13장 소켓을 이용한 통신(2)과 유용한 함수

- 서론
- 예제 프로그램
- 함수
  - recvfrom
  - sendto
  - uname
  - gethostname
  - gethostbyname
  - gethostbyaddr

한빛미디어(주)

- 소켓을 사용하여 비연결형 모델로 통신을 하기 위한 함수와 그 외의 함수

함수	의미
recvfrom	비연결형 모델에서 소켓을 통해 메시지를 수신한다.
sendto	비연결형 모델에서 소켓을 통해 메시지를 송신한다.
uname	호스트의 이름을 포함한 시스템의 정보를 알아온다.
gethostname	호스트의 이름을 알아온다.
gethostbyname	이름으로 지정한 호스트의 네트워크 정보를 알아온다.
gethostbyaddr	인터넷 주소로 지정한 호스트의 네트워크 정보를 알아온다.

【예제13-1】 ex 13-01s.c , 서버(1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04 #include <sys/utsname.h>
05 #include <netdb.h>
06
07 #define SIZE    sizeof(struct sockaddr_in)
08
09 main()
10 {
11    int sockfd;
12     char msg;
13
14     struct utsname info;
15     struct hostent *hent;
16
17     struct sockaddr_in server = {AF_INET, 2007, INADDR_ANY};
18     struct sockaddr_in client;
19     int client_len = SIZE;
```

표준입력 스트림

【예제13-1】 ex 13-01s.c , 서버(2/2)


```
20    uname(&info);
21    printf("node name : %s₩n", info.nodename);
22
23    hent = gethostbyname(info.nodename);
24    printf("official name : %s₩n", hent->h_name);
25
26    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
27
28    bind(sockfd, (struct sockaddr *)&server, SIZE);
29
30    recvfrom(sockfd, &msg, 1, 0, (struct sockaddr *)&client, &client_len);
31
32    printf("recv from client : %c₩n", msg);
33
34    sendto(sockfd, &msg, 1, 0, (struct sockaddr *)&client, client_len);
35
36    close(sockfd);
37 }
```

【예제13-1】 ex 13-01c.c , 클라이언트(2/2)

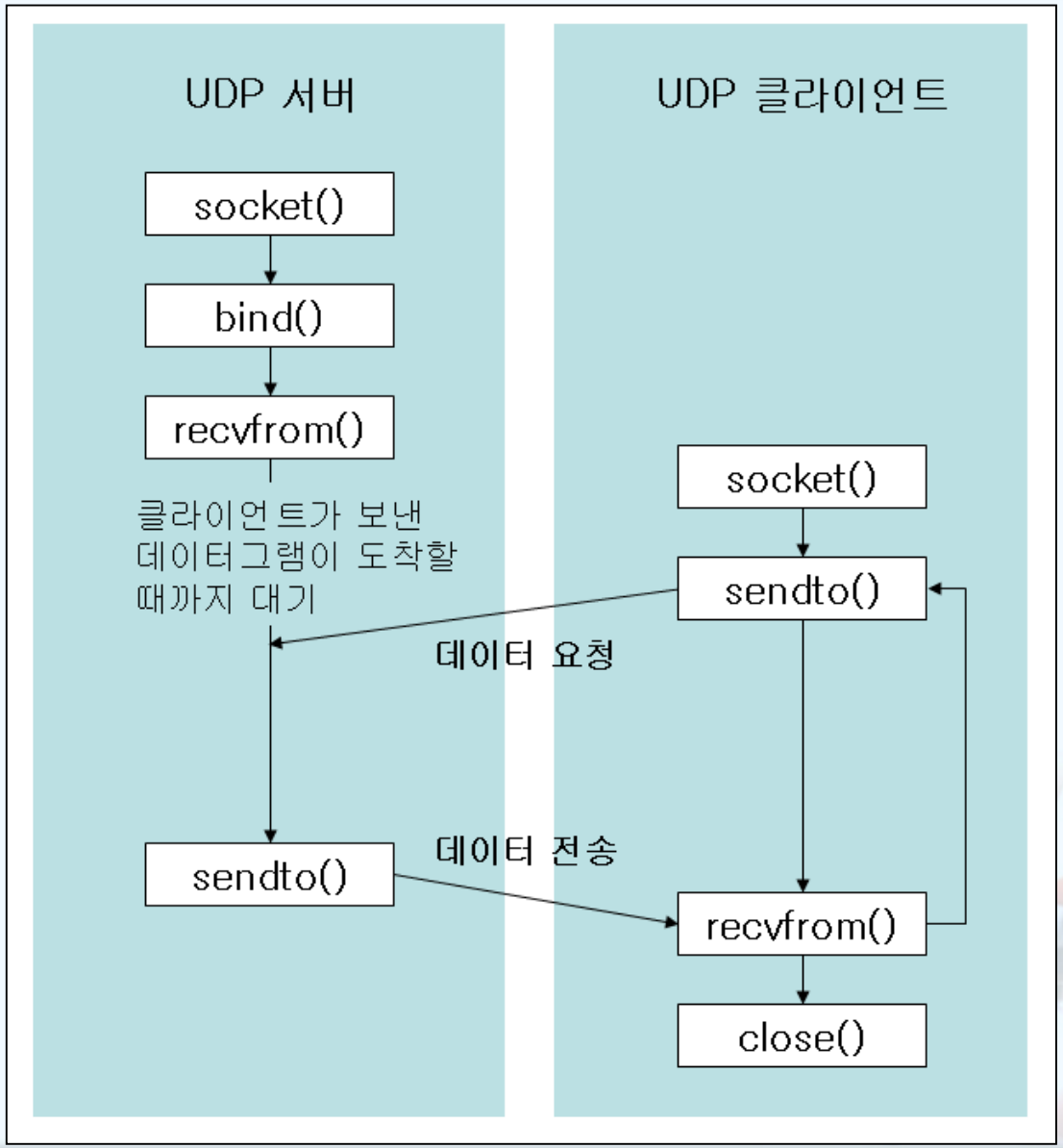

```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04 #include <netdb.h>
05 #include <unistd.h>
06
07 #define SIZE    sizeof(struct sockaddr_in)
08
09 main()
10 {
11    int sockfd;
12     char msg, hostname[1024], *ipaddr;
13
14     struct hostent *hent;
15     struct sockaddr_in client = {AF_INET, INADDR_ANY, INADDR_ANY};
16     struct sockaddr_in server = {AF_INET, 2007};
17     int server_len = SIZE;
```

표준입력 스트림

【예제13-1】 ex 13-01c.c , 클라이언트(2/2)


```
18    gethostname(hostname, 1024);
19    printf("hostname : %s₩n", hostname);
20
21     hent = gethostbyname(hostname);
22     ipaddr = inet_ntoa(*((struct in_addr *)*(hent->h_addr_list)));
23     printf("official name : %s₩n", hent->h_name);
24     printf("IP address : %s₩n", ipaddr);
25
26     server.sin_addr.s_addr = inet_addr(ipaddr);
27
28     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
29
30     msg = 'A';
31     sendto(sockfd, &msg, 1, 0, (struct sockaddr *)&server, server_len);
32     recvfrom(sockfd, &recv, 1, 0, (struct sockaddr *)&server, &server_len);
33     printf("recv from server : %c₩n", msg);
34     close(sockfd);
35 }
```

표준입력 스트림


UDP 서버
UDP 클라이언트
socket()
bind()
recvfrom()
클라이언트가 보낸
데이터그램이 도착할
때까지 대기
socket()
sendto()
데이터 요청
sendto()
데이터 전송
recvfrom()
close()

실행 결과

서버 쪽	클라이언트 쪽
$ ex13-01s node name : ce official name : ce.kumoh.ac.kr recv from client : A $	$ ex13-01c hostname : ce official name : ce.kumoh.ac.kr IP address : 202.31.200.87 recv from server : A $

비연결형 통신 모델에서 소켓을 통해 메시지를 수신한다.

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

int recvfrom(int sockfd, void *buf, size_t len, int flags,
        struct sockaddr *from, socklen_t *fromlen);
```

sockfd	소켓 기술자이다.
buf	수신한 메시지를 저장할 버퍼이다.
len	buf의 크기이다.
flags	recvfrom의 동작 방식을 결정한다. recv와 동일하다.
from	메시지를 송신한 쪽의 정보가 저장된다.
fromlen	from의 바이트 크기이다.
반환값	호출이 성공하면 수신한 메시지의 바이트 크기를 반환하고, 실패하면 -1을 반환한다.

recvfrom 함수

- 보통 비연결형 통신 모델에서 메시지를 수신하기 위해 사용
- 연결형 통신 모델에서도 사용할 수 있다

int sockfd

- 소켓 기술자를 의미한다.
- 보통 recvfrom이 비연결형 통신 모델에서 사용되므로 소켓이 상대 프로세스의 소켓과 연결되어 있지는 않다.

void *buf, size_t len

- 수신한 메시지를 저장하기 위한 버퍼와 버퍼의 크기이다.

int flags

- recvfrom의 동작방식을 결정한다.
  보통 0을 설정한다.

- **struct sockaddr *from**
  - NULL이 아니고 소켓이 비연결형일 경우 메시지를 전송한 측의 정보를 저장한다.
  - NULL일 경우 recvfrom은 recv와 동일하다.
- **socklen_t *fromlen**
  - from의 크기이다.
  - 메시지를 수신한 이후에는 from에 저장된 정보의 실제 길이로 변경된다
- **반환값**
  - 호출이 성공할 경우 수신한 메시지의 바이트 수를 반환한다.
  - 실패하면 -1을 반환한다.

【예제13-2】 ex 13-02.c , 서버(1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06 #define MSGSIZE 1024
07
08 main()
09 {
10      int sockfd;
11      char msg[MSGSIZE];
12
13      struct sockaddr_in server = {AF_INET, 2007, INADDR_ANY};
14
15      struct sockaddr_in client;
16     int client_len = SIZE;
```

표준입력 스트림

【예제13-2】 ex 13-02.c , 서버(2/2)


```
17     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
18
19     bind(sockfd, (struct sockaddr *)&server, SIZE);
20
21     recvfrom(sockfd, &msg, MSGSIZE, 0, (struct sockaddr *)&client,
       &client_len);
22
23     printf("recv from client : %sWn", msg);
24
25     sendto(sockfd, &msg, MSGSIZE, 0, (struct sockaddr *)&client, client_len);
26
27     close(sockfd);
28 }
```

- **비연결형 모델**
  - **소켓 생성을 SOCK_STREAM이 아닌 SOCK_DGRAM으로 하였다.**
  - **메시지 송수신을 위해 sendto, recvfrom을 사용한다.**
  - **accept 함수를 사용하지 않는다.**

비연결형 통신 모델에서 소켓을 통해 메시지를 송신한다.

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

int sendto(int sockfd, const void *msg, size_t len, int flags,
        const struct sockaddr *to, socklen_t tolen);
```

sockfd	소켓 기술자이다.
msg	전송할 메시지를 저장하고 있는 버퍼이다.
len	전송할 메시지의 길이이다.
flags	sendto의 동작 방식을 결정한다. send와 동일하다.
to	메시지를 송신할 상대의 주소 등의 정보이다.
tolen	to의 바이트 크기이다.
반환값	호출이 성공하면 전송한 메시지의 문자 개수를 반환하고, 실패하면 -1을 반환한다.

- **sendto 함수**

  비연결형 통신 모델에서 메시지를 송신할 때 사용한다.

- **int sockfd**

  소켓 기술자로 비연결형 모델에서 사용되므로 상대 프로세스의 소켓과 연결되어 있을 필요는 없다.

- **const void *msg, size_t len**

  송신할 메시지가 담긴 버퍼의 포인터와 메시지의 크기이다.

- **int flags**

  sendto의 동작 방식을 설정하는데 보통 0을 사용한다.

- **const struct sockaddr *to**

  전송한 메시지를 수신하게 되는 상대방의 정보를 저장한다.

- **socklen_t tolen**

  to의 바이트 크기이다.

【예제13-3】 ex 13-03.c (1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06 #define MSGSIZE 1024
07
08 main()
09 {
10     int sockfd;
11     char msg[MSGSIZE], recv[MSGSIZE];
12
13     struct sockaddr_in client = {AF_INET, INADDR_ANY, INADDR_ANY};
14
15     int server_len = SIZE;
16     struct sockaddr_in server = {AF_INET, 2007};
17     server.sin_addr.s_addr = inet_addr("202.31.200.87");
```

표준입력 스트림

【예제13-3】 ex 13-03.c (2/2)


```
18      sockfd = socket(AF_INET, SOCK_DGRAM, 0);
19
20      strcpy(msg, "Hello world!");
21
22      sendto(sockfd, &msg, MSGSIZE, 0,
23            (struct sockaddr *)&server, server_len);
24
25      recvfrom(sockfd, &recv, MSGSIZE, 0,
26               (struct sockaddr *)&server, &server_len);
27
28      printf("reply from server: %sWn", recv);
29
30      close(sockfd);
31  }
```

표준입력 스트림

ex13-02.c의 실행 결과	ex13-03.c의 실행 결과
$ ex13-02 recv from client : Hello world! $	$ ex13-03 reply from server: Hello world! $

[예제 프로그램 완성] 비연결형 모델, 서버쪽(1/3)


```
01 /* Server */
02 #include <sys/types.h>
03 #include <sys/socket.h>
04 #include <netinet/in.h>
05
06 #define SIZE    sizeof(struct sockaddr_in)
07
08 main()
09 {
10     int sockfd;
11     char msg, prev;
12
13     struct sockaddr_in server = {AF_INET, 2007, INADDR_ANY};
14
15     struct sockaddr_in client;
16     int client_len = SIZE;
```

표준입력 스트림

[예제 프로그램 완성] 비연결형 모델, 서버쪽(2/3)

```
17      if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
18      {
19          printf("fail to call socket()₩n");
20          exit(1);
21      }
22
23      if(bind(sockfd, (struct sockaddr *)&server, SIZE) == -1)
24      {
25          printf("fail to call bind()₩n");
26          exit(1);
27      }
28
29      prev = '₩n';
```

[예제 프로그램 완성] 비연결형 모델, 서버쪽(3/3)


```
30      while(1)
31      {
32          if(recvfrom(sockfd, &msg, 1, 0,
33                      (struct sockaddr *)&client, &client_len) == -1)
34          {
35              printf("fail to receive message₩n");
36              continue;
37          }
38
39          printf("%s%c", (prev == '₩n') ? "[recv] " : "", msg);
40          prev =  msg;
41
42          if(sendto(sockfd, &msg, 1, 0,
43                    (struct sockaddr *)&client, client_len) == -1)
44          {
45              printf("fail to receive message₩n");
46              continue;
47          }
48      }
49  }
```

[예제 프로그램 완성] 비연결형 모델, 클라이언트쪽(1/3)

```
01 /* Client */
02 #include <sys/types.h>
03 #include <sys/socket.h>
04 #include <netinet/in.h>
05
06 #define SIZE    sizeof(struct sockaddr_in)
07
08 main()
09 {
10     int sockfd;
11     char msg, prev;
12
13     struct sockaddr_in client = {AF_INET, INADDR_ANY, INADDR_ANY};
14
15     int server_len = SIZE;
16     struct sockaddr_in server = {AF_INET, 2007};
17     server.sin_addr.s_addr = inet_addr("202.31.200.87");
```

표준입력 스트림

[예제 프로그램 완성] 비연결형 모델, 클라이언트쪽(2/3)

```
18      if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
19      {
20          printf("fail to call socket()₩n");
21          exit(1);
22      }
23
24      prev = '₩n';
25
26      while(read(0, &msg, 1) != 0)
27      {
28          if(sendto(sockfd, &msg, 1, 0,
29                    (struct sockaddr *)&server, server_len) == -1)
30          {
31              printf("fail to send message₩n");
32              continue;
33          }
```

표준입력 스트림

[예제 프로그램 완성] 비연결형 모델, 클라이언트쪽(3/3)

```
34          if(recvfrom(sockfd, &msg, 1, 0,
35                      (struct sockaddr *)&server, &server_len) == -1)
36          {
37              printf("fail to receive message₩n");
38              continue;
39          }
40
41
42          printf("%s%c", (prev == '₩n') ? "[recv] " : "", msg);
43          prev = msg;
44      }
45  }
```

ex13-04s.c의 실행결과	ex13-04c.c의 실행결과
$ ex13-04s [recv]Hello world! [recv]Apple is red, banana is yellow. ^C $	$ ex13-04c Hello world! [recv] Hello world! Apple is red, banana is yellow. [recv] Apple is red, banana is yellow. ^C $

현재의 호스트 이름을 구한다

```
#include <sys/utsname.h>
int uname(struct utsname *buf);

#include <unistd.h>
int gethostname(char *name, size_t len);
```

buf	호스트의 시스템 정보가 저장된다.
name	호스트의 이름이 저장된다.
len	name의 길이이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **현재 시스템의 호스트명을 가져온다.**
  - **호스트명은 보통 알파벳과 숫자의 조합으로 이루어져 있다.**

struct utsname 형의 구조체

```
struct utsname {
    char nodename[];        /* 호스트 이름       */
    char sysname[];         /* 운영체제 이름     */
    char version[];         /* 운영체제 버전     */
    char machine[];         /* 머신(CPU)의 종류 */
    char release[];         /* 릴리즈 정보       */
};
```

- **nodename**
  - 현재의 컴퓨터가 네트워크 상에서 가지게 되는 이름
  - 호스트의 이름만 저장되는게 일반적

    ↔ FQDN일 수도 있다.

【예제13-5】 ex 13-05.c


```
01 #include <sys/utsname.h>
02 #include <unistd.h>
03
04 main()
05 {
06         struct utsname info;
07         char myname[1024];
08
09         uname(&info);
10         printf("sysname: %s₩n", info.sysname);
11         printf("nodename: %s₩n", info.nodename);
12         printf("release: %s₩n", info.release);
13         printf("version: %s₩n", info.version);
14         printf("machine: %s₩n", info.machine);
15
16         gethostname(myname, 1024);
17         printf("hostname: %s₩n", myname);
18 }
```

```
$ ex13-05
sysname: Linux
nodename: ce
release: 2.4.18-3smp
version: #1 SMP Thu Apr 18 07:27:31 EDT 2002
machine: i686
hostname: ce
$
```

호스트 이름으로 주소와 관련된 정보를 알아온다

```
#include <netdb.h>

struct hostent *gethostbyname(const char *name);
```

name	호스트의 이름으로 도메인 네임 주소까지 포함한다.
반환값	호출이 성공하면 struct hostent형의 포인터를 반환하고, 실패하면 NULL을 반환한다.

- **const char *name**
  - **호스트의 이름**

    ce와 같이 호스트 이름만일 수 있고

    ce.kumoh.ac.kr과 같이 FQDN일 수 있다.

struct hostent 형의 구조체

```
struct hostent {
    char    *h_name;        /* 정규(공식) 이름        */
    char    **h_aliases;    /* 보조 이름(가명) 리스트 */
    int     h_addrtype;     /* 주소 형태              */
    int     h_length;       /* 주소의 길이            */
    char    **h_addr_list;  /* list of addresses      */
}
```

- **h_name**
  - 호스트의 정규 이름
- **h_aliases**
  - 정규 이름이 아닌 또 다른 이름들로 별명이라고 생각하면 된다.
  - 서버로서 제공하는 서비스에 따라 다양한 이름을 가질 수 있다.

- h_addrtype
  - 호스트의 주소 형태이다.
    IPv4일 경우 AF_INET
    IPv6일 경우 AF_INET6

- h_length
  - 호스트 주소의 바이트 크기이다.
    IPv4일 경우 4바이트
    IPv6일 경우 6바이트

- h_addr_list
  - 주소의 리스트이다.

【예제13-6】 ex 13-06.c (1/2)


```
01 #include <unistd.h>
02 #include <netdb.h>
03
04 main(int argc, char *argv[])
05 {
06     struct hostent *hent;
07     char **ptr;
08
09     if(argc < 2) {
10         printf("%s hostname₩n", argv[0]);
11        exit(1);
12     }
13
14     if((hent = gethostbyname(argv[1])) == NULL) {
15         printf("fail to call gethostbyname()₩n");
16         exit(1);
17    }
```

표준입력 스트림

【예제13-6】 ex 13-06.c (2/2)


```
18      printf("official name : %s₩n", hent->h_name);
19
20      for(ptr = hent->h_aliases; *ptr != NULL; *ptr++)
21          printf("₩talias : %s₩n", *ptr);
22
23      if(hent->h_addrtype == AF_INET) {
24          ptr = hent->h_addr_list;
25          for(; *ptr != NULL; ptr++)
26              printf("₩taddress : %s₩n",
27                                  inet_ntoa(*((struct in_addr
                                    *)*ptr)));
28      }
29  }
```

실행 결과

```
$ ex13-06 www.daum.net
official name : daumtop.daum.akadns.net
        alias : www.daum.net
        address : 222.231.51.78
        address : 211.115.115.211
        address : 211.115.115.212
        address : 222.231.51.77
$ ex13-06 ce
official name : ce.kumoh.ac.kr
        alias : ce
        address : 202.31.200.87
$ ex13-06 www.google.com
official name : www.l.google.com
        alias : www.google.com
        address : 66.102.7.147
        address : 66.102.7.99
        address : 66.102.7.104
$
```

주어진 바이너리 IP주소에 해당하는 호스트의 이름을 구한다

```
#include <netdb.h>

struct hostent *gethostbyaddr(const char *addr, int len, int type);
```

addr	바이너리 형태의 IP주소이다.
len	addr의 바이트 크기이다.
type	프로토콜 군을 지정한다.
반환값	호출이 성공하면 struct hostent형의 포인터를 반환하고, 실패하면 NULL을 반환한다.

const char *addr

- **바이너리 형태의 IP주소이다.**
- **inet_addr 함수가 반환하는 in_addr_t 형의 값에 대한 포인터**

int len

- addr의 바이트 크기이다.
- IPv4일 경우 4, IPv6일 경우 60이 된다

int type

- AF_INET 또는 AF_INET60이다.

【예제13-7】 ex 13-07.c (1/2)


```
01 #include <unistd.h>
02 #include <netdb.h>
03
04 main(int argc, char *argv[])
05 {
06     struct hostent *hent;
07     in_addr_t ipaddr;
08     char **ptr;
09
10     if(argc < 2) {
11         printf("%s ip_address₩n", argv[0]);
12         exit(1);
13     }
14
15     if((ipaddr = inet_addr(argv[1])) == -1) {
16         printf("fail to call inet_addr()₩n");
17         exit(1);
18     }
```

표준입력 스트림

【예제13-7】 ex 13-07.c (2/2)


```
19      if((hent = gethostbyaddr((char *)&ipaddr, 4, AF_INET)) == NULL) {
20          printf("fail to call gethostbyaddr()₩n");
21          exit(1);
22      }
23
24      printf("official name : %s₩n", hent->h_name);
25
26      for(ptr = hent->h_aliases; *ptr != NULL; *ptr++)
27          printf("₩talias : %s₩n", *ptr);
28
29      if(hent->h_addrtype == AF_INET) {
30          ptr = hent->h_addr_list;
31          for(; *ptr != NULL; ptr++)
32              printf("₩taddress : %s₩n",
33                                  inet_ntoa(*((struct in_addr *)*ptr)));
34      }
35  }
```

실행 결과

```
$ ex13-07 202.31.200.87
official name : ce.kumoh.ac.kr
        alias : ce
        address : 202.31.200.87
$ ex13-07 66.102.7.147
official name : mc-in-f147.google.com
        address : 66.102.7.147
$
```