PRÁCTICA 1

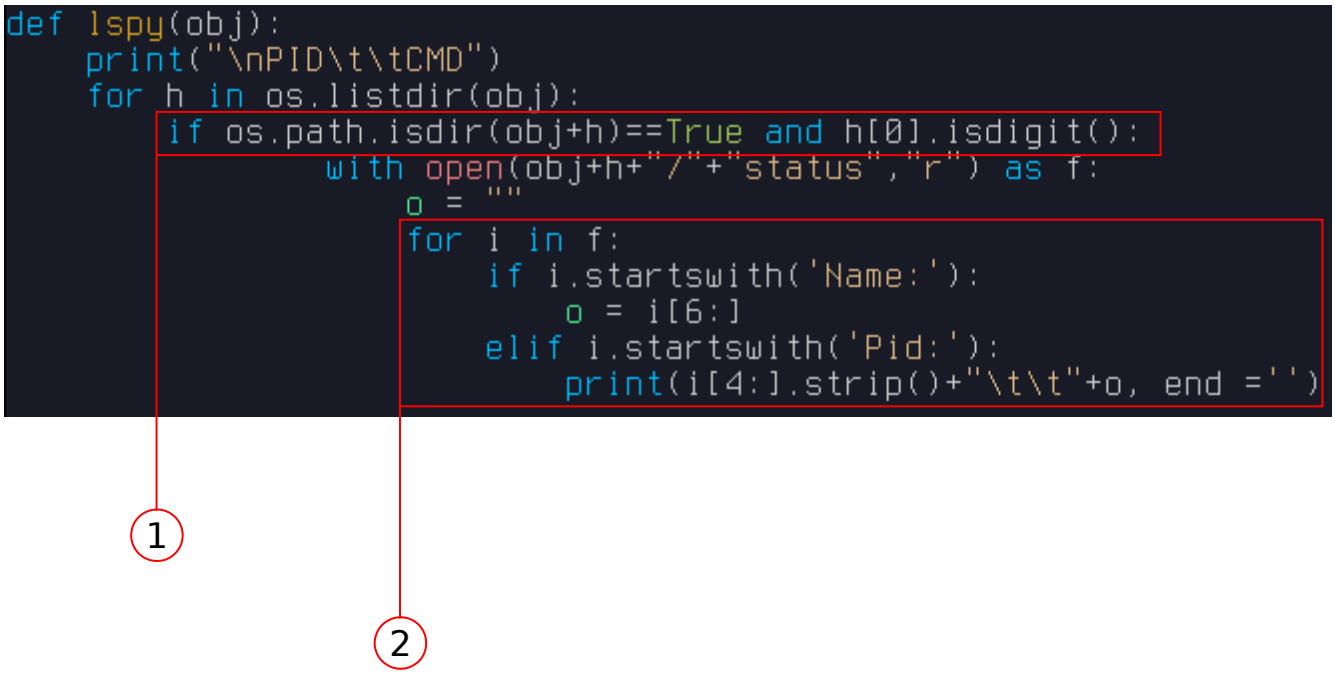
Mendoza Flores Erick Gabriel
Nava Vivas Ana Paola
2CM8

Introducción.

El uso de **python** para generar scripts en los sistemas Unix es de gran ayuda; simplifica el uso de operaciones a través de un lenguaje entendible y sencillo, a diferencia de **bash**, que si bien no es complejo, añade barreras lingüísticas que obstaculizan la resolución de un problema. **Python**, en cambio elimina dichos obstáculos y es incluso capaz de ejecutar subprocesos de **bash** y añadir utilidad a sus entradas y salidas.

En esta práctica, utilizaremos el lenguaje **python3** para describir de manera general los procesos del sistema y sus características principales como uso de memoria, tipo de procesador, etc., tomando como referencia el directorio */proc*.

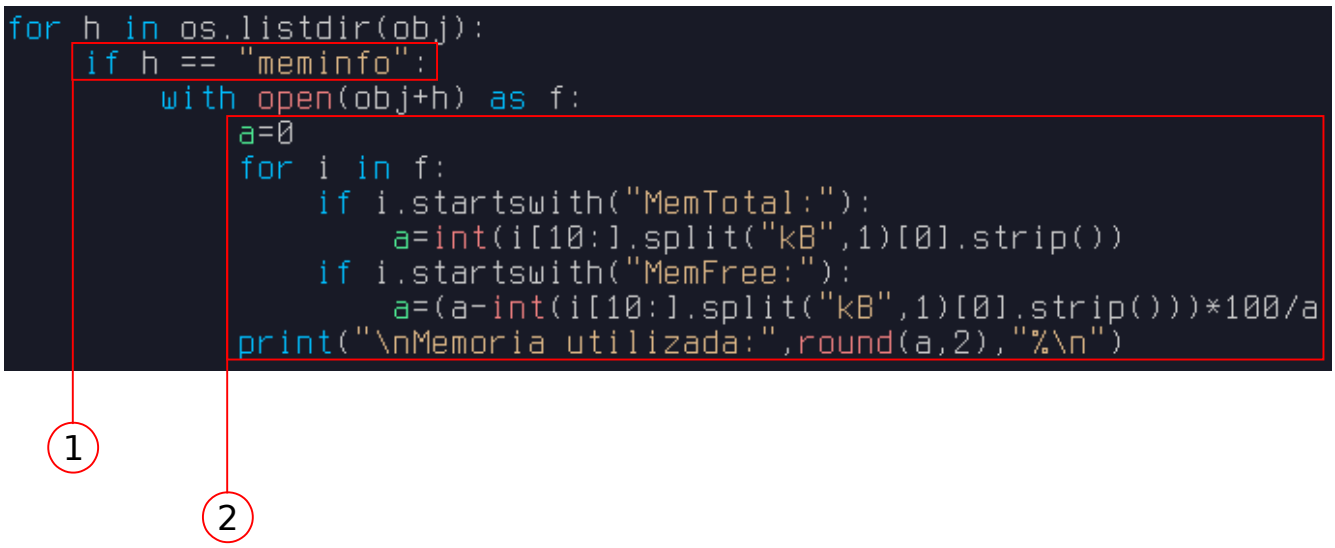
Desarrollo.



En este programa utilizamos la variable *obj* **como el directorio** */proc*.

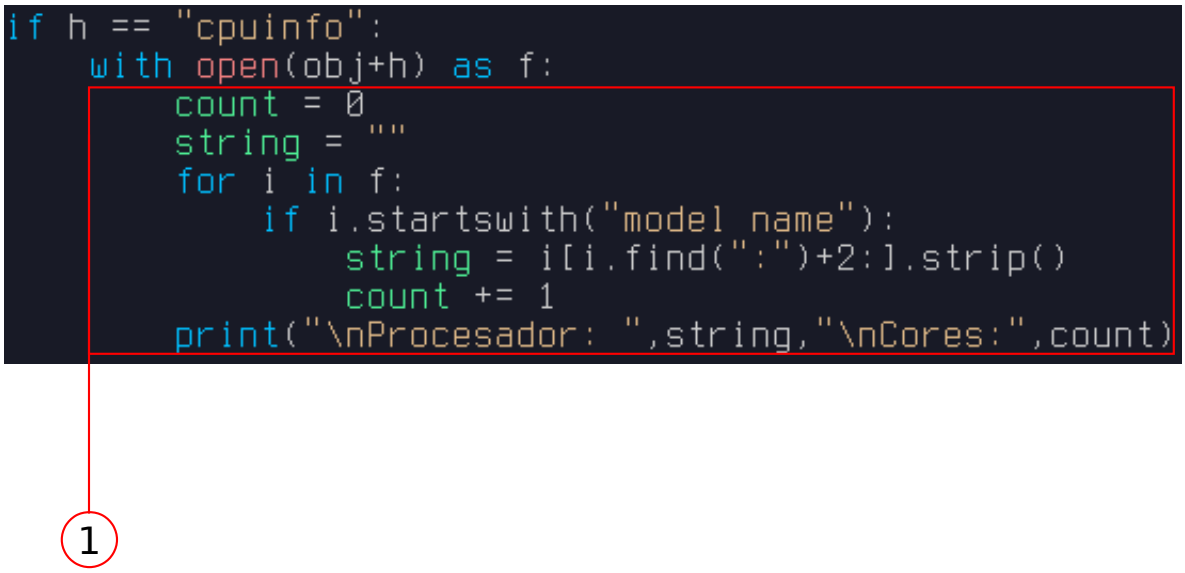
1. De la importación de la paquetería *os*, que se encarga de manipular funciones del sistema operativo, utilizamos la función de *isdir()* que proviene del módulo *path*, para determinar si los objetos dentro de la lista que produce la función *listdir()* son directorios. Sabemos de antemano, por la inspección del directorio que todos los procesos están listados ahí, y que además se encuentran listados por su id de proceso, entonces, también utilizamos como condición que el directorio necesariamente comience con un dígito decimal.

2. Con el archivo *status* dentro de cada directorio abierto, extraemos la información de nombre de proceso (*Name:*)y su id de proceso (*Pid*) a través de la función de cadena *startswith()*, que regresa un valor booleano si la cadena que ejecuta la función comienza con la cadena que recibe la función como argumento, y a través de una sencilla asignación damos formato a la impresión del proceso en pantalla para simular el proceso del comando **bash** *ps*.



1. En este paso abrimos el archivo llamado *meminfo*, responsable de almacenar el estado de la memoria RAM y del swap.

2. Iterando de la misma manera que en los archivos de *status*, buscamos las cadenas que comiencen con *MemTotal* y *MemFree*, y asignamos la primera cadena a una variable, usando las funciones de *split()* y *strip()* para cortar la cadena a partir del noveno carácter y hasta *kB*, en una lista de dos elementos y en eliminar todo el espacio en blanco antes y después de la cadena resultante en la posición 0 respectivamente, y el segundo valor, tras pasar por el mismo proceso que el primero se reasignan a la variable original aplicando una regla de tres para determinar el porcentaje de memoria utilizada por el sistema.



1. Siguiendo los pasos del ejemplo anterior separamos la información del modelo del procesador, pero contamos las veces que aparece (ya que el archivo *cpuinfo* muestra la información de cada núcleo individual) y al final imprimimos toda la información.

Conclusiones.

*Utilizando la plataforma de **python**, nos dimos cuenta de que era mucho más sencillo realizar tareas de sistema y hacerlas con mayor alcance gracias a estos scripts. La mayor dificultad encontrada fue en el momento de buscar los módulos requeridos para **python** y la búsqueda de los archivos adecuados para la lectura.*

- Ana Paola Nava Vivas

*El programa utilizado acortó el tiempo de creación de scripts, ya que el uso del módulo **os** de **python** recrea de manera amigable los comandos de **bash** y lo hace comprensible. El producto final consta de un script más limpio, legible y altamente configurable.*

- Erick Gabriel Mendoza Flores