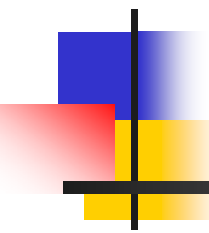
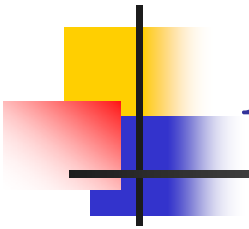


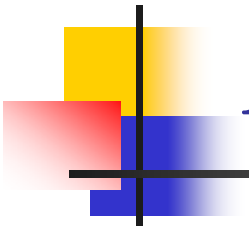
Unix Network Programming
Chapter 14. Advanced I/O Functions

광운대학교 컴퓨터과학과
정보통신 연구실
석사과정 안중현



14.1 Introduction

- 이 장에서 소개되고 있는 내용
  - I/O operation에서 timeout을 설정하는 세가지 방법
  - 세가지 Read/Write 관련 함수
    - recv/send
    - readv/writev
    - recvmsg/sendmsg
  - Ancillary data
  - Socket receive buffer에서 얼마나 많은 데이터를 받을지 알아내는 방법
  - 소켓에서 C 표준 I/O 라이브러리를 사용하는 방법
  - 이벤트를 기다리기 위한 좀더 진보된 방식



14.2 Socket Timeouts

- Socket timeout을 설정하는 방법
  - SIGALRM signal 발생시켜 alarm 호출
    - 정해진 시간이 끝나면 SIGALRM signal 발생시킴.
    - 다른 signal과 구분하기 위한 signal handling 포함
    - 프로세스에서 발생하는 다른 alarm에 의해 영향을 받음
  - select 입출력 Block
    - read write 호출을 blocking 하는 대신에 시간제한을 둔 select에서 입출력을 기다리는 것을 막음
  - 새로운 SO_RCVTIMEO 와 SO_SNDTIMEO 소켓 옵션 사용
    - 구현 시 소켓옵션을 지원하지 않을 수 있다는 문제점

Connect with a Timeout Using SIGALRM

지정한 시간이 지나면 connect를 호출하는 함수

```
#include     "unp.h"

static void  connect_alarm(int);

int connect_timeo(int sockfd, const SA *saptr, socklen_t salen, int nsec)
{
   Sigfunc   *sigfunc;
   int       n;

   sigfunc = Signal(SIGALRM, connect_alarm);
   if (alarm(nsec) != 0)
     err_msg("connect_timeo: alarm was already set");

   if ( (n = connect(sockfd, (struct sockaddr *) saptr, salen)) < 0) {
     close(sockfd);
     if (errno == EINTR)
             errno = ETIMEDOUT;
   }
   alarm(0);                    /* turn off the alarm */
   Signal(SIGALRM, sigfunc);   /* restore previous signal handler */

   return(n);
}

static void
connect_alarm(int signo)
{
   return;            /* just interrupt the connect() */
}
```

Connect with a Timeout Using SIGALRM

```
// 현재 signal handler를 저장
sigfunc = Signal(SIGALRM, connect_alarm);

// 지정한 시간만큼 알람 설정. nsec는 설정 시간
if (alarm(nsec) != 0)
      err_msg("connect_timeo: alarm was already set");

// connect 실패시 (-1 리턴) 소켓을 닫고, 인터럽트 걸렸을 경우 errono값을 ETIMEDOUT으로
      변경
if ( (n = connect(sockfd, (struct sockaddr *) saptr, salen)) < 0) {
      close(sockfd);
      if (errno == EINTR)
            errno = ETIMEDOUT;
}

// 알람을 끄고 이전 signal handler로 복구
alarm(0);                              /* turn off the alarm */
Signal(SIGALRM, sigfunc);              /* restore previous signal handler */

// signal 처리
connect_alarm(int signo)
{
      return;              /* just interrupt the connect() */
}
```

recvfrom with a Timeout Using SIGALRM

```
#include      "unp.h"
static void   sig_alrm(int);

void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
      int       n;
      char      sendline[MAXLINE], recvline[MAXLINE + 1];

      Signal(SIGALRM, sig_alrm);

      while (Fgets(sendline, MAXLINE, fp) != NULL) {
            Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);
            alarm(5);
            if ( (n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL)) < 0) {
                  if (errno == EINTR)
                        fprintf(stderr, "socket timeout₩n");
                  else
                        err_sys("recvfrom error");
            } else {
                  alarm(0);
                  recvline[n] = 0;          /* null terminate */
                  Fputs(recvline, stdout);
            }
      }
}
static void sig_alrm(int signo)
{
      return;                           /* just interrupt the recvfrom() */
}
```

5초 내에 응답이 없으면 recvfrom을 일시 중지시키는 alarm함수를 호출

recvfrom with a Timeout Using SIGALRM

```
// SIGALRM을 위한 signal handler를 설정
Signal(SIGALRM, sig_alrm);

while (Fgets(sendline, MAXLINE, fp) != NULL) {
        Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);
        // recvfrom을 호출하기 전 5초간 기다림
        alarm(5);
        if ( (n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL)) < 0) {
                // 만약 일시 중지 된다면 에러 메시지 출력
                if (errno == EINTR)
                        fprintf(stderr, "socket timeout₩n");
                else
                        err_sys("recvfrom error");
        } else {
                // 서버로 부터 데이터를 받으면 alarm을 해제하고 받은 데이터 출력
                alarm(0);
                recvline[n] = 0;        /* null terminate */
                Fputs(recvline, stdout);
        }
}
// recvform을 일시 중지시키기 위해 사용
static void sig_alrm(int signo)
{
        return;                         /* just interrupt the recvfrom() */
}
```

recvfrom with a Timeout Using select

읽기가 가능해질 때까지
지정한 시간만큼 기다림

```
#include "unp.h"

int readable_timeo(int fd, int sec)
{
    fd_set          rset;
    struct timeval  tv;

    FD_ZERO(&rset);         // 파일 디스크립터 집합의 모든 비트를 0으로
    FD_SET(fd, &rset);      //

    tv.tv_sec = sec;
    tv.tv_usec = 0;

    // 파일 디스크립터 개수 혹은 에러 리턴
    return(select(fd+1, &rset, NULL, NULL, &tv));
            /* 4> 0 if descriptor is readable */
}
```

recvfrom with a Timeout Using select

그림 8.8에 Readable_timeo를 추가한 프로그램

```
#include "unp.h“

voiddg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen){
    int     n;
    char   sendline[MAXLINE], recvline[MAXLINE + 1];

    while (Fgets(sendline, MAXLINE, fp) != NULL) {
        Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);
        if (Readable_timeo(sockfd, 5) == 0) {
        fprintf(stderr, "socket timeout₩n");
        } else {
        n = Recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
        recvline[n] = 0;     /* null terminate */
        Fputs(recvline, stdout);
        }
    }
}
```

recvfrom with a Timeout Using the SO_RCVTIMEO Socket Optioin

```
void
dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
        int                     n;
        char                    sendline[MAXLINE], recvline[MAXLINE + 1];
        struct timeval          tv;

        tv.tv_sec = 5;
        tv.tv_usec = 0;
        Setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv));

        while (Fgets(sendline, MAXLINE, fp) != NULL) {

                Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);

                n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
                if (n < 0) {
                        if (errno == EWOULDBLOCK) {
                                fprintf(stderr, "socket timeout₩n");
                                continue;
                        } else
                                err_sys("recvfrom error");
                }
                recvline[n] = 0;        /* null terminate */
                Fputs(recvline, stdout);
        }
}
```

recvfrom with a Timeout Using the SO_RCVTIMEO Socket Optioin

```
// SO_REVTIMEO는 읽기에만 해당. 쓰기는 SO_SNDTIMEO.
// tv에 지정된 시간만큼 기다림.
Setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv));

// I/O 시간이 만료되면, EWOULDBLOCK을 errno로 지정하고 에러메세지 출력
n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
if (n < 0) {
    if (errno == EWOULDBLOCK) {
        fprintf(stderr, "socket timeout₩n");
        continue;
    } else
        err_sys("recvfrom error");
```

14.3 recv and send Functions

- 표준 함수 read, write와 유사하나 인수가 하나 더 필요

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

ssize_t recv (int sockfd, void *buff, size_t nbytes, int flags);

ssize_t send (int sockfd, const void *buff, size_t nbytes, int flags);
```

- 3번째 인수까지는 같음
- 4번째 인수 값

Flags	Description	recv	send
MSG_DONTROUTE	bypass routing table lookup		•
MSG_DONTWAIT	only this operation is nonblocking	•	•
MSG_OOB	send or receive out-of-band data	•	•
MSG_PEEK	peek at incoming message	•	
MSG_WAITALL	wait for all the data	•	

그림 14.6 flags for I/O functions

14.3 recv and send Functions

- MSG_DONTROUTE:
  - kernal 에게 routing table을 살펴 볼 필요가 없음을 알림.
- MSG_DONTWAIT:
  - I/O에 대하여 nonblocking을 설정하지 않고, I/O 를 수행하며, nonblocking 표시기를 해제 함을 나타낸다.
- MSG_OOB:
  - out-of-band 에서 데이터 송신을 지정한다.
- MSG_PEEK:
  - recv나 recvfrom에서 리턴후 데이터를 버리지 않고 살펴볼 수 있게 한다.
- MSG_WAITALL:
  - kernal에 요구한 바이트만큼 읽기 전에 return 되지 말것을 지시 한다.

14.4 readv and writev Functions

- read, write와 유사하나 한번 호출로 메모리에 흩어져 있는 버퍼로부터 읽고 쓸 수 있다.

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

ssize_t readv (int filedes, const struct iovec *iov, int iovcnt);

ssize_t writev (int filedes, const struct iovec *iov, int iovcnt);
```

```
Struct iovec {
        void    *iov_base;  /* starting address of buffer */
        size_t  iov_len;        /* size of buffer */
};
```

14.5 recvmsg and sendmsg Functions

- 입출력 함수의 가장 일반 적인 형태로 read, readv, recv, recvfrom을 recvmsg로 대체할 수 있다.
- 이 함수는 대부분의 인수를 msghdr에 통합시켰다.

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

ssize_t recvmsg (int sockfd, struct msghdr *msg, int flags);

ssize_t sendmsg (int sockfd, struct msghdr *msg, int flags);
```

```
Struct msghdr {
   void         *msg_name;      /* protocol address */
   socklen_t     msg_namelen;   /* size of protocol address */
   struct iovec *msg_iov;      /* scatter/gather array */
   size_t       msg_iovlen;    /* # elements in msg_iov */
   void         *msg_control;  /* ancillary data; must be aligned
                                  for a cmsghdr structure */
   socklen_t    msg_controllen;/* length of ancillary data */
   int          msg_flags;     /* flags returned by recvmsg() */
};
```

14.5 recvmsg and sendmsg Functions

- msghdr
  - Msg_name
    - 목적지나 송신자의 프로토콜 주소를 저장
    - Recvfrom과 sendto의 다섯번째와 여섯번째 인수
    - 주소가 지정될 필요가 없다면 NULL pointer로
  - msg_namelen
    - 프로토콜 주소의 크기
  - Msg_iov 와 msg_iovlen
    - 입력 또는 출력 버퍼배열을 지정.
  - Msg_control 과 msg_controllen
    - 부수적인 데이터(ancillary data)의 위치와 크기 지정.
  - Msg_flags
    - Msg_flag recvmsg만 사용
    - Recvmsg를 호출하면 flags 인수는 msg_flags로 복사된다
    - Msg_flag는 sendmsg에서 무시된다.

14.5 recvmsg and sendmsg Functions

Flag	Examined by: Send flags Sendto flags Sendmsg flags	Examined by: recv flags recvfrom flags recvmsg flags	Returned by: Recvmsg msg_flags
MSG_DONTROUTE	•		
MSG_DONTWAIT	•	•	
MSG_PEEK		•	
MSG_WAITALL		•	
MSG_EOR	•		•
MSG_OOB	•	•	•
MSG_BCAST			•
MSG_MCAST			•
MSG_TRUNC			•
MSG_CTRUNC			•

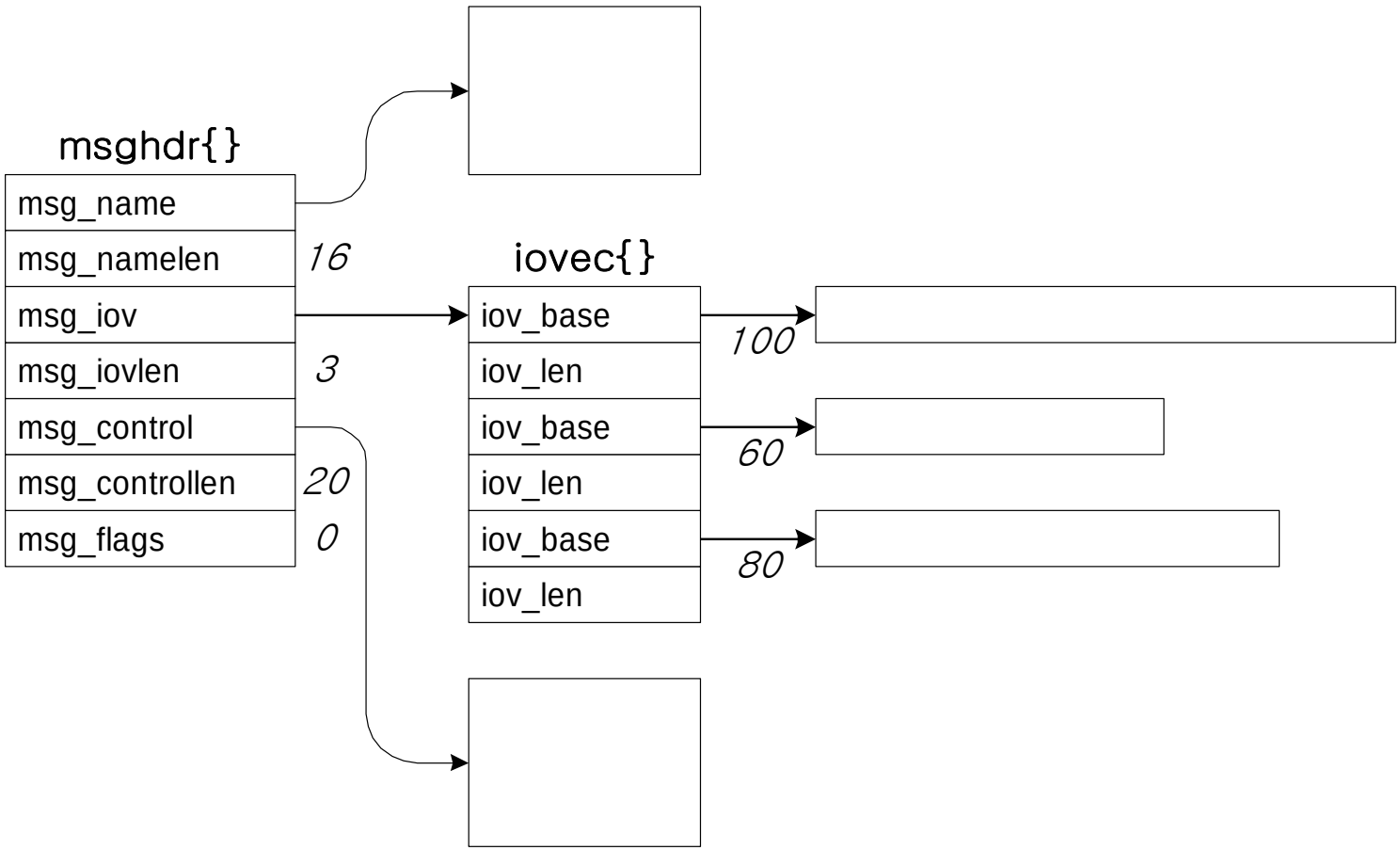
그림 14.7 Summary of input and output flags by various I/O functions

14.5 recvmsg and sendmsg Functions

- 처음 네게 flag는 검사될 뿐 리턴 되지 않는다.
- MSG_BCAST:
  - 이 flag는 BSD/OS에서 새로운 것으로 링크 레이어에서 브로드 케스트로 받다진다면 리턴 된다.
- MSG_MCAST:
  - 이 flag는 BSD/OS에서새로운 것으로 데이터그램이 링크 레이어의 멀티케스트로서 받아진다면 리턴된다.
- MSG_TRUNC:
  - 이 플러그는 데이터 그램이 잘려진다면 리턴된다.
- MSG_CTRUNC:
  - 이 플러그는 부수적 데이터가 잘려진다면 리턴 된다.
- MSG_EOR:
  - 리턴된 데이터가 논리적 데이터 끝이 아니면 해제된다.
- MSG_OOB:
  - 이것은 TCP out-of-band 데이터에서 절대 리턴 되지 않는다.

14.5 recvmsg and sendmsg Functions

- 프로세스가 UDP 소켓에 대해 recvmsg 함수를 호출하려는 때의 데이터 구조



14.8 Data structures when recvmsg is called for a UDP socket

14.5 recvmsg and sendmsg Functions

- 198.69.10.2에서 206.62.226.35로 170 바이트를 보낼 때 데이터

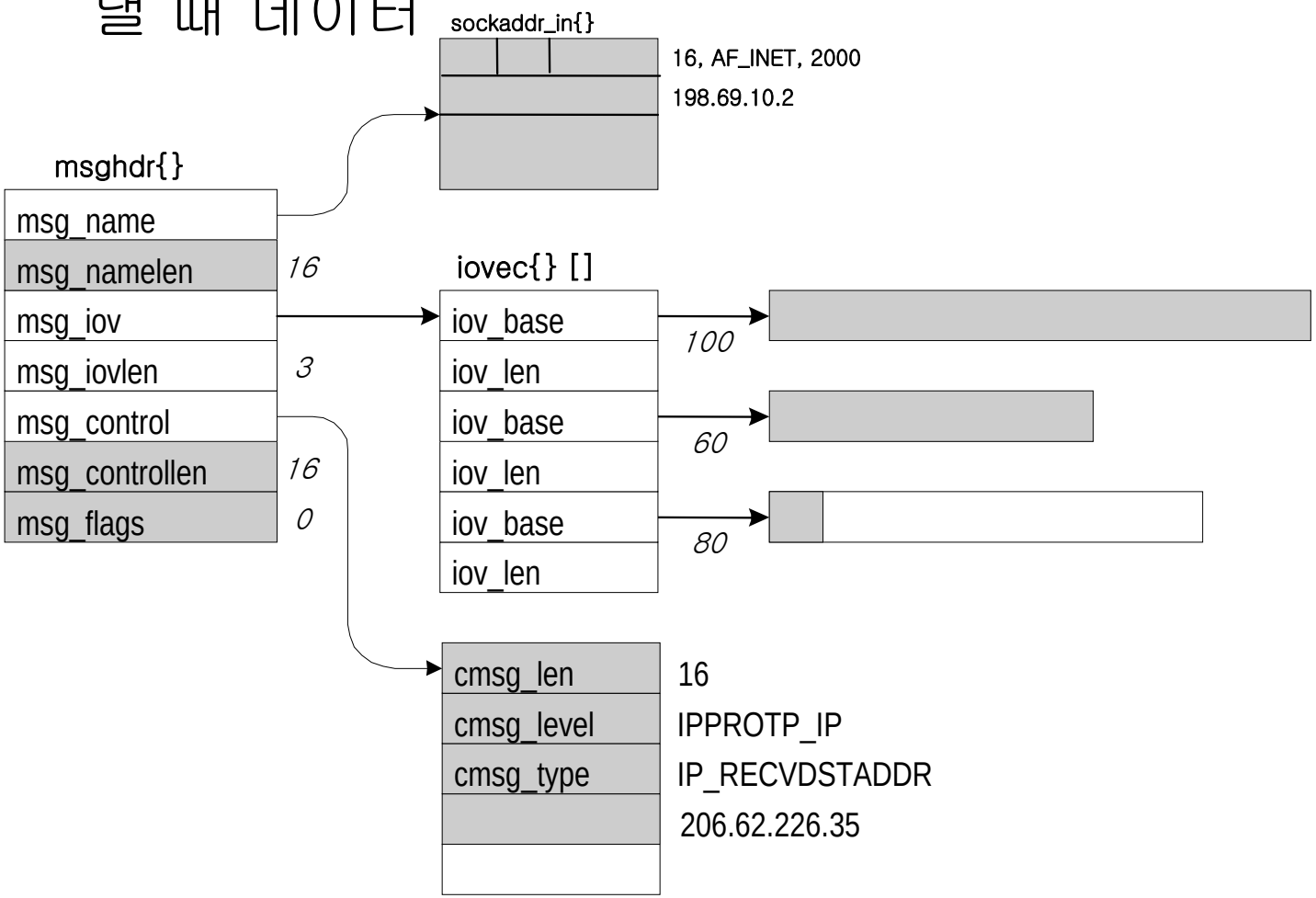


그림 14.9 Update of Figure 14.8 when recvmsg returns

14.6 Ancillary Data

- Ancillary data는 sendmsg와 recvmsg 함수로 msghdr 구조 원소를 사용하여 송수신할 수 있다
- 제어정보라고도 한다

```
struct cmsghrd {
    socklen_t cmsg_len;      // 이 구조체의 길이
    int cmsg_level;          // 전송측 프로토콜
    int cmsg_type;           // 프로토콜 세부 타입
```

Protocol	cmsg_level	Cmsg_type	Description
IPv4	IPPROTO_IP	IP_RECVDSTADDR IP_RECVIF	receive destination address with UDP datagram receive interface index with UDP datagram
IPv6	IPPROTO_IPV6	IPV6_DSTOPTS IPV6_HOPLIMIT IPV6_HOPOPTS IPV6_NEXTHOP IPV6_PKTINFO IPV6_RTHDR	specify / receive destination options specify / receive hop limit specify / receive hop-by-hop options specify next-hop address specify / receive packet information specify / receive routing header
Unix domain	SOL_SOCKET	SCM_RIGHTS SCM_CREDS	send / receive descriptors send / receive user credentials

Figure 14.11 summary of uses for ancillary data.

14.6 Ancillary Data

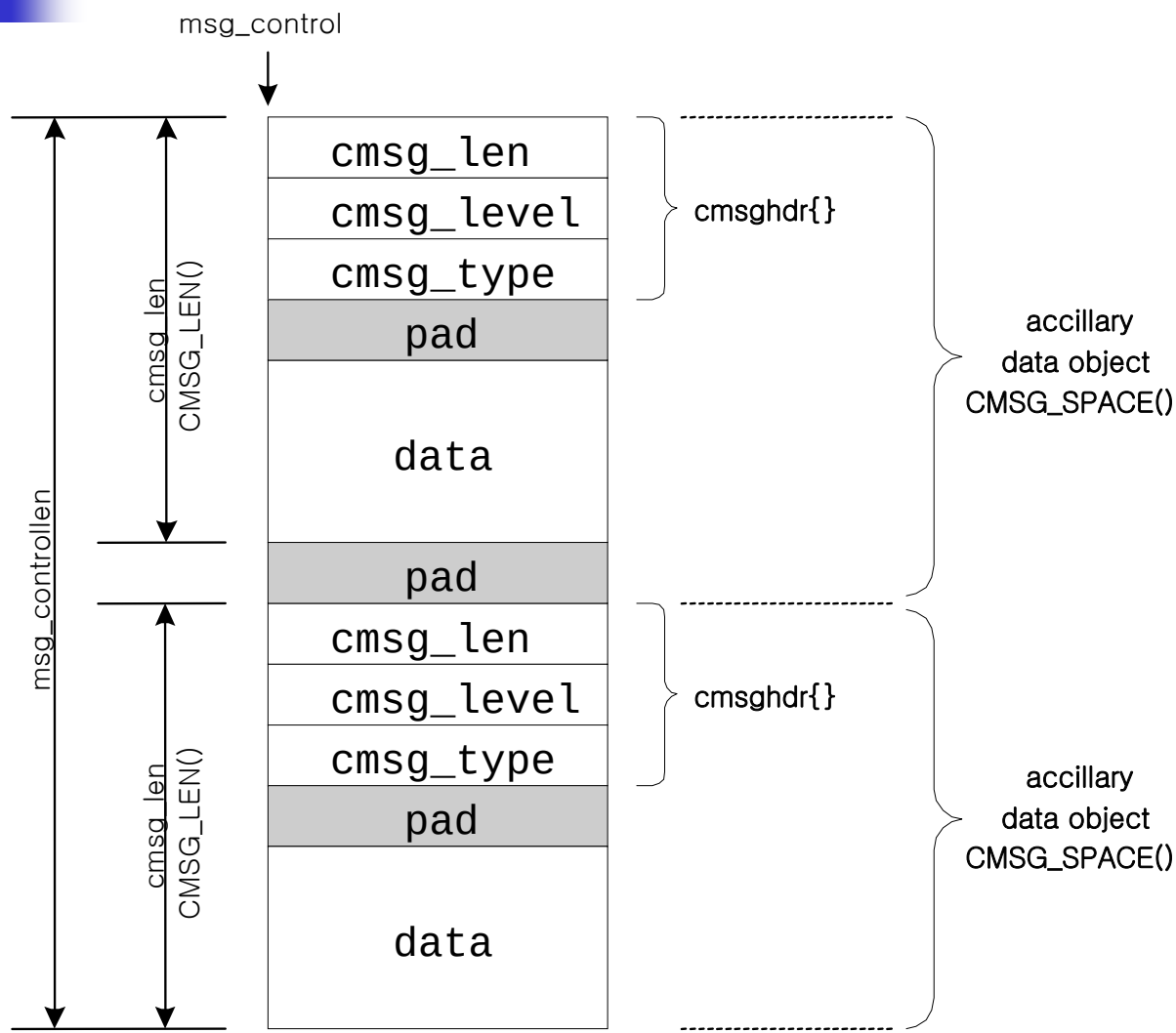
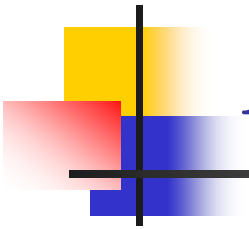


Figure 14.12 Ancillary data containing two ancillary data objects.

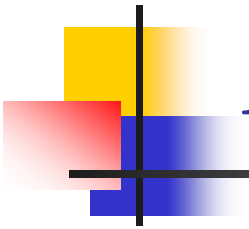
14.6 Ancillary Data

```
#include <sys/socket.h>
#include <sys/param.h> /* for ALIGN macro on many
    implementations */
struct cmsghdr *CMSG_FIRSTHDR(struct msghdr * mhdrptr );
struct cmsghdr *CMSG_NXTHDR(struct msghdr *mhdrptr , struct
    cmsghdr *cmsgptr );
unsigned char *CMSG_DATA(struct cmsghdr *cmsgptr );
unsigned int CMSG_LEN(unsigned int length);
unsigned int CMSG_SPACE(unsigned intlength );
```



14.7 How much Data Is Queued?

- 얼마나 많은 큐가 있는지 데이터를 읽지 않고 아는 세가지 방법.
  - 다른 일을 하려고 하기 때문에 목적이 kernel 에서 block 하는것 아니면 15장에서 설명
  - 읽기를 원하며 프로세스 다른 쪽에서 데이터를 읽을 수 있도록 수신 대기열에 남겨 두기 원한다면 mgs_peek를 사용한다.
  - 여러 implementation 은 ioctl FIONREAD의 command 를 지원한다.



14.8 Sockets and Standard I/O