

Aux 1: Introducción

Por Gabriel Flores y Joaquin Zepeda
EL4203 Programación Avanzada
Profesor: Alberto Castro

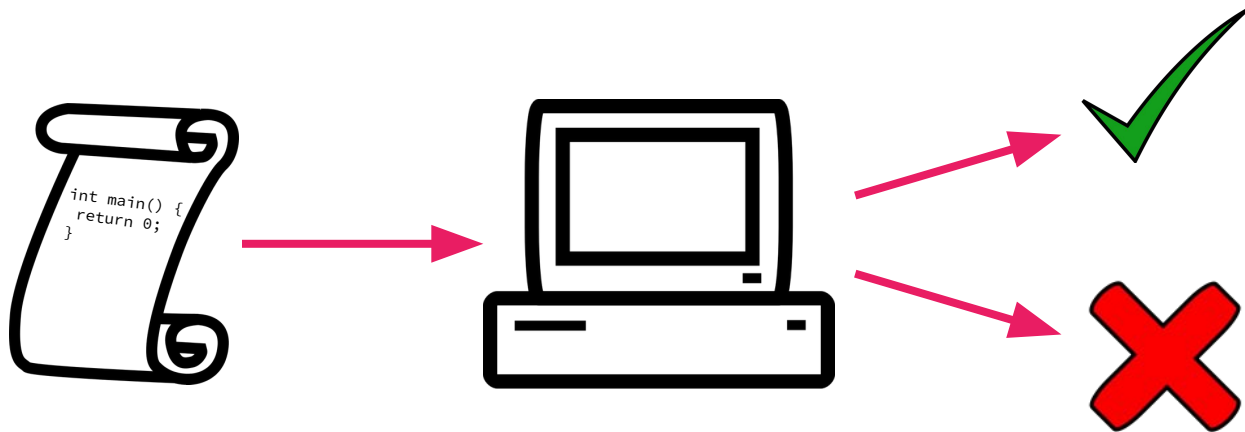
Quiénes somos

Punteo

— — —

- Ejecución “python ejemplo.py”
- Ejecución “gcc ejemplo.c -o ejemplo && ./ejemplo”
- Lenguajes interpretados vs compilados
- Bytecode, SOs, lenguaje máquina, proce
- Ejecución “g++ ejemplo.cpp -o ejemplo && ./ejemplo”
- Problemas propuestos
- Soluciones a los problemas propuestos

¿Qué pasa cuando ejecutamos un programa?



```
import sys
```

```
"""
```

```
ejemplo.py
```

Programa interactivo que imprime la suma de dos enteros.

```
"""
```

```
def run():
```

```
    if len(sys.argv) != 3:
```

```
        print("Uso: %s <num> <num>" % sys.argv[0], file=sys.stderr)
```

```
        exit(1)
```

```
    a = int(sys.argv[1])
```

```
    b = int(sys.argv[2])
```

```
    print(a+b)
```

```
if __name__ == '__main__':
```

```
    run()
```

```
import sys
```

```
"""
```

```
ejemplo.py
```

```
Programa interactivo que imprime la suma de dos enteros.
```

```
"""
```

```
def run():
```

```
    if len(sys.argv) != 3:
```

```
        print("Uso: %s <num> <num>" % sys.argv[0], file=sys.stderr)
```

```
        exit(1)
```

```
    a = int(sys.argv[1])
```

```
    b = int(sys.argv[2])
```

```
    print(a+b)
```

```
if __name__ == '__main__':
```

```
    run()
```

```
user$ ls
```

```
ejemplo.py
```

```
user$ python ejemplo.py
```

```
Uso: ejemplo.py <num> <num>
```

```
user$ python ejemplo.py 23
```

```
Uso: ejemplo.py <num> <num>
```

```
user$ python ejemplo.py 23 19
```

```
42
```

```
user$
```

```
#include <stdio.h>
#include <stdlib.h> // atoi
/*
ejemplo.c
```

```
Programa interactivo que imprime la suma de dos enteros.
*/
```

```
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Uso: %s <num> <num>\n", argv[0]);
        return 1;
    }
    int a = atoi(argv[1]);
    int b = atoi(argv[2]);
    printf("%d\n", a+b);
    return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h> // atoi
/*
ejemplo.c
```

```
Programa interactivo que imprime la suma de dos enteros.
*/
```

```
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Uso: %s <num> <num>\n", argv[0]);
        return 1;
    }
    int a = atoi(argv[1]);
    int b = atoi(argv[2]);
    printf("%d\n", a+b);
    return 0;
}
```

```
user$ ls
ejemplo.c
user$ gcc ejemplo.c -o ejemplo
user$ ls
ejemplo ejemplo.c
user$ ./ejemplo
Uso: ejemplo <num> <num>
user$ ./ejemplo 23
Uso: ejemplo <num> <num>
user$ ./ejemplo 23 19
42
user$
```

Ejemplo código c++

— — —

```
#include <iostream>
```

```
using namespace std;
```

```
/* g++ ejemplo.cpp -o ejemplo && ./ejemplo <num> <num>
```

```
ejemplo.cpp
```

```
Programa que imprime la suma de dos enteros.
```

```
*/
```

```
int main(int argc, char **argv){
```

```
    int integer1{stoi(argv[1])};
```

```
    int integer2{stoi(argv[2])};
```

```
    int sum = integer1 + integer2; //o int sum{integer1 + integer2};
```

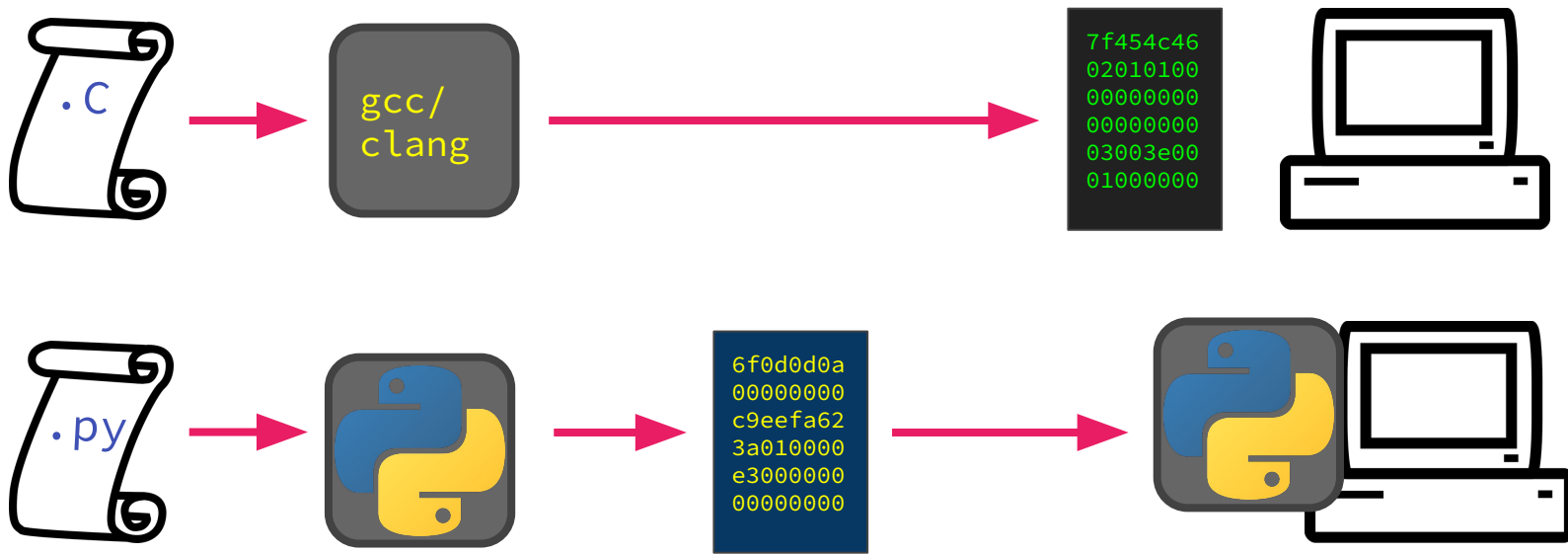
```
    cout << sum << endl;
```

```
    return 0;}
```

0000000000001159 <main>:

```
1159:      55          push    %rbp
115a:      48 89 e5     mov     %rsp,%rbp
115d:      48 83 ec 20  sub     $0x20,%rsp
1161:      89 7d ec     mov     %edi,-0x14(%rbp)
1164:      48 89 75 e0  mov     %rsi,-0x20(%rbp)
1168:      83 7d ec 03  cmpl    $0x3,-0x14(%rbp)
116c:      74 2c        je      119a <main+0x41>
116e:      48 8b 45 e0  mov     -0x20(%rbp),%rax
1172:      48 8b 10     mov     (%rax),%rdx
1175:      48 8b 05 c4 2e 00 00 mov     0x2ec4(%rip),%rax
117c:      48 8d 0d 81 0e 00 00 lea     0xe81(%rip),%rcx
1183:      48 89 ce     mov     %rcx,%rsi
1186:      48 89 c7     mov     %rax,%rdi
1189:      b8 00 00 00 00 mov     $0x0,%eax
118e:      e8 ad fe ff ff call    1040 <fprintf@plt>
1193:      b8 01 00 00 00 mov     $0x1,%eax
1198:      eb 4f        jmp     11e9 <main+0x90>
119a:      48 8b 45 e0  mov     -0x20(%rbp),%rax
119e:      48 83 c0 08  add     $0x8,%rax
11a2:      48 8b 00     mov     (%rax),%rax
11a5:      48 89 c7     mov     %rax,%rdi
11a8:      e8 a3 fe ff ff call    1050 <atoi@plt>
```

¿Qué pasa cuando ejecutamos un programa?

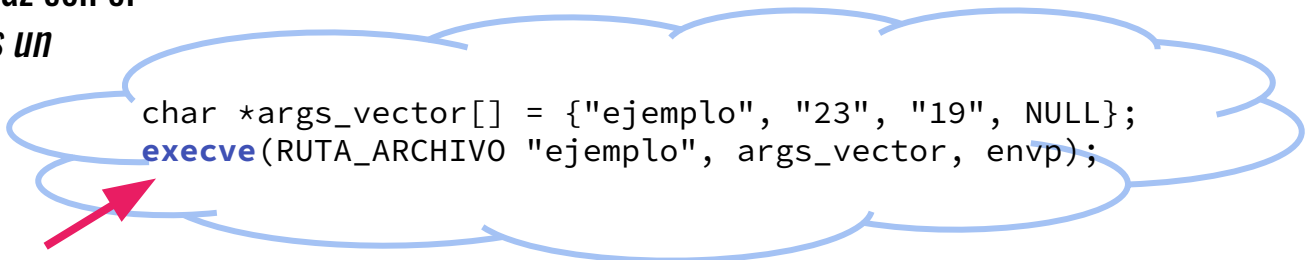


¿Qué pasa cuando ejecutamos un programa?

— — —

```
user$ ./ejemplo 23 19
```

Nuestra interfaz con el
sistema *ya es un
programa!*

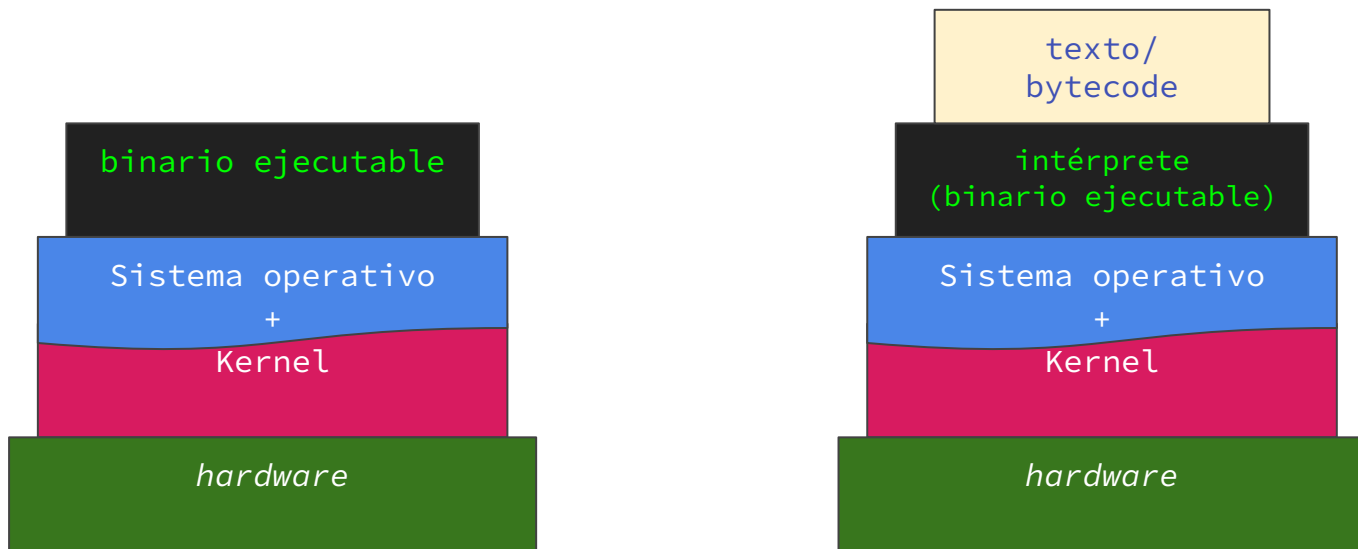


```
char *args_vector[] = {"ejemplo", "23", "19", NULL};  
execve(RUTA_ARCHIVO "ejemplo", args_vector, envp);
```

Depende de cada
sistema

¿Qué pasa cuando ejecutamos un programa?

— — —



Cómo ejecutar código C++ en Google Colab - 1

— — —

Paso 1: escribir un código C++ para ejecutar, para esto se puede subir el archivo a la sesión o utilizar el comando mágico `%%writefile nombre_archivo.cpp` para escribir un archivo dentro de Colab.

Paso 2: compilar y ejecutar el programa, para esto utilizaremos `%script bash`, una celda mágica de Colab que permite escribir comandos para la shell. Luego en esa celda se agregan los comandos para compilar y ejecutar respectivamente:

- Compilar: `g++ nombre_del_programa.cpp -o ejemplo`
- Ejecutar: `./ejemplo`

Cómo ejecutar código C++ en Google Colab - 2

— — —

```
✓ [2] %%writefile hola_mundo.cpp
1 s

#include <iostream>
using namespace std;

int main() {
    cout << "Hola mundo en C++" << endl;
    return 0;
}

Overwriting hola_mundo.cpp

✓ [22] %%script bash
0 s

g++ hola_mundo.cpp -o output && ./output

Hola mundo en C++
```

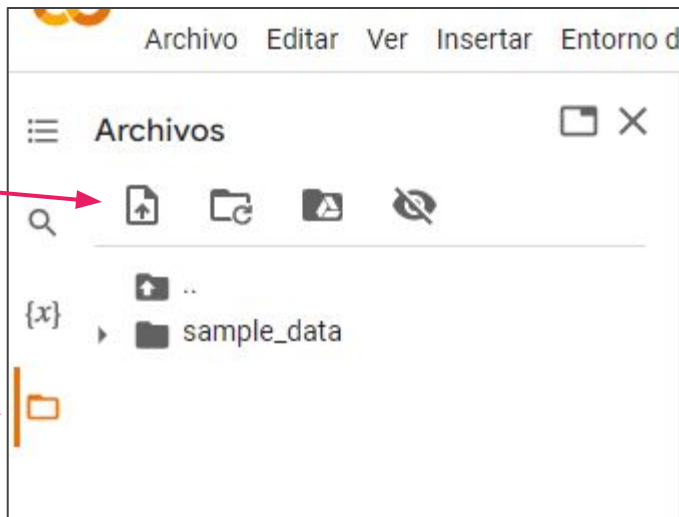
Paso 1: escribiendo el código en el mismo colab.

Paso 2

Cómo ejecutar código C++ en Google Colab - 3

Acá permite subir archivos

Archivos



Una vez subido y actualizar, aparecerá el nombre del archivo y podrá utilizarse el Paso 2.

→ Se recomienda revisar el notebook [Tutorial C++.](#)

Problema propuesto A: Capicúa de un número

Se define capicúa cualquier número que se lee igual de izquierda a derecha, por ejemplo: 121, 1001001, etc. Por otro lado existen otros números que no son capicúas pero cumplen una propiedad:

$$\text{número_no_capicúa} + \text{su_reverso} = \text{capicúa_del_número}$$

Nota/hint: En caso de que el resultado no sea un número capicúa se vuelve a sumar el reverso hasta que lo sea.

Ejemplo capicúa número 57

El número 57 no es un número capicúa por lo que se utiliza la segunda definición:

$$57 + 75 = 132$$

(Como sigue sin ser capicúa se vuelve a sumar el inverso)

$$\rightarrow 132 + 231 = 363$$

Diremos que el 363 es el capicúa del número 57.

Se pide:

Genere 2 funciones:

- `es_capicua(a)`: función que retorna si el número es un capicúa o no. Debe retornar `True` o `False` dependiendo del caso.
- `encontrar_capicua(a)`: debe retornar el valor del capicúa, según las definiciones.

Estas funciones reciben como parámetro un entero. Ejemplos:

- `encontrar_capicua(121)` retorna 121
- `encontrar_capicua(57)` retorna 363

Solución: Capicúa de un número

— — —

```
def es_capicua(a):  
    aux=''   
    largo=len(str(a))  
    string_a=str(a)  
    for i in range(largo):  
        aux+=string_a[largo-i-1]  
  
    if int(aux)==a:  
        return True  
    else:  
        return False
```

```
def invertido(a):  
    aux=''   
    largo=len(str(a))  
    string_a=str(a)  
    for i in range(largo):  
        aux+=string_a[largo-i-1]  
    return int(aux)  
  
def calcular_capicua(a):  
    if es_capicua(a)==True:  
        return a  
    else:  
        while True:  
            suma=a+invertido(a)  
            if(es_capicua(suma)==True):  
                break  
            a=suma  
        return suma
```

Solución más corta: Capicúa de un número

— — —

```
def es_capicua(a):  
    s = str(a)  
    n = len(s)  
    for i in range(n//2):  
        if s[i] != s[n-1-i]:  
            return False  
    return True
```

```
def encontrar_capicua(a):  
    while not es_capicua(a):  
        a += int(str(a)[::-1])  
    return a
```

Problema propuesto B: Aplanar una lista

Se dice que una lista está “plana” si no contiene ninguna otra lista anidada. Programe la función `aplanar(ls)` que aplane una lista:

```
aplanar([[‘a’, [‘b’]], ‘c’]) -> [‘a’, ‘b’, ‘c’]
```

```
aplanar([1,2,[3,[4,[[]],5],6], 7]) -> [1,2,3,4,5,6,7]
```

Nota: Una lista puede estar anidada *arbitrariamente profundo*.

Solución: Aplanar una lista

— — —

```
def extender(la, lb):  
    for x in lb:  
        la.append(x)  
  
def aplanar(ls):  
    new_ls = []  
    for x in ls:  
        if type(x) == list:  
            # new_ls.extend(aplanar(x))  
            extender(new_ls, aplanar(x))  
        else:  
            new_ls.append(x)  
    return new_ls
```