6. 데몬 서버 구축 방법

- 서버 프로그램을 데몬(daemon) 프로세스로 실행시키는 방법을 소개
- inetd를 이용하여 임의의 서버 프로그램을 호스트가 제공하는 네트워크 서비스 프로그램으로 등록하는 방법을 소개

6.1 데몬 프로세스

데몬 프로세스

- 백그라운드로 실행되는 프로세스
- 특정 터미널의 제어와 관계없이 실행
  - 터미널로 메시지를 출력하는 fprintf() 같은 함수를 사용할 수 없으므로 에러 출력이나 메시지는 파일에 기록
- 데몬을 터미널에서 실행시킨 후에 터미널 사용자가 로그 아웃하거나 ctrl-c를 눌러도 종료되지 않음
- 데몬 종료는 kill 명령어나 SIGKILL 시그널을 사용

데몬 프로세스 생성

- **일반 프로그램을 데몬으로 만들 때는 아래와 같은 코드를 프로그램 앞부분에 추가**

```
//프로그램을 데몬 프로세스로 실행시키는 코드
struct sigaction sact;
sigset_t mask;

if ((pid=fork()) != 0)          //자식 프로세스를 생성
   exit(0);                     // 부모 프로세스 종료
setsid();                       // 자식 프로세스가 세션 리더가 됨

// SIGHUP 시그널(hang up, 터미널과 연결이 끊어졌을 때 세션 리더로 보냄)을 무시
sact.sa_handler = SIG_IGN;
sact.sa_flags = 0;
sigemptyset(&sact.sa_mask);
sigadd(&sact.sa_mask, SIGHUP);
sigaction(SIGHUP, &sact);

if ((pid=fork()) != 0)          // 다시 자식프로세스(손자) 생성
   exit(0);                     // 부모 프로세스 종료
chdir("/");                     // 디렉토리 변경
umask(0);                       //새로 생성되는 파일이 임의의 소유권을 갖도록 설정
for (i=0; i<MAXFD; i++)
   close(i);                    // 부모 프로세스가 개설한 모든 소켓을 닫음
//이후에  일반적인 서버 동작 부분을 기술
```

데몬 서버의 종류

- **독립형 데몬**
  - httpd, sendmail, named **등과 같이 항상 실행되면서 서비스를 제공**
- inetd **데몬**
  - **슈퍼데몬 또는 슈퍼서버**(superserver)**라고도 부름**
  - **여러 가지 인터넷 서비스 요청을 받아서 처리하는 종합적인 인터넷 서비스용 데몬**
  - echo, daytime**과 같은 간단한 서비스 요청은** inetd**에서 직접 처리**
  - telnet, ftp**와 같은 요청은** inetd**가 그 서비스를 처리하기 위한 새로운 프로세스를 생성**
    - inetd**에서 각 서비스를 구분하기 위해서 포트번호를 사용**
- **독립형 데몬과 슈퍼데몬을 이용하는 서비스**
  - **독립형 데몬은 특정 서비스를 위한 전용 프로그램이므로 좀더 빠르게 응답**
  - **독립형 데몬은 프로세스가 항상 실행되고 있어야 하므로 비 효율적**
  - **독립형 데몬은 클라이언트의 요청 빈도가 많거나 지속적인 서비스에 주로 사용**
  - **슈퍼 데몬은 요청이 빈번하지 않은 서비스**(telnet **등**) **에 주로 사용**

6.2 독립형 데몬 서버

- 에코 서버를 독립형 데몬 프로세스로 구현

독립형 데몬 프로세스의 에코 서버 (myecho_daemon.c)

- 2장의 일반 에코 서버에 데몬 프로세스를 위한 코드를 추가
  - 클라이언트는 2장의 에코 클라이언트(tcp_echocli.c)를 사용
- 데몬의 실행 여부 확인

```
ps -aux
```

- 데몬 프로세스의 종료
  - 데몬 프로세스의 PID를 구한 후

```
kill -9 pid
```

- myecho_daemon.c 소스

6.3 inetd를 이용한 서버 구축

- inetd 데몬을 이용하여 서버를 구축하는 방법
- 리눅스에서는 inetd대신에 xinetd를 제공

inetd 환경 설정

- 에코 서버 프로그램을 inetd로 서비스하는 방법을 소개
- 인터넷 서비스 등록 파일과 inetd 환경설정 파일에 서비스가 등록되어야 함
  - /etc/services
  - /etc/inetd.conf
- 예 : 5000번 포트로 에코 서비스를 제공시 두 파일에 내용을 추가

```
#/etc/services 파일에 추가
myecho_service      5000/tcp
```

```
#/etc/inetd.conf 파일에 추가
#서비스명          소켓타입  프로토콜  wait/nowait    사용자   경로
myecho_service  stream    tcp          nowait       root     /root/myecho_service
myecho_service
```

  - wait/nowait는 클라이언트의 요청이 들어오면 inetd가 새로운 프로세스를 생성할지, 이전에 서비스중인 서버 프로세스가 종료하기를 기다린 후에 요청할 것인지를 구분
  - /etc/rc.d/init.d/inet restart 명령으로 재시작하면 서비스 등록이 완료

TCP Wrapper

- **인터넷 서비스를 안전하게 제공하기 위하여** TCP Wrapper**를 실행시키는 경우가 있음**
  - **외부에서** telnet, ftp **등의** TCP/IP **응용 서비스를 이용하고자 할 때 서비스 제공 여부를 상대방** IP **주소를 보고 제한하는 기능을 제공**
  - tcpd **데몬이 제공**
  - ftp**가** TCP Wrapper**를 사용하는 경우와 사용하지 않는 경우**

```
# /etc/inetd.conf 파일에서 비교
// TCP Wrapper를 사용하는 경우
ftp stream TCP nowait root /usr/sbin/tcpd in.ftpd –l –a

// TCP Wrapper를 사용하지 않는 경우
ftp stream TCP nowait root /usr/sbin/ftpd ftpd
```

/etc/hosts.allow와 /etc/hosts.deny

- TCP Wrapper에 의해 보호되는 데몬의 접근을 제어하기 위해 /etc/hosts.allow와 /etc/hosts.deny 파일을 사용
- **두 파일의 형식은 동일**

```
daemon_list : client_list : option : option ...
```

  - daemon_list : **접근을 제어하고 싶은 서버 데몬**
  - client_list : **접근을 제어할 클라이언트 리스트**
- IP가 203.211.252.36인 클라이언트에 대해 ftp, telnet을 제한

```
# /etc/hosts.deny 파일
ftp, telnet : 203.211.252.36
# all : 203.211.252.36          // 모든 서비스를 제한
```

- **접근 제어**
  - **접근 허용과 거부의 순서가 중요함**

```
# /etc/hosts.deny
ALL : ALL

# /etc/hosts.allow
ALL : 192.168.10.0/255.255.255.0
```

6.4 xinetd를 이용한 서버 구축

- 리눅스에서는 inetd, TCP wrapper 기능을 xinetd 에서 제공
- xinetd를 이용하여 서버 프로그램을 데몬으로 처리하는 방법

xinetd 환경 설정

- xinetd 환경 설정을 위해서는 아래의 두 파일을 이용
  - /etc/rc.d/init.d/xinetd
  - /etc/xinetd.conf
    - xinetd가 지원하는 인터넷 서비스와 해당 포트 등을 정의
    - defaults 부분과 service 부분으로 나뉨
    - defaults 부분은 xinetd가 관리하는 모든 서비스에 적용
    - service 부분은 특정 데몬 서버를 위한 설정
- default 부분 설정

```
defaults {
  instances = 60                  // 하나의 서비스에 대해 동시 처리 가능한 최대 요청 수
                                  // 이 값을 적절히 설정하여 서비스 거부 공격을 막음
  log_type = SYSLOG authpriv      // 프로그램에 의해 생성되는 로그의 형태
                                  // FILE : 로그 파일의 완전한 경로
                                  // SYSLOG : 시스템의 syslog 기능 이용
  log_on_success = HOST PID       // 서비스 시작시 어떤 정보가 로그에 남을지 정의
  only_from =                     // 서비스를 이용할 수 있도록 허용하는 호스트 정의
                                  // 정의하지 않으면 모든 접근을 거부
  per_source = 5                  // 접속 가능한 최대 접속자 수
  enabled = pop3s imaps           // xinetd에서 지원하는 서비스 이름 설정
}
```

xinetd 환경 설정

- service 부분 설정
  - 특정 서비스에 대한 접근을 설정하는데 사용
  - 에코 서비스를 위한 설정

```
service echo {        // echo는 유일한 이름, /etc/services 에 정의된 이름과 동일
  disable = no                    // no 이어야 서비스 가능
  socket_type = stream            // 소켓 타입
  wait = no
  user = root                     // 서버를 실행시키는 사용자 이름
  type = INTERNAL                 // RPC, INTERNAL, UNLISTED
  id = echo-stream                // 에코 서비스를 TCP, UDP 모두 지원할 경우 id로 구분
  protocol = tcp
  only_from = 210.115.36.0/24 192.168.1.0/24
            // 접근을 허용한 주소, 0.0.0.0 은 모든 호스트의 접근을 허용
  no_access = 207.23.32.44    // 접근을 차단할 주소
```

  - type
    - RPC : /etc/rpc 파일에 정의되어 있는데 그다지 잘 작동되지 않는다.
    - INTERNAL : echo, time, daytime, chargen 및 discard 와 같은 xinetd 가 직접적으로 다루는 서비스
    - UNLISTED : /etc/rpc 또는 /etc/service파일에 정의되어 있지 않은 서비스

xinetd를 이용한 에코 서버(echoserv.c)

```
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <netdb.h>
#include <netinet/tcp.h>

int main()
{
  int cnt = 0;
  char line[1024];

  setvbuf(stdout. NULL, _IOLBF, 0);
  while (fgets(line, sizeof(line), stdin) != NULL)
            pritf("%3i : %s", ++cnt, line);
  return 0;
}
```

- **프로그램에 소켓 관련 함수**(socket, accept 등)**가 없다**
  - **에코 서비스 포트번호로 서비스 요청이 들어오면** xinetd**가** accept()**를 대신 호출하여 클라이언트와 연결하고 위 프로그램을 실행**
  - **연결 후 소켓으로 입출력을 서버 프로그램의 표준 입출력과 내부적으로 연결**

xinetd에 서비스를 등록하는 절차

- **등록 절차**
  - /etc/services **파일에** myecho_serv **서버를 등록**
  - /etc/xinetd.d/myecho_serv **서비스 파일을 생성하고** xinetd**와 관련된 환경을 설정**
  - xinetd **재시작**

```
# /etc/services 파일에 추가
myecho_serv 9200/tcp
```

```
# /etc/xinetd.d/myecho_serv 파일 생성
service myecho_serv {
  disable = no
  socke_type = stream
  wait = no
  user = root
  server = /myserver_path/myecho_serv
  log_on_failuer += USERID
  port = 9200
}
```

```
# xinetd 재시작
/etc/rc.d/init.d/xinetd restart
```

xinetd 서비스의 특징

- **장점**
  - TCP **연결 요청에 대해** socket(), listen(), accept() **함수가 자동으로 처리**
  - **내부적으로** fork()**와** exec() **를 실행하여 등록된 서버 프로그램 실행**
  - **항상 실행되지 않고 필요할 때에만 실행되고 종료**
    - **시스템 자원 절약**
- **단점**
  - **여러 클라이언트가 동시에 연결 요청을 하면, 클라이언트의 수만큼 데몬 프로세스가 실행될 수 있음**
  - **즉시 응답을 보내지 못하여 반응 속도가 느릴 수 있음**
- **자주 실행되지 않는 서비스에 적합**
- **서비스 추가시 보안정책 등을 일괄적으로 설정하는데 편리**

UDP 에코 프로그램

- UDP**는 메시지가 도착했을 때 서버 프로세스를 생성**
  - TCP**는 연결 요청이 들어왔을 때 서버 프로세스를 생성**
  - **소스**

```
int main(int argc, char *argv[]) {
    struct sockaddr_in   peer;
    int rc, len = sizeof(struct sockaddr), pidsz;
    char buf[120];
    FILE *file;

    pidsz = sprintf(buf, "%d: ", getpid());
    rc = recvfrom(0, buf+pidsz, sizeof(buf)-pidsz, 0,
                                  (struct sockaddr *)&peer, &len);

    if(rc < 0) exit(1);
    sendto(1, buf, rc+pidsz, 0, (struct sockaddr *)&peer, len);
    exit(0);
}
```

xinetd에 서비스를 등록하는 절차

- **등록 절차**
  - /etc/services **파일에** myecho_udpserv **서버를 등록**
  - /etc/xinetd.d/myecho_udpserv **서비스 파일을 생성하고** xinetd**와 관련된 환경을 설정**
  - xinetd **재시작**

```
# /etc/services 파일에 추가
myecho_udpserv 9300/udp
```

```
# /etc/xinetd.d/myecho_udpserv 파일 생성
service myecho_serv {
  disable = no
  socke_type = dgram
  wait = no
  user = root
  server = /myserver_path/myecho_udpserv
  log_on_failuer += USERID
  port = 9300
}
```

```
# xinetd 재시작
/etc/rc.d/init.d/xinetd restart
```