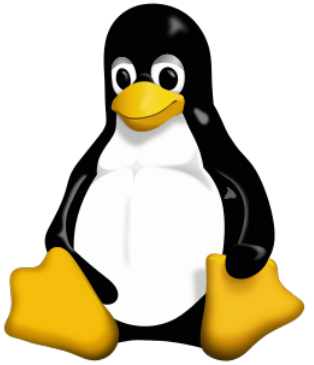


GNU/LINUX Lv0, Lv1

8 – 12 mayo

Temas

Lunes 8. Introducción

Martes 9. Archivos y sistema de archivos

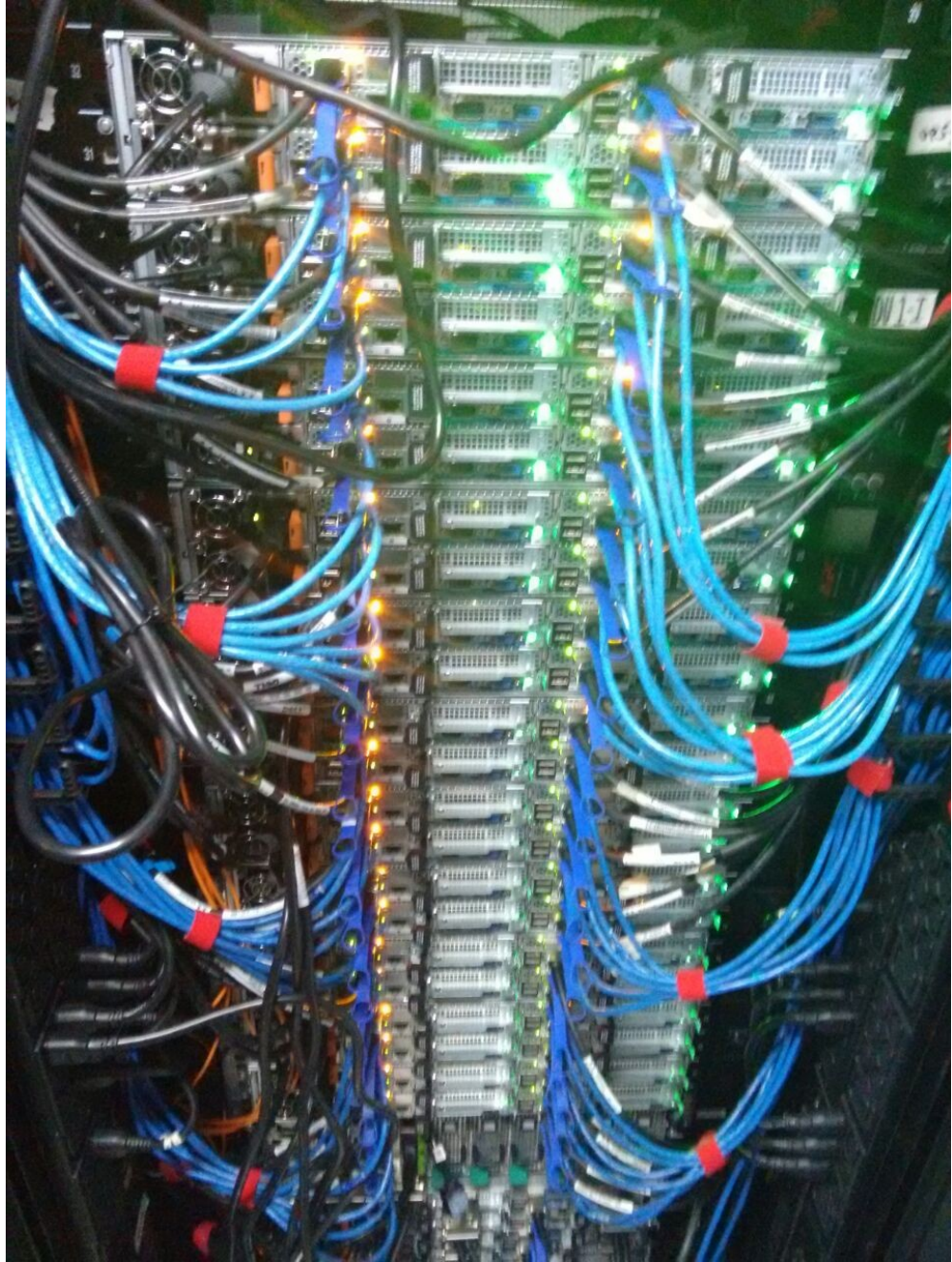
Jueves 11. Ejecución de programas

Viernes 12. Trabajo remoto

Sistema operativo GNU/Linux

- **GNU/Linux** es un sistema operativo que al igual que su predecesor UNIX, es utilizado en la instrustria así como en el campo científico.
- Para **1998** la lista de las 500 computadoras más poderosas el planeta (www.top500.org) reportaba el 46% usando UNIX (SUN, SGI) y solo un equipo (0.2%) con LINUX. Para el **2005** se reportaba que SUN tenia el 0.8%, SGI 3.6%, mientras que LINUX el 72%.
- En noviembre del **2017** la lista reporta el 0.4% usando UNIX y el 99.6% usando LINUX.

**El poder de procesamiento de un sistema de cómputo se basa en la cantidad de operaciones de punto flotante con datos de 64 bits.*



GNU/Linux origenes.

Para ayudarnos a comprender el funcionamiento del sistema operativo **GNU/Linux** nos apoyaremos en sus origenes.

- El desarrollo del **kernel Linux** comienza en **1991** por el entonces estudiante de la universidad de Finlandia **Linus Torvalds**. Inspirandose en el sistema operativo MINI-UNIX o “**MINIX**“ desarrollado por el profresor Andrew S. Tanenbaum para fines didácticos en **1987**.
- Es sus inicios el licenciamiento del **kernel linux** no permitia su distribución comercial, pero en **1992** la versión **0.12** fue publicada bajo el **licenciamiento GPL** del movimiento **GNU**.
- En un sistema de **tipo Unix**, el programa que asigna los recursos de la máquina y se comunica con el hardware se denomina «**núcleo**». **GNU** se usa generalmente con un núcleo llamado «Linux». Esta combinación es el sistema operativo **GNU/Linux**. Millones de personas usan **GNU/Linux**, aunque muchos lo llaman erróneamente «Linux».

UNIX / GNU / LINUX / DISTRIBUCIONES

The UNIX philosophy is to provide many small tools that can be changed or combined to perform new functions. You can create new tools without writing C programs by writing shell scripts.

This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

At its heart is the idea that the power of a system comes more from the relationships among programs than from the programs themselves. Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools.

UNIX / GNU / LINUX / DISTRIBUCIONES

A finales de los años **1960** Dennis Ritchie, Ken Thompson y otros trabajadores de la AT&T (Bell laboratories) desarrollaban un proyecto para generar un sistema operativo **multi-usuario** para un equipo de cómputo "**mainframe GE-645**" de la General Electric Corporation. El nombre para este sistema operativo fue **MULTICS** (**Multiplexed Information and Computer Services**).

Cuando los laboratorios Bell abandonan el proyecto **MULTICS**, Ken Thompson toma la experiencia adquirida en el desarrollo de **MULTICS** y crea un sistema operativo para el equipo **DEC PDP-7** de la Bell laboratories.

Este nuevo sistema operativo (uni-usuario) se llamo originalmente UNICS (**Uniplexed Information and Computing Service**) lo que derivo en el nombre **UNIX**.

UNIX / GNU / LINUX / DISTRIBUCIONES

UNIX / GNU / LINUX / DISTRIBUCIONES

UNIX / GNU / LINUX / DISTRIBUCIONES

- UNIX primera edición **1971**. Lenguaje ensamblador
- UNIX segunda edición **1972**. Lenguaje de programación B
- UNIX tercera edición **1973**. Leguaje de programación C
- UNIX quinta edición **1975**. Licencia de uso para centros de investigación y universidades, código fuente incluido.
- UNIX sexta edición **1976**. Licenciamiento para uso comercial.
- UNIX septima edición **1979**. Se prohibe el uso del código fuente para fines educativos.

Las primeras versiones de UNIX funcionaban sobre equipos DEC PDP de los mismos laboratorior bell o de universidades y centros de investigación

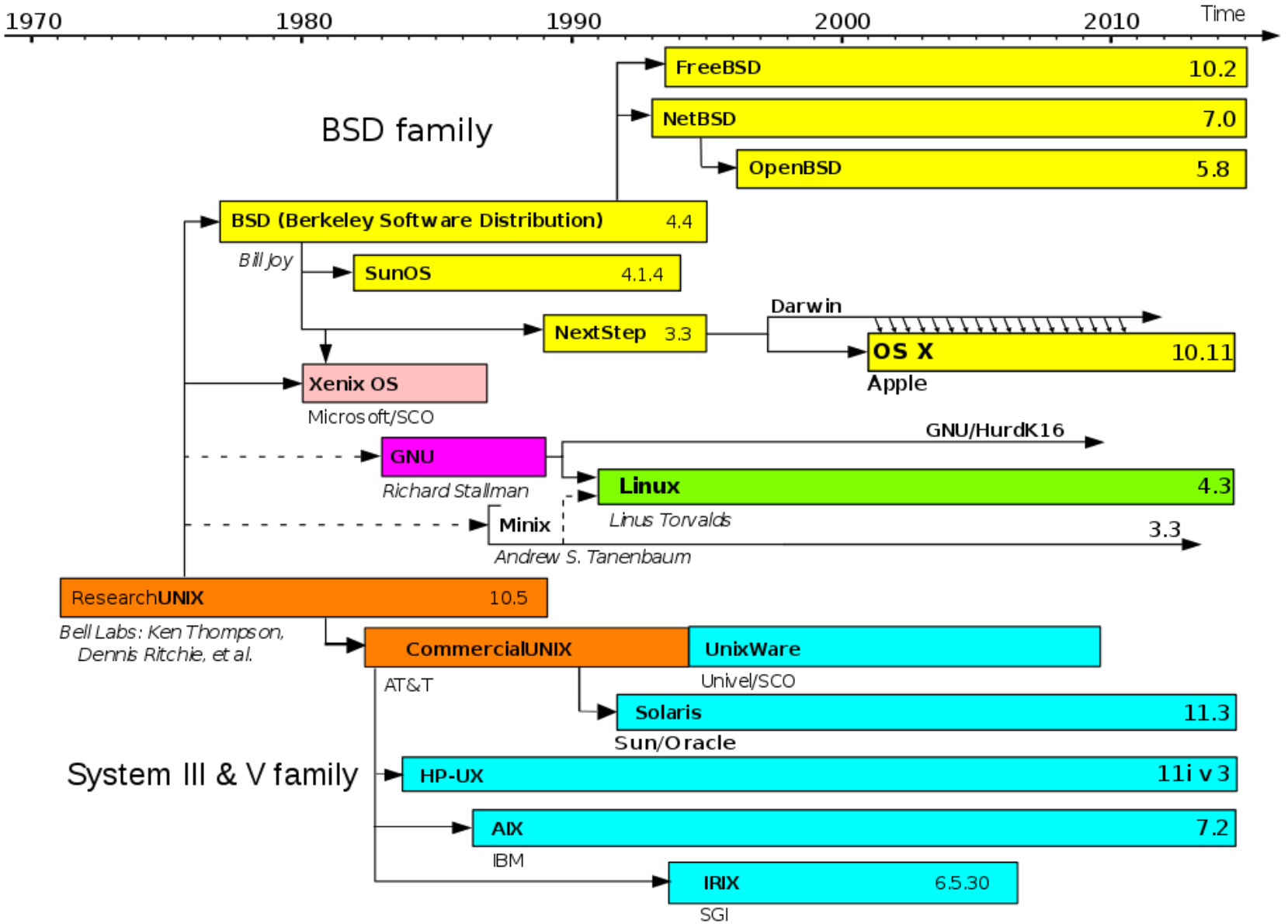
UNIX / GNU / LINUX / DISTRIBUCIONES

- **UNIX System III 1981** (basado en implementaciones internas y externas de ediciones anteriores)

Nota. No existieron versiones I,II o IV.

- **UNIX System V**
  - release 1 (SRV1) 1983
  - release 2 (SRV2) 1984
  - release 3 (SRV3) 1986
  - release 4 (SRV4) 1988

En el periodo **1980** y **1990** UNIX SRV fue considerado uno de los 2 sistemas UNIX mas populares en el marcado (académico y comercial). El otro sistema era conocido como **BSD UNIX** o **Berkeley UNIX**.


1970
1980
1990
2000
2010
Time
BSD family
FreeBSD 10.2
NetBSD 7.0
OpenBSD 5.8
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
NextStep 3.3
Darwin
OS X 10.11
Apple
Xenix OS
Microsoft/SCO
GNU
Richard Stallman
GNU/HurdK16
Linux 4.3
Linus Torvalds
Minix 3.3
Andrew S. Tanenbaum
ResearchUNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
CommercialUNIX
UnixWare
Univel/SCO
AT&T
Solaris 11.3
Sun/Oracle
System III & V family
HP-UX 11i v3
AIX 7.2
IBM
IRIX 6.5.30
SGI

System V R4

Caracteristicas incluidas en **UNIX SVR4**

Kernel + utilerias + interprete de comandos

El **kernel** es un programa de software que interactua directamente con el hardware de la computadora. Cuando el equipo es encendido el kernel es cargado en memoria y permanece en memoria principal durante todo el tiempo que el equipo esta encendido (no es candidato a swap).

Cuando ejecutamos un comando o aplicación de software, estos programas utilizan llamadas al sistema “system calls“ para solicitar servicios al kernel, por ejemplo la lectura de un archivo.

System V R4

Caracteristicas incluidas en **UNIX SVR4**

Kernel + **utilerias** + interprete de comandos

En ocaciones se habla del kernel como el sistema operativo, pero el kernel es solo un programa que se auxilia de otros programas para atender las solicitudes del usuario.

Estos progrmas auxiliares ó utilerias se etregan junto con el kernel para formar el sistema operativo.

El interprete de comandos así como algunos comandos para la administración de archivos formaban parte del kernel en la primeras versiones del kernel UNIX.

System V R4

Caracteristicas incluidas en **UNIX SVR4**

Kernel + **utilerias** + interprete de comandos

V·T·E	**Unix command-line interface programs and shell builtins**	[hide]
File system	cat · chmod · chown · chgrp · cksum · cmp · cp · dd · du · df · file · fuser · ln · ls · mkdir · mv · pax · pwd · rm · rmdir · split · tee · touch · type · umask	
Processes	at · bg · crontab · fg · kill · nice · ps · time	
User environment	env · exit · logname · mesg · talk · tput · uname · who · write	
Text processing	awk · basename · comm · csplit · cut · diff · dirname · ed · ex · fold · head · iconv · join · m4 · more · nl · paste · printf · sed · sort · strings · tail · tr · uniq · vi · wc · xargs	
Shell builtins	alias · cd · echo · test · unset · wait	
Searching	find · grep	
Documentation	man	
Software development	ar · ctags · lex · make · nm · strip · yacc	
Miscellaneous	bc · cal · expr · lp · od · sleep · true and false	
Categories (Standard Unix programs · Unix SUS2008 utilities) · **List**		

System V R4

Caracteristicas incluidas en **UNIX SVR4**

Kernel + utilerias + interprete de comandos

El usuario solicita servicios del sistema operativo escribiendo el los comandos correspondientes, el interprete de comandos toma estos comandos y ejecuta los programas solicitados.

El interprete de comandos oculta los detalles internos del sistema operativo (verificar la existencia del programa solicitado por ejemplo) así como los mecanismos de llamadas al kernel para solicitar la ejecución del programa (solicitar la carga del programa solicitado en memoria para comenzar su ejecución).

El interprete de comandos desde las primeras versiones se conocio también como el programa **shell**.

Acceso a

Linus Torvalds has said that if 386BSD or the GNU HURD kernel had been available at the time, he probably would not have created Linux.

UNIX / GNU / LINUX / DISTRIBUCIONES

Berkeley software distribution. **BSD**

Las primeras entregas de UNIX de los laboratorios bell (**1970**) incluían el código fuente del sistema operativo permitiendo a los invetigadores y estudiantes de las universidades modificar y extender UNIX. La primera versión de UNIX llego a la universidad de **Berkeley** en **1974**.

En **1975** **Ken thompson** toma un sabático de los labratorios bell y trabaja como profesor visitante en **Berkeley**, ahí ayuda en la instalación de UNIX y conoce a **Bill joy**.

Para **1978** **Bill Joy** lanza la primera versión de BSD como un “agregado“ a la versión de UNIX que tenia la universidad.

UNIX / GNU / LINUX / DISTRIBUCIONES

- 2 BSD. La segunda entrega de **BSD** (**1979**), como agregado también, incluia los programa vi y csh.
- 3 BSD (**1979**). Antes de este lanzamiento **BSD** era distribuido como un conjunto de programas que complementaban y/o mejoraban a los distribuidos con UNIX (kernel incluido). Esta versión 3 de **BSD** fue lazada como un sistema operativo completo y patrocinada por la **DARPA**.
- 4.2 BSD (**1983**). Incluye soporte para **TCP/IP**: Transmission Control Protocol (**TCP**) / Internet Protocol (**IP**)
- 4.3 BSD (**1986**). Por problemas legales por el uso del código UNIX surgieron 2 versiones Net/1 (**1989**) y Net/2 (**1991**).
- 4.4 BSD-Lite Release 2 (**1994**). Último lanzamiento de **BSD** y nacimiento de variantes como **FreeBSD**, **NetBSD**, **OpenBSD**.

BSD aporto múltiples mejoras al sistema UNIX original las cuales fueron tomadas por el sistema GNU.

UNIX / GNU / LINUX / DISTRIBUCIONES

Today tens of millions of users are using an operating system that was developed so they could have freedom—but they don't know this, because they think the system is Linux and that it was developed by a student 'just for fun'.

It can't be fair to give all the credit to one secondary contribution (Linux) while omitting the principal contribution (GNU).

— Richard Stallman

UNIX / GNU / LINUX / DISTRIBUCIONES

En septiembre de 1983 **Richard Stallman** hace público el proyecto **GNU** con el objetivo de libertad y control en el uso de un equipo de cómputo, el usuario tiene:

I. La libertad de ejecutar el programa como lo desee, con cualquier propósito (libertad 0).

II. La libertad de estudiar el funcionamiento del programa y adaptarlo a sus necesidades (libertad 1). El acceso al código fuente es un prerrequisito para esto.

III. La libertad de redistribuir copias para ayudar a los demás (libertad 2).

IV. La libertad de mejorar el programa y de publicar las mejoras, de modo que toda la comunidad se beneficie (libertad 3). El acceso al código fuente es un prerrequisito para esto.

El proyecto GNU garantiza estos derechos legalmente por medio del licenciamiento GNU GPL “copyleft“.

UNIX / GNU / LINUX / DISTRIBUCIONES

La primera actvidad del proyecto **GNU** fue el desarrollo se un sistema operativo kernel + utilerias + interprete de comandos liberados bajo el licenciaiento **GPL**.

Richard Stallman al ser un usuario UNIX decidio que el sistema operativo GNU fuera “tipo UNIX“, esto es se debia comportan como un sistema operativo UNIX pero sin utilizar nada de su código fuente.

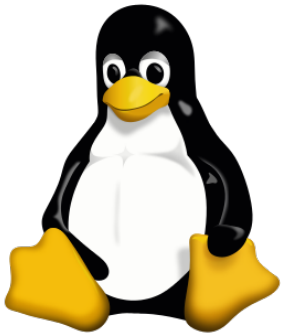
El nombre GNU es un acrónimo recursivo de “**GNU is not UNIX**“ Gnu no es unix.

https://www.gnu.org/philosophy/philosophy.html

UNIX / GNU / LINUX / DISTRIBUCIONES

Todo usuario de un equipo de cómputo necesita un sistema operativo. Así, el primer elemento en la agenda del software libre tenía que ser un sistema operativo libre.

A principios de **1990** ya habían encontrado o programado los componentes principales excepto uno, el núcleo. En **1991** Linus Torvalds programó Linux, un núcleo similar a Unix, y en **1992** lo convirtió en software libre. La combinación de Linux con el sistema GNU, que ya estaba prácticamente completo, formó un sistema operativo completo: el sistema **GNU/Linux**.

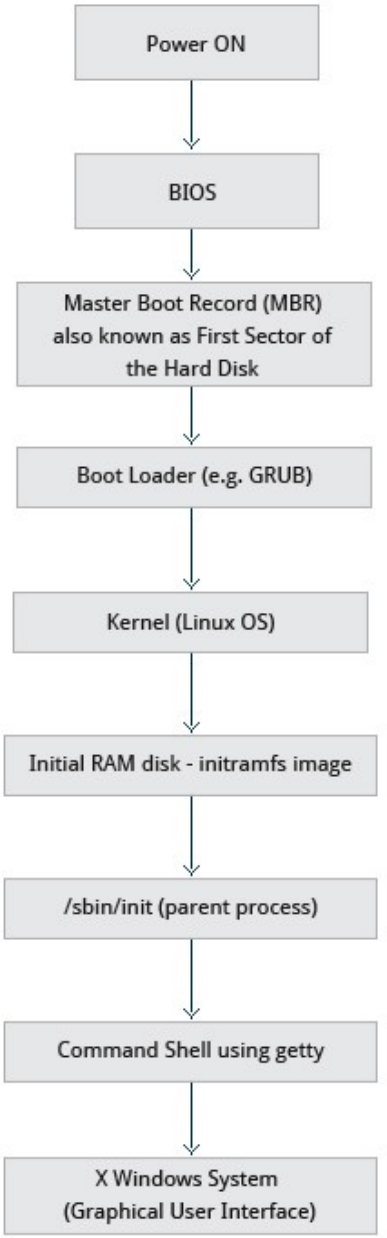

+

= GNU/Linux

UNIX / GNU / LINUX / DISTRIBUCIONES

Los componentes base proporcionados por el proyecto GNU son:

- GNU Compiler Collection **GCC.**
- GNU C Library **glibc.**
- GNU Core Utils **coreutils.**
- GNU Binary Utilities **binutils.**
- GNU Bash Shell **bash.**
- GNOME Desktop Environment

UNIX / GNU / LINUX / DISTRIBUCIONES



UNIX / GNU / LINUX / DISTRIBUCIONES

El primer objetivo del proyecto **GNU** fue crear un sistema operativo formado por software libre.

Para 1992 el sistema operativo **GNU** tenia todas las piezas (utilerias y shell) pero no el kernel: **GNU Hurd**.

Con el lanzamiento del **kernel linux** bajo la licencia **GPL**, por primera vez fue posible ejecutar un sistema operativo compuesto solamente por software libre.

La entrega de las herramientas **GNU** junto con el **kernel linux** y otros programas comenzo a llamarse **distribución linux**.

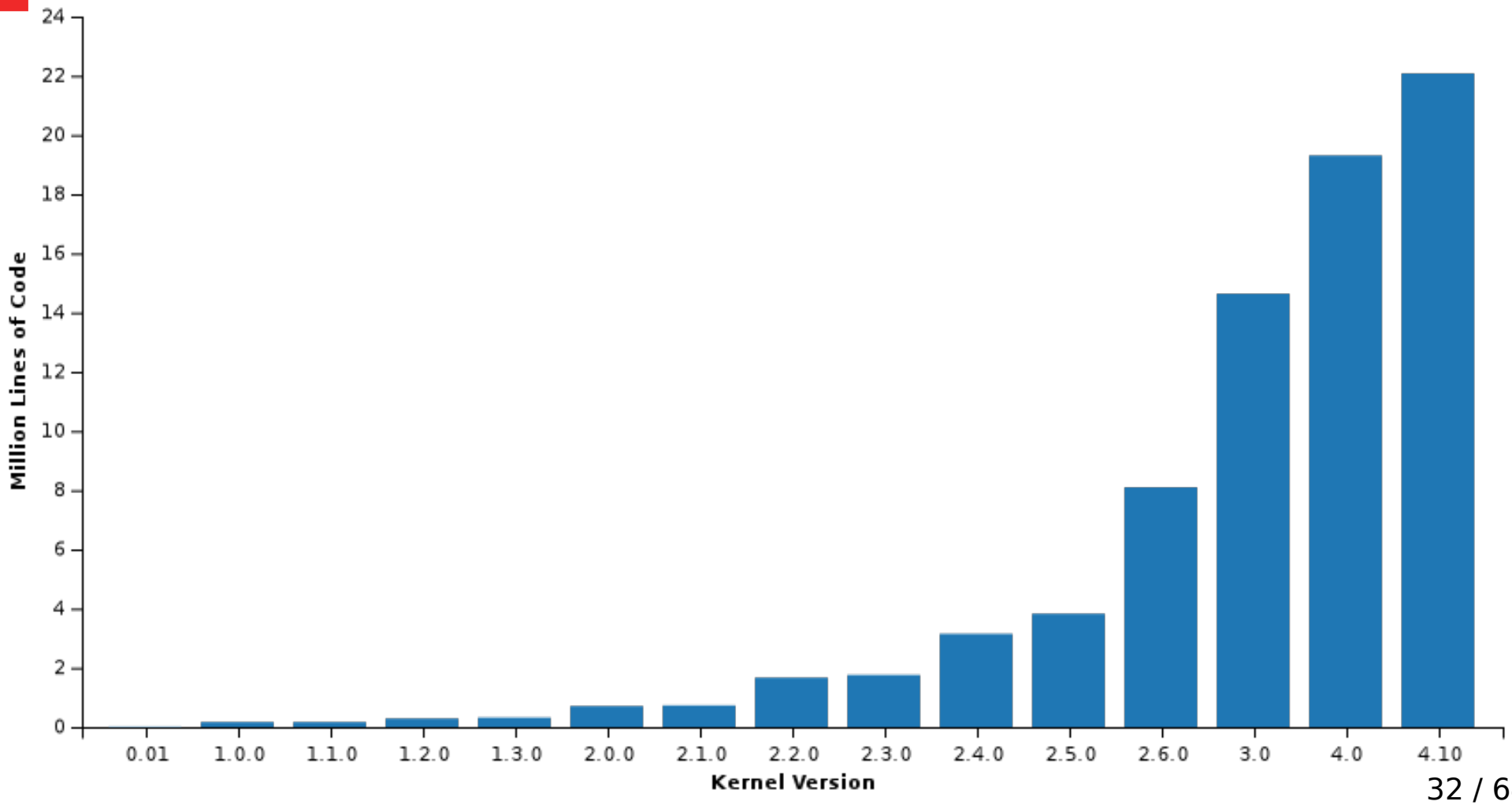
El proyecto GNU llama a la combinación de GNU y el kenel linux "**GNU/Linux**".

UNIX / GNU / LINUX / DISTRIBUCIONES

Lanzamientos del kernel linux

Versión	Fecha	Característica principal
1.0	14/03/1994	Soporte uni procesador i386
2.0	09/06/1996	Soporte multi-procesador SMP
2.2	20/01/1999	Soporte PowerPC (IBM / os x)
2.4	04/01/2001	Soporte bluetooth
2.6	17/12/2003	Soporte PAE > 2GB RAM
3.0	22/07/2011	Cambio en el # de lanzamiento
4.0	12/04/2015	Cambio en el # de lanzamiento
4.11	30/04/2017	Lanzamiento más reciente

UNIX / GNU / LINUX / DISTRIBUCIONES



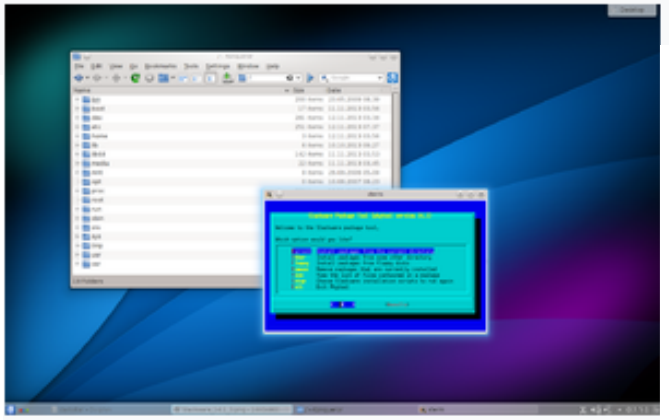
UNIX / GNU / LINUX / DISTRIBUCIONES

SLACKWARE 1993

DEBIAN 1993

RED HAT 1994

Slackware



Red Hat Linux

Debian

UNIX / GNU / LINUX / DISTRIBUCIONES

CENTOS 2004

UBUNTU 2004

http://distrowatch.com/

Ubuntu

CentOS

Temas

Lunes 8. Introducción

Martes 9. Archivos y sistema de archivos

Jueves 11. Ejecución de programas

Viernes 12. Trabajo remoto

Sistema operativo GNU/Linux

Un sistema operativo tipo UNIX esta compuesto por:

- Kernel
- Herramientas o utilerias
- Interprete de comandos o shell

Sistema operativo GNU/Linux

Un sistema operativo tipo UNIX esta compuesto por

- Kernel
- Herramientas o utilerias
- Interprete de comandos o shell

Estos componentes en una distribución GNU/LINUX son:

- Linux
- GNU (coreutils,
- Bash

Archivos y sistema de archivos

Toda la información almacenada en el sistema operativo GNU/Linux, a partir de ahora solo sistema, en archivos.

Tecnicamente un archivo es una serie de bytes almacenados en un sistema de almacenamiento persistente como un disco duro.

Dentro del sistema hay 2 archivos importantes, de texto (ASCII) y binarios.

No se espera que el archivo sea almacenado en una estructura en particular, es el usuario quien da el formato y posteriormente los programas son hechos para esperar información acorde al formato.

Ejemplos: archivo passwd
curso.lsvp:x:50001:50001::/home/curso.lsvp:/bin/bash

Nombre de archivos

Para identificar un archivo generalmente damos un nombre que nos ayude para asociarlo con su contenido. El sistema de archivos (software auxiliar al kernel) asigna un número único a cada archivo, este número se llama i-nodo.

El nombre de un archivo puede contener cualquier carácter exepto - / -.

Se recomienda que el nombre de un archivo no contenga los siguientes caracteres ya que tienen un significado para el shell:

| & ; ( ) < > [] { } ~ “ ‘ * \ ? #

Directorios

Un directorio es una forma de organizar los archivos en el sistema.

Para el sistema de archivos un directorio es un archivo más cuyp su contenido son los nombre e i-nodos de otros archivos.

Cuando el admisnitrador del sistema nos crea una cuenta, necesariamente se crea un directorio de ingreso o “login“ en donde somos posicionados cada ves que ingresamos en el sistema.

El directorio de ingreso es conocido también como directorio HOME

Sesion pasada

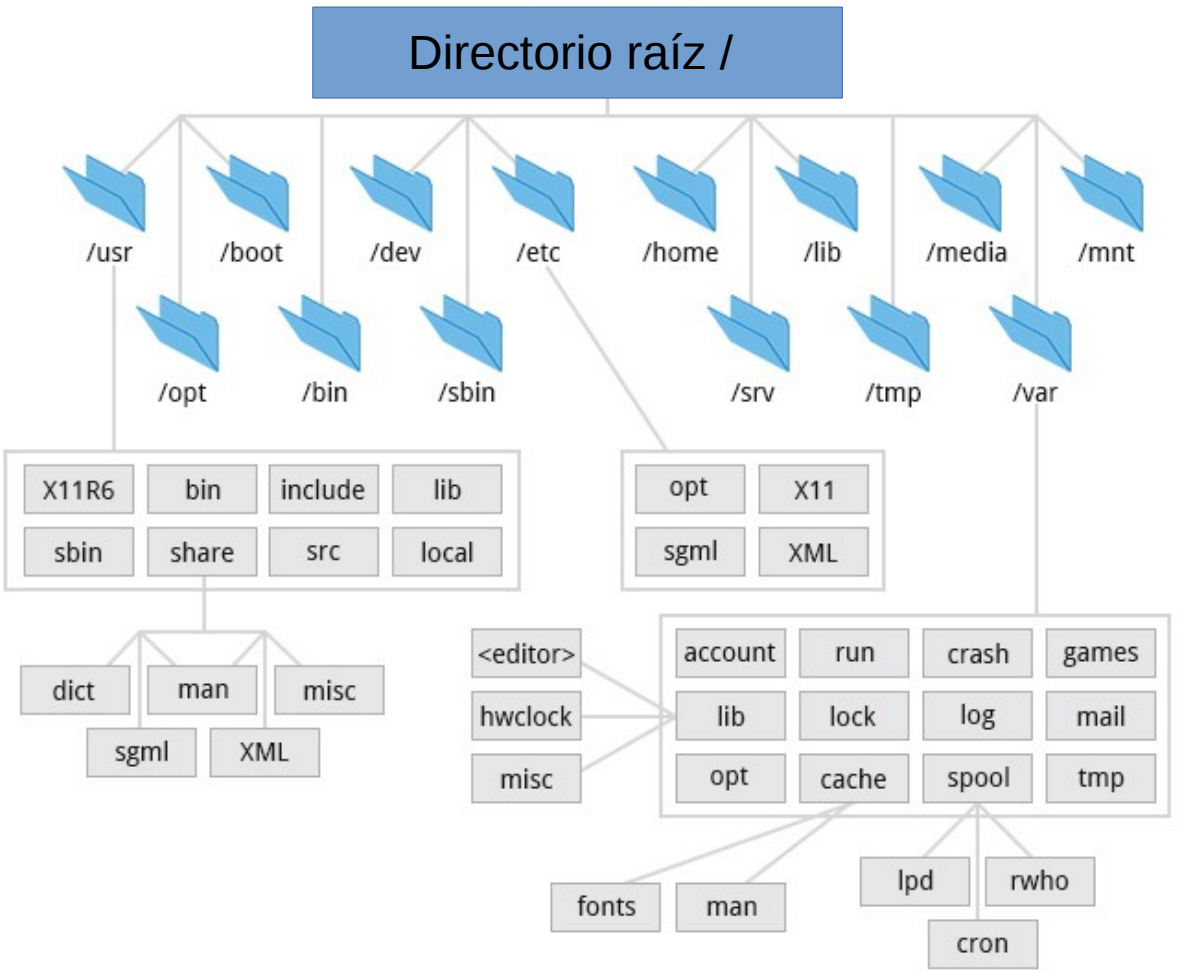
- Una distribución Linux esta compuesta por **Kernel** + **sistema GNU** (shell bash incluido) + **otros programas**
- Caulquier distribución Linux es un sistema “**tipo UNIX**“
- En el sistema tipo UNIX existen 2 tipos de archiso de **texto** y **binarios**
- El interprete de comandos o shell es el programa que sirve de interface entre el **usuario** y el **kernel**
- Al ejecutarse cualquier programa se conectan 3 canales de comunicación **salida estándar(1)**, **error estándar(2)** y **entrada estándar**
- Al terminar la ejecución de un programa éste reporta un **estado de salida,** 0 si la terminación fue normal, otro valor en caso contrario (significado en base a la documentación del programa)
- Existen **carácteres de control** para bash, estos pueden ser imprimibles en pantalla o no.

Caracter de escape \

Para indicar al shell que un carácter no debe ser interpretado como de control, es necesario “escaparlo“ anteponiendo el carácter barra invertida o carácter de escape: - \ -

```
echo  anuncio ;  date

echo anuncio \; date

echo ;


echo “mensaje“


Escapar caracter enter
```

Estructura del sistema de archivos



Rutas absolutas y rutas relativas

Para indicar la ubicación en disco de un archivo hay que indicar la lista completa de directorios que contienen a el archivo. En esta lista, que denominaremos **ruta de acceso**, cada directorio está separado del siguiente directorio por el signo / y sin dejar espacios en blanco.

Por ejemplo:

/usr/bin/wc

Para facilitar las cosas existen ciertos directorios especiales:

/ Es el directorio raíz. El superior de todos.

. Es el directorio actual. En el que nos encontramos en un momento dado.

.. Es el directorio superior, o directorio padre, al que nos encontramos.

El único directorio que no tiene directorio superior es el directorio raíz.

Rutas absolutas y rutas relativas

Una ruta absoluta es aquella que parte del directorio raíz. Las rutas absolutas son válidas en cualquier caso. Ejemplo:

```
/home/usuario/.bashrc
/home/soporte/Descargas/YouTube\ Audio\ \(MP3\)/
"/home/soporte/Descargas/YouTube Audio (MP3)"
```

Rutas absolutas y rutas relativas

Ruta relativa.

Es una ruta que parte del directorio actual como origen. Esta ruta sólo es válida desde un directorio actual concreto , es decir es relativa a un Directorio. Ejemplos:

```
cat ../../.bash_profile
nl ../etc/passwd
```

En este caso estamos haciendo referencia al archivo .bash_profile que está dos directorios por encima del directorio actual.

Comandos, programas, subrutinas (librerias)

En terminos del shell un comando puede ser:

- Un alias
- Una función
- Comando interno
- Un archivo en la ruta de directorios definidos en la variable PATH

Si el archivo no esta en formato ejecutable “**ELF**“ (SVR4 Executable and Linkable Format), el shell asume que se trata de un script y comienza a ejecutar línea por línea

Si el script comienza con los caractres **#!** el shell invoca al programa correspondiente para interpretar dicho script

Comandos, programas, subrutinas (librerias)

Los directorios estándar para almacenar archivos ejecutables (llamados también comando o programas) en un sistema tipo UNIX son:

/bin

/usr/bin

/usr/local/bin

/sbin

/usr/sbin

/usr/local/sbin

Estos directorios son almacenados en una varible que la shell consulta para la busqueda de archivos que coincidan con la instrucción escritra por el usuario.

El comand **which** nos reporta la ruta del program a ejecutar

Comandos, programas, subrutinas (bibliotecas)

Los directorios estándar para almacenar archivos con subrutinas o llamadas al sistema (conocidas como biliotecas) en un sistema tipo UNIX son:

/lib

/usr/lib

/usr/local/lib

/lib64

/usr/lib64

/usr/local/lib64

Estos archivos son de formato ELF (pero no del tipo ejecutable si no del tipo código objeto y biblioteca compartida)

Los comandos file, **objdump** pueden ayudarnos a investigar el tipo de archivo.

Documentación en el sistema.

```
info ls
```

```
man man
```

Usuarios, grupos y permisos

En sistemas tipo UNIX, sistemas **multi-usuarios**, se lleva el control de quién creo un archivo y quién es permitido accesarlo posteriomente.

Cada usuario tiene asignado un identificador único User ID así como un nombre de usuario

UID → número

Nombre usuario → cadena de caracteres

Un usuario que crea un archivo es dueño de ese archivo y puede asignar **bits de acceso** o **permisos** para diferentes propositos

Adicionalmente exite la figura de grupo, grupo de ususarios, como un mecanismo de administración general del sistema (internos, externos, colaboradores, alumnos, etc. son grupos definidos en el LSVP).

GUID → número

Grupo → cadena de caracteres

Usuarios, grupos y permisos

Bits de acceso o permisos.

001 - x - 1: Ejecución

010 - w - 2: Escritura

100 - r - 4: Lectura

Estos bits o permisos se establecen para el dueño del archivo, el grupo al cual pertenece el usuario y cualquier otro usuario.

Permisos → usuario,grupo,otros

Para establecer los permisos el comando chmod reconoce los formatos texto y octal:

chmod u=xwr,g=x,o=x → chmod 711

chmod u=rw,g=r,o=r → chomod 644

chmod u=rwx,g=,o= → chmod 700

Chmod a=x → chmod u=x,g=x,o=x → chmod 444

Obtener la cantidad de trabajos por usuario.

more / less / grep / awk / vim

Los programas paginadores permiten ver el contenido de un archivo de forma interactiva. Se recomienda el uso de “less“ sobre “more“

Para salir de un programa paginador, presionar la tecla q.

Less:

Presionar / durante la apertura de un archivo permite la busqueda de un patrón de caracteres.

N : nos lleva la siguiente coincidencia del patrón

n : nos lleva la anterior coincidencia del patrón

G : nos lleva al fin del archivo

g : nos lleva al inicio del archivo

&PATRON : solo muestra las líneas con PATRON

& : El PATRON es nulo, se muetran todas la líneas

-N : enumera las líneas

more / less / grep / awk / vim

Global search Regular Expression and Print: grep.

Copia en STDOUT las líneas que contienen el patrón especificado

grep **pedro** archivo

Opciones más utilizadas.

-w, --word-regexp

-i, --ignore-case

-v, --invert-match

-c, --count

-q, --quiet, --silent

-e PATTERN, --regexp=PATTERN

Obtener la cantidad de trabajos en ejecución por usuario en el cluster

more / less / grep / awk / vim

Aho, Weinberger, Kernighan: AWK

Lenguaje para el procesamiento de archivos de texto

awk permite establecer patrones y acciones.

Se procesa una línea a la vez, cuando el patrón se encuentra se realiza la acción.

Cada acción se aplica a la línea que contiene el patrón.

Para procesar cada línea AWK la separa por campos (por default utiliza espacios en blanco o tabulador). Por ejemplo la siguiente línea contien 3 campos:

2016-09-13T20:43:36 160 33195822407

El primer campo es referenciado como $1, el segundo como $2, etc.

awk ‘**programa**‘ archivos a procesar

Programa → patrón {acción}

awk ‘{print}‘ archivo

more / less / grep / awk / vim

Aho, Weinberger, Kernighan: AWK

```
awk ‘{print $0}‘ archivo
awk ‘{print $1}‘ archivo
awk ‘{print $0,$1}‘ archivo
awk ‘{print $2 * $3}‘ archivo
awk ‘ $3 > 0 { print $2 * $3}‘ archivo
awk ‘ $3 == 0 {print $0}‘ archivo
awk ‘ $1 == “pedro“ {print $0}‘ archivo
awk ‘ /pedro/ {print $0}‘ archivo
awk ‘{SUMA += $2; print SUMA}‘
awk ‘{SUMA += $2} END {print SUMA}‘
awk ‘BEGIN {print “-- TOTALES --“} {SUMA += $2} END {print SUMA}‘
```

Editor VIM

Tiene 2 modos de funcionamiento:

- **Edición**
- **Comando**

Al iniciar el editor vi se encuentra en modo comando, para alternar al modo edición presionar la tecla i.

Para cambiar a modo comando presionar la tecla de Esc.

n dd borra n líneas a partir de la línea actual

x borra un carácter

yy copia una línea

p pega una línea

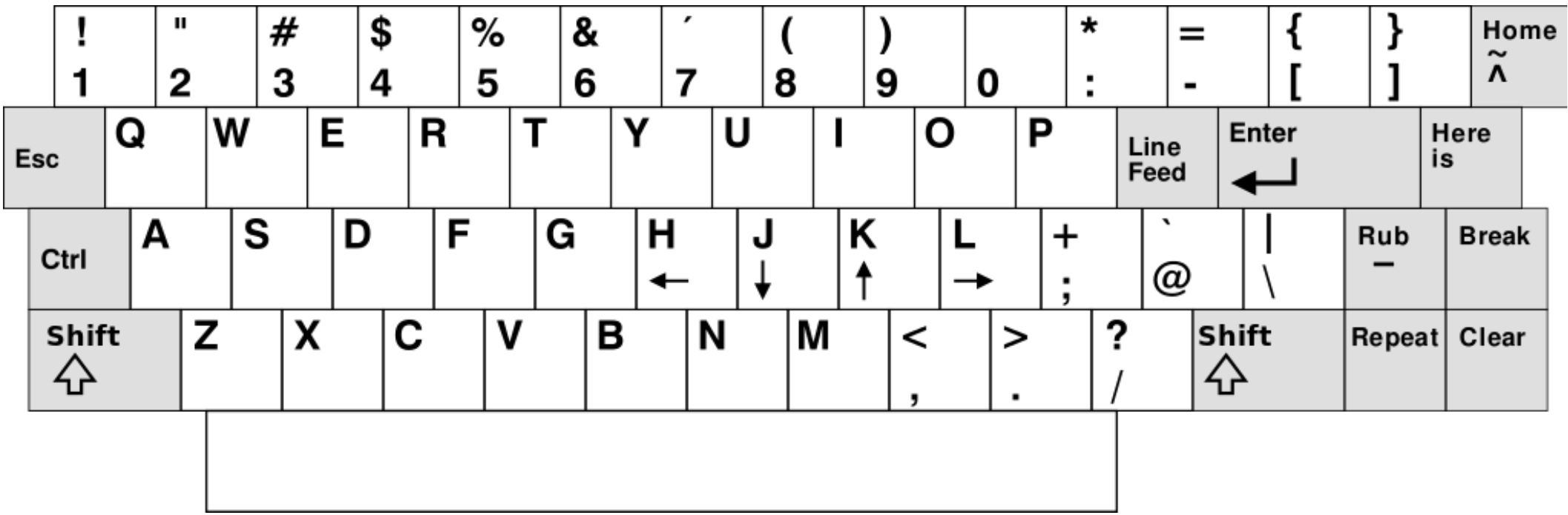
:n movernos a la líena n en el archivo

:w escribir el archivo

:q terminar la ejecución de vim

:q! terminar la ejecución de vim sin guardar cambios


ADM
Information Display


! 1
" 2
3
$ 4
% 5
& 6
´ 7
(8
) 9
0
* :
= -
{ [
}]
Home ~ ^
Esc
Q
W
E
R
T
Y
U
I
O
P
Line Feed
Enter
Here is
Ctrl
A
S
D
F
G
H
J
K
L
+ ;
` @
| \
Rub
Break
Shift
Z
X
C
V
B
N
M
< ,
> .
? /
Shift
Repeat
Clear

1. El comando **squeue -l** reporta en STDOUT los trabajos asignados por el programa administrador de recursos SLURM (https://slurm.schedmd.com/overview.html).

2. La información reportada son caracteres **ASCCI** oredenados en columnas

- Número de trabajo (**JOBID**)

- **PARTITION**

- Nombre del trabajo (**NAME**)

- Usuario dueño del trabajo (**USER**)

- Estado del trabajo (**STATE**)

- Tiempo en ejecución del trabajo (**TIME**)

- Tiempo máximo de ejecución permitido (**TIME_LIMI**)

- Número de servidores (nodos de cómputo) asignados al trabajo (**NODES**)

- Lista de los servidores (nodos de cópmuto) asignados al tabajo (**NODELIST(REASON)**)

3. Hay columnas (o campos) que nos interan extraer de la “salida“ del comando squeue

1. El comando **squeue -l** reporta en STDOUT los trabajos asignados por el programa administrador de recursos SLURM (https://slurm.schedmd.com/overview.html).

2. La información reportada son caracteres **ASCCI** oredenados en columnas

- Número de trabajo (**JOBID**)
- **PARTITION**
- Nombre del trabajo (**NAME**)
- Usuario dueño del trabajo (**USER**)
- Estado del trabajo (**STATE**)
- Tiempo en ejecución del trabajo (**TIME**)
- Tiempo máximo de ejecución permitido (**TIME_LIMI**)
- Número de servidores (nodos de cómputo) asignados al trabajo (**NODES**)
- Lista de los servidores (nodos de cópmuto) asignados al tabajo (**NODELIST(REASON)**)

3. Hay columnas (o campos) que nos interan extraer de la “salida“ del comando squeue

Obtener la lista de los usuarios con trabajos en el sistema SLURM

Por cada usuario en la lista del punto anterior, obtener sus trabajos:

- Los trabajos en estado de ejecución **RUNNING**
- Los trabajos en estado de espera de recursos **PENDING**

¿Cuantos nodos tiene asignado cada usuario?

¿Cuanto tiempo lleva utilizando los recursos cada usuario?

¿Es posible enviar esta información por correo cada hora?