Part 03. 리눅스 소켓 프로그래밍

Chapter 03.
UDP & 유닉스 도메인 소켓

진행 순서

Chapter 03_01	UDP 소켓 통신 개요
Chapter 03_02	sendto
Chapter 03_03	recvfrom
Chapter 03_04	UDP 소켓 통신 실습
Chapter 03_05	유닉스 도메인 소켓

Chapter 03_01 UDP 소켓 통신 개요

UDP (User Datagram Protocol)
UDP는 RFC 768에 문서화 된 간단한 메시지 지향 전송 계층 프로토콜입니다.
UDP는 헤더 및 페이로드의 무결성 검사 (체크섬을 통해)를 제공하지만 메시지 전달 및 UDP에 대해 상위 계층 프로토콜을 보장하지 않습니다. 계층은 일단 전송 된 UDP 메시지의 상태를 유지하지 않습니다. 이러한 이유로 UDP를 신뢰할 수 없는 데이터 그램 프로토콜 이라고도 합니다.

UDP datagram header

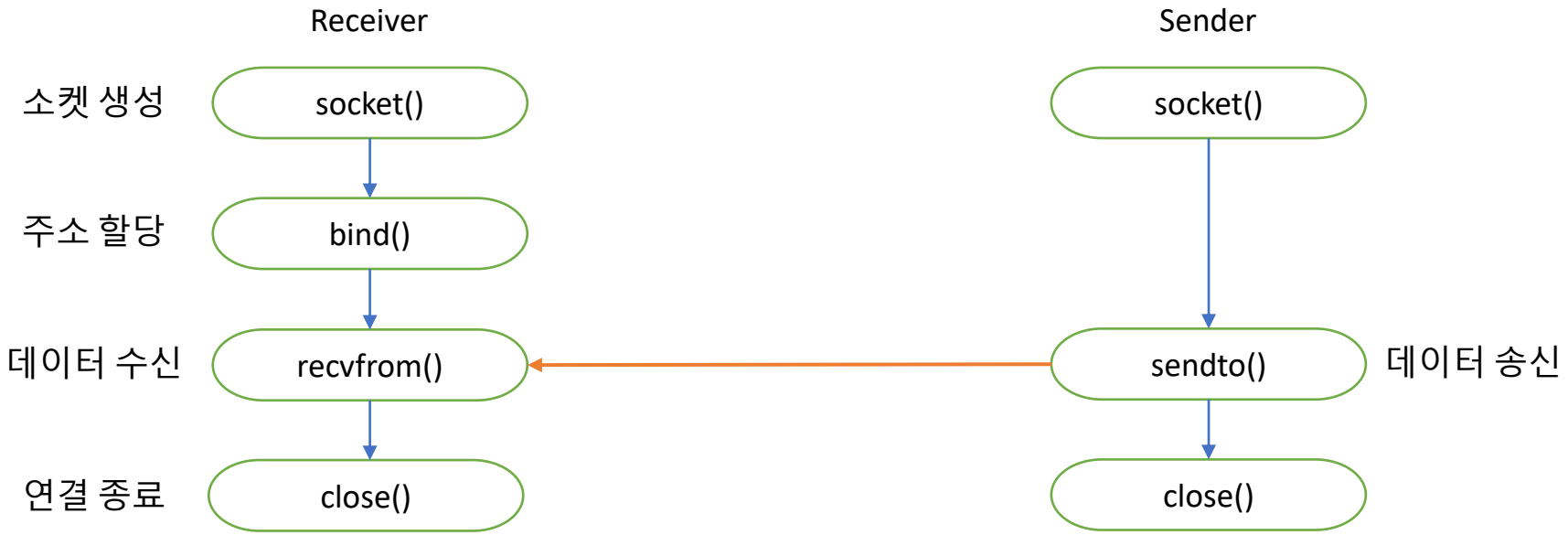
Offsets Octet		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Length																Checksum														

```
</usr/src/kernels/3.10.0-862.11.6.el7.x86_64/include/uapi/linux/udp.h>
struct udphdr {
        __be16  source;
        __be16  dest;
        __be16  len;
        __sum16 check;
};
```

Chapter 03_01 UDP 소켓 통신 개요

UDP 소켓은 비연결 지향형(connectionless), 연결 과정이 없으므로 사용이 간단하고 빠릅니다.
가벼우므로 실시간성을 요구하는 음성/화상 등 스트리밍 데이터에 많이 사용됩니다.
이에 반해 전송 확인이 어려우며 재전송 메커니즘 등이 존재하지 않아 신뢰성은 떨어집니다.

UDP 소켓 통신 함수 호출 흐름



Chapter 03_02 sendto

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

ssize_t sendto (int sockfd,  const void *buf,  size_t len,  int flags,  const struct sockaddr *dest_addr,  socklen_t addrlen);
```

sendto() 시스템 호출은 다른 소켓으로 메시지를 전송하는 데 사용됩니다.

sockfd 인수는 송신 소켓의 파일 디스크립터입니다.
송신 대상 주소는 크기를 지정하는 addrlen과 함께 dest_addr에 의해 제공됩니다.
메시지는 buf에 있으며 길이는 len입니다.

전송 시 사용할 작동 플래그를 지정하는 flags는 send()의 flags와 동일합니다.

성공하면 호출은 전송 된 문자 수를 반환합니다. 오류가 발생하면 -1이 반환되고 errno가 적절하게 설정됩니다.

Chapter 03_03 recvfrom

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

ssize_t recvfrom (int sockfd,  void *buf,  size_t len,  int flags,  struct sockaddr *src_addr,  socklen_t *addrlen);
```

recvfrom() 호출은 소켓에서 메시지를 수신하는 데 사용되며
연결 지향 여부에 관계없이 소켓에서 데이터를 수신하는 데 사용될 수 있습니다.

src_addr이 NULL이 아니고 기본 프로토콜이 출발지 주소를 제공하면 이 출발지 주소가 채워집니다.
src_addr이 NULL이면 아무 것도 채워지지 않습니다. 이 경우 addrlen이 사용되지 않으며 NULL 이어야 합니다.
addrlen 인수는 value-result 인수로, 호출자는 src_addr와 연관된 버퍼 크기를 호출하기 전에 초기화 해야하며
출발지 주소의 실제 크기를 나타내도록 리턴시 수정되어야 합니다.
제공된 버퍼가 너무 작으면 리턴된 주소가 잘립니다. 이 경우 addrlen은 호출에 제공된 것보다 큰 값을 반환합니다.

성공적으로 완료되면 메시지 길이를 리턴합니다.
메시지가 제공된 버퍼에 비해 너무 길면 메시지를 받는 소켓 유형에 따라 초과 바이트가 삭제 될 수 있습니다.
오류가 발생한 경우 -1을 리턴합니다. 오류가 발생하면 오류를 표시하도록 errno가 설정됩니다.

Chapter 03_04 UDP 소켓 통신 실습

receiver.c

```
#include <stdio.h>
#include <unistd.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;
   struct sockaddr_in receiver_addr;
   struct sockaddr_in sender_addr;
   int sender_len;
   char buff[1024] = {0,};
   int recvlen, sendlen;

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

```
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  if (sockfd == -1) {
     perror("socket() error");
     exit(1);
  }

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

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

```
while(1) {
    memset(buff, 0, sizeof(buff));
    sender_len = sizeof(sender_addr);
    recvlen = recvfrom(sockfd, buff, sizeof(buff)-1, 0, (struct sockaddr *)&sender_addr, &sender_len);
    if (recvlen == -1) {
       perror("recvfrom() error");
       exit(1);
    }

    buff[recvlen] = '\0';
    if (strcmp(buff, ".\n") == 0)
       break;

    printf("Received Data: %s\n", buff);

    snprintf(buff, sizeof(buff), "OK!");
    sendlen = strlen(buff);

    if (sendto(sockfd, buff, sendlen, 0, (struct sockaddr *)&sender_addr, sizeof(sender_addr)) != sendlen) {
       perror("sendto() error");
       exit(1);
    }
  }

  close(sockfd);
  return 0;
}
```

receiver.c

```
#include <stdio.h>
#include <unistd.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;
   struct sockaddr_in receiver_addr;
   char buff[1024] = {0,};
   int bufflen, recvlen, receiver_len;

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

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

   memset(&receiver_addr, 0, sizeof(receiver_addr));
   receiver_addr.sin_family = AF_INET;
   receiver_addr.sin_addr.s_addr = inet_addr(argv[1]);
   receiver_addr.sin_port = htons(atoi(argv[2]));
```

sender.c

sender.c

```
while(1) {
        printf("Input message (quit: .): ");
        memset(buff, 0, sizeof(buff));
        fgets(buff, sizeof(buff), stdin);
        bufflen = strlen(buff);

        if (sendto(sockfd, buff, bufflen, 0, (struct sockaddr *)&receiver_addr, sizeof(receiver_addr)) != bufflen) {
            perror("sendto() error");
            exit(1);
        }

        if (strcmp(buff, ".\n") == 0)
            break;

        recvlen = recvfrom(sockfd, buff, sizeof(buff)-1, 0, (struct sockaddr *)&receiver_addr, &receiver_len);
        if (recvlen == -1) {
            perror("recvfrom() error");
            exit(1);
        }

        buff[recvlen] = '\0';
        printf("Reply received: %s\n", buff);
    }

    close(sockfd);
    return 0;
}
```

Chapter 03_04 UDP 소켓 통신 실습

```
[parallels@localhost ch03]$ gcc -g receiver.c -o receiver
[parallels@localhost ch03]$ ./receiver
Usage: ./receiver <port>
[parallels@localhost ch03]$ ./receiver 1234
Received Data: hello

Received Data: world

[parallels@localhost ch03]$
```

```
[parallels@localhost ch03]$ gcc -g sender.c -o sender
[parallels@localhost ch03]$ ./sender
Usage: ./sender <IP> <Port>
[parallels@localhost ch03]$ ./sender 127.0.0.1 1234
Input message (quit: .): hello
Reply received: OK!
Input message (quit: .): world
Reply received: OK!
Input message (quit: .): .
[parallels@localhost ch03]$
```

Chapter 03_05 유닉스 도메인 소켓

유닉스 도메인 소켓이란?
도메인의 범위가 유닉스 시스템에 국한됩니다. 즉 로컬 호스트에 국한되므로 외부에서 접속할 수 없습니다.
유닉스 도메인 소켓은 IPC의 개념으로 사용됩니다.
유닉스 도메인 소켓은 소켓 생성 시 도메인을 AF_UNIX로 생성 후 bind 호출 시 소켓에 연결된 파일이 생성됩니다.

```
struct sockaddr_un {
   sa_family_t sun_family;            /* AF_UNIX */
   char       sun_path[108];          /* Pathname */
};

int sfd, cfd;
struct sockaddr_un my_addr;

sfd = socket(AF_UNIX, SOCK_STREAM, 0);
if (sfd == -1)
   handle_error("socket");

memset(&my_addr, 0, sizeof(struct sockaddr_un)); /* Clear structure */
my_addr.sun_family = AF_UNIX;
strncpy(my_addr.sun_path, MY_SOCK_PATH, sizeof(my_addr.sun_path) - 1); /* 유닉스 도메인 소켓 경로 */

if (bind(sfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr_un)) == -1)
   handle_error("bind");
…
```

Chapter 03_05 유닉스 도메인 소켓

The following example shows how to bind a stream socket in the UNIX (AF_UNIX) domain, and accept connections:

```
#include <sys/socket.h>
#include <sys/un.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define MY_SOCK_PATH "/somepath"
#define LISTEN_BACKLOG 50

#define handle_error(msg) \
   do { perror(msg); exit(EXIT_FAILURE); } while (0)

int main(int argc, char *argv[])
{
   int sfd, cfd;
   struct sockaddr_un my_addr, peer_addr;
   socklen_t peer_addr_size;

   sfd = socket(AF_UNIX, SOCK_STREAM, 0);
   if (sfd == -1)
      handle_error("socket");
```

```
   memset(&my_addr, 0, sizeof(struct sockaddr_un)); /* Clear structure */
   my_addr.sun_family = AF_UNIX;
   strncpy(my_addr.sun_path, MY_SOCK_PATH,
   sizeof(my_addr.sun_path) - 1);

   if (bind(sfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr_un)) == -1)
      handle_error("bind");

   if (listen(sfd, LISTEN_BACKLOG) == -1)
      handle_error("listen");

   /* Now we can accept incoming connections one at a time using accept(2) */
   peer_addr_size = sizeof(struct sockaddr_un);
   cfd = accept(sfd, (struct sockaddr *) &peer_addr, &peer_addr_size);
   if (cfd == -1)
      handle_error("accept");

   /* Code to deal with incoming connection(s)... */

   /* When no longer required, the socket pathname, MY_SOCK_PATH
      should be deleted using unlink(2) or remove(3) */
}
```