

Part 03. 리눅스 소켓 프로그래밍

# Chapter 05.

## 소켓 옵션 & 주소 정보

# 진행 순서

Chapter 05_01	소켓 옵션
Chapter 05_02	소켓 주소 정보

## Chapter 05\_01    소켓 옵션

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

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

getsockopt() 및 setsockopt()는 파일 디스크립터 sockfd가 참조하는 소켓의 옵션을 조작합니다.  
옵션은 여러 프로토콜 수준에 존재할 수 있습니다.

소켓 옵션을 조작 할 때 옵션이 상주하는 레벨과 옵션 이름을 지정해야 합니다.  
소켓 API 레벨에서 옵션을 조작하기 위해 레벨은 SOL\_SOCKET으로 지정됩니다.  
다른 레벨에서 옵션을 조작하기 위해 옵션을 제어하는 적절한 프로토콜의 프로토콜 번호가 제공됩니다.  
예를 들어, 옵션이 TCP 프로토콜에 의해 해석됨을 나타내려면,  
레벨은 TCP의 프로토콜 번호(IPPROTO\_TCP)로 설정되어야 합니다.

optval 및 optlen 인수는 setsockopt()의 옵션 값에 액세스하는 데 사용됩니다.  
getsockopt()의 경우 요청 된 옵션의 값이 리턴 될 버퍼를 식별합니다.  
getsockopt()의 경우, optlen은 초기에 optval이 가리키는 버퍼의 크기를 포함하고 반환 된 값의 실제 크기를 표시하도  
록 수정 된 값 결과 인수입니다. 옵션 값을 제공하거나 반환하지 않으면 optval은 NULL 일 수 있습니다.  
대부분의 소켓 레벨 옵션은 optval에 int 인수를 사용합니다.

성공하면 0이 반환됩니다. 오류가 발생하면 -1이 반환되고 errno가 적절하게 설정됩니다.

Chapter 05\_01      소켓 옵션

소켓 레벨 (SOL\_SOCKET) 옵션  
# man 7 socket 참조

TCP 레벨 (IPPROTO\_TCP) 옵션  
# man 7 tcp 참조

IP 레벨 (IPPROTO\_IP) 옵션  
# man 7 ip 참조

- SO\_ACCEPTCONN
- SO\_BINDTODEVICE
- SO\_BROADCAST
- SO\_BSDCOMPAT
- SO\_DEBUG
- SO\_DOMAIN
- SO\_ERROR
- SO\_DONTROUTE
- SO\_KEEPALIVE (\*)
- SO\_LINGER (\*)
- SO\_MARK
- SO\_OOBINLINE
- SO\_PASSCRED
- SO\_PEEK\_OFF
- SO\_PEERCREC
- SO\_PRIORITY
- SO\_PROTOCOL
- SO\_RCVBUF (\*)
- SO\_RCVBUFFORCE
- SO\_RCVLOWAT / SO\_SNDLOWAT
- SO\_RCVTIMEO / SO\_SNDTIMEO (\*)
- SO\_REUSEADDR (\*)
- SO\_REUSEPORT
- SO\_SNDBUF (\*)
- SO\_SNDBUFFORCE
- SO\_TIMESTAMP
- SO\_TYPE
- TCP\_CONGESTION
- TCP\_CORK
- TCP\_DEFER\_ACCEPT
- TCP\_INFO
- TCP\_KEEPCNT
- TCP\_KEEPIDL
- TCP\_KEEPINTVL
- TCP\_LINGER2
- TCP\_MAXSEG (\*)
- TCP\_NODELAY (\*)
- TCP\_QUICKACK
- TCP\_SYNCNT
- TCP\_USER\_TIMEOUT
- TCP\_WINDOW\_CLAMP

- IP\_ADD\_MEMBERSHIP
- IP\_ADD\_SOURCE\_MEMBERSHIP
- IP\_BLOCK\_SOURCE
- IP\_DROP\_MEMBERSHIP
- IP\_DROP\_SOURCE\_MEMBERSHIP
- IP\_FREEBIND
- IP\_HDRINCL
- IP\_MSFILTER
- IP\_MTU
- IP\_MTU\_DISCOVER
- IP\_MULTICAST\_ALL
- IP\_MULTICAST\_IF
- IP\_MULTICAST\_LOOP
- IP\_MULTICAST\_TTL
- IP\_NODEFRAG
- IP\_OPTIONS
- IP\_PKTINFO
- IP\_RECVERR
- IP\_RECVOPTS
- IP\_RECVORIGDSTADDR
- IP\_RECVTOS
- IP\_RECVTTL
- IP\_RETOPTS
- IP\_ROUTER\_ALERT
- IP\_TOS
- IP\_TRANSPARENT
- IP\_TTL
- IP\_UNBLOCK\_SOURCE

## Chapter 05\_01 소켓 옵션

## SO\_KEEPALIVE

연결 지향 소켓(TCP)에서 연결 유지 메시지를 보낼 수 있습니다. (0이면 off, 0이 아니면 on)  
일정 시간마다 상태확인을 위한 패킷을 보내 세션을 유지할 수 있습니다.

```
Int sockopt = 1;
If (setsockopt(fd, SOL_SOCKET, SO_KEEPALIVE, &sockopt, sizeof(sockopt)) == -1) { /* error */ }
```

## SO\_LINGER

인자로 linger 구조체 사용

```
struct linger {
    int l_onoff; /* linger active */
    int l_linger; /* how many seconds to linger for */
};
```

on 시 close() 또는 shutdown() 시 소켓에 대해 대기중인 모든 메시지가 성공적으로 전송되거나  
지연 시간 초과에 도달 할 때까지 반환되지 않습니다.

off 시 호출이 즉시 리턴되고 백그라운드에서 종료됩니다. (송신버퍼에 데이터가 남아있어도 즉시 파괴)

```
struct linger so_linger = {.l_onoff=1, .l_linger=0};
if (setsockopt(fd, SOL_SOCKET, SO_LINGER, &so_linger, sizeof(so_linger)) == -1) { /* error */ }
```

## SO\_RCVBUF / SO\_SNDBUF

소켓 송수신 버퍼 최대 크기를 바이트 단위로 설정하거나 가져옵니다.

커널은 setsockopt를 사용하여 설정 한 경우 이 값을 두 배로 늘리고 (부가적인 정보를 위해 사용)  
getsockopt에 의해 이 두 배의 값을 반환합니다.

기본값은 /proc/sys/net/core/rmem\_default 파일에 의해 설정되며

허용되는 최대 값은 /proc/sys/net/core/rmem\_max 파일에 의해 설정됩니다.

이 옵션의 최소값 (두배)은 256입니다.

## Chapter 05\_01     소켓 옵션

## SO\_RCVTIMEO / SO\_SNDTIMEO

인자로 `timeval` 구조체를 사용하며,  
송수신 관련 함수들이 블로킹 모드에서 동작할 때 타임아웃 시간을 지정합니다.  
타임아웃 시간이 지나도록 리턴되지 않으면 블록된 함수는 에러리턴(EAGAIN) 됩니다.

## SO\_REUSEADDR

`bind()` 호출에 제공된 주소가 로컬 주소로 재사용 가능함을 나타냅니다.  
AF\_INET 소켓의 경우 LISTENING 상태의 소켓을 제외하고 바인드 될 수 있음을 의미합니다.  
(TIME\_WAIT 상태의 소켓도 강제로 바인드하여 재사용 가능)

## TCP\_MAXSEG

송신 TCP 패킷의 최대 세그먼트 크기(MSS)입니다.  
인터페이스 MTU보다 큰 값은 적용되지 않습니다.  
(MTU 1500 인 경우, TCP 헤더 20, IP 헤더 20 바이트 제외, MSS는 1460 바이트)  
TCP는 또한 제공된 값보다 최소 및 최대 경계를 부과합니다.

## TCP\_NODELAY

설정된 경우 Nagle 알고리즘을 비활성화합니다.  
즉, 적은 양의 데이터만 있더라도 세그먼트는 가능한 한 빨리 전송됩니다.  
설정하지 않으면 전송하기에 충분한 양이 될 때까지 데이터가 버퍼링되어  
작은 패킷이 자주 전송되지 않으므로 네트워크 사용률이 떨어집니다.

## Chapter 05\_02     소켓 주소 정보

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

```
int getsockname(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

getsockname()은 addr가 가리키는 버퍼에 소켓 sockfd가 바인드 된 현재 주소를 리턴합니다.

addrlen 인수는 addr이 가리키는 공간 크기 (바이트)를 나타내도록 초기화되어야 합니다. 리턴 시 소켓 주소의 실제 크기를 포함합니다.

제공된 버퍼가 너무 작으면 리턴 된 주소가 잘립니다. 이 경우 addrlen은 호출에 제공된 것보다 큰 값을 반환합니다.

성공하면 0이 반환됩니다. 오류가 발생하면 -1이 반환되고 errno가 적절하게 설정됩니다.

## Chapter 05\_02

## 소켓 주소 정보

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(int argc, char *argv[])
{
    char myIP[16];
    unsigned int myPort;
    struct sockaddr_in server_addr, my_addr;
    int sockfd;

    if (argc < 3) {
        printf("Usage: %s <IP> <Port>\n", argv[0]);
        exit(1);
    }

    // Connect to server
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket error");
        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(argv[1]);
server_addr.sin_port = htons(atoi(argv[2]));

```

```

// Connect to server
if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
    perror("connect error");
    close(sockfd);
    exit(1);
}

```

```

// Get my ip address and port
bzero(&my_addr, sizeof(my_addr));
int 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);

```

```

close(sockfd);
return 0;
}

```



## Chapter 05\_02

## 소켓 주소 정보

```
[root@localhost ch02]# ls
client client.c server server.c
[root@localhost ch02]#
[root@localhost ch02]# ./server
Usage: ./server <port>
[root@localhost ch02]# ./server 80
accepted host(IP: 127.0.0.1, Port: 37754)
finish.
[root@localhost ch02]#
```

```
CentOS
[parallels@localhost ch04]$ gcc -g getsockname_example.c -o getsockname_example
[parallels@localhost ch04]$ ./getsockname_example
Usage: ./getsockname_example <IP> <Port>
[parallels@localhost ch04]$ ./getsockname_example 127.0.0.1 80
Local ip address: 127.0.0.1
Local port : 37754
[parallels@localhost ch04]$
```

## Chapter 05\_02     소켓 주소 정보

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

```
int getpeername(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

getpeername()은 addr가 가리키는 버퍼에 소켓 sockfd에 연결된 피어의 주소를 리턴합니다.

addrlen 인수는 addr이 가리키는 공간 크기 (바이트)를 나타내도록 초기화되어야 합니다. 리턴 시 소켓 주소의 실제 크기를 포함합니다.

제공된 버퍼가 너무 작으면 리턴 된 주소가 잘립니다. 이 경우 addrlen은 호출에 제공된 것보다 큰 값을 반환합니다.

성공하면 0이 반환됩니다. 오류가 발생하면 -1이 반환되고 errno가 적절하게 설정됩니다.

## Chapter 05\_02

## 소켓 주소 정보

```

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int sockfd;
    int client_sockfd;
    int client_len;
    int state;
    struct sockaddr_in clientaddr, serveraddr, myaddr, test;

    if (argc < 2) {
        printf("Usage: %s <Port>\n", argv[0]);
        exit(1);
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket error");
        exit(1);
    }

```

```

memset(&serveraddr, 0, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons(atoi(argv[1]));

state = bind(sockfd, (struct sockaddr *)&serveraddr, sizeof(serveraddr));
if (state == -1) {
    perror("bind error");
    exit(0);
}

state = listen(sockfd, 5);
if (state == -1) {
    perror("listen error");
    exit(0);
}

client_len = sizeof(clientaddr);
client_sockfd = accept(sockfd, (struct sockaddr *)&clientaddr, &client_len);

getpeername(client_sockfd, (struct sockaddr *)&myaddr, &client_len);

printf("Port   : %d\n", ntohs(myaddr.sin_port));
printf("address : %s\n", inet_ntoa(myaddr.sin_addr));

close(client_sockfd);
return 1;
}

```

## Chapter 05\_02

## 소켓 주소 정보

```
CentOS
[root@localhost ch04]# gcc -g getpeername_example.c -o getpeername_example
[root@localhost ch04]# ./getpeername_example
Usage: ./getpeername_example <Port>
[root@localhost ch04]# ./getpeername_example 8888
Port      : 46544
address   : 127.0.0.1
[root@localhost ch04]#
```

```
CentOS
[parallels@localhost ch04]$ ./getsockname_example 127.0.0.1 8888
Local ip address: 127.0.0.1
Local port : 46544
[parallels@localhost ch04]$
```