**Advanced Socket Programing Clause**

1) getsockopt()

Use : Retrieve information about socket options

Syntax :

#include <sys/types.h>

#include <sys/socket.h>

int getsockopt(int socket\_descriptor,

int level,

int option\_name,

char \*option\_value,

int \*option\_length)

Example :

int rc;

int s;

int option\_value;

int option\_len;

struct linger l;

int getsockopt(int s, int level, int option\_name,

char \*option\_value,

int \*option\_len);

⋮

/\* Is out-of-band data in the normal input queue? \*/

option\_len = sizeof(int);

rc = getsockopt(

s, SOL\_SOCKET, SO\_OOBINLINE, (

char \*) &option\_value, &option\_len);

if (rc == 0)

{

if (option\_len == sizeof(int))

{

if (option\_value)

/\* yes it is in the normal queue \*/

else

/\* no it is not

\*/

}

}

⋮

/\* Do I linger on close? \*/

option\_len = sizeof(l);

rc = getsockopt(

s, SOL\_SOCKET, SO\_LINGER, (char \*) &l, &option\_len);

if (rc == 0)

{

if (option\_len == sizeof(l))

{

if (l.l\_onoff)

/\* yes I linger \*/

else

/\* no I do not \*/

}

}

**2)** setsocketopt**()**

# Use : Set options associated with a socket

# Syntax :

# #include <[sys/socket.h](https://pubs.opengroup.org/onlinepubs/000095399/basedefs/sys/socket.h.html)> int setsockopt(int *socket*, int *level*, int *option\_name*,        const void \**option\_value*, socklen\_t *option\_len*);

# Example :

int rc;

int s;

int option\_value;

struct linger l;

int setsockopt(int s, int level, int option\_name,

char \*option\_value,

int option\_len);

⋮

/\* I want out of band data in the normal input queue \*/

option\_value = 1;

rc = setsockopt(s, SOL\_SOCKET, SO\_OOBINLINE,

(char \*) &option\_value, sizeof(int));

⋮

/\* I want to linger on close \*/

l.l\_onoff = 1;

l.l\_linger = 100;

rc = setsockopt(s, SOL\_SOCKET, SO\_LINGER,

# (char \*) &l, sizeof(l));

# 3) getpeername() —

# Use - Get the name of the peer connected to a socket

# Syntax –

#include <sys/types.h>

#include <sys/socket.h>

int getpeername(int socket\_descriptor,

struct sockaddr \*destination\_address,

int \*address\_length)

# Example –

int s;

struct sockaddr\_in server, addr;

socklen\_t len;

// make a socket

s = socket(PF\_INET, SOCK\_STREAM, 0);

// connect to a server

server.sin\_family = AF\_INET;

inet\_aton("63.161.169.137", &server.sin\_addr);

server.sin\_port = htons(80);

connect(s, (struct sockaddr\*)&server, sizeof(server));

// get the peer name

// we know we just connected to 63.161.169.137:80, so this should print:

// Peer IP address: 63.161.169.137

// Peer port : 80

len = sizeof(addr);

getpeername(s, (struct sockaddr\*)&addr, &len);

printf("Peer IP address: %s\n", inet\_ntoa(addr.sin\_addr));

# printf("Peer port : %d\n", ntohs(addr.sin\_port));

# 4) getsockname() -

# Use - Return the name of the local socket

**Syntax –**

#include <[sys/socket.h](https://pubs.opengroup.org/onlinepubs/007904875/basedefs/sys/socket.h.html)>  
  
int getsockname(int *socket*, struct sockaddr \*restrict *address*,  
       socklen\_t \*restrict *address\_len*);

**Example –**

#include <sys/socket.h>

⋮

int addrlen;

int rc;

int server\_sock;

struct sockaddr\_un server\_addr;

⋮

addrlen = sizeof(server\_addr);

rc = getsockname(server\_sock,(struct sockaddr \*)&server\_addr, &addrlen);

if (rc == -1){

printf(“GETSOCKNAME ERROR = %d\n”, sock\_errno());

}

# 5) read() —

# USE - Read from a file or socket

# Format –

#define\_POSIX\_SOURCE

#include <unistd.h>

# ssize\_t read(int *fs*, void \**buf*, size\_t *N*);

**Example –**

⁄\* CELEBR03

This example opens a file and reads input.

\*⁄

#define \_POSIX\_SOURCE

#include <fcntl.h>

#include <unistd.h>

#undef \_POSIX\_SOURCE

#include <stdio.h>

main() {

int ret, fd;

char buf[1024];

system("ls -l ⁄ >| ls.output");

if ((fd = open("ls.output", O\_RDONLY)) < 0)

perror("open() error");

else {

while ((ret = read(fd, buf, sizeof(buf)-1)) > 0) {

buf[ret] = 0x00;

printf("block read: \n<%s>\n", buf);

}

close(fd);

}

unlink("ls.output");

}

# 6) write() —

# Use - Write data on a file or socket

# Syntax –

#define \_POSIX\_SOURCE

#include <unistd.h>

# ssize\_t write(int *fs*, const void \**buf*, size\_t *N*);

**Examle –**

⁄\* CELEBW35

This example writes a certain amount of bytes to a file, using write().

\*⁄

#define \_POSIX\_SOURCE

#include <fcntl.h>

#include <sys⁄stat.h>

#include <sys⁄types.h>

#include <unistd.h>

#undef \_POSIX\_SOURCE

#include <stdlib.h>

#include <stdio.h>

#define mega\_string\_len 1000000

main() {

char \*mega\_string;

int fd, ret;

char fn[]="write.file";

if ((mega\_string = (char\*) malloc(mega\_string\_len)) == NULL)

perror("malloc() error");

else if ((fd = creat(fn, S\_IWUSR)) < 0)

perror("creat() error");

else {

memset(mega\_string, '0', mega\_string\_len);

if ((ret = write(fd, mega\_10111000000string, mega\_string\_len)) == -1)

perror("write() error"00);

else printf("write() wrote %d bytes\n", ret);

close(fd);

unlink(fn);

}

}