Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Catedra Tehnologia Informației

Laboratorul 1.

Programarea în rețea

A efectuat: studenta gr. FI-171 Ciocanu Carolina

A examinat: Buldumac Oleg

Chişinău, 2020

Condiții:

1. Să se creeze o cerere(request) GET HTTP către web-serverul website-ului unite.md utilizînd socket.

2. Cu ajutorul expresiilor regulate să se extragă toate imaginile din raspunsul primit din punctul 1

3. Utilizînd Socket, HTTP și firele de execuții să se descarce toate imaginile găsite din puncul 2 într-un

Folder.

Cod Sursă în limbajul php - main.php:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Main</title>

</head>

<body>

<?php

$host = "<http://unite.md>";

$fp = fsockopen("tcp:// www.unite.md", 80, $errno, $errstr, 30);

$images = array();

if (!$fp) {

echo "$errstr ($errno)<br />\n";

} else {

$out = "GET / HTTP/1.0\r\nHost: utm.md\r\nAccept: \*/\*\r\n\r\n";

$out .= "Host: www.unite.md\r\n";

$out .= "Connection: Close\r\n\r\n";

$content = "";

$regEx = '/^[^?]\*\.(jpg|jpeg|gif|png)/';

$img = array();

$imgSrc = array();

fwrite($fp, $out);

while (!feof($fp)) {

// ex 1 start get content with get http with socket

$data = htmlspecialchars(fgets($fp, 128));

if (trim($data) != "") {

// echo $data . "<br>";

$content = strval($data);

if(strpos($content, 'img')) {

array\_push($img, $content);

}

}

}

foreach($img as $image){

// ex 2 sart to get all images from ex 1

$data = html\_entity\_decode($image);

$libxml\_previous\_state = libxml\_use\_internal\_errors( true );

$doc = new DOMDocument();

$doc->loadHTML($data);

libxml\_use\_internal\_errors( $libxml\_previous\_state );

$xpath = new DOMXPath($doc);

$src = $xpath->evaluate('string(//img/@src)');

if(preg\_match('/^[^?]\*\.(jpg|gif|png)/', $src, $matches)){

echo $src . "<br>";

if(!strpos(strval($src), '/')){

array\_push($imgSrc, $host.$src);

} else {

array\_push($imgSrc, $src);

}

} else {

$lazy = $xpath->evaluate('string(//img/@lazy)');

if(preg\_match('/^[^?]\*\.(jpg|gif|png)/', $lazy, $matches)) {

echo $lazy . "<br>";

if(!strpos(strval($lazy), '/')){

array\_push($imgSrc, $host.$lazy);

} else {

array\_push($imgSrc, $lazy);

}

} else {

$wide = $xpath->evaluate('string(//img/@wide)');

if(preg\_match('/^[^?]\*\.(jpg|gif|png)/', $wide, $matches)) {

echo $wide . "<br>";

if(!strpos(strval($wide), '/')){

array\_push($imgSrc, $host.$wide);

} else {

array\_push($imgSrc, $wide);

}

} else {

echo "FALSE <br>";

}

}

}

}

// ex 3 start with download images

echo "<pre>";

print\_r($imgSrc);

foreach($imgSrc as $img) {

$file\_name = basename($img);

if(file\_put\_contents($file\_name, file\_get\_contents($img))){

echo "File downloaded sccessfully";

} else {

echo "File downloading failed";

}

}

fclose($fp);

}

?>

</body>

</html>

Întrebări:

**• Ce este un Socket ?**

Un socket este un punct final al unei legături de comunicare cu două sensuri între două programe care rulează în rețea. Un socket este legat de un număr de port, astfel încât stratul TCP poate identifica aplicația la care datele sunt destinate să fie trimise. Un punct final este o combinație între o adresă IP și un număr de port. Fiecare conexiune TCP poate fi identificată în mod unic prin cele două puncte finale ale acesteia. Astfel poți avea mai multe conexiuni între gazdă și server.

**• Scopul protocolului HTTP**

Obiectul principal al manipulării în HTTP este resursa la care se referă URI (Uniform Resource Identifier) în solicitarea clientului. De obicei, aceste resurse sunt fișiere stocate pe server, dar pot fi obiecte logice sau ceva abstract. O caracteristică a protocolului HTTP este abilitatea de a specifica în cerere și răspuns un mod de reprezentare a aceleiași resurse în diverși parametri: format, codare, limbă etc. (în special, un antet HTTP este utilizat pentru aceasta). Datorită capacității de a specifica metoda de codificare a mesajului, clientul și serverul pot schimba date binare, deși acest protocol este unul text.

**• Ce este un fir de execuție ?**

Un fir de executie este unitatea de executie a unui proces. Fiecarui fir de executie i se asociaza o secventa de instructiuni, un set de registri CPU si o stiva. Procesul nu executa instructiuni, este un spatiu de adresare comun pentru unul sau mai multe fire de executie. Firele de executie sunt cele care executa instructiunile. Firele de executie seamana cu procesele, pot fi la fel planificate pentru executie. Principala diferenta este ca firul se executa in spatiul de adresare al procesului caruia apartine si poate modifica valori care sunt vazute si de celelalte fire care apartin aceluiasi proces. Din aceasta cauza apare necesitatea ca firele sa comunice intre ele, adica trebuie sincronizat accesul la datele utilizate in comun. Sincronizarea asigura siguranta datelor, adica prin sincronizare se previn situatile ca un fir sa modifice o variabila care este tocmai utilazata de catre un alt fir de executie.

**• Diferența dintre TCP si UDP**

Atât protocoalele TCP și UDP sunt protocoale utilizate pentru trimiterea de biți de date - cunoscute ca pachete - prin Internet. Ambele protocoale se bazează pe protocolul IP. Cu alte cuvinte, dacă trimiteți un pachet prin TCP sau UDP, acel pachet este trimis la o adresă IP. Aceste pachete sunt tratate în mod similar, deoarece sunt redirecționate de pe computerul dvs. către routerele intermediare și până la destinație.

Când o aplicație utilizează UDP, pachetele sunt trimise doar destinatarului. Expeditorul nu așteaptă să se asigure că destinatarul a primit pachetul - pur și simplu trimite în continuare pachetele următoare. Dacă destinatarul pierde câteva pachete UDP aici și acolo, acestea sunt doar pierdute - expeditorul nu le va retrimite. Pierderea tuturor acestor cheltuieli înseamnă că dispozitivele pot comunica mai repede.

TCP este vorba despre fiabilitatea-pachetele trimise cu TCP sunt urmărite astfel încât nu se pierde sau se pierde date în tranzit. Din acest motiv, descărcările de fișiere nu devin corupte, chiar dacă există probleme în rețea. Desigur, dacă destinatarul este complet offline, calculatorul dvs. va renunța și veți vedea un mesaj de eroare care spune că nu poate comunica cu gazda de la distanță.

**• Ordinea în care octeții sunt secvențializați în rețea**

În sistemele moderne de calcul și de comunicații digitale, informația este de obicei reprezentată ca o secvență de octeți. În cazul în care numărul nu poate fi reprezentat printr-un octet, contează în ce ordine acești octeți sunt scrise în memoria computerului sau transmise prin linii de comunicare. Adesea, alegerea ordinului de scriere a octeților este arbitrară și determinată numai prin convenție.

Comandă senior la juniori

Comanda de la cea mai senior la cea mai junior: An...A0. Această ordine corespunde ordinii obișnuite de scriere a cifrelor arabe, de exemplu, numărul o sută douăzeci și trei ar fi fost scrise în aceeași ordine ca 123. În aceeași ordine este obișnuit să scrieți octeți în literatura tehnică și educațională, cu excepția cazului în care este indicată în mod clar o altă ordine.

Comandă de la Junior la Senior – A0...An

Această comandă de înregistrare este acceptată în memoria computerelor personale cu procesoare de arhitectură x86 și, prin urmare, este numită uneori ordinul Intel byte (cu numele companiei care a creat arhitectura x86). Procesoarele moderne x86 vă permit să lucrați cu operanzi cu un, doi, patru și opt octeți. Cu această ordine de octeți, circumstanța este foarte convenabilă ca atunci când mărimea (numărul de octeți) al operandului crește, valoarea primului său octet să rămână aceeași: 3210 → 3210´0000. Pentru ca de la cea mai mare la cea mai mică, valoarea se va schimba, de exemplu: 0123 → 0000´0123;

Comandă comutabilă - În mod obișnuit, ordinea de octeți este selectată programatic în timpul inițializării sistemului de operare, dar poate fi selectată și de jumperii hardware de pe placa de bază. În acest caz, este mai corect să vorbim despre ordinea de byte a sistemului de operare. Comanda de octeți comutabilă este uneori numită engleză. bi-endian.

Comandă mixtă - Ordinea de byte mixtă este uneori folosită atunci când se lucrează cu numere mai lungi decât un cuvânt mașină. Un număr este reprezentat de o secvență de cuvinte de mașină care sunt scrise într-un format care este natural pentru o arhitectură dată, dar cuvintele în sine urmează în ordine inversă.

**• Cele patru nivele ale stivei TCP/IP și șapte nivele ale modelului OSI**

Niveluri TCP / IP :

1. Strat de aplicație
2. Strat de transport
3. Stratul de internet (strat de rețea) (strat de internet),
4. Strat de legătură (strat de acces rețea).

Standardul OSI are in componenta **7 nivele:**

1. **Physical** (Fizic)
2. **Data Link** (Legatura de Date)
3. **Network** (Retea)
4. **Transport** (Transport)
5. **Session** (Sesiune)
6. **Presentation** (Prezentare)
7. **Application** (Aplicatie)

**• Din ce cauză nu puteți transmite cereri HTTP website-ului utm.md utilizînd portul 80 ?**

Website-ul utm.md folosește protocolul https și nu http, însă website-ul unite.md folosește protocolul http. Iar portul 80 este unul dintre numerele de port utilizate cel mai frecvent în suita TCP (Transmission Control Protocol). Orice client Web / HTTP, cum ar fi un browser Web, utilizează portul 80 pentru a trimite și primi pagini web solicitate de la un server HTTP. Gestionează toate cererile bazate pe HTTP care provin de la un computer, indiferent de numărul de solicitări și de inițierea clienților Web. În mod similar, serverul HTTP răspunde la toate solicitările primite la portul 80.

**• Cum se creează o conexiune Socket ?**

Pentru a crea o conexiune Socket, trebuie să știți 2 lucruri despre server, locația rețelei sale (adresa IP) și ce port pe care rulează. Pașii implicați în crearea unui socket pe partea serverului sunt următorii:

1. Creați o priză cu apelul de sistem socket ().
2. Legați soclul la o adresă folosind apelul de sistem de legare ().
3. Ascultați conexiunile cu apelul de sistem listen ().

4. Acceptați o conexiune cu apelul de sistem accept ().

5 Trimiteți și primiți date.

**• Cum se transmit date intr-un Socket ?**

Utilizând socket TCP pentru interacțiunea client-server, o modalitate de a trimite diferite tipuri de date către server și de a putea să-l diferențieze între cei doi este să dedici primul octet (sau mai multe dacă ai mai mult de 256 tipuri de mesaje) ca un fel de identificator. Dacă primul octet este unul, atunci este mesajul A, dacă este 2, atunci mesajul său B. Un mod ușor de a trimite acest lucru prin soclu este de a utiliza DataOutputStream / DataInputStream:

**• Cum se citesc datele dintr-un Socket ?**

Creăm o instanță a clasei InputStreamReader numită streamReader. Trecem în parametrul, sock.getInputStream (). Aceasta primește fluxul de intrare al obiectului socket creat de noi. Apoi, creăm un obiect BufferedReader numit cititor și trecem în parametrul, streamReader. Aceasta se leagă în fluxul de intrare de la obiectul socket pe care l-am creat la clasa BufferedReader, astfel încât să putem citi datele din socket. Apoi, creăm o valoare denumită variabilă și o setăm egală cu reader.readLine (). Aceasta este citită în valoarea din obiectul BufferedReader, cititor. Apoi, ieșim valoarea și închidem obiectul BufferedReader odată terminat.

**• Care sunt metodele HTTP?**

**1. GET**: este cea mai folosită metodă, fiind utilizată atunci când serverului i se cere o resursă.

**2. HEAD**: se comportă exact ca metoda GET, dar serverul returnează doar antetul resursei, ceea ce permite clientului să inspecteze antetul resursei, fără a fi nevoit să obțină și corpul resursei.

**3. PUT**: metoda este folosită pentru a depune documente pe [server](https://ro.wikipedia.org/wiki/Server), fiind inversul metodei GET.

**4. POST**: a fost proiectată pentru a trimite date de intrare către server.

**5. DELETE**: este opusul metodei PUT.

**6. TRACE**: este o metodă folosită de obicei pentru diagnosticare, putând da mai multe informații despre traseul urmat de legătura HTTP, fiecare server proxy adăugându-și semnătura în antetul Via.

**7. OPTIONS**: este folosită pentru identificarea capacităților serverului Web, înainte de a face o cerere.

**8. CONNECT**: este o metodă folosită în general de serverele intermediare.

• **Codurile de stare HTTP (200, 301, 302, 401, 404, 405, 500)**

200 – OK - Cererea a reușit.

301 – Schimbat permanent - Resursei solicitate i s-a atribuit un nou URI permanent și orice referințe viitoare la această resursă ar trebui să utilizeze unul dintre URI-urile returnate. Clienții cu funcții de editare a legăturilor ar trebui să re-conecteze automat referințele la URI-solicitare la una sau mai multe dintre referințele noi returnate de server, acolo unde este posibil. Acest răspuns este în cache, dacă nu este indicat altfel.

302 - Găsite - el a solicitat resursa se află temporar sub un URI diferit. Întrucât redirecționarea ar putea fi modificată ocazional, clientul ar trebui să utilizeze URI-ul Cerere pentru viitoarele solicitări. Acest răspuns este în cacheabil numai dacă este indicat de un câmp antet Cache-Control sau Expir.

401 – Neautorizat - Cererea necesită autentificarea utilizatorului. Răspunsul TREBUIE să includă un câmp antet WWW-Autentificare (secțiunea 14.47) care conține o provocare aplicabilă resursei solicitate.

404 - Nu a fost gasit - Serverul nu a găsit nimic care să corespundă URI-ului de solicitare. Nu se indică dacă starea este temporară sau permanentă. Codul de stare 410 (Gone) TREBUIE utilizat dacă serverul știe, printr-un mecanism intern configurabil, că o resursă veche nu este permanent disponibilă și nu are o adresă de redirecționare. Acest cod de stare este utilizat în mod obișnuit atunci când serverul nu dorește să dezvăluie exact de ce cererea a fost refuzată sau când nu se aplică niciun alt răspuns.

405 - metoda nepermisa - Metoda specificată în linia de solicitare nu este permisă pentru resursa identificată de URI de solicitare. Răspunsul TREBUIE să includă un antet Allow care conține o listă de metode valide pentru resursa solicitată.

500 - eroare interna a serverului - The server encountered an unexpected condition which prevented it from fulfilling the request.

**• Ce este HTTP securizat ?**

S-HTTP (Secure HTTP) este o extensie la Hypertext Transfer Protocol (HTTP) care permite schimbul sigur de fișiere pe World Wide Web. Fiecare fișier S-HTTP este fie criptat, conține un certificat digital sau ambele. Pentru un document dat, S-HTTP este o alternativă la un alt protocol de securitate binecunoscut, Secure Sockets Layer (SSL). O diferență majoră este că S-HTTP permite clientului să trimită un certificat pentru autentificarea utilizatorului, în timp ce, folosind SSL, doar serverul poate fi autentificat. S-HTTP este mai probabil să fie utilizat în situațiile în care serverul reprezintă o bancă și necesită autentificare de la utilizator care este mai sigur decât un utilizator și o parolă.

**• Diferența dintre HTTP și HTTPS**

Mai jos sunt câteva dintre principalele diferențe între protocoalele HTTP și HTTPS, în nicio ordine particulară:

1 .URL-ul HTTP din bara de adrese a browserului dvs. este http: // și URL-ul HTTPS este https:

2. HTTP nu este securizat în timp ce HTTPS este securizat.

3. HTTP trimite date peste portul 80 în timp ce HTTPS utilizează portul 443.

4. HTTP funcționează la nivelul aplicației, în timp ce HTTPS funcționează la nivelul transportului.

5.Nu sunt necesare certificate SSL pentru HTTP, cu HTTPS este necesar să aveți un certificat SSL și să fie semnat de un CA.

6.HTTP nu necesită validare de domeniu, în condițiile în care HTTPS necesită cel puțin validarea domeniului și anumite certificate chiar necesită validarea documentelor legale.

7.Nici o criptare în HTTP, cu HTTPS datele sunt criptate înainte de a fi trimise.

**• Diferența dintre Socket și WebSocket**

Websockets utilizează socket în implementarea lor. Websockets se bazează pe un protocol standard (acum în apel final, dar încă nu final) care definește o conexiune „strângere de mână” și mesaj „cadru”. Cele două părți parcurg procedura de strângere de mână pentru a accepta reciproc o conexiune și apoi utilizează formatul standard de mesaj („cadru”) pentru a transmite mesaje înainte și înapoi.

**• Diferența dintre un proces și un fir de execuție**

Proces:

Proces înseamnă că orice program este în execuție. Blocul de control al procesului controlează funcționarea oricărui proces. Blocul de control de proces conține informații despre procese, de exemplu: prioritate proces, id proces, stare proces, CPU, registru etc. Un proces poate crea alte procese care sunt cunoscute sub numele de procese copil. Procesul necesită mai mult timp pentru a se încheia și este izolat înseamnă că nu împărtășește memoria cu niciun alt proces.

Fir:

Firul este segmentul unui proces înseamnă că un proces poate avea mai multe fire și aceste mai multe fire sunt conținute în cadrul unui proces. Un fir are 3 stări: rulare, gata și blocat. Firul durează mai puțin timp pentru a se termina în comparație cu procesul de proces și ca firele de proces nu se izolează.

**• Ce este un Deadlock și când el apare ?**

O blocare are loc atunci când mai multe procese încearcă să acceseze aceeași resursă în același timp.  
Un proces pierde și trebuie să aștepte ca celălalt să termine.  
Un impas apare atunci când procesul de așteptare se menține în continuare la o altă resursă de care prima are nevoie înainte de a putea termina.  
Deci, un exemplu:  
Resursa A și resursa B sunt utilizate de procesul X și de procesul Y  
X începe să folosească A.  
X și Y încearcă să înceapă să utilizeze B  
Y 'câștigă' și primește B primul  
acum Y trebuie să folosească A  
A este blocat de X, care așteaptă Y

**• Pentru ce este nevoie de metoda join() în programarea multithreading?**

Clasa Thread oferă metoda join () care permite unui thread să aștepte până când un alt thread își termină execuția. Dacă t este un obiect Thread al cărui fir se execută în prezent, atunci t. join () se va asigura că t se încheie înainte ca următoarea instrucțiune să fie executată de program.

Concluzii:

În urma efectuării lucrării de laborator nr. 1 am studiat socket și cum se poate cu ajutorul lui de preluat datele de la un website pe calculatorul local. Deasemenea am creat un program care îmi permite să citesc datele de pe website-ul unite.md si utm.md și am învățat cum pot să accesez și să descarc fișierele incluse în pagină utilizând regex . Am înțeles main cum să utilizez protocolul http și portul 80. Deasemenea am înțeles cum diferența dintre protocolul http și https.