



INFORMATICA I

malloc y realloc

Ing. Juan Carlos Cuttitta

*Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Departamento de Ingeniería Electrónica*

30 de julio de 2020

Enunciado del problema

Asignar memoria dinamicamente al vector que contiene las direcciones de los nombres ingresados y utilizar la memoria justa para cada nombre.

La idea es que si reservé espacio para un vector que pueda almacenar 256 bytes pero ingreso un nombre que ocupa 5 bytes, utilicemos los recursos conocidos para que sólo se usen los espacios justos de memoria para esos 5 bytes y no los 256 bytes para cada nombre ingresado).

Termina el programa cuando un nombre comienza con el símbolo @

Ejemplo de malloc y realloc

```
1 int main (void)
2 {
3     int     i=0,j=0;
4     char    c,nombres[256];
5     char *  p;
6     char ** adr;
7
8     adr= (char **) malloc(sizeof(char *));
9     do{
10         fgets(nombres , 256 , stdin );
11         j = strlen(nombres);
12         p = (char *) malloc (j*sizeof(char));
13         strcpy ( p , nombres);
14         *(adr + i) = p;
15         c = (*(adr+i));
16         if ( c != '@' ){
17             adr = (char **) realloc( adr ,(i+2)* sizeof(char *));
18             i++;
19         } else{
20             free(p);
21             *(adr + i) = NULL;
22         }
23     } while( c != '@' );
24     for(i=0 ; *(adr+i) != NULL ;i++){
25         printf("nombre%d :% s ",i, *(adr+i) );
26         free(*(adr+i) );
27     }
28     free (adr);
29     return 0;
30 }
```

malloc y realloc en Arquitectura X86-32 bits

[illegible]

```
char nombres[256]
```

[illegible]

malloc y realloc en Arquitectura X86-32 bits

[illegible]

```
adr= (char **) malloc(sizeof(char *));
```

```
char nombres[256]
```



malloc y realloc en Arquitectura X86-32 bits

```
fgets( nombres , 256 , stdin );
```

```
char nombres[256]
```

[illegible][illegible]

malloc y realloc en Arquitectura X86-32 bits

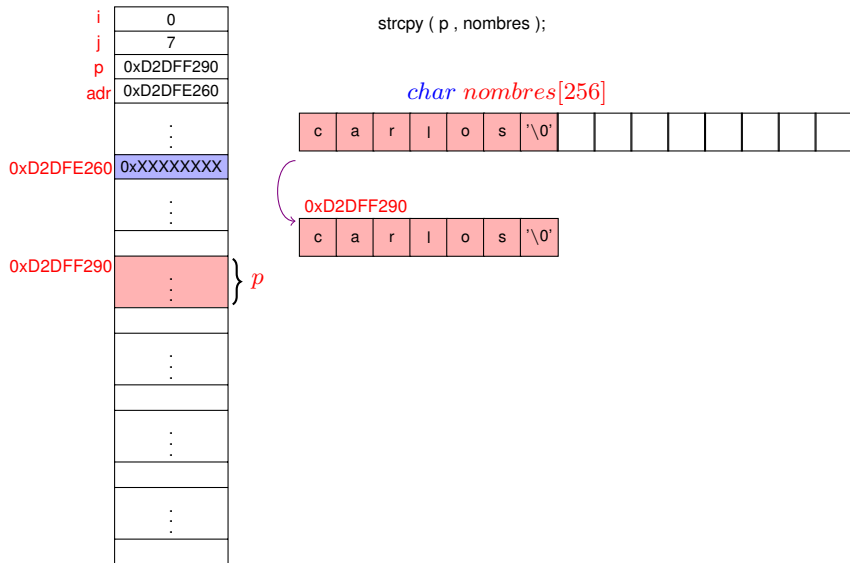
[illegible]

```
j = strlen( nombres );
```

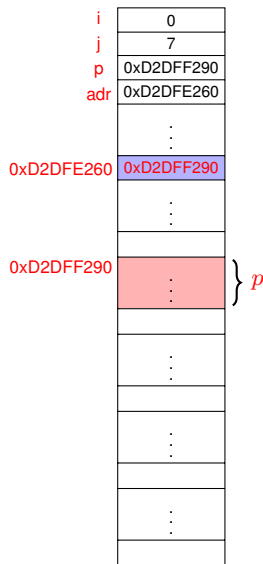
```
char nombres[256]
```

c	a	r	l	o	s	'\0'								
---	---	---	---	---	---	------	--	--	--	--	--	--	--	--

malloc y realloc en Arquitectura X86-32 bits



malloc y realloc en Arquitectura X86-32 bits



$*(\text{adr} + i) = p;$

char nombres[256]

c	a	r	l	o	s	'\0'								
---	---	---	---	---	---	------	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



* (*(adr+i)) contiene al primer caracter y como es != '@'

```
adr = (char **) realloc( adr , ( i+2 ) * sizeof(char *) )
```

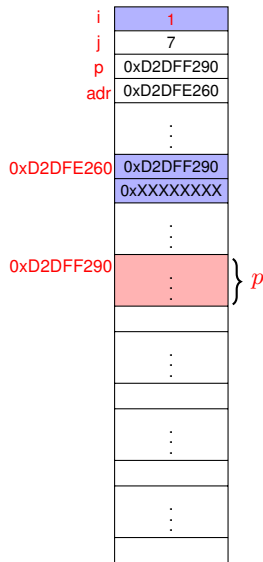
```
char nombres[256]
```

c	a	r	l	o	s	'\0'								
---	---	---	---	---	---	------	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



i++;

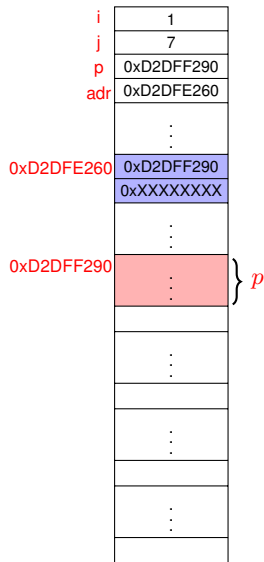
char nombres[256]

c	a	r	l	o	s	'\0'									
---	---	---	---	---	---	------	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

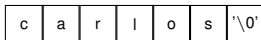


```
fgets( nombres , 256 , stdin );
```

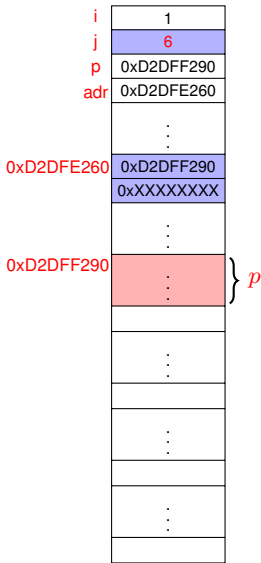
char nombres[256]



0xD2DFF290



malloc y realloc en Arquitectura X86-32 bits



```
j = strlen( nombres );
```

```
char nombres[256]
```

[illegible]

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



```
p = (char *) malloc ( j * sizeof(char) );
```

char nombres[256]

p	e	d	r	o	'\0'									
---	---	---	---	---	------	--	--	--	--	--	--	--	--	--

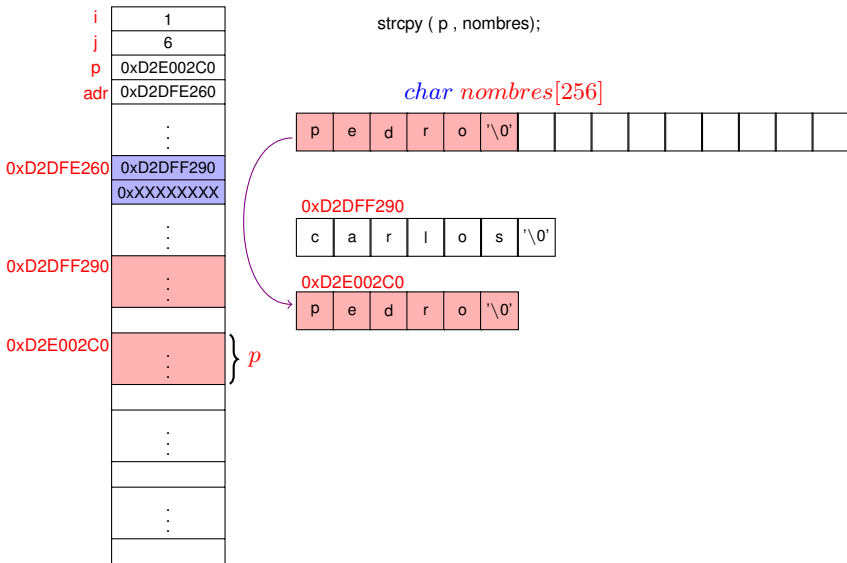
0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

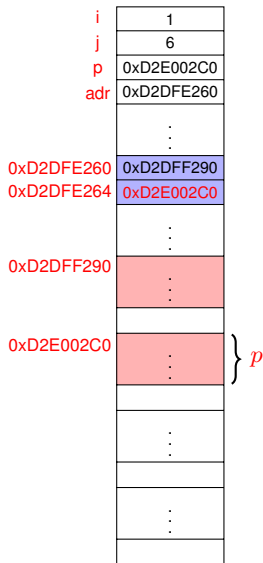
malloc y realloc en Arquitectura X86-32 bits

```
strcpy ( p , nombres);
```

char nombres[256]



malloc y realloc en Arquitectura X86-32 bits



$*(adr + i) = p;$

char nombres[256]

p	e	d	r	o	'\0'									
---	---	---	---	---	------	--	--	--	--	--	--	--	--	--

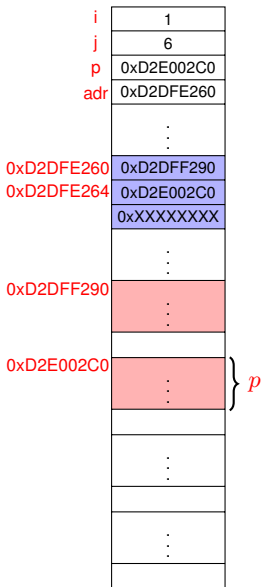
0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



$*(\text{adr} + i)$ contiene al primer caracter y como es != @

`adr = (char **) realloc(adr , (i+2) * sizeof(char *))`

char nombres[256]

p	e	d	r	o	'\0'									
---	---	---	---	---	------	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



i++;

char nombres[256]

p	e	d	r	o	'\0'									
---	---	---	---	---	------	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

```
fgets( nombres , 256 , stdin );
```

char nombres[256]

j	o	s	e	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

i	2
j	6
p	0xD2E002C0
adr	0xD2DFE260
	⋮
0xD2DFE260	0xD2DFF290
0xD2DFE264	0xD2E002C0
	0XXXXXXXXX
	⋮
0xD2DFF290	⋮
	⋮
0xD2E002C0	⋮
	⋮
	⋮
	⋮
	⋮
	⋮

} *p*

malloc y realloc en Arquitectura X86-32 bits



j = strlen(nombres);

char nombres[256]

j	o	s	e	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



```
p = (char *) malloc ( j * sizeof(char) );
```

char nombres[256]

j	o	s	e	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

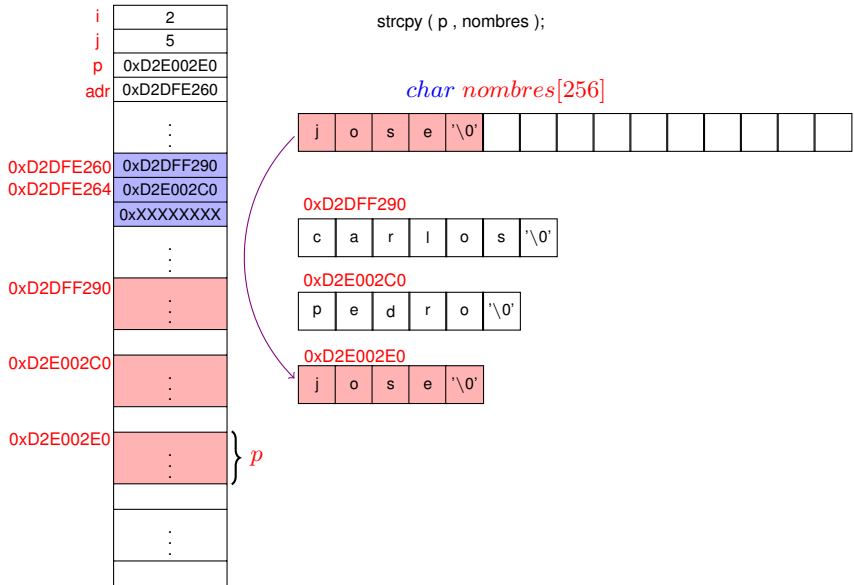
0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



malloc y realloc en Arquitectura X86-32 bits



$*(adr + i) = p;$

char nombres[256]

j	o	s	e	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



$\ast(\ast(\text{adr} + i))$ contiene al primer caracter y como es != @

$\text{adr} = (\text{char} \ast\ast) \text{realloc}(\text{adr}, (i + 2) \ast \text{sizeof}(\text{char} \ast))$

char nombres[256]

j	o	s	e	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

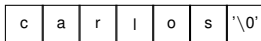


i++

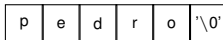
char nombres[256]



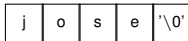
0xD2DFF290



0xD2E002C0



0xD2E002E0



malloc y realloc en Arquitectura X86-32 bits

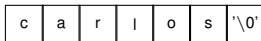


```
fgets( nombres , 256 , stdin );
```

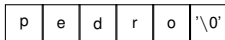
char nombres[256]



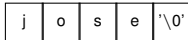
0xD2DFF290



0xD2E002C0



0xD2E002E0



malloc y realloc en Arquitectura X86-32 bits



j = strlen(nombres);

char nombres[256]

@	c	l	a	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

i	3
j	5
p	0xD2E00300
adr	0xD2DFE260
	⋮
0xD2DFE260	0xD2DFF290
0xD2DFE264	0xD2E002C0
0xD2DFE268	0xD2E002E0
0xD2DFE26C	0XXXXXXXX
	⋮
0xD2DFF290	⋮
	⋮
0xD2E002C0	⋮
	⋮
0xD2E002E0	⋮
	⋮
0xD2E00300	⋮

} p

```
p = (char *) malloc ( j * sizeof(char) );
```

char nombres[256]

@	c	l	a	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

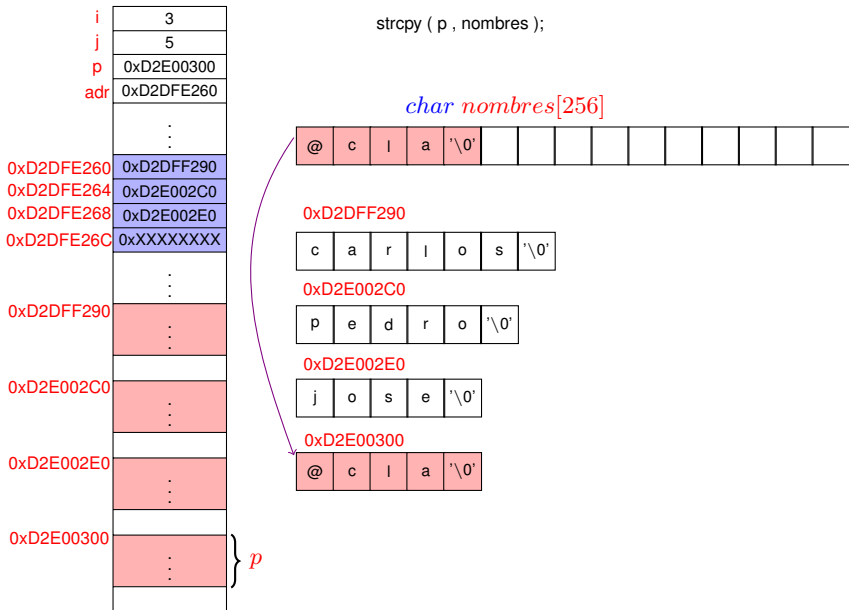
0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits



malloc y realloc en Arquitectura X86-32 bits



$*(adr + i) = p;$

char nombres[256]

@	c	l	a	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

0xD2E00300

@	c	l	a	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

i	3
j	5
p	0xD2E00300
adr	0xD2DFE260
	⋮
0xD2DFE260	0xD2DFF290
0xD2DFE264	0xD2E002C0
0xD2DFE268	0xD2E002E0
0xD2DFE26C	0xD2E00300
	⋮
0xD2DFF290	⋮
	⋮
0xD2E002C0	⋮
	⋮
0xD2E002E0	⋮
	⋮
	⋮
	⋮

$*(\text{adr}+i)$ contiene al primer caracter y es @, salimos del loop

`free(p);`

char nombres[256]

@	c	l	a	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

i	3
j	5
p	0xD2E00300
adr	0xD2DFE260
	⋮
0xD2DFE260	0xD2DFF290
0xD2DFE264	0xD2E002C0
0xD2DFE268	0xD2E002E0
0xD2DFE26C	NULL
	⋮
0xD2DFF290	⋮
	⋮
0xD2E002C0	⋮
	⋮
0xD2E002E0	⋮
	⋮
	⋮
	⋮

$*(\text{adr} + i) = \text{NULL};$

char nombres[256]

@	c	l	a	'\0'										
---	---	---	---	------	--	--	--	--	--	--	--	--	--	--

0xD2DFF290

c	a	r	l	o	s	'\0'
---	---	---	---	---	---	------

0xD2E002C0

p	e	d	r	o	'\0'
---	---	---	---	---	------

0xD2E002E0

j	o	s	e	'\0'
---	---	---	---	------

malloc y realloc en Arquitectura X86-32 bits

i	3
j	5
p	0xD2E00300
adr	0xD2DFE260
	⋮
0xD2DFE260	0xD2DFF290
0xD2DFE264	0xD2E002C0
0xD2DFE268	0xD2E002E0
0xD2DFE26C	NULL
	⋮
0xD2DFF290	⋮
	⋮
0xD2E002C0	⋮
	⋮
0xD2E002E0	⋮
	⋮
	⋮
	⋮

Por último imprime los nombres ingresados y libera los bloque de memoria reservado

```
char nombres[256]
```

[illegible]