Parámetros por línea de comandos

Se le llaman parámetros de línea de comandos a todo lo que se escriba después del nombre del programa en la invocación de un comando. Por ejemplo, hemos dicho que para compilar se usa una línea de la forma:

```
gcc hola.c -o hola
```

En este caso, estamos invocando al compilador `gcc` para que compile `hola.c` y genere el archivo ejecutable `hola`. Esto es posible, ya que el sistema operativo le pasa al gcc los parámetros que escribió el usuario al invocarlo, de forma que los parámetros que recibe gcc son:

```
"gcc", "hola.c", "-o", "hola"
```

En nuestro código, para recibir estos parámetros se utiliza, tradicionalmente, la *firma* de la función `main` que recibe parámetros:

```
int main(int argc, char *argv[]);
```

En este caso recibimos en `main` los parámetros de la invocación del programa. El primer parámetro será la cantidad de parámetros que se reciben y el segundo parámetro en un vector de punteros a caracteres.

Como se vio anteriormente, cada puntero a caracteres, es la dirección de memoria de un bloque de caracteres.

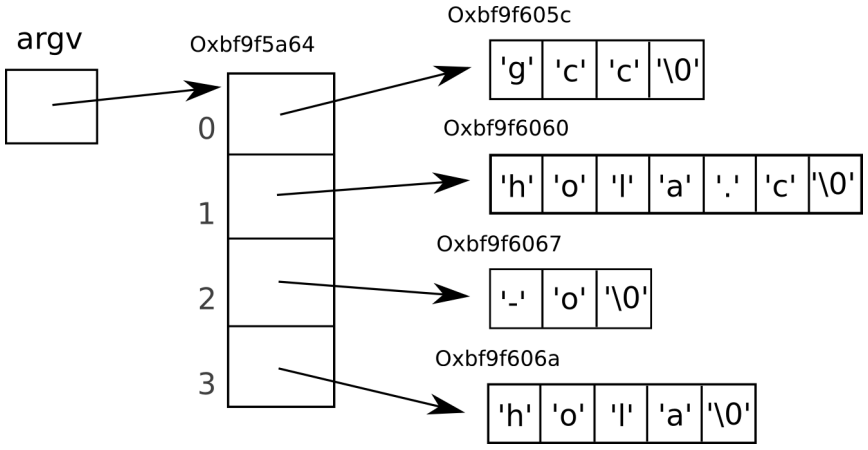

Figura 1: Estructura de los parámetros recibidos por la línea de comandos al compilar

Notar que la primera cadena apuntada por `argv`, es el nombre del comando que se llamó.

1. Ejemplo de recibir parámetros por línea de comandos

En UNIX existe un comando llamado **echo** cuya única función es imprimir todos los parámetros que se reciben por línea de comandos. Cada parámetro se imprime con un espacio entre parámetro y parámetro. Este sencillo programa puede escribirse en C de la siguiente forma.

```
1 #include <stdio.h>
2 int main(int argc, char *argv[])
3 {
4     if (argc > 1) {
5         printf("%s", argv[1]);
6     }
7     for (int i=2; i<argc; i++) {
8         printf(" %s", argv[i]);
9     }
10    printf("\n");
11    return 0;
12 }
```

En este código imprimimos todos los parámetros recibidos, omitiendo el nombre del comando (`argv[0]`). Una vez compilado se lo puede invocar de la siguiente forma:

```
./echo primer_parámetro  segundo_parámetro   etc
```

Y sin importar la cantidad de espacios entre un parámetro y otro el resultado será:

```
primer_parámetro segundo_parámetro etc
```