2016116783 김성록

Example 1

echo_epollserv.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#define BUF_SIZE 100
#define EPOLL_SIZE 50
void error_handling(char *message);
int main(int argc, char *argv[]){
   int serv_sock;
   int clnt_sock;
   struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t adr_sz;
    char buf[BUF_SIZE];
    int str_len, i;
    struct epoll_event *ep_events;
    struct epoll_event event;
    int epfd, event_cnt;
    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sock==-1)
        error_handling("socket() error");
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    serv_addr.sin_port=htons(atoi(argv[1]));
    if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
```

```
error_handling("bind() error");
    if(listen(serv_sock,5)==-1)
        error_handling("listen() error");
    epfd=epoll_create(EPOLL_SIZE);
    ep_events=malloc(sizeof(struct epoll_event)*EPOLL_SIZE);
    event.events=EPOLLIN;
    event.data.fd=serv_sock;
    epoll_ctl(epfd, EPOLL_CTL_ADD, serv_sock, &event);
    while(1){
        event_cnt=epoll_wait(epfd, ep_events, EPOLL_SIZE, -1);
        if(event_cnt==-1){
            puts("epoll_wait() error");
        }
        for(i=0;i<event_cnt;i++){</pre>
            if(ep_events[i].data.fd==serv_sock){
                adr_sz=sizeof(clnt_addr);
                clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &adr_sz);
                event.events=EPOLLIN;
                event.data.fd=clnt_sock;
                epoll_ctl(epfd, EPOLL_CTL_ADD, clnt_sock, &event);
                printf("connected client: %d \n", clnt_sock);
            }
            else{
                str_len=read(ep_events[i].data.fd, buf, BUF_SIZE);
                if(str_len==0){
                    epoll_ctl(epfd, EPOLL_CTL_DEL, ep_events[i].data.fd, NULL);
                    close(ep_events[i].data.fd);
                    printf("closed client: %d \n",ep_events[i].data.fd);
                }
                else{
                        write(ep_events[i].data.fd, buf, str_len);
                }
            }
        }
    }
    close(serv_sock);
    close(epfd);
    return 0;
}
void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n',stderr);
    exit(1);
}
```

```
user@DESKTOP-L301CH6:~/Network_programming/Assignment8$ ./epollserv1 2999
connected client: 5
closed client: 6
client client.
client client client.
client client.
client client.
client client.
client client.
client.
client client.
client.
client client.
clie
```

Example 2

echo EPLTserv.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#define BUF_SIZE 4
#define EPOLL_SIZE 50
void error_handling(char *message);
int main(int argc, char *argv[]){
    int serv_sock;
    int clnt_sock;
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t adr_sz;
    char buf[BUF_SIZE];
    int str_len, i;
    struct epoll_event *ep_events;
    struct epoll_event event;
    int epfd, event_cnt;
    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }
    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sock==-1)
        error_handling("socket() error");
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
```

```
serv_addr.sin_port=htons(atoi(argv[1]));
    if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
        error_handling("bind() error");
    if(listen(serv_sock,5)==-1)
        error_handling("listen() error");
    epfd=epoll_create(EPOLL_SIZE);
    ep_events=malloc(sizeof(struct epoll_event)*EPOLL_SIZE);
    event.events=EPOLLIN;
    event.data.fd=serv_sock;
    epoll_ctl(epfd, EPOLL_CTL_ADD, serv_sock, &event);
    while(1){
        event_cnt=epoll_wait(epfd, ep_events, EPOLL_SIZE, -1);
        if(event_cnt==-1){
            puts("epoll_wait() error");
            break;
        puts("return epoll_wait");
        for(i=0;i<event_cnt;i++){</pre>
            if(ep_events[i].data.fd==serv_sock){
                adr_sz=sizeof(clnt_addr);
                clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &adr_sz);
                event.events=EPOLLIN;
                event.data.fd=clnt_sock;
                epoll_ctl(epfd, EPOLL_CTL_ADD, clnt_sock, &event);
                printf("connected client: %d \n", clnt_sock);
            }
            else{
                str_len=read(ep_events[i].data.fd, buf, BUF_SIZE);
                if(str_len==0){
                    epoll_ctl(epfd, EPOLL_CTL_DEL, ep_events[i].data.fd, NULL);
                    close(ep_events[i].data.fd);
                    printf("closed client: %d \n",ep_events[i].data.fd);
                }
                else{
                        write(ep_events[i].data.fd, buf, str_len);
                }
            }
        }
    close(serv_sock);
    close(epfd);
    return 0;
}
void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n',stderr);
    exit(1);
}
```

```
user@DESKTOP-L301CH6:~/Network_prog
                                      user@DESKTOP-L301CH6:~/Network_prog
ramming/Assignment8$ gcc -Wall -o E
                                      ramming/Assignment8$ ./echo client
PLT echo EPLTserv.c
                                      127.0.0.1 1
user@DESKTOP-L301CH6:~/Network_prog
                                      234
ramming/Assignment8$ ./EPLT 1234
                                      Connected.....
return epoll wait
                                      Input message(Q to quit): I Love Pr
connected client: 5
                                      ogramming
return epoll wait
                                      Message form server : I LoInput mes
return epoll wait
                                      sage(Q to quit): do you like progra
return epoll wait
return epoll wait
                                      Message form server : ve Programmin
return epoll wait
return epoll wait
                                      Input message(Q to quit): q
return epoll wait
                                      user@DESKTOP-L301CH6:~/Network_prog
                                      ramming/Assignment8$
return epoll wait
return epoll_wait
closed client: 5
```

Example 3

echo_EPETserv.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#define BUF_SIZE 4
#define EPOLL_SIZE 50
void error_handling(char *message);
void setnonblockingmode(int fd);
int main(int argc, char *argv[]){
   int serv_sock;
   int clnt_sock;
   struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
```

```
socklen_t adr_sz;
char buf[BUF_SIZE];
int str_len, i;
struct epoll_event *ep_events;
struct epoll_event event;
int epfd, event_cnt;
if(argc!=2){
    printf("Usage : %s <port>\n", argv[0]);
    exit(1);
}
serv_sock = socket(PF_INET, SOCK_STREAM, 0);
if(serv_sock==-1)
    error_handling("socket() error");
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
serv_addr.sin_port=htons(atoi(argv[1]));
if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");
if(listen(serv_sock,5)==-1)
    error_handling("listen() error");
epfd=epoll_create(EPOLL_SIZE);
ep_events=malloc(sizeof(struct epoll_event)*EPOLL_SIZE);
setnonblockingmode(serv_sock);
event.events=EPOLLIN;
event.data.fd=serv_sock;
epoll_ctl(epfd, EPOLL_CTL_ADD, serv_sock, &event);
while(1){
    event_cnt=epoll_wait(epfd, ep_events, EPOLL_SIZE, -1);
    if(event_cnt==-1){
        puts("epoll_wait() error");
        break;
    }
    puts("return epoll_wait");
    for(i=0;i<event_cnt;i++){</pre>
        if(ep_events[i].data.fd==serv_sock){
            adr_sz=sizeof(clnt_addr);
            \verb|clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &adr_sz);|\\
            event.events=EPOLLIN|EPOLLET;
            event.data.fd=clnt_sock;
            epoll_ctl(epfd, EPOLL_CTL_ADD, clnt_sock, &event);
            printf("connected client: %d \n", clnt_sock);
        }
        else{
            str_len=read(ep_events[i].data.fd, buf, BUF_SIZE);
            if(str_len==0){
                epoll_ctl(epfd, EPOLL_CTL_DEL, ep_events[i].data.fd, NULL);
                close(ep_events[i].data.fd);
                printf("closed client: %d \n",ep_events[i].data.fd);
```

```
else if(str_len<0){</pre>
                    if(errno==EAGAIN)
                        break;
                }
                else{
                        write(ep_events[i].data.fd, buf, str_len);
                }
            }
        }
    }
    close(serv_sock);
    close(epfd);
    return 0;
}
void setnonblockingmode(int fd){
    int flag=fcntl(fd, F_GETFL, 0);
    fcntl(fd, F_SETFL, flag|0_NONBLOCK);
}
void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n',stderr);
    exit(1);
}
```

```
user@DESKTOP-L301CH6:~/Netw
                              user@DESKTOP-L301CH6:~/Netw
ork_programming/Assignment8
                              ork programming/Assignment8
$ ./EPET 9111
                              $ ./echo client 127.0.0.1 9
return epoll wait
                              111
connected client: 5
                              Connected.....
return epoll wait
                              Input message(Q to quit): i
return epoll wait
                               like cimputer promgramming
return epoll wait
                              Message form server : i liI
return epoll wait
                              nput message(Q to quit): do
return epoll wait
                               you like programming?
user@DESKTOP-L301CH6:~/Netw
                              Message form server : ke cI
ork programming/Assignment8
                              nput message(Q to quit): go
$ I I
                              od bye
                              Message form server : impuI
                              nput message(Q to quit): q
                              user@DESKTOP-L301CH6:~/Netw
                              ork programming/Assignment8
                              $
```

Problem

Design a program by yourself which uses epoll concepts

objective

Let's make message queue program that accumulate logs from client program.

motivation

Trying to find service fit with epoll, finally find message queue using edge trigger epoll. I think it perfectly good for message queue service to save the logs

solution

using edge trigger epoll, when edge trigger event occur, server get message from client and send the message to host that connected with server first. except first client(host), all of client's message will be transmitted to host client (first one).

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#define BUF SIZE 4
#define EPOLL_SIZE 50
void error_handling(char *message);
int main(int argc, char *argv[]){
   int serv_sock;
    int clnt_sock;
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t adr_sz;
    char buf[BUF_SIZE];
    char *host = "hostn";
    char *client = "client\n";
    int str_len, i;
    struct epoll_event *ep_events;
    struct epoll_event event;
    int epfd, event_cnt;
    int host_sock;
    int active_count;
    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }
    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sock==-1)
        error_handling("socket() error");
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    serv_addr.sin_port=htons(atoi(argv[1]));
    if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
        error_handling("bind() error");
    if(listen(serv_sock,5)==-1)
        error_handling("listen() error");
```

```
epfd=epoll_create(EPOLL_SIZE);
    ep_events=malloc(sizeof(struct epoll_event)*EPOLL_SIZE);
    event.events=EPOLLIN;
    event.data.fd=serv_sock;
    epoll_ctl(epfd, EPOLL_CTL_ADD, serv_sock, &event);
    while(1){
        event_cnt=epoll_wait(epfd, ep_events, EPOLL_SIZE, -1);
        if(event_cnt==-1){
            puts("epoll_wait() error");
            break;
        }
        for(i=0;i<event_cnt;i++){</pre>
            if(ep_events[i].data.fd==serv_sock){
                adr_sz=sizeof(clnt_addr);
                clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &adr_sz);
                event.events=EPOLLIN;
                event.data.fd=clnt_sock;
                epoll_ctl(epfd, EPOLL_CTL_ADD, clnt_sock, &event);
                printf("connected client: %d \n", clnt_sock);
                if(!active_count){
                    host_sock=clnt_sock;
                    write(clnt_sock, host, strlen(host));
                }
                else{
                    write(clnt_sock, client, strlen(host));
                }
                active_count++;
            }
            else{
                str_len=read(ep_events[i].data.fd, buf, BUF_SIZE);
                if(str_len==0){
                    epoll_ctl(epfd, EPOLL_CTL_DEL, ep_events[i].data.fd, NULL);
                    close(ep_events[i].data.fd);
                    printf("closed client: %d \n",ep_events[i].data.fd);
                    if(active_count==1)
                        host_sock=0;
                    active_count--;
                }
                else{
                    write(host_sock, buf, str_len);
                }
            }
        }
    }
    close(serv_sock);
    close(epfd);
    return 0;
}
void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n',stderr);
    exit(1);
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define BUF_SIZE 1024
void error_handling(char *message);
int main(int argc, char* argv[]){
        int sock;
        struct sockaddr_in serv_addr;
        char message[BUF_SIZE];
        int str_len;
        int host=0;
        if(argc != 3){
                printf("Usage: %s <IP> <port>\n", argv[0]);
        }
        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock==-1)
                error_handling("socket() error");
        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));
        if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
        else{
            str_len = read(sock, message, BUF_SIZE-1);
            message[str_len] = 0;
            if(!strcmp(message, "host\n")){
                host=1;
                puts("Connected as a host!");
            }
            else
                puts("Connected as a client!");
        while(1){
                if (!host){
                    fputs("Input message(Q to quit): ", stdout);
                    fgets(message, BUF_SIZE, stdin);
                    if(!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
                    write(sock, message, strlen(message));
                }
                else{
                    str_len = read(sock, message, BUF_SIZE-1);
                    message[str_len] = 0;
                    printf("%s", message);
```

```
}
    close(sock);
return 0;
}

void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

result

server

```
user@DESKTOP-L301CH6:~/Network_programming/Assignment8$ ./server 9112
connected client: 5
connected client: 6
connected client: 8
closed client: 8
closed client: 6
closed client: 7
```

host

```
user@DESKTOP-L301CH6:~/Network_programming/Assignment8$ ./client 127.0.0.1 9112
Connected as a host!
[01:43] DB connected!
[01:43] client connected!
[01:44] docker connected!
[01:44] client request quety to DB 192.168.0.5
[01:44] DB get query from client 10.0.0.8
[01:45] DB return result to client 10.0.0.8
[01:45] client get response from DB 192.168.0.5
[01:50] client request entering docker with id:15926816, password:5629019
[01:50] docker get request from client 10.0.0.8 with id:15926816, password:5629019
[01:51] docker face inter error (code:1191)
[01:52] docker fail to restart system. shut down docker...
[01:58] client cannot get data from docker (timeout)
```

client 1,2,3

```
user@OESKTOP-L301CH6:-/Network_programming/Assignment
$$./client 127.0.0.1 9112
Connected as a client!
Input message(Q to quit): [01:43] DB connected!
Input message(Q to quit): [01:43] DB get query from c
lient 10.0.0.8
Input message(Q to quit): [01:45] DB return result to
client 10.0.0.8
Input message(Q to quit): [01:45] DB return result to
client 10.0.0.8
Input message(Q to quit): q
user@OESKTOP-L301CH6:-/Network_programming/Assignment

$$./client 127.0.0.1 9112
Connected as a client!
Input message(Q to quit): [01:44] docker connected!
Input message(Q to quit): [01:45] client get response
from DB 192.168.0.5
Input message(Q to quit): [01:50] client request enter
ing docker with id:1592680.5
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:52] docker fail to rest
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from docker (timeout)
Input message(Q to quit): [01:58] client cannot get da
ta from do
```