

UNIX SOCKET - HELPER FUNCTIONS

http://www.tutorialspoint.com/unix_sockets/socket_helper_functions.htm

Copyright © tutorialspoint.com

This chapter describes all the helper functions, which are used while doing socket programming. Other helper functions are described in the chapters: **Ports and Services**, and Network **Byte Orders**.

The *write* Function

The *write* function attempts to write *nbyte* bytes from the buffer pointed by *buf* to the file associated with the open file descriptor, *fildes*.

You can also use *send* function to send data to another process.

```
#include <unistd.h>

int write(int fildes, const void *buf, int nbyte);
```

Upon successful completion, *write* returns the number of bytes actually written to the file associated with *fildes*. This number is never greater than *nbyte*. Otherwise, -1 is returned.

Parameters

- **fildes**: It is a socket descriptor returned by the socket function.
- **buf**: It is a pointer to the data you want to send.
- **nbyte**: It is the number of bytes to be written. If *nbyte* is 0, *write* will return 0 and have no other results if the file is a regular file; otherwise, the results are unspecified.

The *read* Function:

The *read* function attempts to read *nbyte* bytes from the file associated with the open file descriptor, *fildes*, into the buffer pointed to by *buf*.

You can also use *recv* function to read data to another process.

```
#include <unistd.h>

int read(int fildes, const void *buf, int nbyte);
```

Upon successful completion, *read* returns the number of bytes actually written to the file associated with *fildes*. This number is never greater than *nbyte*. Otherwise, -1 is returned.

Parameters

- **fildes**: It is a socket descriptor returned by the socket function.
- **buf**: It is the buffer to read the information into..
- **nbyte**: It is the number of bytes to read.

The *fork* Function:

The *fork* function creates a new process. The new process called the child process will be an exact copy of the calling process *parentprocess*. The child process inherits many attributes from the parent process.

```
#include <sys/types.h>
#include <unistd.h>

int fork(void);
```

Upon successful completion, `fork` returns 0 to the child process and the process ID of the child process to the parent process. Otherwise -1 is returned to the parent process, no child process is created and `errno` is set to indicate the error.

Parameters

- **void**: It means no parameter is required.

The *bzero* Function

The *bzero* function places *nbyte* null bytes in the string *s*. This function is used to set all the socket structures with null values.

```
void bzero(void *s, int nbyte);
```

This function does not return anything.

Parameters

- **s**: It specifies the string which has to be filled with null bytes. This will be a point to socket structure variable.
- **nbyte**: It specifies the number of bytes to be filled with null values. This will be the size of the socket structure.

The *bcmp* Function

The *bcmp* function compares byte string *s1* against byte string *s2*. Both strings are assumed to be *nbyte* bytes long.

```
int bcmp(const void *s1, const void *s2, int nbyte);
```

This function returns 0 if both strings are identical, 1 otherwise. The *bcmp* function always returns 0 when *nbyte* is 0.

Parameters

- **s1**: It specifies the first string to be compared.
- **s2**: It specifies the second string to be compared.
- **nbyte**: It specifies the number of bytes to be compared.

The *bcopy* Function

The *bcopy* function copies *nbyte* bytes from string *s1* to the string *s2*. Overlapping strings are handled correctly.

```
void bcopy(const void *s1, void *s2, int nbyte);
```

This function does not return anything.

Parameters

- **s1**: It specifies the source string.
- **s2v**: It specifies the destination string.
- **nbyte**: It specifies the number of bytes to be copied.

The *memset* Function

The *memset* function is also used to set structure variables in the same way as **bzero**. Take a look at its syntax, given below.

```
void *memset(void *s, int c, int nbyte);
```

This function returns a pointer to void; in fact, a pointer to the set memory and you need to cast it accordingly.

Parameters:

- **s**: It specifies the source to be set.
- **c**: It specifies the character to set on nbyte places..
- **nbyte**: It specifies the number of bytes to be set.

Loading [MathJax]/jax/output/HTML-CSS/jax.js