# Socket Programming 2

Jo, Heeseung

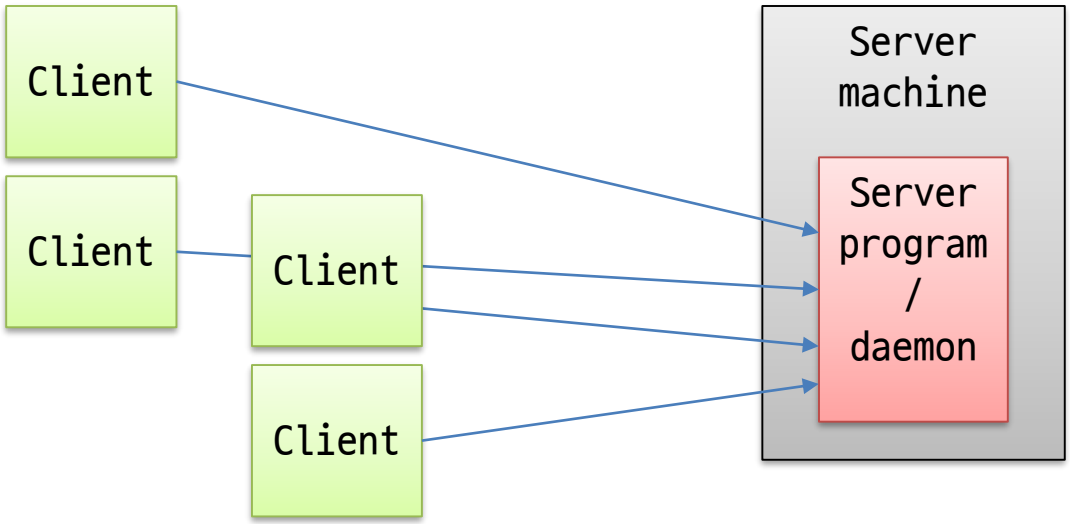
# 반복서버 vs. 동시동작서버

반복서버

- 데몬 프로세스가 직접 모든 클라이언트의 요청을 차례로 처리

동시동작서버

- 데몬 프로세스가 직접 서비스를 제공하지 않고, 서비스와 관련있는 다른 프로세스를 fork 함수로 생성해 클라이언트와 연결시켜줌



# [예제 12-1] (1) 반복서버(서버)

```
...
#define PORTNUM 9001

int main(void) {
    char buf[256];
    struct sockaddr_in sin, cli;
    int sd, ns, clientlen = sizeof(cli);

    memset((char *)&sin, '\0', sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(PORTNUM);
    sin.sin_addr.s_addr = inet_addr("0.0.0.0");

    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }
    if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
        perror("bind");
        exit(1);
    }
    if (listen(sd, 5)) {
        perror("listen");
        exit(1);
    }
```

소켓 주소구조체 생성

소켓 생성

클라이언트 접속 대기

# [예제 12-1] (1) 반복서버(서버)

```
    while (1) {
        if ((ns = accept(sd, (struct sockaddr *)&cli, &clientlen)) == -1) {
            perror("accept");
            exit(1);
        }
        sprintf(buf, "%s", inet_ntoa(cli.sin_addr));
        printf("*** Send a Message to Client(%s)\n", buf);

        strcpy(buf, "Welcome to Network Server!!!");
        if (send(ns, buf, strlen(buf) + 1, 0) == -1) {
            perror("send");
            exit(1);
        }

        if (recv(ns, buf, sizeof(buf), 0) == -1) {
            perror("recv");
            exit(1);
        }
        printf("** From Client : %s\n", buf);
        close(ns);
    }
    close(sd);

    return 0;
}
```

클라이언트 접속

클라이언트에 정보전송

클라이언트의 데이터 수신

# [예제 12-1] (2) 반복서버(클라이언트)

```
...
#define PORTNUM 9001

int main(void) {
    int sd;
    char buf[256];
    struct sockaddr_in sin;

    memset((char *)&sin, '\0', sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(PORTNUM);
    sin.sin_addr.s_addr = inet_addr("192.168.162.133");

    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }

    if (connect(sd, (struct sockaddr *)&sin, sizeof(sin))) {
        perror("connect");
        exit(1);
    }
```

소켓 주소구조체 생성

소켓 생성

서버에 연결 요청

# [예제 12-1] (2) 반복서버(클라이언트)

```
 if (recv(sd, buf, sizeof(buf), 0) == -1) {
        perror("recv");
        exit(1);
    }

    printf("** From Server : %s\n", buf);

    strcpy(buf, "I want a HTTP Service.");
    if (send(sd, buf, sizeof(buf) + 1, 0) == -1) {
        perror("send");
        exit(1);
    }

    close(sd);

    return 0;
}
```

서버의 데이터 수신

서버에 데이터 송신

서버

```
# ex12_1s.out
```

클라이언트

```
# ex12_1c.out
** From Server : Welcome to Network Server!!!
```

서버

```
# ex12_1s.out
** From Client : I want a HTTP Service.
```

서버

```
# ex12_1s.out
** From Client : I want a HTTP Service.
```

# [예제 12-2] (1) 동시 동작 서버(서버)

```
...
#define PORTNUM 9002

int main(void) {
    char buf[256];
    struct sockaddr_in sin, cli;
    int sd, ns, clientlen = sizeof(cli);

    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }

    memset((char *)&sin, '\0', sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(PORTNUM);
    sin.sin_addr.s_addr = inet_addr("0.0.0.0");

    if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
        perror("bind");
        exit(1);
    }

    if (listen(sd, 5)) {
        perror("listen");
        exit(1);
    }
```

# [예제 12-2] (1) 동시 동작 서버(서버)

```
    while (1) {
        if ((ns = accept(sd, (struct sockaddr *)&cli, &clientlen)) == -1) {
            perror("accept");
            exit(1);
        }
        switch (fork()) {
            case 0:
                close(sd);
                strcpy(buf, "Welcome to Server");
                if (send(ns, buf, strlen(buf) + 1, 0) == -1) {
                    perror("send");
                    exit(1);
                }

                if (recv(ns, buf, sizeof(buf), 0) == -1) {
                    perror("recv");
                    exit(1);
                }
                printf("** From Client: %s\n", buf);
                //sleep(5);
                exit(0);
        }
        close(ns);
    }
    return 0;
}
```

fork로 자식 프로세스 생성

자식 프로세스가 클라이언트로 메시지 보내고 데이터 수신

# [예제 12-2] 실행결과

```
# ex12_2s.out
** From Client : I want a HTTP Service.
```

- 클라이언트는 ex12_1c.c를 포트번호만 바꾸고 그대로 사용

- 클라이언트가 접속했을 때 서버의 실행상태

```
# ps -ef ¦ grep ex12
root  7175  7172   0 09:55:32 pts/2      0:00 ex12_2s.out
root  7172  1571   0 09:55:18 pts/2      0:00 ex12_2s.out
```

  - 서버 프로세스가 2개 임을 알 수 있다.
  - 7172는 부모 프로세스, 7175는 자식 프로세스

# Ex: thread 기반 동시동작서버

실습내용

- fork() 기반의 동시동작서버를 pthread를 이용하여 thread 기반 동시동작서버로 변경