CN-LAB MID

Naman Mulani

EDITOR.cpp

```
#include <bits/stdc++.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
using namespace std;

void sendSfd(int sfd)
{
sleep(1);
int usfd=socket(AF_UNIX, SOCK_DGRAM, 0);
struct sockaddr_un uAddr;
uAddr.sun_family = AF_UNIX;
strcpy(uAddr.sun_path, "socketfile");

struct iovec e = {NULL, 0};
char cmsg[CMSG_SPACE(sizeof(int))];
struct msghdr m = {(void*)&uAddr, sizeof(uAddr), &e, 1, cmsg, sizeof(cmsg), 0};
struct cmsghdr *c = CMSG_FIRSTHDR(&m);
c->cmsg_level = SOL_SOCKET;
c->cmsg_type = SCM_RIGHTS;
c->cmsg_len = CMSG_LEN(sizeof(int));
*(int*)CMSG_DATA(c) = sfd;
if(sendmsg(usfd, &m, 0)<0){
perror("sendmsg err");
exit(1);
```

```
}
cout<<"sent sfd"<<endl;
}

int main ()
{
int fd1 = fileno(popen("./reporter1" , "r"));
int fd2 = fileno(popen("./reporter2" , "r"));
int fd3 = fileno(popen("./reporter3" , "r"));

struct pollfd pfd[3];
pfd[0].fd = fd1; pfd[0].events = POLLIN;
pfd[1].fd = fd2; pfd[1].events = POLLIN;
pfd[2].fd = fd3; pfd[2].events = POLLIN;

while (1)
{
int s = poll(pfd , 3 , 1000);

if (s>0)
{
for (int i=0;i<3;i++)
{
if (pfd[i].revents & POLLIN)
{
cout<<"got message "<<endl;
char buffer[100];
int n = read(pfd[i].fd , buffer , sizeof(buffer)-1);
buffer[n] = '\0';

cout<<"I am editor and i read "<<buffer<<endl;

if (buffer[0]=='d')
{
cout<<"Now i will send the report to document ... "<<endl;
sleep(2);
sendSfd(pfd[i].fd);
sleep(10);
}
else
{
cout<<"Now i am sending to news"<<endl;
int sfd = socket(AF_INET , SOCK_STREAM , 0);
if (sfd==-1)
```

```
{
cout<<"Error in creating socket"<<endl;
return 1;
}

struct sockaddr_in serveraddress;
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(3000);
serveraddress.sin_addr.s_addr = inet_addr("127.0.0.0");

int t = connect (sfd , (sockaddr *)&serveraddress , sizeof(serveraddress));
if (t==-1)
{
cout<<"Error in connecting "<<endl;
return 1;
}

send(sfd , buffer , sizeof(buffer) , 0);

cout<<"sended"<<endl;
}
}
}
}
}
}
```

LIVE.cpp

```
#include <bits/stdc++.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/un.h>
```

```
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
using namespace std;

int main ()
{
int sfd = socket(AF_INET , SOCK_STREAM , 0);
if (sfd==-1)
{
cout<<"Error in creating socket"<<endl;
return 1;
}

struct sockaddr_in serveraddress;
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(9999);
serveraddress.sin_addr.s_addr = INADDR_ANY;

int binderror = bind (sfd , (struct sockaddr *)&serveraddress , sizeof(serveraddress));
if (binderror == -1)
{
cout<<"Error in binding "<<endl;
return 0;
}
int listenerror = listen(sfd , 10);
if (listenerror == -1)
{
cout<<"Listening error is found "<<endl;
return 0;
}

while (1)
{
struct sockaddr_in clientaddr;
socklen_t len = sizeof(clientaddr);
int newfd = accept(sfd , (struct sockaddr*)&clientaddr , &len);
if (newfd == -1)
{
```

```
cout<<"not"<<endl;
continue;
}

char buffer[100];
int n = recv(newfd , buffer , sizeof(buffer)-1 , 0);
buffer[n] = '\0';

cout<<"I am the live server , and i got the news : "<<buffer<<endl;

close(newfd);
}
}
```

NEWS1.cpp

```
#include <bits/stdc++.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
#include <sstream>
using namespace std;

int main ()
{
int sfd = socket(AF_INET , SOCK_STREAM , 0);
```

```
if (sfd==-1)
{
cout<<"Error in creating socket"<<endl;
return 1;
}

int reuse = 1;
if (setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&reuse, sizeof(reuse)) < 0)
perror("setsockopt(SO_REUSEADDR) failed");
if (setsockopt(sfd, SOL_SOCKET, SO_REUSEPORT, (const char*)&reuse, sizeof(reuse)) < 0)
perror("setsockopt(SO_REUSEPORT) failed");

struct sockaddr_in serveraddress;
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(3000);
serveraddress.sin_addr.s_addr = INADDR_ANY;

int binderror = bind (sfd , (struct sockaddr *)&serveraddress , sizeof(serveraddress));
if (binderror == -1)
{
cout<<"Error in binding "<<endl;
return 0;
}
int listenerror = listen(sfd , 10);
if (listenerror == -1)
{
cout<<"Listening error is found "<<endl;
return 0;
}

while (1)
{
struct sockaddr_in clientaddr;
socklen_t len = sizeof(clientaddr);
int newfd = accept(sfd , (struct sockaddr*)&clientaddr , &len);
if (newfd == -1)
{
cout<<"not"<<endl;
continue;
}

char buffer[100];
int n = recv(newfd , buffer , sizeof(buffer)-1 , 0);
```

```
string str (buffer);
stringstream s (str);
string word;

s>>word;
if (word[0]>='0' && word[0]<='9')
{
int sfd = socket(AF_INET , SOCK_STREAM , 0);
if (sfd==-1)
{
cout<<"Error in creating socket"<<endl;
return 1;
}

struct sockaddr_in serveraddress;
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(stoi(word));
serveraddress.sin_addr.s_addr = inet_addr("127.0.0.0");

int t = connect (sfd , (sockaddr *)&serveraddress , sizeof(serveraddress));
if (t==-1)
{
cout<<"Error in connecting "<<endl;
return 1;
}

string remaining = str.substr(word.length()+1);
const char * buf = remaining.c_str();

cout<<"sending it to the live server "<<endl;
send(sfd , buf , strlen(buf)+1 , 0);
cout<<"sended to live server"<<endl;
}
else
{
cout<<"read the news by news: "<<buffer<<endl;
}
close(newfd);
}
}
```

DOCUMENT.cpp

```
#include <bits/stdc++.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
using namespace std;

struct data
{
int low;
int high;
int arr[100];
};

int getSfd()
{
cout<<"My pid : "<<getpid()<<endl;
int usfd=socket(AF_UNIX, SOCK_DGRAM, 0);
struct sockaddr_un un;
un.sun_family = AF_UNIX;
unlink("socketfile");
strcpy(un.sun_path, "socketfile");

if (bind(usfd, (struct sockaddr*)&un, sizeof(un)) < 0) {
perror("usfd bind err");
return 1;
}
char buf[512];
```

```
struct iovec e = {buf, 512};
char cmsg[CMSG_SPACE(sizeof(int))];
struct msghdr m = {NULL, 0, &e, 1, cmsg, sizeof(cmsg), 0};
if(recvmsg(usfd, &m, 0)<0){
perror("usfd recvmsg err");
exit(1);
}
struct cmsghdr *c = CMSG_FIRSTHDR(&m);
int sfd = *(int*)CMSG_DATA(c);
return sfd;
}

int main ()
{
while (1)
{
int sfd = getSfd();

char buffer[100];
int n = read(sfd , buffer , sizeof(buffer)-1);
buffer[n] = '\0';

cout<<"I am document store and i read "<<buffer<<endl;

close(sfd);
}
}
```

REPORTER1.cpp

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

```
#include <arpa/inet.h>
#include <sys/un.h>
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
using namespace std;

int main()
{
while(1)
{
char buff[100];
fgets(buff,100,stdin);
write(1,&buff,sizeof(buff));
fflush(stdout);
}
}
```

REPORTER2.cpp

```
#include <bits/stdc++.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <csignal>
#include <arpa/inet.h>
#include <poll.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
#include <fstream>
#include <sys/wait.h>
#include <sys/shm.h>
using namespace std;

int main ()
{
while (1)
{
sleep(13);
cout<<"d"<<endl;
sleep(0.5);
cout<<"This is the message i am sending for docuemnt"<<endl;
fflush(stdout);
}
}
```