CSE 244
SİSTEM PROGRAMLAMA
FİNAL PROJESİ RAPORU

Büşra ARSLAN
131044021

Öğr. Üye : Prof.Dr. Erkan ZERGEROĞLU
Teslim Tarihi : 29 Mayıs 2016

Gebze Teknik Üniversitesi
Çayırova, Kocaeli

Socket için kullanılan dosyalar:

restart.c
restart.h
uici.c
uici.h
uiciname.c
uiciname.h

Bu dosyalardaki kodlar okul kitabından alınmıştır.
Kaynakça:
http://usp.cs.utsa.edu/usp/programs.html

ALINAN PARAMETRELER

./fileShareServer <port_number>
./client <ip_address> <port_number>

Yararlanilan Kaynakçalar

//Description:
//signal icin
kaynak:http://www.csl.mtu.edu/cs4411.choi/www/Resource/signal.pdf
//Kaynakca: restart.c restart.h uici.c uici.h uiciname.c uiciname.h dosyalari
//kitabin kodlarindan yararlandim.

//getcwd() path bulan fonksiyon icin yararlanilan kaynak
//http://www.qnx.com/developers/docs/660/index.jsp?topic=%//2Fcom.qnx.doc.neutrino.lib_ref%2Ftopic%2Fg%2Fgetcwd.html
//ip_address check yararlanilan kaynak:
//http://stackoverflow.com/questions/16971518/inet-pton-with-all-zero-ip-address

//socket yararlanilan kaynak:
////http://www.cs.dartmouth.edu/~campbell/cs50/socketprogramming.html

//DirWalk kodunda yararlanilan kaynak
//https://gist.github.com/semihozkoroglu/737691

fileShareServer

////Librarys//////
```
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <sys/stat.h>
#include <unistd.h>
#include <pthread.h>
#include <fcntl.h>
#include <time.h>
#include <sys/types.h>
#include <signal.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <semaphore.h>
#include <sys/msg.h>
#include <sys/sem.h>
#include <sys/socket.h>   //socket
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#include <limits.h>
#include <ctype.h>
#include <netinet/in.h>
```

//Kullanilan diger c dosyalari
```
#include "restart.h"
#include "uici.h"
#include "uiciname.h"
```

////Defines///////////////
```
#define MAXLINE 4096 /*max text line length*/
#define LISTENQ 50 /*maximum number of client connections*/
```

```
#define MAXSIZE 100000
#define MAX_PATH_SIZE 1024
#define Size 1000
```

Struct Yapıları

```
struct ClientServer{
    pid_t client_pid;
    int choose;    //menu secimi
    int time;
};
```

//Global Variables

```
struct ClientServer CS;
pid_t server_pid;
pid_t arr_client_pid[LISTENQ];    //signal de tum clientlari kill yapmak icin
char arr_client_path[LISTENQ][LISTENQ];   //for sendFile
int arr_client_size=0;            //arr_client_pid 'nin size'ini tuttum.
int arr_client_path_size=0;
char filename[MAX_PATH_SIZE];    //sendFile
char temp_arr[MAXSIZE];          //for sendFile
struct timespec start, stop;
char client_path[PATH_MAX + 1];         //for sendFile
char temp_arr2[MAXSIZE];   //for sendFile
```

//function prototype

```
void signal_callback_handler(int signum);
void listServer();
void logFileWrite(void);
void logFileWrite(void);
```

client

//////Librarys///////

```
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <sys/stat.h>
#include <unistd.h>
#include <pthread.h>
#include <fcntl.h>
#include <time.h>
#include <sys/types.h>
#include <signal.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <semaphore.h>
#include <sys/msg.h>
#include <sys/sem.h>
#include <sys/socket.h> //socket
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#include <limits.h>
#include <ctype.h>
#include <arpa/inet.h>
#include <netinet/in.h>

//Kullanilan diger c dosyalari
#include "restart.h"
#include "uici.h"
#include "uiciname.h"
```

////Defines///////////////

```
#define MAXLINE 4096 /*max text line length*/
#define LISTENQ 50 /*maximum number of client connections*/
```

```
#define MAXSIZE 100000
#define MAX_PATH_SIZE 1024
#define Size 1000

//Struct
struct ClientServer{
    pid_t client_pid;
    int choose;     //menu secimi
    int time;
};

//global degiskenler
sstruct ClientServer CS;
pid_t arr_client_pid[LISTENQ];  //signal de tum clientlari kill yapmak icin
int arr_client_size=0;      //arr_client_pid 'nin size'ini tuttum.
char backup[Size][Size];      //sendFile
int status1;

char filename[MAX_PATH_SIZE];  //sendFile
char temp_arr[MAXSIZE];   //for sendFile
char temp_arr2[MAXSIZE];   //for sendFile


//////////Function Prototype//////////////////
ctrl-c komutu icin fonksiyon. Log dosyalarına da burada yazılır.
>void signal_callback_handler(int signum);
clientın bulundugu yerdeki dosyalari client terminalinde ekrana basar
>void listLocal();
menu hakkında bilgi verir
>void help();
ctrl-c komutunun icinde çağırılır bu fonksiyon
void logFileWrite(void);
```

PROGRAMIN ÇALIŞMA PRENSİBİ

1)Program başladığında bir menü bölümü açılır. Menü de client yada server ölmediği sürece menü bitmez.
2)clienta istenen 5 program icin sayi girilerek secim yapilir.
3)client fileShareServer arasindaki baglanti socket ile sağlanmaktadir.
4)send() ve recv() komutları ile client server arasinda bilgi alışverişi yapılmıştır.
5)ctrl-C komutu geldiğinde Server tüm clientları öldürür ve kendini de öldürüp çıkar.
6)ctrl-C komutu clienta geldiğinde sadece kendini öldürür.
7)client ve server , pid ile bağlanti sağlandığında pid isimleri ile .log dosyalari oluşturur.
8) server'da fork yapildi. Her gelen clienti multiple olarak aliyor.
9)Birden fazla client bağlantısı sağlandı.

YAPILANLAR

ListLocal

Ödevlerimde yaptığım path bulma fonksiyonunu kullandim. Bu fonksiyonda fork yapılarak aranan directory içerisindeki dosyalar bir arraye kopyalandi. Hangi clientta çalışıyorsa o clientin bulunduğu path üzerinde bulunan tüm dosyaları ekrana basar.Dosyaları path uzantıları ile birlikte verir.

Listserver

Ödevlerimde yaptığım path bulma fonksiyonunu kullandim. Bu fonksiyonda fork yapılarak aranan directory içerisindeki dosyalar bir arraye kopyalandi. listServer serverin kendi bulunduğu path üzerindeki tum pathleri server terminalinde ekrana basar.

LsClients

client pid lerini alır ve servera gönderir. Server bunları bir arraye atar. Sonra ctrl-C komutu geldiğinde bu client_pid arrayini öldürür.
Sonra bu arrayi lsClients komutu geldiğinde clientlarin ekranina basar.

SendFile

Bu bölümde ufak bir menü daha oluşur.
[1]<filename>
[2]<filename> <clientid> şeklinde.
>Birincide Client dosya ismine bakar bulunduğu path üzerinde bu dosya varsa bu dosyanın içeriğini ve dosya adını servera gönderir. Server bu dosya adında kendi pathi üzerinde bir dosya açar ve aldığı dosya içerikli arrayi bu dosyaya yazar. Böylece çalıştığı clienttaki dosyayı kendi pathi üzerne aktarmış olur.Dosya transferi sağlanır.
>İkinci komutta clientlar arasinda clientip'si alarak sonra bağlanan clienttan daha önce bağlanan clienta içinde bulunan bir dosya aktarımı yapıldı. Clienttan send ve recv komutlarıyla alınan filename ve clientip'si ile aktarım başarıyla gerçekleştirildi. client_path arrayi tutuldu ve client_ip arrayi tutuldu bunların ortak indexlerinde o ip adresinin path de vardır diye düşünülerek ortak bağdaşım kuruldu.

help

Menüler hakkında bilgi içerigini client ekranında gösterir.

DİĞER YAPILANLAR

Timer

clientin bağlandığı ve komut aldığı an arasında gecen zamanı ekrana bastırdım.

>client ile server arasi bulunan yerlere connection sağlandı şeklinde kısa notlar düşüldü.