

Tutorial de Unix

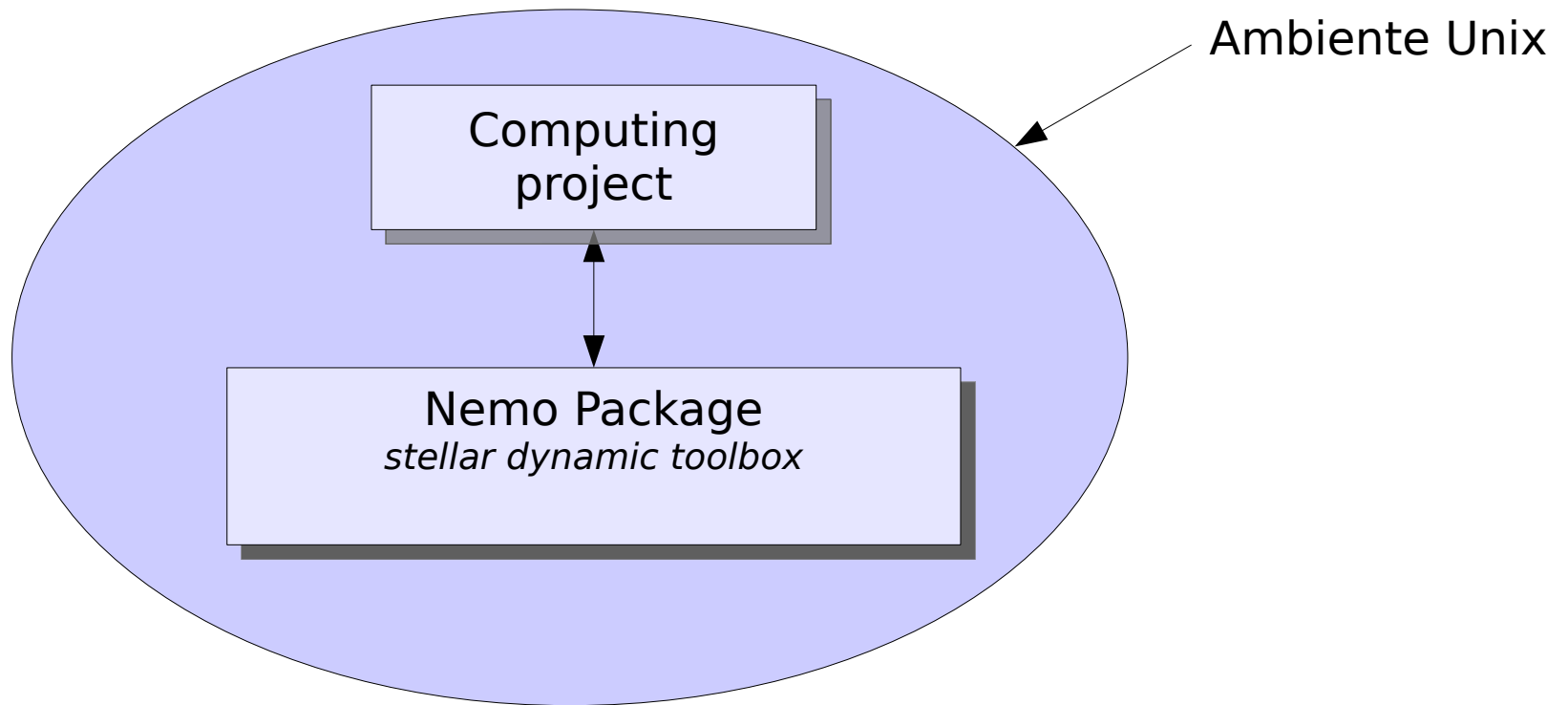
Mario A. Rodríguez-Meza
Instituto Nacional de Investigaciones Nucleares

Plan

- Introducción
- Login y ambiente gráfico
- El Shell de Unix
- Archivos y directorios
- Redireccionando la E/S (I/O)
- Pipelines y filtros
- Control de procesos

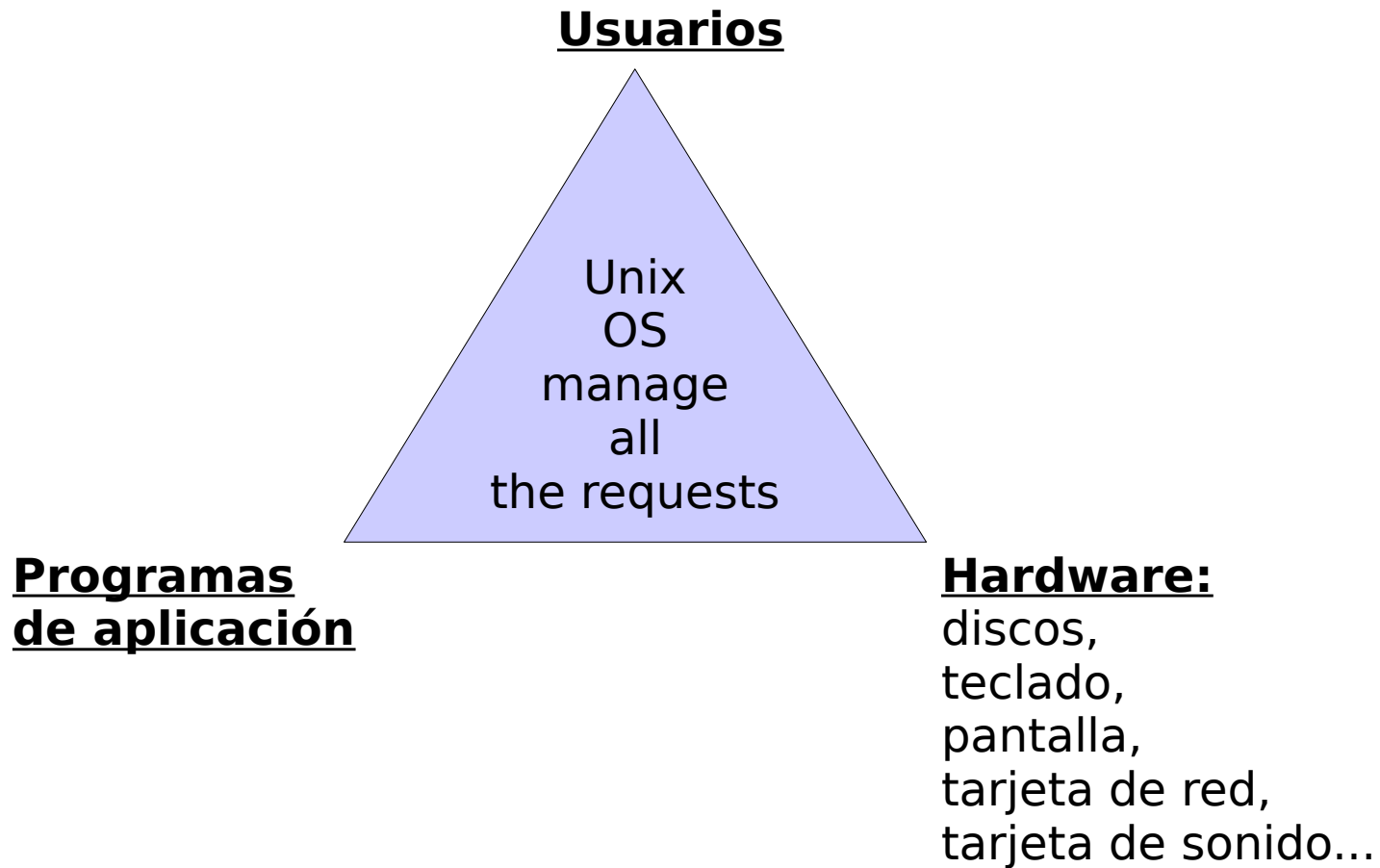
Introducción

- ¿Porqué un tutorial sobre Unix?



¿Qué es Unix?

- Un Sistema operativo como DOS o Windows XP



¿Qué es Unix?

- Inventado por AT&T Bell Labs al final de los 60's
- Características de Unix:
 - multi usuario
 - multi tarea
 - Confiable
 - Libre de virus hasta ahora

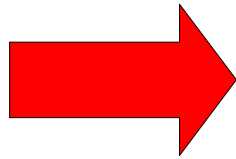
¿Qué es Unix?

- Implementación propietaria:
 - Solaris (Sun Microsystem)
 - HP UX / Digital Unix (HP)
 - Aix (IBM)
 - Mac OS X (Apple)
- Implementación libre: **Linux** (+100)
- Creador **Linus Torvald** - 1991
 - Mandriva (Mandriva)
 - Fedora core (RedHat)
 - OpenSuse (Novel)
 - Ubuntu, etc....

Login y ambiente gráfico

login

- Unix: un sistema de cuentas de usuarios
 - Seguridad: previene que personas no autorizadas entren al sistema
 - Provee de recursos (software) a cada usuario



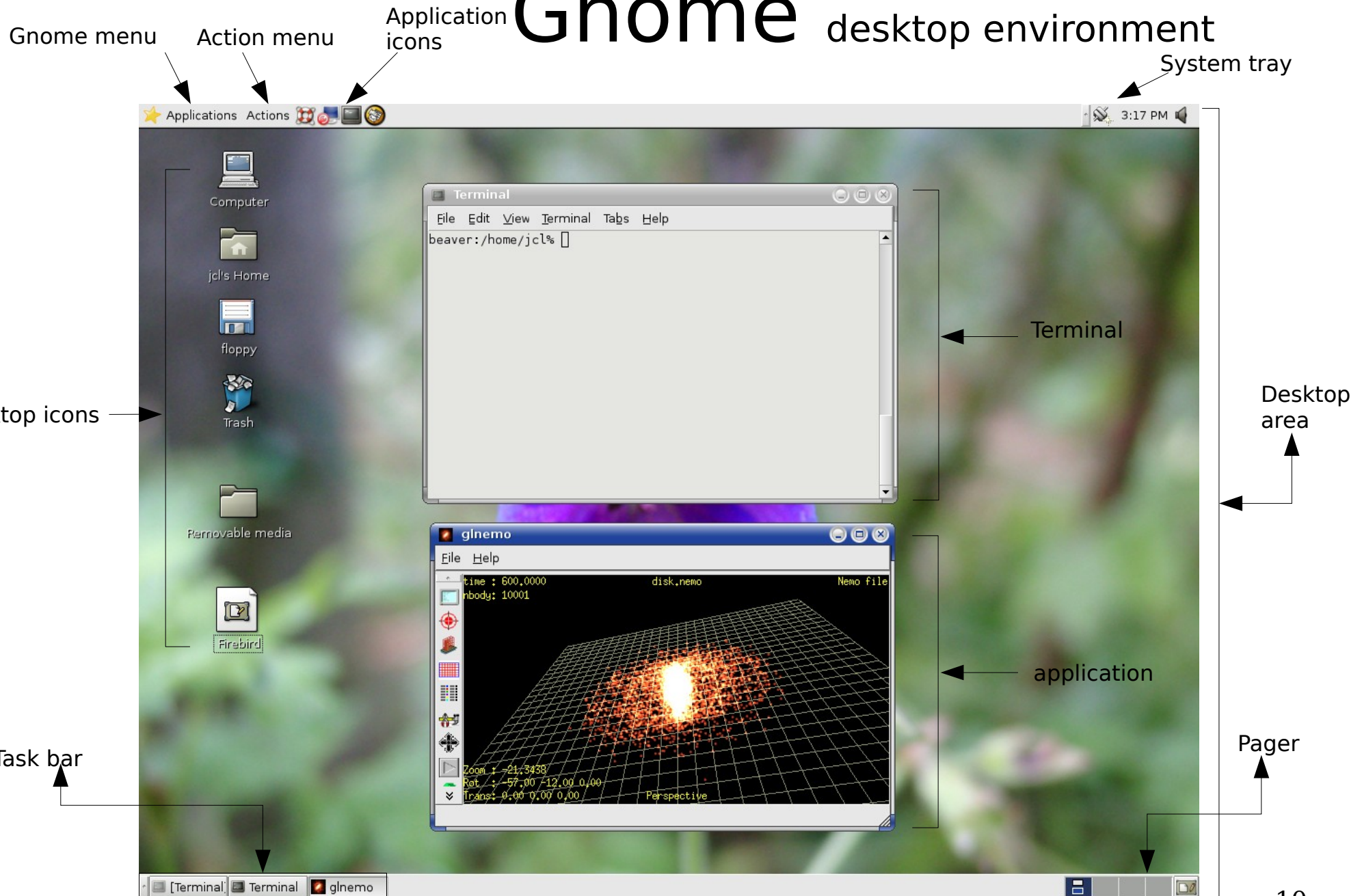
Mecanismo de control de acceso

- Logging in (conectandose):
 - Nombre de usuario (user name)
 - Clave (Password) (oculta)

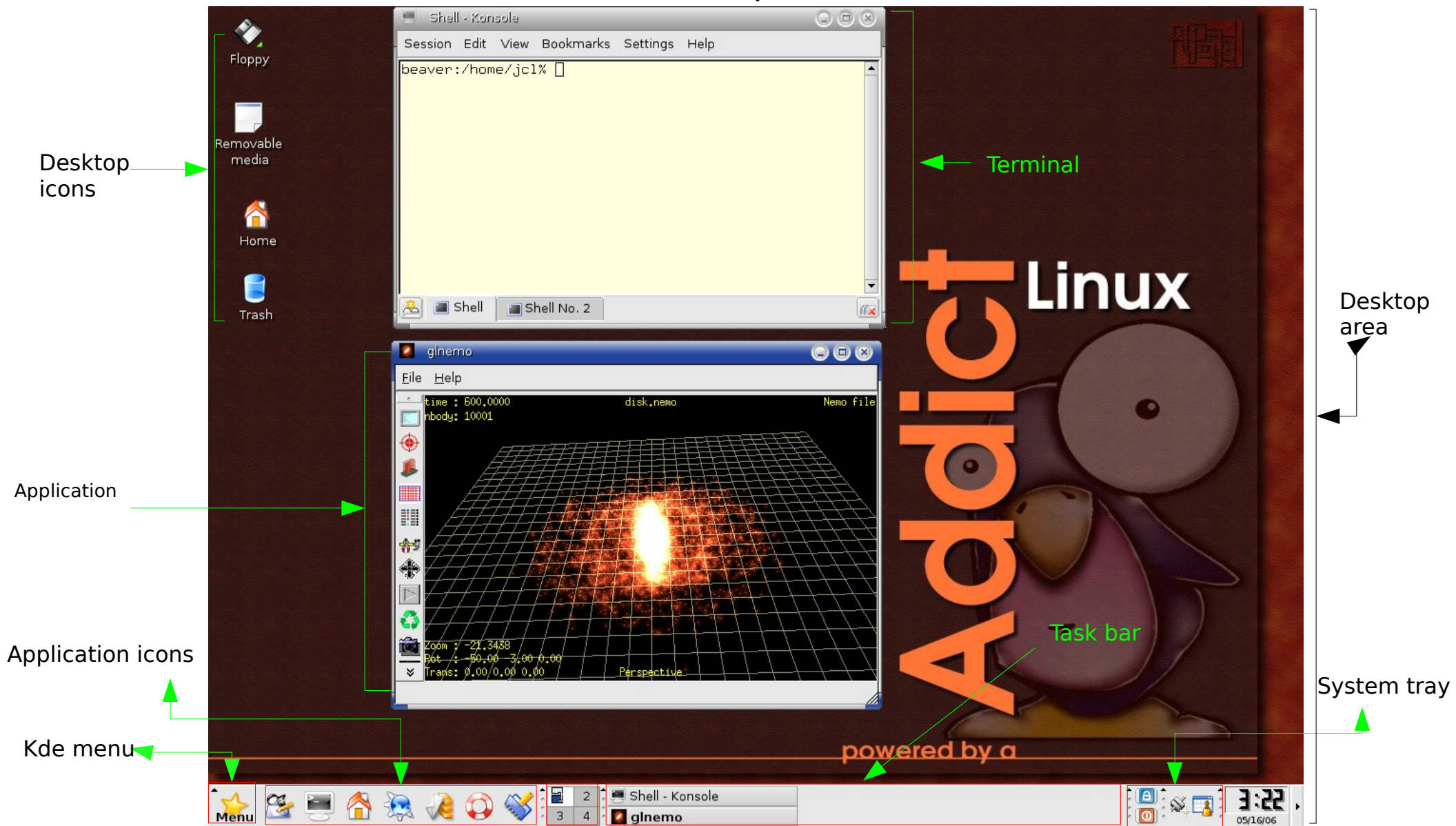
Ambiente gráfico

- Ambiente gráfico parecido a windows XP o Mac OS X
 - El usuario puede interactuar fácilmente con las aplicaciones
 - Manejadores de ventanas + un ambiente de escritorio (desktop)
- Ambiente de ventanas (sistema X windows or X11):
 - Ambiente para correr y manejar aplicaciones con una interface gráfica de usuario (GUI)
- Ambiente de escritorio (Desktop):
 - GUI similar para todas las aplicaciones
 - Gnome
 - KDE

Gnome desktop environment



KDE desktop environment



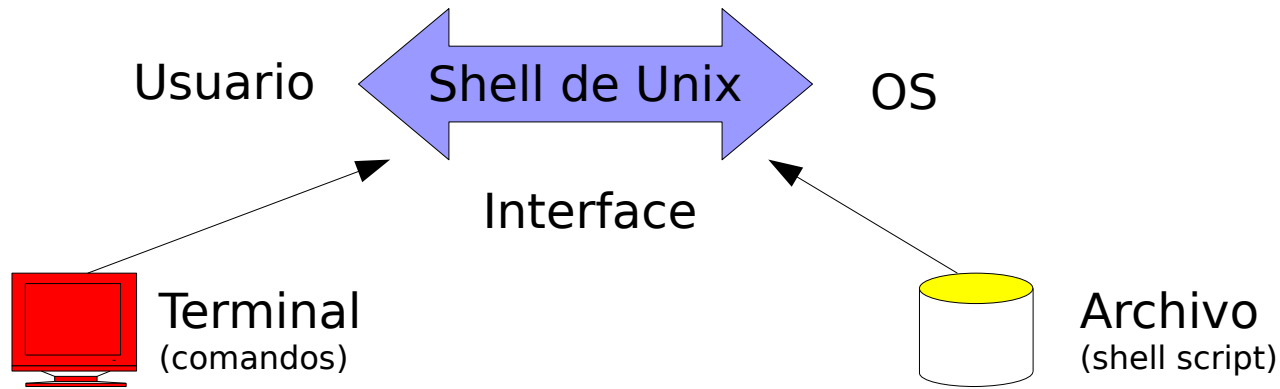
Taller de CAMB, ININ, 18 de febrero de 2011

Pager

El shell de Unix

El shell de Unix

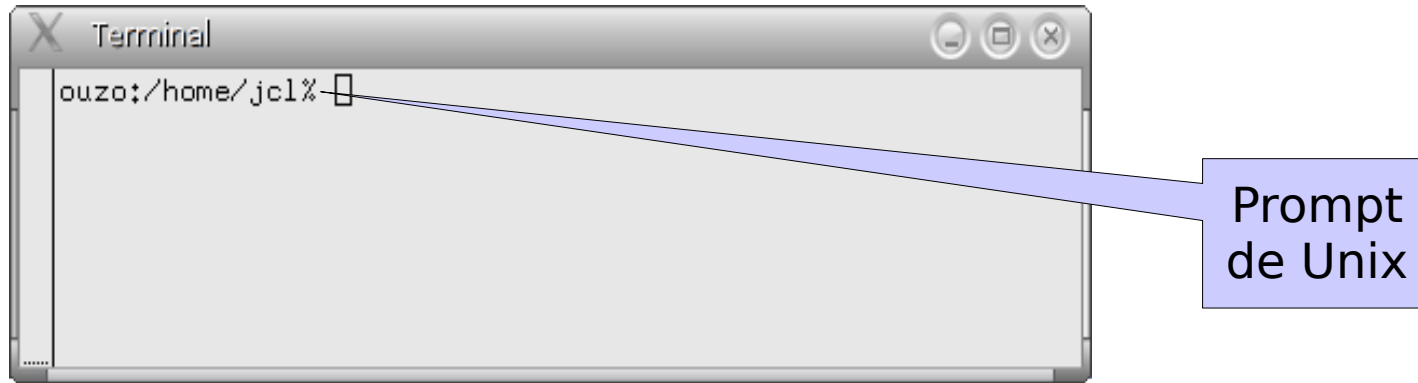
- Tanto un language de comandos como de programación



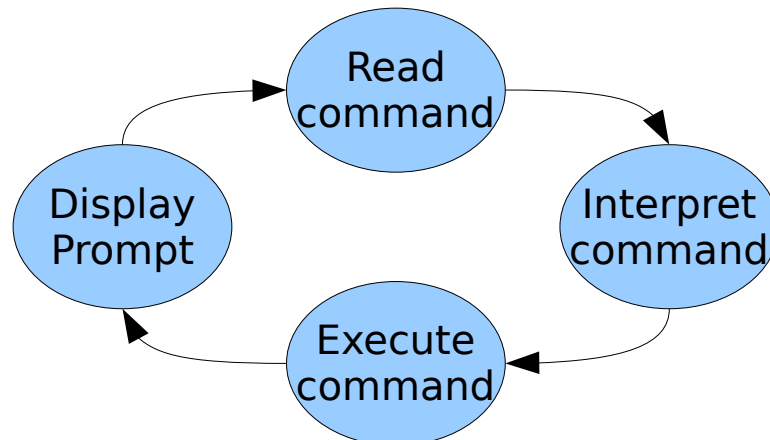
- Manejo de archivos y directorios
- Correr programas
- Checar procesos

Interprete de comandos

- Para tener acceso al shell de Unix comience una terminal
 - Terminal, Xterm, konsole etc....



- El shell es un programa conocido como un interprete:



Intérprete de comandos

- Ejecutando un comando en el shell:

La forma básica de un comando de unix es:

commandname [-options] [arguments]

Ejemplo:

ls -l /tmp

Da una lista detallada del directorio /tmp

- Abortando la ejecución de un comando en el shell:

para abortar comando en ejecución, tecleé : **Control+C**

- Obteniendo ayuda en Unix

Para ver los manuales en línea, use el comando **man** seguido por el nombre del comando del cual necesitamos la ayuda

Ejemplo:

man ls

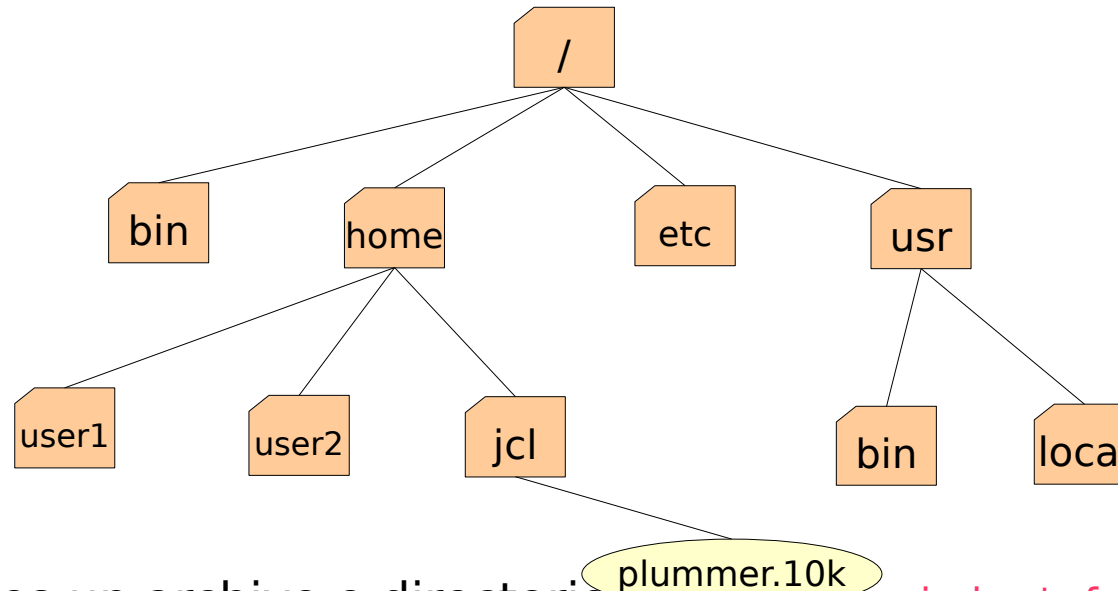
Atajos en Shell

- Completando con el uso de Tab (tcsh and bash)
 - Tecleé el comienzo del nombre de un archivo/directorio/comando, pulse <tab>, y el shell completará el nombre. Pulse <ctrl+d> para obtener todas las posibilidades para completar
- Historia de la ejecución de comandos
 - use up/down-arrow para circular por todos los comandos ejecutados previamente
- Uso de caracteres especiales
 - Caracteres especiales que pueden ser usados para ajustar los nombres de archivos o directorios
 - * Cero o más caracteres
 - ? Un carácter exactamente
 - Ejemplos:
 - `ls *.txt`
 - `ls data.0?.in`

Archivos y directorios

El sistema de archivos de Unix

- Jerarquía tipo árbol invertido



- Cada item es un archivo o directorio (Directory = window's folder)
- Cada directorio puede contener archivos y directorios
- Cada archivo o directorio está especificado con una trayectoria (PATH) comenzando desde "/" (la raíz o root),

Ejemplo: /home/jcl/plummer.10k

- Hay un directorio especial que pertenece al usuario para que coloque sus archivos y directorios (\$HOME): /home/username

Control de acceso -1-

- En UNIX, los archivos y directorios tienen **control de acceso**:

Permisos de Acceso:

- Read r
- Write w
- Execute x

aplica



Pertenencia:

- User u
- Group g
- Other o

• **Derechos de acceso sobre los archivos**

- r Indica permiso de lectura: para leer y copiar el archivo
- w indica permiso para escribir: para cambiar un archivo
- x Indica permiso para ejecución: para correr un archivo

• **Derechos de acceso sobre los directorios**

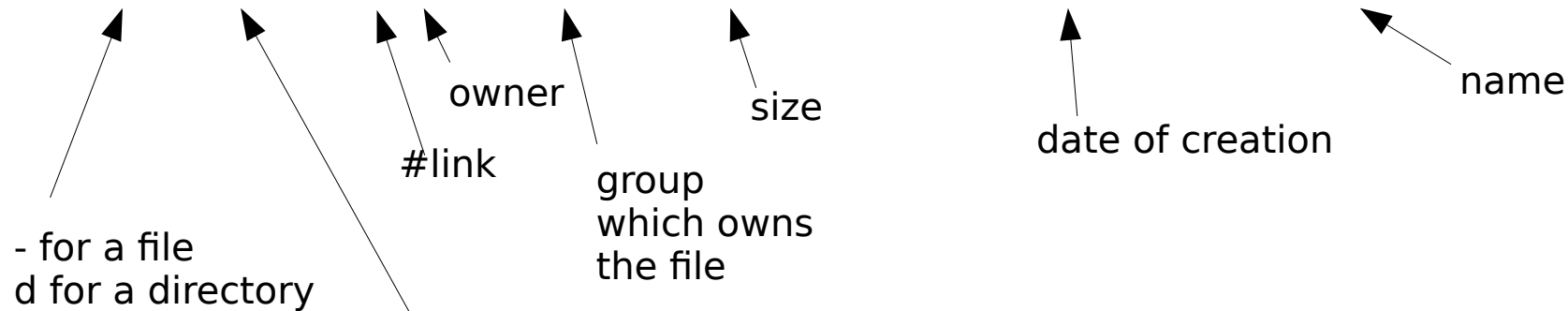
- r Permite a los usuarios obtener una lista de archivos en un directorio
- w Significa que los usuarios pueden borrar archivos del directorio o mover archivos dentro de él
- x significa el derecho para acceder archivos en el directorio

Control de acceso -2-

- Ejemplo :
Para ver el permiso de un archivo, usamos el comando **ls -l**:

% ls -l plumber.10k

-	r	w	-	r	-	-	1	j	c	l	dynam	560269	May 3 15:43	plumber.10k
---	---	---	---	---	---	---	---	---	---	---	-------	--------	-------------	-------------



9 symbols for access permissions:

rw- : read and write for the owner

r-- : read for the group

r-- : read for all the others

Control de acceso -3-

- El cambio de los permisos puede hacerse con el comando **chmod** **[options]** **file**
[options] es una combinación: **ownership**(**ugo**)**action**(**+ -=**)**permissions**(**rwX**)

- **Ejemplo 1:**

```
% ls -l plummer.10k
-rw-r--r-- 1 jcl dynam 560269 May  3 15:43 plummer.10k

% chmod go+rw plummer.10k
% ls -l plummer.10k
-rw-rw-rw- 1 jcl dynam 560269 May  3 15:43 plummer.10k
```

Este comando **add** permisos de lectura y escritura para los grupos “**group**” y “**others**”

- **Ejemplo 2:**

```
% chmod o-rw plummer.10k
% ls -l plummer.10k
-rw-rw---- 1 jcl dynam 560269 May  3 15:43 plummer.10k
```

Este comando **remueve** permisos de lectura y escritura para el grupo “**others**”

Manipulando archivos y directorios -1-

- Trabajando con un directorio
 - El comando **pwd** regresa el nombre del directorio de trabajo en donde estamos
 - El comando **cd** cambia al directorio dado en el argumento. Sin argumento nos cambia al directorio HOME
 - El comando **cd ..** te mueve arriba un nivel en el árbol de directorios. “..” es conocido como el directorio “parent”
 - El tilde **~** representa el directorio “home”, **~username** representa el directorio “home” de “username”
- Creando/removiendo un directorio
 - El comando **mkdir** crea un directorio
 - El comando **rmdir** remueve un directorio

```
% pwd
/home/mar
% mkdir bin
% cd bin
% pwd
/home/mar/bin
% cd ~mar
% pwd
/home/mar
```

Recuerde que:

El comando **man** nos da la ayuda en-línea sobre un comando
-> man mkdir

Manipulando archivos y directorios -2-

- Copiando archivos

- **cp** *source destination* => copy *source* file to *destination*
 - if *destination* does not exist, *source* will be duplicated into *destination*
 - if *destination* is a directory, *source* will be copied into *destination*
 - if *destination* is an existing file, it will be overwritten

- Copiando directorios

- **cp -r** *source destination* => copy *source* directory to *destination*
 - if *destination* does not exist, *source* will be duplicated
 - if *destination* is a directory, *source* will be copied into *destination*
 - if *destination* is an existing file, it will be overwritten

- Ejemplo

```
% cp plumber.10k tmp/  
% ls tmp  
plumber.10k  
% cp -r bin tmp  
% cp plumber.10k tmp/qq  
% ls -F tmp  
bin/ plumber.10k qq
```

Recuerde que:

El comando **man** nos da la ayuda en-línea sobre el comando

-> man cp

Manipulando archivos y directorios -3-

- Moviendo y renombrando archivos/directorios
 - **mv** *source destination* => move *source* file/directory to *destination*
if *destination* does not exist, *source* will be rename into *destination*
if *destination* is a directory, *source* will be moved into *destination*
if *destination* exist, *source* will be rename into *destination* which will be overwritten
- Removiendo archivos
 - **rm** *file* => *remove existing file*
- Ejemplo

```
% mv plumber.10k tmp  
% mv tmp tmp_new  
% rm tmp_new/plumber.10k
```

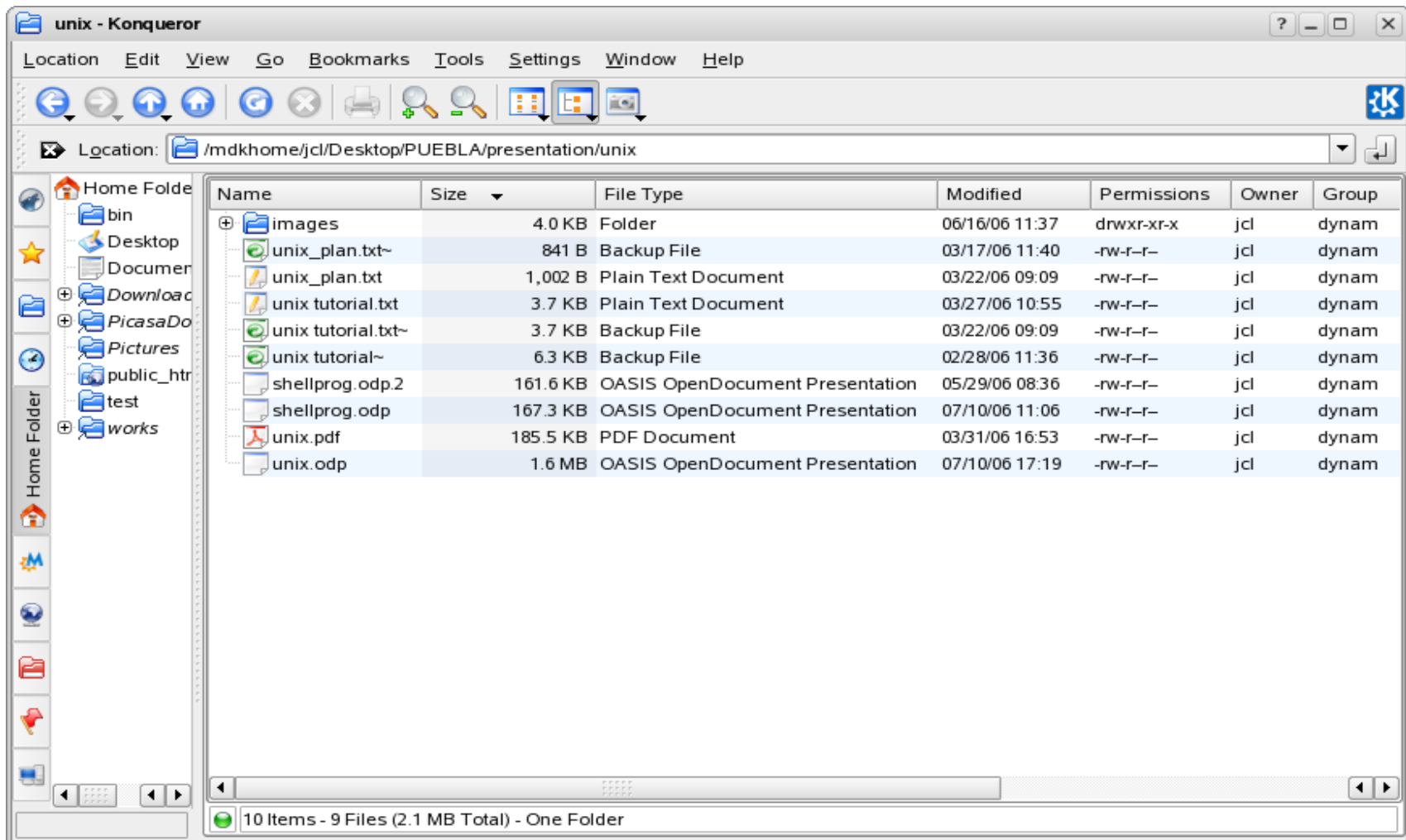
Recuerde que:

El comando **man** nos da la ayuda en-línea sobre un comando

-> man mv

Manipulando archivos y directorios -4-

- Usando el ambiente del “Desktop”: -konqueror file manager-



Comando útiles

- **cat** [options] filename

Muestra en la pantalla el contenido de una archivo

- **less** [options] filename

Muestra el contenido de un archivo, se puede usar las flechas “up/down” para deslizarse hacia arriba o hacia abajo

- **wc** [options] filename

(word count) regresa cuantas líneas, palabras y caracteres hay en un archivo

- **gzip** [options] filename

Comprime un archivo

- **lp -d printer_queue** filename

Imprime el archivo “filename” usando la impresora “printer_queue”

- **du** [options] filename

Regresa el tamaño de un archivo y/o directorio

Redireccionando la entrada y la salida (I/O)

Redirección

- En Unix, cualquier programa abre 3 archivos
 - **Standard input:**
 - Proveé una forma de enviar datos a el programa
 - Por defecto la entrada es leida de el teclado
 - **Standard output:**
 - Proveé una forma para la salida de datos del programa
 - Por defecto la salida es la pantalla
 - **Standard error**
 - Proveé una forma para que los programas escriban mensajes de error
 - Por defecto la salida de error es la pantalla

Redireccionando la entrada (input) <

- Usamos el símbolo < para redirigir la entrada a un comando
- La sintaxis usual es: *command < filename*
- Ejemplo: El comando *sort* ordena alfabeticamente o numéricamente una lista. Sort está esperando datos de la entrada estándar (the standard input).

```
% cat usersfile
peter
jcl
lia
% sort < usersfile
jcl
lia
peter
```

Redireccionando la salida (output) >

- Usamos el símbolo > para redirigir la salida de un comando
- La sintaxis usual es: *command > filename*
- Ejemplo: El comando **sort** ordena alfabeticamente o numéricamente una lista. Sort está esperando datos de la entrada estándar.

```
% cat usersfile
yaya
mar
tmatos
% sort < usersfile > usersfile.sorted
% cat usersfile.sorted
mar
tmatos
yaya
```

Redireccionando la salida >>

- Usamos el símbolo >> para **apendizar** la salida estándar a un archivo
- La sintaxis usual es: *command >> filename*
- Ejemplo:

```
% cat usersfile >> usersfile.sorted
% cat usersfile.sorted
mar
tmatos
yaya
yaya
mar
tmatos
```

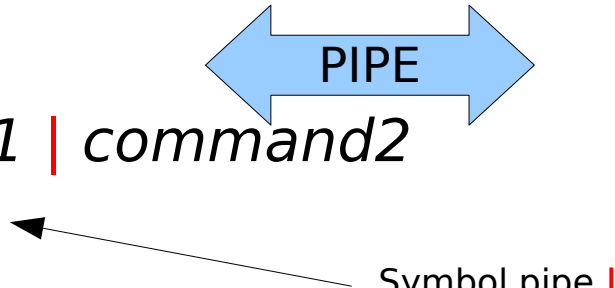
Redireccionando el error >&

- Depende de la clase de shell
 - **csh/tcsh** uses **>&** symbol
 - **bash/sh/ksh** uses **2>** symbol
- Ejemplo:

```
% cat usersfile /etc/shadow > file.stdo >& file.stde
% cat file.stdo
yaya
mar
tmatos
% cat file.stde
cat: /etc/shadow: Permission denied
```


Pipelines y filtros

Pipelines

- Pipelines son una forma de conectar procesos usando el mecanismo **pipe**
 - Process1 (standard output) (standard input) process2
- Sintaxis: *command1* | *command2*

Symbol pipe |
- El mecanismo Pipe es muy usado en la programación con unix o en su uso cotidiano
- Ejemplo: El comando **who**
 - 1st método usando redirección I/O:
 - % who > users.txt
 - % sort < users.txt
 - 2nd método usando el mecanismo pipe:
 - % who | sort

Filtros

- Los **Filtros** pueden leer la entrada de la entrada estándar, procesarla, y listar el resultado en la salida estándar
- El comando **grep** es uno de los filtros útiles en Unix:
 - Busca línea por línea por un patrón especificado
 - ... y regresa como salida cualquier línea que se ajusta al patrón
- Sintaxis de grep: `grep [-options] pattern [file]`
- Ejemplo:
 - `% grep 'students' /etc/passwd`
 - Busca en el archivo `/etc/passwd`, todas las líneas que contienen el patrón `'students'`

Recuerde que:

El comando **man** nos da la ayuda en-línea sobre un comando.

-> `man grep`

Control de procesos

Administración de procesos -1-

- Ambiente Multitasking: muchos procesos pueden correr al mismo tiempo
- Process ID : cada proceso que está corriendo tiene un identificador único llamado el PID
- Viendo procesos:
 - El comando **ps** reporta una instantanea de del estado de los procesos en ejecución
 - Sintaxis: **ps [options]**
 - Ejemplo:
 - % ps -eaf (da un listado completo de todos los procesos corriendo)
 - % ps -fu yaya (da un listado completo de todos los procesos de la yaya que están corriendo)

UID	PID	PPID	C	STIME	TTY	TIME	CMD
jcl	6331	6330	0	May09	tty1	00:00:00	-tcsh
jcl	6390	6331	0	May09	tty1	00:00:00	/bin/sh /usr/X11R6/bin/startx
jcl	6401	6390	0	May09	tty1	00:00:00	xinit /home/jcl/.xinitrc -- -deferglyphs 16
jcl	6437	6401	0	May09	tty1	00:00:00	sh /home/jcl/.xinitrc
jcl	6439	6437	0	May09	tty1	00:00:00	/bin/sh /usr/bin/startkde

Administración de procesos -2-

- Viendo procesos :

- El comando **top** reporta la dinámica en tiempo-real de un sistema corriendo

- ejemplo:

% top

top - 11:20:23 up 2 days, 53 min, 1 user, load average: 0.01, 0.04, 0.07

Tasks: 143 total, 2 running, 141 sleeping, 0 stopped, 0 zombie

Cpu(s): 3.8% us, 0.9% sy, 0.1% ni, 94.6% id, 0.3% wa, 0.3% hi, 0.0% si

Mem: 1555868k total, 1478448k used, 77420k free, 25120k buffers

Swap: 2096440k total, 2732k used, 2093708k free, 462992k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6601	yaya	15	0	348m	286m	21m	S	1.9	18.9	41:58.31	firefox-bin
1	root	16	0	1536	524	464	S	0.0	0.0	0:00.54	init
2	root	34	19	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
3	root	10	-5	0	0	0	S	0.0	0.0	0:01.83	events/0
4	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
9	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
21	root	10	-5	0	0	0	S	0.0	0.0	0:00.21	kblockd/0

Administración de procesos -3-

- Matando procesos:
 - Es algunas veces necesario terminar un proceso
 - Sólo el dueño del proceso puede terminarlo
 - El comando **kill** es usado para terminar un proceso
 - sintaxis: **kill [-options] PID**
 - Ejemplo: para forzar la terminación de un trabajo con PID=666, use la opción -9
% kill -9 666
 - Con el despliegue de la utileria top, es también posible acabar con un trabajo
 - Tan sólo presione la tecla 'k'

Trabajos en el “background” y en el “foreground” -1-

- 3 estatus para un proceso que está corriendo:
 - **Foreground:** el shell regresa el “prompt” de Unix mientras que el proceso actual está corriendo
 - **Background:** el “prompt” de Unix es regresado inmediatamente después de que se ha enviado al proceso al “background”
 - **Suspended:** el proceso en el “foreground” es detenido (paused)

Trabajos en el “background” y en el “foreground” -2-

- Corriendo procesos en el “background” [**&**]:
 - Para mandar a un proceso al “background”, agregue el símbolo **&** al final de la línea de comando
 - Ejemplo:

% sleep 10	(espera 10 seconds antes de regresar el “prompt” de comandos %)
% sleep 10 &	(corre sleep en el “background”)
[1] 23322	([1]=job number, 23322=running command's PID)

Trabajos en el “background” y en el “foreground” -3-

- Suspendiendo un trabajo en el “foreground”
 - Para suspender un trabajo en el “foreground” que está en este momento conectado a tú terminal, tecleé **CTRL-Z**
 - Un proceso suspendido no usa CPU
- Trayendo un trabajo del “background” a el “foreground”
 - Para reconectar un trabajo en el “background” tecleé **fg**

FIN ... ¡Gracias!