**University of Prishtina**

Faculty of Electrical and Computer Engineering

Kodra e Diellit, p.n.

10000 - Prishtinë, Kosova

PROGRAMI PËR KOMUNIKIM MES PROCESEVE PËRMES METODËS NAMED PIPES NE WINDOWS

Punuan: Eltion Musa, Profesori: Msc. Artan Mazrekaj

Genc Balaj.

Janar, 2020

Përmbajtja

[PERSHKRIMI I PROBLEMIT 2](#_Toc29412348)

[TEKNOLOGJIA 3](#_Toc29412349)

[3.IMPLEMENTIMI 4](#_Toc29412350)

[3.1 Paraqitja vizuale e programit 5](#_Toc29412351)

[3.2 FUNKSIONET KRYESORE TE PERDORURA 5](#_Toc29412352)

[3.2.1 CreateNamedPipe 6](#_Toc29412353)

[3.2.2 CreateFile 6](#_Toc29412354)

[3.2.3 WriteFile 7](#_Toc29412355)

[3.2.4 ReadFile 7](#_Toc29412356)

[3.3 Implemenitimi i programit ne kod 8](#_Toc29412357)

[3.3.1 Procesi A 8](#_Toc29412358)

[3.3.2 Procesi B 13](#_Toc29412359)

[REFERENCAT 18](#_Toc29412360)

# PERSHKRIMI I PROBLEMIT

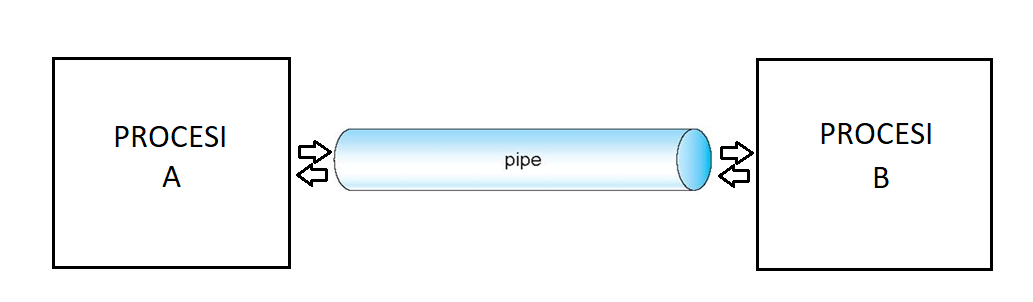
Përmes platformës Windows është mundësuar komunikimi mes proceseve duke shfrytëzuar metodën e ashtuquajtur “Named Pipes”.

Një Named Pipe mundëson komunikimin mes proceseve, i cili mund te jetë i dyanshëm, full-duplex dhe nuk nevojitet marrëdhënia prind-fëmijë mes proceseve.

Pas krijimit te një Named Pipe ai mund te shfrytëzohet për komunikim nga shumë procese njëkohësisht. Named Pipes vazhdojnë te ekzistojnë edhe pasi që komunikimi mes dy proceseve të ketë përfunduar.

Named Pipes përkrahet nga Windows dhe UNIX sistemet mirëpo implementimi i tyre ne këto sisteme është i ndryshëm.

Për dallim nga UNIX-i, Windows mundëson komunikim më të mirë sepse përkrahë komunikimin Full-Duplex dhe lejohet komunikimi message-oriented krahas atij byte-oriented.



# TEKNOLOGJIA

Gjuha Programuese : C/C++,

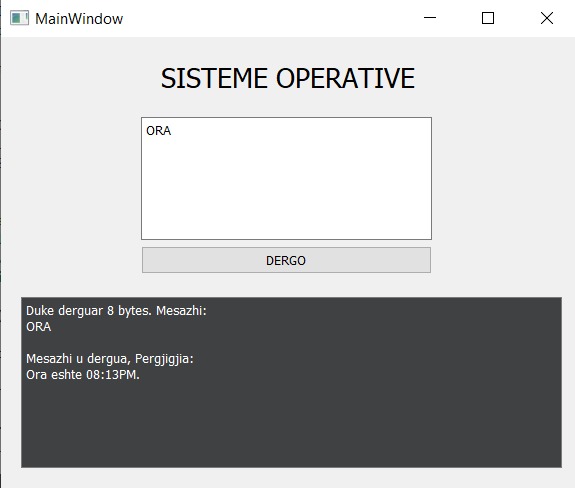
Editori : QT Creator 4.11.0

Kompailer : MinGW 7.3.0

Platforma : Microsoft Windows 10.

# IMPLEMENTIMI

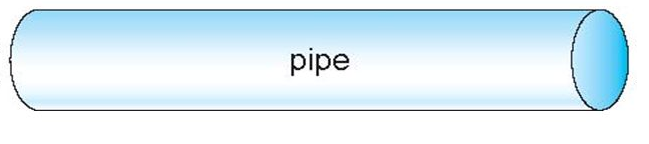
## 3.1 Paraqitja vizuale e programit



KOHA

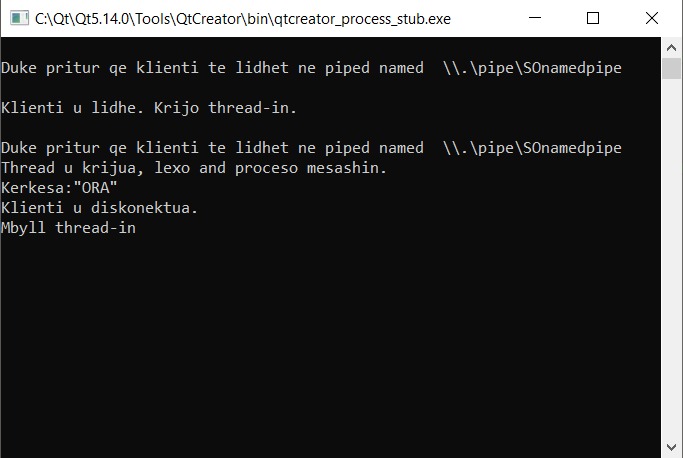
08:13PM

Lidhet me pipe



Krijon pipe-in

KOHA



08:13PM

## 3.2 FUNKSIONET KRYESORE TE PERDORURA

Per implementimin e programit janë përdorur funksionet nga libraria e Windows-it ne C

### 3.2.1 CreateNamedPipe

**HANDLE** **CreateNamedPipe**(

**LPCSTR**                lpName,

**DWORD**                 dwOpenMode,

**DWORD**                 dwPipeMode,

**DWORD**                 nMaxInstances,

**DWORD**                 nOutBufferSize,

**DWORD**                 nInBufferSize,

**DWORD**                 nDefaultTimeOut,

**LPSECURITY\_ATTRIBUTES** lpSecurityAttributes

);

lpName – Emri unik i pipe.

\.\pipe<i>pipename

Psh. \.\pipe\SOnamedpipe

dwOpenMode – modi i qasjes ne pipe

**PIPE\_ACCESS\_DUPLEX**

**PIPE\_ACCESS\_INBOUND**

**PIPE\_ACCESS\_OUTBOUND**

dwPipeMode – Modi I mesazhit te pipe

**PIPE\_TYPE\_BYTE**

**PIPE\_TYPE\_MESSAGE**

nMaxInstances – Numri maksimal i instancave qe mund te krijohen per pipe

NULL – I pa limituar

nOutBufferSize – Numri I bajtave per output buffer

nInBufferSize – Numri I bajtave per input buffer

nDefaultTimeOut – Koha per sa pipe mund te jete I zene

### 3.2.2. CreateFile

Përdoret per tu konektuar ne named pipe

**HANDLE** **CreateFile**(

**LPCSTR**                lpFileName,

**DWORD**                 dwDesiredAccess,

**DWORD**                 dwShareMode,

**LPSECURITY\_ATTRIBUTES** lpSecurityAttributes,

**DWORD**                 dwCreationDisposition,

**DWORD**                 dwFlagsAndAttributes,

**HANDLE**                hTemplateFile

);

lpFileName – Emri i pipe ku dëshirojmë te konektohemi

dwDesiredAccess – Qasja ne pipe

**GENERIC\_READ**

**GENERIC\_WRITE**

lpSecurityAttributes – Atributet e sigurisë

### 3.2.3 WriteFile

**BOOL** **WriteFile**(

**HANDLE**       hFile,

**LPCVOID**      lpBuffer,

**DWORD**        nNumberOfBytesToWrite,

**LPDWORD**      lpNumberOfBytesWritten,

**LPOVERLAPPED** lpOverlapped

);

hFile – Një hadle per pipe

lpBuffer – Pointer qe pointon tek buffer-i i te dhënave qe do te shkruhen ne pipe.

nNumberOfBytesToWrite – Numri i bajtave qe do te shkruhen ne pipe.

lpNumberOfBytesWritten – Numri i bajtave te shkruar ne pipe.

### 3.2.4 ReadFile

**BOOL** **ReadFile**(

**HANDLE**       hFile,

**LPVOID**       lpBuffer,

**DWORD**        nNumberOfBytesToRead,

**LPDWORD**      lpNumberOfBytesRead,

**LPOVERLAPPED** lpOverlapped

);

hFile – Një hadle per pipe

lpBuffer – Pointer qe pointon tek buffer-i ku ruhen te dhenat e lexuara nga pipe.

nNumberOfBytesToRead – Numri i bajtave qe do te lexohen nga pipe.

lpNumberOfBytesRead – Numri i bajtave te lexuara nga pipe.

## 3.3 Implementimi i programit ne kod

### 3.3.1 Procesi A

#include <windows.h>

#include <stdio.h>

#include <tchar.h>

#include <strsafe.h>

#include <string>

#include <vector>

#include <time.h>

using namespace **std**;

#define **BUFSIZE** 512

DWORD WINAPI **InstanceThread**(LPVOID);

**VOID** **GetAnswerToRequest**(**LPTSTR**, **LPTSTR**, **LPDWORD**);

int **main**()

{

   BOOL   fConnected = FALSE;

   DWORD  dwThreadId = 0;

   HANDLE hPipe = INVALID\_HANDLE\_VALUE, hThread = NULL;

   LPCTSTR lpszPipename = **TEXT**("\\\\.\\pipe\\SOnamedpipe");

*//Loop-a krijon ne neamed pipe dhe pret qe klienti te konektohet*

*//Kur klienti konektohet nje thread krjohet per te kryer kumunikimi*

*//me te. Me pas threadi kryen analizen e kerkeses dhe shkruan ne buffer*

*//pergjigjen e kerkeses derisa programi kryesore pret per klient te tjere*

   for (;;)

   {

**\_tprintf**( **TEXT**("\nDuke pritur qe klienti te lidhet ne piped named  %s\n"), lpszPipename);

       hPipe = **CreateNamedPipe**(

           lpszPipename,*// Emri i pipe-it*

           PIPE\_ACCESS\_DUPLEX,*// lexo/shkruaj acessi*

           PIPE\_TYPE\_MESSAGE |*// lloji i komunikimit ne pipe PIPE\_TYPE\_BYTE/PIPE\_TYPE\_MESSAGE*

           PIPE\_READMODE\_MESSAGE |*// message-read modi*

           PIPE\_WAIT,*// modi i bllokimit*

           PIPE\_UNLIMITED\_INSTANCES,*// numri maksimal i instancave*

           BUFSIZE,*// madhesia e output buffer-it*

           BUFSIZE,*// madhesia e input buffer-it*

           0,*// time-out*

           NULL);

      if (hPipe == INVALID\_HANDLE\_VALUE)

      {

**\_tprintf**(**TEXT**("CreateNamedPipe deshtoje, Kodi=%d.\n"), **GetLastError**());

          return false;

      }

*//Prit deri sa te konektohet klienti*

      fConnected = **ConnectNamedPipe**(hPipe, NULL) ?

         TRUE : (**GetLastError**() == ERROR\_PIPE\_CONNECTED);

      if (fConnected)

      {

**printf**("\nKlienti u lidhe. Krijo thread-in.\n");

*// Krijon thread-in per klientin*

         hThread = **CreateThread**(

            NULL,*// attributet e siguris*

            0,*// madhsia e steku-t*

            InstanceThread,*// funksioni i thredit*

            (LPVOID) hPipe,*// parametrat e funksionit*

            0,*// suspended*

            &dwThreadId);*// kthen Id e thread-it*

         if (hThread == NULL)

         {

**\_tprintf**(**TEXT**("Krijimi i thread-it deshtoj, Kodi=%d.\n"), **GetLastError**());

            return false;

         }

         else **CloseHandle**(hThread);

       }

      else

*// Mbyll pipe-in nese klienti nuk ka mundur te lidhet*

**CloseHandle**(hPipe);

   }

   return true;

}

*//Funksioni i thread-it*

DWORD WINAPI **InstanceThread**(LPVOID lpvParam)

{

   HANDLE hHeap      = **GetProcessHeap**();

   TCHAR\* pchRequest = (TCHAR\*)**HeapAlloc**(hHeap, 0, BUFSIZE\*sizeof(TCHAR));

   TCHAR\* pchReply   = (TCHAR\*)**HeapAlloc**(hHeap, 0, BUFSIZE\*sizeof(TCHAR));

   DWORD cbBytesRead = 0, cbReplyBytes = 0, cbWritten = 0;

   BOOL fSuccess = FALSE;

   HANDLE hPipe  = NULL;

**printf**("Thread u krijua, lexo and proceso mesashin.\n");

*// The thread's parameter is a handle to a pipe object instance.*

   hPipe = (HANDLE) lpvParam;

*// Lexo te dhenat nga pipe deri ne fund*

   while (1)

   {

      fSuccess = **ReadFile**(

         hPipe,*// handle-i i pipe*

         pchRequest,*// Bufferi ku ruhen te dhenat e lexuara nga pipe*

         BUFSIZE\*sizeof(TCHAR),*// madhsia e bufferit*

         &cbBytesRead,*// numri i bajtave te lexuar*

         NULL);*// jo overlapped I/O*

      if (!fSuccess || cbBytesRead == 0)

      {

          if (**GetLastError**() == ERROR\_BROKEN\_PIPE)

          {

**\_tprintf**(**TEXT**("Klienti u diskonektua.\n"));

          }

          else

          {

**\_tprintf**(**TEXT**("Leximi deshtoj, Kodi=%d.\n"), **GetLastError**());

          }

          break;

      }

*// Proceso kerkesen*

**\_tprintf**( **TEXT**("Kerkesa:\"%s\"\n"), pchRequest );

**GetAnswerToRequest**(pchRequest, pchReply, &cbReplyBytes);

*// Shkruaj pergjigjjen ne pipe*

      fSuccess = **WriteFile**(

         hPipe,*// handle-i i pipe*

         pchReply,*// Bufferi prej te cilit shkruhet*

         cbReplyBytes,*// numri i bajtave per te shkruar*

         &cbWritten,*// numri i bajtave te shkruar*

         NULL);*// jo overlapped I/O*

      if (!fSuccess || cbReplyBytes != cbWritten)

      {

**\_tprintf**(**TEXT**("Shkrimi deshtoj, Kodi=%d.\n"), **GetLastError**());

          break;

      }

  }

**FlushFileBuffers**(hPipe);

**DisconnectNamedPipe**(hPipe);*//Diskonekto pipe*

**CloseHandle**(hPipe);*//mbylle handle e pipe*

    \

*//Liro bufferin*

**HeapFree**(hHeap, 0, pchRequest);

**HeapFree**(hHeap, 0, pchReply);

**printf**("Mbyll thread-in\n");

   return 1;

}

**std**::**vector**<**std**::**wstring**> **spitWstring**(**std**::**wstring** a);

**wstring** **getTime**();

*//Bene procesimin e kerkeses se klientit dhe vendos pergjigjjen ne buffer*

**VOID** **GetAnswerToRequest**( **LPTSTR** pchRequest, **LPTSTR** pchReply, **LPDWORD** pchBytes ) {

**std**::wstring kerkesa = **std**::**wstring**(pchRequest);

    vector<wstring> kerkesaArray= **spitWstring**(kerkesa);

    wstring c = L"KATROR";

    if(kerkesaArray[0] == L"KATROR"){

*int numri = stoi(kerkesaArray[1]);*

**StringCchCopy**( pchReply, BUFSIZE, **to\_wstring**(numri\*numri).**c\_str**());

    }else if(kerkesaArray[0] == L"ORA"){

        wstring koha = **getTime**();

**StringCchCopy**(pchReply, BUFSIZE, koha.**c\_str**());

    }else{

**StringCchCopy**(pchReply, BUFSIZE, L"OK. Mesazhi u pranua");

    }

*//Madhesia e pergjigjjes*

    \*pchBytes = (**lstrlen**(pchReply)+1)\*sizeof(TCHAR);

}

*//Merr kohen nga sistemi*

**wstring** **getTime**(){

    wchar\_t koha [80];

    time\_t t = **time**(nullptr);

    tm \*timestamp = **localtime**(&t);

**wcsftime** (koha,80,L"Ora eshte %I:%M%p.",timestamp);

    return koha;

}

*//E ndane wstring sipas delimiterit ' '*

**std**::**vector**<**std**::**wstring**> **spitWstring**(**std**::**wstring** a){

**std**::vector<**std**::wstring> b;

**std**::wstring st = L"";

    for (int i = 0; i < a.**length**(); i++) {

        if (a[i] == L' ') {

            b.**push\_back**(st);

        }

        else{

            st += a[i];

        }

    }

    b.**push\_back**(st);

    return b;

}

### 3.3.2 Procesi B

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <Windows.h>

#include <stdio.h>

#include <conio.h>

#include <tchar.h>

#include <QMessageBox>

*//Madhsia e buffer-it*

#define **BUFSIZE** 512

*//GUI*

**MainWindow**::**MainWindow**(**QWidget** \*parent)

    : **QMainWindow**(parent)

    , **ui**(new **Ui**::MainWindow)

{

    ui->**setupUi**(this);

}

**MainWindow**::**~MainWindow**()

{

    delete ui;

}

void **MainWindow**::**on\_pushButton\_clicked**(){

    QMessageBox msgBox;

**SendToPipe**(ui->textEdit->**toPlainText**().**toStdWString**().**c\_str**());

}

void **MainWindow**::**Logger**(**std**::**string** a){

    QString b = **QString**::**fromStdString**(a);

    ui->textEdit\_2->**append**(b);

}

void **MainWindow**::**Logger**(**std**::**wstring** a){

    QString b = **QString**::**fromStdWString**(a);

    ui->textEdit\_2->**append**(b);

}

*//Kumunikimi me pipe*

int **MainWindow**::**SendToPipe**(**LPCTSTR** lpvMessage)

{

   HANDLE hPipe;

   TCHAR  chBuf[BUFSIZE];

   BOOL   fSuccess = FALSE;

   DWORD  cbRead, cbToWrite, cbWritten, dwMode;

   LPCTSTR lpszPipename = **TEXT**("\\\\.\\pipe\\SOnamedpipe");

*// Tenton ta hapë pipe-in*

   while (1)

   {

      hPipe = **CreateFile**(

         lpszPipename,*// emri i pipe-it*

         GENERIC\_READ |*// qasja shkrim/lexim*

         GENERIC\_WRITE,

         0,*// no sharing*

         NULL,*// atributet e sigurisë*

         OPEN\_EXISTING,*// hap pipe-in ekzistues*

         0,*// atributet default*

         NULL);*// file template(jo)*

*// Kontrollo për errors*

      if (hPipe != INVALID\_HANDLE\_VALUE)*//nuk ka error*

         break;

*// Kontrollo nese ka error-a tjerë perpos nese pipe është i zënë*

      if (**GetLastError**() != ERROR\_PIPE\_BUSY)

      {

**Logger**("ERROR: Nuk u ralizua lidhja. Kodi="+**std**::**to\_string**(**GetLastError**()));

         return -1;

      }

*// Pipe është i zënë, prit 20 sekonda.*

      if ( ! **WaitNamedPipe**(lpszPipename, 20000))

      {

**Logger**("Nuk mund te hapet Pipe, Pipe busy. Timeout 20s ");

         return -1;

      }

   }

*// Pipe eshte konektuar. modi per lexim*

   dwMode = PIPE\_READMODE\_MESSAGE;

   fSuccess = **SetNamedPipeHandleState**(

      hPipe,*// pipe handle*

      &dwMode,*// modi i ri i pipe*

      NULL,*// numri maksimal i bajtave*

      NULL);*// koha maksimale*

   if ( ! fSuccess)

   {

**Logger**("Pipe handle dështoj. KODI="+**GetLastError**());

      return -1;

   }

*// Dërgo mesazhin te procesi B.*

   cbToWrite = (**lstrlen**(lpvMessage)+1)\*sizeof(TCHAR);

**Logger**( "Duke derguar "+**std**::**to\_string**(cbToWrite)+" bytes. Mesazhi:");

**Logger**(**std**::**wstring**(lpvMessage));

   fSuccess = **WriteFile**(

      hPipe,*// handle per pipe*

      lpvMessage,*// mesazhi*

      cbToWrite,*// gjatësia e mesazhit*

      &cbWritten,*// bajtat e shkruar*

      NULL);*// jo overlapped*

   if ( ! fSuccess)

   {

**Logger**("Shkrimi ne Pipe deshtoj. Kodi="+ **std**::**to\_string**(**GetLastError**()));

      return -1;

   }

**Logger**("\nMesazhi u dergua, Pergjigjia:");

   do

   {

*// Lexo prej pipe.*

      fSuccess = **ReadFile**(

         hPipe,*// handle i pipe-it*

         chBuf,*// buffer per pranimin e përgjigjes*

         BUFSIZE\*sizeof(TCHAR),*// madhësia e buffer-it*

         &cbRead,*// numri i bajtave te lexuar*

         NULL);*// jo overlapped*

      if ( ! fSuccess && **GetLastError**() != ERROR\_MORE\_DATA )

         break;

**Logger**( chBuf );

   } while ( ! fSuccess);*// perserit loop nese ka te dhena me shume*

   if ( ! fSuccess)

   {

**\_tprintf**( **TEXT**("Leximi nga Pipe deshtoj. Kodi=%d\n"), **GetLastError**() );

      return -1;

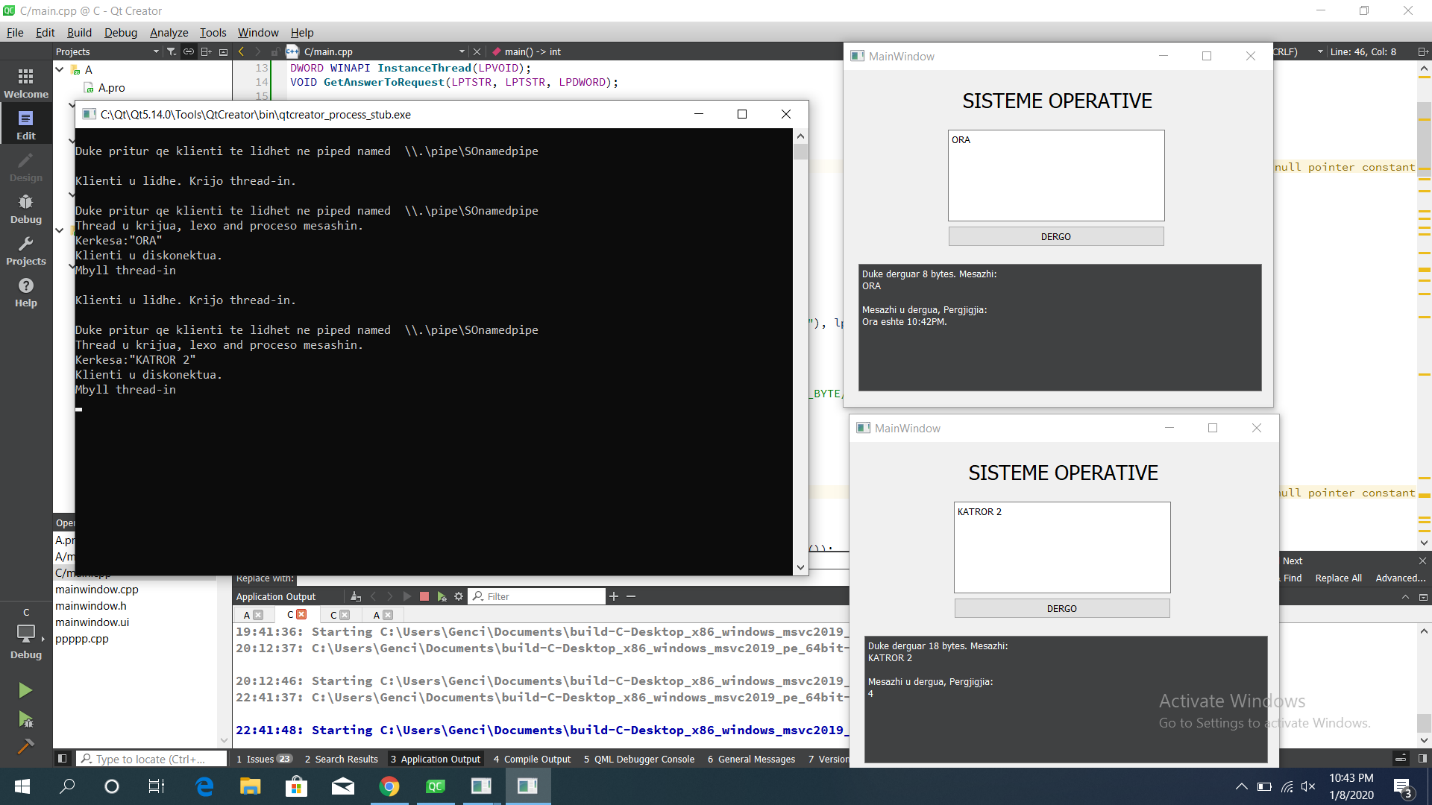
   }

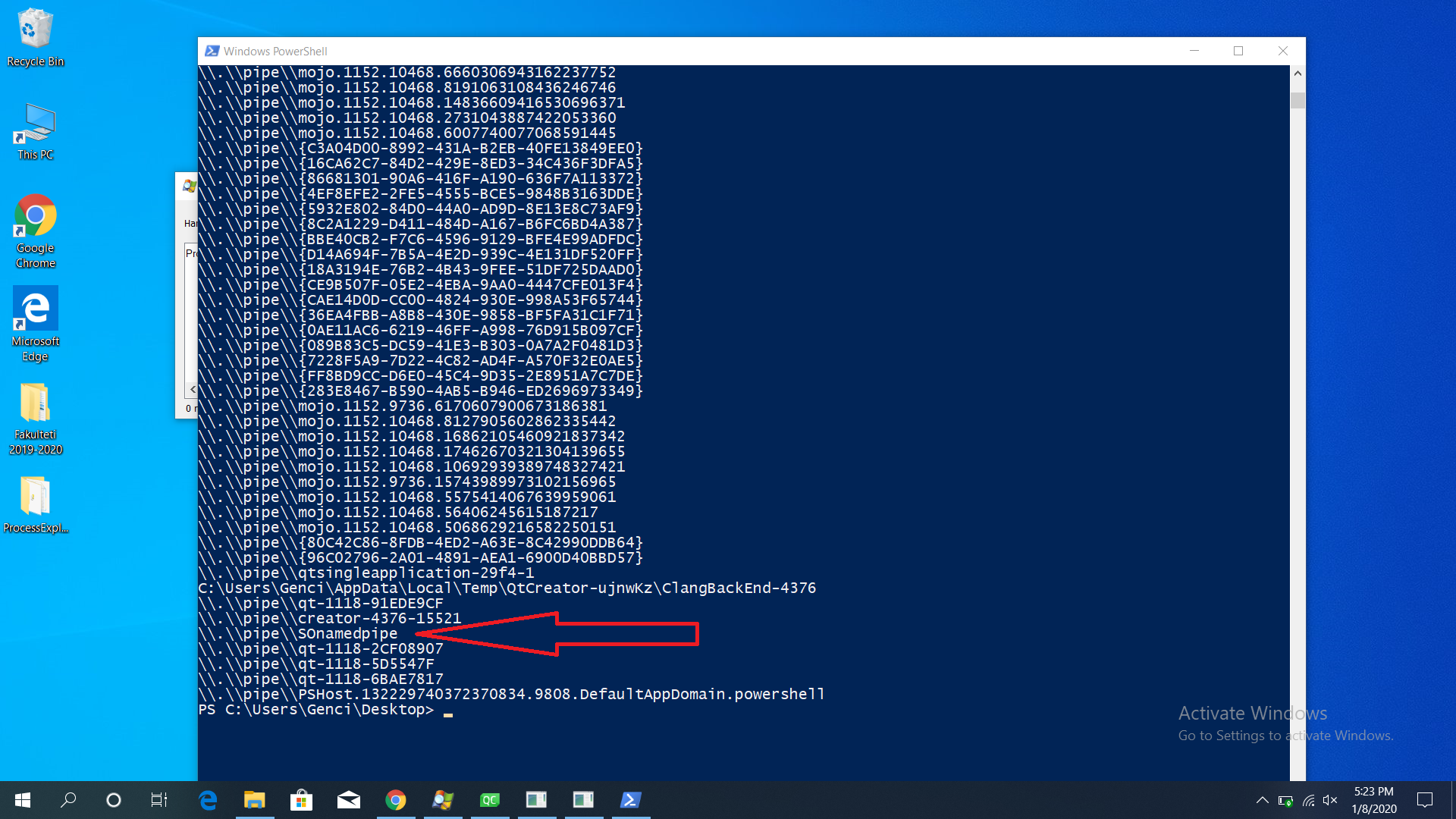
**\_getch**();

**CloseHandle**(hPipe);

   return 0;

}



Shkrimi ne një pipe nga shume procese.

Lista e te gjitha pipe ne widows. Komanda: [System.IO.Directory]::GetFiles("\\.\\pipe\\")

# REFERENCAT

Operating System Concepts 10th Edition - Avi Silberschatz, Peter Baer Galvin, Greg Gagne.

Microsoft win32 Documentation.

<https://docs.microsoft.com/en-us/windows/win32/ipc/pipes>

<https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes>

<https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipe-type-read-and-wait-modes>