PROTOCOALE DE COMUNICATIE: Tema #1
Server de fisiere

Termen de predare: 31 MARTIE 2015

Titulari curs: *Valentin CRISTEA, Gavril GODZA, Florin POP*

Responsabili Tema: **Cristian Chilipirea**

Implementati un server de fisiere care sa poata raspunde la urmatoarele comenzi, fiecare avand forma de comunicatie:

- `ls dirName` serverul trimite clientului lista fisierelor din directorul `dirName`

```
Client              Server
      -----ls .---------->
      <----num------------
      <----file_0---------
      <----file_1---------
      <----file_2---------
      <----....-----------
      <----file_[num-1]-
```

- `cd dirName` serverul schimba directorul curent la directorul `dirName`

```
Client             Server
       -----cd .----->
```

- `cp fileName` serverul trimite clientului un fisier `fileName` din directorul curent (clientul il scrie in fisierul cu acelasi numele precedat de `"new_"` din directorul executabilului)

```
Client                        Server
      -----cp fileName------>
      <----size-------------
      <----part_1------------
      <----part_2------------
      <----part_3------------
      <----........----------
      <----part_[size-1]-----
```

- `sn fileName` clientul trimite serverului un fisier `fileName` din directorul curent (serverul il scrie in fisierul cu numele precedat de `"new_"` din directorul executabilului)

```
Client                       Server
      -----sn fileName------>
      ----size-------------->
      ----part_1------------>
      ----part_2------------>
      ----part_3------------>
      ----........---------->
      ----part_[size-1]----->
```

- `exit exit` serverul isi termina executia

```
Client                       Server
      -----exit exit----->
```

Oricate din aceste comenzi pot fi primite in orice ordine pana cand se primeste comanda `exit exit`. Aceste

comenzi sunt atomice, nu se pot intercala.

Tema trebuie sa permita rulare in 3 moduri diferite:

- modul simplu folosind protocolul *STOP AND WAIT* – fiecare pachet este urmat de un pachet ACK, fie ca este un pachet de la client, fie ca este de la server
  folosind acest procol comunicatia pentru ls ar deveni

```
 Client              Server
      -----ls .------->
      <----ACK---------
      <----num---------
      -----ACK-------->
      <----file_0------
      -----ACK-------->
      <----file_1------
      -----ACK-------->
      <----file_2------
      -----ACK-------->
      <----....--------
      -----ACK--------->
      <----file_[num-1]-
      -----ACK--------->
```

  Acest lucru se intampla cu toate comenziile prezentate mai sus, fiecare transmisie fortand un ACK in directia opusa
- modul de *verificare paritate si retransmitere* – pastreaza protocolul STOP AND WAIT dar primul byte din fiecare pachet este folosit ca bit de paritate, (pachetul poate acum avea dimensiune maxima 1399) daca este detectat un pachet eronat se cere retransmiterea lui prin inlocuirea ACK cu NACK

```
+-----------------------------------+
| Parity byte|Byte 0|Byte 1|....|Byte N|
+-----------------------------------+
```

  Din parity byte doar bitul cel mai din dreapta este folosit.
- modul de *transmitere cu metoda hamming si corectie* – pastreaza protocolul STOP AND WAIT dar fiecare byte din pachetul trimis va trebui "codat" folosind metoda hamming prezentata in laboratorul 3, fiecare byte va ocupa astfel 2 bytes (mai exact 12 biti) , se vor putea astfel transmite pachete de maxim 700 de bytes. La primire fiecare grup de 2 bytes trebuie corectat urmat de decodarea acestora in byte-ul initial.
  Datele codate hamming pe trebuie sa aiba urmatoarea forma (in biti):

```
+----------------------+
|0 |1 |2 |3 |4 |5 |6 |7 |
+----------------------+
|D1|D2|D3|D4|D5|D6|D7|D8|
+----------------------+
```

  Devine HiHi, adica (in biti):

```
+-----------------------------------------------+
|0 |1 |2 |3 |4 |5 |6 |7 |0 |1 |2 |3 |4 |5 |6 |7 |
+-----------------------------------------------+
|  |  |  |  |P1|P2|D1|P4|D2|D3|D4|P8|D5|D6|D7|D8|
+-----------------------------------------------+
```

  Aici P1,P2,P4,P8 sunt biti de paritate, iar D1-8 sunt biti din byte-ul initial.
  Pachetul va avea urmatoarea forma (in bytes):

```
+-------------------------------------------------+
|0 |1 |2 |3 |4 |5 |6 |7 |8 |.............|N*2|N*2+1|
+-------------------------------------------------+
|H0|H0|H1|H1|H2|H2|.....................|HN |HN   |
+-------------------------------------------------+
```

Aici HiHi are forma de mai sus si reprezinta codarea primului byte de date in 2 bytes.
Testarea temei se va face ruland serverul scris de voi alaturi de clientul OFERIT. Din acest motiv trebuie sa respectati cu strictete formatele prezentate in enunt.

Mentiuni

- dirname, filename acest lucru se aplica tuturor numelor de fisiere nu pot contine spatii si nu au o lungime mai mare de 255 de caractere.
- nu aveti voie sa modificati documentul lib.h
- se presupune ca toate caile dirName exista si toate fisierele fileName exista
- fisierul poate incapea in memoria programului (este de ordinul 1-2 MB)
- nu este necesar decat scrierea serverului dar va recomandam sa implementati si un client al vostru pentru o mai buna intelegere a algormilor si eficientizare a debug-ing-ului
- functiile de send si recv sunt blocante, aveti grija ca serverul sa fie o oglinda a clientului, sa respectati perfect protocolul.
- si pachetele ACK/NACK pot fi corupte, pentru a diferentia intre cele 2 verificati marimea pachetului, aceasta NU poate fi corupta
- toate comenziile sunt formate din 2 cuvinte, pentru a lista directorul curent se foloseste ls .
- in listare tuturor fisierelor dintr-un director se considera si . alaturi de ..
- alaturi de tema aveti executabilul clientului si fisierul .o al acestuia pentru a-l putea compila pe orice sistem
- clientul si serverul trebuie sa primeasca parametrii parity, hamming sau nimic, verificati testele pentru a confirma
- aveti la dispozitie un set de teste, sunteti indemnati sa le extindeti pe acestea
- sunteti indemnati sa folositi doar libraria oferita si limbajul C, temele vor fi testate automat

Hints

Va recomandam sa folositi:

- sprintf
- sscanf
- strlen
- sizeof
- memcpy
- chdir
- opendir
- readdir
- fseek
- ftell
- rewind
- fread
- fwrite
- exit

Nu uitati de operatiile binare

- setarea unui bit
  `number |= 1 << x;`
- stergerea unui bit
  `number &= ~(1 << x);`
- schimbarea unui bit
  `number ^= 1 << x;`

- extragerea valorii unui bit
  `bit = (number >> x) & 1;`

Unde number e numarul cu care lucrati, iar x este pozitia bitului de interes

Aveti extrem de mare grija la:

- stringuri vs date binare (majoritatea functiilor ce incep cu s intorc rezultat relativ la \n)
- numarul de bytes a unei valori si numarul de biti, e foarte usor sa faceti overflow.