

- 네트워크 관련 명령어

- telnet

- 원격 컴퓨터에 로그인하고 명령을 실행할 수 있는 텍스트 기반의 네트워크 프로토콜
- 기본적인 tcp 연결 생성용으로 사용

- nc

- 다양한 네트워크 작업을 수행할 수 있는 커맨드라인 도구
- TCP 및 UDP 프로토콜을 사용하여 소켓을 열고 데이터를 송수신할 수 있으며, 클라이언트와 서버 간의 연결을 설정할 수 있음.

- ifconfig

- 네트워크 인터페이스의 설정을 확인하고 변경하는 명령어

- ping
  - 네트워크 연결 상태를 확인
- traceroute :
  - 목적지까지 패킷이 어떤 경로를 통해 전송되는지 추적
- netstat
  - 네트워크 연결 상태 및 활성화된 포트를 확인
- dig
  - DNS(Domain Name System) 조회를 수행

- ssh
  - 원격 서버에 안전하게 로그인하고, 컴퓨터 간에 파일을 전송
- scp
  - ssh 프로토콜을 사용하여 로컬과 원격 시스템 간에 파일을 복사
- curl
  - 웹 페이지나 파일을 다운로드
- tcpdump
  - 네트워크 패킷을 캡처하고 분석

- nc

- 네트워크 연결과 입출력을 다루는 유틸리티 프로그램
- TCP/UDP 네트워크 연결
  - nc [options] [host] [port]
  - nc 192.168.0.10 80
  - UDP 연결
    - nc -u [options] [host] [port]
- 파일 전송
  - 수신 측에서 실행할 명령어: nc -l [port] > [filename]
  - 송신 측에서 실행할 명령어: nc [host] [port] < [filename]

- nc

- 포트 스캔

- nc -zv [host] [port-range]
    - -zv 옵션: 포트 스캔 모드를 설정
    - -z 옵션: just scan for listening daemons, without sending any data to them
    - -v 옵션: 상세 정보를 출력
    - [host]: 호스트 이름이나 IP 주소
    - [port-range]: 스캔할 포트 범위
    - 예) nc -zv www.programmers.com 80-90

- ifconfig

- ifconfig : 현재 시스템에 설정된 네트워크 인터페이스의 정보를 출력
- ifconfig [인터페이스명] : 특정 네트워크 인터페이스의 정보만 출력
- ifconfig [인터페이스명] [IP주소] [넷마스크] : 특정 네트워크 인터페이스의 IP주소와 넷마스크를 설정
- ifconfig [인터페이스명] up/down : 특정 네트워크 인터페이스를 활성화/비활성화

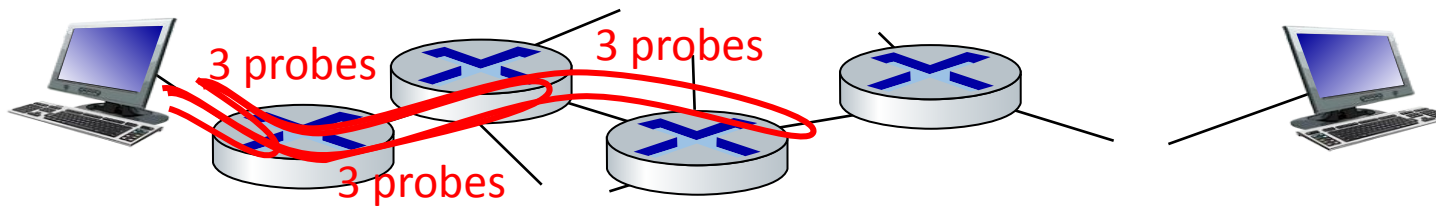
- ping

- ping [호스트명/IP주소] : 해당 호스트나 IP주소로 ICMP 패킷을 보내서 도달 여부를 확인
- ping [호스트명/IP주소] -c [패킷수] : 지정된 패킷 수만큼 ICMP 패킷을 보내고 응답을 기다림
- ping [호스트명/IP주소] -i [인터벌] : ICMP 패킷을 보내는 간격을 지정
- ping [호스트명/IP주소] -s [패킷크기] : ICMP 패킷의 크기를 지정



- traceroute

- traceroute [호스트명/IP주소] : 해당 호스트나 IP주소로 ICMP 패킷을 보내서 경로를 추적
- traceroute [호스트명/IP주소] -m [최대홉수] : 최대 홉 수를 지정
- traceroute [호스트명/IP주소] -p [포트번호] : 목적지 호스트에 도달하는 데 사용될 포트를 지정



- netstat

- netstat : 현재 활성화된 네트워크 연결 정보를 출력
- netstat -a : 모든 네트워크 연결 정보를 출력
- netstat -t : TCP 연결 정보만 출력
- netstat -u : UDP 연결 정보만 출력

- dig

- dig [도메인명] : 지정된 도메인의 DNS 정보를 조회
- dig [도메인명] +short : 짧은 형태로 조회 결과를 출력
- dig [도메인명] +trace : 조회한 DNS 정보의 경로를 추적

- ssh

- ssh [호스트명/IP주소] : 해당 호스트나 IP주소로 SSH 접속
- ssh [호스트명/IP주소] -p [포트번호] : SSH 접속 시 사용할 포트를 지정
- ssh [호스트명/IP주소] -i [사용자명] : SSH 접속 시 사용할 사용자명을 지정

- scp
  - SSH 프로토콜을 이용하여 로컬 시스템과 원격 시스템 사이에서 파일을 복사
  - scp [옵션] [원본 파일] [대상 경로]
    - -r: 디렉토리를 복사
    - -P: SSH 포트를 지정
    - -v: 디버그 모드로 실행

- curl

- URL을 이용하여 데이터를 전송하거나 받는 명령어
- 주로 HTTP, HTTPS, FTP 등의 프로토콜을 사용하여 데이터를 전송
- curl [옵션] [URL]
  - -i: 헤더 정보를 출력
  - -X: HTTP 메소드를 지정
  - -H: HTTP 요청 헤더를 지정
  - -d: POST 요청 시 전송할 데이터를 지정
  - -o: 출력 파일을 지정
- curl https://www.example.com

- curl

- REST API 호출: API 엔드포인트 URL과 HTTP 메소드, 요청 헤더, 전송할 데이터 등을 지정
- API:
  - POST `https://example.com/api/users`
- curl 명령
  - `curl -X POST -H "Content-Type: application/json" -d '{"username": "sanghwan", "password": "11111111"}' https://example.com/api/users`

- tcpdump

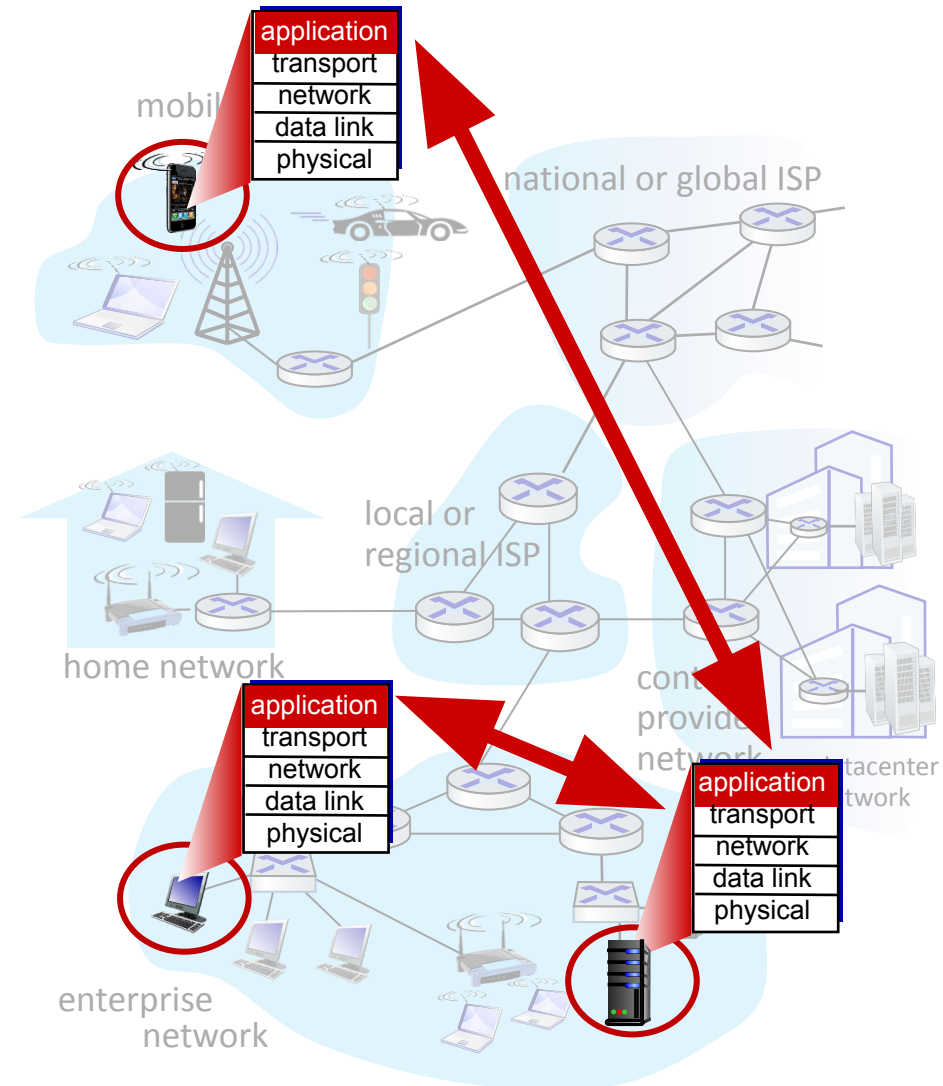
- 네트워크 트래픽을 캡처하고 분석하기 위한 명령어
- 패킷 분석 및 디버깅, 보안 감사 등에 사용
- tcpdump [옵션] [식별자]
  - -i: 캡처할 인터페이스를 지정
  - -n: IP 주소 및 포트 번호를 숫자 형태로 출력
  - -s: 캡처할 패킷의 최대 크기를 지정
  - -v: 패킷을 자세하게 출력
  - -c: 캡처할 패킷의 수를 지정
  - -w: 캡처한 패킷을 파일에 저장



- tcpdump

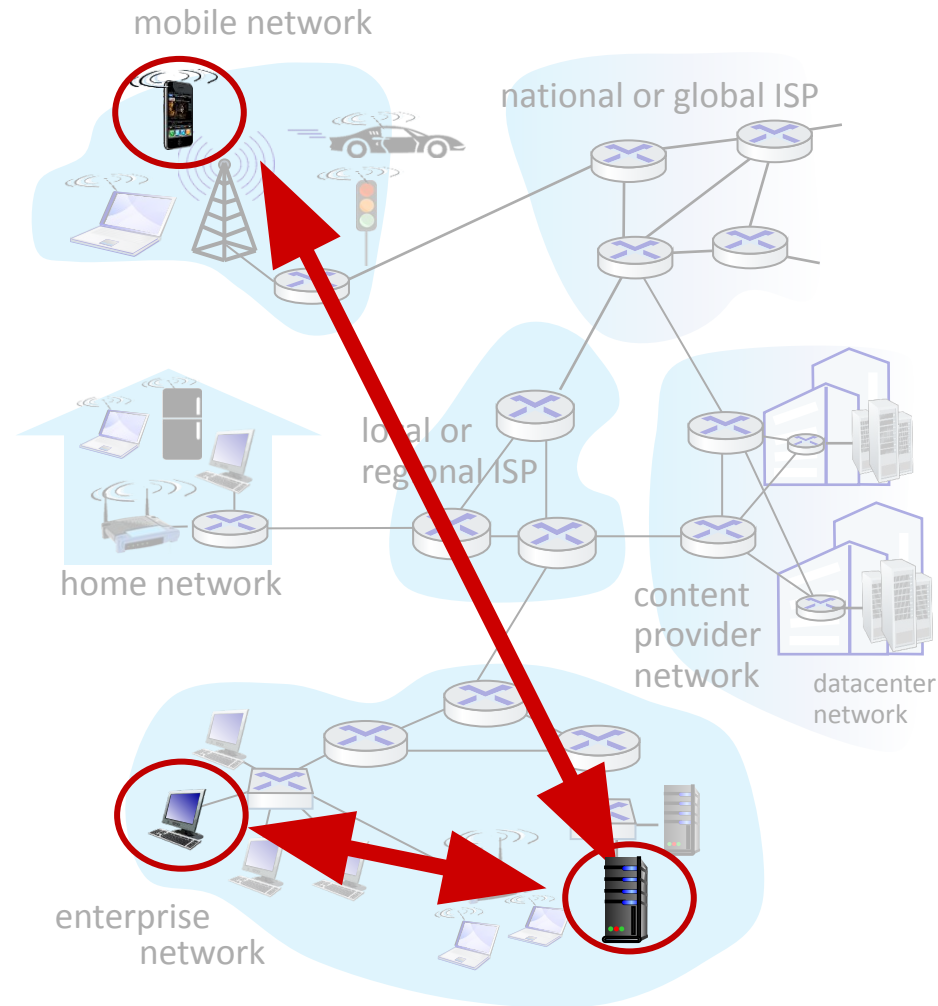
- 특정 IP 주소에서 전송되는 모든 패킷을 캡처
  - tcpdump host <ip-address>
- 특정 포트에서 발생하는 패킷을 캡처
  - tcpdump port <port-number>
- 특정 프로토콜을 사용하는 패킷을 캡처
  - tcpdump <protocol>

- 소켓 프로그래밍 이론 및 실습 1
  - 네트워크 응용 개요

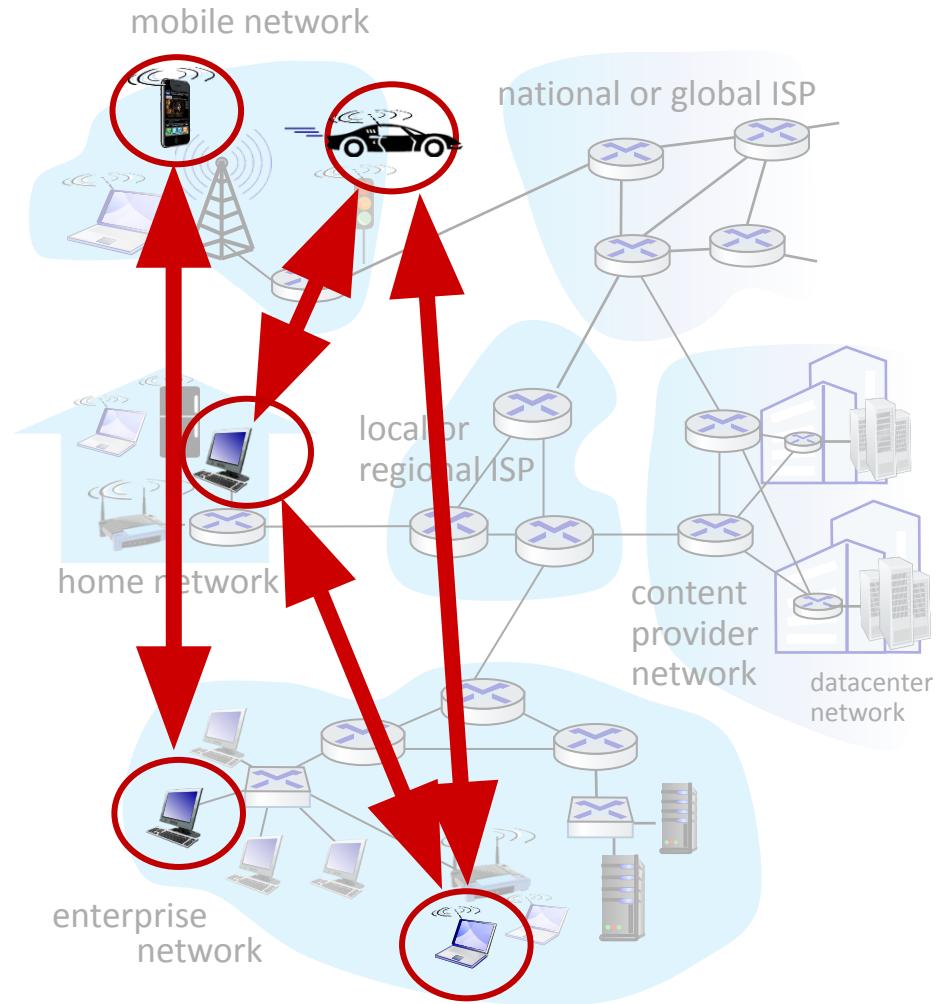


# 클라이언트-서버 (Client-server) 방식

20



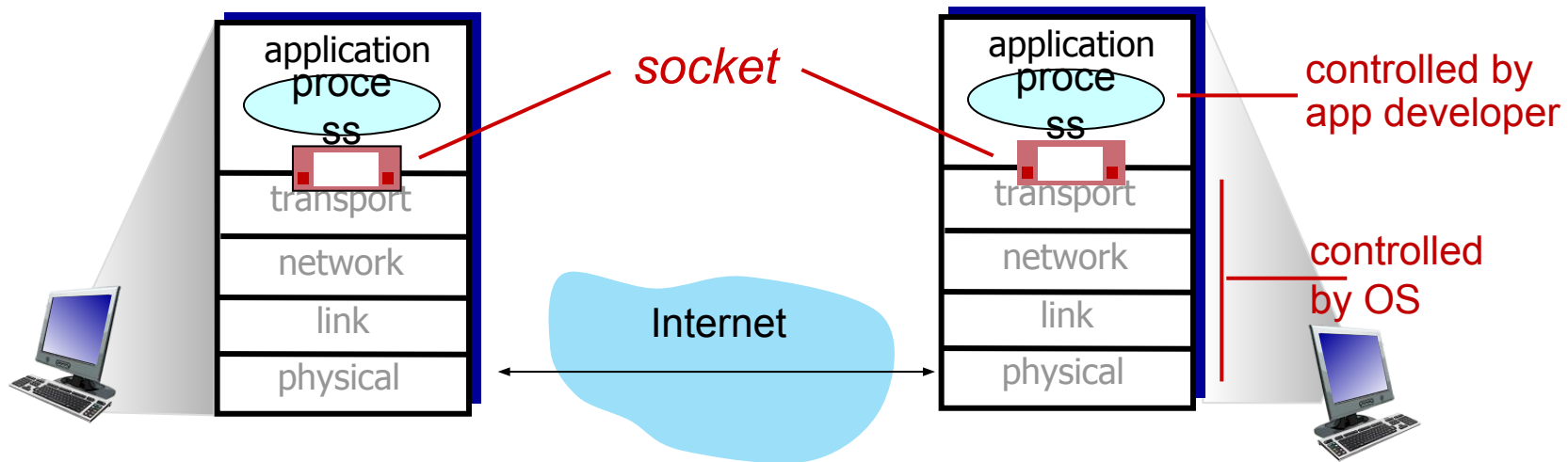
# P2P 방식 (Peer-peer)



- 컴퓨터들 간의 통신을 가능하게 하는 엔드포인트
- 소켓을 사용하여 서버와 클라이언트 간에 데이터를 전송
- 스트림 소켓 (Stream Socket):
  - 전송 제어 프로토콜(TCP)을 기반으로 하는 소켓
  - 연결 지향적이며 신뢰성 있는 데이터 전송
  - 웹 브라우징, 이메일 전송 등의 애플리케이션에서 주로 사용
- 데이터그램 소켓 (Datagram Socket):
  - 사용자 데이터그램 프로토콜(UDP)을 기반으로 하는 소켓
  - 비연결 지향적이며 신뢰성이 떨어지지만 속도가 빠름
  - 데이터 전송 순서를 보장하지 않음
  - 음성 및 동영상 스트리밍, 온라인 게임 등의 애플리케이션에서 주로 사용

# 소켓

23



# 전송계층 프로토콜

24

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP



- 소켓 프로그래밍 이론 및 실습 1
  - TCP 서버 (C언어)

# Sample Program – TCP Server server.c

26

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

헤더 파일

# Sample Program – TCP Server server.c

27

```
int main(int argc, char *argv[]) {
    struct sockaddr_in server, remote;
    int request_sock, new_sock;
    int bytesread, addrlen;
    int i;
    char buf[BUFSIZ];
    if (argc != 2) {
        (void) fprintf(stderr, "usage: %s port\n", argv[0]);
        exit(1);
    }
    if ((request_sock = socket(AF_INET, SOCK_STREAM,
                             IPPROTO_TCP)) < 0) {
        perror("socket");
        exit(1);
    }
```

Creating socket descriptor  
for TCP

# Sample Program – TCP Server server.c

28

```
memset((void *) &server, 0, sizeof (server));
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons((u_short)atoi(argv[1]));
if (bind(request_sock, (struct sockaddr *)&server, sizeof (server)) < 0) {
    perror("bind");
    exit(1);
}
if (listen(request_sock, SOMAXCONN) < 0) {
    perror("listen");
    exit(1);
}
```

Bind socket and  
address

# Sample Program – TCP Server server.c

29

```
for (;;) {  
    addrlen = sizeof(remote);  
    new_sock = accept(request_sock,  
                      (struct sockaddr *)&remote, &addrlen);  
    if (new_sock < 0) {  
        perror("accept");  
        exit(1);  
    }  
    printf("connection from host %s, port %d, socket %d\n",  
          inet_ntoa(remote.sin_addr),  
          ntohs(remote.sin_port), new_sock);  
}
```

Accept new  
connections

# Sample Program – TCP Server server.c

30

```
for (;;) {
```

```
    bytesread = read(new_sock, buf, sizeof(buf) - 1);
```

```
    if (bytesread <= 0) {
```

```
        close(new_sock);
```

```
        break;
```

```
    }
```

```
    buf[bytesread] = '\0';
```

```
    printf("%s: %d bytes from %d: %s\n", argv[0], bytesread, new_sock, buf);
```

```
    for(i = 0; i < bytesread; i++)
```

```
        buf[i] = toupper(buf[i]);
```

```
    if (write(new_sock, buf, bytesread) != bytesread)
```

```
        perror("echo");
```

```
    }
```

```
}
```

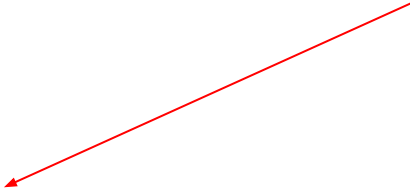
```
}
```

Read data from  
connected socket

Write data to  
connected socket

- `gcc -o srv server.c`
- 클라이언트는 nc나 telnet 사용

```
struct sockaddr
{
    u_short sa_family;
    __SOCKADDR_COMMON (sa_); /* Common data: address family
and length. */
    char sa_data[14];        /* Address data. */
};
```





# 소켓 관련 구조체

```
#include <netinet/in.h>
```

```
struct sockaddr_in {  
    short    sin_family; // e.g. AF_INET  
    unsigned short sin_port; // e.g. htons(3490)  
    struct in_addr sin_addr; // see struct in_addr, below  
    char      sin_zero[8]; // zero this if you want to  
};
```

```
struct in_addr {  
    unsigned long s_addr; // load with inet_aton()  
};
```

- socket - create an endpoint for communication

```
#include <sys/types.h>      /* See NOTES */
```

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

```
int socket(int domain, int type, int protocol);
```

- getsockopt, setsockopt - get and set options on sockets

```
#include <sys/types.h>      /* See NOTES */
```

```
#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);
```

- bind - bind a name to a socket

```
#include <sys/types.h>      /* See NOTES */  
#include <sys/socket.h>
```

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

- listen - listen for connections on a socket

```
#include <sys/types.h>      /* See NOTES */  
#include <sys/socket.h>
```

```
int listen(int sockfd, int backlog);
```

- accept, accept4 - accept a connection on a socket

```
#include <sys/types.h>      /* See NOTES */  
#include <sys/socket.h>
```

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

```
#define _GNU_SOURCE          /* See feature_test_macros(7) */  
#include <sys/socket.h>
```

```
int accept4(int sockfd, struct sockaddr *addr, socklen_t *addrlen, int flags);
```

- `inet_aton`, `inet_addr`, `inet_network`, `inet_ntoa`, `inet_makeaddr`, `inet_lnaof`, `inet_netof` - Internet address manipulation routines

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

```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

```
int inet_aton(const char *cp, struct in_addr *inp);
```

- `inet_pton` - convert IPv4 and IPv6 addresses from text to binary form

```
#include <arpa/inet.h>
```

```
int inet_pton(int af, const char *src, void *dst);
```



- htonl, htons, ntohl, ntohs - convert values between host and network byte order

```
#include <arpa/inet.h>
```

```
uint32_t htonl(uint32_t hostlong);
```

```
uint16_t htons(uint16_t hostshort);
```

```
uint32_t ntohl(uint32_t netlong);
```

```
uint16_t ntohs(uint16_t netshort);
```

- gethostbyname, gethostbyaddr, sethostent, gethostent, endhostent, h\_errno, perror, hstrerror, gethostbyaddr\_r, gethostbyname2, gethostbyname2\_r, gethostbyname\_r, gethostent\_r - get network host entry

```
#include <netdb.h>
```

```
extern int h_errno;
```

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

- 소켓 프로그래밍 이론 및 실습 1
  - TCP 클라이언트 (C 언어)

# Sample Program – TCP Client client.c

44

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

# Sample Program – TCP Client client.c

45

```
int main(int argc, char *argv[])
{
    struct hostent *hostp;
    struct sockaddr_in server;
    int sock;
    char buf[BUFSIZ];
    int bytesread;
    if(argc != 3) {
        (void) fprintf(stderr, "usage: %s host port\n", argv[0]);
        exit(1);
    }
    if ((sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
        perror("socket");
        exit(1);
    }
}
```

Creating Socket  
Descriptor for TCP

# Sample Program – TCP Client client.c

46

```
if ((hostp = gethostbyname(argv[1])) == 0) {  
    fprintf(stderr, "%s: unknown host\n", argv[2]);  
    exit(1);  
}
```

Get Host IP Addr

```
memset((void *) &server, 0, sizeof (server));  
server.sin_family = AF_INET;  
memcpy((void *) &server.sin_addr, hostp->h_addr, hostp->h_length);  
server.sin_port = htons((u_short)atoi(argv[2]));
```

```
if (connect(sock, (struct sockaddr *)&server, sizeof (server)) < 0) {  
    (void) close(sock);  
    fprintf(stderr, "connect");  
    exit(1);  
}
```

Connect to Server

# Sample Program – TCP Client client.c

47

```
for (;;) {  
    /* data from keyboard */  
    if (!fgets(buf, sizeof buf, stdin)) {  
        exit(0);  
    }  
    if (write(sock, buf, strlen(buf)) < 0) {  
        perror("write");  
        exit(1);  
    }  
    bytesread = read(sock, buf, sizeof buf);  
    buf[bytesread] = '\0';  
    printf("%s: got %d bytes: %s\n", argv[0], bytesread, buf);  
}  
}
```

Input from Keyboard

Write to SD.  
Send the bytestream  
to the server

Read byte stream  
from the server

- connect - initiate a connection on a socket

```
#include <sys/types.h>      /* See NOTES */
```

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

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



# TCP Client 테스트

- `nc -l 10000`
  - 서버 대응
  - Waiting for **tcp** port 10000
- `gcc -o cli client.c`

# Sample Program – TCP

- 서버와 클라이언트 모두 구현한 프로그램 사용
- `gcc -o srv server.c`
- `gcc -o cli client.c`

- 소켓 프로그래밍 이론 및 실습 1
  - UDP 서버 (C 언어)

# Sample Program – UDP Server udpserver.c

52

```
int main(int argc, char *argv[])
{
    int sockid, nread, addrlen;
    struct sockaddr_in my_addr, client_addr;
    char msg[50];
    if(argc != 2) {
        printf("%s myportid\n", argv[0]);
        return 0;
    }
    printf("Server: creating socket\n");
    if ( (sockid = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        { printf("Server: socket error: %d\n",errno); exit(0); }
```



Creating UDP  
socket

# Sample Program – UDP Server udpserver.c

53

```
printf("Server: binding my local socket\n");
```

```
memset((char *) &my_addr, 0, sizeof(my_addr));
```

```
my_addr.sin_family = AF_INET;
```

```
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
my_addr.sin_port = htons(atoi(argv[1]));
```

Bind UDP socket and address

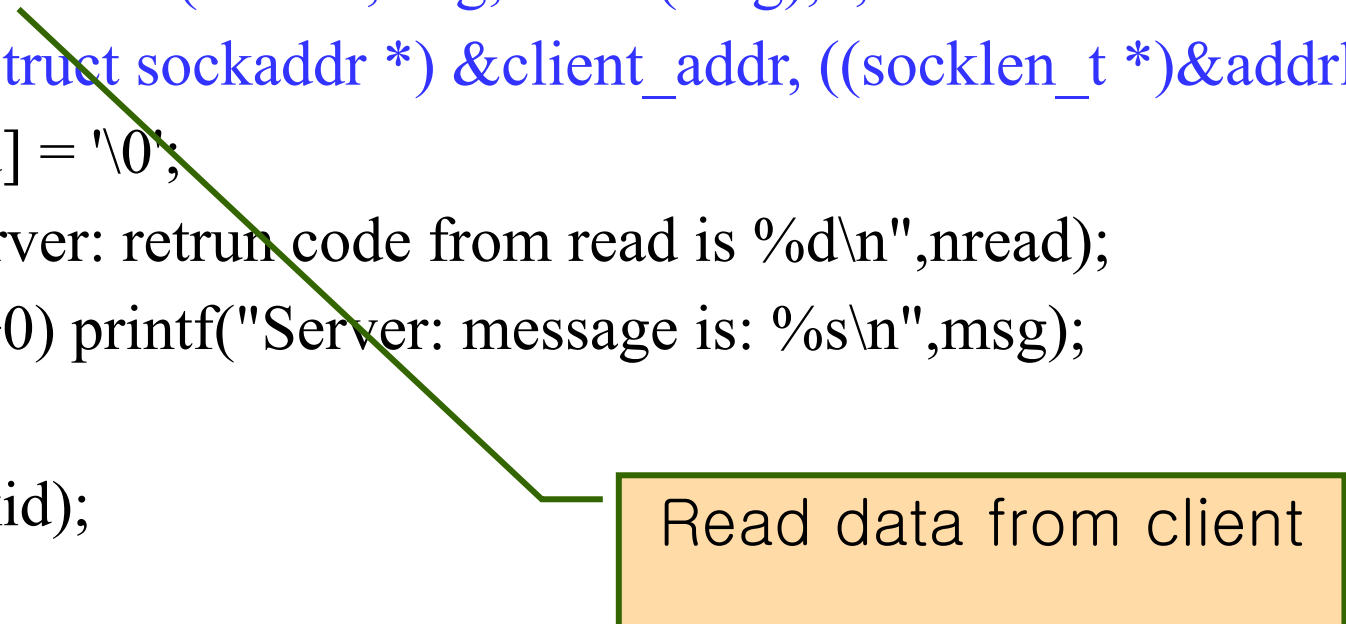
```
if ( (bind(sockid, (struct sockaddr *) &my_addr,  
        sizeof(my_addr)) < 0) )
```

```
{ printf("Server: bind fail: %d\n",errno); exit(0); }
```

# Sample Program – UDP Server udpserver.c

54

```
printf("Server: starting blocking message read\n");  
nread = recvfrom(sockid,msg,sizeof(msg),0,  
                (struct sockaddr *) &client_addr, ((socklen_t *)&addrlen);  
msg[nread] = '\0';  
printf("Server: return code from read is %d\n",nread);  
if (nread >0) printf("Server: message is: %s\n",msg);  
  
close(sockid);  
}
```



Read data from client

- recv, recvfrom, recvmsg - receive a message from a socket

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

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

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

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

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

# UDP 서버 테스트

- Using nc (netcat) utility
  - `nc -u localhost 10000`
- `gcc -o udpsrc udpserver.c`



- 소켓 프로그래밍 이론 및 실습 1
  - UDP 클라이언트 (C 언어)

# Sample Program – UDP Client udpcli.c

58

```
int main(int argc, char *argv[])
{
    int sockid, retcode;
    struct sockaddr_in my_addr, server_addr;
    char msg[128];
    if(argc != 4) {
        printf("%s myport serveraddr serverport\n", argv[0]);
        return 0;
    }
    printf("Client: creating socket\n");
    if ( (sockid = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    { printf("Client: socket failed: %d\n",errno); exit(0); }
```



Creating UDP  
socket

# Sample Program – UDP Client udpcli.c

59

```
printf("Client: binding my local socket\n");
```

```
memset((char *) &my_addr, 0, sizeof(my_addr));
```

```
my_addr.sin_family = AF_INET;
```

```
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
my_addr.sin_port = htons(atoi(argv[1]));
```

Bind UDP socket and  
address

```
if ( (bind(sockid, (struct sockaddr *) &my_addr,  
        sizeof(my_addr)) < 0) )
```

```
{ printf("Client: bind fail: %d\n",errno); exit(0); }
```

# Sample Program – UDP Client udpcli.c

60

```
printf("Client: creating addr structure for server\n");  
bzero((char *) &server_addr, sizeof(server_addr));  
server_addr.sin_family = AF_INET;  
server_addr.sin_addr.s_addr = inet_addr(argv[2]);  
server_addr.sin_port = htons(atoi(argv[3]));
```

Send message to  
server

```
printf("Client: initializing message and sending\n");  
sprintf(msg, "Hello world");  
retcode = sendto(sockid,msg,strlen(msg),0,(struct sockaddr *) &server_addr,  
sizeof(server_addr));
```

- send, sendto, sendmsg - send a message on a socket

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

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

```
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
```

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

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

# UDP Client 테스트

- `nc -l 10000 -u`
- `gcc -o udpcli udpcli.c`

# UDP Client 테스트

- `gcc -o udpsrv udpserver.c`
- `gcc -o udpcli udpcli.c`

- 소켓 프로그래밍 이론 및 실습 1
  - UDP 서버/클라이언트 개선 (C 언어)



- UDP Server가 메시지를 받고 끝나는 것이 아니라, 추가적인 작업을 진행하도록 함.
  - 받은 메시지를 대문자로 변경하여 클라이언트에 전송
  - 클라이언트의 정보 출력
  - 루프를 돌면서 계속 메시지 처리

**소스코드 수정 실습!**

- 소켓 프로그래밍 이론 및 실습 1
  - 소켓 I/O Multiplexing

- `select()`
  - I/O 멀티플렉싱 시스템 콜, 여러 개의 파일 디스크립터를 감시
  - 어떤 디스크립터가 읽거나 쓰기 가능한 상태인지 확인하여 해당 디스크립터를 선택
  - 이를 통해 단일 프로세스가 여러 개의 I/O 요청을 동시에 처리
  - `select()` 함수가 호출되면, 블로킹 되어있다가 어떤 파일 디스크립터가 선택되면 반환
  - `select()` 함수가 반환한 뒤, 선택된 파일 디스크립터에 대해 처리를 수행

# Multiplexing File Descriptors

- select, pselect, FD\_CLR, FD\_ISSET, FD\_SET, FD\_ZERO - synchronous I/O multiplexing
- ```
/* According to POSIX.1-2001, POSIX.1-2008 */
```
- ```
#include <sys/select.h>
```
- ```
/* According to earlier standards */
```
- ```
#include <sys/time.h>
```
- ```
#include <sys/types.h>
```
- ```
#include <unistd.h>
```
- ```
int select(int nfds, fd_set *readfds, fd_set *writefds,
```
- ```
          fd_set *exceptfds, struct timeval *timeout);
```

# Multiplexing File Descriptors

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

```
int select(int numdes,
```

```
    fd_set *readmask,
```

```
    fd_set *writemask,
```

```
    fd_set *exceptmask,
```

```
    struct timeval *timeout);
```

FD\_SETSIZE (1024)

poll(2) 사용

**numdes** : 확인할 파일 디스크립터의 범위를 지정. 확인할 파일 디스크립터의 값 중 가장 큰 값 + 1

**readmask** : 읽기 가능한 상태를 확인할 파일 디스크립터의 집합

writemask : 쓰기 가능한 상태를 확인할 파일 디스크립터의 집합

exceptmask : 예외 상태를 확인할 파일 디스크립터의 집합

**timeout** : select 함수가 블록되는 시간을 설정. NULL이면 무한 대기

# Multiplexing File Descriptors

```
int n;  
fd_set mask;  
FD_SET(n, &mask) /* set bit n */  
FD_CLR(n, &mask) /* clear bit n */  
result = FD_ISSET(n, &mask) /* test bit n */  
FD_ZERO(&mask) /* clear all bits */
```

예) **FD\_SET**(**fileno**(**stdin**), &readmask);

# Multiplexing File Descriptors

```
fd_set rmask, mask;
FD_ZERO(&mask);
FD_SET(sock, &mask);
FD_SET(fileno(stdin), &mask);

for ( ; ; ) {
    rmask = mask;
    nfound = select(FD_SETSIZE, &rmask, (fd_set *)0, (fd_set *)0,
                    &timeout);
    if (FD_ISSET(fileno(stdin), &rmask)) {
        fgets(buf, sizeof buf, stdin);
    }
    if (FD_ISSET(sock, &rmask)) {
        ...
    }
    ...
}
```

# Example : Multiplexing File Descriptors

72

- serverorg.c
- clientorg.c

코드 설명



# Example : Multiplexing File Descriptors

- 동작 테스트
  - 서버 1개, 클라이언트 2개 실행
- `gcc -o srv serverorg.c`
- `gcc -o cli clientorg.c`
  
- `./src 10000`
- `./cli 10000 localhost`
- `./cli 10000 localhost`

- 소켓 프로그래밍 이론 및 실습 1
  - 소켓 관련 함수

- `int inet_aton(const char *cp, struct in_addr *inp);`
- `in_addr_t inet_addr(const char *cp);`
- `in_addr_t inet_network(const char *cp);`
- `char *inet_ntoa(struct in_addr in);`
- `struct in_addr inet_makeaddr(in_addr_t net, in_addr_t host);`
- `in_addr_t inet_lnaof(struct in_addr in);`
- `in_addr_t inet_netof(struct in_addr in);`
- `int inet_pton(int af, const char *src, void *dst);`

# 소켓 옵션 관련 함수 예제

- `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);`

- optname 값이 socket level인지 특정 protocol에 대한 설정인지를 지정하는 값
  - SOL\_SOCKET : socket API level
  - IPPROTO\_IP : IP protocol level
  - IPPROTO\_TCP : TCP protocol level

# SOL\_SOCKET level의 상수

- SO\_ACCEPTCONN : accept된 connection 여부 조회 (get only) : 1이면 accept(2)된 connection.
- SO\_BROADCAST : datagram socket에 broadcast flag값을 (set/get)
- SO\_DOMAIN : socket에 설정된 domain값 (ex. AF\_INET, AF\_UNIX 등...)을 얻는다. (get only)
- SO\_ERROR : socket error를 읽고 지움. (get only)
- SO\_DONTROUTE : gateway를 통해서 전송을 금지하고 직접 연결된 host끼리만 전달하게 함. (set/get)
- SO\_KEEPALIVE : cconnection-oriented socket에서 keep alive message를 전송할 수 있도록 함. (set/get)
- SO\_LINGER : linger option 설정 (set/get)
  - struct linger {
  - int l\_onoff; /\* linger active \*/
  - int l\_linger; /\* how many seconds to linger for \*/
  - };
  - l\_onoff를 1로 설정하면, close(2), shutdown(2) 함수를 실행하면 미전송된 데이터를 정상적으로 전송하거나
  - linger timeout이 도래되면 함수를 return함. 그렇지 않으면 바로 return되고 background로 작업하게 됨.

# SOL\_SOCKET level의 상수

- SO\_OOBINLINE : out of bound data를 직접 읽을 수 있게 set/get (주로 X.25에서 사용)
- SO\_PROTOCOL : socket에 설정된 protocol을 읽음. (/etc/protocols)
- SO\_RCVBUF : socket에서 읽을 수 있는 최대 buffer의 크기를 set/get함
- SO\_REUSEADDR : bind(2) 시에 local 주소를 재사용할 것인지 여부를 set/get함
- SO\_SNDBUF : socket에서 write할 수 있는 최대 buffer의 크기를 set/get함
- SO\_TYPE : 설정된 socket의 type(ex. SOCK\_STREAM, SOCK\_DGRAM)을 get함

- 소켓 프로그래밍 이론 및 실습 1
  - TCP 클라이언트 (파이썬)



# Python TCP Client

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```

- nc를 활용한 테스트
  - nc -l 12000

- 소켓 프로그래밍 이론 및 실습 1
  - TCP 서버 (파이썬)

# Python TCP Server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    connectionSocket.close()
```

- nc를 활용한 테스트
  - nc localhost 12000

- 소켓 프로그래밍 이론 및 실습 1
  - UDP 클라이언트 (파이썬)

# Python UDP Client

87

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message.encode(), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage.decode()
clientSocket.close()
```

# Python UDP Client

- nc를 활용한 테스트
  - nc -l 12000 -u



- 소켓 프로그래밍 이론 및 실습 1
  - UDP 서버 (파이썬)

# Python UDP Server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("", serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

# Python UDP Server

- nc를 활용한 테스트
  - nc -u localhost 12000

- [http://www-net.cs.umass.edu/ntu\\_socket/](http://www-net.cs.umass.edu/ntu_socket/)
  - Socket Programming in C, select(), etc
- <http://java.sun.com/docs/books/tutorial/essential/concurrency/>
  - Java Thread
- Sample code
  - client.c, server.c
    - TCP Example
  - udpcli.c, udpserver.c
    - UDP Example
  - clientorg.c serverorg.c
    - Multiple connections with select()
  - SimpleThreads.java
    - Java Thread Example
  - TCPClientTh.java TCPServerTh.java
    - TCP Client, Server Example with Java thread

- 소켓 프로그래밍 이론 및 실습 1
  - 예제 0: 단체 채팅 서버 구현

# Ex0: 단체 채팅 서버 구현

- 이 과제는 단체 채팅 서버를 구현하는 내용
- 채팅 클라이언트는 nc 로 대신
- 서버 프로그램 실행 후 클라이언트 접속

```
sanghwan@PC: ~/dbox/consulting/programmers/KDT/ex0_chatserver
sanghwan@PC:~/dbox/consulting/programmers/KDT/ex0_chatserver$ gcc -o cs chatsrv.c
sanghwan@PC:~/dbox/consulting/programmers/KDT/ex0_chatserver$ ./cs 10000
Student ID : 20000000
Name : Sanghwan
connection from host 127.0.0.1, port 63697, socket 4 slot 0
connection from host 127.0.0.1, port 63699, socket 5 slot 1
connection from host 127.0.0.1, port 63705, socket 6 slot 2
```

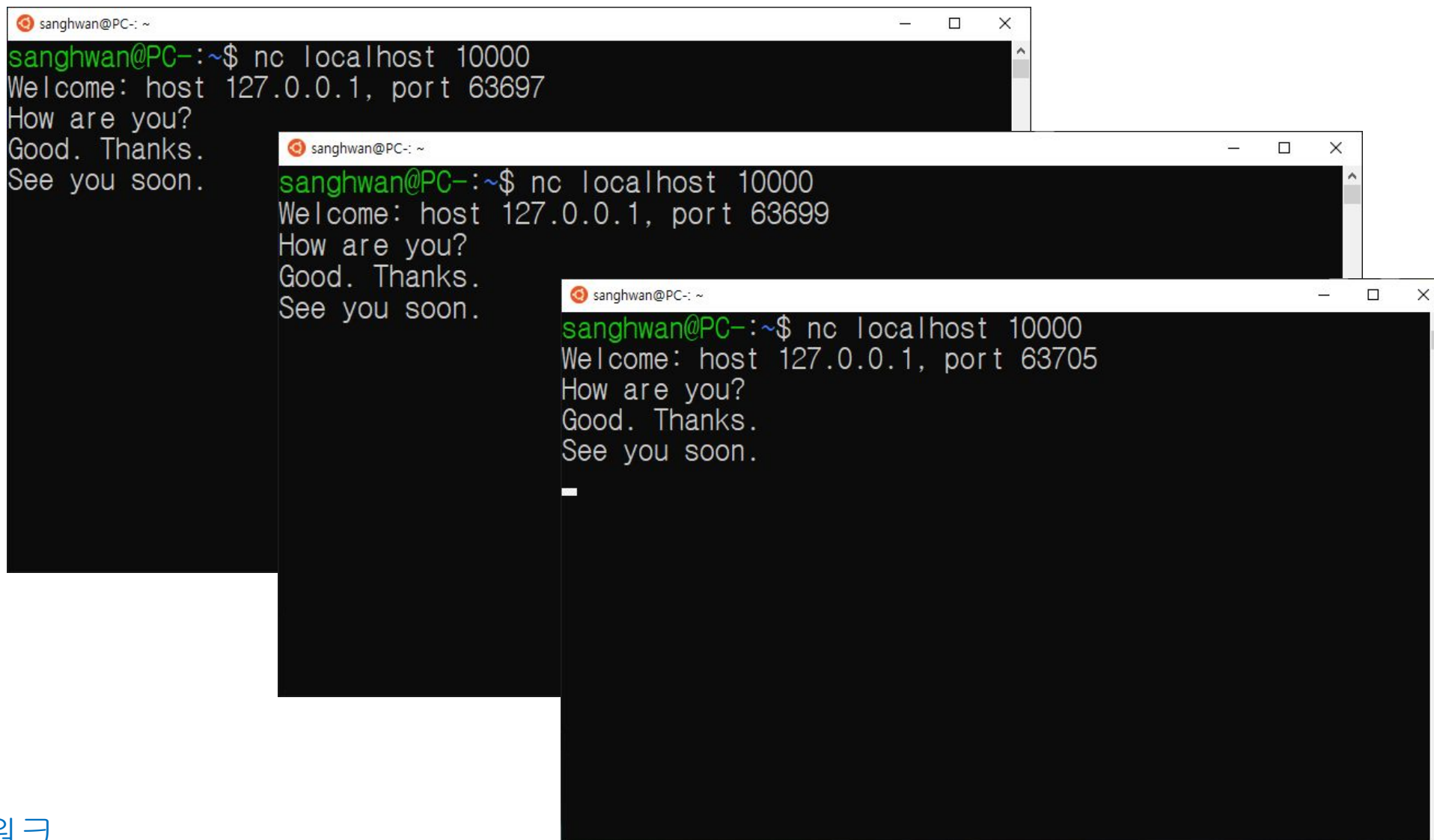
# Ex0: 단체 채팅 서버 구현

95

- 클라이언트가 접속 시 서버는 **Welcome** 메시지 전송
- 한 클라이언트에서 메시지 전송 시 다른 클라이언트 모두 출력

# Ex0: 단체 채팅 서버 구현

96



The image displays three overlapping terminal windows, each representing a client connected to a chat server. The windows are titled 'sanghwan@PC: ~'. Each window shows the following sequence of text:

```
sanghwan@PC-:~$ nc localhost 10000
Welcome: host 127.0.0.1, port 63697
How are you?
Good. Thanks.
See you soon.
```

The first window (top-left) shows the connection to port 63697. The second window (middle) shows the connection to port 63699. The third window (bottom-right) shows the connection to port 63705. Each window has a scrollbar on the right side.



# Ex0: 단체 채팅 서버 구현

97

- 구현 팁
  - 서버는 각 클라이언트가 접속시 해당 정보를 배열에 저장
  - `select()` 함수를 사용하여 여러 개의 소켓을 감시
  - 하나의 소켓에서 입력이 들어오면 클라이언트 배열에 저장된 다른 소켓으로 그 내용을 그대로 출력

# Ex0: 단체 채팅 서버 구현

98

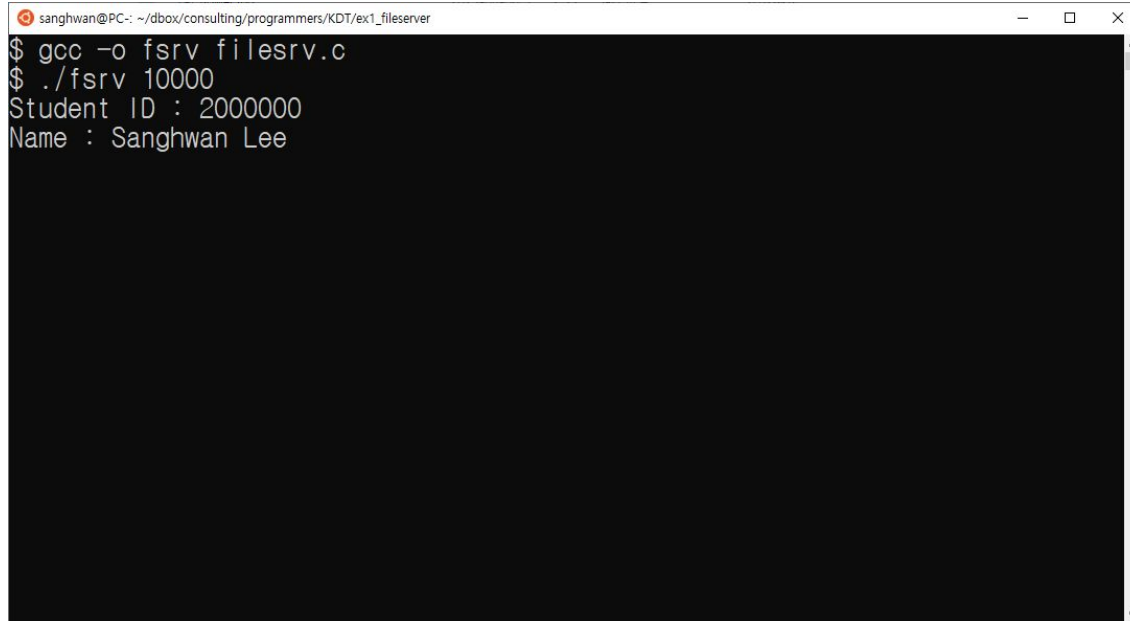
- 구현 진행 (1시간)

- 소켓 프로그래밍 이론 및 실습 1
  - Ex1: 파일 서버 구현

# Ex1: 파일 서버 구현

100

- 이 과제는 파일 서버를 구현하는 내용
- 파일 클라이언트는 nc 로 대신
- 서버 프로그램 실행 후 클라이언트 접속

A terminal window with a black background and white text. The window title is "sanghwan@PC: ~/dbox/consulting/programmers/KDT/ex1\_fileserver". The command history shows: "\$ gcc -o fsrv filesrv.c", "\$ ./fsrv 10000", and the program output: "Student ID : 2000000" and "Name : Sanghwan Lee".

```
sanghwan@PC: ~/dbox/consulting/programmers/KDT/ex1_fileserver
$ gcc -o fsrv filesrv.c
$ ./fsrv 10000
Student ID : 2000000
Name : Sanghwan Lee
```

# Ex1: 파일 서버 구현

101

- 기능 1
  - 특정 확장자를 가진 파일 목록 다운로드

```
sanghwan@PC: ~  
sanghwan@PC-:~$ echo "LS html" | nc localhost 10000  
biga.html  
bigb.html  
kkk.html  
sanghwan@PC-:~$ echo "LS jpg" | nc localhost 10000  
flower.jpg  
sanghwan@PC-:~$ echo "LS c" | nc localhost 10000  
filesrv.c  
sanghwan@PC-:~$
```

LS ext: 이 명령이 들어오면 서버는 현재 디렉토리에서 파일 확장자가 **ext**인 파일을 찾아서 **tcp** 연결을 통해 전송한다. 전송시에는 한 라인에 한 파일명을 전송한다. 구체적으로는 라인 구분을 “\r\n”으로 해서 각 파일에 대해서 “파일명\r\n”의 형태로 전송하면 된다.

# Ex1: 파일 서버 구현

102

- 기능2
  - 주어진 이름의 파일 다운로드

```
sanghwan@PC: ~  
sanghwan@PC-:~$ ls *.html  
ls: cannot access '*.html': No such file or directory  
sanghwan@PC-:~$ echo "GET biga.html" | nc localhost 10000 > biga.html  
sanghwan@PC-:~$ ls *.html  
biga.html  
sanghwan@PC-:~$ ls -l *.html  
-rw-r--r-- 1 sanghwan sanghwan 10818 Apr 26 13:39 biga.html  
sanghwan@PC-:~$
```

GET filename: 이 명령이 들어오면 서버는 filename에 해당하는 파일을 찾아서 tcp 연결을 통해 전송한다. 만약 존재하지 않으면 "FILE NOT FOUND\r\n" 을 클라이언트로 전송한다.

# Ex1: 파일 서버 구현

103

- 기능 3
  - 파일 업로드

```
sanghwan@PC: ~  
sanghwan@PC-:~$ ls *.zip  
ot.zip  
sanghwan@PC-:~$ echo "PUT ot1.zip" | cat - ot.zip | nc localhost 10000  
^C  
sanghwan@PC-:~$ cmp ot.zip ~/dbox/consulting/programmers/KDT/ex1_fileserve  
r/ot1.zip  
sanghwan@PC-:~$ _
```

PUT filename: 이 명령이 들어오면 서버는 **filename** 이라는 파일을 현재 디렉토리에 생성한 후, 이 **tcp** 연결을 통해 명령 라인 이후에 들어오는 모든 내용을 이 **filename** 파일에 쓴다.

# Ex1: 파일 서버 구현

104

- 구현 진행 (1시간)