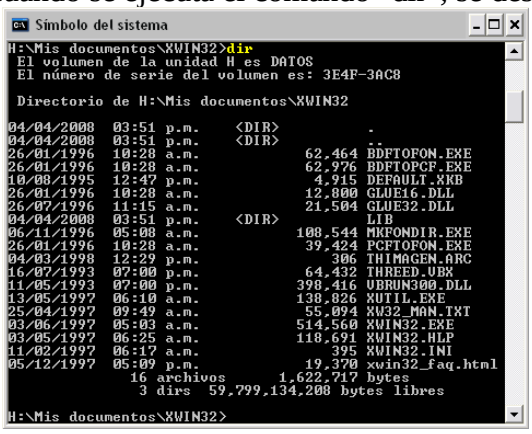
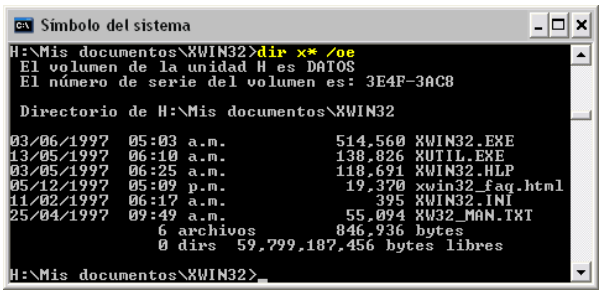
Argumentos en la línea de comandos (CLI)

Fuente 1: https://agora.pucp.edu.pe/inf2170681/11.htm
Fuente 2: https://www.eumus.edu.uy/eme/ensenanza/electivas/python/2014/CursoPython_clase07.html

Cuando se invoca un programa desde la línea de comandos del sistema operativo, en general, se observa que el programa se ejecuta directamente.
Por ejemplo, en MS Windows cuando se ejecuta el comando "dir", se despliega:



Ocurre, además que los programas pueden cambiar el modo en que se ejecutan si es que admiten parámetros en la línea de comandos.
Este es el caso de la orden "dir", que permite colocarlos; por ejemplo, la orden puede darse como "dir x*" o "dir x* /oe". En el primer caso aparecerán sólo los archivos cuyos nombres empiecen con la letra "x" y en el segundo los archivos que empiecen con "x" pero ordenados por su extensión.
Como se ve en las figuras siguientes.



Para que un programa pueda comportarse de ese modo, su módulo principal debe ser capaz de recibir parámetros.

Argumentos de la función main en C++

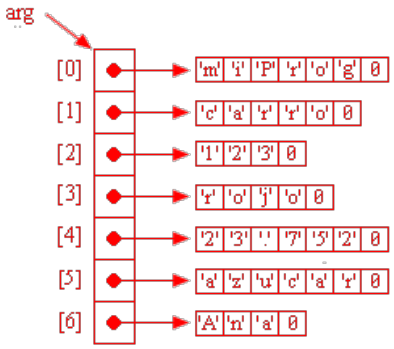
En el caso del C/C++, es la función main la encargada de recibir estos parámetros a través de la línea de comandos del sistema operativo.
Para poder ingresar parámetros a un programa en C/C++ se debe modificar el encabezado de la función main.
La sintaxis que se debe seguir es rigurosa, la cantida de parámetros y el tipo de éstas estás ya está dado y no se puede modificar. Esto quiere decir que no se puede elegir el tipo y cantidad de parámetros que tendrá.
La misma responde a:

int main(int argc, char *argv[]){ }	int main(int argc, char **argv){ }

La función main sólo puede terner dos parámetros, el primero es un parámetro por valor de tipo entero, en este caso argc, y el segundo es un doble puntero char (o de otro modo, un arreglo de punteros a char), en este caso argv, que también es un parámetor por valor.

Cuando se ejecute el programa desde la línea de comandos del sistema operativo, si se quiere desea proveer información al programa por medio de los parámetros, se debe colocar el nombre del programa seguido de una serie de palabras separadas por espacios en blanco. Por ejemplo:

Cada una de las palabras que se colocan en la línea de comandos, incuyendo el nombre del programa (en algunos compiladores el nombre del programa incluye su ruta de acceso), son tomados por el mismo programa y con ellas se arma un arreglo de punteros, como se observa en la figura:



Cada elemento de la estructura apunta a espacio de memoria en donde se han colocado las palabras que ingresaron a través de la línea de comandos del sistema operativo, este arreglo es referenciado por el argumento argv.
Luego, el sistema cuenta la cantidad de elementos que se crearon en el arreglo y coloca ese valor en el argumento argc.
Finalmente, el número de elementos del arreglo y la dirección de éste, son recibidos por la función main, para emplearlos según convenga.

El siguiente programa muestra, de manera simple, como usar los parámertos ingresados desde la línea de comandos del sistema operativo (el programa se denomina paramCLI.cpp).

```cpp
#include <iostream>

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

  std::cout << "Lista de " << argc << " parametros recibidos:" << std::endl;
  for (int i = 0; i < argc; i++){
    std::cout << "argv[" << i << "]: "<< argv[i] << std::endl;
  }

  return 0;
}
```

Al ejecute este programa desde la línea de comandos, con los argumentos:

```
$ ./paramCLI Un ejemplo simple del 09 / 24
```

Se obtiene el siguiente resultado:

```
Lista de 8 parametros recibidos:
argv[0]: ./paramCLI
argv[1]: Un
argv[2]: ejemplo
argv[3]: simple
argv[4]: del
argv[5]: 09
argv[6]: /
argv[7]: 24
```

Argumentos de la función main en Python

En el caso del Python, la recepción de los parámetros enviados a través de la línea de comandos del sistema operativo, se realiza mediante el módulo sys.
Particularmente, la función sys.argv representa la lista de los argumentos de la línea de comandos que se la han pasado al script.
Al ser una lista, es posible acceder a ellas a través de un número índice que empieza en 0.
Es decir, 0 es la primera posición de la lista y corresponde al nombre del script.

El mismo ejemplo anterior en este lenguaje es:

```python
import sys

def main():
    argc = len(sys.argv)
    print(f"Lista de {argc} parametros recibidos:")
    for i in range(argc):
        print(f"argv[{i}]: {sys.argv[i]}")

if __name__ == "__main__":
    main()
```

Con la orden de ejecución siguiente:

```
$ python3 paramCLI.py Un 2º ejemplo del 09 / 24
```

Se obtiene un resultado similar:

```
Lista de 8 parametros recibidos:
argv[0]: paramCLI.py
argv[1]: Un
argv[2]: 2º
argv[3]: ejemplo
argv[4]: del
argv[5]: 09
argv[6]: /
argv[7]: 24
cesar@puma3d:/media/cesar/Data17/D
```

Módulo argparse

La función sys.argv permite procesar argumentos en la línea de comando de manera sencilla, pero muy limitada.
Python dispone del módulo argparse, que permite procesar la línea de comando de forma mucho más flexible y con funcionalidades adicionales.

El módulo argparse posee muchas más funciones que sys.argv, lo que permite programar interfaces con más posibilidades. Entre las múltiples funcionalidades que ofrece argparse, se tiene:

- argumentos posicionales y por nombre
- argumentos opcionales
- manejo de argumentos de diversos tipos
- argumentos con valores múltiples

Este módulo es parte de la biblioteca estándar de Python, y por tanto puede importarse directamente con la directiva import argparse.

Más información en https://www.eumus.edu.uy/eme/ensenanza/electivas/python/2020/clase_08b.html

Ejemplos

Fragmento de programa C++

```cpp
// ================================================================================
int main(int argc, char* argv[]){

  if(argc == 2 and (string(argv[1]) == "-h" or string(argv[1]) == "--help")){
    cerr << "Sintaxis:" << endl;
    cerr << "c4ascii display font.txt \"mensaje\" \"caracter_de_frente\" [\"caracter_de_fondo\"]" << endl;
    cerr << "c4ascii read font.txt archivo_entrada.txt" << endl;
    exit(EXIT_SUCCESS);
  }else if(argc < 4){
    cerr << "Por favor, tipee de 4 a 6 argumentos o -h" << endl;
    exit(EXIT_FAILURE);
  }
```

Fragmento de programa Python

```python
# Flujo del módulo principal
if __name__ == '__main__':
  print ("Linea ejecutada: ", sys.argv)
  args =  sys.argv[1:]
  if 1 < len (args) <= 3:
    try:
      s = Sumador(args)
      suma = s.sumaNums()
      print ("Ingresado: ", args)
      print ("suma: ", suma)
      sys.exit()
```