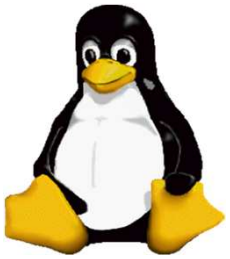
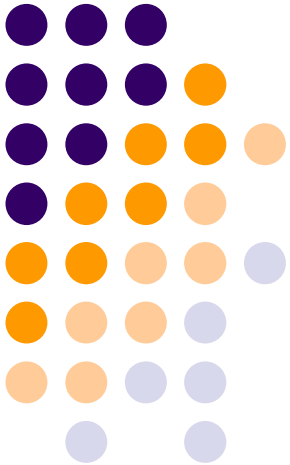
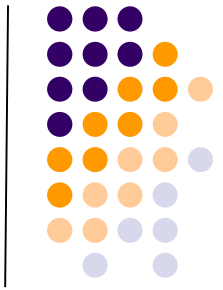
Unidad 6

Administración de S.O. GNU/Linux



UNIDAD 6
Administración de S.O. Linux

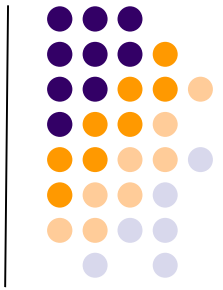


0. Jerarquía sistema de archivos Linux
1. Ejecución de programas como administrador
2. Usuarios y grupos
3. Gestión de programas
4. Control de procesos
5. Monitorización y registros del sistema
6. Servicios
7. Automatizar tareas del sistema

*. Administración remota

*. Programación de scripts

0. Jerarquía sistema de archivos Linux

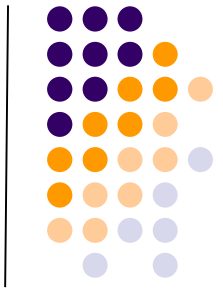


- Estructura del sistema de archivos
  - Linux ordena y clasifica todos los ficheros y directorios que componen el sistema de archivos
  - El sistema está jerarquizado, partiendo del directorio raíz /
  - Cuenta con una completa organización del sistema de archivos → cada tipo de archivo tiene un directorio destino
  - Esta clasificación permite una localización más rápida de cualquier archivo sin necesidad de recurrir a programas de búsqueda y una mayor limpieza de su contenido

0. Jerarquía sistema de archivos Linux

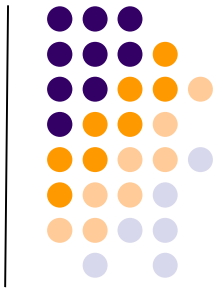
- Estructura del sistema de archivos

0. Jerarquía sistema de archivos Linux



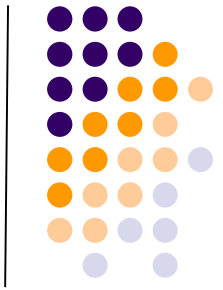
- Estructura del sistema de archivos
  - /**bin** → contiene los archivos ejecutables esenciales del sistema (sobre todo para utilizar en modo texto)
  - /**boot** → ficheros necesarios para el arranque (GRUB y núcleo Linux)
  - /**dev** → archivos virtuales relacionados con los dispositivos internos y externos (discos, teclados, tarjetas expansión…)
  - /**etc** → ficheros de configuración de los programas y del sistema. Individuales y clasificados por directorios
  - /**home** → directorios personales de usuarios (excepto root)

0. Jerarquía sistema de archivos Linux



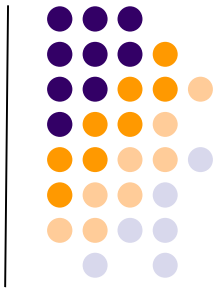
- Estructura del sistema de archivos
  - /**lib** → bibliotecas y módulos del núcleo. Su uso es compartido por las aplicaciones, para evitar duplicidades
  - /**media** → punto de montaje de particiones externas (unidades ópticas, USB, unidades de otros S.O., etc.)
  - /**mnt** → similar al anterior, pero solo para unidades de uso esporádico
  - /**proc** → sistema de archivos virtual, con datos de solo lectura relacionados con los dispositivos y el SO (*swaps*, *cpuinfo*, *crypto*,…)
  - /**root** → directorio personal de root

0. Jerarquía sistema de archivos Linux



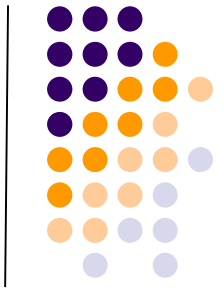
- Estructura del sistema de archivos
  - /**sbin** → archivos ejecutables destinados a la administración del SO
  - /**srv** → datos internos de los servicios accesibles a todos los usuarios
  - /**tmp** → directorio temporal, en cada arranque se borra
  - /**usr** → ficheros relacionados con programas opcionales (los que no necesita el SO para realizar sus funciones básicas). Incluye *games*, *include*, *lib*, *local*, *share*,…
  - /**var** → archivos con datos de los servicios. Su contenido es modificado continuamente. Contiene varios subdirectorios: *cache*, *lib*, *lock*, *log*, *mail*, *opt*, *run*, *spool*…

1. Ejecución de programas como administrador



- Comando sudo
  - En todo sistema Linux existe un usuario root que tiene todos los privilegios sobre el sistema
  - En Ubuntu deshabilitado por defecto (por seguridad)
  - En su lugar se usa el comando **sudo** para realizar tareas administrativas
  - Con sudo, Ubuntu permite que otros usuarios puedan ejecutar tareas propias del root
  - Cuando queremos ejecutar una tarea de root, el sistema comprueba que tenemos permiso para ejecutarla, pide la contraseña del usuario que ha ejecutado sudo, y solo si es correcta inicia la tarea (por defecto, recordada durante 15 minutos)

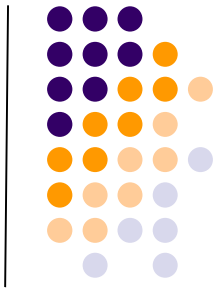
1. Ejecución de programas como administrador



- Comando sudo
  - Fichero de configuración de sudo: **/etc/sudoers**
    - Se indica qué usuarios pueden ejecutar tareas como root y qué acciones pueden llevar acabo

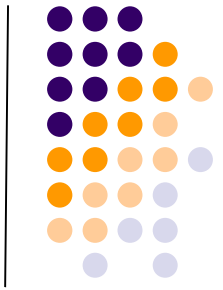
```
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

1. Ejecución de programas como administrador



- Comando sudo
  - Fichero: /**var/log/auth.log**
    - Mantiene un registro de todas las actividades realizadas por los usuarios autorizados en /etc/sudoers
  - El fichero sudoers se puede editar con cualquier editor, aunque se recomienda usar el comando **visudo**
    - Hace una verificación del archivo sudoers antes de guardar para comprobar que no hay errores sintácticos
    - Si encuentra errores nos indica dónde está el error y nos pregunta qué queremos hacer
    - Además, bloquea el archivo para que varios usuarios no puedan editarlo simultáneamente

1. Ejecución de programas como administrador



- Comando sudo
  - Registrar usuario en sudo: $ **sudo adduser usuario sudo**
  - Cambiar el tiempo que sudo recuerda la contraseña (si queremos un entorno más seguro):

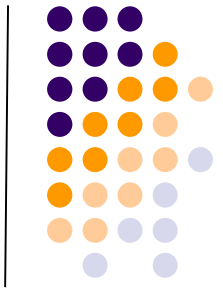
    $ sudo visudo

Buscar línea: Defaults env_reset

y añadir: Defaults env_reset, **timestamp_timeout = X**

X indica el tiempo en minutos
si ponemos un 0 pedirá la contraseña cada vez que se utilice sudo

1. Ejecución de programas como administrador

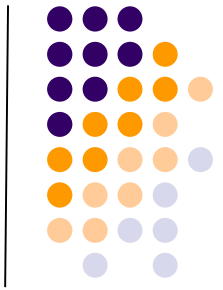


- Comandos gksu y gksudo
  - **gksu** (y **gksudo**) sirven para lanzar aplicaciones gráficas en entornos gráficos
  - Si ejecutamos aplicaciones gráficas con sudo podemos tener problemas con los permisos de usuario de nuestros archivos
  - Con sudo estamos trabajando como root, pero dentro del entorno del usuario que estemos usando en ese momento. En este caso, podrían crearse ficheros nuevos en nuestras carpetas en los que el propietario sea root
  - Con gksu y gksudo, además de usar la aplicación como root, estamos utilizando su propio entorno de trabajo (no se guardarán en las carpetas del usuario ficheros de root)

1. Ejecución de programas como administrador

- Comandos gksudo y gksu
  - Uso:
    gksu <nombre_programa>
    gksudo <nombre_programa>
  - Si lanzamos el comando gksu sin argumentos, aparece una ventana para escribir el nombre del programa a ejecutar:

1. Ejecución de programas como administrador



- Comando su
  - Permite cambiar de usuario en el mismo terminal en el que se está trabajando, manteniendo su entorno de usuario:

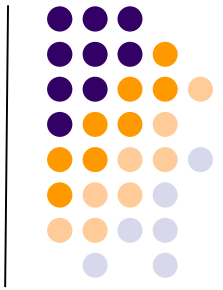
    $ **su** <nuevo_usuario>
  - Con la opción **-**, se ejecutan los scripts de inicio del nuevo usuario. En este caso tendremos una sesión idéntica a la obtenida con la entrada por login de ese usuario:

    $ **su -** <nuevo_usuario>
  - Cambiar a root sin tener que activar la cuenta:

    $ **su** (siguiendo en el entorno de usuario)

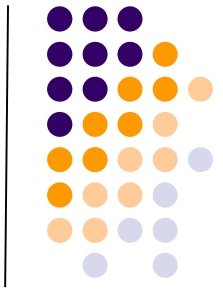
    $ **su -** (cambiamos al directorio personal y entorno de root)
  - Para salir de la sesión como root: # **exit**

2. Usuarios y grupos en Linux



- Introducción
  - La gestión de usuarios y de grupos la tiene que realizar un usuario con privilegios de administración o root
  - Para gestionar usuarios y grupos gráficamente:
    - Herramienta **gnome-system-tools**

2. Usuarios y grupos en Linux

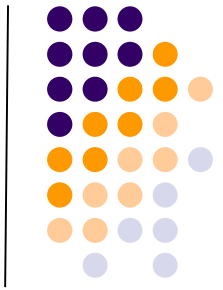


- Gestión de usuarios
  - La información principal de los usuarios se almacena en el fichero /**etc/passwd**
  - Contiene las cuentas de usuario existentes
  - Estructura:

    usuario:x:UID:GID:Comentarios:Directorio_Home:Shell

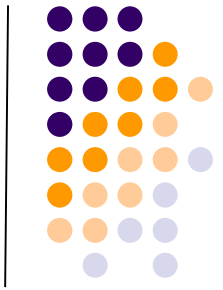
    - usuario indica el nombre de la cuenta de usuaurio
    - x indica que el password se encuentra en el archivo /etc/shadow
    - UID del usuario y GID del grupo principal
    - Comentarios
    - Directorio home del usuario
    - Shell que va a usar el usuario

2. Usuarios y grupos en Linux



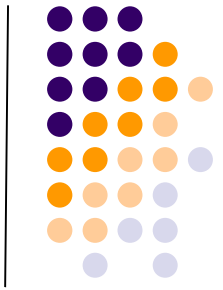
- Gestión de usuarios
  - El fichero /**etc/shadow** contiene las contraseñas encriptadas de los usuarios
  - En el fichero de configuración /**etc/login.defs** están definidas variables que controlan aspectos relacionados con la creación de usuarios y sus contraseñas: longitud de contraseñas, número de días de validez, etc.

2. Usuarios y grupos en Linux



- Gestión de usuarios
  - **Creación de usuarios**
    - Comando: **useradd** [opciones] *nombre_usuario*
      - Crea un usuario indicando como parámetros información adicional sobre ese usuario
    - Opciones:
      - **-g**: Grupo principal que queremos tenga el usuario (debe existir)
      - **-d**: Carpeta home del usuario (suele ser */home/nombre_usuario*)
      - **-m**: Crear carpeta home si es que no existe
      - **-s**: Intérprete de comandos o shell del usuario (suele ser /bin/bash)

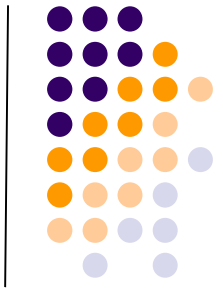
2. Usuarios y grupos en Linux



- Gestión de usuarios
  - **Creación de usuarios**
    - El comando useradd permite crear muchos usuarios automáticamente mediante archivos de comandos (scripts)
    - Recomendable que el nombre de usuario sea en minúsculas
    - Además de letras también puede contener números y algunos signos como guiones normales y guiones bajos
    - Ejemplo:

$ *sudo useradd -g contabilidad -d /home/pedro -m -s /bin/bash pedro*

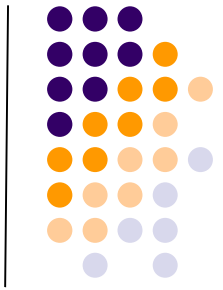
2. Usuarios y grupos en Linux



- Gestión de usuarios
  - **Creación de usuarios**
    - Comando: **passwd** [opciones] *nombre_usuario*
      - Establece o cambia la contraseña a un usuario
    - Opciones:
      - **-x**: caducidad de la contraseña en días
      - **-n**: fuerza a cambiar la contraseña transcurridos ´n´ días
      - **-w**: días de antelación con los que avisa para hacer el cambio
      - **-e**: hace que la contraseña caduque inmediatamente, forzando al usuario a cambiarla
    - Ejemplo:

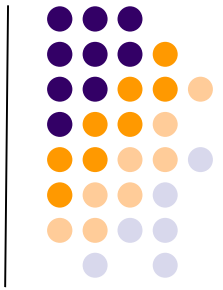
$ *sudo passwd –w 3 –x 15 –n 2 paco*

2. Usuarios y grupos en Linux



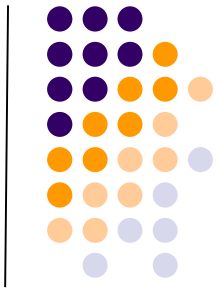
- Gestión de usuarios
  - **Creación de usuarios**
    - Comando: **adduser**
      - Crea un usuario indicando sus datos mediante de forma interactiva mediante un asistente
      - Realmente es un enlace al comando useradd

2. Usuarios y grupos en Linux



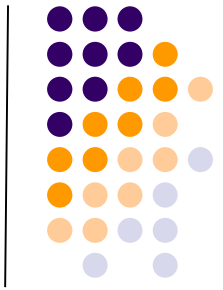
- Gestión de usuarios
  - **Modificación de usuarios**
    - Comando: **usermod** [opciones] *nombre_usuario*
      - Permite cambiar el nombre del usuario, su carpeta home, su intérprete de comandos, grupos a los que pertenece, etc.
      - Opciones:
        - **-d**: modifica la carpeta personal del usuario
          - **-m**: mueve el contenido de la carpeta home a una nueva ubicación
        - **-g**: cambia el grupo principal del usuario
        - **-l**: cambia el nombre de la cuenta (login)
        - …

2. Usuarios y grupos en Linux



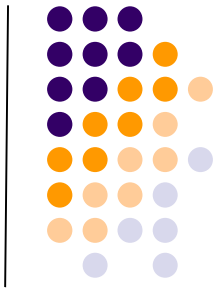
- Gestión de usuarios
  - **Eliminación de usuarios**
    - Comando: **userdel** [opciones] *nombre_usuario*
      - Opciones:
        - **-r**: elimina su carpeta personal

2. Usuarios y grupos en Linux



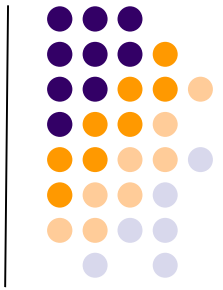
- Gestión de grupos
  - En Linux todo usuario pertenece necesariamente a un grupo
  - Todos los privilegios concedidos al grupo son heredados automáticamente por los usuarios o grupos que contiene
  - Un grupo puede incluir uno o varios usuarios o/y otros grupos
  - Cuando damos de alta a un usuario, por defecto pertenece a un grupo que el sistema crea con el mismo nombre del usuario
  - Además, Linux crea tras su instalación una serie de grupos a los que podemos agregar los usuarios que vayamos creando, como root, users, admin, ssh,…

2. Usuarios y grupos en Linux



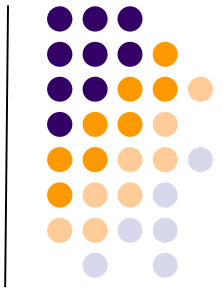
- Gestión de grupos
  - La información de los grupos se almacena en el archivo **/etc/group**
  - Contiene los nombres de los grupos y una lista de los usuarios que pertenecen a cada grupo
  - Cada línea representa un grupo y contiene 4 campos
    - Nombre del grupo (recomendable que no tenga más de 8 caracteres)
    - Contraseña cifrada o una x que indica la existencia de una clave en /etc/shadow
    - Número GID
    - Lista de los miembros del grupo, separados por comas (sin espacios)

2. Usuarios y grupos en Linux



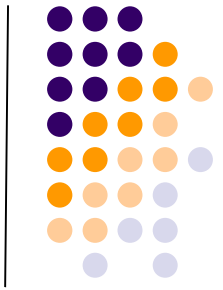
- Gestión de grupos
  - **Creación de grupos**
    - Comando: **groupadd** [opciones] *nombre_grupo*
  - **Modificación de grupos**
    - Comando: **groupmod** [opciones] *nombre_grupo*
    - Opciones:
      - **-n**: nuevo_nombre
      - **-g**: cambia el gid

2. Usuarios y grupos en Linux



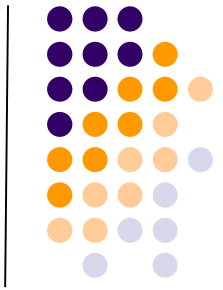
- Gestión de grupos
  - **Eliminación de grupos**
    - Comando: **groupdel** [opciones] *nombre_grupo*
      - Si algún usuario tuviera al grupo a eliminar como grupo primario, no se eliminará el grupo
  - **Añadir usuario a un grupo**
    - Comando: **adduser** *nombre_usuario nombre_grupo*
  - **Quitar usuario de un grupo**
    - Comando: **deluser** *nombre_usuario nombre_grupo*

2. Usuarios y grupos en Linux



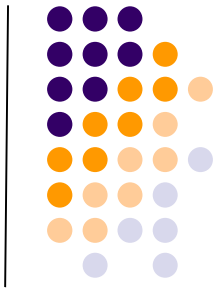
- Otros comandos
  - **Cambiar información de un usuario**
    - Comando: **chfn** [opciones] *nombre_usuario*
      - Opciones:
        - **-f**: cambia nombre completo del usuario
  - **Mostrar grupos a los que pertenece un usuario**
    - Comando: **groups** *nombre_usuario*

2. Usuarios y grupos en Linux



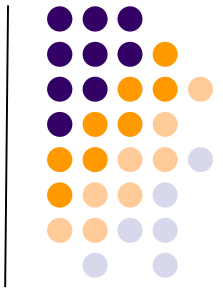
- Otros comandos
  - **Cambiar propietario de un fichero o carpeta**
    - Comando: **chown** [opciones] *nombre_nuevo_usuario nombre_archivo*
      - Opciones:
        - **-R**: afecta a todos los archivos y subcarpetas dentro de la carpeta indicada
  - **Cambiar grupo de un archivo o carpeta**
    - Comando: **chgrp** [opciones] *nombre_nuevo_grupo nombre_archivo*
      - Opciones:
        - **-R**: afecta a todos los archivos y subcarpetas dentro de la carpeta indicada

3. Gestión de programas en Linux



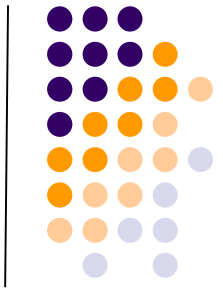
- Introducción
  - En Linux podemos instalar programas de varias formas:
    - A través de **repositorios**
    - Mediante ficheros descargables preparados para su instalación directa (**.deb**, **.bin**, **.run**)
    - Compilando su código fuente
  - Los programas están formadas por paquetes de software
    - Un **paquete** es un archivo comprimido con una estructura que permite ser tratado por herramientas de gestión de software para realizar su instalación, desinstalación y actualización
  - Los paquetes mantienen **dependencias** entre sí
    - La instalación de un paquete puede depender de que se instale también otro, recomendar que se instale o entrar en conflicto uno ya instalado

3. Gestión de programas en Linux



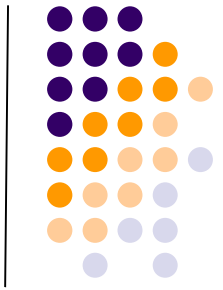
- Gestión de paquetes con APT
  - **APT** (advanced packaging tool) es una herramienta utilizada para la gestión de paquetes .deb desde la línea de comandos
  - Sus programas principales son **apt-get**, **apt-cache** y **apt**
  - Gestionan los paquetes accesibles a través de los repositorios configurados en el fichero **/etc/apt/sources.list**
  - Los repositorios de Ubuntu se dividen en cuatro tipos:
    - *Main*: soportado oficialmente por Ubuntu. Libre
    - *Restricted*: soportado oficialmente por Ubuntu. No es libre
    - *Universe*: no soportado oficialmente por Ubuntu. Libre
    - *Multiverse*: no soportado oficialmente por Ubuntu. No es libre
  - También se pueden añadir repositorios de terceros

3. Gestión de programas en Linux



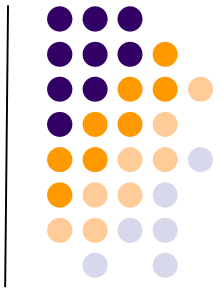
- Gestión de paquetes con APT (apt-get)
  - Opciones **apt-get**
    - ✓ Instalar paquetes: $ sudo **apt-get install** <paquetes>
    - ✓ Reinstalar paquete: $ sudo **apt-get --reinstall** <paquete>
    - ✓ Desinstalar paquetes: $ sudo **apt-get remove** <paquetes>
    - ✓ Desinstalar paquetes eliminando los archivos de configuración:
      $ sudo **apt-get purge** <paquetes>
      $ sudo **apt-get remove ---purge** <paquetes>
    - ✓ Resolver dependencias: $ sudo **apt-get -f install**

3. Gestión de programas en Linux



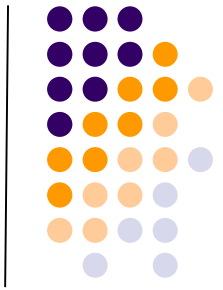
- Gestión de paquetes con APT (apt-get)
  - Opciones **apt-get**
    - ✓ Actualizar el listado de paquetes disponibles en el repositorio:
      $ sudo **apt-get update**
    - ✓ Actualizar todos los paquetes instalados sin desinstalar paquetes y sin instalar paquetes nuevos:
      $ sudo **apt-get upgrade**
    - ✓ Actualizar todos los paquetes instalados pero resolviendo conflictos de forma “inteligente” (evita actualizar un paquete si, al hacerlo, se rompe una dependencia):
      $ sudo **apt-get dist-upgrade**

3. Gestión de programas en Linux



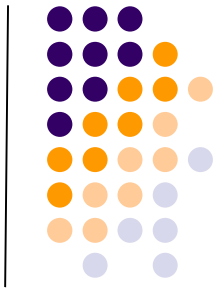
- Gestión de paquetes con APT (apt-get)
  - Opciones **apt-get**
    - ✓ Actualizar la caché de paquetes y revisar la existencia de dependencias rotas: $ sudo **apt-get check**
    - ✓ Borrar totalmente el repositorio local que contiene los ficheros de los paquetes descargados: $ sudo **apt-get clean**
    - ✓ Borrar solo aquellos paquetes que ya no se pueden descargar o que son claramente inservibles: $ sudo **apt-get autoclean**
    - ✓ Desinstalar paquetes instalados automáticamente para satisfacer dependencias de otro paquete, pero que ya no son necesarios (apt-get no elimina los paquetes dependientes del programa elimininado): $ sudo **apt-get autoremove**

3. Gestión de programas en Linux

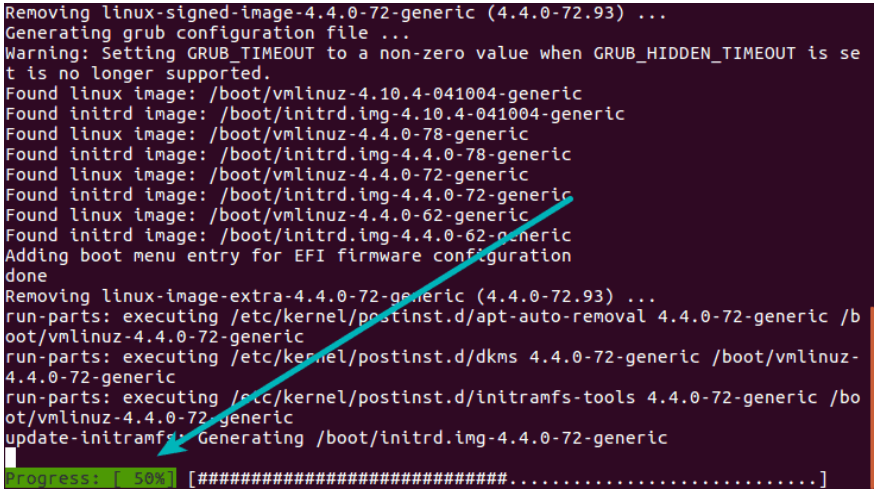


- Gestión de paquetes con APT (apt-cache)
  - **apt-cache** sirve para ofrecer información sobre la base de datos de paquetes Debian ("la caché de paquetes")
  - Opciones **apt-cache**
    - ✓ Obtener más información de un paquete específico (información de paquetes instalados o no):
      $ sudo **apt-cache show** <paquete>
    - ✓ Para saber de qué paquetes depende:
      $ sudo **apt-cache depends** <paquete>
    - ✓ Mostrar estadísticas de paquetes: $ sudo **apt-cache stats**
    - ✓ Mostrar todos los paquetes instalados en el sistema:
      $ sudo **apt-cache pkgnames**

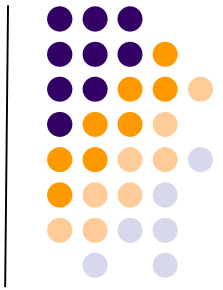
3. Gestión de programas en Linux



- Gestión de paquetes con APT (apt)
  - **apt s**e desarrolla en 2014 para simplificar la gestión de paquetes de APT (desde Ubuntu 16.04)
  - Tiene opciones muy similares a **apt-get** aunque añade algunas mejoras (barra de progreso, información sobre el número de paquetes que se pueden actualizar, resultados más claros y ordenados…)

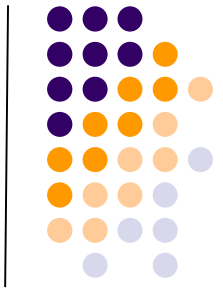


3. Gestión de programas en Linux



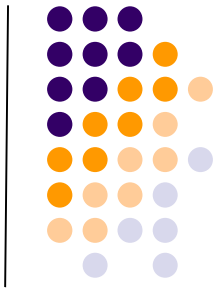
- Gestión de paquetes con APT (apt)
  - Opciones **apt**
    - ✓ Instalar paquetes: $ sudo **apt install** <paquetes>
    - ✓ Desinstalar paquetes: $ sudo **apt remove** <paquetes>
    - ✓ Desinstalar paquetes eliminando los archivos de configuración: $ sudo **apt purge** <paquetes>
    - ✓ Eliminar dependencias que ya no son necesarias: $ sudo **apt autoremove** <paquetes>

3. Gestión de programas en Linux



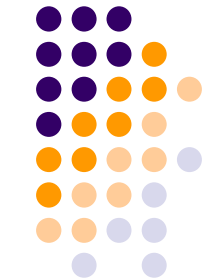
- Gestión de paquetes con APT (apt)
  - Opciones **apt**
    - ✓ Actualizar el listado de paquetes disponibles en el repositorio: $ sudo **apt update**
    - ✓ Actualizar todos los paquetes instalados: $ sudo **apt upgrade**
    - ✓ Actualizar todos los paquetes instalados gestionando las dependencias: $ sudo **apt full-upgrade**
    - ✓ Buscar un programa: $ sudo **apt search** <programa>
    - ✓ Mostrar información de paquete: $ sudo **apt show** <paquete>
    - ✓ Listar paquetes según un criterio: $ sudo **apt list** <criterio>
    - ✓ Editar sources.list: $ sudo **apt edit-sources**

3. Gestión de programas en Linux



- Gestión de paquetes con aptitude
  - **aptitude** es un gestor de aplicaciones en modo texto basado en apt que simplifica la gestión de paquetes en Linux
  - También usa los repositorios configurados en "/etc/apt/sources.list"
  - A diferencia de apt-get, es capaz de eliminar las dependencias de un paquete desinstalado, por lo que no deja paquetes huérfanos
  - El requisito para que funcione bien aptitude es que todos los paquetes se hayan instalado con él, ya que no reconoce las dependencias instaladas con apt-get o con otras herramientas

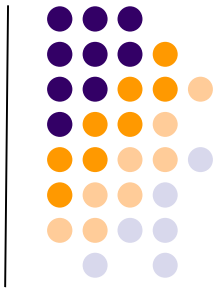
3. Gestión de programas en Linux



- Gestión de paquetes con aptitude
  - Mediante el comando *$ sudo aptitude* accedemos a una interfaz visual de instalación basada en texto

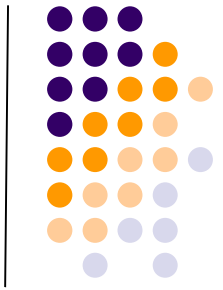
```
 Acciones  Deshacer  Paquete  Solucionador  Buscar  Opciones  Vistas  Ayuda
C-T: Menú ?: Ayuda q: Salir u: Actualizar g: Descarga/Instala/Elimina Paqs
aptitude 0.6.3                          #Roto: 3     Se usará 7426 kB de espacio de TamDesc: 4929
--- Paquetes nuevos (31800)
--- Paquetes instalados (795)
--- Paquetes obsoletos y creados localmente (65)
--- Paquetes virtuales (3998)
--- Tareas (709)




Estos paquetes se han añadido a Debian desde la última vez que borró la lista de paquetes «nuevos».
(elija «Olvidar paquetes nuevos» del menú Acciones para vaciar la lista).

Este grupo contiene 31800 paquetes.
```

3. Gestión de programas en Linux



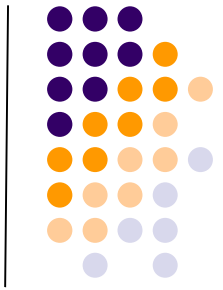
- Gestión de paquetes con aptitude
  - Opciones **aptitude**
    - ✓ Instalar paquetes: $ sudo **aptitude install** <paquetes>
    - ✓ Desinstalar paquetes: $ sudo **aptitude remove** <paquetes>
    - ✓ Desinstalar paquetes (incluyendo archivos de configuración): $ sudo **aptitude purge** <paquetes>
    - ✓ Actualizar el listado de paquetes disponibles en el repositorio: $ sudo **aptitude update**
    - ✓ Actualizar el sistema con las actualizaciones de paquetes disponibles: $ sudo **aptitude upgrade**
    - ✓ Actualizar todos los paquetes instalados, eliminando solo los no usados y añadiendo los paquetes necesarios para resolver dependencias: $ sudo **aptitude safe-upgrade**

3. Gestión de programas en Linux



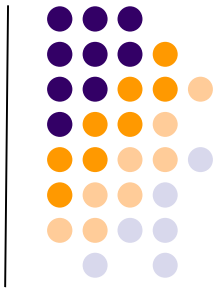
- Gestión de paquetes con aptitude
  - Opciones **aptitude**
    - ✓ Eliminar restos de paquetes ya desinstalados:
      $ sudo **aptitude clean**
    - ✓ Borrar la caché de paquetes obsoletos y que ya no figuren en los repositorios: $ sudo **aptitude autoclean**
    - ✓ Buscar un paquete en los repositorios:
      $ **aptitude search** <paquete>
    - ✓ Bloquear un paquete para que no sea actualizado (válido solo para aptitude, apt no sabe que está bloqueado):
      $ **aptitude hold** <paquete>
    - ✓ Quitar bloqueo de un paquete para que pueda ser actualizado (válido solo para aptitude): **$ aptitude unhold** <paquete>

3. Gestión de programas en Linux



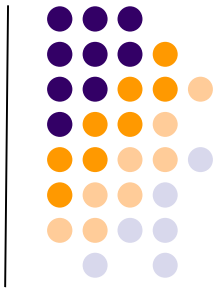
- Gestión de paquetes con dpkg
  - Herramienta para instalar, compilar, eliminar y manipular los paquetes .deb
  - Es una herramienta de más bajo nivel que apt o aptitude por lo que se utiliza para casos muy concretos
  - Opciones de **dpkg**
    - ✓ Instalar paquete a través de un fichero:
      $ sudo **dpkg -i** <paquete>.deb
    - ✓ Desinstalar paquetes (no elimina archivos de configuración):
      $ sudo **dpkg -r** <paquete>
    - ✓ Desinstalar paquetes (incluyendo archivos de configuración):
      $ sudo **dpkg -P** <paquete>

3. Gestión de programas en Linux



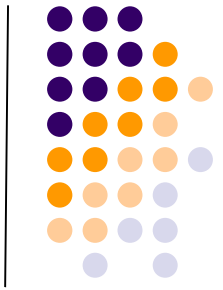
- Gestión de paquetes con dpkg
  - Opciones de **dpkg**
    - ✓ Reconfigurar un paquete ya instalado:
      $ sudo **dpkg-reconfigure** <paquete>
    - ✓ Bloquear un paquete para que no sea actualizado:
      $ echo “<paquete> hold” | dpkg --set-selections
    - ✓ Quitar bloqueo de un paquete para que pueda ser actualizado:
      $ echo “<paquete> install” | dpkg --set-selections

4. Control de procesos



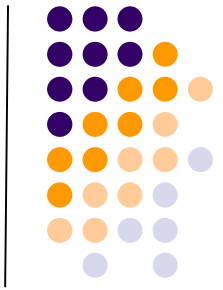
- Introducción
  - Cada una de las tareas que se realiza en el sistema es un **proceso**
  - Cada vez que se ejecuta una aplicación o programa se inicia un nuevo proceso
  - Mediante la herramienta *Monitor del sistema* se pueden gestionar procesos gráficamente

4. Control de procesos



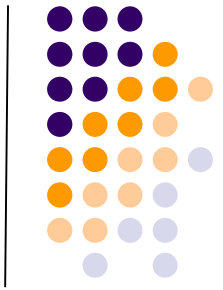
- Comandos para control de procesos
  - **ps** → muestra los procesos en ejecución
    - **ps -AL** muestra un listado largo con información sobre TODOS los procesos
    - **ps aux** muestra listado de todos los procesos de todos los usuarios junto a la CPU y memoria que consumen
    - **ps –u usuario** muestra los procesos de un usuario concreto

4. Control de procesos



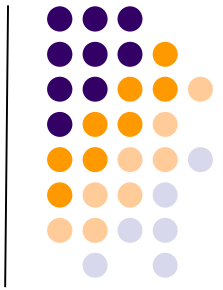
- Comandos para control de procesos
  - **kill** <pid> → finaliza un proceso
    - Hay que especificar su pid (podremos visualizarlo con ps) e incluso la identificación del proceso padre ppid
    - Si matamos un proceso y con ps observamos que no ha finalizado, podemos indicar una finalización incondicional del mismo con **kill pid -9**
    - Con otros parámetros se pueden enviar otras señales a los procesos (lista completa de opciones con **kill -l**)

4. Control de procesos



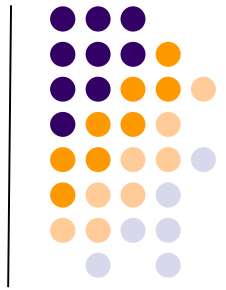
- Comandos para control de procesos
  - **killall** <nombre_proceso> → finaliza un proceso junto con todos sus subprocesos (no se indica el pid de los procesos, sino su nombre)
  - Puede suceder que al ver los procesos activos, el que se desea finalizar tiene distintos subprocesos lanzados (dispone de varios pid) → para matarlo definitivamente habría que utilizar varias veces kill o usar killall una sola vez
  - Para pausar un proceso se puede usar la opción **kill -STOP** <pid>
  - Para reanudarlo **kill -CONT** <pid>
  - **xkill** → elimina el proceso asociado a la ventana que se marca (en entornos gráficos)

4. Control de procesos



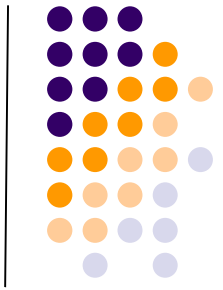
- Prioridad de los procesos
  - En Linux los procesos tienen asignado un valor de prioridad que va de +19 (menos prioritario) a -20 (más prioritario)
  - Los procesos con más prioridad podrán acceder a más recursos del equipo, principalmente tiempo de procesador
  - Cuando se ejecuta un programa el SO le asigna automáticamente una prioridad. El valor por defecto es 0
  - Comandos:
    - **nice** <valor_prioridad> <nombre_proceso>→ lanza un proceso con una prioridad determinada
    - **renice –n** <valor_prioridad> <pid> → cambia la prioridad de un proceso en ejecución

4. Control de procesos



- Tipos de ejecución de los procesos
  - En Linux un programa puede arrancar en **primer plano** o en **segundo plano**
  - Cuando se ejecuta en primer plano vemos en todo momento su estado de ejecución (es la ejecución habitual)
  - En este caso, para poder ejecutar otro comando hay que esperar a que este termine o abrir una nueva terminal
  - Al ejecutar un comando en segundo plano se regresa inmediatamente al shell y se pueden ejecutar otros comandos, mientras el comando anterior continúa ejecutándose

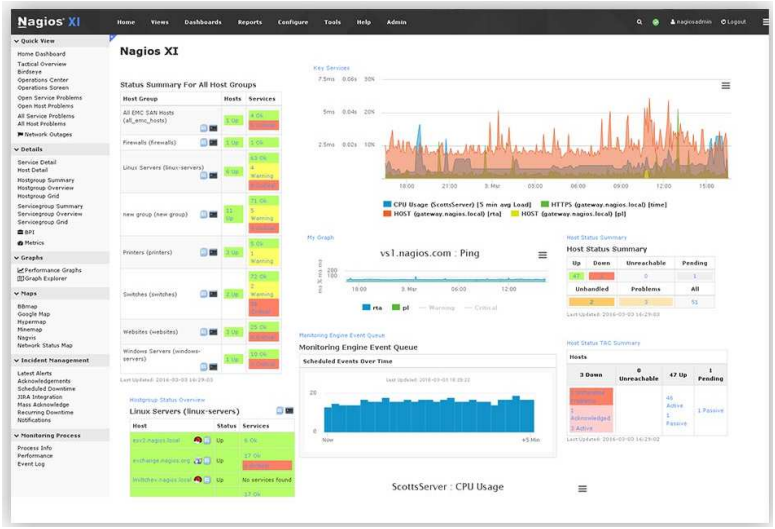
4. Control de procesos

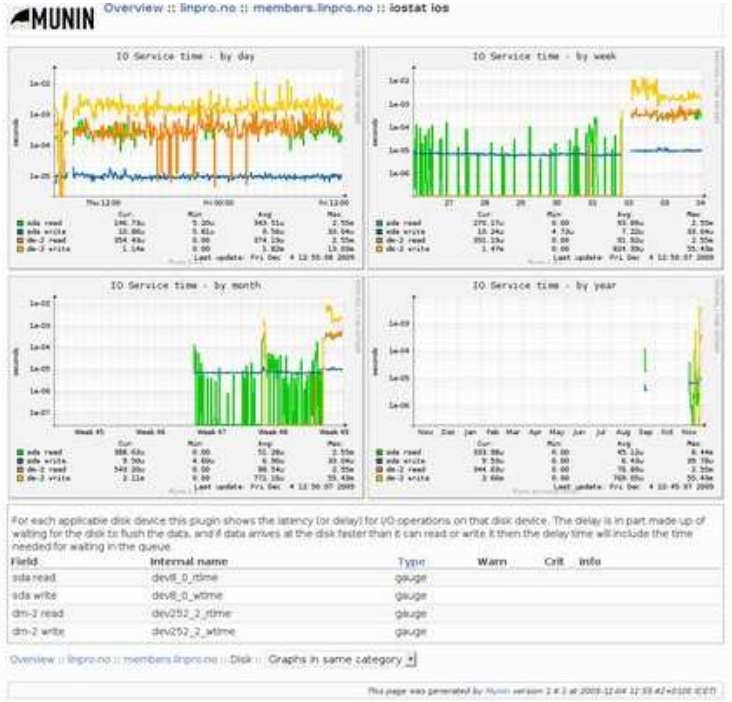


- Tipos de ejecución de los procesos
  - **&** → al final del comando, ejecuta directamente en segundo plano
  - **bg** → lista los trabajos parados o en segundo plano
  - **fg** → trae los trabajos más recientes a primer plano
    - **fg <n>** → trae el trabajo <n> a primer plano
  - **jobs** → lista los trabajos solo del terminal en el que estamos trabajando, apareciendo todos los procesos corriendo en primer y segundo plano
  - **Ctrl + Z** → pausa el trabajo en curso
  - **Ctrl + C** → finaliza trabajo en curso

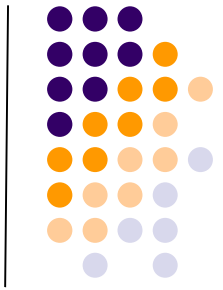
5. Monitorización y registros

- Monitorización de sistemas
  - Monitorizar sistemas permite detectar sobrecargas, intentos de intrusión en el sistema, errores de aplicaciones y demonios, etc.



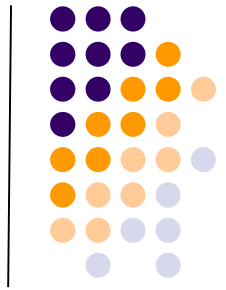


5. Monitorización y registros



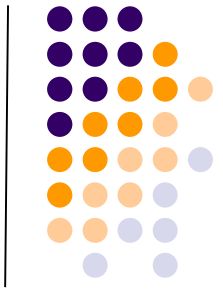
- Monitorización de procesos
  - **top** → muestra una lista de procesos en tiempo real, permitiendo realizar varias acciones sobre ellos: matarlos, cambiar prioridad, y mostrando además información sobre el uso de los recursos consumidos (CPU, RAM, etc.)
  - **htop** → similar a top, pero además permite ordenar los procesos por parámetros, como el uso de CPU y RAM (hay que instalarla, no viene por defecto)

5. Monitorización y registros



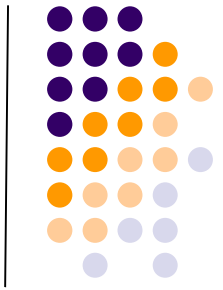
- Monitorización del hardware
  - Paquete systat
    - mpstat, iostat, vmstat, dstat
  - Uso de memoria y disco
    - free
    - iotop, df, du
  - Otros
    - uptime, w, who
    - xosview
  - En el directorio /proc hay diversos ficheros con información sobre el sistema

5. Monitorización y registros



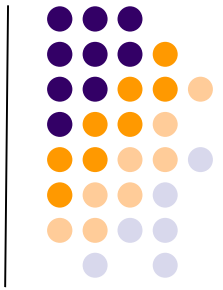
- Registros del sistema
  - Linux cuenta con una serie de ficheros de registro o logs (bitácoras) que ayuden al administrador del sistema en el mantenimiento del servidor y de la red
  - En estos ficheros se va almacenando información sobre todos los eventos que ocurren en el sistema
  - Pueden servir para la localización de errores y ataques al sistema, para el seguimiento de la actividad de usuarios en el sistema, conocer qué ocurre durante el arranque del equipo, etc.

5. Monitorización y registros



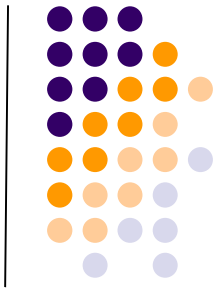
- Ficheros logs
  - La mayoría situados en /**var**/**log** aunque puede haber aplicaciones que generen logs en sus propias carpetas
    - /**var**/**log**/**messages** → archivo principal de registros donde varias aplicaciones del sistema y demonios almacenan mensajes de interés (desactivado por defecto en Ubuntu, sustituido por */var/log/syslog*)
    - /**var**/**log**/**syslog** → log del sistema y del kernel con información sobre eventos que suceden en el sistema y en sus programas
    - /**var**/**log**/**dmesg** → información generada por el kernel durante el arranque (se puede ver su contenido con el comando **dmesg**)

5. Monitorización y registros



- Ficheros logs
  - **/var/log/auth.log** → mensajes sobre autenticación de usuarios y permisos
  - **/var/log/boot.log** → mensajes relacionados con el arranque
  - **/var/log/daemon.log** → mensajes sobre demonios o servicios que funcionan en el sistema
  - **/var/log/dpkg.log** → registro de los paquetes binarios instalados
  - **/var/log/kern.log** → registros del kernel, generados por klogd
  - **/var/log/Xorg.0.log** → mensajes de Xorg (servidor gráfico)
  - **/var/log/lpr.log** → mensajes relacionados con la impresión
  - **/var/log/debug** → mensajes de depuración

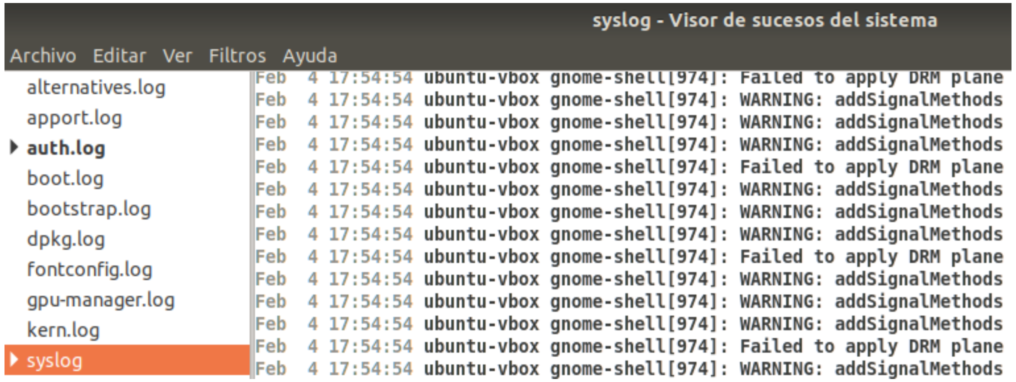
5. Monitorización y registros



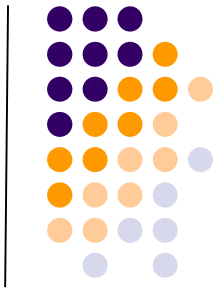
- Manejo de logs
  - Comandos para visualización
    - **cat** concatenado con **grep** sirve para buscar cadenas de texto concretas → *cat /var/log/syslog | grep ntp*
    - **less** (consume pocos recursos al no cargar todo el contenido en memoria; uso de cursores) → *less /var/log/syslog*
    - **tail** (muestra las últimas líneas de un fichero [10 por defecto]; para ver más opciones –n) → *tail -n 20 /var/log/syslog*

5. Monitorización y registros

- Manejo de logs
  - Monitorización de logs
    - Para ver los sucesos que ocurren en tiempo real
      - **tail -f** → tail -f /var/log/syslog
        - Salimos con Ctrl + C
      - **less** → less /var/log/syslog y pulsamos Shift + f
    - Utilidades gráficas
      - *Visor de archivos de suceso*

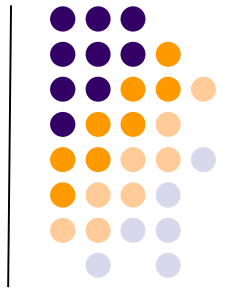


5. Monitorización y registros



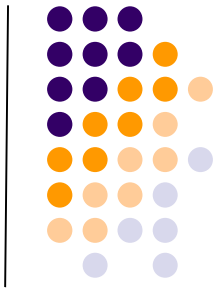
- Manejo de logs
  - Rotación de ficheros logs
    - Con el tiempo los logs pueden crecer mucho y complicar su manejo
    - El comando **logrotate** permite archivar los ficheros log, creando ficheros nuevos con la información más actual
    - Se ejecuta automáticamente cada cierto tiempo, aunque también puede ejecutarse manualmente
    - Cuando es ejecutado, coge el log actual y le añade un ".1" al final. Cuando haya otra rotación más adelante pasará a tener un ".2" al final y así sucesivamente.
    - Ejecución: *sudo logrotate -vf /etc/logrotate.conf*
    - Archivo de configuración: */etc/logrotate.conf*

6. Servicios en Linux



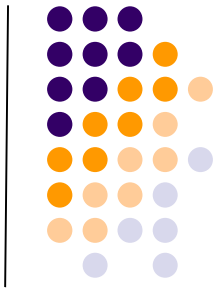
- Introducción
  - Los servicios son programas que se ejecutan en segundo plano y ofrecen una función concreta dentro del sistema
  - Pueden ser iniciados al arrancar el sistema, al entrar en un *runlevel* (nivel de ejecución) determinado o manualmente cuando sean necesarios
  - A los servicios en Linux que no requieren de la intervención del usuario para iniciarse se les llama demonios (*daemons*)
  - Ejemplos de servicios: gestión de la conexión de red, comprobación de actualizaciones, sincronización de hora, gestión de tareas programadas, monitorización del sistema, impresión, cortafuegos, etc.

6. Servicios en Linux

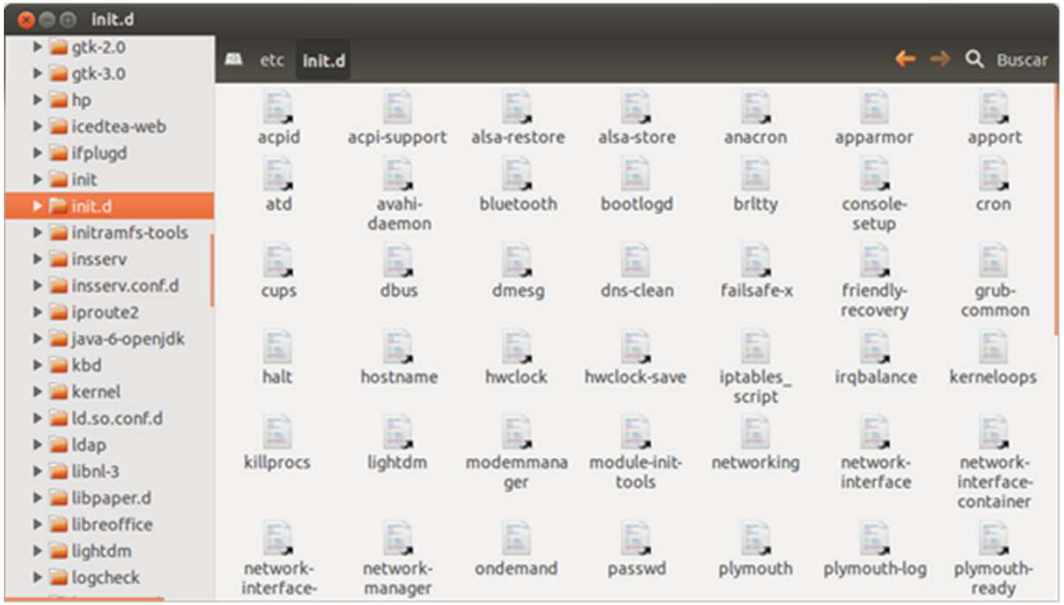


- Introducción
  - En sistemas Linux tradicionales, **init** es el programa encargado de controlar la secuencia de arranque, siendo el encargado de iniciar todos los servicios
  - Situado en /**sbin**/**init**, recibe el **PID 1**, al ser el primer programa en iniciar
  - Actualmente hay tres métodos de inicialización de servicios:
    - **System-V Linux (SysV)** → método tradicional de Linux, la ejecución de los distintos procesos está gestionada según distintos niveles de ejecución o *runlevels*
    - **Upstart** → se añadió en Ubuntu 6.10 como mejora al init clásico, pero se abandonó en Ubuntu 16.04
    - **systemd** → desde Ubuntu 16.04, usado también por otras distribuciones (Fedora, Red Hat, CentOS,…)

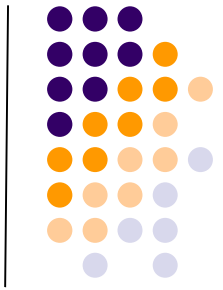
6. Servicios en Linux



- Método clásico System-V
  - En SysV, los programas ejecutados como demonios pueden estar ubicados en cualquier lugar del sistema de archivos, pero todos utilizan un **script** para manejarlos localizados en **/etc/init.d**

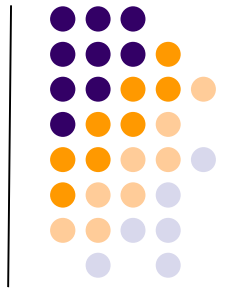


6. Servicios en Linux



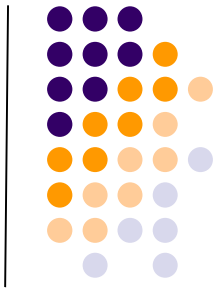
- Método clásico System-V
  - Comandos para gestión de servicios init
    - ✓ Iniciar servicio: $ *sudo etc/init.d/<nombre_servicio>* **start**
    - ✓ Detener servicio: $ *sudo /etc/init.d/<nombre_servicio>* ***stop***
    - ✓ Detener e iniciar servicio:
      $ *sudo /etc/init.d/<nombre_servicio>* ***restart***
    - ✓ Recarga la configuración sin detener el servicio:
      $ *sudo /etc/init.d/<nombre_servicio>* ***reload***
    - ✓ Muestra estado del servicio:
      $ sudo /etc/init.d<nombre_servicio> ***status***

6. Servicios en Linux



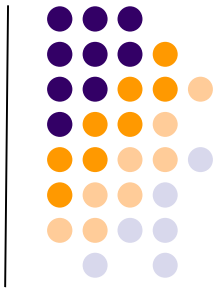
- Runlevels
  - Los **runlevels** o **niveles de ejecución** son los diferentes métodos o modos de arranque y parada que se pueden encontrar en un SO Linux
  - Existen 7 niveles de ejecución en total (en Debian/Ubuntu no se hace distinción entre los niveles 2 a 5)
    - **runlevel 0** → cierra el sistema (apagado)
    - **runlevel 1** → modo monousuario. Sin conexión de red, solo tiene acceso el administrador. Para mantenimiento del sistema
    - **runlevel 2 a 5** → modo multiusuario, con entorno gráfico si está instalado. Configurables. El 5 es el predeterminado en Ubuntu
    - **runlevel 6** → reiniciar la máquina

6. Servicios en Linux



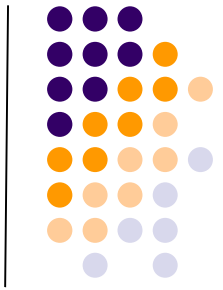
- Runlevels
  - Comandos
    - ✓ Ver runlevel en el que se encuentra el sistema: $ **runlevel**
    - ✓ Cambiar el sistema de runlevel:
      $ ***init*** *<**nº** que indique el runlevel al que queremos ir>*
  - El sistema System-V solo es capaz de ejecutar o parar procesos automáticamente ante la entrada en un *runlevel*
  - Para indicar los procesos que se han de ejecutar o parar en cada *runlevel* existen los directorios ***/etc/rcX.d/***
    (X es el número de *runlevel)*
  - Dentro de estos directorios, hay enlaces simbólicos a los scripts existentes en ***/etc/init.d***

6. Servicios en Linux



- Runlevels
  - Los nombres de estos enlaces empiezan por la letra “S” (Start/iniciar) o “K” (Kill/parar), seguida de un número y el nombre del servicio
  - El número de dos dígitos, de 00 a 99, indica el orden en que se inicia o para el servicio (el servicio con número menor se iniciará antes que otro con uno mayor)
  - Para que un servicio no se inicie en un determinado *runlevel*, cambiaremos el nombre al archivo del servicio deseado del directorio “*etc/rcX.d*” correspondiente, cambiando la S inicial por la K

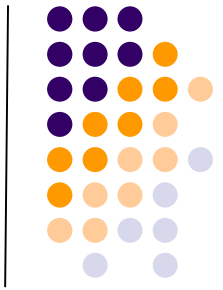
6. Servicios en Linux



- Runlevels
  - Seleccionar un runlevel en el arranque de Ubuntu
    - ✓ En el fichero "**etc/init/rc-sysinit.conf**" modificamos la línea:

      …
      env DEFAULT_RUNLEVEL=2
      …
  - Con independencia del runlevel seleccionado se ejecutarán todos los servicios de ***/etc/rcS.d***

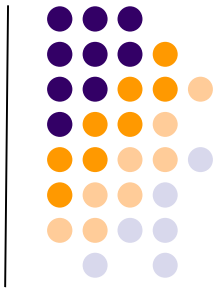
6. Servicios en Linux



- Método systemd
  - Método cada vez más utilizado en Linux
  - En lugar de muchos scripts de inicialización, utiliza un único programa que usa archivos de configuración individuales para cada servicio

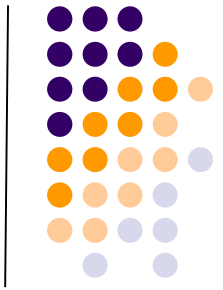
```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ systemctl stop bluetooth.service
ubuntu@ubuntu:~$ systemctl start bluetooth.service
ubuntu@ubuntu:~$ systemctl status bluetooth.service
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service;
 enabled; vendor preset: enabled)
   Active: active (running) since Tue 2015-05-05 21:31:11
 UTC; 6s ago
 Main PID: 4355 (bluetoothd)
   CGroup: /system.slice/bluetooth.service
           └─4355 /usr/sbin/bluetoothd -n
```

6. Servicios en Linux



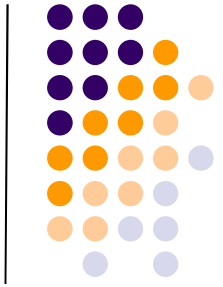
- Método systemd
  - systemd utiliza el comando **systemctl**
    - ✓ Iniciar servicio: $ *sudo* ***systemctl start*** *<nombre_servicio>*
    - ✓ Detener servicio: $ *sudo* ***systemctl stop*** *<nombre_servicio>*
    - ✓ Detener e iniciar servicio:
      $ sudo ***systemctl restart*** *<nombre_servicio>*
    - ✓ Recarga la configuración sin detener el servicio:
      $ sudo ***systemctl reload*** *<nombre_servicio>*
    - ✓ Muestra estado del servicio:
      $ sudo ***systemctl status*** *<nombre_servicio>*
    - ✓ Habilitar servicio: $ *sudo* ***systemctl enable*** *<nombre_servicio>*
    - ✓ Deshabilitar servicio: $ *sudo* ***systemctl disable*** *<nombre_servicio>*

6. Servicios en Linux



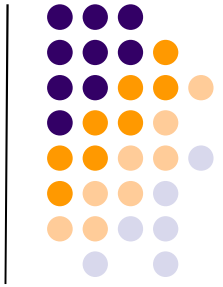
- Comando service
  - El comando **service** permite gestionar servicios basados tanto en *System-V* como en *Upstart*
    - ✓ Iniciar servicio: $ *sudo* ***service*** *<nombre_servicio>* ***start***
    - ✓ Detener servicio: $ *sudo* ***service*** *<nombre_servicio>* ***stop***
    - ✓ Detener e iniciar servicio:
      $ sudo ***service*** *<nombre_servicio>* ***restart***
    - ✓ Recarga la configuración sin detener el servicio:
      $ sudo ***service*** *<nombre_servicio>* ***reload***
    - ✓ Muestra estado del servicio:
      $ sudo ***service*** *<nombre_servicio>* ***status***
    - ✓ Lista de todos los servicios: $ *service --status-all*

7. Automatizar tareas del sistema



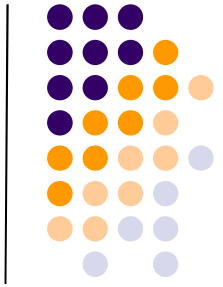
- Introducción
  - Hay una serie de tareas de administración del sistema que pueden interesar que se ejecuten automáticamente
  - Ejemplos: realizar copias de seguridad por la noche, programar el apagado automático de los equipos a una hora determinada, saber qué usuarios están conectados en una máquina, etc
  - Linux dispone de una serie de comandos y servicios para llevar a cabo esta tarea, como: **at**, **cron** y **anacron**
  - Aplicación gráfica: *gnome-schedule*

7. Automatizar tareas del sistema



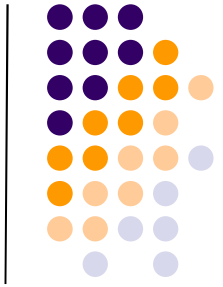
- Comando at
  - Permite planificar la ejecución de determinadas tareas en un instante concreto
  - Pueden usar esta herramienta tanto el administrador del sistema como el resto de usuarios
  - Sintaxis
    - `at [opciones] [hora] [fecha]`
  - Tras ejecutar el comando el sistema se queda a la espera de que el usuario indique la serie de comandos a ejecutar
    - Se finaliza con <Ctrl + D>

7. Automatizar tareas del sistema



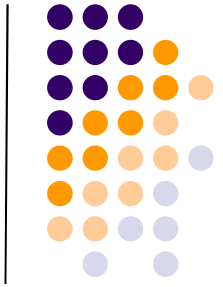
- Comando at
  - Formato de horas
    - HH:MM
    - si la hora ya ha pasado la ejecuta el día siguiente
  - Formato de fechas
    - MMDDAA o MM/DD/AA
  - Opción `now + X unidades`
    - Cuenta el tiempo indicado en X (`minutes, hours, days`) a partir del momento actual

7. Automatizar tareas del sistema



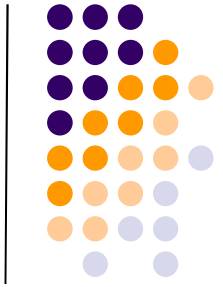
- Comando at
  - `at -f task hora fecha`
    - Ejecuta los comandos incluidos en el archivo task
  - Ver listado de tareas creadas
    - **`atq`**
    - `at -l`
  - Eliminar tareas
    - **`atrm  número_tarea`**
    - `at -d  número_tarea`

7. Automatizar tareas del sistema



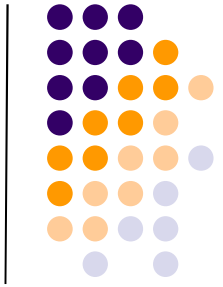
- Comando at
  - Permisos
    - root siempre puede utilizar `at`
    - Resto de usuarios: ficheros **`/etc/at.allow`** y **`/etc/at.deny`**
      - Si `/etc/at.allow` existe ⇨ solo los usuarios que aparecen en este fichero tienen permiso para utilizar at
      - Si `/etc/at.allow` no existe ⇨ se comprueba `/etc/at.deny` y todos los usuarios que no aparezcan aquí tienen permiso para utilizar at
      - Si `/etc/at.deny` está vacio ⇨ todo el mundo puede utilziarlo
      - Si no existe ninguno de los ficheros ⇨ solo puede utilizarlo el administrador del sistema

7. Automatizar tareas del sistema



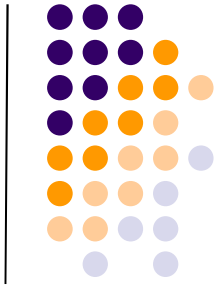
- cron y crontab
  - `cron` es un demonio que permite ejecutar comandos o procesos periódicamente (cada minuto, día, semana, mes)
  - Cada minuto comprueba la lista de trabajos y los arranca cuando se cumpla su hora
  - Los programas y horas se especifican en el fichero `crontab`
  - Cada usuario puede tener su `crontab`

7. Automatizar tareas del sistema



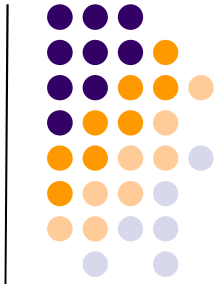
- cron y crontab
  - Sintaxis crontab
    - `crontab [-u usuario] fichero`
    - `crontab [-u usuario] {-l | -r | -e}`
  - Opciones
    - `-u usuario`: solo disponible para root, permite ver o modificar las tareas programadas para otro usuario
    - `fichero`: reemplaza la lista de tareas programadas por las que aparezcan en este fichero
    - `-l`: lista de todas las tareas programadas
    - `-r`: elimina todas las tareas programadas
    - `-e`: edita la lista de tareas programadas

7. Automatizar tareas del sistema



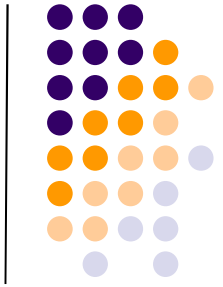
- cron y crontab
  - Creación de tareas cron
    - `crontab -e`
    - Si ponemos sudo delante será una tarea de root; si no, la tarea pertenecerá al usuario que estemos utilizando
  - Aparecerá un editor de texto en el que cada línea que escribamos será una tarea periódica
  - Formato
    - `<minuto> <hora> <día> <mes> <día_semana> <comando>`
    - En cada campo puede haber un número, una lista de números separados por comas, un rango separado por guiones o un * que indica todos los valores posibles
    - El domingo puede tener valor 0 ó 7

7. Automatizar tareas del sistema



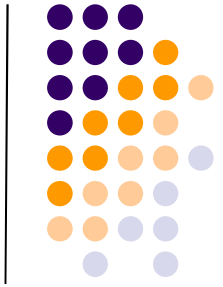
- cron y crontab
  - En vez de especificar los 5 primeros campos se pueden usar las siguientes cadenas
    - `@reboot`: se ejecuta al iniciarse la máquina
    - `@yearly`: una vez al año
    - `@monthly`: mensualmente
    - `@weekly`: semanalmente
    - `@daily`: diariamente
    - `@hourly`: se ejecuta una vez por hora

7. Automatizar tareas del sistema



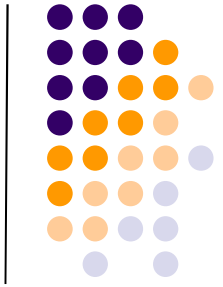
- cron y crontab
  - Tareas periódicas de administración
    - crontab almacena en `/var/spool/cron/crontabs/usuario` la lista de tareas para cada usuario
    - cron examina periódicamente estos ficheros y el fichero `/etc/crontab`, que contiene la lista de tareas periódicas propias del sistema (no conviene modificar este archivo)
      - Estas tareas tienen el mismo formato que la lista de tareas de usuarios creada con `crontab -e` pero con un campo más (el nombre del usuario que lleva a cabo la tarea)
    - Además, cron lee todos los ficheros de `/etc/cron.d` como complementos a las tareas del sistema del `/etc/crontab`

7. Automatizar tareas del sistema



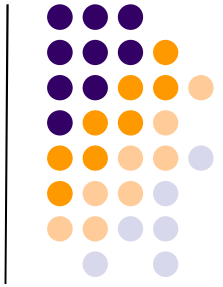
- cron y crontab
  - Tareas periódicas de administración
    - Las tareas que se han de ejecutar simultáneamente se agrupan en un mismo directorio
      - `/etc/cron.hourly`
      - `/etc/cron.daily`
      - `/etc/cron.weekly`
      - `/etc/cron.monthly`
    - En `crontab` podemos indicar como acción `run-parts <directorio>` ⇨ se ejecutarán todos los scripts contenidos en ese directorio
      - Estos scripts deben comenzar forzosamente por #!/bin/bash
    - De esta forma añadir una nueva tarea resulta sencillo, ya que no hay que tocar /etc/crontab

7. Automatizar tareas del sistema



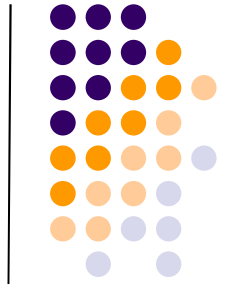
- cron y crontab
  - Permisos
    - Ficheros `/etc/cron.allow` y `/etc/cron.deny`
      - Si `/etc/cron.allow` existe ⇨ solo los usuarios que aparecen en este fichero tienen permiso para utilizar cron
      - Si `/etc/cron.allow` no existe y `/etc/cron.deny` sí ⇨ todos los usuarios que aparezcan en este último no tendrán permiso para ejecutarlo
      - Si `/etc/cron.allow` no existe y `/etc/cron.deny` está vacio o no existe ⇨ todo el mundo puede utilizar cron
  - Ejecución de aplicaciones gráficas
    - Añadir "`env DISPLAY=:0`"

7. Automatizar tareas del sistema



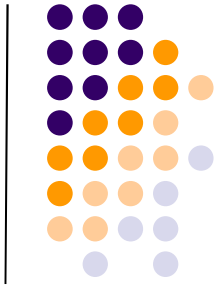
- anacron
  - cron funciona bien en sistemas que funcionen de modo ininterrumpido pero si un sistema permanece un tiempo apagado y se enciende, cron activará simultáneamente todas las tareas pendientes ⇨ se podría saturar el equipo
  - Para equipos que no van a estar operativos continuamente se recomienda usar anacron
  - anacron se inicia al encender la máquina y controla qué tareas periódicas deberían haberse realizado desde la última vez que se apagó el sistema y las ejecuta
  - En el fichero `/etc/anacrontab` se establece el periodo y el retraso para activar las tareas periódicas

*. Administración remota



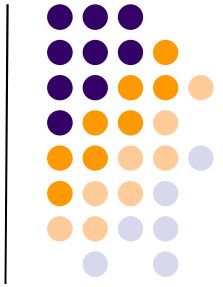
- SSH
  - SSH = Secure Shell, Intérprete de Órdenes Seguro
  - Sirve para acceder a máquinas remotas a través de la red a través de un cliente
  - En Ubuntu utilizaremos *OpenSSH (Open Secure Shell),* versión libre del protocolo SSH
  - Al instalar *OpenSSH* tendremos un conjunto de aplicaciones que nos van a permitir:
    - La administración remota a través de un terminal
    - El envío o copia de archivos entre dos máquinas
    - Ejecutar aplicaciones gráficas remotamente
    - Incorporar un ftp seguro

*. Administración remota



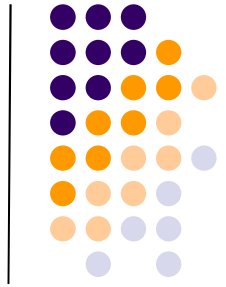
- Resumen del servicio SSH
  - Fichero de configuración: **/etc/ssh/sshd_config**
    - Se puede cambiar el puerto (por defecto el 22), los usuarios de la máquina que pueden ser accesibles por SSH, diferentes opciones de seguridad y el registro de actividad
  - Tras hacer modificaciones, hay que reiniciar el servicio:
    $ sudo /etc/init.d/ssh restart
  - Arranque/parada del servicio:
    - $ sudo /etc/init.d/ssh start
    - $ sudo /etc/init.d/ssh stop

*. Administración remota



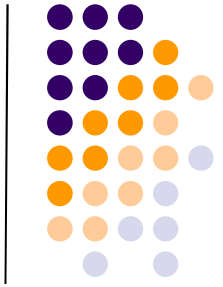
- Resumen del servicio SSH
  - Paquetes a instalar en el servidor:
    openssh-server y openssh-client
  - Paquetes a instalar en el cliente: openssh-client
  - Comprobación del funcionamiento: $ ssh localhost
  - Conexión con servidor SSH:
    $ **ssh usuario_remoto@equipo_remoto**
    - En equipo_remoto se puede indicar el nombre del equipo o la IP del equipo remoto
    - La primera vez que se conecte alguien desde el cliente se instalará el certificado de autentificación del servidor (responder “yes” a la pregunta)

*. Administración remota



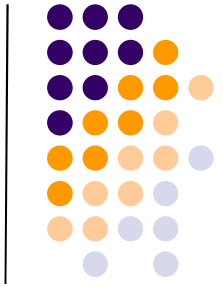
- Ejecución de aplicaciones gráficas con SSH
  - Mediante SSH es posible ejecutar aplicaciones gráficas en el servidor y visualizarlas y manejarlas desde cliente (no confundir con ver el escritorio remoto)
  - La ventaja de este sistema es que no es necesario tener instalado el programa en el ordenador cliente, ya que se ejecuta completamente en el servidor
  - Tanto el cliente como el servidor deben estar ejecutando servidores de ventanas X (X-Windows)
  - Conexión**: ssh -X usuario_remoto@equipo_remoto**
  - En el terminal del equipo cliente ejecutaremos las aplicaciones remotas, indicando el nombre del archivo de la aplicación

*. Administración remota



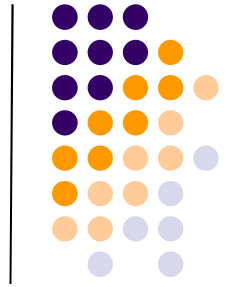
- Copiar ficheros remotamente con SSH
  - Comando: **scp**
  - Copiar archivo del servidor al cliente:
    $ **scp usuario@maquina:archivo carpeta_destino_cliente**
  - Copiar archivo del cliente al servidor:
    $ **scp archivo usuario@maquina:carpeta_destino_servidor**
  - Copiar una carpeta y subcarpetas del cliente al servidor:
    $ **scp -r archivo usuario@maquina: carpeta_destino_servidor**
- Servidor FTP seguro
  - Comando: **sftp**
  - Conexión: **$ sftp equipo_remoto**

*. Administración remota



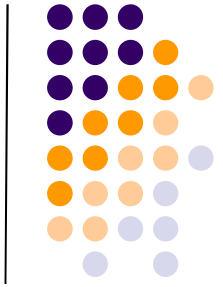
- Denegar acceso por SSH
  - Evitar acceso remoto del usuario root
    - Editamos el archivo /etc/ssh/sshd_config y establecemos PermitRootLogin no
    - Reiniciar servicio: $ sudo /etc/init.d/ssh restart
  - Evitar acceso remoto a cualquier usuario
    - Editamos el archivo /etc/ssh/sshd_config y añadimos la linea DenyUsers usuario1, usuario2
    - Se puede añadir también a root
    - Reiniciar servicio
  - Denegar acceso por SSH a equipos
    - Editamos el archivo /etc/hosts.deny y establecemos la linea sshd:IP
    - Reiniciar servicio

*. Administración remota



- Administración remota con VNC
  - **VNC** = *Virtual Network Computing*, Computación en red virtual
  - Software que permite conectar remotamente con equipos desde un entorno gráfico
  - La información no viaja encriptada, por lo que no resulta un sistema seguro para administrar servidores (más adecuado para administrar equipos clientes dentro de una red local)
  - Se basa en una estructura cliente-servidor
  - Incorpora dos módulos de gestión: *Server (*el equipo que lo tiene instalado podrá ser monitorizado) y *Viewer* (el equipo podrá monitorizar a otros)

*. Programación de scripts



- Programación de scripts
  - Crear script: $ `sudo nano script.sh`

    ```
    #! /bin/bash
    …
    …
    …
    ```

  - Dar permisos de ejecución: $ `sudo chmod +x script.sh`
  - Ejecución: $ `./script.sh`