ZADÁNÍ:

Úkolem je vytvoření serveru v jazyce C/C++ komunikujícího prostřednictvím protokolu HTTP, který bude poskytovat různé informace o systému. Server bude naslouchat na zadaném portu a podle url bude vracet požadované informace. Server musí správně zpracovávat hlavičky HTTP a vytvářet správné HTTP odpovědi. Typ odpovědi bude text/plain. Komunikace se serverem by měla být možná jak pomocí webového prohlížeče, tak nástroji typu wget a curl.

Server bude přeložitelný pomocí Makefile, který vytvoří spustitelný soubor hinfosvc.

Tento server bude spustitelný s argumentem označující lokální port na kterém bude naslouchat požadavkům:

.\hinfosvc 12345

**Server bude možné ukončit pomocí CTRL+C. Server bude umět zpracovat následující tři typy dotazů, které jsou na server zaslané příkazem GET:**

1. Získání doménového jména

Vrací síťové jméno počítače včetně domény, například:

GET http://servername:12345/hostname

merlin.fit.vutbr.cz

2. Získání informací o CPU

Vrací informaci o procesoru, například:

GET http://servername:12345/cpu-name

Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz

3. Aktuální zátěž

Vrací aktuální informace o zátěži. Tento vyžaduje výpočet z hodnot uvedených v souboru /proc/stat (viz odkaz níže). Výsledek je například:

GET http://servername:12345/load

65%

Potřebné informace pro odpověď lze v systému získat pomocí některých příkazů systému (uname, lscpu) a/nebo ze souborů v adresáři /proc.

IMPLEMENTACE

Implementace serveru bude v jazyce C/C++. Pro implementaci serveru je nutné využít knihovnu soketů. Není přípustné využívat knihovny pro zpracování HTTP a podobně - cílem je lightweight server, který má minimum závislostí.

Součástí projektu bude dokumentace, kterou bude představovat soubor Readme.md, jenž bude obsahovat:

• stručný popis projektu

• způsob spuštění projektu

• příklady použití projektu

HODNOCENÍ

Hodnotí se funkčnost implementace (3/4 hodnocení) a její kvalita (1/4 hodnocení):

• struktura projektu

• srozumitelnost a jednoduchost kódu

• dokumentace

• množství závislostí na dalších knihovnách (rozumný balanc mezi tím co si napsat sám a co použít z knihoven)

TESTOVÁNÍ

Vyzkoušejte si, zda Vám fungují alespoň základní varianty, tedy:

.\hinfosvc 12345 &

curl http://localhost:12345/hostname

curl http://localhost:12345/cpu-name

curl http://localhost:12345/load

DOPORUČENÍ

Zadání nepostihuje ani nemůže postihovat veškeré možnosti, které Vás napadnou při řešení nebo mohou nastat. V takovém případě je nutné navrhnou přijatelné a logické řešení. V případě, že si řešením nejste jisti se optejte prostřednictvím fóra.

ODKAZY

• Protokol HTTP RFC7231 - https://tools.ietf.org/html/rfc7231

• HTTP pro vývojáře: https://developer.mozilla.org/en-US/docs/Web/HTTP

• Výpočet CPU Load: https://stackoverflow.com/questions/23367857/accurate-calculation-of-cpu-usage-given-in-percentage-in-linux

**1) vytvořte si funkci, která vám vrátí aktuální využití procesoru**

**2) zjistete si hostname (/proc/sys/kernel/hostname)**

**3) zjistete si cpu name (popen("cat /proc/cpuinfo | grep "model name" | head -n 1 | awk '...)**

**4) socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP); - vytvoří socket**

**5) setsockopt na 1 pro SO\_REUSEADDR a SO\_REUSEPORT - zajistí že nebude záhadně padat při testování**

**6) struct sockaddr\_in address;**

**address.sin\_family = AF\_INET;**

**address.sin\_port = htons(tvujPort);**

**address.sin\_addr.s\_addr = INADDR\_ANY;**

**7) bind(socket, address) - nastaví socket**

**8) listen() - socket začne přijímat spojení**

**9) while true do accept() - pak můžete se socketem pracovat jako se souborem**

10) pokud spojení začíná na "GET /hostname" (strncmp) nebo jiný z těch tří endpointů...

11) tak do socketu zapiš "HTTP/1.1 200 OK\r\nContent-Type: text/plain;\r\n\r\n" a tvou odpověď, bez nového řádku na konci

12) Jinak zapiš odpověď se stavem "400 Bad Request"

13) Pokud tak ještě neučinils, vyčti zbytek dat v socketu ať si třeba prohlížeč nestěžuje

14) close() end while