2022 CAT-CERT 스터디 보고서

컴퓨터정보공학부 202221168 김지혜

2022.05.23#C\_study-8

실습-c드라이브에 tmp.txt를 읽는 프로그램을 구현하라

소스코드

#include<stdio.h>

int main() {

FILE\* fp = fopen("C:\\text/tmp.txt", "r");

char str[30] = { 0, };

fgets(str, 30, fp);

printf("%s", str);

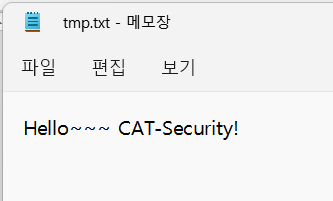
fclose(fp);

}

출력 결과

텍스트이(가) 표시된 사진

자동 생성된 설명



#실습과제 1

친구관리프로그램 추가 기능

파일 입출력을 사용하여 구현하기

소스코드

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  typedef struct \_friend {  char\* name;  char\* message;  int status;  }friend;  friend\* list[100];  int index;  int Addfriend();  int Delfriend();  int Viewfriend();  int Allfriend();  int Savefriend();  int Loadfriend();  int main() {  int num = 0;  while (1) {  friend\* frd = 0;  printf("==========Friend Menu==========\n");  printf("\t1. Add friend\n");  printf("\t2. Delete friend\n");  printf("\t3. View friend\n");  printf("\t4. All friend\n");  printf("\t5. Save friend\n");  printf("\t6. Load friend\n");  printf("\t7. Exit\n");  printf("===============================\n");  printf("Choose menu:");  scanf("%d", &num);  if (num == 1)  Addfriend();  else if (num == 2)  Delfriend();  else if (num == 3)  Viewfriend();  else if (num == 4)  Allfriend();  else if (num == 5)  Savefriend();  else if (num == 6)  Loadfriend();  else {  exit(0);  }  }  }  int Addfriend() {  int size = 0;  int msize = 0;  friend\* frd = 0;  int len;  len = strlen(list);  frd = (friend\*)malloc(sizeof(friend)); //friend자료형크기만큼 할당  printf("name size:");  scanf("%d", &size);  frd->name = (char\*)malloc(size); //name사이즈만큼 할당  printf("name: ");  scanf("%s", (\*frd).name);  printf("message size:");  scanf("%d", &msize);  frd->message = (char\*)malloc(msize);  printf("message :");  scanf("%s", (\*frd).message);  printf("status (put 0(:offline) or 1(:away) or 2(:online):");  scanf("%d", &(\*frd).status);  printf("\n");  list[index] = frd;  index++;  }  int Delfriend() {  printf("삭제할 친구의 인덱스번호를 적어주세요!:");  scanf("%d", &index);  free(list[index]);  list[index] = NULL;  printf("삭제되었습니다\n");  printf("\n");  }  int Viewfriend() {  char search[100];  printf("검색할 친구의 이름을 적어주세요! : ");  scanf("%s", &search);  for (int i = 0; list[i] != '\x00'; i++) {  if (list[i]->name != NULL) {  if (strcmp(search, list[i]->name) == 0) {  printf("MY FRIEND!!\n");  printf("name : %s\n", (\*list[i]).name);  printf("message : %s\n", (\*list[i]).message);  if ((\*list[i]).status == 0)  printf("status : offline\n");  if ((\*list[i]).status == 1)  printf("status : away\n");  if ((\*list[i]).status == 2)  printf("status : online\n");  printf("\n");  }  }  }  }  int Allfriend() {  for (int i = 0; i < 100; i++) {  if (list[i] != NULL) {  printf("name : %s\n", list[i]->name);  printf("message : %s\n", list[i]->message);  if ((\*list[i]).status == 0)  printf("status : offline\n");  if ((\*list[i]).status == 1)  printf("status : away\n");  if ((\*list[i]).status == 2)  printf("status : online\n");  printf("\n");  }  }  }  int Savefriend() {  FILE\* fp = fopen("friend.txt", "w");  for (int i = 0; i < 100; i++) {  if (list[i] != NULL) {  fprintf(fp, "%s ", list[i]->name);  fprintf(fp, "%s ", list[i]->message);  if ((\*list[i]).status == 0)  fprintf(fp, "offline\n");  if ((\*list[i]).status == 1)  fprintf(fp, "away\n");  if ((\*list[i]).status == 2)  fprintf(fp, "online\n");  }  }  fclose(fp);  }  int Loadfriend() {  FILE\* fp = fopen("friend.txt", "rt");  char namee[100];  char messagee[100];  char statuse[100];  while (!feof(fp)) {  fscanf(fp, "%s %s %s", &namee, &messagee, &statuse);  printf("%s %s %s\n", namee, messagee, statuse);  }  fclose(fp);  } |

출력결과

텍스트이(가) 표시된 사진

자동 생성된 설명

조사과제

#1 파일 디스크립터와 파일 포인터 개념과 차이점

|  |  |  |
| --- | --- | --- |
|  | 파일 디스크립터 | 파일 포인터 |
| 개념 | 시스템으로부터 할당받은 파일 또는 소켓에 부여된 정수  (파일과 소켓은 함수 호출과 같은 생성과정을 거쳐야 할당됨)  OS별로 특별한 기능(보안,압축)을 사용하고자 한다면 파일디스크립터를 쓰는 게 용이할 것임  파일을 제어하기 위한 번호 | FILE\*타임의 포인터 변수  Cbywns IO라이브러리에서 사용하는 IO개체, 스트림기반 동작  파일포인터는 C표준에서 제정한 파일디스크립터의 래퍼자료구조 |
| 차이점 | OS에서 제공하는 입출력 식별자  ->특정한 플랫폼에서만 동작함 | C라는 언어에서 제정한 표준입출력 객체  ->어느 플랫폼에서든 동일한 동작이 이루어질것임 |

->파일포인터가 파일디스크립터를 포함함

#2 네트워크 포트 개념

포트: 논리적인 접속 장소

**잘 알려진 포트**

: 포트번호 0 ~ 1023번 까지를 의미합니다. 또한 공식으로 지정되어 있는 것도 있고 아닌 것도 있음. 잘 알려진 포트인 이유는 사람들이 잘 알고 있기도 하고 이미 알려져 있는 것들이 많기 때문임

-> HTTP는 80번, HTTPS는 443 포트로 접속

**등록된 포트:**

포트번호 1024 ~ 49151번은 등록된 포트입니다. 등록된 포트는 사용자가 직접 등록할 수 있는 포트를 의미하며 등록해서 사용할 수 있음

**동적 포트:**

포트번호 49152 ~ 65535 번은 동적 포트입니다. 이것은 수시로 변경되는 포트를 의미하며 인터넷이나 시스템에서 사용할 때 동적으로 할당하는 포트를 의미함. 알아서 포트를 할당하며 windows에서는 netstat 명령어로 확인해보면 이러한 동적 포트들이 사용되고 있는 것을 볼 수 있음. 등록된 포트(registered port)와는 다르게 지정하지 않아도 알아서 사용할 수 있다는 것을 알 수 있음

#3 TCP vs UDP

전송계층: 송신자와 수신자를 연결하는 통신서비스를 제공하는 계층(데이터 전달 담당)

TCP,UDP=>데이터를 보내기 위해 사용하는 프로토콜

|  |  |
| --- | --- |
| TCP | UDP |
| 연결형 서비스를 지원하는 프로토콜, 인터넷환경에서 기본으로 사용함  IP랑 함께 사용하는데 IP가 데이터 배달을 처리하면 TCP는 패킷 추적 및 관리   1. 연결형서비스, 가상회선방식   ->발신지와 수신지를 연결하여 패킷을 전송하기 위한 논리적 경로를 배정한다는 말   1. 3way-handshaking과정을 통해 연결을 설정하고 4way-handshaking을 통해 해제함 2. 흐름 및 혼잡을 제어하고 높은신뢰성 보장 3. UDP보다 속도가느림(CPU가 사용하기 때문에 속도를 영향을줌)   ->패킷에 대한 응답을 해야하기 때문에 성능이 낮음   1. 서버와 클라이언트는 1대1로 연결 2. 스트림전송으로 전송데이터의 크기가 무제한 3. Streaming 서비스에 불리함(손실된 경우 재전송요청) | 데이터를 데이터그램단위로 처리하는 프로토콜  ->데이터그램(독립적인 관계를 지닌패킷)  TCP와 달리 비연결형 프로토콜로 연결을 위해 할당되는 논리적인 경로가 없어 다른 경로로 전송되고 패킷은 독립적인 관계를 지니게 됨   1. 비연결형 서비스로 데이터그램방식 제공 2. 정보를 주고받을 때 정보를 보내거나 받는 신호절차X 3. 최소한의 오류만 검출하고 신뢰성이 낮음 4. TCP보다 속도가 빠름(패킷에 순서를 부여하여 재조립, 흐릅제어 기능처리하지 않음)   -> 네트워크 부하가 적음   1. 소켓대신 IP기반으로 데이터를 전송함, 연결자체가 없어서 서버와 클라이언트 구분이 없음 2. 흐름제어가 없어서 패킷이 제대로 전송디었는지 오류가 없는지 확인할 수 없음 3. 신뢰성보다 성능이 중요시 되는 경우에 사용 |

패킷:데이터를 보내기 위한 경로배정을 효율적으로 하기위해 데이터를 조각으로 나눠서 전달하는 데 그 조각을 패킷이라고 함

#4 검색을 통해 소켓통신 예시코드 완성, 코드 분석

(주석 꼼꼼하게 달기)

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

클라이언트, 서버 통신 실행 캡쳐

+netstat명령어 조사 후 사용하여 소켓통신 확인

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Netstat : 자신의 컴퓨터와 연결된 모든 네트워크 연결을 보여주는 명령어

Netstat

-a : 모든 포트를 표시해줌

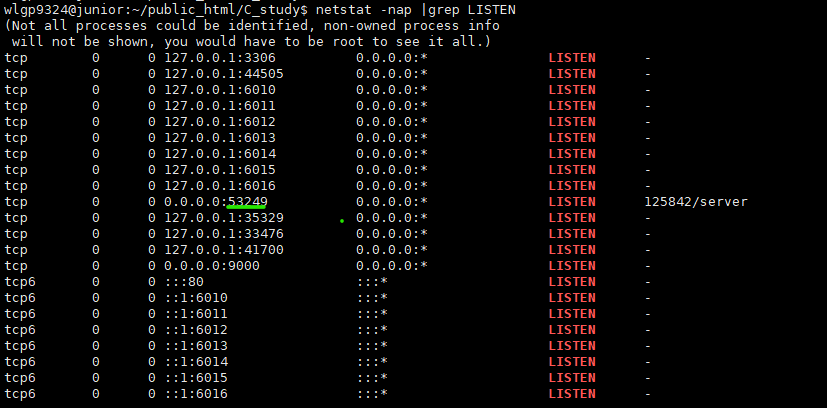
-n : “IP주소: 포트” 형태로 보여줌

-o : 프로세스ID를 표시해줌

-ntlp : tcp정보 중 listening 상태인 소켓 정보를 10진수로 변환하여 보여주며 실행되고 있는 프로그램과 PID정보 출력

**netstat -nap | grep LISTEN**

**: 현재 listen 중인 포트 표시**

****