Práctica 1: Administración de SSOO e Introducción a Python

Sistemas Distribuídos

Ejercicios de Comandos de Linux

1. Indique las particiones que requiere Linux para instalarse correctamente y explique para qué se utiliza cada una.
2. Indique la secuencia de órdenes/comandos para consultar las características de su equipo (tipo y tamaño del disco, y memoria RAM) y la distribución de las particiones que tiene actualmente. Explique la distribución de las particiones de su equipo (p. ej., su función y utilidad).
3. Razone si es posible redimensionar el espacio de las particiones garantizando la seguridad y consistencia de la información y en caso afirmativo, indique cómo lo haría.
4. Explique la diferencia entre una ruta relativa y una ruta absoluta. Si nos encontramos en el Escritorio (Desktop) del File System Hierarchy Standard (FHS) y queremos ir al directorio /lib, ¿cuál sería la forma más rápida de acceder? Indique la secuencia de comandos a utilizar.
5. Qué variable de entorno posibilita que los comandos básicos, ubicados en `/bin`, se puedan ejecutar desde cualquier directorio del sistema de ficheros sin tener que indicar la ruta absoluta.
6. Tras un cambio en la configuración de red, mediante línea de comandos ¿cómo harías para que los cambios tomen efecto sin necesidad de reiniciar el sistema?
7. Comente 3 comandos de superusuario que estime interesantes, indicando utilidad y exponiendo un ejemplo práctico.
8. Explique el concepto de grupo y la relación, si la hay, con el concepto de usuario.
9. Practique las operaciones necesarias en su máquina para dar respuesta a las siguientes preguntas:
    - ¿Cómo se llama el usuario que está usando la terminal?
    - ¿En qué grupo está dicho usuario?

- ¿Cuántos usuarios existen en el equipo?
- Liste los grupos del sistema.
- Liste los usuarios que pertenecen al grupo sudo.
- Cree un usuario llamado `sd` que tenga como directorio de inicio `/home/sd`.
- Muestre el `UID` que tiene asignado y a qué grupo pertenece inicialmente al ser creado.
- Cree un grupo llamado `grupo_sd`.
- Haga que el usuario `sd` pertenezca al nuevo grupo.
- ¿Cómo puede ver/comprobar que pertenece a ambos grupos?
- Cambia el directorio de inicio del usuario creado, para que pase a ser `/home/comunes`.

Ejercicios de Python

1. Estudie y comprenda el funcionamiento de los ejemplos que puede encontrar en el Campus Virtual en la carpetas:
    - "Basic examples" (ver Anexo: Introducción a Python).
    - "Array examples" (ver Anexo: Conjuntos, Arrays, Diccionarios, Listas y Tuplas en Python).
    - "Strings and Loops examples" (ver Anexo: Cadenas y Bucles en Python).
    - "File examples'' (ver Anexo: Ficheros y el Módulo os en Python).
2. Existe una variable de entorno que nos permite ejecutar comandos ubicados en /bin, sin necesidad de escribir la ruta completa, ni situarnos en el directorio en cuestión, muestra en pantalla el valor que contiene. ¿Qué módulo permite ver esto usando `import nombreDelModulo` al principio del script?
3. Haga un script en Python que cree una copia de un fichero cualquiera. Puede implementar una función propia o utilizar una existente. A continuación, utilice la librería/módulo necesario para comprobar que los ficheros anteriores son iguales.
4. Implemente la función `accum(x, y, z)` para que devuelva la suma de aquellos parámetros que incluyan un número par. Por ejemplo, la invocación `accum(5, 4, 2)` deberá devolver como resultado 6. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si alguno de los tres argumentos `x, y`, o `z` no es un valor entero.
5. Implemente la función `list_add(mylist, e)` para que añada el elemento `e` a la lista `mylist` y devuelva la lista resultante. Por ejemplo, la invocación de `list_add([5, 2], 4)` deberá devolver como resultado `[5, 2, 4]`. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si el elemento `e` es nulo `(None)`.
6. Implemente la función `list_del(mylist, e)` para que elimine la primera ocurrencia del elemento `e` de la lista `mylist` y devuelva la lista resultante. Por ejemplo, la invocación `list_del([5, 2, 4], 2)` deberá devolver como resultado `[5, 4]`. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si el elemento e es nulo `(None)` o la lista `mylist` está vacía.

7. Implemente la función `dict_add(mydict, t)` para que añada la tupla (clave, valor) `t` pasada por parámetro al diccionario `mydict`. Por ejemplo, la invocación `dict_add({1: 'manzana'}, {2, 'fresa'})` deberá devolver como resultado `{1: 'manzana', 2: 'fresa'}`. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si el elemento `t` es nulo `(None)` o no es una tupla de dos elementos.
8. Implemente la función `prime(a, b)` para que devuelva una lista con los números primos en el intervalo cerrado [a, b]. Por ejemplo, la invocación `prime(2, 10)` deberá devolver como resultado `[2, 3, 5, 7]`. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si los parámetros `a` o `b` no son enteros o son nulos `(None)`.
9. Implemente la función `get_file_info(filename)` para que devuelva una tupla con el tamaño en bytes del fichero, cuyo nombre se indica como parámetro `filename`, en la primera posición, y una lista con las palabras acabadas con el carácter 's' que contenga el fichero, en segunda posición. Por ejemplo, la invocación `get_file_info('mifichero.txt')`, suponiendo que `'mifichero.txt'` contiene el texto “La casa está pintada en muchos colores", devolverá la tupla `(39, ['muchos', 'colores'])`. Se deberán generar las siguientes excepciones en caso de ser necesario:
    - `TypeError` si el parámetro `filename` no es una cadena o es nulo `(None)`.
    - `OSError` si el fichero indicado no existe.
10. Implemente una función en Python que realice la intersección de dos listas. Tenga en cuenta que no puede haber elementos repetidos.
11. Implemente una función en Python que realice la unión de dos listas. Tenga en cuenta que no puede haber elementos repetidos.
12. Implemente una función `copiar(origen,destino)` que copie el contenido del fichero `origen`, en el fichero `destino` (usando `open()`).
13. Imprima por pantalla el listado de directorios de inicio de los usuarios que hay en el sistema (p. ej., /home/root , /home/osboxes, . . .). Pista: hay un fichero con esta información. También existe una función muy interesante en Python llamada `split`, que convierte de string a lista.
14. Implemente un script en Python, utilizando el módulo `os`, que liste todos los ficheros del directorio actual junto a su tamaño en bytes. Por último, el script mostrará la suma total del tamaño de los ficheros del directorio. Se deben incluir, además, los ficheros existentes en subdirectorios.

15. Realice un script en Python que mueva al directorio actual, todos los archivos contenidos en subdirectorios del mismo. Tenga en cuenta que un directorio puede contener a su vez otros directorios y que la longitud en la jerarquía del árbol no está definida y por tanto, el script debe funcionar para todos los casos.

16. Realizar un script en Python que imprima por pantalla el directorio de trabajo actual, junto a la lista de ficheros existentes en dicho directorio. Posteriormente, el mismo script permitirá al usuario renombrar un fichero. Para ello solicitará al usuario el nombre del fichero que quiere renombrar y el nuevo nombre que quiere darle. Se deben gestionar correctamente las posibles excepciones que puedan darse en la ejecución del script.

17. Realizar un script en Python que combine todos los ficheros de texto (.txt) existentes en el directorio de trabajo actual en un único fichero de texto, llamado "`union.txt`". Tanto los ficheros con una extensión distinta, como los que se encuentren en subdirectorios, deberán ignorarse.

Anexo

Introducción a Python

Estudie los siguientes capítulos del libro Python® Notes for Professionals:

- Chapter 2: Python Data Types
- Chapter 3: Indentation
- Chapter 4: Comments and Documentation
- Chapter 9: Simple Mathematical Operators
- Chapter 11: Boolean Operators
- Chapter 13: Variable Scope and Binding
- Chapter 14: Conditionals
- Chapter 15: Comparisons
- Chapter 29: Basic Input and Output
- Chapter 33: Functions
- Chapter 74: The Print Function
- Chapter 88: Exceptions

Conjuntos, Arrays, Diccionarios, Listas y Tuplas en Python

Estudie los siguientes capítulos del libro Python ® Notes for Professionals:

- Chapter 8: Set
- Chapter 17: Arrays
- Chapter 19: Dictionary
- Chapter 20: List
- Chapter 22: List slicing (selecting parts of lists)
- Chapter 28: Tuple

Cadenas y Bucles en Python

Estudie los siguientes capítulos del libro Python® Notes for Professionals:

- Chapter 16: Loops
- Chapter 40: String Formatting

- Chapter 41: String Methods

Ficheros y el Módulo os en Python

Estudie los siguientes capítulos del libro Python® Notes for Professionals:

- Chapter 30: Files & Folders I/O
- Chapter 31: os.path
- Chapter 51: The os Module