**Programs to demonstrate the usage of Advanced socket system calls like  
getsockopt( ), setsockopt(), getpeername ( ),getsockname( ),readv( ) and writev().**

**Program:**

**/\*GET SET SOCK\*/**

#include<stdio.h>

#include<stdlib.h>

#include<errno.h>

#include<string.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<netinet/tcp.h>

int main()

{

int sockfd,maxseg,sendbuff,optlen;

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))<0)

{

perror("socket");

exit(0);

}

optlen=sizeof(maxseg);

if(getsockopt(sockfd,IPPROTO\_TCP,TCP\_MAXSEG,(char \*)&maxseg,&optlen)<0)

{

perror("get sockopt1");

exit(0);

}

printf("\n TCP maxseg=%d",maxseg);

sendbuff=12324;

if(setsockopt(sockfd,SOL\_SOCKET,SO\_SNDBUF,(char \*)&sendbuff,sizeof(sendbuff))<0)

{

perror("set socketopt");

exit(0);

}

optlen=sizeof(sendbuff);

if(getsockopt(sockfd,SOL\_SOCKET,SO\_SNDBUF,(char \*)&sendbuff,&optlen)<0)

{

perror("getsockopt2");

exit(0);

}

printf("\n send buffer size=%d\n",sendbuff);

}

**OUTPUT:**

[mca02-08@localhost ~]$ gcc getsetsock.c

[mca02-08@localhost ~]$ ./a.out 5647

TCP maxseg=536

send buffer size=24648

**getsockname:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define SERVER\_ADDR "172.217.160.99"

#define SERVER\_PORT 80

int main()

{

char myIP[16];

unsigned int myPort;

struct sockaddr\_in server\_addr, my\_addr;

int sockfd;

// Connect to server

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

perror("Can't open stream socket.");

exit(-1);

}

// Set server\_addr

bzero(&server\_addr, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_ADDR);

server\_addr.sin\_port = htons(SERVER\_PORT);

// Connect to server

if (connect(sockfd, (struct sockaddr \*) &server\_addr, sizeof(server\_addr)) < 0) {

perror("Connect server error");

close(sockfd);

exit(-1);

}

// Get my ip address and port

bzero(&my\_addr, sizeof(my\_addr));

socklen\_t len = sizeof(my\_addr);

getsockname(sockfd, (struct sockaddr \*) &my\_addr, &len);

inet\_ntop(AF\_INET, &my\_addr.sin\_addr, myIP, sizeof(myIP));

myPort = ntohs(my\_addr.sin\_port);

printf("Local ip address: %s\n", myIP);

printf("Local port : %u\n", myPort);

return 0;

**}**

**Getpeername:**

**using getpeername sample c function code**

**/\* getpeer.c**

**\* Demonstrate getpeername(2):**

**\*/**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <netinet/in.h>

#include <arpa/inet.h>

/\*

\* This function accepts as input a socket

\* for which a peer socket address must be

\* determined for it. Then the address

\* is converted into a string and returned.

\*

\* If an error occurs, NULL is returned.

\*/

char \* peer\_addr(int s, char \*buf, size\_t bufsiz) {

int z; /\* Status return code \*/

struct sockaddr\_in adr\_inet;/\* AF\_INET \*/

int len\_inet; /\* length \*/

/\*

\* Obtain the address of the socket:

\*/

len\_inet = sizeof adr\_inet;

z = getpeername(s, (struct sockaddr \*)&adr\_inet, &len\_inet);

if ( z == -1) {

return NULL; /\* Failed \*/

}

/\*

\* Convert address into a string

\* form that can be displayed:

\*/

z = snprintf(buf,bufsiz, "%s:%u",

inet\_ntoa(adr\_inet.sin\_addr),

(unsigned)ntohs(adr\_inet.sin\_port));

if ( z == -1 ) {

return NULL; /\* Buffer too small \*/

}

}

**Writev():**

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <sys/uio.h>

int main ( )

{

struct iovec iov[3];

ssize\_t nr;

int fd, i;

char \*buf[] = {

"The term buccaneer comes from the word boucan.\n",

"A boucan is a wooden frame used for cooking meat.\n",

"Buccaneer is the West Indies name for a pirate.\n" };

fd = open ("buccaneer.txt", O\_WRONLY | O\_CREAT | O\_TRUNC);

if (fd == −1) {

perror ("open");

return 1;

}

/\* fill out three iovec structures \*/

for (i = 0; i < 3; i++) {

iov[i].iov\_base = buf[i];

iov[i].iov\_len = strlen(buf[i]) + 1;

}

/\* with a single call, write them all out \*/

nr = writev (fd, iov, 3);

if (nr == −1) {

perror ("writev");

return 1;

}

printf ("wrote %d bytes\n", nr);

if (close (fd)) {

perror ("close");

return 1;

}

return 0;

}

**Running the program produces the desired result:**

**$ ./writev**

**wrote 148 bytes**

**As does reading the file:**

**$ cat buccaneer.txt**

**The term buccaneer comes from the word boucan.**

**A boucan is a wooden frame used for cooking meat.**

**Buccaneer is the West Indies name for a pirate.**

**readv( ) example:**

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/uio.h>

int main ( )

{

char foo[48], bar[51], baz[49];

struct iovec iov[3];

ssize\_t nr;

int fd, i;

fd = open ("buccaneer.txt", O\_RDONLY);

if (fd == −1) {

perror ("open");

return 1;

}

/\* set up our iovec structures \*/

iov[0].iov\_base = foo;

iov[0].iov\_len = sizeof (foo);

iov[1].iov\_base = bar;

iov[1].iov\_len = sizeof (bar);

iov[2].iov\_base = baz;

iov[2].iov\_len = sizeof (baz);

/\* read into the structures with a single call \*/

nr = readv (fd, iov, 3);

if (nr == −1) {

perror ("readv");

return 1;

}

for (i = 0; i < 3; i++)

printf ("%d: %s", i, (char \*) iov[i].iov\_base);

if (close (fd)) {

perror ("close");

return 1;

}

return 0;

}