List of Libft Functions

Part I

Memset

void *memset(void *str, int c, size_t n)

- *copies the character c (unsigned char) to the first n characters of the string pointer to by the argument str*

bzero

void bzero(void *s, int byte)

- *Places byte null bytes in the string s. It's used to set all the socket structures with null values.*

memcpy

void *memcpy(void *str1, const void *str2, size_t n)

- *Copies n characters from memory area str2 to memory area str1.*

memccpy

void *memccpy(void *dest, const void *src, int c, size_t n)

- *Copies from one memory source to another, but stops after the first occurrence of c, or after n bytes are copies (whichever comes first).*

memmove

void *memmove(void *strq, const void *str2, size_t n)

- *Copies n characters from str2 to str1, but for overlapping memory blocks, memmove() is a safer approach than memcpy().*

memchr

void *memchr(const void *str, int c, size_t n)

- *Searches for the first occurrence of the character c in the first n bytes of the string pointed to by str.*

memcmp

int memcmp(const void *str1, const void *str2, size_t n)

- *Compares the first n bytes of memory area str1 and memory area str2.*

strcat

char *strcat(char *dest, const char *src)

- *Appends the string pointed to by src to the end of the string pointed to by dest.*

strncat

char *strncat(char *dest, const char *src, size_t n)

- *Appends string pointed to by dest up to n characters long.*

List of Libft Functions

strlcat

size_t strlcat(char *dest, const char *src, size_t size)

- *Concats chars from src to dest and null-terminates the rest of the string. As much of src is copied into dest as there is space for.*

strchr

char *strchr(const char *str, int c)

- *returns a pointer to the first occurrence of the character c in str, or NULL if char isn't found.*

strrchr

char *strrchr(const char *str, int c)

- *Return pointer to the last occurrence of character in str. If not found, returns NULL.*

isdigit

int isdigit(int c)

- *Returns nonzero value if c is a digit, else returns 0.*

isascii

int isascii(int c)

- *Returns 1 if ASCII, 0 if not.*

isprint

int isprint(int c)

- *Returns 1 if c is a printable character; else, 0.*

atoi

int atoi(const char *str)

- *converts the string argument str to an integer (type int)*

isalnum

int isalnum(int c);

- *checks if the passed character is alphanumeric*

toupper

int toupper(int c)

- *converts lowercase letter to uppercase.*

to lower

int tolower(int c)

- *converts uppercase letter to lowercase.*

List of Libft Functions

strlen

size_t strlen(const char *str)

- *Computes the length of the string* str *up to, but not including the terminating null character.*

strdup

char *strdup*const char *s1)

- *Returns a pointer to a new string, which is the duplicate of the string pointed to by s1. The returned pointer can be passed to free(). A null pointer is returned if the new string cannot be created.*

strcpy

char *strcpy(char *dest, const char *src)

- *Copies the string pointed to by src to dest.*

strncpy

char *strncpy(char (dest, const char *src, size_t num)

- *Copies the first num characters of src to dest. If the end of the src string is found before num characters have been copied, dest is padded with zeros until a total of num characters have been written to it.*

strstr

char *strstr(const char *haystack, const char *needle)

- *Finds the first occurrence of the substring needle in the string haystack. The terminating '\0' characters are not compared.*

strnstr

char *strnstr(const char *big, const char *little, size_t len)

- *Locates the first occurrence of the null-terminated string little in the string big, where not more than len characters are searched. Characters that appear after a '\0' character are not searched. Since the strnstr function is a FreeBSD specific API, it should only be used when portability is not a concern.*

strcmp

int strcmp(const char *str1, const char *str2)

- *Compares the string pointed to by str1 to the string pointed to by str2.*

strncmp

int strncmp(const char *str1, const char *str2, size_t n)

- *Compares at most the first n bytes of str1 and str2.*

List of Libft Functions

isalpha

int isalpha(int c)

- *Checks if the passed character is alphabetic.*

Part II

ft_memalloc

void *ft_memalloc(size_t size)

- *Allocated (with malloc) and returns a "fresh" memory area. The memory allocated is initialized to 0. If the allocation fails, the function returns NULL.*

ft_memdel

void ft_memdel(void **ap)

- *Takes as a parameter the address of a memory area that needs to be freed with free, then puts the pointer to NULL.*

ft_strnew

char *ft_strnew(size_t size)

- *Allocates (with malloc) and returns a "fresh" string ending with '\0'. Each character of the string is initialized at '\0'. If the allocation fails the function returns NULL.*

ft_strdel

void ft_strdel(char **as)

- *Takes as a parameter the address of a string that needs to be freed with free, then sets its pointer to NULL.*

ft_strclr

void ft_strclr(char *s)

- *Sets every character of the string to the value '\0'.*

ft_striteri

void ft_striteri(char *s, void (*f)(unsigned int, char *))

- *Applies the function f to each character of the string passed as argument, and passing its index as first argument. Each character is passed by address to f to be modified if necessary.*

List of Libft Functions

ft_striter

void ft_striter(char *s, void (*f)(char *))

- *Applies the function f to each character of the string passed as argument. Each character is passed by address to f to be modified if necessary.*

ft_strmap

char *ft_strmap(char const *s, char (*f)(char))

- *applies the function f to each character of the string given as argument to create a "fresh" new string (with malloc) resulting from the successive applications of f.*

ft_strmapi

char *ft_strmapi(char const *s, char (*f)(unsigned int, char))

- *Applies the unction f to each character of the string passed as argument by giving its index as first argument to create a "fresh" new string (with malloc) resulting from the successive applications of f.*

ft_strequ

int ft_strequ(char const *s1, char const *s2)

- *Lexicographical comparison between s1 and s2. If the two strings are identical the function returns 1, otherwise it returns 0.*

ft_strnequ

int ft_strnequ(char const *s1, char const *s2, size_t n)

- *Lexicographical comparison between s1 and s2.*

ft_strsub

char *ft_strsub(char const *s, unsigned int start, size_t len)

- *Allocates (with malloc) and returns a "fresh" substring from the string given as argument. The substring begins at index start and is of size len. If start and len aren't referring to a valid substring, the behavior is undefined. If the allocation fails, the function returns NULL.*

ft_strjoin

char *ft_strjoin(char const *s1, char const *s2)

- *Allocates (with malloc) and returns a "fresh" string ending with '\0', the result of the concatenation of s1 and s2. If the allocation fails the function returns NULL.*

List of Libft Functions

ft_strtrim

char *ft_strtrim(char const *s)

- *Allocates (with malloc) and returns a copy of the string given as argument without whitespace at the beginning or at the end of the string. The following will be considered white spaces: ' ', '\n', and '\t'. If s has no whitespace at the beginning or at the end, the function returns a copy of s. If the allocation fails the function returns NULL.*

ft_strsplit

char **ft_strsplit(char const *s, char c)

- *Allocates (with malloc) and returns an array of "fresh" strings (all ending with '\0', including the array self) obtained by splitting s using the character c as a delimiter. If the allocation fails the function returns NULL. Example: ft_strsplit("*hello*fellow***students*", '*') returns the array ["hello", "fellow", "students"].*

ft_itoa

char *ft_itoa(int n)

- *Allocate (with malloc) and returns a "fresh" string ending with '\0' representing the integer n given as argument. Negative numbers must be supported. If the allocation fails, the function returns NULL.*

ft_putchar

void ft_putchar(char c)

- *Outputs the character c to the standard output.*

ft_putstr

void ft_putstr(char const *s)

- *Outputs the string s to the standard output.*

ft_putendl

void ft_putendl(char const *s)

- *Outputs the string s to the standard output followed by a '\n'.*

ft_putnbr

void ft_putnbr(int n)

- *Outputs the integer n to the standard output.*

ft_putchar_fd

void ft_putchar_fd(char c, int fd)

- *Outputs the char c to the file descriptor fd.*

List of Libft Functions

ft_putstr_fd

void ft_putstr_fd(char const *s, int fd)

- *Outputs the string s to the file descriptor fd*

ft_putendl_fd

void ft_putendl_fd(char const *s, int fd)

- *Outputs the string s to the file descriptor fd followed by a '\n'.*

ft_putnbr_fd

void ft_putnbr_fd(int n, int fd)

- *Outputs the integer n to the file descriptor fd.*

Bonus Functions

ft_lstnew

t_list *ft_lstnew(void const *content, size_t content_size)

- *Allocates (with malloc) and returns a "fresh" link. The variables content and content_sizeof the new link are initialized by copy of the parameters of the function. If the parameter content is null, the variable content is initialized to NULL and the variable content_size is initialized to 0 even if the parameter content_size isn't. The variable next is initialized to NULL. If the allocation fails, the function returns NULL.*

ft_lstdelone

void ft_lstdelone(t_list **alst, void (*del)(void *, size_t))

- *Takes as a parameter a link's pointer address and frees the memory of the link's content using the function del given as a parameter, then frees the link's memory using free. The memory of next must not be freed under any circumstance. Finally, the pointer to the link that was just freed must be set to nULL (similar to the function ft_memdel in the mandatory section).*

ft_lstdel

void ft_lstdel(t_list **alst, void (*del)(void *, size_t))

- *Takes as a parameter the address of a pointer to a link and frees the memory of this link and every successor of that link using the functions del and free. Finally, the pointer to the link that was just freed must be set to NULL (similar to ft_memdel).*

ft_lstadd

void ft_lstadd(t_list **alst, t_list *new)

- *Adds the element new at the beginning of the list.*

List of Libft Functions

ft_lstiter

void ft_lstiter(t_list *lst, void (*f)(t_list *elem))

- *Iterates the list let and applies the function f to each link.*

ft_lstmap

t_list *ft_lstmap(t_list *lst, t_list *(*f)(t_list *elem))

- *Iterates a list let and applies the function f to each link to create a "fresh" list (using malloc) resulting from the successive applications of f. If the allocation fails, the function returns NULL.*