#include "stdafx.h"

#include <windows.h>

#include <stdio.h>

#include <conio.h>

#ifdef \_UNICODE

typedef wchar\_t TCHAR;

#else

typedef char TCHAR;

#endif

void menu()

{

printf("1.Add elem\n");

printf("2.Pre Order Display\n");

printf("3.Post Order Display\n");

printf("4.In Order Display\n");

printf("5.Delete node\n");

printf("6.Search node\n");

printf("7.Exit\n");}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int choise=0;

while(choise==0)

{printf("Enter choise (0-menu): ");

scanf ("%d", &choise);

if(choise==0)

{

menu();

}}

HANDLE Daddy;

LPCSTR AdresNameFile = ("\\\\.\\pipe\\MyPipe");

//LPWSTR AdresNameFile =TEXT ("C:\\Users\\Arkadii\\Desktop\\OS\\Data.txt");

Daddy = CreateNamedPipe(AdresNameFile, PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT,

1, 128, 128,

PIPE\_UNLIMITED\_INSTANCES,

NULL);

if (Daddy == INVALID\_HANDLE\_VALUE)

{

printf("Daddy's PIPE doesn't createeeeeeeeee");

printf(" NNNNNNOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO");

system("pause");

return GetLastError();

}

PROCESS\_INFORMATION ProcessInfo; //This is what we get as an [out] parameter

ZeroMemory(&ProcessInfo, sizeof(PROCESS\_INFORMATION));

STARTUPINFO StartupInfo; //This is an [in] parameter

ZeroMemory(&StartupInfo, sizeof(StartupInfo));

StartupInfo.cb = sizeof(STARTUPINFO); //Only compulsory field

bool process = CreateProcess("C:\\Users\\L2(client).exe",

NULL,

NULL, NULL, true,

CREATE\_NEW\_CONSOLE,// CREATE\_NEW\_CONSOLE|CREATE\_SUSPENDED

NULL, NULL,

&StartupInfo,

&ProcessInfo);

WaitForSingleObject(ProcessInfo.hProcess, 5);

bool connect = ConnectNamedPipe(Daddy, NULL);

if (connect == false)

{

int i = GetLastError();

if (i == 997 || i == 536)

{

}

else {

return GetLastError();

};

};

while (choise!=7){

DWORD written;

if (!WriteFile(Daddy, &choise, sizeof(int), &written, NULL))

{

printf("Error! Can not write in file\n");

system("pause");

return GetLastError();

}

// WaitForSingleObject(ProcessInfo.hProcess,2);

Sleep(1000);

ReadFile(Daddy, &choise,sizeof(int), &written , NULL);

printf("\nAnswer = %d\n", choise);

printf("Enter choise (0-menu): ");

scanf ("%d", &choise);

if(choise==0)

{

menu();

printf("Enter choise (0-menu): ");

scanf ("%d", &choise);

}

if(choise==7)

{

WriteFile(Daddy, &choise, sizeof(int), &written, NULL);

}

}

CloseHandle(Daddy);

system("pause");

return 0;

}

#include "stdafx.h"

#include <windows.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <dos.h>

#include "tree.c"

typedef struct bin\_tree node;

node\* dummy\_tree()

{

node \*root= NULL;

insert(&root, 2);

insert(&root, 4);

insert(&root, 3);

insert(&root, 6);

insert(&root, 1);

insert(&root, 7);

insert(&root, 2);

insert(&root, 7);

insert(&root, 1);

return root;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i;

node \*root=NULL;

node \*tmp=NULL;

root=dummy\_tree();

int choise=0;

HANDLE Son;

LPCSTR AdresNameFile = "\\\\.\\pipe\\MyPipe";

Sleep (12);

Son = CreateFile(AdresNameFile, GENERIC\_READ | GENERIC\_WRITE , 0, NULL, OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL, NULL);

if (Son == INVALID\_HANDLE\_VALUE)

{

printf("I'm Son.\n Daddy doesn't createeeeeeeeee");

getchar();

printf("NNNNNNOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO");

return GetLastError();

};

// bool connect = ConnectNamedPipe(Son, &l);

/\*if (connect == false)

{

int i = GetLastError();

if (i == 997 || i == 536)

{

}

else {

return GetLastError();

};

};\*/

DWORD in;

while(choise!=7)

{

if (!ReadFile(Son, &choise,sizeof(int), &in , NULL))

/\*{

printf("I CAN'T READ FILE");

return GetLastError();

};\*/

/\*Son = CreateFile(AdresNameFile, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, TRUNCATE\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (Son == INVALID\_HANDLE\_VALUE)

{

printf("Son doesn't open file");

getchar();

printf("NNNNNNOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO");

ExitProcess(1);

return 1;

};\*/

printf("%d", choise);

if(choise==1)

{

printf("Enter elem\n");

scanf("%d", &i);

insert(&root, i);

print\_inorder(root);

printf("Elem in tree\n");

}

if(choise==2)

{

printf("Pre Order Display\n");

print\_preorder(root);

}

if(choise==3)

{

printf("Post Order Display\n");

print\_postorder(root);

}

if(choise==4)

{

printf("In Order Display\n");

print\_inorder(root);

}

if(choise==6)

{

printf("Search node. Enter one\n");

scanf("%d", &i);

tmp = find(root, i);

if (tmp)

{

printf("Searched node=%d\n", tmp->data);

}

else

{

printf("Data Not found in tree.\n");

}

}

if(choise==5)

{

printf("enter node value\n");

scanf("%d", &i);

tree\_remove(&root, i);

}

if(choise>8 && choise<1)

{printf("wrong choise");

}

DWORD write;

WriteFile(Son, &choise, sizeof(int), &write, NULL);

if (Son == INVALID\_HANDLE\_VALUE)

{

printf("Error! Can not write data in file\n");

ExitProcess(1);

return 1;

}

Sleep(100);

}

if(choise==7)

{

printf("Exit\n");

// char Out[20];

// char \*PtrToOut;

// PtrToOut = Out;

// wsprintf(PtrToOut, "%d", SUM);

// WriteFile(Son, PtrToOut, 15, 0, NULL);

// if (Son == INVALID\_HANDLE\_VALUE)

// {

// printf("Error! Can not write data in file\n");

// ExitProcess(1);

// return 1;

// };

ExitProcess(1);

return 0;

}

ExitProcess(1);

return 0;

}

LPCVOID Error(int i)

{

LPCVOID Errors;

if (i == 1)

{

Errors = TEXT("you bad write \n");

return Errors;

};

return 0;

};



