Différence avec strcpy, strncpy, strlcpy, memcpy

```
char *strcpy(char *dst, const char *src);
char *strncpy(char * dst, const char * src, size_t len);
size_t strlcpy(char *dst, const char *src, size_t dstsize);
void *memcpy(void *dst, const void *src, size_t n);
```

- strcpy copie toute la chaîne source et s'arrête lorsqu'elle rencontre la première terminaison '\0'. Elle ajoute à la fin le '\0'. Elle ne copie pas au-délà de cette terminaison.
- strncpy copie au plus len caractères de la chaîne source et s'arrête au premier '\0'. Elle ajoute pas un '\0'.Elle rempli la différence avec des '\0'
- strlcpy copie au plus dstsize - 1 éléments où dstsize est la taille de dst.
- memcpy prend comme paramètre une taille et peut copier au-delà de cette terminaison!.

- memcpy accepte en paramètre un pointeur void, alors que strcpy et strncpy acceptent en paramètre un char.
    - -> memcpy copie un nombre spécifié d'octets d'une source mémoire, vers une autre.
    - -> strcpy et strncpy copient le contenu d'une chaîne de caractère vers une autre.
    - -> memcpy agit sur la mémoire plutôt que sur le contenu.
    - -> strcpy et strncpy agissent sur le contenu plutôt que sur la mémoire.

Ce n'est que strcpy et strlcpy qui rajoutent toujours un '\0' à la fin de la copie! memcpy et strncpy ne le font pas!

EXEMPLE:

```
char *src;
src = "hello world!";
char dst[13] = "xxxxxxxxxxxx\0";
```

source est un constant char -> a un '\0' a la fin donc il y a 13 éléments dont les index vont de 0 à 12.

On déclare une chaîne dst avec autant d'éléments et contenant un '\0' pour marquer la fin de la chaîne.

On veut copier la chaîne entière. A noter que strcpy va toujours copier la chaîne entière.

élément n	src	strcpy (src, dst)	strncpy (src, dst, 13)	strlcpy (src, dst, 13)	memcpy (src, dst, 13)
[0]	h	h	h	h	h
[1]	e	e	e	e	e
[2]	l	l	l	l	l
[3]	l	l	l	l	l
[4]	o	o	o	o	o
[5]					
[6]	w	w	w	w	w
[7]	o	o	o	o	o
[8]	r	r	r	r	r
[9]	l	l	l	l	l
[10]	d	d	d	d	d
[11]	!	!	!	!	!
[12]	\0	\0	\0	\0	\0

- dans cette exemple toutes les fonctions copient de la même manière.

EXEMPLE:

```
char *src;
src = "hello world!";
char dst[13] = "xxxxxxxxxxxx\0";
```

source est un constant char -> a un '\0' a la fin donc il y a 13 éléments dont les index vont de 0 à 12.

On déclare une chaîne dst avec autant d'éléments et contenant un '\0' pour marquer la fin de la chaîne.

On veut copier 5 éléments seulement. A noter que strcpy va toujours copier la chaîne entière.

élément n	src	strcpy	strncpy	strlcpy	memcpy
		(src, dst)	(src, dst, 5)	(src, dst, 5)	(src, dst, 5)
[0]	h	h	h	h	h
[1]	e	e	e	e	e
[2]	l	l	l	l	l
[3]	l	l	l	l	l
[4]	o	o	o	\0	o
[5]			x	x	x
[6]	w	w	x	x	x
[7]	o	o	x	x	x
[8]	r	r	x	x	x
[9]	l	l	x	x	x
[10]	d	d	x	x	x
[11]	!	!	x	x	x
[12]	\0	\0	\0	\0	\0

- strcpy copie la chaîne entière et ajoute un '\0' à la fin.

- strncpy et memcpy copient 5 éléments et ne rajoute pas de '\0' à la fin. Les élélemts suivants restent inchangés.

- strlcpy copie dstsize - 1 éléments de la chaîne src vers la chaîne dst. Elle rajoute un '\0' à l'élément numéro dstsize - 1

EXEMPLE:

```
char *src;
src = "hello\0world!";
char dst[13] = "xxxxxxxxxxxx\0";
```

source est un constant char -> a un '\0' a la fin, même si au milieu il y a déjà un '\0'. Donc il y a 13 éléments dont les index vont de 0 à 12.

On déclare une chaîne dst avec autant d'éléments et contenant un '\0' pour marquer la fin de la chaîne.

On veut copier la chaîne entière. A noter que strcpy va toujours copier la chaîne entière.

élément n	src	strcpy	strncpy	strlcpy	memcpy
		(src, dst)	(src, dst, 13)	(src, dst, 13)	(src, dst, 13)
[0]	h	h	h	h	h
[1]	e	e	e	e	e
[2]	l	l	l	l	l
[3]	l	l	l	l	l
[4]	o	o	o	o	o
[5]	\0	\0	\0	\0	\0
[6]	w	x	\0	x	w
[7]	o	x	\0	x	o
[8]	r	x	\0	x	r
[9]	l	x	\0	x	l
[10]	d	x	\0	x	d
[11]	!	x	\0	x	!
[12]	\0	\0	\0	\0	\0

- strcpy arrête la copie au premier '\0' et rajoute un '\0' à fin. Le reste de la chaîne est inchangé.
- strncpy arrête la copie au premier '\0' et remplit la différence len - i avec des '\0' (ici de l'élement 5 jusqu'au 12)
- strlcpy arrête la copie au premier '\0' et ajoute un '\0'. Le reste de la chaîne est inchangé.
- memcpy ne s'arrête pas au 1er '\0' parce qu'il dépend d'une taille et non de la fin de la chaîne!

EXEMPLE:

```
char *src;
src = "hello world!";
char dst[12] = "xxxxxxxxxxx\0";
```

source est un constant char -> a un '\0' a la fin. Donc il y a 13 éléments dont les index vont de 0 à 12.

On déclare une chaîne dst avec 12 éléments et contenant un '\0' pour marquer la fin de la chaîne.

On veut copier 12 éléments de chaîne src vers la chaîne dst. A noter que dst est plus petite src.

élément n	src	strcpy	strncpy	strlcpy	memcpy
		(src, dst)	(src, dst, 12)	(src, dst, 12)	(src, dst, 12)
[0]	h	overflow	h	h	h
[1]	e		e	e	e
[2]	l		l	l	l
[3]	l		l	l	l
[4]	o		o	o	o
[5]	\0				
[6]	w		w	w	w
[7]	o		o	o	o
[8]	r		r	r	r
[9]	l		l	l	l
[10]	d		d	d	d
[11]	!		!	\0	!
[12]	\0				

- strcpy fait un overflow parce qu'il essaye de copier la chaîne entière de src, alors que dst a moins d'espace

- strncpy et memcpy copient la chaîne mais ne rajoutent pas un '\0' à la fin.

- strlcpy ajoute un '\0' à l'élément numéro dstsize - 1.