Vysoké učení technické v Brně

FIT

IPK - Počítačové komunikace a sítě
2018

Projekt - Klient-server pro získání informace o uživatelích

Jiří Juřica / xjuric29

Obsah

Příprava

Soubor /etc/passwd

Při plánování návrhu programů jsem se zamýšlel, zda je struktura souboru passwd nějak standardizovaná, především jestli je v něm pevné pořadí položek a zda každá položka má nějakou maximální délku.
Mé bádání bylo úspěšné tak na půl. Pořadí položek v souboru je pevně dané, odděleno znakem ':' dvojtečka dle následujícího řádku (man 5 passwd, 1997 [cit. 2018-03-12]):

> login_name:passwd:UID:GID:user_name:directory:shell

Z položek má dohledatelnou maximální velikost jen login, který je omezen na 32 znaků. Při návrhu jsem zjišťoval zbylé velikosti odvozováním přímo z dat na studentských serverech, kde mi pro directory vyšla nejdelší cesta na 24 znaků a nejdelší user_name na 27 znaků. Pro jistotu jsem se rozhodl počítat s až 64 znaky pro obě položky.
(Understanding /etc/passwd File Format [nixCraft]. Dostupné z: https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/)

Návrh protokolu

Pro návrh protokolu jsem využil pouze záznamů a pdf z přednášek IPK, kde se rozebíral princip fungování protokolů jako je DNS, SMPT, FTP apod. Z toho mi vyšlo, že pro klienta budu mít tři dotazy:

> C: SEND USERINFO {LOGIN}
> C: SEND HOMEPATH {LOGIN}
> C: SEND USERLIST {PREFIX}

Dotazování začíná klíčovým slovem SEND, následuje mezera a v závislosti na požadavku je třeba vybrat z příslušného klíčového slova, za kterým opět následuje mezera a konkrétní login (ve třetím případě je možné zasílat požadavek i bez prefixu pro vrácení seznamu všech loginů).
Odpověď serveru vždy začíná slovem OK nebo ERROR podle toho, zda se požadavek povedlo úspěšně splnit a dále se liší v závislosti na zaslaném dotazu. Pro SEND USERINFO {LOGIN} bude vráceno:

> S: OK {USERINFO}

Pro SEND HOMEPATH {LOGIN}:

> S: OK {HOMEPATH}

Pro SEND USERLIST {PREFIX}:

> S: OK {ITEMCOUNT}

Kde ITEMCOUNT je počet položek, jež mají být očekávány. Než je server odešle, je třeba zaslat, že je klient připraven ke zpracování:

> C: READY

Server následně posílá všechny nalezené položky oddělené znakem nového řádku a jako poslední zašle dvojtečku, která se v loginu objevit nemůže, protože slouží jako oddělovač v

souboru passwd. Po provedení logického příkazu (SEND nebo SEND + READY) server zrovna ukončí spojení a na další požadavek nečeká.

Implementace

V rámci implementace nakonec vzniklo 12 pomocných funkcí mimo mainy obou programů, z nichž bych rád zmínil fci readline, která vylepšuje načítání řádků na libovolnou velikost a přibližuje tak programování v c o trochu víc k vyšším programovacím jazykům. Mimo to jsou pro běh celé aplikace nejdůležitější fce getUserInfo, getHomePath a getLoginList, které připravují data pro zaslání klientovi.
Pokud by se mělo jednat o reálnou aplikaci, bylo by lepší načíst si veškeré údaje do nějaké hashovací tabulky a pouze jednorázově načíst soubor a následně rychleji prohledávat tabulku. Tato implementace mi ale v projektu chybí, rozhodl jsem se jej zbytečně nekomplikovat.

Demonstrace běhu serveru

Výpis z telnetu:

```
C: SEND USERINFO jirkaj
S: OK Jiří Juřica,,,


C: SEND HOMEPATH jirkaj
S: OK /home/jirkaj

C: SEND USERLIST r
S: OK 2
C: READY
S: root
S: rtkit
S: :
```