



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE  
ESCUELA DE INGENIERÍA  
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN  
IIC2233 - PROGRAMACIÓN AVANZADA

# Actividad 11

1º semestre 2018

31 de mayo

## I/O: archivos y bytes

### Introducción

Tú y tu grupo de estudio compraron los apuntes más valiosos de Programación Avanzada en China. Sin embargo, para que nadie lo sepa, te los han enviado codificados y escondidos entre muchas carpetas. Afortunadamente tienes las instrucciones para decodificarlos. Si sigues las indicaciones de esta actividad, al finalizarla deberías ser capaz de visualizar correctamente los apuntes provenientes de China.

### Parte 1: Encontrar Archivos (1.2 puntos)

Para poder resolver el primer problema, debes crear un escenario de múltiples archivos en tu computador, lo que podrás hacer automáticamente ejecutando el *script* `esconder.py`. El archivo `esconder.py` creará  $N$  carpetas contenedoras de  $M$  carpetas cada una. En alguna de estas  $M \times N$  carpetas se esconderán aleatoriamente dos archivos corruptos: `marciano64.png` y `marcianozurdo.pep`. La ubicación de estos archivos cambiará a la hora de corregir. Tu primera misión es encontrar automáticamente el *path* a cada uno de esos archivos corruptos, para poder leer esos archivos posteriormente.

Si quieres saltarte esta parte y seguir con la actividad, puedes asumir que los *paths* a los archivos corruptos son los entregados en la carpeta de esta actividad. Sin embargo, no tendrás el puntaje de esta parte.

### Parte 2: Algoritmos (2.4 puntos)

Luego de haber encontrado los archivos corruptos, deberás crear dos algoritmos para decodificar los archivos entregados. Los algoritmos son los siguientes:

#### 1. Algoritmo base64

Este algoritmo se lo debes aplicar a los bytes del archivo `marciano64.png`.

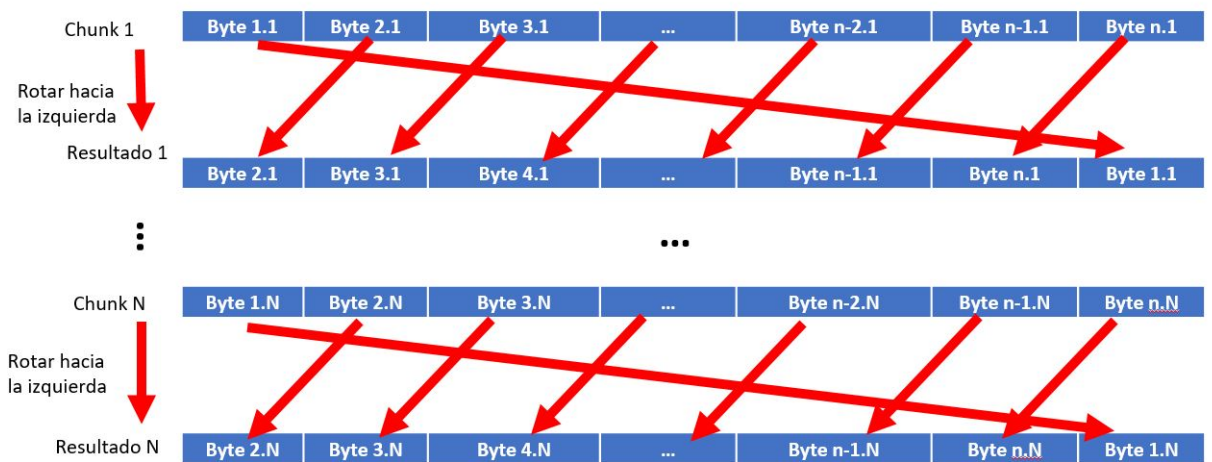
- Leer los *bytes* del archivo `marciano64.png`.
- Transformar cada *byte* a su carácter ASCII correspondiente.
- Cada carácter obtenido se cambia por su valor base64 decimal. Por ejemplo, “E”  $\rightarrow$  4 tal y como se indica en la tabla.

Carácter	Valor Base64
A – Z	0 – 25
a - z	26 – 51
0 – 9	52 – 61
+	62
/	63

- Los valores decimales obtenidos se convierten a su equivalente en binario de 6 *bits*. Si el número queda menor a 6 *bits*, se deben agregar ceros a la izquierda hasta completar 6.
- Los conjuntos de 6 *bits* se concatenan, formando una gran cadena de *bits*.
- La gran cadena de *bits* obtenidos en el paso anterior se divide en grupos de 8 *bits*.
- Finalmente los números binarios de 8 *bits* se convierten en sus equivalentes decimales.

## 2. Algoritmo Rotar hacia la izquierda (*rotate left*)

- A diferencia del algoritmo base64, que se aplica directo a todos los *bytes* del archivo, este algoritmo se aplica a una sub-secuencia de *bytes*, que denominaremos *chunk*. El tamaño de cada *chunk* se definirá a la hora de juntar los archivos.
- El algoritmo consiste en que a cada *chunk* se le rota un *byte* a la izquierda, es decir, todos avanzan una posición y el primero pasa a la última posición.



## Parte 3: Juntar los archivos (2.4 puntos)

El archivo resultante debe llamarse **resultado.png**. Para comprobar el correcto funcionamiento de tu actividad, este archivo debe visualizarse correctamente como imagen.

Para crear este archivo de resultado, debes juntar *chunks* de ambos archivos en un *bytearray*. Estos se agregarán al *bytearray* de manera intercalada partiendo por el archivo **marcianozurdo.pep**, de la siguiente manera:

*chunk* 1: Aplicar algoritmo 2 al primer *chunk* del archivo **marcianozurdo.pep**, y agregar el resultado al *bytearray*.

*chunk* 2: Agregar el primer *chunk* del archivo `marciano64.png`, y agregar el resultado al *bytearray*.

*chunk* 3: Aplicar algoritmo 2 al segundo *chunk* del archivo `marcianozurdo.pep`, y agregar el resultado al *bytearray*.

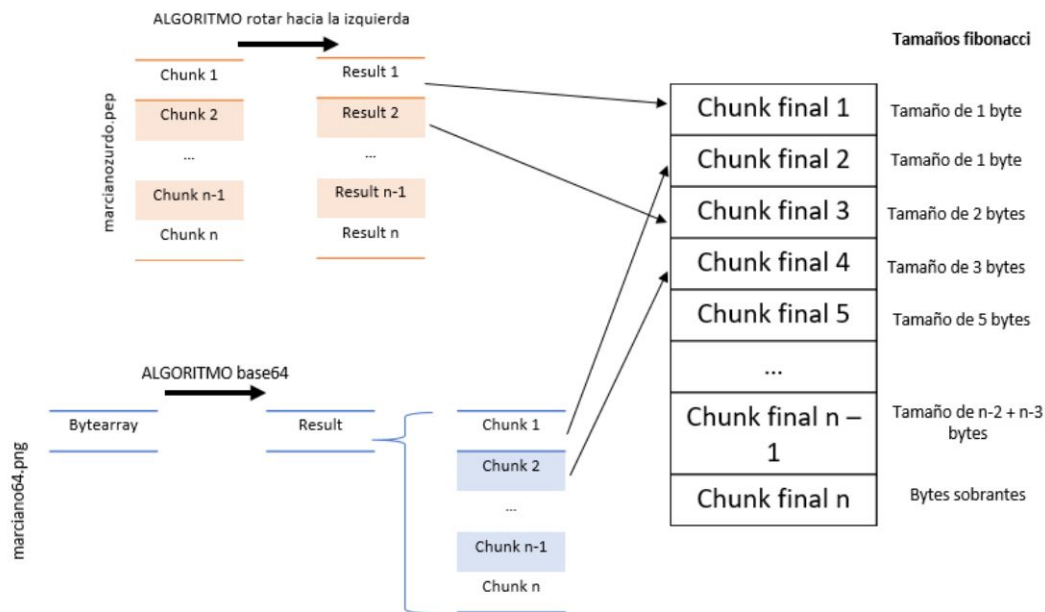
*chunk* 4: Agregar el segundo *chunk* del archivo `marciano64.png`, y agregar el resultado al *bytearray*.

*chunk* n: ... Iterativamente hasta que no queden *bytes* por leer.

El tamaño de cada *chunk* del archivo `resultado.png` viene determinado por la sucesión de Fibonacci partiendo del 1. Por lo tanto, el *chunk* 1 será de 1 *byte*, luego el *chunk* 2 será de 1 *byte*, el *chunk* 3 será de 2 *bytes*, así sucesivamente según la sucesión de Fibonacci, que se define de la siguiente manera<sup>1</sup> <sup>2</sup>:

$$1, 1, 2, 3, 5, 8, \dots, n_1, n_2, n_1 + n_2, \text{resto bytes}$$

En la siguiente imagen se muestra un resumen de lo que deben hacer:



## Notas

Deben tener en cuenta varias cosas para trabajar con los números binarios:

- El comando `bin(número)` de Python transforma un número en su equivalente binario, sin embargo entrega un "0b" a la izquierda del resultado que hay que remover.
- La función `int(binario, 2)` transforma el número binario a `int`.
- Para generar los dígitos de Fibonacci se recomienda el uso de **generadores**.

Además:

<sup>1</sup>El largo del último *chunk* puede ser menor al tamaño indicado por la sucesión de Fibonacci

<sup>2</sup>Cuidado: Notar que los tamaños de los *chunks* son para el archivo final. Eso significa, por ejemplo, que el segundo *chunk* que se extraiga de `marciano64.png` debe ser de tamaño 3.

- Se descontará puntaje (**0.2**) si suben a GitHub alguno de los archivos entregados o las carpetas generadas con `esconder.py`. Se recomienda fuertemente usar un `.gitignore`.
- Para pasar de ASCII a carácter se puede usar el comando `chr()` de Python.
- Está prohibido usar la librería `base64`
- Pueden ocupar la librería `os` para recorrer las carpetas.

## Requerimientos

- (1.2 pts): Encontrar archivos
  - (1.2 pts): Se encuentran los archivos perdidos.
- (2.8 pts): Algoritmos
  - (0.8 pts): Se implementa el algoritmo rotar hacia la izquierda
  - (1.2 pts): Se implementa el algoritmo base64
  - (0.4 pts): Se aplica de forma correcta el algoritmo rotar hacia la izquierda.
  - (0.4 pts): Se aplica de forma correcta el algoritmo base64
- (2.0 pts): Juntar los archivos.
  - (0.6 pts): Se determina correctamente el tamaño de los *chunks*.
  - (0.6 pts): Se concatenan de manera correcta los *chunks*.
  - (0.8 pts): Se obtiene la imagen final.

## Entrega

- **Lugar:** En su repositorio de GitHub en la **carpeta** `Actividades/AC11/`
- **Hora:** 16:30