Laboratorio de Administración y Gestión de Redes y Sistemas
Grado en Ingeniería Telemática, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Formato de la memoria de prácticas

La memoria de estas prácticas debes escribirla en un fichero en formato markdown. El fichero estará en tu cuenta del laboratorio. El dia del examen, recogeremos automáticamente tus prácticas.

1. Es imprescindible que respetes al pie de la letra los nombres de los ficheros especificados en el guión. Una letra mal puesta equivale a una práctica no presentada (o un examen no presentado). Si bien dispondrás de un script que verificará que has usado los nombres correctos

2. Redacta la memoria describiendo escuetamente todo lo que haces. Esta memoria te será muy útil para preparar el examen práctico (recuerda que podrás llevarla al examen).

   No merece la pena que te preocupes de tener una redacción muy cuidada: copia y pega órdenes y resultados, describe telegráficamente lo que haces. Vete preparando la memoria a la vez que trabajas, no lo dejes para el final.

   a) Cuando un paso del guión te pida una orden de shell o similar, es recomendable que primero ejecutes la orden y luego copies y pegues en la memoria. Si resulta algún texto, pégalo también.

   b) Cuando el guión pida que escribas o edites un fichero, basta con que lo modifiques en su sitio. No es necesario que copies y pegues en la memoria.

Práctica 1.1. Directorios de las prácticas

Ahora prepararás el directorio de la práctica 1 y el fichero de texto para la memoria de la esta práctica.

1. Entra en tu cuenta del laboratorio Linux de la ETSIT. Crea el directorio `~/lagrs`. Aquí guardarás la mayoría de tu trabajo de prácticas en la asignatura

2. Ponle permisos `rwx------`

   Debes mantenerlo así todo el curso. Recuerda que debes ser autor del 100% de tus prácticas y no permitir que ningún compañero tuyo tenga acceso a ningún fragmento.

3. Crea el directorio

   `~/lagrs/practica01`

4. Crea el fichero

   `~/lagrs/practica01.md`

   Observa que el nombre de este fichero **no** es

   ```
   ~/lagrs/practica01/practica01.md       # ¡Este no es el nombre correcto!
   ```

   Por el momento déjalo vacío.

5. Seguiremos este convenio en todas las prácticas, un directorio para cada práctica y un fichero de texto plano para cada práctica, fichero que cuelga de `~/lagrs`

Práctica 1.2. Uso básico de vi

El objetivo de esta práctica es que sepas usar al menos las órdenes elementales de vi. Si no te gusta este editor, podrás usar algún otro editor sencillo en modo texto. Pero para poder instalar un editor nuevo, necesitas usar vi.

1. Usando vi, crea el fichero `~/lagrs/practica01/ejemplo.txt` y escribe en él 4 titulares de cualquier periódico de hoy. Escribe 3 o 4 faltas de ortografía, intencionadamente. Guárdalo.

2. Vuelva a abrirlo y corrige las faltas.

Práctica 1.3. Uso de un editor sin gráficos

Ahora empezarás a usar el fichero de la memoria de prácticas. Escribe la memoria de este apartado y de todos los apartados siguientes de esta práctica en el fichero que creaste en el apartado 1.1. `~/lagrs/practica01.md`

Es necesario que manejes con soltura funciones al menos intermedias de algún editor de texto sin gráficos. La recomendación es vi/vim, pero también puedes elegir uno más sencillo.

1. Lee las transparencias sobre http://gsyc.urjc.es/ mortuno/lagrs/editores.pdf. Arranca al menos una vez los editores en modo texto presentados (vim, mcedit, joe y nano). Elige uno, para usar en esta asignatura. ¿Cuál has elegido? ¿por qué?. Puedes cambiar de opinión mas adelante, pero en tal caso, indícalo (recuerda que estas prácticas las recogeremos el día del examen).

2. Es muy conveniente conocer atajos de teclado. Es mucho más eficaz memorizar, por ejemplo, 2 al día durante 5 días que intentar retener 10 atajos 1 día. Aprende unos cuantos de esta forma, al menos media docena, e indícalo aquí. Ejemplo:

   ```
   2021.09.24
   Elijo vi porque es el más potente. Practico el copy-paste con yy p
   2021.09.28
   Me coloco en una palabra y pulso asterisco para buscar esa misma palabra
   en el texto. n minúscula para repetir la búsqueda hacia adelante y N mayúscula
   para buscar hacia atrás.
   2021.09.30
   Practico ma, mb, mc para poner las marcas a, b y c en un texto. Vuelvo
   a las marcas con 'a 'b 'c
   ```

Práctica 1.4. Gestión de contraseñas

Guarda dos o tres contraseñas, de prueba o reales, usando

1. gpg

2. LibreOffice

3. KeePassx. O alguna herramienta similar como Bitwarden.

Usa recordatorios de contraseña en todos los casos, en un fichero de texto aparte. Guarda los ficheros donde quieras, describe brevemente en la memoria los pasos que has seguido

Práctica 1.5. Secret Sharing

Usa ssss para descomponer una contraseña en 6 trozos, de forma que baste con 4 para restaurarlos. Recomponla.

Práctica 1.6. Vagrant

En este ejercicio probarás Vagrant, lanzando una máquina virtual de VirtualBox. Si quieres hacer esta práctica en tu ordenador de casa, tendrás que instalar Vagrant y VirtualBox si no lo tenías ya.

1. Crea un *project directory* de Vagrant en el directorio `~/lagrs/vbox01` de tu pc (de casa o del laboratorio)
2. Prepara una máquina Ubuntu 20.04 LTS Focal Fossa.
3. Lanza la máquina y entra por ssh, usando vagrant.
4. Comprueba que el *project directory* del host está montado dentro de la máquina virtual en `/vagrant`

   Para ello, edita un fichero en este directorio desde el *guest* o el *host*, guárdalo y vuelve a editarlo desde la otra máquina (el *host* si antes usaste el *guest* y viceversa).
5. Apaga la máquina desde vagrant, sin destruirla.
6. Si hiciste la práctica en casa o en tu portátil, copia el directorio `~/lagrs/vbox01` de tu pc de al directorio `~/lagrs` de tu cuenta del laboratorio. De esta forma, el directorio `~/lagrs/vbox01` de ambas máquinas será idéntico.

Práctica 1.7. Usuarios y grupos

En este ejercicio practicarás con los usuarios y grupos de linux.

1. Vuelve a entrar en la máquina virtual del apartado anterior.
2. Crea un nuevo usuario, con el nombre que quieras.
3. Abre una sesión de este usuario (con la orden `su`)
4. Este usuario no tendrá privilegios para ejecutar sudo, pero intenta lanzar alguna orden y observa qué pasa.
5. Haz que este usuario sí pueda ejecutar sudo. Pruébalo.
6. Con este usuario, crea dos nuevos grupos con el nombre que quieras. Mete al usuario en estos grupos. Comprueba que están incluidos.
7. Prueba la orden `newgrp`, observa su efecto.

Práctica 1.8. tar

Prueba el uso de tar y bz2 para comprimir y descomprimir ficheros, tal y como hemos visto en las primeras páginas de la presentación *Administración de Servicios y Aplicaciones*

Práctica 1.9. split

Ahora practicarás con el troceado de ficheros. Supongamos que quieres mover la máquina que acabas de crear a un host distinto. No es un fichero excesivamente grande pero sí tiene cierto tamaño, puede ser conveniente trocearlo. Es lo que practicarás aquí.

1. Desde el interfaz gráfico de VirtualBox, exporta la máquina virtual que acabas de crear. Usa el fomato ova. Llévalo al directorio auxiliar que prefieras.
2. En ese directorio auxiliar, crea un fichero README.txt. En este fichero escribe el hash MD5 del fichero .ova.

3. Comprime este directorio con el .ova y el .txt en un fichero `.tgz`. Ponle el nombre que te parezca adecuado.
4. Divídelo en trozos, del tamaño que te parezca adecuado.
5. Destruye la máquina virtual en VirtualBox (como prefieras, desde Vagrant o desde el GUI)
6. Borra el .ova, conservando el fichero de texto con el hash.
7. Borra el tgz. (En otras palabras, borra todo menos los trozos y el hash)
8. Reconstruye el tgz a partir de los trozos.
9. Extrae los ficheros, comprueba que el hash es correcto.
10. Importa el .ova desde el GUI de VirtualBox, comprueba que funciona correctamente.

Práctica 1.10. ssh sin contraseñas

Configura la máquina virtual que preparaste con vagrant para entrar en tu cuenta del laboratorio sin escribir contraseña.

Práctica 1.11. ssh sin contraseñas, desde el laboratorio

Configura tu cuenta del laboratorio para entrar desde cualquier máquina hasta cualquier máquina, sin escribir contraseña.

Práctica 1.12. FreeFileSync

Ahora practicarás el uso de FreeFileSync para simular que sincronizas tu pc de casa con el laboratorio. Empieza creando en tu pc las carpetas `~/simula_labo/lagrs` y `~/simula_casa/lagrs`. Vamos a imaginar que

1. La carpeta `~/simula_casa/lagrs` es la carpeta `lagrs` de tu ordenador de casa.
2. La carpeta `~/simula_labo/lagrs` es la carpeta `lagrs` de tu cuenta en el laboratorio.

Una vez que lo tengas todo preparado, ya puedes hacer esta práctica:

1. Prepara un fichero `~/simula_labo/lagrs/pueba.txt`. Escribe cualquier texto.
2. Simula que vas a casa y allí añades una línea más. (por ejemplo *ahora estoy en casa*). Simula que vuelves al laboratorio y que vuelves de nuevo a casa. Recuerda que cada vez que estás *en casa*, tienes que sincronizar al principio y al final de la sesión.

Práctica 1.13. Conflictos con FreeFileSync

1. Operando de forma análoga al ejercicio anterior, y con los mismos ficheros, simula un conflicto de sincronización. Observa los mensajes de error de FreeFileSync.
2. Resuelve el conflicto. Comprueba que FreeFileSync ya no muestra errores.

Práctica 1.14. Sincronización real de tu cuenta (Práctica recomendable)

Prepara en tu PC de casa un directorio con nombre *lagrs* , en el directorio que prefieras (mis documentos, escritorio...) y sincroniza allí tu cuenta *lagrs* del laboratorio. Mantenlo sincronizado todo el curso. Te será util como copia de seguridad y para trabajar indistintamente en ambos entornos. Recuerda que eres responsable de custodiar tus prácticas: tu directorio del laboratorio podría perderse. (Sería raro pero no imposible, el servicio ofrecido es de tipo *best effort*).

Práctica 1.15 Invocación de la shell

1. Observa los ficheros de inicio de la shell de tu cuenta del laboratorio. Créalos si no los tienes. ¿Cuándo se ejecuta `.bashrc`? ¿Solamente en las shell de login? ¿Solamente en las shell que no son de login? ¿O en ambos tipos de shell? ¿Por qué?
2. Comprueba que los diferentes tipos de ficheros que se tienen que ejecutar en la invocación de la shell (interactivo y de login, interactivo no de login, no interactivo) se comportan como cabe esperar. Usa trazas como

   ```
   echo prueba blabla
   ```

   o

   ```
   echo prueba blabla >> /tmp/mi_traza.txt
   ```

3. Prepara tu cuenta del laboratorio para que el `.bash_profile` invoque al `.bashrc` (si es que no está así ya). Es recomendable que también lo hagas en tu pc de casa.
4. Define en `.bashrc` un alias que pueda usarse para un ataque o una broma. Haz una demostración.

Práctica 1.16. Instalación de Docker

En este apartado instalarás Docker en una máquina Linux

1. Instala el paquete *docker.io* en la máquina virtual *vbox01*
2. Comprueba que la instalación de docker es correcta, lanzando un contenedor de tipo *holamundo* basada en debian.
3. Haz una prueba de tipo *holamundo* basada en ubuntu.
4. Instala Docker en la máquina virtual.
5. Haz una prueba de tipo *holamundo* basada en debian para verificar la instalación.

Práctica 1.17. Uso básico de imágenes

En este apartado empezarás a usar los contenedores en la máquina virtual *vbox01*. Mientras el enunciado no especifique lo contrario, usa imágenes basadas en Ubuntu 20.04.

1. Crea un contenedor interactivo. No le pongas nombre. Ejecuta alguna orden básica de la shell. Indica alguna orden básica que esté disponible y alguna otra que no.
2. No detengas el contenedor. Crea otro contenedor interactivo, poniéndole el nombre xxxxc01, donde xxxx son las primeras 4 letras de tu nombre de usuario en el laboratorio. Para jperez, sería jperc01.

3. En un nuevo terminal, usa las ordenes de docker listar contenedores e imágenes. Explica lo que estás viendo.

4. Termina la ejecución de los contedores y observa el estado de las imágenes y de los contenedores detenidos.

5. Comprueba que el sistema de ficheros dentro del contenedor no es persistente. Esto es: escribe algún fichero, apaga el contenedor, vuelva a entrar y observa que ha desaparecido.

Práctica 1.18. Creación de una imagen de un contenedor

1. Crea una cuenta en docker hub.

2. Prepara la imagen `test/banner` descrita en las transparencias, pero llámala `tulogin/banner` siendo *tulogin* tu nombre de usuario en docker hub.

3. Lanza un contenedor con esa imagen.

4. Modifica la imagen para que una vez lanzada, no solo muestre el banner sino que abra una shell

5. Lanza un contenedor con la imagen para comprobar que puedes ejecutar la shell.

6. Sube la imagen a docker hub.

Práctica 1.19. Servidor remoto

Ahora probarás el uso de docker sin la máquina virtual vbox01. En lo sucesivo, harás todas las prácticas usando como cliente docker tu puesto del laboratorio, y como servidor docker, la máquína del laboratorio preparada para este proposito.

Desde una máquina del laboratorio (y solo desde una máquina del laboratorio) podrás acceder a la máquina *dockerio*, de dirección 10.110.100.21. Es un *servidor de contenedores* para los estudiantes de esta asignatura.

1. Comprueba que puedes entrar por ssh.

2. Configura el fichero `~/.ssh/config` de tu puesto del laboratorio para poder entrar a esta máquina usando el nombre *dockerio* (no la direccción IP).

3. Configura tu cuenta del laboratorio para poder entrar en *dockerio* por ssh sin teclear contraseñas.

4. Configura tu puesto del laboratorio para que use *dockerio* como servidor de docker.

5. Prueba ahora las práctica 1.17 y 1.18 en el servidor de contenedores.

Práctica 1.20. Creación de una imagen personalizada

En este apartado prepararás una imagen sencilla de un contenedor. Se llamará *¡TULOGIN¿/cal*, y lo único que hará será invocar a la orden de shell `cal` para mostrar el calendario del mes actual y concluir. Recuerda que esta práctica, y todas las posteriores, las ejecutarás siempre en *dockerio*, no será necesario que el enunciado lo diga explícitamente.

Vamos a establecer los siguientes convenios, que mantendremos el resto de las prácticas:

- Usaremos ubuntu:20.04 como distribución base
- Si la imagen se llama `cal`, y tu login en el laboratorio es `jperez`, los distintos contenedores que ejecutarás a partir de ella se llamarán `jpercal01`, `jpercal02`, etc

  Esto es: las primeras 4 letras de tu nombre de usuario, el nombre de la imagen y un número de dos dígitos.

- Todo lo necesario para construir y lanzar esta imagen estará en `~/lagrs/cal`
- El fichero `~/lagrs/cal/construye.sh` será un script para construir la imagen `<TULOGIN>/cal`
- El fichero `~/lagrs/cal/lanza_jpercal01.sh` será un script para lanzar la imagen `jpercal01`.

  El fichero `~/lagrs/cal/lanza_jpercal02.sh` será un script para lanzar la imagen `jpercal02`, y así sucesivamente (si quisiéramos lanzar más contenedores)

  Observa que los scripts `lanza_jpercal01.sh` `lanza_jpercal02.sh` incluye en su nombre los nombres de los contenedores, porque varios contenedores podrán compartir la misma imagen inicial.

  Observa que el script `construye.sh` no incluye el nombre de la imagen, porque dentro del directorio `~/lagrs/cal/` solo habrá ficheros relativos a la imagen `<TULGIN>/cal`
- El directorio `~/lagrs/cal/context` contendrá el contexto para esta imagen. Por tanto, los ficheros Dockerfile y entrypoint.sh estarán, respectivamente, `~/lagrs/cal/context/Dockerfile` y `~/lagrs/cal/context/entrypoint.sh`

Atendiendo a estos convenios,

1. Prepara la imagen del contenedor solicitado. La utilidad cal está en el paquete bsdmainutils.
2. Prepara el script `lanza_jpercal01.sh` y lánzalo.
3. Prepara el script `lanza_jpercal02.sh` y lánzalo.
4. Ejecuta `docker ps -a` y `docker images`, y observa que, naturalmente, se puede comprobar que ambos contenedores están basados en la misma imagen.

Práctica 1.21. Montaje bind

Ahora prepararás una imagen que cree un contenedor, con un usuario, que haga un montaje de tipo bind.

Siguiendo el criterio establecido en el apartado anterior, y suponiendo que tu usuario sea `jperez`

- El directorio del contenedor será `~/lagrs/bind/`
- El directorio contexto, `~/lagrs/bind/context`
- El contenido del directorio contexto:

  `~/lagrs/bind/context/Dockerfile`

  `~/lagrs/bind/context/entrypoint.sh`
- El script de creación de la imagen, `~/lagrs/bind/construye.sh`
- El script de lanzamiento del contenedor, `~/lagrs/bind/lanza_jperbind01.sh`

Funcionamiento del contenedor:

- El contenedor tendrá un usuario, con un nombre igual a tu nombre de usuario en el laboratorio.
- El contenedor ejecutará una shell.
- El contenedor montará tu directorio *home* (del servidor de contenedores) en el directorio `/home/jperez` del contenedor

Prueba del contenedor:

1. Lanza el contenedor y escribe en el directorio montado un fichero vacío con nombre `hola_jperez`
2. Comprueba que ese directorio es persistente. Esto es, sal del contenedor, vuelve a lanzarlo y observa que el fichero `hola_jperez` sigue en su sitio.

Práctica 1.22. sshfs

Ahora practicarás con sshfs. Escribe un script llamado `~/lagrs/practica01/monta_tmp` que monte los directorios `/tmp` de tres máquinas cualquiera de las que estén disponibles en el laboratorio (consulta el *parte de guerra*). Por ejemplo `f-l2108-pc05`, `f-l2108-pc07`, `f-l2108-pc08` y en `~/lagrs/practica01/tmp01`, `~/lagrs/practica01/tmp02`, `~/lagrs/practica01/tmp03` respectivamente.

Comprueba que el resultado es el esperado: entra por ssh en esas máquinas, edita algún fichero en el directorio `/tmp/`, comprueba que puedes editar el mismo fichero en el directorio montado en tu máquina local.

Práctica 1.23. Contenedor con fichero hosts

En este apartado prepararás un contenedor preparado para hacer ping y ssh a las máquinas del laboratorio, usando solo el nombre de host, no el FQDN (Fully Qualified Domain Name).

Esta imagen se llamará `<TULOGIN>/caa` (contenedor aa), estará basada en Ubuntu 20.04 y configurado en español.

Siguiendo el convenio descrito en el apartado anterior:

- Los contenedores lanzados a partir de esta imagen se llamarán jpercaa01, jpercaa02, etc
- Todos los ficheros necesarios estarán en `~/lagrs/caa`
- Los scritps se llamarán

  `~/lagrs/caa/construye.sh`

  `~/lagrs/caa/lanza_jpercaa01.sh`

  `~/lagrs/caa/lanza_jpercaa02.sh`

  (donde jper representa las primeras 4 letras de tu login)
- El directorio contexto será

  `~/lagrs/caa/context`

  La imagen se creará con los ficheros

  `~/lagrs/caa/context/Dockerfile`

  `~/lagrs/caa/context/entrypoint.sh`

Para conseguir que los contenedores conozcan las direcciones IP de las máquinas del laboratorio, tendrás que añadir al fichero `/etc/hosts` de cada imagen las entradas correspondientes al laboratorio, que encontrarás en el fichero `/etc/hosts` de cualquier puesto.

Para ello

- Prepara, en el directorio contexto, un fichero `delta_hosts` que contenga las entradas necesarias.
- Haz que este fichero aparezca en el directorio `/tmp/` de la imagen.
- Haz que cada vez que se inicie la imagen, se añadan estas entradas al `/etc/hosts` del contenedor.

  (En docker no es posible borrar ni reemplazar el fichero `/etc/hosts` de una imagen, pero sí puedes modificarlo añadiendo entradas)
- Haz que desde los contenedores se pueda hacer ifconfig, ping y ssh. Para ello necesitarás los paquetes ubuntu adecuados, búscalos en internet.

  Puedes instalar algún paquete adicional si quieres, con tal de que no resulte una imagen mucho más pesada.

Comprueba que una vez lanzado el contenedor, puedes hacer ping o entrar por ssh a los puestos del laboratorio que lo soporten, sin necesidad de escribir el FQDN, esto es, sin añadir al nombre el dominio aulas.gsyc.urjc.es.

Ten en cuenta que:

- Solo algunas máquinas del laboratorio acepta conexión por ssh.
- El cortafuegos del laboratorio no permite el tráfico de paquetes ICMP (lo que incluye el ping) desde fuera de su subred. Por tanto, podrás ver que se ejecuta la orden ping y que se envía la petición a la dirección IP adecuada, pero no verás respuesta.

Práctica 1.24. Conectividad entre contenedores

En este apartado instalarás sshd en un contenedor y probarás la red bridge de Docker.

1. Los contenedores estarán basados en Ubuntu 20.04.
2. El nombre de la imagen que crearás será `<TULOGIN>/cab`

   Como en la práctica 1.20, al nombre de cada contenedor le añadirás como prefijo las primeras 4 letras de tu usuario, y como sufijo, un número de dos dígitos.
3. Al igual que en la práctica 1.20, este contenedor estará configurado en español, desde él debe ser posible acceder por ssh a los puestos del laboratorio sin usar su FQDN. También lanzar peticiones ping e usar ifconfig.
4. Lanzarás dos contenedores con esta imagen.

   Por tanto, los nombres de los ficheros necesarios serán:

   `~/lagrs/cab/context/Dockerfile`

   `~/lagrs/cab/context/entrypoint.sh`

   `~/lagrs/cab/construye.sh`

   `~/lagrs/cab/lanza_jpercab01.sh`

   `~/lagrs/cab/lanza_jpercab02.sh`
5. Prepara la imagen de forma que el contenedor que la ejecute tenga lanzado el demonio servidor de ssh.
6. Averigua la dirección IP de cada contenedor
7. Haz ping entre ambos
8. Añade un usuario a `jpercab01`

   Hazlo de forma interactiva, esto es, desde la shell. Elige el nombre y contraseña que quieras.
9. Abre una sesión desde `jpercab02` hasta `jpercab01`

Práctica 1.25. Control de integridad

Usa *diff* para averigual el nombre de 3 o 4 de las utilidades que puedes lanzar de forma gráfica desde el escritorio.

Práctica 1.26. Benchmark de cpu

Observa los BogoMIPS de los puestos de diversas salas de la escuela. Observa la información ofrecida por `inxi`. Haz lo mismo con algún ordenador u ordenadores que tengas en casa. Averigua las características principales de estas CPU: año, generación, características principales. Prepara una pequeña tabla (en markdown) con todo esto.

Práctica 1.27. Benchmark de red

Escribe un pequeño informe basado en *iperf* sobre la calidad de alguna pequeña red de datos. Puede ser la de tu casa o la de alguna otra red a la que tengas acceso. Puedes comparar la conexión entre diferentes puntos o a diferentes horas del día. Puedes observar el efecto de elemenos como el microndas o la lavadora.

Revisión de los nombres de los ficheros

Ejecuta `~mortuno/revisa practicas lagrs` para comprobar que los nombres de los programas son los correctos.

Laboratorio de Administración y Gestión de Redes y Sistemas 2022-2023
Grado en Ingeniería Telemática
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 2.1. Acceso a un listado de procesos

El objetivo de esta práctica es que te familiarices con la librería *subprocess* para acceder a la shell de Linux desde Python.

Crea el directorio `~/lagrs/practica02`

Escribe un script en Python 3 con el nombre `~/lagrs/practica02/mi_top.py`. Incluye un comentario en las primeras líneas tu nombre, apellidos y login.

Debe mostrar un listado con el nombre de los procesos de sistema que más CPU consuman en ese momento. Aunque hay funciones específicas de Python para obtener información sobre los procesos, usa *subprocess*, invocando a `top -n 1`. En otras palabras, el programa debe tomar la salida de esta orden, procesarla en Python, filtrarla y escribir lo filtrado en la salida estándar. En otras palabras: ofrecer un subconjunto de la información ofrecida por `top -n 1`.

- El script mostrará los procesos de sistema que ocupen un porcentaje de CPU mayor a 0.0.
- En este listado debe aparecer el nombre del proceso, su pid, el nombre del usuario propietario, el uid del usuario, los nombres de todos los grupos a los que pertenece el usuario y el porcentaje de CPU consumido por el proceso. Para obtener la información relativa al usuario, emplea la orden de shell `id`.
- La información debe aparecer en formato de columnas. Similar a lo que hace `top`.
- No uses regexp.
- No incluyas ninguna cabecera con los nombres de los campos.
- Sobra decir que es necesario que organices tu programa en funciones. Esto es algo que debes hacer siempre, también con programas cortos como este.

Práctica 2.2. Enlaces simbólicos

En este ejercicio usarás los enlaces simbólicos para tener diferentes versiones del mismo programa, pero con el mismo nombre. Esto es, tendrás un enlace simbólico que podrá apuntar a diferentes versiones del mismo programa.

1. Por si te equivocas en este ejercicio, haz una copia de `~/lagrs/practica02/mi_top.py` en el directorio que prefieras. Por ejemplo en `~/tmp` (seguramente tendrás que crearlo)
2. Renombra la praćtica `~/lagrs/practica02/mi_top.py` como `~/lagrs/practica02/mi_top01.py`
3. Crea un enlace simbólico llamado

   `~/lagrs/practica02/mi_top.py`

   que apunte a

   `~/lagrs/practica02/mi_top01.py`.

Práctica 2.3. Regexp

Haz una copia de `~/lagrs/practica02/mi_top01.py` llamada `~/lagrs/practica02/mi_top02.py` y modifícala para usar expresiones regulares, con comentarios dentro de cada regexp.

Haz que `~/lagrs/practica02/mi_top.py` ahora apunte a esta versión.

Práctica 2.4. Opciones

Usando la librería *optparse*, añade al menos 3 opciones al script `mi_top02.py`. (Sin contar la *h* de *help*, con esta serían al menos 4). Elige las opciones que desees, todas deben extraer y/o agregar diversa información de `top`. Usa regexp, con comentarios, en alguna o en todas estas opciones. El comportamiento que tenía el script en la fase anterior puede ser una de esas opciones.

Antes de implementar las opciones que hayas elegido, consúltalas con el profesor.

Práctica 2.5. Módulos

Lleva parte del código del programa anterior a uno o varios módulos.

1. Los módulos deberán llamarse *TULOGIN_xxxxx.py*, esto es, empezarán por tu nombre de usuario en el laboratorio (en minúscula) y a continuación, lo que te parezca más adecuado. P..e *jperez_shell.py*, *mgarcia_opciones.py*, etc.

2. Los módulos estarán en el directorio `~/lagrs/lib`. Por tanto, tendrás que incluir este directorio en la variable de entorno *PYTHONPATH*. Hazlo en el fichero *.bashrc* y asegúrate de que tu práctica 3.1.3 funciona correctamente.

3. Antes de hacer nada, el programa comprobará que

   - Realmente *PYTHONPATH* existe y incluye `~/lagrs/lib`. No importa si contiene otros directorios.

     Naturalmente, esto tiene que funcionar para cualquier usuario, esto es, el programa buscará el directorio `~/lagrs/lib` del usuario que ejecuta este programa (por ejemplo el profesor), no de tu usuario en particular.
   - Tus módulos realmente están en su sitio. Usa para ello la librería `os.path`. No la hemos visto en clase pero es muy sencilla, localiza documentación sobre su uso con cualquier buscador. Observa que aquí sí es necesario que el nombre del módulo comience por tu login, no por el del usuario que ejecuta este programa (por ejemplo el profesor).

   Si alguna de estas comprobaciones falla, el programa mostrará un error describiendo el problema y concluirá.

Práctica 2.6. Creación de un bot de telegram

1. Instala Telegram en tu móvil, en tu tablet o en tu portátil.

2. Crea un bot de telegram, con el nombre que quieras.

3. Ponle una foto al perfil del bot.

4. Escribe un programa con el nombre `~/lagrs/practica02/hola_telegram.py` que envíe un mensaje de tipo *hola mundo* a tu cliente Telegram y luego entre en un bucle infinito atendiendo a los mensajes que reciba (de cualquier usuario). Por cada mensaje recibido, envía una respuesta cualquiera al cliente y muestra todo en la pantalla del servidor, de forma legible (no el volcado en bruto del diccionario).

5. Usa *format* para componer la respuesta. No uses la concatenación de cadenas con el operador `+`

6. El token no estará en el fuente. El programa leerá el token desde el fichero `token.txt`, que estará en el directorio actual. (En este caso, el directorio `~/lagrs/practica02/`, pero no escribas este trayecto, que el script lea el fichero desde el directorio actual)

7. Si el fichero con el token no existe o no se puede leer, el programa mostrará un mensaje de error por *stderr* y morirá. O si lo prefieres, levantará una excepción que describa el problema.

Práctica 2.7. Recorrido de directorios

Escribe una función en Python 3, con el nombre que quieras en el programa que quieras, que reciba:

- Un nombre de directorio. (Que puede incluir la virgulilla representado el directorio *home*)
- Un intervalo de tiempo expresado como número (entero) de días.

La función recorrerá el directorio y todos sus subdirectorios, recursivamente y devolverá dos valores:

- Una lista conteniendo el nombre de todos los ficheros cuya fecha de modificación esté comprendida entre la fecha actual y el intervalo de tiempo indicado. Los nombres de los ficheros deben incluir el trayecto.
- Un *status*:
  - Si todo ha ido bien, será la cadena *ok*, en minúscula.
  - En otro caso, una cadena describiendo el problema que haya sucedido. La función debe notificar de esta forma que el directorio no existe o que no se puede leer, o que el ńumero de dias pedido es erróneo (no es un número entero positivo).

Por ejemplo, si el directorio es `~/lagrs` y el intervalo es 7, devolverá los nombres de los ficheros dentro de este directorio que hayan sido modificados en la última semana. Y como status, la cadena *ok*. Si no hay ningún fichero que cumpla los requisitos pedidos, simplemente la lista estará vacía.

Práctica 2.8. Bot de telegram (2)

Escribe un programa en Python 3 con el nombre `~/lagrs/practica02/recientes.py`. Contendrá el código para hacer un bot de telegram que ofrezca la funcionalidad de la práctica anterior al usuario. Los parámetros de entrada (nombre de directorio e intervalo de tiempo) se los debe preguntar al usuario. Tienes libertad para diseñar este inteface de usuario, con tal de que tenga forma de *diálogo*: el bot irá preguntando, el usuario responderá.

Tu script debe controlar el tamaño de la respuesta: si es excesivamente larga, truncarla. Fija este valor en tu código fuente, como te parezca adecuado (Telegram no adminte mensajes de más de 4K, pero seguramente deberías poner un límite más estricto)

Práctica 2.9. Intervalos extendidos. (OPTATIVA)

Si lo deseas, puedes mejorar la práctica anterior: añadiendo la posibilidad de fijar intervalos de tiempo que concluyan en cualquier momento, no solo la fecha actual. También especificar los instantes de tiempo de forma más detallada (hora, minuto y segundo). O incluir otros atributos en la búsqueda como tamaño, propietario, permisos, etc

Laboratorio de Administración y Gestión de Redes y Sistemas 2022-2023
Grado en Ingeniería Telemática
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Observaciones

- Crea el fichero `~/lagrs/practica03.txt`, que contendrá la memoria que escribas sobre este bloque de prácticas. En la primera línea, indica tu nombre y login.

Práctica 3.1. FHS

Busca en los exámenes de teoría resueltos de cursos anteriores todas las preguntas sobre FHS, *Filesystem Hierarchy Standard*, indica en qué convocatorias han aparecido y resume su contenido.

Práctica 3.2. Recode

Crea el directorio `~/lagrs/practica03/recode` y escribe dentro los ficheros que solicite este ejercicio

1. Genera 2 o 3 ficheros de texto plano, copiando y pegando desde cualquier página web. Ponles extensión .txt
2. Comprueba la codificación empleada en estos fichero
3. Comprueba la codificación empleada en tu máquina
4. Haz una copia de cada fichero en una codificación distinta. (latin1, alguna codificación unicode distinta a la codificación por omisión...)
   Ponle nombres añadiendo un sufijo que indique la codificación, entre el nombre de fichero y la extensión `.txt`. Ejemplo:

   ```
   quijote.latin1.txt   hans_reiser.utf16.txt
   ```

Práctica 3.3. netstat

Abre una sesión ssh entre tu ordenador y un puesto del laboratorio. Lanza netstat en tu ordenador y analiza el resultado. Lánzalo en el puesto y analiza el resultado

Práctica 3.4. tmux

El objetivo de esta práctica es que uses *tmux* para mantener demonios corriendo sin terminales abiertos, así como multiplexar sesiones.
Prepararás una sesión de tmux, con dos ventanas:

- Una con dos paneles, uno para el script *tictac* y otro para la orden *vmstat*.
- Otra con una única panel, para la orden top.

Hazlo según los siguientes pasos:

1. Empieza familiarizándote con tmux, probando los ejemplos de las transparencias. No hace falta que documentes esto en la memoria.
2. Crea el fichero `~/lagrs/practica03/tictacTULOGIN` y pega en él este script. (TULOGIN serán las primeras 4 letras tu nombre de usuario en el laboratorio).

```
#!/bin/bash
fichero_salida=/tmp/log.$USER.txt
while true
do
    sleep 1
    echo -n "tic"  >> $fichero_salida
    sleep 1
    echo " tac"  >> $fichero_salida
done
```

Pruébalo usando `tail -f` (que escribe en la salida estándar las últimas líneas de un fichero, y queda a la espera de que haya cambios en el fichero, para escribirlos también).

3. Busca una máquina linux del laboratorio a la que puedas entrar por ssh.
4. Copia `tictac` al directorio `/tmp/` de la máquina remota.
5. Entra por ssh en la máquina remota.
6. Haz lo necesario para invocar a *tictacTULOGIN* y que quede funcionando en una sesión de tmux.
7. Deja *tictacTULOGIN* corriendo y sal de la máquina remota.
8. Vuelve a entrar por ssh y recupera el control de la sesión de *tmux* con *tictacTULOGIN*.
9. Crea dentro de la ventana donde está *tmux* otro panel. Ejecuta en él `vmstat 2`.
10. Muévete entre un panel y otro.
11. Deja todo corriendo. Crea otra ventana y ejecuta en ella la orden `top`.
12. Sal de la máquina remota, dejando todo en marcha (tictacTULOGIN, vmstat y top).
13. Vuelva a entrar y comprueba que sabes recuperar todos los programas (los dos paneles de la primera ventana y el panel único de la segunda ventana)
14. Mata con `Crl C` los procesos de las tres ventanas. Cierra todas las ventanas. Comprueba (con ps) que no queda ninguna sesión de tmux. Cierra la sesión de ssh.

Práctica 3.5. Túnel ssh inverso

En este ejercicio probarás el uso de un túnel inverso de ssh, que te permitirá acceder desde cualquier lugar de internet a un servicio en una dirección privada, detrás de un NAT, sin necesidad de *port forwarding* (*abrir puertos*). Usarás tu host del laboratorio como proxy, que como sabes tiene una dirección IP pública (aunque para este ejercicio bastaría cualquier IP accesible para el cliente, podría ser por ejemplo una IP privada del laboratorio). También puedes hacer este ejercicio en casa, usando tu ordenador como proxy.

Usaremos un servicio de prueba, similar a un *holamundo*. Son un par de scripts en python, cliente y servidor. El servidor recibe un número arábigo y devuelve el número romano correspondiente. Colocarás este servicio detrás del NAT creado por VirtualBox, en una red privada. El proxy, *delante* del NAT permitirá acceder este servicio.

- Observa que si haces este ejercicio en casa, tu host no tiene una dirección IP pública, sino una IP privada creada por el NAT de tu router. Al colocar el servicio detrás del NAT de VirtualBox, estará detrás de dos NATs. Con este ejercicio, permitirás que el servicio esté disponible en la red de tu casa, no en la red creada por VirtualBox. En otras palabras, pasará de estar *tras dos NATs* a estar solo *tras un NAT*.

1. Descarga romanserver.py y romanclient.py en `~/bin`, dale permisos de ejecución y comprueba que el directorio `~/bin` está incluido en tu variable de entorno PATH.

2. Prueba romanserver.py y romanclient.py, en cualquier puerto TCP de tu host del laboratorio o de tu ordenador. Encontrarás sus instrucciones de uso al lanzarlos sin argumentos.

3. Usando vagrant, lanza la máquina virtual de la práctica 1 (vbox01).

4. Instala tmux en la máquina y abre varias ventanas (varios terminales)

5. Comprueba que puedes entrar por ssh desde vbox01 hasta tu ordenador.

   - Si estás en tu casa, necesitaras la dirección privada de tu máquina en la red privada que tu router ha creado en tu casa. Puedes usar *ifconfig* para averiguarla.

6. Para poder usar romanserver.py en vbox01, instala python.

7. Copia romanserver.py en cualquier directorio de vbox01 y lánzalo en el puerto 8000.

8. Usando un túnel inverso de ssh, haz que este servicio esté disponible en el puerto 9000 de tu ordenador.

9. Prueba el túnel usando romanclient.py en tu host.

Práctica 3.6 Cron

En esta práctica probarás el uso básico de cron. Los puestos físicos del laboratorio no tienen cron instalado, ya que son máquinas que no están siempre encendidas. Haz el ejercicio en uno de los puestos virtuales del laboratorio o en tu ordenador de casa.

Describe brevemente las tablas de cron que usas, en el fichero `~/lagrs/practica03.txt`

1. Comprueba que cron funciona: Programa en el host una tarea que ejecute cada minuto *touch* sobre el fichero `/tmp/test_cron_tulogin` (donde *tulogin* es tu login en el laboratorio). Consulta la fecha de este fichero, si todo está en orden, borra esta entrada de la tabla de cron

2. Escribe el script `~/lagrs/practica03/escribe_log` en bash o python que añada la cadena *probando cron*, junto con la fecha y la hora, al fichero `~/lagrs/log.txt` Como en cualquier log, debes añadir al final, sin borrar los mensajes precedentes.

   No le pongas extensión al script, ni `.sh` ni `.py`.

   Puedes usar el mandato de shell `date`

3. Programa una tarea que ejecute el script anterior cada minuto. Cuando veas que funciona, modifica la entrada para que se ejecute a las 9 de la mañana, de lunes a viernes.

Revisión de los nombres de los ficheros

Ejecuta `~mortuno/revisa practicas lagrs` para comprobar que los nombres de los programas son los correctos.

http://hdl.handle.net/10115/20117