Projekt - Triviální distribuovaný souborový systém

Cílem projektu je implementovat *klienta* pro triviální (read-only) distribuovaný souborový systém. Tento systém používá URL pro identifikaci souborů a jejich umístění. Systém pro přístup k souborům používá File Service Protocol (FSP). V projektu bude stačit implementovat pouze jeden typ požadavku, kterým je příkaz `GET`. Systém používá symbolických jmen, které jsou překládány na IP adresy pomocí protokolu Name Service Protocol (NSP). Tento protokol umožňuje získat IP adresu a číslo portu, kde daný souborový server běží.

Jmenný prostor souborové systému

Distribuovaný souborový systém používá Simplified URL (SURL) schématu pro adresování souborů. SURL řetězec má následující strukturu:

```
PROTOCOL://SERVER_NAME/PATH
```

- `PROTOCOL` je název protokolu pro přístup k souboru. V tomto projektu se uvažuje pouze FSP.
- `SERVE_NAME` je jméno serveru. Jméno serveru může být tvořeno alfanumerickými znaky a `"-"`, `"_"` a `"."`.
- `PATH` je cesta k souboru na serveru. Cesta používá znak `"/"` pro oddělení adresářů.

Příklady

Toto je příklad URL souboru `file.txt`, který se nachází na serveru `some.server` v adresáři `foo/bar`.

```
fsp://some.server/foo/bar/file.txt
```

Toto je příklad dlouhého, ale validního názvu serveru:

```
fsp://another.server.with.a.quite.long.name/file_in_root_folder.txt
```

Name Service Protocol

NSP protokol slouží pro překlad jména serveru na IP adresu a číslo portu serveru. Ano, na rozdíl od DNS systému se zde symbolické jméno překládá nejen na IP adresu, ale také na číslo portu, kde příslušná služba běží. Protokol je velmi jednoduchý a používá pro komunikaci UDP spojení. Klient pošle dotaz na server a čeká na odpověď. Formát dotazu je:

```
WHEREIS doménové_jméno
```

Formát odpovědi je:

```
OK IP_adresa:číslo_portu
```

V případě špatně sestaveného dotazu nebo formátu jména je vrácena chyba:

```
ERR Syntax
```

nebo v případě, že dané jméno neexistuje je chyba:

```
ERR Not Found
```

Příklad

První příklad ukazuje komunikaci v případě korektního dotazu:

```
WHEREIS nejaky.muj.server


OK 147.229.8.12:4567
```

Druhý příklad ukazuje situaci s chybou:

```
WHEREIS tento.server.neexistuje


ERR Not Found
```

FSP Protocol

FSP protokol je značně zjednodušená obdoba protokolu HTTP. Pro komunikaci se používá TCP spojení. Klient pošle požadavek na server a čeká na odpověď.

Požadavek má tvar:

```
COMMAND PATH VERSION
HEADERS

MESSAGE DATA (OPTIONAL)
```

Jediným požadavkem, který budete implementovat je příkaz GET:

```
GET název_souboru FSP/1.0
Hostname: doménové_jméno
Agent: fit_login
```

Za požadavkem následuje tak jako v případě HTTP prázdný řádek! V požadavku do hlavičky Agent uveďte Vás login. Toto je důležité pro hodnocení projektu!

Řádky požadavku a záhlaví musí všechny končit CR LF (v C řetězcem `"\r\n"`). Prázdný řádek se musí skládat pouze z CR LF a žádných jiných znaků.

Odpověď závisí na tom, zda bylo možné požadavek zpracovat. Odpověď má hlavičku a datovou část. Hlavička je oddělena prázdným řádkem:

```
VERSION STATUS
HEADERS

MESSAGE DATA (OPTIONAL)
```

Správná odpověď vypadá takto:

```
FSP/1.0 Success
Length: délka_odpovědi

...obsah souboru...
```

V případě, že server nerozumí požadavku, například špatná syntax, pak je vrácena chyba:

```
FSP/1.0 Bad Request
Length: délka_odpovědi

Bližší popis chyby...
```

Nebo když soubor není nalezen, je vrácena tato chyba:

```
FSP/1.0 Not Found
Length: délka_odpovědi

Bližší popis chyby...
```

Pro ostatní nespecifikované chyby vrací server tuto odpověď:

```
FSP/1.0 Server Error
Length: délka_odpovědi

Bližší popis chyby...
```

Všechny uvedené parametry v požadavku či odpovědi jsou povinné.

Příklady

První příkladu ukazuje korektní požadavek:

```
GET loremipsum.txt FSP/1.0
Hostname: nejaky.muj.server
Agent: xrysav02
```

a toto je správná odpověď:

```
OK
Length: 574

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
Ipsum has been the industry's standard dummy text ever since the 1500s, when an
unknown printer took a galley of type and scrambled it to make a type specimen
book. It has survived not only five centuries, but also the leap into electronic
typesetting, remaining essentially unchanged. It was popularised in the 1960s
with the release of Letraset sheets containing Lorem Ipsum passages, and more
recently with desktop publishing software like Aldus PageMaker including versions
of Lorem Ipsum.
```

Druhý příklad ukazuje pokus o přístup k neexistujícímu souboru:

```
GET secret.txt FSP/1.0
Hostname: nejaky.muj.server
Agent: xrysav02
```

s přílušnou chybovou odpovědí:

```
Not Found
Length: 29

File not found on the server.
```

Klient (fileget)

Vaším úkolem je vytvořit nástroj - klienta tohoto systému v jazyce C/C++ nebo Python. Tento nástroj pojmenujte `fileget`.

Klient bude umět stahovat soubory ze serverů systému. Je tedy nutné v klientovi implementovat oba protokoly (NSP,FSP). Klient uloží stažený soubor pod jeho jménem do aktuálního adresáře. V případě, že nastane chyba, klient tuto chybu vhodným způsobem vypíše a žádný soubor nebude vytvořen.

Klienta bude možné spustit s těmito paramtery:

```
fileget -n NAMESERVER -f SURL
```

- `NAMESERVER` - IP adresa a číslo portu jmenného serveru.
- `SURL` - SURL souboru pro stažení. Protokol v URL je vždy `fsp`.

Oba parametry jsou povinné. Jejich pořadí je volitelné.

Příklad

Následující příkaz stáhne soubor `file.txt` ze serveru jménem `foo.bar`. Pro překlad jména serveru se použije jmenný server `147.229.8.12`:

```
$ ./fileget -n 147.229.8.12:3333 -f fsp://foo.bar/file.txt
```

V lokálním adresáři bude vytvořen soubor `file.txt`.

Index

Na serveru existuje vždy speciální soubor `index`, který poskytuje seznam souborů umístěných na serveru. Získání indexu je pomocí příkazu:

```
$ ./fileget -n 127.0.0.1:3333 -f fsp://muj.server.number.one/index
```

Index je textový soubor, kde na každém řádku je název jednoho souboru. Obsah souboru `index` může vypadat například takto:

```
soubor1.txt
somefile.jpg
soubor2.bin
```

Hromadný download

V případě, že je místo cesty k souboru uveden znak `*` bude klient stahovat všechny soubory, které se na serveru nachází:

```
$ ./fileget -n 127.0.0.1:3333 -f fsp://muj.server.number.one/*
```

Pro zjištění obsahu serveru je nejprve nutné stáhnou `index` soubor, který pak bude poskytovat názvy jednotlivých souborů pro stažení.

Testovací server

Pro účely testování a pro vyhodnocení projektu je k dispozici implementace testovacího serveru, který zahrnuje jmenný server a souborové servery.

Testovací server se spouští s parametry:

```
fsptest -p LOCAL_ENDPOINT -r ROOTFOLDER
```

- `LOCAL_ENDPOINT` - lokální adresa a číslo portu na kterém naslouchá jmenný server.
- `ROOTFOLDER` - složka, kde jsou uloženy soubory pro jednotlivé souborové servery.

Tento testovací server spustí jeden jmenný server na zvoleném portu a tolik souborových serverů, kolik je složek v uvedeném adresáři. Název složky pak představuje jméno serveru. Soubory ve složce serveru jsou pak dostupné ke stažení.

Příklad

V případě této struktury:

```
ipkdata
 |
 |- muj.server.number.one
 |   |- soubor1.txt
 |   |- somefile.jpg
 |   |- soubor2.bin
 |- dalsi.server
     |- nejakyfile.txt
     |- readme.md
```

budou vytvořeny dva souborové servery s názvy `muj.server.number.one` a `dalsi.server`. První server bude mít tři soubory, druhý pak dva soubory. Jestliže bude server spuštěn na lokálním počítači s těmito parametry:

```
$ fsptest -p 127.0.0.1:3333 -r ./ipkdata
```

Bude stažení druhého souboru z prvního serveru provedeno příkazem:

```
$ ./fileget -n 127.0.0.1:3333 -f fsp://muj.server.number.one/somefile.jpg
```

Vyzkoušejte si

Pro vyzkoušení si komunikaci se serverem je možné použít například nástroje `nc`. Tento umožňuje TCP (podobně jako telnet) a UDP komunikaci. Testovací server si můžete vyzkoušet takto (upraveno pro zobrazení - symboly `<` označují, že se jedná o vstup na stdin; podobně `>` neuvidíte neboť se jedná o označení stdout v příkladech):

```
$ nc -u 127.0.0.1 3333
< WHEREIS muj.server.number.one
> OK 127.0.0.1:62458

< WHEREIS server.co.neexistuje
> ERR Not Found

< WHERE IS chybna.syntax.prikazu
> ERR Syntax
```

Stejně tak si můžete vyzkoušet komunikaci se souborovým serverem:

```
$ nc 127.0.0.1 62458
< GET index FSP/1.0
< Agent: xrysav02
< Hostname: muj.server.number.one
<
> FSP/1.0 Success
> Length:23
>
> soubor1.txt
> somefile.jpg
> soubor2.bin
```

Hodnocení

Hodnotí se primárně funkčnost odevzdaného řešení. Polovina bodů je za zvládnutí základní operace - stažení požadovaného souboru. Čtvrtina bodů je za ošetření různých nesprávných, ale očekávatelných problémů. Poslední čtvrtina bodů je za implementaci funkce stažení všech souborů ze serveru. Při výskytu nečekaných situací by klent neměl spadnout či se zaseknout. Korektní je vypsat chybu a ukončit se, popřípadě se pokusit operaci zopakovat.

Funkčnost

Implementované funkce	Bodů	Vysvětlení
základní GET	10	Implementuje základní funkcionalitu. Tedy stáhne a uloží požadovaný soubor.
Nestandardní situace	5	Ošetření nestandardních situací: i) Jmenný server neodpoví na dotaz, ii) souborový server odmítne nebo nečekaně ukončí spojení, iii) je zadán neplatný vstup, iv) jmenný server vrací nesprávnou odpověď, nebo v) souborový systém vrací nesprávnou odpověď.

Implementované funkce	Bodů	Vysvětlení
GET ALL	5	Dokáže stáhnout všechny soubory z jednoho serveru.