Socket Options

Sanjaya Kumar Jena

Computer Sc. & Engineering
Faculty of Engineering, ITER
SOA, Deemed To Be University
Bhubaneswar-30

Introduction

There are various ways to get and set the options that affect a socket:

- The **`getsockopt`** and **`setsockopt`** functions
- The **`fcntl`** function
- The **`ioctl`** function

Here, we will cover the **`getsockopt`** and **`setsockopt`** functions.

Socket options for following categories: generic, IPV4, IPV6, TCP, and SCTP

`getsockopt` Function

These two functions apply only to sockets.

```
#include <sys/socket.h>

int getsockopt (int sockfd, int level, int optname
    , void * optval, socklen_t *optlen);

returns: 0 if OK, -1 on error
```

- **`sockfd`** must refer to an open socket descriptor
- **`level`** specifies the code in the system that interprets the option: the general socket code or some protocol-specific code (e.g., IPv4, IPv6, TCP, or SCTP).
- **`optval`** is a pointer to a variable into which the current value of the option is stored by **`getsockopt`**
- The size of this variable is specified by the final argument, as a value-result for **`getsockopt`**

`setsockopt` Function

These two functions apply only to sockets.

```
#include <sys/socket.h>

int setsockopt (int sockfd, int level, int optname
    , const void * optval socklen_t
optlen);

Both return: 0 if OK, -1 on error
```

- **`sockfd`** must refer to an open socket descriptor
- **`level`** specifies the code in the system that interprets the option: the general socket code or some protocol-specific code (e.g., IPv4, IPv6, TCP, or SCTP).
- **`optval`** is a pointer to a variable from which the new value of the option is fetched by **`setsockopt`**
- The size of this variable is specified by the final argument, as a value for **`setsockopt`**

Socket option that can be queried

level	optname	get	set	Description	Flag	Datatype void *optval
SOL_SOCKET	SO_RCVBUF	•	•	Receive buffer size		int
	SO_SNDBUF	•	•	send buffer size		int
	SO_REUSEADDR	•	•	Allow local address reuse	•	int
	SO_REUSEPORT	•	•	Allow local port reuse	•	int
	SO_LINGER	•	•	Linger on close if data to send		linger {}
	SO_SNDTIMEO	•	•	Send timeout		timeval {}
	SO_RCVTIMEO	•	•	Send timeout		timeval {}
IPPROTO_IP	⋮					
IPPROTO_ICMPV6	⋮					
IPPROTO_IPV6	⋮					
IPPROTO_IP or IPPROTO_IPV6	⋮					
IPPROTO_TCP	⋮					
IPPROTO_SCTP	⋮					

- ☞ The "**Datatype**" column shows the datatype of what the **`optval`** pointer must point to for each option.
- ☞ The notation of two braces to indicate a structure, as in **`linger`**{} to mean a **`struct linger`**

Generic Socket options

level	optname	get	set	Description	Flag	Datatype void *optval
SOL_SOCKET	SO_RCVBUF	•	•	Receive buffer size		int
	SO_SNDBUF	•	•	send buffer size		int
	SO_REUSEADDR	•	•	Allow local address reuse	•	int
	SO_REUSEPORT	•	•	Allow local port reuse	•	int
	SO_LINGER	•	•	Linger on close if data to send		linger {}
	SO_SNDTIMEO	•	•	Send timeout		timeval {}
	SO_RCVTIMEO	•	•	Send timeout		timeval {}
	SO_BROADCAST	•	•	Permit sending of broadcast datagram	•	int
	SO_DEBUG	•	•	Enable debug tracing	•	timeval {}
	SO_DONTROUTE	•	•	Bypass routing table lookup	•	int
	SO_ERROR	•		Get pending error and clear		int
	SO_KEEPALIVE	•	•	periodically test if connection still alive	•	int
	SO_OOBINLINE	•	•	Leave received out-of-band data inline	•	int
	SO_RCVLOWAT	•	•	Receive buffer low-water mark		int
	SO_SNDLOWAT	•	•	Send buffer low-water mark		int
	SO_TYPE	•		Get socket type		int
	SO_USELOOPBACK	•	•	Routing socket gets copy of what it sends	•	int

Summary of transport layer socket options

`level`	`optname`	get	set	Description	Flag	Datatype `void *optval`
IPPROTO_TCP	TCP_MAXSEG	•	•	TCP maximum segment size		int
	TCP_NODELAY	•	•	Disable Nagle algorithm	•	int
IPPROTO_SCTP	⋮					

getsockopt() Example

```
int main()
{
 int on,tn;
 int sockfd;
 socklen_t len;
 len=sizeof(socklen_t);
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 getsockopt(sockfd,SOL_SOCKET,SO_RCVBUF,&on,&len);
 printf("Reeive buffer size=%d\n",on);
 return 0;
}
```

setsockopt() Example

```
int main()
{
 int on,tn;
 int sockfd;
 socklen_t len;
 len=sizeof(socklen_t);
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 getsockopt(sockfd,SOL_SOCKET,SO_RCVBUF,&on,&len);
 printf("Reeive buffer size=%d\n",on);
 tn=65535;
 setsockopt(sockfd,SOL_SOCKET,SO_RCVBUF,&tn,len);
 getsockopt(sockfd,SOL_SOCKET,SO_SNDBUF,&tn,&len);
 printf("buffer size=%d\n",tn);
 return 0;
}
```

Setting SO_REUSEADDR Socket Option

A TCP socket is created that is bound to the server's well-known port and set the SO_REUSEADDR socket option in case connection exist on this port.

```
int on,listenfd;
listenfd=socket(AF_INET,SOCK_STREAM,0);
 : : : :
setsockopt(listenfd,SOL_SOCKET,SO_REUSEADDR,&on,
   sizeof(on));
```

Setting Receive Timeout Socket Option

```
int listenfd;
struct timeval tv;
tv.tv_sec=10;
tv.tv_usec=0;
listenfd=socket(AF_INET,SOCK_STREAM,0);
 : : : :
setsockopt(listenfd,SOL_SOCKET,SO_RCVTIMEO,&tv,
   sizeof(tv));
```

Setting Send Timeout Socket Option

```
int listenfd
struct timeval tv;
tv.tv_sec=10;
tv.tv_usec=0;
listenfd=socket(AF_INET,SOCK_STREAM,0);
  : : : :
setsockopt(listenfd,SOL_SOCKET,SO_SNDTIMEO,&tv,
    sizeof(tv));
```

THANK YOU