

Tema 5 Server web asincron



- Data publicare: **20.05.2022**
- Deadline hard: **31.05.2022, ora 23:55**

Obiectivele temei

- Aprofundarea noțiunilor legate de lucru cu socketi.
- Deprinderi de implementare și proiectare a aplicațiilor care folosesc operații asincrone și alte operații I/O avansate.
- Aprofundarea utilizării API-ului pentru operații I/O avansate la nivelul sistemelor de operare Linux și Windows.

Enunț

Să se implementeze un server web care să folosească operații avansate de intrare/ieșire:

- operații asincrone pe fișiere;
- operații non-blocate pe socketi;
- zero-copying;
- multiplexarea operațiilor I/O.

Pentru implementare se va folosi API-ul de operații I/O avansate specific sistemelor de operare Linux și Windows:

-
- & friends/
-

Serverul web va utiliza API-ul modern de multiplexare pentru a aștepta conexiuni din partea clientilor: (Linux) și Windows. Pe conexiunile realizate se vor recepționa cereri din partea clientilor și apoi se vor distribui răspunsurile către aceștia.

Clientii și serverul vor comunica folosind protocolul HTTP. Pentru parsarea cererilor HTTP din partea clientilor recomandăm folosirea acestui parser HTTP, disponibil și în cadrul directorului de resurse ale temei. Va trebui să folositi un callback pentru obținerea căi către resursa locală solicitată de client. Tot în directorul de resurse, găsiți un exemplu simplificat de folosire a parser-ului.

Serverul implementează o **funcționalitate limitată** a protocolului HTTP, aceea de a **transmite fișiere clientilor**. Serverul va furniza fișiere din directorul `AWS_DOCUMENT_ROOT`, definit în cadrul antetului temei. Fișiere se găsesc doar în subdirectorul `AWS_DOCUMENT_ROOT/static/`, respectiv `AWS_DOCUMENT_ROOT/dynamic/`, iar request path-uri corespunzătoare vor fi, de exemplu, `AWS_DOCUMENT_ROOT/static/test.dat`, respectiv `AWS_DOCUMENT_ROOT/dynamic/test.dat`. Prelucrarea fișierelor va fi următoarea:

- Fișierele din directorul `AWS_DOCUMENT_ROOT/static/` sunt fișiere statice care vor fi transmise clientilor folosind API de zero-copying .
- Fișierele din directorul `AWS_DOCUMENT_ROOT/dynamic/` sunt fișiere pentru care se presupune că este necesară o fază de post-procesare din partea serverului. Aceste fișiere vor fi citite de pe disc folosind API asincron și apoi vor fi transmise către clienti. Transmiterea va folosi socketi non-blocanți (Linux) și Overlapped I/O pe socketi (Windows).
- Pentru request path-uri invalide se va transmite un mesaj .

După transmiterea unui fișier, conform protocolului HTTP, conexiunea este închisă.

Precizări/recomandări pentru implementare

- Implementarea temei presupune existența unei mașini de stări pentru fiecare conexiune, pe care să o interogați și actualizați periodic pe măsura desfășurării transferului.
 - Recomandăm crearea unei structuri care să gestioneze o conexiune, starea acesteia, conținutul bufferelor.
 - Definițiile macro-urilor și structurilor utile se găsesc în header-ului temei.
- Răspunsurile HTTP vor avea codul 200 pentru fișiere existente și 404 pentru fișiere inexistente.
 - Un răspuns valid este format din antetul HTTP, conținând directivele aferente, două newline-uri (`\r\n\r\n`), urmat de conținutul efectiv (fișierul).
 - Exemple de răspunsuri găsite în fișierul de test al parser-ului sau în .
 - Puteti folosi directive de request predefinite, precum Date, Last-Modified etc.
 - Directiva Content-Length trebuie să preciseze dimensiunea conținutului HTTP (datelor efective) la nivel de octetii.
 - Directiva Connection trebuie inițializată la close.
- Portul pe care serverul web ascultă pentru conexiuni este definit în cadrul header-ului temei ca macro: `AWS_LISTEN_PORT`.
- Directorul rădăcină raportat la care se caută resursele/fișierele este definit în cadrul header-ului temei ca macro: `AWS_DOCUMENT_ROOT`.

Resurse utile

- Pentru parsarea cererii HTTP și obținerea request path-ului puteți folosi acest parser HTTP disponibil și în directorul de resurse ale temei.
 - Pentru compilare, folosiți fișierul `Makefile`.
 - Consultați fișierul `README.md`.
 - Un exemplu specific de folosire a parser-ului pentru obținerea request path-ului este disponibil în .
- Pentru aspecte de depanare și tratare a erorilor, folosiți fișierele header disponibile.
- Folosiți subdirectorul `samples/` din directorul aferent fiecărui sistem de operare pentru exemple de utilizare a API-ului pe socketi.
 - Găsiți implementat ca exemplu un server echo bazat pe API avansat de multiplexare.
 - Găsiți implementat ca exemplu un server care trimit un răspuns HTTP.

Depanare/troubleshooting

- Pentru depanarea serverului, recomandăm folosirea și a .
- Pentru netcat, dacă dorîți să comunicați folosind HTTP, va trebui să transmiteți o cerere specifică (spre exemplu `GET /path/to/resource HTTP/1.0`). Exemplu de folosire:


```
$ nc localhost 8888
GET / HTTP/1.0
```
- Sau:


```
echo -ne "GET / HTTP/1.0\r\n\r\n" | nc -q 2 localhost 8888
```

Informații generale SO

- Catalog
- Documentație și alte resurse
- Feed RSS
- Hall of SO
- Listă de discuții
- Mașini virtuale
- Trimitere teme

Informatii SO 2021-2022

- Examen
 - Examen CA/CC 2012-2013
 - Examen CA/CC 2013-2014
 - Examen CA/CC 2014-2015
 - Examen CA/CC 2015-2016
 - Examen CA/CB/CC 2016-2017
 - Examen CA/CB/CC 2017-2018
 - Examen CA/CB/CC 2018-2019
 - Examen CA/CB/CC 2019-2020
 - Examen CA/CB/CC 2020-2021
- Anunțuri
- Calendar
- Echivalări teme
- Distincții
- Karma Awards
- SO Need to Know
- Reguli generale și notare
- Orar și împărtire pe semigrupe

Laboratoare

- Resurse
 - **Windows - Tips&Tricks**
 - Tutorial Visual Studio
 - Linia de comandă în Windows
 - C/SO Tips
 - Macro-ul DIF
 - GDB
 - Function Hooking and Windows Dll Injection
 - IPC
 - Online
 - Oprofile
 - Recapitulare
 - Thread-uri - Extra
 - Visual Studio Tips and Tricks
 - windows-video
 - Laborator 01 - Introducere
 - Laborator 02 - Operații I/O simple
 - Laborator 03 - Procese
 - Laborator 04 - Semnale
 - Laborator 05 - Gestiona memoria
 - Laborator 06 - Memoria virtuală
 - Laborator 07 - Profiling & Debugging
 - Laborator 08 - Threaduri Linux
 - Laborator 09 - Threaduri Windows
 - Laborator 10 - Operații IO avansate - Windows
 - Laborator 11 - Operații IO avansate - Linux
 - Laborator 12 - Implementarea sistemelor de fișiere

Curs

- Capitol 01: Introducere
- Capitol 02: Interfață sistemului de fișiere
- Capitol 03: Procese
- Capitol 04: Planificarea execuției, IPC
- Capitol 05: Gestiona memoriei
- Capitol 06: Memoria virtuală
- Capitol 07: Analiza executabilelor și proceselor
- Capitol 08: Securitatea memoriei
- Capitol 09: Fire de execuție
- Capitol 10: Sincronizare
- Capitol 11: Dispozitive de intrare/ieșire
- Capitol 12: Implementarea sistemelor de fișiere
- Capitol 13: Networking în sistemul de operare
- Capitol 14: Analiza performanței

Teme

- Tema Asistenți - Guardian process

- Exemple de utilizare găsiți în teste.

Precizări pentru Windows

- Operatiile pe socket și fișiere ([ReadFile](#), [WSASend](#), [TransmitFile](#)) vor fi operații asincrone realizate folosind [Overlapped I/O](#).
- Așteptarea încheierii operațiilor asincrone (atât pe fișiere cât și pe socket) se va realiza unificat, folosind [I/O Completion Ports](#).
- Pentru implementare, recomandăm să porniți de la [exemplul de echo server pus la dispoziție](#).
 - [API-ul I/O Completion Ports este apelat prin intermediul wrapper-elor din cadrul header-ului \[w_iocp.h\]\(#\).](#)
 - Socketii creati folosind apelul [socket](#) sunt socketi ce pot fi folosiți pentru [operații asincrone](#).
 - Pentru operații asincrone de comunicare pe socket foloșiti funcțiile [WSASend](#), respectiv [WSARecv](#).
 - Structura [WSAOVERLAPPED](#) conține informații necesare pentru operațiile asincrone pe socketi. Este echivalentă structurii [OVERLAPPED](#).
 - Pentru adăugarea socket-ului listener în I/O Completion Port, va trebui să folosiți apelul [AcceptEx](#), așa cum se observă și în [exemplul de echo server](#).
 - [AcceptEx](#) permite notificarea unei acțiuni în momentul în care aceasta are loc, împreună cu binding-ul socket-ului.
 - Pentru operații asincrone cu sistemul de fișiere, foloșiti funcțiile [ReadFile](#) și [WriteFile](#) cu argumentele de Overlapped I/O activate.
 - Pentru obținerea de informații despre o operație asincronă încheiată (prin intermediul structurii [OVERLAPPED](#)), foloșii [GetOverlappedResult](#) (fișiere), respectiv [WSAGetOverlappedResult](#) (socket).
 - Transmisarea de fișiere statice se realizează folosind [TransmitFile](#).
- Tema se va rezolva folosind doar funcții Win32. Se pot folosi de asemenea și funcțiile de formatare [printf](#), [scanf](#), funcțiile de alocare de memorie [malloc](#), [free](#) și funcțiile de manipulare a sirurilor de caractere ([strcat](#), [strupr](#) etc.).
- Pentru partea de I/O și procese se vor folosi doar funcții Win32. De exemplu, funcțiile [open](#), [read](#), [write](#), [close](#) nu trebuie folosite; în locul acestora foloșii [CreateFile](#), [ReadFile](#), [WriteFile](#), [CloseHandle](#).

- Contestări
- Git. Indicații folosire GitLab
- Indicații generale teme
- Hackathon SO
- Tema 1 Multi-platform Development
- Tema 2 Bibliotecă stdio
- Tema 3 Loader de Executabile
- Tema 4 Planificator de threaduri
- Tema 5 Server web asincron

Table of Contents

- Tema 5 Server web asincron
 - Obiectivele temei
 - Enunț
 - Precizări/recomandări pentru implementare
 - Resurse utile
 - Depanare/troubleshooting
 - Precizări pentru Windows
 - Precizări pentru Linux
 - Testare
 - Resurse de suport
 - FAQ
 - Supor, întrebări și clarificări

Precizări pentru Linux

- Atât citirea cât și scrierea peste socketi se realizează doar la notificarea datea de [API-ul specific sistemului de operare](#), folosind [epoll](#).
 - Tot folosind [epoll](#) se va aștepta notificarea încheierii operațiilor asincrone pe fișiere.
- Pentru implementare, recomandăm să porniți de la [exemplul de echo server pus la dispoziție](#).
 - [API-ul epoll este apelat prin intermediul wrapper-elor din cadrul header-ului \[w_epoll.h\]\(#\).](#)
- Scrierea pe socket, atât în cazul fișierelor statice, cât și în cazul fișierelor dinamice, se realizează non-blocant: socketii sunt marcați ca non-blocanți, iar la un apel de scriere se scrie cât permite buffer-ul socketului. La următoarea notificare din partea [API-ul specific](#) se va realiza o nouă scriere și aşa mai departe.
 - Pentru configurarea unui socket ca non-blocant puteți folosi [fcntl](#) (flag-ul [O_NONBLOCK](#)).
- Pentru implementarea operațiilor I/O, foloșii funcțiile din familia [io_setup](#).
 - Pentru utilizarea funcțiilor va trebui să realizezi link-area cu [biblioteca libaio](#) și includerea header-ului [<libaio.h>](#).
 - Recomandăm folosirea unei variabile de tipul [io_context_t](#) și a unui descriptor [eventfd](#) pentru fiecare conexiune.
 - Pentru închiderea operațiilor asincrone, foloșii [io_getevents](#).
 - Puteți consulta [acest exemplu de integrare a operațiilor I/O asincrone cu eventfd](#).
- Transmisarea de fișiere statice se realizează folosind [sendfile](#).
- Tema se va rezolva folosind doar funcții POSIX. Se pot folosi de asemenea și funcțiile de formatare din familia [printf](#), funcțiile de alocare de memorie [malloc](#), [free](#) și funcțiile de lucru cu siruri de caractere ([strcat](#), [strupr](#) etc.)
- Pentru partea de I/O se vor folosi doar funcții POSIX și funcții pentru operații asincrone. De exemplu, funcțiile [fopen](#), [fread](#), [fwrite](#), [fclose](#) nu trebuie folosite; în locul acestora foloșii [open](#), [io_setup](#), [io_submit](#), [close](#).
- Scrierea și citirea pe socketi se vor face doar prin funcțiile [send\(\)](#) și [recv\(\)](#). Nu este permisă folosirea funcțiilor [read\(\)](#) și [write\(\)](#) pentru lucrul cu socketi.

Testare

- Corectarea temelor se va realiza automat cu ajutorul unor suite de teste publice:
 - teste Linux – [tema5-checker-lin.zip](#)
 - teste Windows – [tema5-checker-win](#)
- Indicații despre utilizarea suitei de teste se găsesc în fișierul [README](#) din cadrul arhivei.
- Pentru testare, pe Windows, se folosește o versiune a utilitarului [netcat](#); puteți descărca arhiva completă (inclusiv executabilul necesar) [de aici](#) (parola este nc).
- Pentru evaluare și corecțare tema va fi uploadată folosind [interfața vmchecker](#).
- În urma compilării temei trebuie să rezulte un executabil denumit [aws](#) (Linux), respectiv [aws.exe](#) (Windows).
 - Pe Windows, va trebui ca, în urma compilării, să rezulte și fișierul obiect [aws.obj](#) (pentru verificările de simboluri folosind [nm](#)).
- Suita de teste conține un set de teste. Trecerea unui test conduce la obținerea punctajului aferent acestuia.
 - În urma rulării testelor, se va acorda, în mod automat, un punctaj total. Punctajul total maxim este de 90 de puncte, pentru o temă care trece toate testele. La acest punctaj se adaugă 10 puncte care reprezintă aprecierea temei de către asistențul care o corectează.
 - Cele 100 de puncte corespund la 10 puncte din cadrul notei finale.
- Pot exista penalizări în caz de întârzieri sau pentru neajunsuri de implementare sau de stil.
- Pe lângă penalizările precizate în cadrul [listei de punctări](#), se vor avea în vedere următoarele elemente:
 - 2p** Linux și Windows – fișierele statice nu sunt transmise prin mecanismul de zero-copy
 - 0.5p** Linux și Windows – aloare statică a unui vector de conexiuni
 - 4p** Linux - nu se folosesc doar [send](#) și [recv](#) pentru operațiile cu socketi
 - 2p** Linux - foloșirea de operații blocante pe socket în locul operațiilor non-blocante
 - 1p** Linux - [recv/send/sendfile](#) sunt apelate într-o buclă și nu pe baza evenimentelor semnalizate de [epoll](#)
 - 2p** Linux - operațiile I/O asincrone pe fișiere nu sunt așteptate "integrat" folosind [eventfd](#) și [epoll](#)
 - 2p** Windows - așteptarea încheierii operațiilor asincrone (atât pe fișiere cât și pe socketi) nu este realizată unificat, folosind I/O Completion Ports
 - 1p** Linux și Windows – alte implementări nevalide la nivelul sistemului asincron de comunicatie
- O temă care nu folosește operații I/O asincrone pe fișiere: [io_*](#) (din [libaio](#)) pe Linux, respectiv Overlapped I/O pe Windows va pierde punctajul pentru testele cu fișiere dinamice (testele 26-35)
- Testul 0** din cadrul checker-ului temei verifică automat coding style-ul surselor voastre. Ca referință este folosit [stilul de coding din kernelul Linux](#). Acest test nu oferă sau scade puncte, dar poate fi folosit orientativ de către echipa de corecțare pentru a penaliza problemele grave de coding style. Important este să vă definiți un stil și să-l folosiți consecvent. Pentru mai multe informații despre un cod de calitate citiți [pagina de recomandări](#).

Resurse de suport

- Cursuri
 - [Curs 10 - Dispozitive de intrare/ieșire](#)
 - [Curs 11 - Networking în sistemul de operare](#)
- Laboratoare
 - [Laborator 10 - Operații I/O avansate -- Windows](#)
 - [Laborator 11 - Operații I/O avansate -- Linux](#)
- [Parser-ul HTTP recomandat pentru parsarea cererilor](#).
- [Tutorial HTTP](#)
- [Resurse tema 5](#)
- Sample-uri pentru lucru cu socketi
 - [Sample-uri Linux](#)

- [Sample-uri Windows](#)
- [Teste](#)
 - [Teste Linux](#)
 - [Teste Windows](#)
- [Interfață de upload vmchecker](#)
- [Lista de discuții a cursului](#)



Resursele temei se găsesc în repo-ul [temei](#) de pe GitHub.

FAQ

- **Q:** Tema se poate face în C++?
- **A:** Nu.

Suport, întrebări și clarificări

Pentru întrebări sau nelămuriri legate de temă folosiți [forumul temei](#).

Orice întrebare pe forum **trebuie** să conțină o descriere cât mai clară a eventualei probleme. Întrebări de forma: "Nu merge X. De ce?" fără o descriere mai amănunțită vor primi un răspuns mai greu. Înainte să postați o întrebare pe forum citiți și celelalte întrebări(dacă există) pentru a vedea dacă întrebarea voastră a fost deja adresată sub o altă formă(in cazul în care răspunsul din partea echipei vine mai greu este mai rapid să căutați voi deja printre întrebările existente).



ATENȚIE să nu postați imagini cu părți din soluția voastră pe forumul pus la dispoziție sau orice alt canal public de comunicație. Dacă veți face acest lucru, vă asumați răspunderea dacă veți primi copiat pe temă.

Logged in as: Bogdan Mihai TUDORACHE (108609) (bogdan.tudorache99)

so/teme/tema-5.txt · Last modified: 2022/05/20 21:39 by ionut.mihalache1506

[Old revisions](#)

[Media Manager](#) [Manage Subscriptions](#) [Back to top](#)