Sistemas Operativos 2014-2015

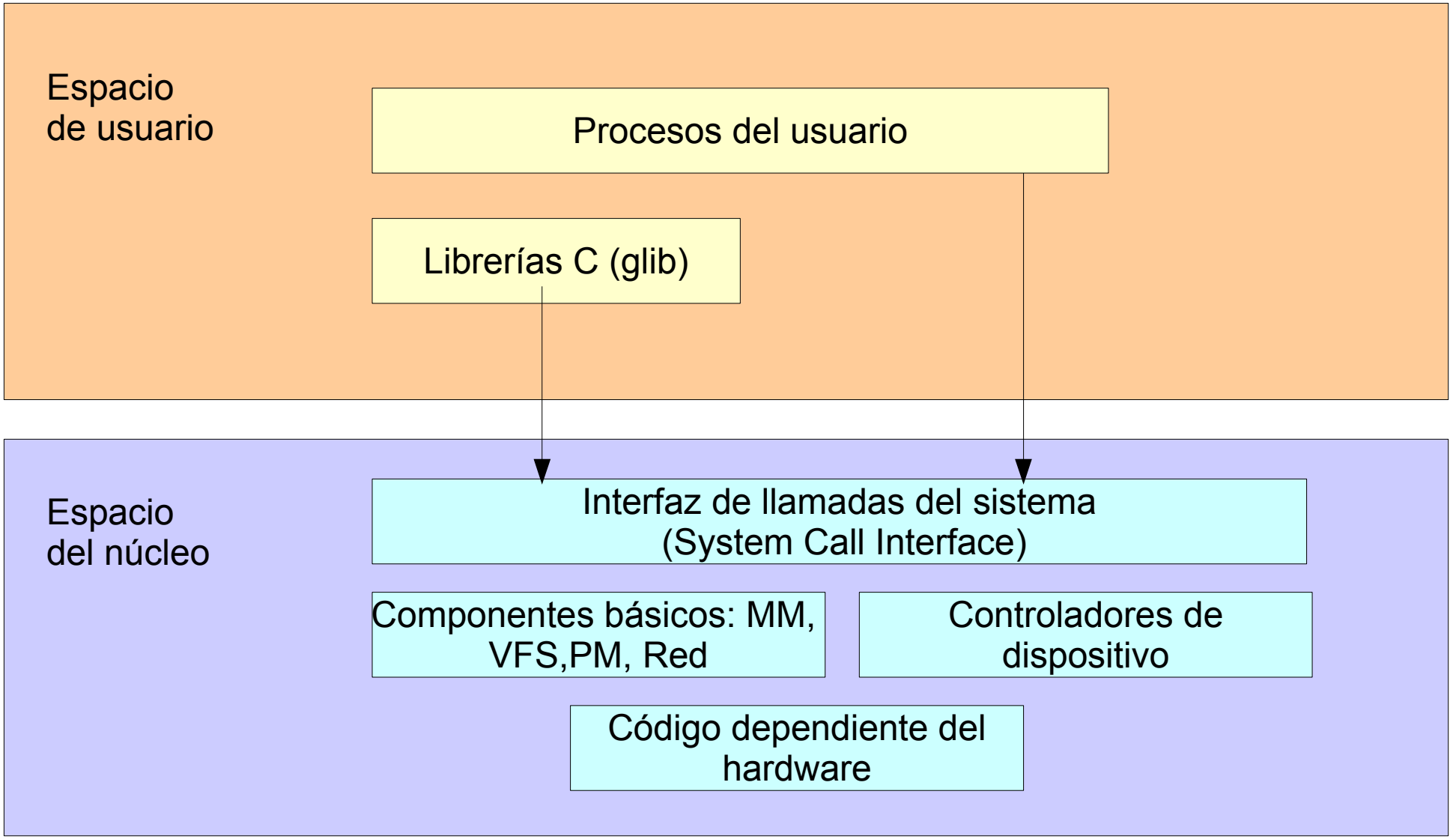
Introducción a línea de comandos en Linux

Contenidos

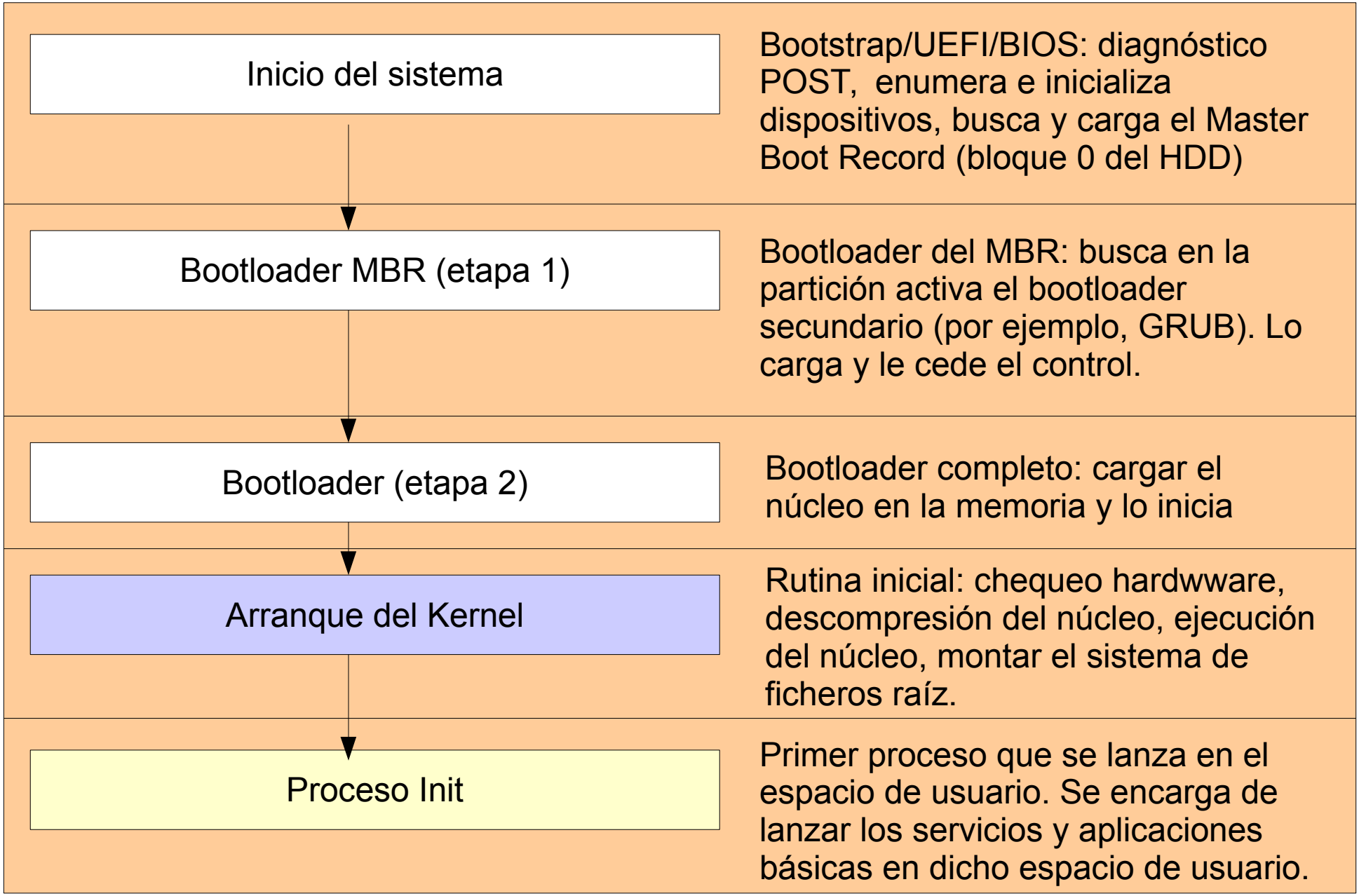
- Imagen general de Linux.
- Resumen del proceso de arranque del sistema.
- Imagen inicial de los procesos en Linux.
  - Jerarquía de procesos: hilos, procesos, grupos de procesos, sesiones
  - Terminal, consola Linux, el programa login y la shell.

- Autenticación y autorización en Linux
- Introducción al sistema de archivos. Comandos `cd`, `ls`, `mkdir`, `pwd`, `cat`.
- Discretionary Access Control (DAC). Comandos `chmod`, `chown` y `chgrp`.

Imagen general de GNU/Linux



El arranque (simplificado)



Arranque del sistema

1. Llega a la CPU una señal de RESET motivada por un encendido del sistema o por una reinicialización.
2. La CPU inicializa el contador de programa a una dirección predefinida de la memoria. En esa dirección está el *bootstrap* inicial.
   - Dicho programa debe estar almacenado en una memoria no volátil (ROM o Flash) por que la RAM está en un estado indeterminado en el momento del arranque.
   - En los PC el bootstrap forma parte del *firmware* (BIOS o UEFI) de las placas madres.

Arranque del sistema

3. El *bootstrap* debe realizar diversas tareas:

- **Diagnostico de la máquina** (POST o Power-on selft-test)
  El *bootstrap* se detiene en este punto si el sistema no supera el diagnostico.
- **Inicializar el sistema**. Por ejemplo los registros de la CPU, controladores de dispositivos, contenido de la memoria, etc.
- **Iniciar el sistema operativo**.

Arranque del sistema

4. Se inicia el sistema operativo:

- En **consolas de videojuegos, móviles y otros dispositivos empotrados** se almacena el sistema operativo en alguna forma de memoria de sólo lectura (ROM o Flash). Como la ejecución en esas memorias es más lenta que en la RAM muchas veces se suele copiar el sistema a la RAM durante el arranque.
- **En sistemas operativos de gran envergadura**, incluidos los de propósito general, el sistema se almacena en disco. El *bootstrap* lee de una posición fija del disco (generalmente el bloque 0, *donde en los PC se aloja el MBR o Master Boot Record)* el *gestor de arranque*, lo copia en memoria y lo ejecuta. El *gestor de arranque* es el programa que sabe como iniciar el sistema operativo así que: explora el sistema de ficheros en busca del núcleo del sistema, lo carga e inicia su ejecución.

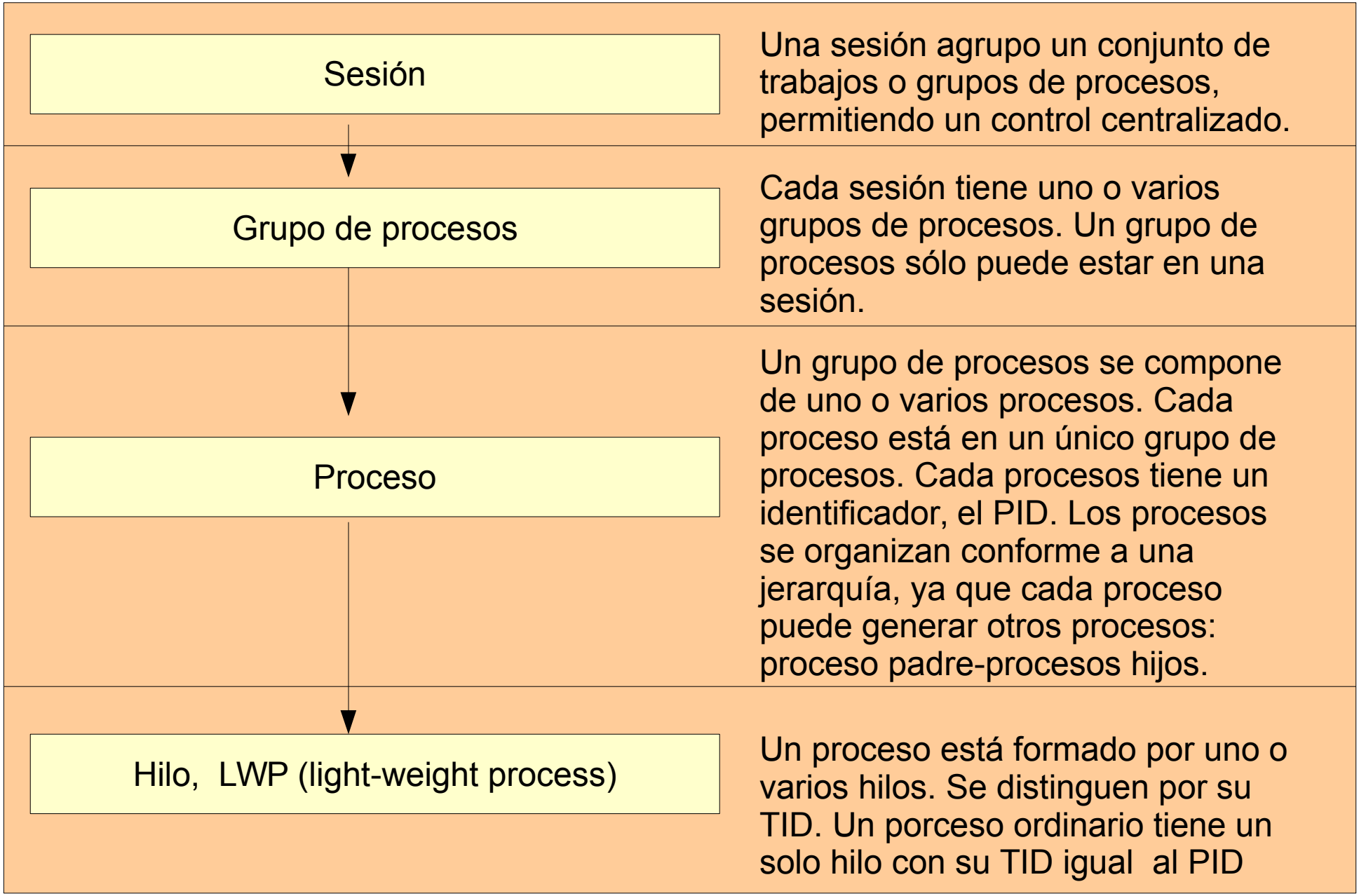
Arranque UNIX/Linux

5. Se inicia el núcleo del sistema operativo:
   - Se realiza la mayor parte de la configuración del sistema: interrupciones, gestión de memoria, inicialización de dispositivo y controladores, montaje del sistema de ficheros raíz, creación del proceso inactivo, etc.
   - Se crea el proceso *init* (PID 1) a partir de la carga del programa *init* almacenado en el sistema de ficheros raíz.
   - El planificador de la CPU toma el control efectivo de la gestión del sistema y el núcleo queda dormido. Puesto que la función del planificador es asignar procesos a la CPU, el proceso *init* es escogido y comienza su ejecución.
6. El proceso *init* lanza los scripts encargados de configurar los servicios (*demonios*) del sistema (p.ej. logger, gestión de dispositivos, impresión, etc.) y de crear el entorno de usuario.

Inicio de sesión en UNIX/Linux

7. El proceso *init* inicia un proceso *login* conectado a cada terminal del sistema. El proceso *init* monitoriza los procesos *login*, para reiniciarlos en caso de que alguno muera, y se duerme a la espera.
8. Los procesos *login* se encargan de autenticar a los usuarios y de iniciar su sesión:
   a) Muestran una pantalla de *inicio de sesión* donde se solicita el nombre del usuario y su contraseña.
   b) Autentican al usuario comprobando las credenciales proporcionadas por el mismo.
   c) Si la autenticación es positiva, el proceso *login* cambia su identidad actual (generalmente la de *root*) por la del usuario que quiere iniciar sesión y sustituye su programa actual por el del *intérprete de comandos* o *shell*.

Organización de los procesos



Las sesiones y los grupos de procesos

- Las sesiones son iniciadas por algún proceso. Este proceso se asocia a la nueva sesión como lider de un nuevo grupo de procesos (*gdp*), en principio formado por el mismo.
- En una sesión hay dos tipos de *gdps*. El gdp en primer plano (sólo puede haber uno) y los *gdps* en segundo plano.
- La interacción con la terminal de control queda asociada exclusivamente con el *gdp* en primer plano.

El proceso de login y la shell

- Una shell es un programa especial con el que el usuario puede interactuar perimitiéndole la ejecución de programas y scripts de comandos y dotándole de facilidades para automatizar estas tareas.
- Cuando en la consola o en un emulador de terminal hacemos login, el resultado final en caso de autenticarnos correctamente es el lanzamiento de una shell con una nueva sesión asociada.
- La terminal de control de la sesión es la terminal con la que interactuamos con la shell.

Autenticación y autorización

- Dos conceptos básicos bien diferenciados:
  - Autenticación: verificar la identidad del usuario. Usualmente lo hacemos con el password, pero existen otros medios.
  - Autorización: política de acceso a los recurso del sistema. ¿Quién puede hacer esto sobre este recurso?
- Así pues el login es una forma de autenticación que descansa sobre la tecnología necesaria para comprobar de forma segura que el usuario es quien dice ser.

Los usuarios en Linux

- Cada usuario tiene un UID (id de usuario único).
- Tipos de usuarios:
  - El usuario root: uid=0, acceso total.
  - Usuarios o cuentas del sistema: bin,daemon, adm, lp, tienene permisos especiales para tareas concretas, no tienen contraseña ya que no se inician sesiones de usuario, se suelen generar automáticamente al instalar linux o determinadas aplicaciones.
  - Usuarios normales. Tienen password y un directorio de trabajo (HOME). Suelen tener permisos completos en sus directorios de trabajo.

Autenticación UNIX tradicional (I)

- El fichero /etc/passwd contiene información sobre la cuenta del usuario. Lo puede leer cualquier usuario del sistema. Pueder ser escrito sólo por el root y los miembros del grupo.
- El fichero /etc/shadow contiene los passwords cifrados de los usuarios. Sólo el root y los miembros del grupo root pueden leer este fichero.
- Antiguamente la clave cifrada se guardaba también en el fichero /etc/passwd, lo cuál era un problema de seguridad.

Autenticación UNIX tradicional (II)

- Campos del fichero /etc/passwd:
  - Nombre de cuenta.
  - Contraseña cifrada o un asterisco.
  - Identificador de usuario (UID)
  - Identificador del grupo principal del usuario (GID)
  - GECOS. Información del usuario no usada por el sistema, como la dirección de la oficina el teléfono.
  - Ruta del directorio personal del usuario
  - Ruta al ejecutable de la shell que usa este usuario.

/etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
nacho:x:1000:1000:nacho,,,:/home/nacho:/bin/bash
messagebus:x:101:104::/var/run/dbus:/bin/false
haldaemon:x:102:105:Hardware abstraction layer,,,:/var/run/hald:/bin/false

Autenticación UNIX tradicional (IV)

- Campos del fichero /etc/shadow:
  - Nombre de cuenta.
  - Contraseña cifrada.
  - Días transcurridos desde el 01/01/70 hasta que se cambió la contraseña por última vez.
  - Número de días que deben transcurrir para que se pueda cambiar la contraseña nuevamente.
  - Número de días que deben transcurrir hasta que la contraseña deba cambiarse nuevamente.
  - Días desde la expiración para que la contraseña se inhabilite
  - Fecha de caducidad de la cuente.

Los grupos de usuarios

- Cada usuario puede pertenecer a uno o varios grupos. Cada grupo tiene un identificador de grupo (GID).
- Todo usuario tiene un grupo principal que es usado por defecto al crear nuevos archivos.
- En muchos sistemas, cuando se crea un usuario automáticamente se crea un grupo principal para el usuario con el mismo nombre que el usuario.
- La configuración de grupos se mantiene con el fichero /etc/groups

El sistema de ficheros (I)

- El sistema de fichero se ve como una sola jerarquía cuya raíz es /
- Todo está en la misma jerarquía. Un usuario podría montar un nuevo dispositivo de almacenamiento y accedería a él en una ruta que parte de /.
- Un path completo sería `/usr/lib`
- Si el path no empieza en / se considera relativo al directorio actual de trabajo (PWD).
- El PWD se denota con un punto: `./Música`

El sistema de ficheros (II)

- La ruta exacta del PWD es privada para cada proceso y puede cambiarla con la llamada del sistema **chdir**.
- En la BASH el PWD se cambia usando el comando cd:
  - cd / (cambiar a la raíz)
    cd /usr/lib (cambiar a /usr/lib)
  - cd Música (cambiar al subdirectorio Música dentro del directorio actual)
  - cd (cambiar al directorio personal del usuario)
- El PWD se puede conocer usando el comando `pwd` o accediendo a la variable `$PWD`.

Directorios del sistema

- **/bin, /usr/bin, /usr/sbin, /usr/local/bin,:**
  Directorios de ficheros ejecutables.
- **/etc:** Directorio de los ficheros de configuración, así como scripts de arranque, parada y configuración.
- **/tmp /var/tmp, /usr/tmp:**
  Directorio para ficheros temporales.
- **/home:** Directorio que contiene los directorios de los usuarios, por ejemplo /home/juan.
- **/lib, /usr/lib:**
  Directorios para librerías.
- **/dev:**
  Directorio para dispositivos.
- **/var/log, /usr/log.**
  Directorio para ficheros de registro

Comando ls (l)

- `ls [opciones] [ruta]`
  sirve para listar el contenido de un directorio
- Ejemplos:
  - `ls -l`
    usa el formato de lista larga en lugar de mostrar sólo los nombres.
  - `ls -a`
    lista todo, incluso el contenido que empieza por '.'
  - `ls -R`
    entrando recursivamente en todos los directorios.
  - `ls -ld [ruta]` o `ls -ld [ruta]`
    muestra información del directorio indicado no de su contenido.

Comando ls (II)

- Más ejemplos:
  - `ls -i` o `ls -li`
    muestra el número de inodo (identificador único del archivo).
  - `ls -t` o `ls -tr`
    ordena el contenido por fecha de modificación. La opción '-r' invierte el orden.
  - `ls -t | head -3`
    muestra los últimos 3 ficheros modificados.
  - `ls -l | less`
    muestra el contenido página a página.

Comando mkdir

- Ejemplos mkdir:
  - `mkdir nuevodir`
    crea el directorio nuevodir en el PWD
  - `mkdir nuevo1 nuevo2 nuevo3`
    crea varios directorios nuevos en el PWD
  - `mkdir -p p1/p2/p3`
    fuerza la creción de toda la estructura. Es decir p1, p2 y p3; uno dentro del otro.

Comando cp

- Ejemplos cp:
  - `cp fich1 fich1copia`
    copia el archivo fich1 en fich1copia
  - `cp fich1 ruta`
    copia el archivo fich1 en el directorio ruta
  - `cp -R dir1 dir2`
    copia el directorio dir1 en dir2. Sin esta opción `cp` no copia directorios. Si dir2 existe, dir1 es copiado dentro. Si no, la copia de dir1 se llama dir2.
  - `cp -p file1 dir1`
    copia preservando todos los atributos del archivo (permisos, propietario, grupo, etc).

Comando mv

- Ejemplos mv:
  - `mv file1 file2 dir1`
    mueve los archivos file1 y file2 a dir1.
  - `mv dir1 dir2`
    mueve el directorio dir1 a dir2. Si dir2 existe, dir1 es movido dentro de dir2. Si no existe, dir1 es renombrado como dir2.

Comando rm

- Ejemplos rm:
  - `rm file1`
    elimina file1.
  - `rm dir2`
    si dir2 es un directorio, este comando falla.
  - `rm -r dir3`
    borra recursivamente el directorio dir3. Puede pedir confirmación.
  - `rm -rf dir4`
    borra recursivamente sin preguntar. Es muy peligroso.

Comando rm

- Mas ejemplos rm:
  - `rm -ri dir`
    borra recursivamente, pero siempre pide confirmación para el borrado de los archivos.
  - `rmdir dir1`
    elimina dir1, sólo si está vacío.

Comando cat

- El comando cat concatena ficheros y los muestra en la salida estándar.
- Ejemplos cat:
  - `cat file1 file2`
    concatena file1 y file2 y muestra el contenido en la consola
  - `cat file1 file2 > file3`
    concatena file1 y file2 y guarda el contenido en file3.
  - `cat -s file1`
    muestra file1 en la consola eliminando las líneas vacías repetidas.

Comando chown (I)

- Modifica el usuario y el grupo al que pertenece un archivo o directorio.
- El usuario root puede modificar cualquier archivo o directorio, con cualquier usuario.
- Un usuario normal sólo podrá modificar el grupo, para poner alguno a los que pertenece (al menos en la mayoría de los UNIX).

Comando chown (II)

- Ejemplos chown:
  - `chown usuario fichero`
    modificar el propietario del fichero
  - `chown usuario:grupo fichero`
    `chown usuario.grupo fichero`
    modificar el usuario y el grupo del fichero
  - `chown -R usuario directorio`
    modificar recursivamente todos los archivos y directorios incluidos en el directorio.
  - `chown :grupo archivo`
    `chown .grupo archivo`
    modifica sólo el grupo. Funciona como chgrp.

Comando chgrp

- Ejemplos chgrp:
  - `chgrp grupo fichero`
    modificar el grupo del fichero
  - `chgrp -R grupo directorio`
    modificar recursivamente todos los archivos y directorios incluidos en el directorio.

Comando chmod (I)

- Sirve para modificar las autorizaciones sobre archivos y directorios.
  - Afecta a la lectura, escritua, ejecución.
  - Afecta al listado, búsqueda en directorios y escritura de directorios.
- Sólo podremos modificar los atributos de un fichero o directorio si somos los propietarios del mismo (a excepción del root que puede modificar cualquiera).

Comando chmod (II)

- El sistema de autorizaciones UNIX diferencia para cada elemento del sistema de archivos las siguientes clases de usuarios:
  - usuario, grupo, otros y todos.
- Para cada uno de ellos se pueden establecer permisos con chmod.
- Los permisos efectivos para un usuario se determinan por el siguiente orden de prioridad:
  - usuario, grupo y otros.

Comando chmod (III)

- Por ejemplo,
  - si el usuario pertenece al grupo propietario del archivo y éste no tiene asignado el permiso de lectura,...
  - ...pero el usuario coincide con el usuario propietario del archivo y este sí tiene asignado el permiso de lectura,...
  - **...la lectura será un permiso efectivo para el usuario.**

Comando chmod (IV)

- Ejemplos chmod:
  - `chmod o= archivo`
    elimina todos los permisos para otros
  - `chmod o=r archivo`
    añade r y elimina w y x)
  - `chmod go+r archivo`
    añade r en grupo y otros
  - `chmod go+r-w archivo`
    añade r y elimina w para grupo y otros

Comando chmod (V)

- Más ejemplos chmod:
  - `chmod 750 archivo`
    establece el archivo rwx para el usuario, rx para el grupo y sin ningún permiso para otros
- 750 = máscara de permisos en octal
  - 4 = r (permiso de lectura)
    2 = w (permiso de escritura)
    1 = x (permiso de ejecución)
  - rwx = 4 + 2 + 1 = 7
    rx = 4 + 1 = 5

Comando chmod con archivos (I)

- **Denegación del permiso de lectura en un archivo.**
  - No se puede acceder al contenido del archivo, pero sí a aspectos como: tamaño, fecha de última modificación, propietario, etc.
- **Denegación del permiso de escritura en un archivo.**
  - No podemos modificar el contenido pero sí aspectos como la fecha de última modificación o el propietario.
- **Denegación del permiso de ejecución.**
  - No podemos lanzar la ejecución del archivo.

Comando chmod con archivos (II)

- **Permiso set user ID, setuid o SUID (u+s).**
  - Al ejecutarse el archivo, el usuario efectivo del proceso es el usuario propietario del archivo.
  - Un ejemplo son los comandos `su` o `sudo` que permiten realizar operaciones como si fuéramos el root porque ese es su propietario.
  - Otro ejemplo, es el comando passwd que permite cambiar el password del usuario.
- **Permisto set group ID, setgid o SGID (g+s).**
  - Al ejecutarse el archivo, el grupo efectivo del proceso es el grupo propietario del archivo.

Comando chmod con archivos (III)

- **Permiso sticky bit (+t).**
  - En la actualidad no se usa.
  - En sistemas antiguos se usaba para que tras ejecutar el programa y terminar, el código del mismo se almacenara en el espacio de intercambio (swap), permitiendo que la ejecución posterior del mismo programa fuera más rápida.

Comando chmod con directorios (I)

- **Denegación del permiso de lectura en un directorio.**
  - No se pueden realizar búsquedas dentro del directorio. Por ejemplo, listar su contenido.
  - Sí se puede acceder a los archivos dentro del directorio.
  - Es como si no pudieramos acceder a la representación interna del directorio (salvo a la resolución de nombres) pero al contenido de los archivos.
- **Denegación del permiso de escritura en un directorio.**
  - No podemos agregar, renombrar o eliminar elementos del directorio.
  - Sí podemos modificar los archivos en su interior (si disponemos de los permisos para ello).

Comando chmod con directorios (II)

- **Denegación del permiso de “ejecución”.**
  - No podemos cambiar el PWD al directorio.
  - Podemos usar la representación interna del directorio para hacer búsquedas, pero no se puede acceder al contenido de los archivos.

Comando chmod con directorios (III)

- **Permiso set user ID, setuid (SUID) en directorios (u+s).**
  - Este permiso no tiene efecto.
- **Permiso set group ID, setgid (SGID) en directorios (g+s)**
  - Este permiso hace que cuando un usuario cree un archivo o directorio dentro del directorio afectado por el mismo, el grupo propietario del nuevo elemento no sea el grupo efectivo del proceso, sino el del directorio padre.
- **Permiso sticky bit en directorios (+t).**
  - Este permiso prohibe que los usuarios puedan renombrar, mover o borrar archivos, salvo los que pertenezcan a ellos mismos (la excepción, como siempre es el root).

Comando umask

- Cuando creamos un archivo, ¿con qué permisos se crea?
  - El comando umask complementado con 666, nos devuelve estos permisos.
  - Por ejemplo, si umask devuelve 022, los permisos son 666 – 022 = 644 (rw-r—r--)
  - Por tanto **umask indica los permisos que no se establecerán al crear un archivo**.
  - Se puede modificar. Por ejemplo “umask 000” establecería los permisos a 666 (rw-rw-rw-)

Inodos (I)

- La entidad básica del sistema de ficheros es el inodo.
  - Cada fichero tiene un inodo.
  - Cada inodo tiene un número único.
- Un inodo contiene información sobre:
  - Tipo, propietario, tamaño, permisos, marca temporal y localizaciones en el disco donde se encuentra el fichero.
  - Los inodos no contienen nombres de fichero.
- Los inodos con su información se almacenan en una tabla indexada por el número de inodo.

Enlaces duros

- Los directorios deben entenderse como un fichero que contiene una tabla, mediante la cuál desde un nombre de fichero se obtiene en número de inodo.
  - Un inodo puede tener más de un nombre, en el mismo o en directorios diferentes. Estos nombres se denominan **enlaces duros** o **hard links**.
  - Podemos crear hard links de diversas formas. El primero cuando creamos un fichero y luego podemos establecer más con el comando ln.

Enlaces simbólicos

- Los **soft links**, **symbolic links** o **enlaces simbólicos** son otra cosa.
- Se trata de ficheros que contienen a su vez un nombre de fichero.
- En el proceso de búsqueda de una ruta, cuando se encuentra un soft link el sistema se redirige automáticamente al nombre contenido en él.

Comando ln

- Establece un hard link o soft link
- Ejemplos ln:
  - `ln nuevonombre ficheroyaexistente` crea un hard link.
  - `ln -s nuevonombre nombreyaexistente` crea un soft link.
-

Dispositivos y archivos especiales

- Un inodo también puede referirse a un dispositivo.
  - De esta manera los dispositivos quedan representados en el sistema de ficheros (ver archivos en /dev)
  - Ejemplo: /dev/null representa un dispositivo que descarta la información que se le envía.
- Además de los ficheros regulares, directorios, enlaces simbólicos y dispositivos, pueden existir otro elementos referenciados por los inodos.