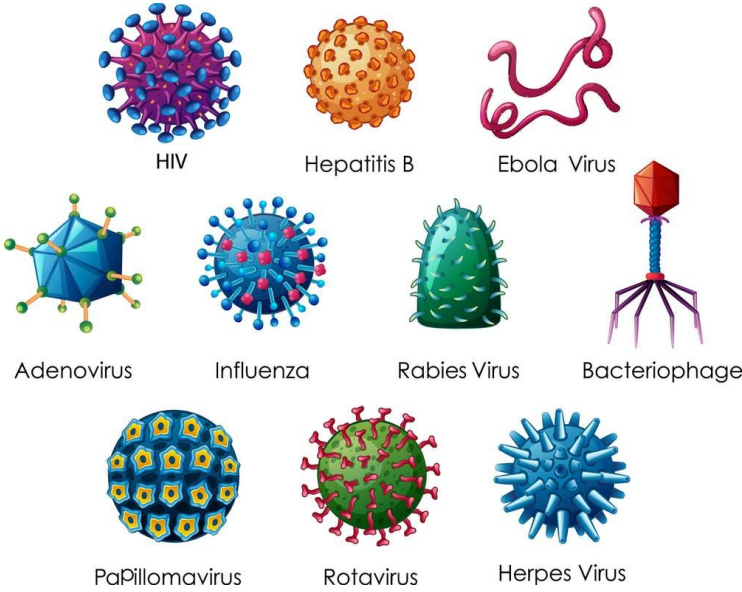


# Reinos de la vida



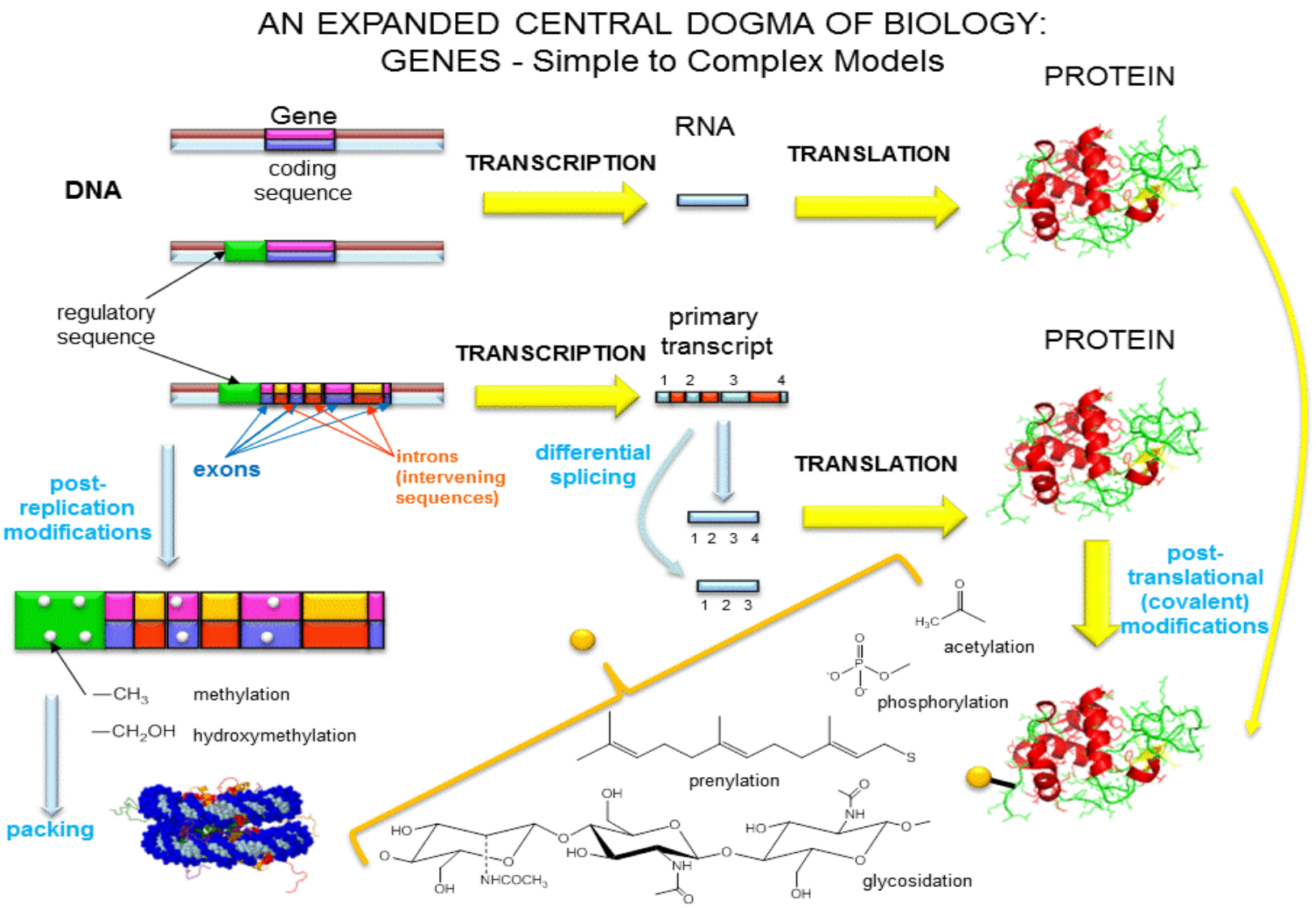

## Y los “no vivos”




**Virus**

**La Bioinformática estudia a todos ellos**

Si hay algo que aprendimos
en estos ultimos diez años
es que tan complejo se volvio todo....



# Biology timeline

**1953**

- Watson & Crick proposed the double helix model for DNA based x-ray data obtained by Franklin & Wilkins.

**1955**

- The sequence of the first protein to be analysed, bovine insulin, is announed by F.Sanger.

**1962**

- Margaret Dayhoff (remember that name!) is renowned as the first ever bioinformatician. Recognizing the potential for computational methods to solve biomedical problems, in 1962 she released the first de novo sequence assembler in collaboration with Robert Ledley6. The assembler looked at primary protein structure and was written in the programming language Fortran on punch-cards.

**1970**

- The details of the Needleman-Wunsch algorithm for sequence comparison are published.

**1972**

- The first recombinant DNA molecule is created by Paul Berg and his group.

**1975**

- Two-dimensional electrophoresis, where separation of proteins on SDS polyacrylamide gel is combined with separation according to isoelectric points, is announced by P.H.O'Farrel.

**1977**

- The phi X 174 (or ΦX174) bacteriophage was the first genome to be sequenced by F. Sanger.

**1977**

- The full description of the Brookhaven PDB (http://www.pdb.bnl.gov) is published
- Allan Maxam and Walter Gilbert (Harvard) and Frederick Sanger (U.K. Medical Research Council), report methods for sequencing DNA.

**1980**

- The first complete gene sequence for an organism (FX174) is published. The gene consists of 5,386 base pairs which code nine proteins

# Informatics timeline

**1969**

- The ARPANET is created by linking computers at Standford and UCLA.

**1973**

- The Brookhaven Protein DataBank is announeced (Acta.Cryst.B,1973,29: 1764). Robert Metcalfe receives his Ph.D from Harvard University. His thesis describes Ethernet.

**1974**

- Vint Cerf and Robert Khan develop the concept of connecting networks of computers into an "internet" and develop the Transmission Control Protocol (TCP).

**1975**

- Microsoft Corporation is founded by Bill Gates and Paul Allen

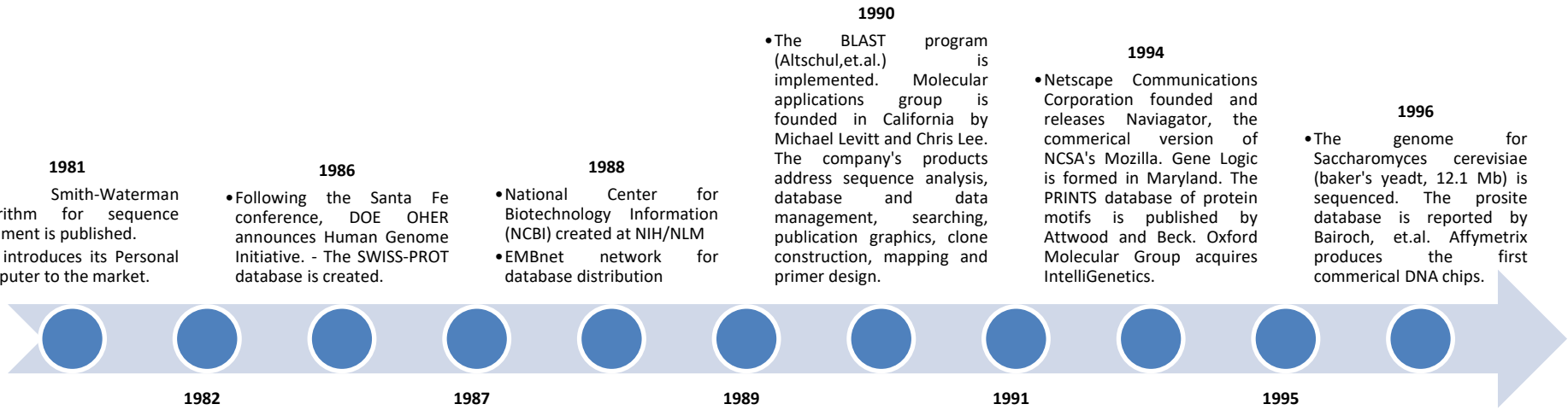
**1976**

- The Unix-To-Unix Copy Protocol (UUCP) is developed at Bell Labs.

**1978**

- The first Usenet connection is established between Duke and the University of North Carolina at Chapel Hill by Tom Truscott, Jim Ellis and Steve Bellovin.

# Bioinformatics timeline



**1981**

- The Smith-Waterman algorithm for sequence alignment is published.
- IBM introduces its Personal Computer to the market.

**1982**

- Genetics Computer Group (GCG) created as a part of the University of Wisconsin of Wisconsin Biotechnology Center. The company's primary product is The Wisconsin Suite of molecular biology tools.
- GenBank Release 3 made public
- Phage lambda genome sequenced

**1986**

- Following the Santa Fe conference, DOE OHER announces Human Genome Initiative. - The SWISS-PROT database is created.

**1987**

- Perl (Practical Extraction Report Language) is released by Larry Wall

**1988**

- National Center for Biotechnology Information (NCBI) created at NIH/NLM
- EMBnet network for database distribution

**1989**

- The genetics Computer Group (GCG) becomes a privatae company. Primary products: Anaconds, Asp, Cameleon and others (molecular modeling, drug design, protein design).

**1990**

- The BLAST program (Altschul,et.al.) is implemented. Molecular applications group is founded in California by Michael Levitt and Chris Lee. The company's products address sequence analysis, database and data management, searching, publication graphics, clone construction, mapping and primer design.

**1991**

- The research institute in Geneva (CERN) announces the creation of the protocols which make -up the World Wide Web.

**1994**

- Netscape Communications Corporation founded and releases Naviagator, the commerical version of NCSA's Mozilla. Gene Logic is formed in Maryland. The PRINTS database of protein motifs is published by Attwood and Beck. Oxford Molecular Group acquires IntelliGenetics.

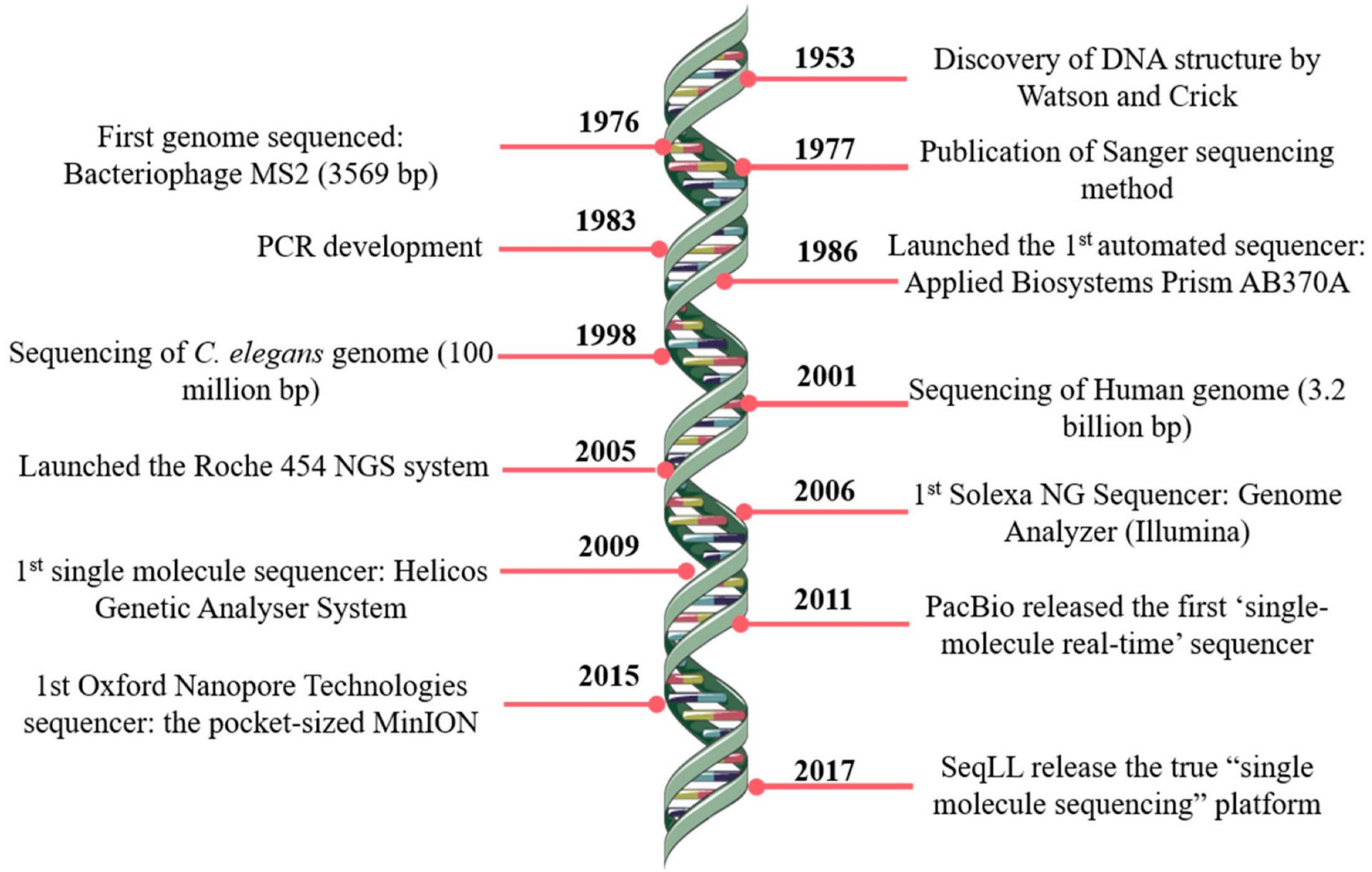
**1995**

- The Haemophilus influenzea genome (1.8) is sequenced. The Mycoplasma genitalium genome is sequenced.

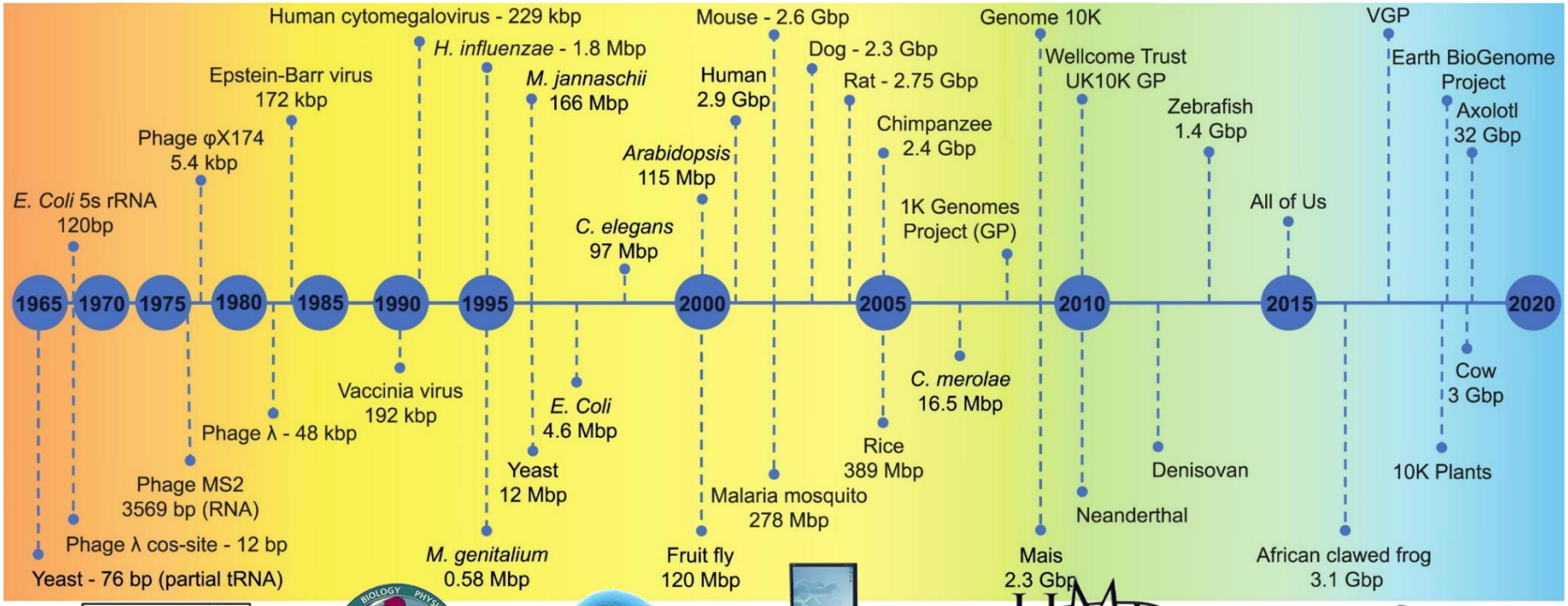
**1996**

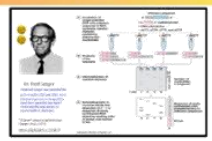
- The genome for Saccharomyces cerevisiae (baker's yeadt, 12.1 Mb) is sequenced. The prosite database is reported by Bairoch, et.al. Affymetrix produces the first commerical DNA chips.

# Milestones in History of Bioinformatics

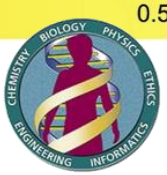


# A Chronological History of Genome sequencing

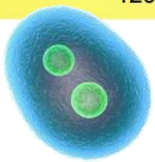


Sanger method




Human Genome Project



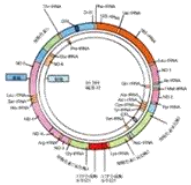
Complete eukaryotic genome

Second generation sequencer：454 GS20


Research Human Microbiome Project

Nanospace sequencing


Human mitochondrial genome sequence



Complete cell genome

Complete the Human Genome Project

Second generation sequencer：Genetic Analyzer 2

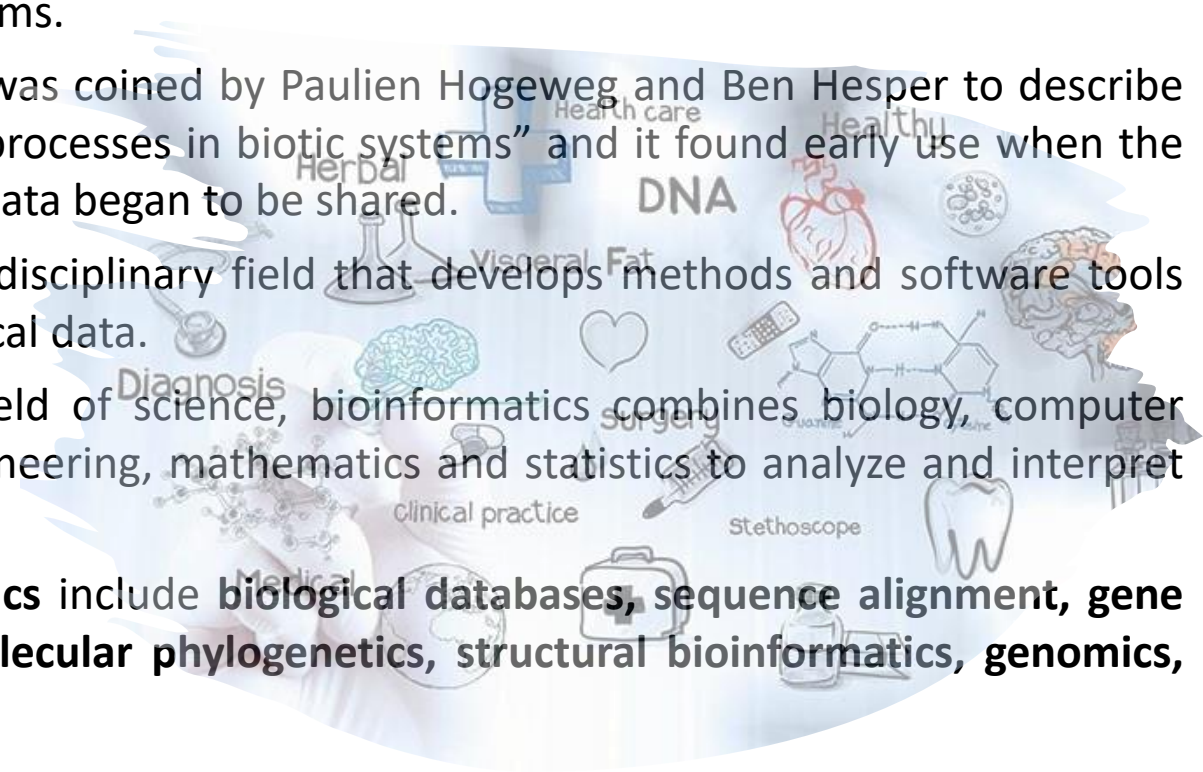
Third generation sequencer：PacBio RS

The third stage Human microbiome project

https://doi.org/10.1016/j.csbj.2019.11.002

## Applications of Bioinformatics

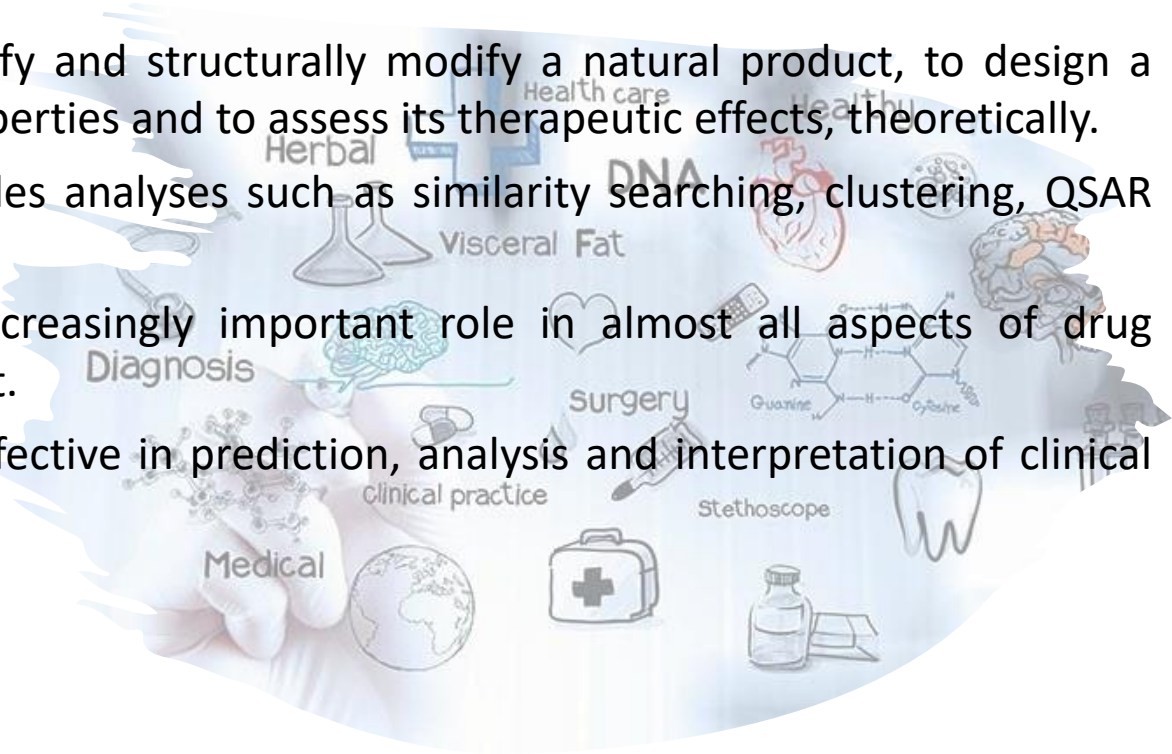
- **With a large number of prokaryotic and eukaryotic genomes completely sequenced** and more forthcoming, access to the genomic information and synthesizing it for the **discovery of new knowledge have become central themes of modern biological research**.
- **Mining the genomic information requires the use of sophisticated computational tools.**
  - the careful storage, organization and indexing of information in order to tackle the new challenges in the genomic era.
- **Information science has been applied to biology to produce a field is called bioinformatics.**
  - It is concerned with the state of- the-art computational tools available to solve biological research problems.
  - The term bioinformatics was coined by Paulien Hogeweg and Ben Hesper to describe "the study of informatic processes in biotic systems" and it found early use when the first biological sequence data began to be shared.
  - Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data.
  - As an interdisciplinary field of science, bioinformatics combines biology, computer science, information engineering, mathematics and statistics to analyze and interpret biological data.
- The **key areas of bioinformatics** include **biological databases, sequence alignment, gene and promoter prediction, molecular phylogenetics, structural bioinformatics, genomics, and proteomics.**



# Applications of Bioinformatics

• Bioinformatics has become essential for **basic genomic and molecular biology research, biotechnology and biomedical sciences**. The main uses of bioinformatics include:

- Bioinformatics plays a vital role in the areas of structural genomics, functional genomics, and nutritional genomics.
- It covers emerging scientific research and the exploration of proteomes from the overall level of intracellular protein composition (protein profiles), protein structure, protein-protein interaction, and unique activity patterns (e.g. post-translational modifications).
- Bioinformatics is used for transcriptome analysis where mRNA expression levels can be determined.
- Bioinformatics is used to identify and structurally modify a natural product, to design a compound with the desired properties and to assess its therapeutic effects, theoretically.
- Cheminformatics analysis includes analyses such as similarity searching, clustering, QSAR modeling, virtual screening, etc.
- Bioinformatics is playing an increasingly important role in almost all aspects of drug discovery and drug development.
- Bioinformatics tools are very effective in prediction, analysis and interpretation of clinical and preclinical findings.



## Molecular medicine

- The human genome will have profound effects on the fields of biomedical research and clinical medicine.
- The completion of the human genome and the use of bioinformatic tools means that we can search for the genes directly associated with different diseases and begin to understand the molecular basis of these diseases more clearly.
- This new knowledge of the molecular mechanisms of disease will enable better treatments, cures and even preventative tests to be developed.

## Personalised medicine

- Clinical medicine will become more personalised with the development of the field of pharmacogenomics.
- This is the study of how an individual's genetic inheritence affects the body's response to drugs.
- Today, doctors have to use trial and error to find the best drug to treat a particular patient as those with the same clinical symptoms can show a wide range of responses to the same treatment.
- In the future, doctors will be able to analyse a patient's genetic profile and prescribe the best available drug therapy and dosage from the beginning.

## Preventative medicine

- With the specific details of the genetic mechanisms of diseases being unravelled, the development of diagnostic tests to measure a persons susceptibility to different diseases may become a distinct reality.

## Gene therapy

- In the not too distant future with the use of bioinformatics tool, the potential for using genes themselves to treat disease may become a reality.
- Gene therapy is the approach used to treat, cure or even prevent disease by changing the expression of a person's genes.

## Drug development

- At present all drugs on the market target only about 500 proteins.
- With an improved understanding of disease mechanisms and using computational tools to identify and validate new drug targets, more specific medicines that act on the cause, not merely the symptoms, of the disease can be developed.
- These highly specific drugs promise to have fewer side effects than many of today's medicines.

## Antibiotic resistance

- Scientists have been examining the genome of *Enterococcus faecalis*-a leading cause of bacterial infection among hospital patients.
- They have discovered a virulence region made up of a number of antibiotic-resistant genes that may contribute to the bacterium's transformation from a harmless gut bacteria to a menacing invader.
- The discovery of the region, known as a pathogenicity island, could provide useful markers for detecting pathogenic strains and help to establish controls to prevent the spread of infection in wards.

## Biotechnology

- The archaeon *Archaeoglobus fulgidus* and the bacterium *Thermotoga maritima* have potential for practical applications in industry and government-funded environmental remediation.
- These microorganisms thrive in water temperatures above the boiling point and therefore may provide the DOE, the Department of Defence, and private companies with heat-stable enzymes suitable for use in industrial processes
- Other industrially useful microbes include, *Corynebacterium glutamicum* which is of high industrial interest as a research object because it is used by the chemical industry for the biotechnological production of the amino acid lysine.
- The substance is employed as a source of protein in animal nutrition.
- Biotechnologically produced lysine is added to feed concentrates as a source of protein, and is an alternative to soybeans or meat and bonemeal.
- *Lactococcus lactis* is one of the most important micro-organisms involved in the dairy industry.
- Researchers anticipate that understanding the physiology and genetic make-up of this bacterium will prove invaluable for food manufacturers as well as the pharmaceutical industry, which is exploring the capacity of *lactis* to serve as a vehicle for delivering drugs.

## The reality of bioweapon creation

- Scientists have recently built the virus poliomyelitis using entirely artificial means.
- They did this using genomic data available on the Internet and materials from a mail-order chemical supply.
- The research was financed by the US Department of Defence as part of a biowarfare response program to prove to the world the reality of bioweapons.
- The researchers also hope their work will discourage officials from ever relaxing programs of immunisation.
- This project has been met with very mixed feelings.

## Crop improvement

- Comparative genetics of the plant genomes has shown that the organisation of their genes has remained more conserved over evolutionary time than was previously believed.
- These findings suggest that information obtained from the model crop systems can be used to suggest improvements to other food crops.
- At present the complete genomes of Arabidopsis thaliana (water cress) and Oryza sativa (rice) are available.

## Insect resistance

- Genes from *Bacillus thuringiensis* that can control a number of serious pests have been successfully transferred to cotton, maize and potatoes.
- This new ability of the plants to resist insect attack means that the amount of insecticides being used can be reduced and hence the nutritional quality of the crops is increased.

## Microbial genome applications

- The arrival of the complete genome sequences and their potential to provide a greater insight into the microbial world and its capacities could have broad and far reaching implications for environment, health, energy and industrial applications.
- For these reasons, in 1994, the US Department of Energy (DOE) initiated the MGP (Microbial Genome Project) to sequence genomes of bacteria useful in energy production, environmental cleanup, industrial processing and toxic waste reduction.
- By studying the genetic material of these organisms, scientists can begin to understand these microbes at a very fundamental level and isolate the genes that give them their unique abilities to survive under extreme conditions

## Waste cleanup

- *Deinococcus radiodurans* is known as the world's toughest bacteria and it is the most radiation resistant organism known.
- Scientists are interested in this organism because of its potential usefulness in cleaning up waste sites that contain radiation and toxic chemicals.

## Development of Drought resistance varieties

- Progress has been made in developing cereal varieties that have a greater tolerance for soil alkalinity, free aluminium and iron toxicities.
- These varieties will allow agriculture to succeed in poorer soil areas, thus adding more land to the global production base.
- Research is also in progress to produce crop varieties capable of tolerating reduced water conditions.

## Climate change Studies

- Increasing levels of carbon dioxide emission, mainly through the expanding use of fossil fuels for energy, are thought to contribute to global climate change.
- Recently, the DOE (Department of Energy, USA) launched a program to decrease atmospheric carbon dioxide levels.
- One method of doing so is to study the genomes of microbes that use carbon dioxide as their sole carbon source.

## Alternative energy sources

- Scientists are studying the genome of the microbe *Chlorobium tepidum* which has an unusual capacity for generating energy from light

## Evolutionary studies

- The sequencing of genomes from all three domains of life, eukaryota, bacteria and archaea means that evolutionary studies can be performed in a quest to determine the tree of life and the last universal common ancestor.

## Comparative Studies

- Analysing and comparing the genetic material of different species is an important method for studying the functions of genes, the mechanisms of inherited diseases and species evolution.
- Bioinformatics tools can be used to make comparisons between the numbers, locations and biochemical functions of genes in different organisms.

## Veterinary Science

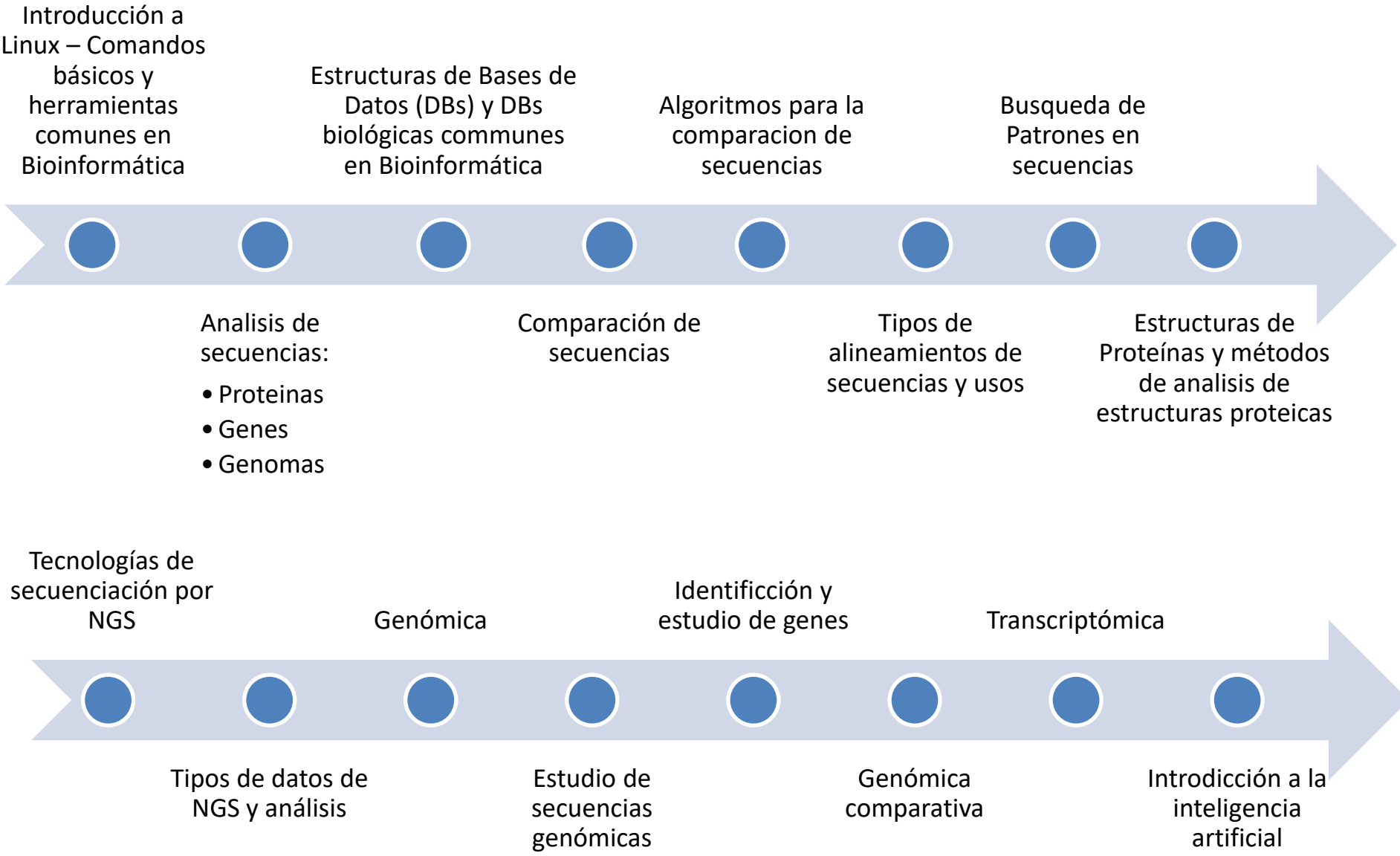
- Sequencing projects of many farm animals including cows, pigs and sheep are now well under way in the hope that a better understanding of the biology of these organisms will have huge impacts for improving the production and health of livestock and ultimately have benefits for human nutrition.

## Biological diversity

- The study of different organism and sampling of the ecosystem through the application of new technologies

https://microbenotes.com/bioinformatics-introduction-and-applications/
https://www.biologydiscussion.com/biodiversity/bioinformatics/bioinformatics-history-coverage-components-and-applications/11294

# Contenido de la materia - Clases



# Introducción a la Bioinformática en Linux

# Hardware: Componentes físicos

## Generalidades. Definiciones básicas. Hardware

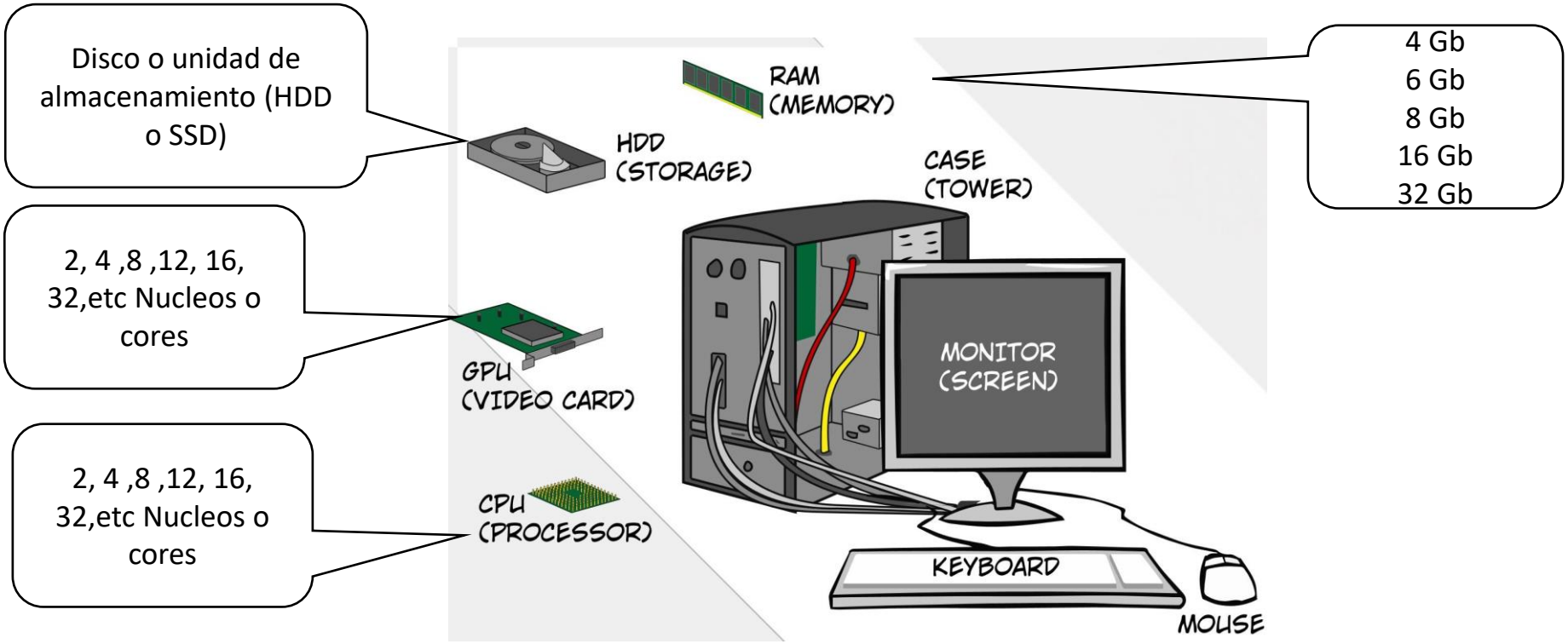
El significado del término **hardware** no es fácil de expresar en español con una sola palabra; literalmente se debe entender como “conjunto de útiles duros"; en el contexto que nos ocupa, el hardware de un computador

**Hardware** ***es el conjunto de dispositivos físicos que lo componen***,

**Software**, **designa los programas que puede ejecutar el computador**.

En cierto modo, el **hardware** es comparable al cerebro o, más generalmente, al cuerpo físico del computador mientras que el **software** será lo equivalente a las ideas que pueblan el cerebro.

# Componentes de una computadora




La **memoria de acceso aleatorio** (*Random Access Memory*, **RAM**) se utiliza como memoria de trabajo de computadoras y otros dispositivos para el sistema operativo, los programas y la mayor parte del software. En la RAM se cargan todas las instrucciones que ejecuta la unidad central de procesamiento (procesador) y otras unidades del computador, además de contener los datos que manipulan los distintos programas.

La **unidad central de procesamiento** o **unidad de procesamiento central** (conocida por las siglas **CPU**, del inglés: C*entral Processing Unit*), es el hardware dentro de un ordenador u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

La ***unidad de almacenamiento*** es el **lugar** donde se guarda la información de manera permanente

# What is the Memory Capacity of a Human Brain?

- The human brain's memory capacity in the average adult can store **trillions of bytes** of information.

- In a Stanford Study, it was reported that the cerebral cortex alone has **125 trillion synapses.**
- In another study, it was reported that **1 synapse can store 4.7 bits of information**. Neurons are the cells which processes and transmits messages within the brain, and synapses are the bridges between neurons which carry the transmitted messages.

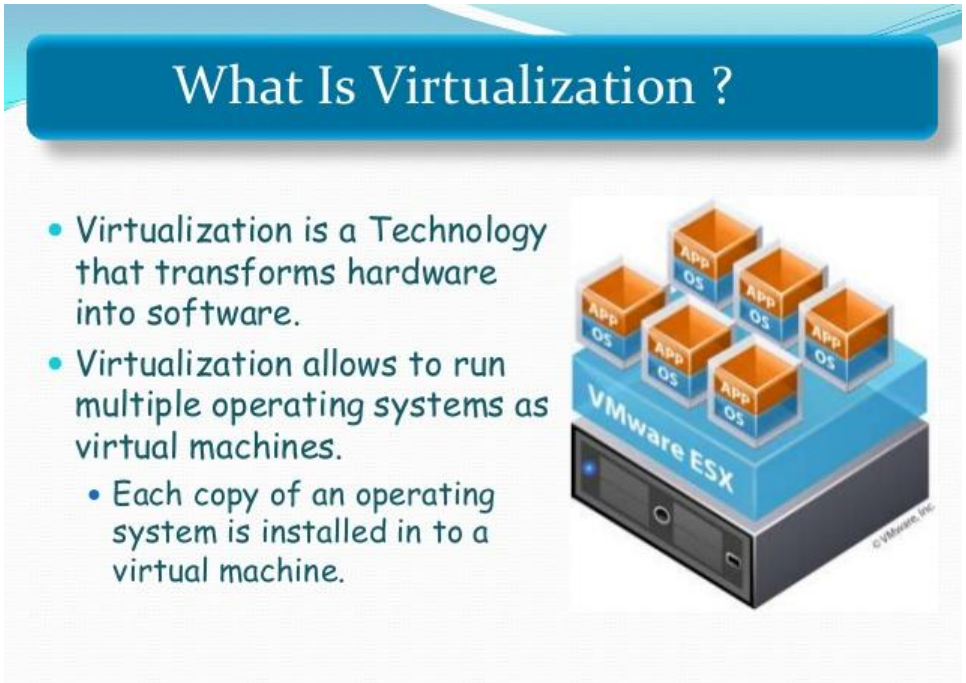
- Running the numbers – **125 trillion synapses – 4.7 bits/synapse, and about 1 trillion bytes equaling 1 TB (Terabyte)**.

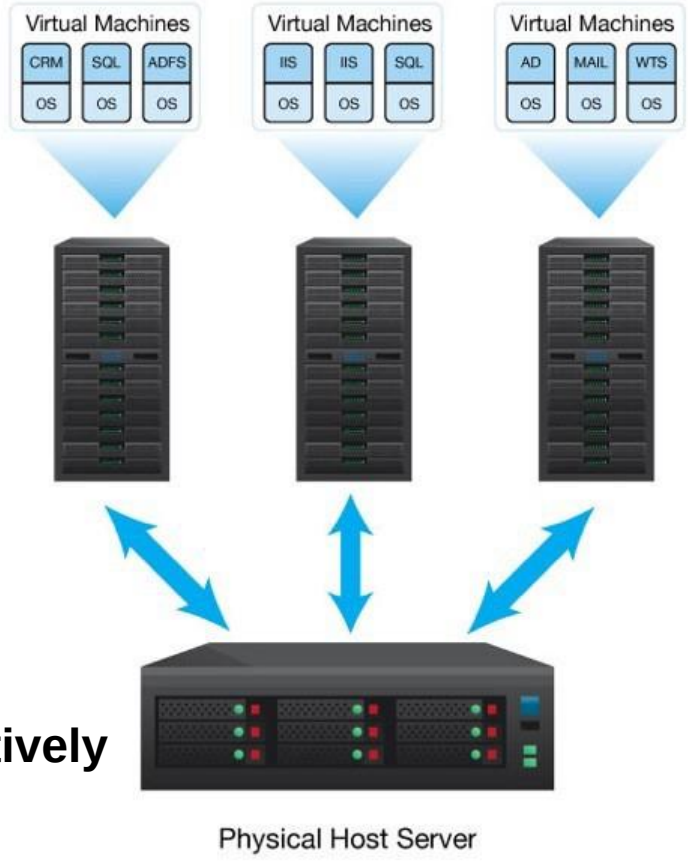
- This storage capacity is an amount over **74 Terabytes** (just in the cerebral cortex alone)
- according to a 2010 article in Scientific American, the memory capacity of the human brain was reported to have the equivalent of **2.5 petabytes of memory capacity.**

As a number, a "petabyte" means 1024 terabytes or a million gigabytes, so the average adult human brain has the ability to store the equivalent of 2.5 million gigabytes digital memory.

# Virtualización y máquinas virtuales

**Virtualization** refers to the practice of breaking down the physical infrastructure of computing and networking resources into smaller, reusable and more flexible units.




- **Method which aims to use the resources effectively**
- **Isolation of computing resources from users.**

# Virtualización y máquinas virtuales

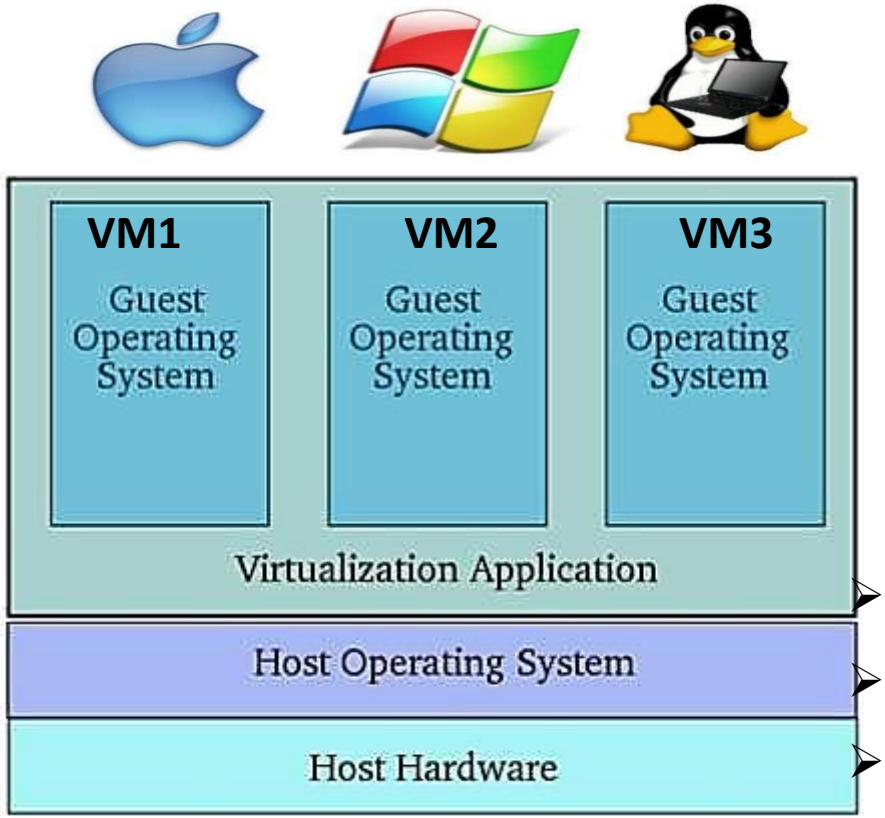
- These virtual machines share the physical resource between each other.

- Each virtual machine has own operating system.

- Hardware sharing is set by an interface (**Hardware Emulation**):
  - Full emulation of complete hardware
  - Supports different hardware components on a
  - different hardware infrastructure
  - Enables each virtual machine to work with its own processor, RAM-everything physical computer contains.

# OS Virtualization

**Guest OS operates in virtual machines within the virtualization application which runs on top of the host OS in the same way as any other application.**




- **Por ejemplo VirtualBox**
- **Por ejemplo Windows**
- **Todas las características que tienen sus computadoras**

# Virtualizadores

## VirtualBox

### Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.


## VMware Workstation Player

VMware Workstation Player is an ideal utility for running a single virtual machine on a Windows or Linux PC. Organizations use Workstation Player to deliver managed corporate desktops, while students and educators use it for learning and training.

The free version is available for non-commercial, personal and home use. We also encourage students and non-profit organizations to benefit from this offering.

Commercial organizations require commercial licenses to use Workstation Player.

Need a more advanced virtualization solution? Check out Workstation Pro.

Try Workstation 15.5 Player for Windows

Try Workstation 15.5 Player for Linux

Cookie Settings

# ¿Como se utiliza la virtualización en bioinformática?

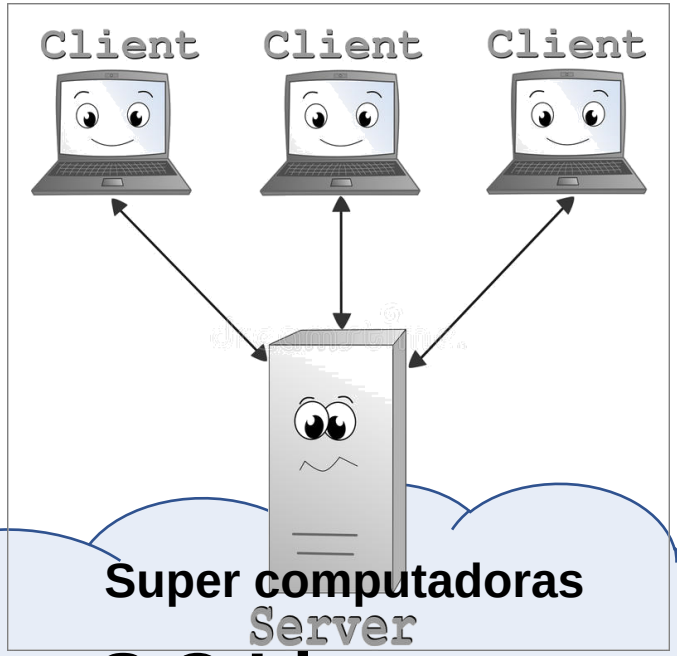
**Own computer**

**S.O Linux**

**No virtualizado:** Utiliza Linux como sistema operativo principal

**Virtualizado**: Utiliza Linux como guest en otro sistema operativo (Limitacion de recursos)

**Remote conection**




**Super computadoras**

**S.O Linux**

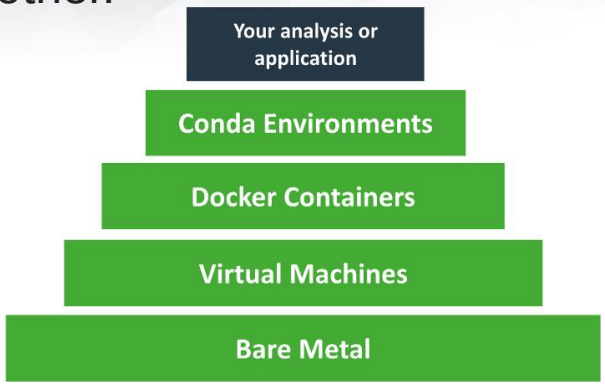
**No virtualizado:** Utiliza Linux como sistema operativo principal

**Virtualizado**: Utiliza Linux como guest en otro sistema operativo.

# Enviroments and containers

The term **environment** refers to the state of a **computer**, determined by a combination of software, basic hardware, and which programs are running.

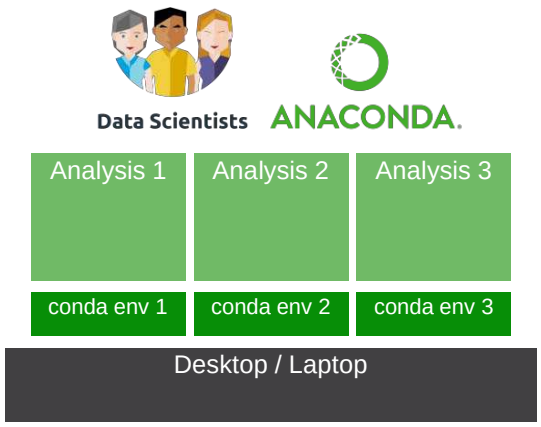
A **container** is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.



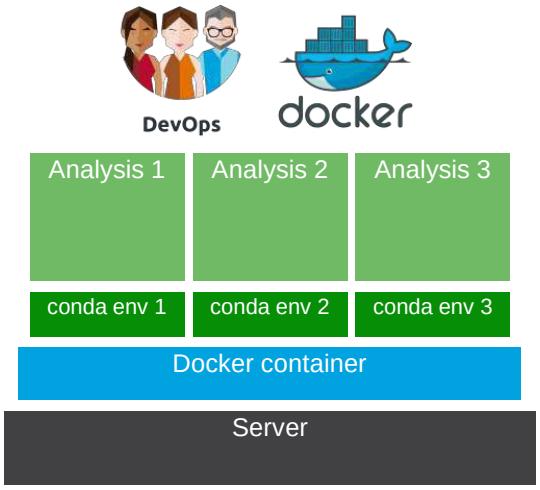

ANACONDA
Powered by Continuum Analytics

docker

VAGRANT

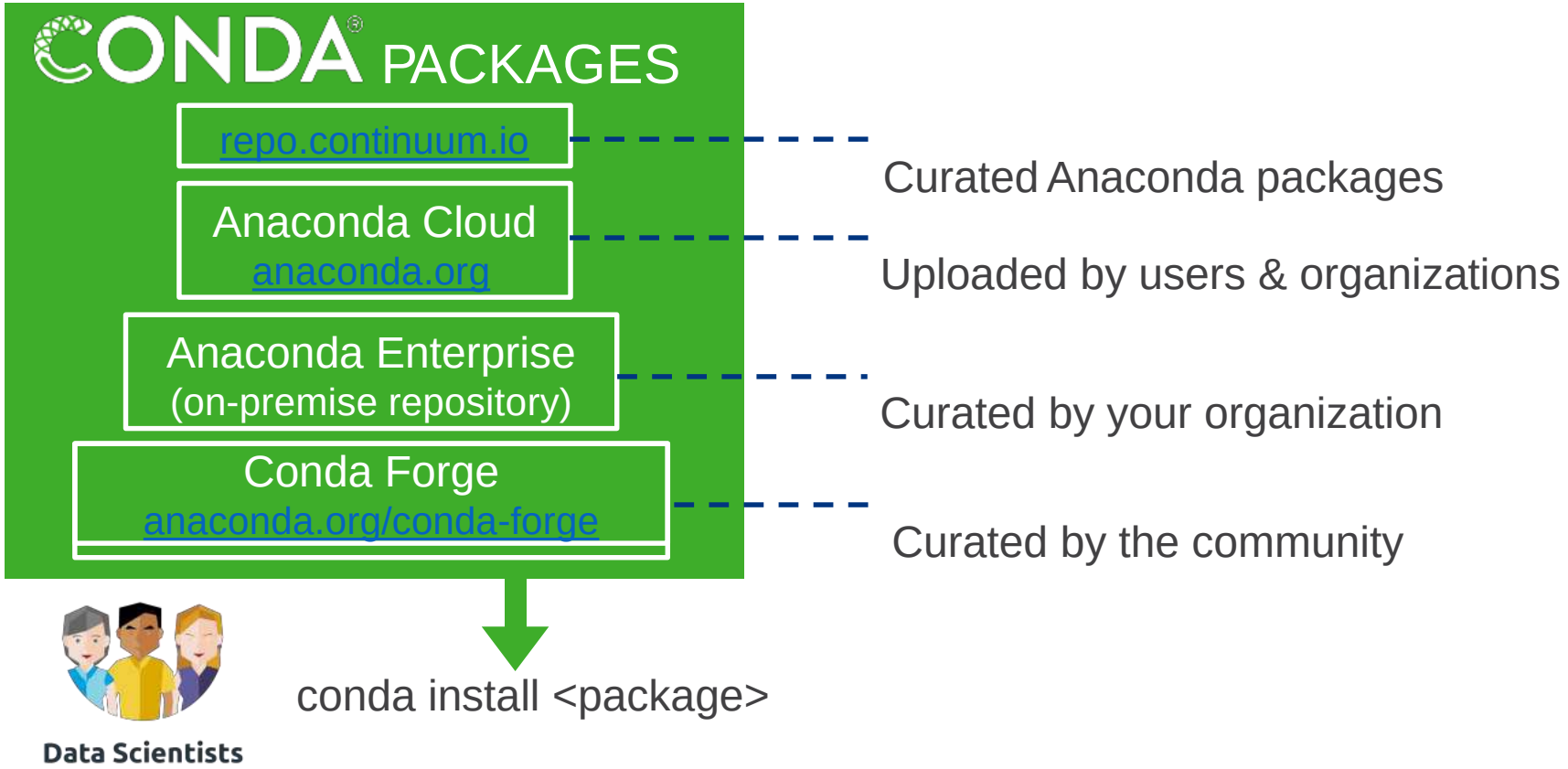



**Data Science Development**




**Data Science Deployment**

## Creating & Curating conda Packages for Data Scientists



# Enforcer of Safety and Correctness

- **pre-compiled packages only**
  will never require a compiler in production, or unexpectedly invoke one on you

- **environment integrity**
  disk-mutating operations are wrapped in a transaction, and rolled back in the event of errors

- **environment correctness**
  conda enforces compatibility of packages within environments

# Channels for User Empowerment

- **the channel is a component of a package's identity**
  first-class citizen in package specifications
- **easy package building**
  with conda-build, a dedicated tool with engaged and dedicated code contributors
- **channels enable community**
  independent contributors, independent packaging communities, and corporations

**DOWNLOAD** Anaconda or miniconda
continuum.io/downloads

# Conda is a Community

CONDA-FORGE



OMNIA







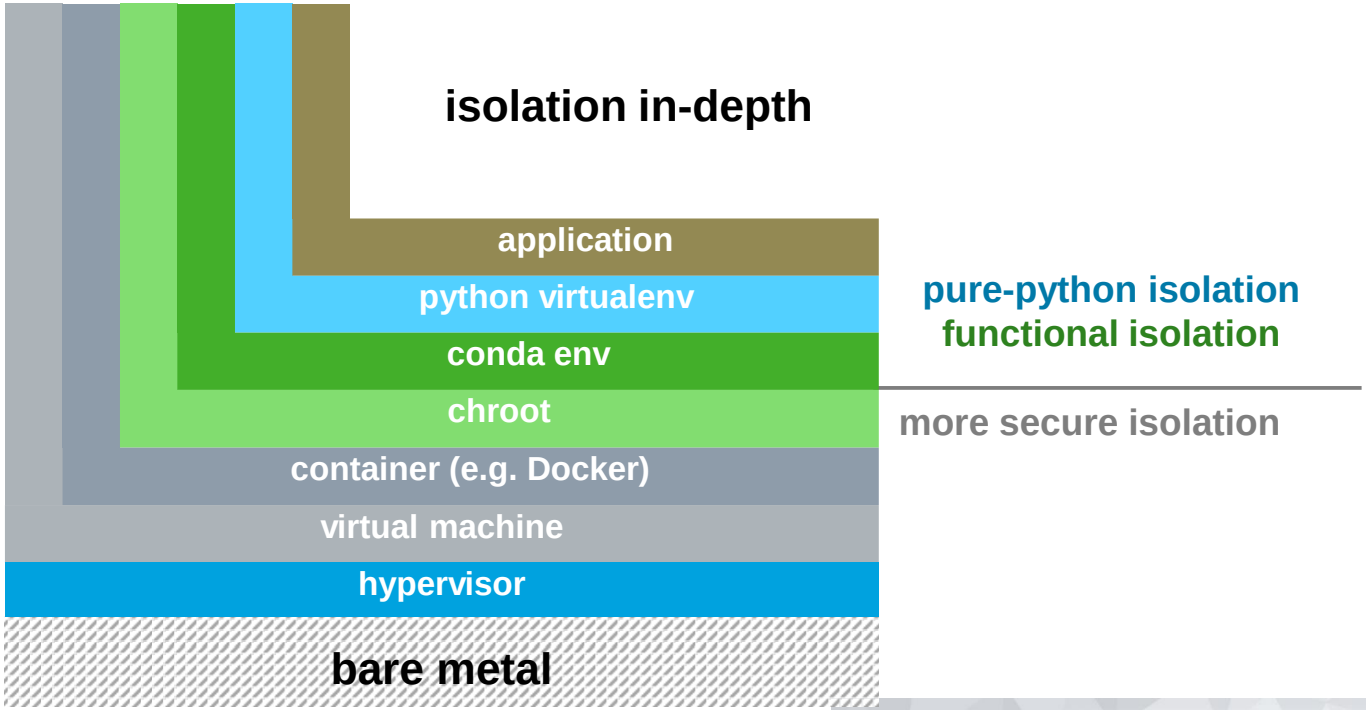


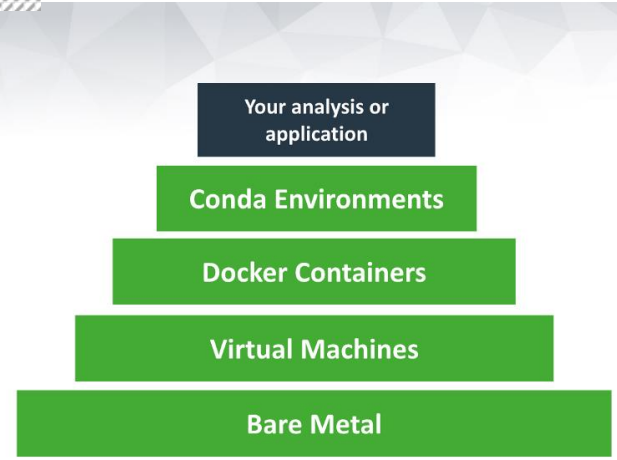




# Resumamos por capas

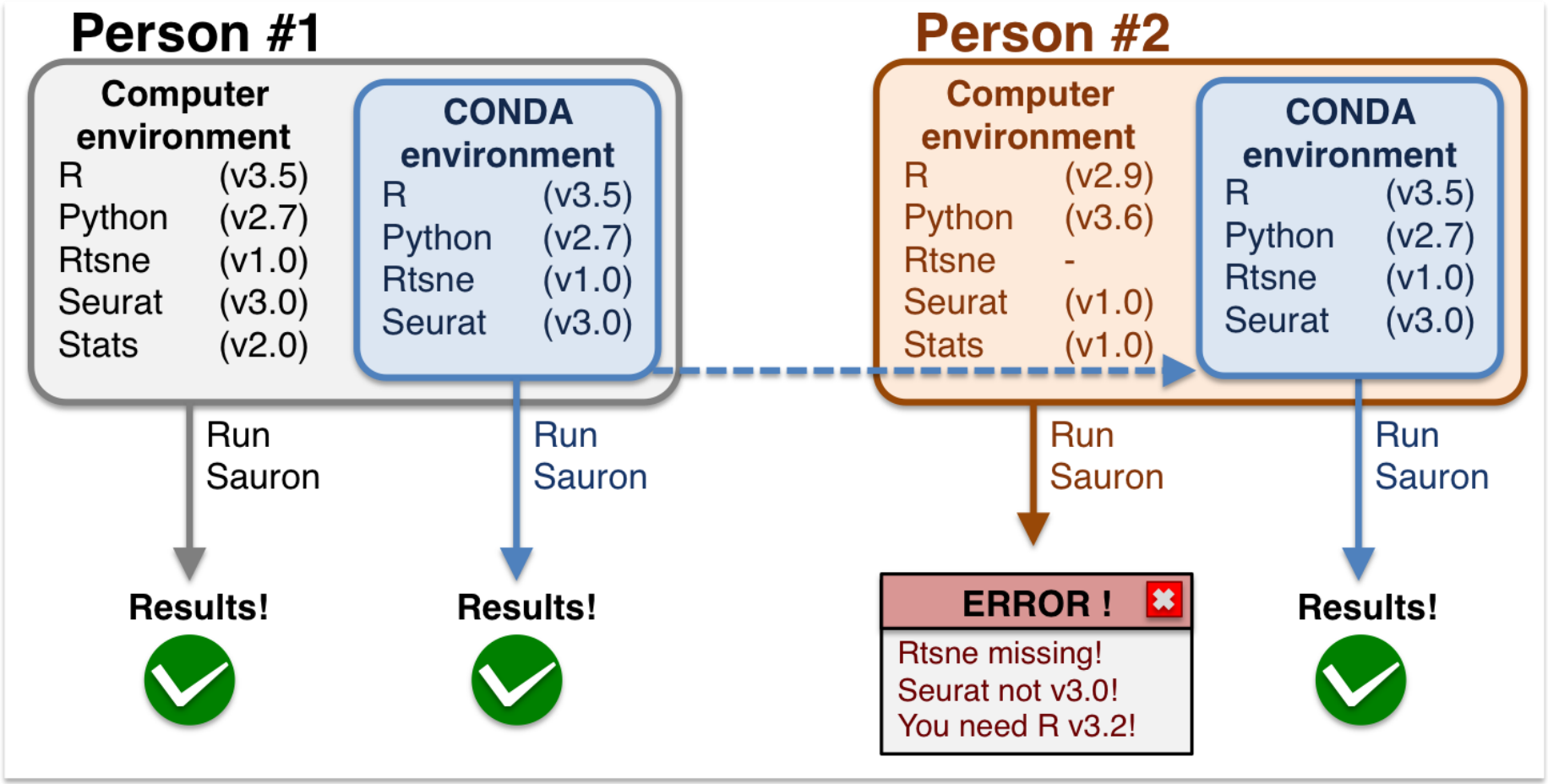
## Layers of Process Isolation



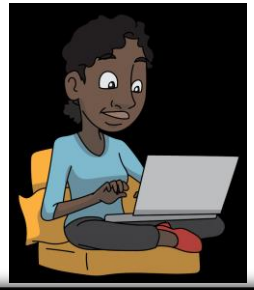
# ¿Por que usamos un enviroment?

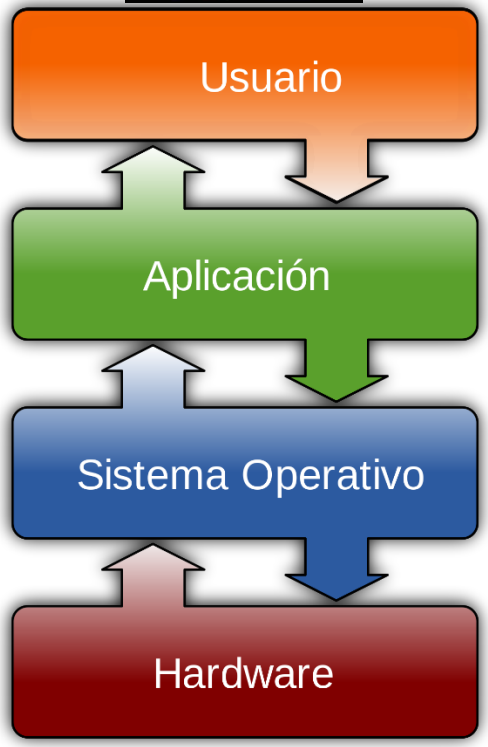
El sistema de ***enviroments*** trae equilibrio al conjunto de programas para que no haya diferencias o errores en las dependencias de modo que se pueda llegar a un mismo resultado usando los mismos paquetes, las mismas librerías y las mismas versiones de softwares **haciéndolo independiente del hardware y del sistema operativo garantizando la reproducibilidad de los resultados**



# ¿Qué es un sistema operativo (OS)?

**sistema operativo** Conjunto de órdenes y programas que controlan los procesos básicos de una computadora y permiten el funcionamiento de otros programas y la interacción con el usuario.







Dentro de las tareas que realiza el sistema operativo, en particular, se ocupa de **gestionar la memoria de nuestro sistema y la carga de los diferentes programas**, para ello cada programa tiene una prioridad o jerarquía y en función de la misma contará con los recursos de nuestro sistema por más tiempo que un programa de menor prioridad. **El sistema operativo se ocupa también de correr procesos**.

**De primer plano:** requieren de la interacción del usuario, es el caso de un navegador web, un editor de texto, un programa de diseño de imágenes.

**De segundo plano:** son aquellos programas que no requieren del usuario y habitualmente no poseen una interfaz gráfica o "pantalla". Un ejemplo de este tipo de procesos es el anti-virus.

## Es libre

La principal ventaja que tienen muchos sistemas operativos basados en Linux es que es software libre y de código abierto. Eso significa, en la práctica, que todos pueden acceder al código fuente del sistema operativo, revisarlo y contribuir a su desarrollo.

## Es abierto

Podemos tocar, ajustar y tunear Linux hasta niveles insospechados. Desde elegir el entorno de escritorio que queremos utilizar

## Es (generalmente) más seguro

Hace unos años sucedió que las Fuerzas Armadas de los Estados Unidos sufrieron el ataque de un virus informático que inmovilizó algunos sistemas tan críticos como el que controla algunos de sus aviones no tripulados, como el Predator. ¿A qué sistema operativo migraron? A uno basado en Linux.

## Está en todos lados

Aunque no lo veamos, Linux realmente está en todas partes. Hay sistemas operativos que usan el núcleo de Linux aunque no lo parezca. Ej. plataformas que usan Linux:

- Android
- Chrome OS
- webOS (en Palm, en algunos dispositivos de HP, y en televisores de LG)
- Tizen
- Steam OS

## Puede revitalizar tu ordenador antiguo

El núcleo de Linux es extremadamente ligero y, junto con un entorno de escritorio sin demasiada floritura, **es capaz de funcionar en ordenadores que tengan más de diez años**.

# ¿Por qué Linux es mejor para programadores y desarrolladores?

## 1.Código abierto

Linux es un sistema operativo de código abierto, lo que significa que está abierto al público. **Un programador puede ver y editar o contribuir al código fuente que se usó para crearlo.** Pueden crear su propia versión del sistema operativo que puede ayudarlos con áreas especializadas o estratégicas. Otros países también están desarrollando sus propios sistemas operativos basados en el código fuente.

## 2.Sin restricciones

Linux no tiene restricciones. No es necesario que aguardes interminablemente una actualización de funciones o un parche de seguridad y obstáculos para actualizar la licencia de usuario. Un sistema Linux es muy estable y**menos propenso a malware y virus**. Se puede usar en instituciones educativas, en casa. En cualquier sitio.

## 3.Estructura configurable

Linux está estructurado como capas (Kernel, hardware, IO y UI), que son extremadamente configurables. Puedes cambiar algo si no te gusta la forma en que está funcionando. Linux no requiere una interfaz gráfica de usuario para interactuar, es decir, solo con la línea de comando (Shell). Un sistema Linux se puede ajustar con precisión para aprovechar las posibilidades máximas del hardware.

Como código de fuente abierta, los usuarios tienen derecho a ver y modificar el código fuente e incluso crear el suyo propio.

## 4.Menos requisito de hardware

Linux es muy eficiente en términos de los recursos del sistema. Dado que Linux se puede personalizar, la instalación para los usuarios y los requisitos específicos de hardware son fáciles. El procedimiento de instalación flexible permite a los usuarios elegir lo que quieran instalar. Esto incluso permite a los programadores instalar Linux en hardware antiguo y permite el uso óptimo de todos los recursos de hardware disponibles.

## 5.Disponibilidad de las herramientas de programación

Linux es gratis, es por eso que todo el software básico (que es necesario para un usuario típico o incluso un usuario avanzado) está disponible. Hay muchos programas educativos disponibles bajo Linux. Las contrapartes de software profesional para autoedición, edición de fotos, edición de audio y edición de videotambién están disponibles.

## ¿Por qué Linux es mejor para programadores y desarrolladores?

### 6.Grandes comunidades de soporte

Dado que Linux ya existía desde hace más de 26 años, a lo largo de los años había creado un fuerte apoyo comunitario. Los programadores pueden encontrar soporte fácilmente a través de Internet. Los foros y otros sitios web entusiastas de Linux pueden ayudarte de inmediato cuando tengas alguna pregunta en mente. Sin duda, nunca estarás solo con este sistema operativo.

### 7.Fácil de instalar

Como programador, instalar el sistema operativo Linux nunca debería ser un problema. Presionar las teclas necesarias durante el arranque y navegar por las indicaciones en pantalla será fácil porque Linux no es tan diferente de instalar un programa desde otros sistemas operativos.

### 8.Demanda

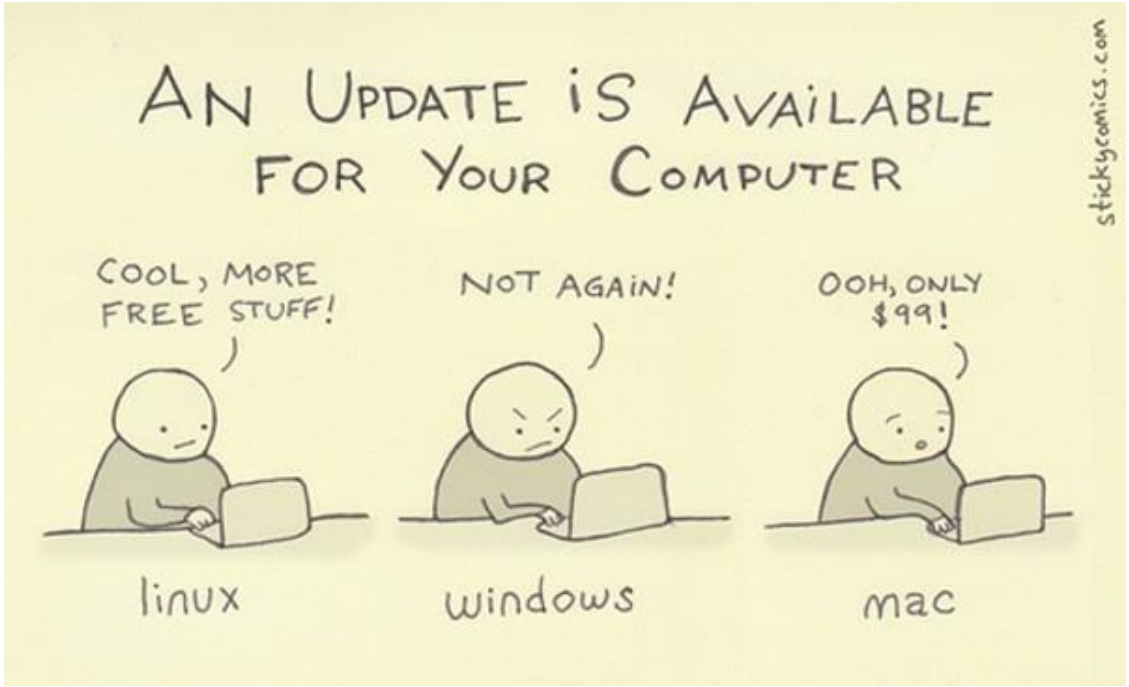
La demanda de Linux es excelente. Dado que es de código abierto y gratuito, cada vez más programadores desean crear su propia versión de Linux que pueda ser exitosa en el futuro y pueda ayudar a particulares o incluso al gobierno.

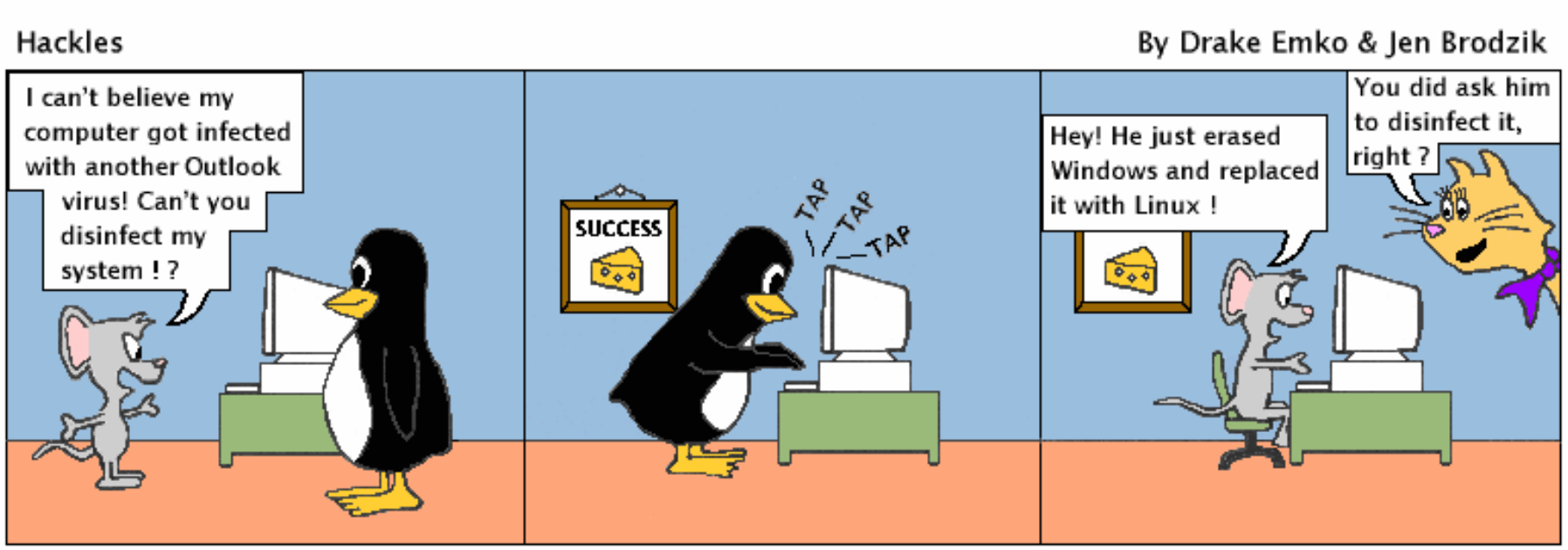
### 9.Actualizaciones fáciles

Las actualizaciones en Linux son muy fáciles. Como todo el tipo de Linux tiene su propio repositorio central de software. Actualizaciones regulares que están disponibles e incluso el sistema se puede actualizar sin reiniciarlo. Los usuarios pueden incluso automatizar el proceso de actualización. A diferencia de Windows, la actualización consume menos almacenamiento y datos de Internet.

### 10.Personalización

Linux incluye diferentes tipos de opciones para el software. Esto significa que las aplicaciones como losprocesadores de texto y los navegadores web se pueden cambiar según tu elección. Los usuarios de Linux también pueden elegir pantallas de sistema, gráficos y otros componentes de interfaz de usuario que mejor se adapten a su imagen. La idea es que, con Linux, los programadores pueden usar varios programas pequeños pero brillantes que se pueden combinar para escribir programas realmente poderosos y utilidades para que los usen.


AN UPDATE IS AVAILABLE FOR YOUR COMPUTER
stickycomics.com
COOL, MORE FREE STUFF!
NOT AGAIN!
OOH, ONLY $99!
linux
windows
mac


Hackles
By Drake Emko & Jen Brodzik
I can't believe my computer got infected with another Outlook virus! Can't you disinfect my system ! ?
SUCCESS
TAP
TAP
TAP
Hey! He just erased Windows and replaced it with Linux !
You did ask him to disinfect it, right ?
http://hackles.org
Copyright © 2001 Drake Emko & Jen Brodzik

# GNU/Linux

**GNU/Linux** es la denominación técnica y generalizada que reciben una serie de sistemas operativos de tipo Unix, que también son multiplataforma, multiusuario y multitarea, pero libres.[1] Estos sistemas operativos están formados mediante la combinación de varios proyectos, entre los cuales destaca el entorno GNU, encabezado por el programador estadounidense Richard Stallman junto a la Free Software Foundation, una fundación cuyo propósito es difundir el software libre, así como también el núcleo de sistema operativo conocido como «Linux», encabezado por el programador finlandés Linus Torvalds.[2]

https://histinf.blogs.upv.es/2011/12/23/historia-de-linux/

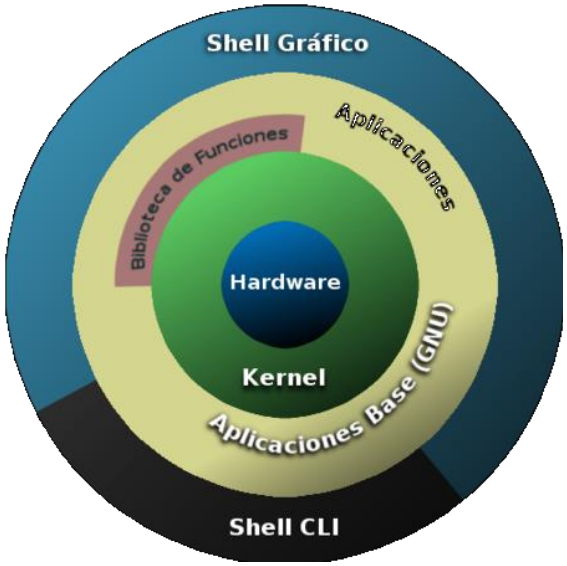

## Server oriented

RedHat, CentOS, openSuse, Ubuntu Server, Debian, ...

## Home oriented

Fedora, Linux Mint, openSuse, Ubuntu, Gentoo, Android, ...

# UNIX OS – Algunos conceptos




- What?
  - Collection of software that manages computer hardware resources and provides common services for programs
- Kernel
  - An OS kernel manages computer resources (e.g. CPU, RAM, internal/external devices) and allows programs to use
  - these resources
- Shell
  - = Command Line Interpreter (CLI) is a text-only interface between the user and the kernel. Its function is to execute commands from the terminal window
- Programs
  - External programs can be installed. The OS comes with many built-in utilities

# EVENTS LEADING TO CREATION OF LINUX

**1969: Ken Thompson, de AT&T Bell Laboratories, desarrolló el sistema operativo Unix, adaptándolo a las necesidades de un entorno de investigación**

**1975: Berkeley lanzó su propia versión de Unix (BSD). Esta versión de Unix se convirtió en la principal competidora de la versión de los laboratorios Bell de ATT&T**

**1980-1990: se fragmentó BSD, fue superado por Linux y dio origen a Mac OS X**

**1983: Richard Stallman started the GNU project with the goal of creating a free UNIX-like operating system. As part of this work, he wrote the GNU General Public License (GPL)**

**1991: Linus Torvalds announced Linux Kernel - Linus anuncio la primera versión oficial de Linux (versión 0.02). Con esta versión Linus pudo ejecutar Bash (GNU Bourne Again Shell) y gcc (El compilador GNU de C)**

**1992: Linux became Open Source**

**1994: Someone registered Linux trademark and it was not Linus Torvalds**

**1995: First Linux Expo**

**1996: Tux gets to be the symbol of Linux**

**1997: GNOME Project is born**

**1998: KDE 1.0 released**

**2000: Steve Jobs made an offer that Linus refused**

**2002: Red Hat Enterprise Linux released**

**2003: Attempt to install backdoor in Linux kernel**

**2004: Ubuntu 4.10 released**

**2005: Linus Torvalds created Git**

**2007: Linux powered netbook arrived (Asus)**

**2008: Android version 1.0 released**

**2009: Google announced Chrome OS** (Chromebooks)

**2010: Red Hat became first billion dollar open source company**

**2011: Microsoft was one of the top 5 contributors to Linux Kernel**

**2015: Microsoft has its own version of Linux**

https://histinf.blogs.upv.es/2011/12/23/historia-de-linux/

https://itsfoss.com/25-years-linux/

# ¿Por qué Linux es mejor para Bioinformática?

- Es libre, de código abierto y gratuito
- Es liviano, estable y permite optimizar recursos
- Funciona en prácticamente todos los computadores
- Permite trabajar con un gran volumen de datos
- Permite concatenar procesos de manera que los programas trabajen en serie (pipelines)
- Tiene gran soporte de la comunidad de programadores y científica
- Permite programa tu propio código en diferentes lenguajes de programación

One definition of bioinformatics is "the use of computers to analyze biological problems."
As biological data sets have grown larger and biological problems have become more complex, the requirements for computing power have also grown.
Computers that can provide this power generally use the Unix operating system -

http://bl831.als.lbl.gov/~jamesh/powerpoint/unix_commands.ppt

Slightly more advanced:
http://bl831.als.lbl.gov/~jamesh/elves/manual/tricks.html

# Basic unix commands that everyone should know

(Even if you have a mac)

# What the ~*&?!

| | |
|---|---|
| ~ | “tilde” indicates your home directory: `/home/you` |
| * | “star”: wildcard, matches anything |
| ? | wildcard, matches any one character |
| ! | History substitution, do not use |
| & | run a job in the background, or redirect errors |
| # % | special characters for most crystallography programs |
| ` \ ( [ " ' | back-quote, backslash, etc. special to shell |
| _ | underscore, use this instead of spaces!!! |

# Where am I?

**pwd**

Print name of the “current working directory”

This is the default directory/folder where the shell program will look first for programs, files, etc.  It is “where you are” in Unix space.

# What is a directory?

**`/home/yourname/whatever`**

Directories are places you put files. They are represented as words connected by the "/" character. On Windows, they use a "\", just to be different. On Mac, they are called "folders". Whatever you do…

**DO NOT PUT SPACES**

In directory/file names!

# What have we here?

# `ls`

List contents of the current working directory

`ls -l` - long listing, with dates, owners, etc.

`ls -lrt` - above, but sorted by time

`ls -lrt /home/yourname/something`

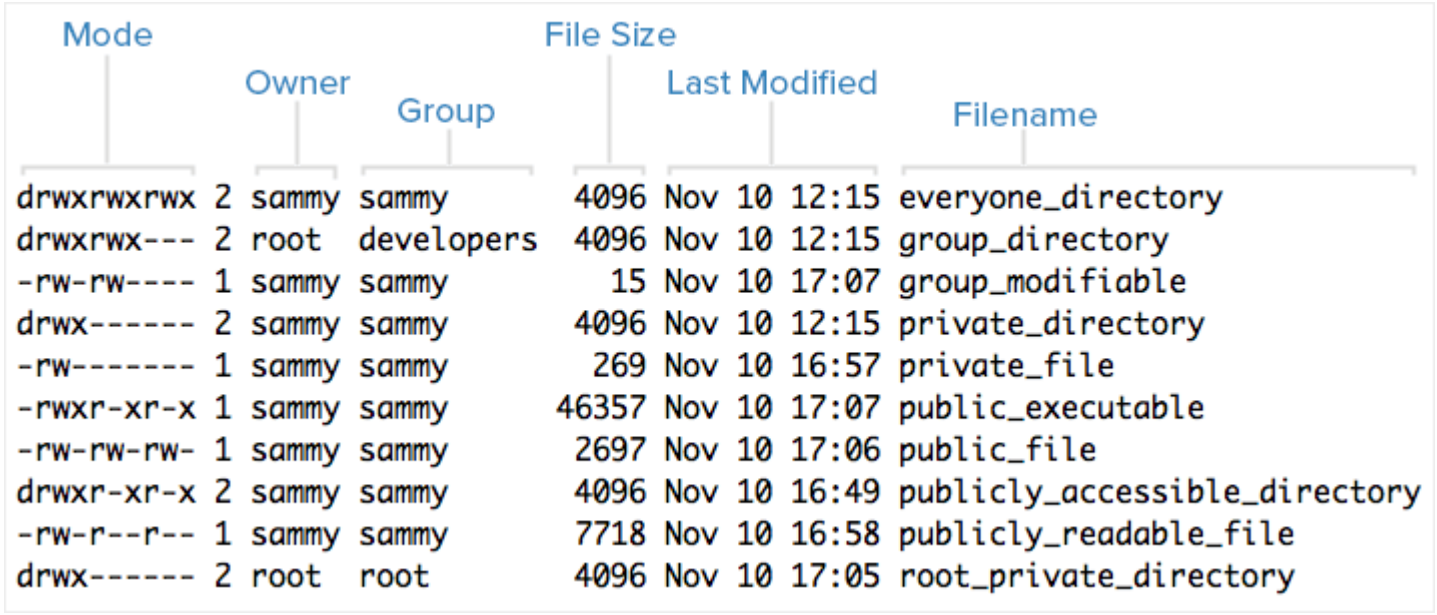
- long-list a different directory

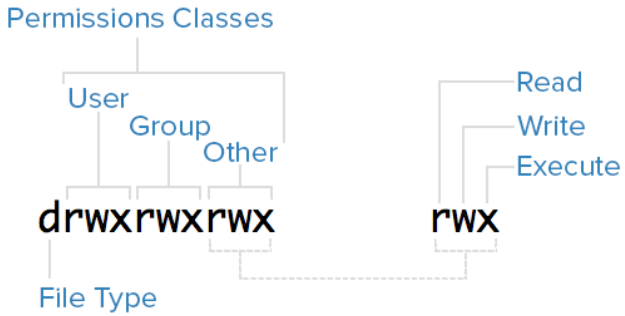
# Viewing Ownership and Permissions

In Linux, each and every file is owned by a single user and a single group, and has its own access permissions. Let's look at how to view the ownership and permissions of a file.

The most common way to view the permissions of a file is to use `ls` with the long listing option, e.g. `ls -l myfile`. If you want to view the permissions of all of the files in your current directory, run the command without an argument, like this:

```
ls -l
```



# Permissions Classes

From the diagram, we know that *Mode* column indicates the file type, followed by three triads, or classes, of permissions: user (owner), group, and other. The order of the classes is consistent across all Linux distributions.

Let's look at which users belong to each permissions class:

- **User**: The *owner* of a file belongs to this class
- **Group**: The members of the file's group belong to this class
- **Other**: Any users that are not part of the *user* or *group* classes belong to this class.

**Here is a quick breakdown of the access that the three basic permission types grant a user:**

**Read**

For a normal file, read permission allows a user to view the contents of the file.
For a directory, read permission allows a user to view the names of the file in the directory.

**Write**

For a normal file, write permission allows a user to modify and delete the file.
For a directory, write permission allows a user to delete the directory, modify its contents (create, delete, and rename files in it), and modify the contents of files that the user has write permissions to.

**Execute**

For a normal file, execute permission allows a user to execute a file (the user must also have read permission). As such, execute permissions must be set for executable programs and shell scripts before a user can run them.

For a directory, execute permission allows a user to access, or traverse, into (i.e. `cd`) and access metadata about files in the directory (the information that is listed in an `ls -l`).

# Permissions Classes

## Examples of Modes (and Permissions)

Now that know how to read the mode of a file, and understand the meaning of each permission, we will present a few examples of common modes, with brief explanations, to bring the concepts together:

- -rw-------: A file that is only accessible by its owner
- -rwxr-xr-x: A file that is executable by every user on the system. A “world-executable” file
- -rw-rw-rw-: A file that is open to modification by every user on the system. A “world-writable” file
- drwxr-xr-x: A directory that every user on the system can read and access
- drwxrwx---: A directory that is modifiable (including its contents) by its owner and group
- drwxr-x---: A directory that is accessible by its group

# “Permission denied” !?

# chmod

Change the “permission” of a file.

`chmod a+r filename.txt`

- make it so everyone can read it

`chmod u+rwx filename.txt`

- make it you can read/write/execute it

`chmod -R u+rw /some/random/place`

- make it so you can read/write everything under a directory

# Go somewhere else?

# cd

Change the current working directory

`cd /tmp/yourname/`
- go to your temporary directory

`cd -`
- go back to where you just were

`cd`
- no arguments, go back “home”

“home” is where your login starts

# A new beginning…

# `mkdir`

Create a new directory.

`mkdir ./something` - make it

`cd ./something` - go there

`ls` - check its is empty

# How do I get help?

# `man`

Display the manual for a given program

`man ls` - see manual for the "`ls`" command
`man tcsh` - learn about the C shell
`man bash` - learn about that other shell
`man man` - read the manual for the manual

to return to the command prompt, type "`q`"

# Move it!

# mv

Move or rename a file.  If you think about it, these are the same thing.

```
mv stupidname.txt bettername.txt
```

- change name

```
mv stupidplace/file.txt ../betterplace/file.txt
```

- same name, different directory

```
mv stupidname_*.img bettername_*.img
```

Will not work! Never ever do this!

# Copy machine


Copy a file. This is just like "`mv`" except it does not delete the original.

```
cp stupidname.txt betternname.txt
```

- change name, keep original

```
rm stupidname.txt
```

- now this is the same as "`mv`"

# Destroy! Destroy!

# `rm`

Remove a file forever. There is no “trash” or “undelete” in unix.

`rm unwanted_file.txt`

- delete file with that name

`rm -f /tmp/yourname/*`

- forcefully remove everything in your temporary directory.

Will not prompt for confirmation!

# less is more

# `more`

Display the contents of a text file, page by page

`more filename.txt` – display contents

`less filename.txt` – many installs now have a replacement for "`more`" called "less" which has nicer search features.

to return to the command prompt, type "`q`"

# After the download…

# `gunzip`

File compression and decompression

`gunzip ~/Downloads/whatever.tar.gz`

- decompress

`gzip ~/Downloads/whatever.tar`

- compress, creates file with `.gz` extension

# Where the %$#& is it?

# `find`

Search through directories, find files

`find ./ -name 'important*.txt'`

- look at everything under current working directory with name starting with "`important`" and ending in "`.txt`"

`find / -name 'important*.txt'`

- will always find it, but take a very long time!

# Did I run out of disk space?

# df du

Check how much space is left on disks

`df` - look at space left on all disks

`df .` - look at space left in the current working directory

`du -sk . | sort -g`

- add up space taken up by all files and subdirectories, list biggest hog last

# Why so slow?

# ps top

Look for programs that may be eating up CPU or memory.

`top` - list processes in order of CPU usage

`jobs` - list jobs running in background of current terminal

`ps -fHu yourname`

- list jobs belonging to your account in order of what spawned what

# Die Die Die!

# `kill`

Stop jobs that are running in the background

`kill %1` - kill job [1], as listed in "`jobs`"

`kill 1234` - kill job listed as 1234 by "`ps`" or "`top`"

`kill -9 1234` - that was not a suggestion!

`kill -9 -g 1234` – seriously kill that job and the program that launched it

Linux Cheat sheet
by A.Mahouachi

## 1 File Commands

**ls [options] file**

options

-a: show hidden files
-A: show hidden files except . and ..
-d: only show directories
-h: human readable size
-i: inode info
-l: long list format
-m: output as csv
-n: numeric uid and guid
-r: sort in reverse order
-S: sort by file size
-t: sort by modification time

**tree [options] dir**

options

-d: only directories
-f: show full paths
-P pattern: only matching pattern
-I pattern: except matching pattern
-h: print sizes in human readable format
-C: use colors
-L max: max level depth

**cp [options] source dest**

options

-b: backup dest before overwrite
-r: recursive
-f: force
-l: link files instead of copy
-P: dont follow sym links
-i: interactive
-u: copy only if source newer than dest

**mv [options] source dest**

options

-b: backup dest before overwrite
-f: force
-i: interactive
-u: move only if source newer than dest

**ln [options] file link**

options

-s: sym link (hard by default)
-f: overwrite link if exists
-b: backup old link before overwrite

**rm [options] file**

options

-f: force
-i: interactive
rm – -foo if file name is -foo

**chmod [options] mode file(s)**

options

-R: recursive

symbolic mode

format: [ugoa][[+-=][perms]],...
example: u+x,o-wx,g-w
u: owner
g: group
o: others
a: all
+: add mode
-: remove mode
=: exact mode
r: read
w: write
x: execute files and search for dirs
X: search for dirs
s: setuid or setguid
t: sticky bit

numeric mode

format: [0-7]1,4
example: 755
first digit: setuid(4), setguid(2)
second digit: owner perms
third digit: group perms
fourth digit: others perms
read: 4
write: 2
execute: 1

**find path [options] [tests] [actions]**

options:

-mindepth: start from min level in hierarchy
-maxdepth: end with max level in hierarchy

tests:

-name "xyz*": name like xyz*
-iname "xyz*": like -name but case insensitive
-type d: only directories
-type f:only files
-mtime 0: modified < 1 day
-mtime -x: modified < x days
-mtime +x: modified > x days
-mmin: like -mtime but in minutes
-size +100M: size > 100mb
-size -100M: size < 100mb (k for kb, G for gb)
-perm /o+w: writable by others
! -perm /o+r : not readable by others

actions:

-print: print matching
-delete: rm matching files
-exec cmd '{}' ; : run cmd for every match
-exec cmd '{}' + : run cmd at the end of search
-exec rm -rf '' : rm -rf matching items
-fprint /tmp/result: write matches to /tmp/result

**diff [options] files**

options

-r: recursive
-w: ignore whitespaces
-B: ignore blank lines
-q: only show file names
-x".sync*": exclude files with path like .sync*

**grep [options] pattern files**

options

-i: ignore case
-P: pattern is a perl regex
-m: stop after m matches
-n: also show matching line number
-R: recurse directories
-c: only show matching lines count
–exclude=glob : exclude these
–include=glob : only consider these

**cat [options] file(s)**

options

-v: non ascii chars except tab and eol
-T: show tabs
-t: equivalent to -vT
-E: show eol end of line
-e: equivalent to -vE
-A: equivalent to -vET
-s: remove repeat empty lines

**tail [options] file**

options

-f: show end of file live
-35: show last 35 lines
-q: be quiet

**head [options] file**

options

-35: show first 35 lines
-q: be quiet

**tac file(s)**

print files starting from last line

**cut [options] file**

options

-d char: use char as delimiter
-f 1,3,5: print fields 1, 3 and 5

**uniq [options] input output**

options

-c: prefix lines by number of occurrences
-d: only print duplicate lines
-u: only print unique lines

**sort [options] file**

options

-n: numeric sort
-b: ignore blank lines
-f: ignore case
-r: reverse order

**tar [options] file**

options

-f file: archive file
-c: create
-t: list
-x: extract
-C DIR: cd to DIR
-z: gzip
-j: bzip2

**du [options] file**

options:

-c: a grand total
-h: human readable
-L: dereference sym links
-P: no dereference of sym links
-s: total for each argument
–exclude=pattern
-max-depth=N: dont go deeper than N

**df [options] file**

options:

-h: human readable
-i: list inodes info
-P: no dereference of sym links

## 2 Process Commands

**ps [options]**

options:

-e: all processes
-f: full listing
-H: show hierarchy
-p pid: this process pid
-C cmd: this command name cmd
-w: wide output
-ww: to show long command lines
-l: long listing, including wchan
-o x,y,z: show columns x y z
-o user,pid,cmd: show columns user, pid command
-N: negation
-u user: processes owned by user
-u user -N: processes not owned by user
–sort=x,y: x y are columns in ps output
–sort=user: sort by user
–sort=+time: sort by cpu time asc
–sort=-time: sort by cpu time desc
–sort=size: sort by memory size
–sort=vsize: sort by vm size

**top [options]**

options

-d x: refresh every x seconds
-p pid1 -p pid2: only processes with pid1 pid2
-c : show command lines

interactive commands

space: udpdate display
n: change number of displayed processes
up and down: browse processes
k: kill a process
o: change order
T: sort by time
A: sort by age
P: sort by cpu
M: sort by memory
c: display/hide command line
m: display/hide memory
t: display/hide cpu
f: manage list of displayed columns
up and down: move between columns
d: display/hide the selected column
q: apply and quit the field mgmt screen

**pgrep [options] pattern**

options

-l : show pid and process name
-a : show pid and full command line
-n : if more than one show newest
-o : if more than one show oldest
-u uid : show only processes of uid
-c : count results

## 3 Network & Remote

**ssh [options] user@host ["cmd1;cmd2"]**

options:

-2: force protocol 2
-o StrictHostKeyChecking=no: ignore warnings due to remote host key change
-X: forward X11 display

**wget [options] url**

options:

-b: run in background
-o file: print wget output in file
-o /dev/null: suppress wget output
-q: be quiet
-d: debug
-O file: save response to file
-c: resume file download
-S: print server headers
-T N: timeout after N seconds
–user=user: basic http auth user
–password=password: basic http auth password
–save-cookies file: save cookies to file
–load-cookies file: use file as cookies
–post-data=string
–post-file=file
–no-check-certificate: ignore ssl certificate

**curl [options] url**

options:

-H header: like -H "Host: st.com"
-u <user:password>: basic http auth
-s: be silent
-S: show errors if silent mode
-L: follow new location in case 301
–data "field=value": x-www-form-urlencoded query
–data-binary data: post data as is without encoding
–data-binary @filename: post filename content as is
-X method: use PUT, GET, POST etc.
–request method: use PUT, GET, POST etc.

**mail [options] to-address**

options:

-s subject: email with subject
-c address1,address2: cc copy
-b address1,address2: bcc copy
mail -s 'hello there' 'joe@st.com' < somefile

## 4 Terminal

Ctrl+C: halt current command
Ctrl+Z: pause current command
bg %1: resume paused command in background
fg %1: resume paused command in foreground
Ctrl+D: logout
Ctrl+W: remove a word from current line
Ctrl+U: remove current line
Ctrl+A: go to beginning of current line
Ctrl+E: go to end of current line