MAC448 – PRC
BCC – 1º Semestre de 2009
Exercício-Programa I – Servidor http – Bolsa de Valores – entregar 13Abr09

O objetivo deste servidor http é apresentar ao browser quando consultado, uma tabela com a posição de 5 ações pré-definidas negociadas em bolsa.

A posição inicial é obtida de um arquivo `Posicao.txt` cujo conteúdo é o seguinte:

```
GGBR4     11.67     11.80     0         0
ITSA4      7.46      8.00     0         0
PETR4     27.70     28.00     0         0
USIM5     24.50     26.00     0         0
VALE5     28.00     29.00     0         0
```

A cada consulta pelo browser é enviada a ele a seguinte tabela:

Ação	Valor de Compra	Valor de Venda	Quantidade Total Negociada	Valor Total Negociado
GGBR4	11.67	11,80		
ITSA4	7.46	8.00		
PETR4	27.70	28.00		
USIM5	24.50	26.00		
VALE5	28.00	29.00		

As duas últimas colunas se iniciam zeradas, mas no decorrer do dia, à medida que as operações de compra e venda vão se realizando os valores são incrementados pelo servidor.

Para isso, leia o arquivo `Posicao.txt` e armazene os dados em 5 vetores:
String ACAO[5], float [5] VC, VV, VTN, int [5] QTN

Além de consultas de browsers, o servidor recebe de um outro cliente mensagens de texto com o seguinte conteúdo:

```
<Ação> <operação> <valor> <quantidade>
```

Exemplos: (C=compra e V=venda)

```
PETR4    C    28.00      1000
VALE5    V    28.30      5000
USIM5    C    24.60      1000
PETR4    C    27.90     10000
VALE5    C    28.20       500
```

A partir de cada uma destas mensagens recebidas, o servidor deve atualizar a tabela em memória.

Este cliente simula o pregão diário da Bolsa, criando mensagens aleatórias para o servidor. Como sugestão, usar `random()` para gerar:

a) $0 \le i \le 4$ indice em ACAO[5] – qual a ação que será operada
b) 0 ou 1 C ou V (compra ou venda)
c) $0 \le x \le 2$ novo valor de compra/venda = valor anterior +- x% (mais ou menos alternados)
d) $0 \le y \le 10$ nova quantidade negociada = y*1000

O servidor atende a 2 tipos de clientes. Portanto tem que ser multithreading. Na verdade além do programa principal que fica esperando a conexão de clientes (browsers), uma thread que fica esperando a próxima mensagem de atualização do cliente que simula o pregão.

Em resumo, o servidor terá a seguinte estrutura:

Principal:
```
Leia o arquivo Posicao.txt e guarde nas tabelas(ACAO,VC, VV, QTN, VTN)
Dispare as Threads abaixo;
while (true) {
   Espere conexão de cliente;
   Envie tabela;
}
```

Thread1:
```
// recebe mensagens de operações e atualiza tabelas
while (true) {
   Espere chegar msg;
   Atualize tabela (ACAO,VC, VV, QTN, VTN);
}
```

Thread2:
```
// grava o arquivo Posição.txt atualizado
while (true) {
   Espere 30 segundos;
   Formate em texto as tabelas (ACAO,VC, VV, QTN, VTN);
   Grave Posição.txt
}
```

O cliente gerador de mensagens terá a seguinte estrutura:

Cliente:
```
// gera mensagens de operação e envia ao servidor
while (true) {
   Gere nova mensagem no formato:
        <Ação> <C ou V> <Valor de C ou V> <Quantidade>
   Envie ao servidor;
   Espere t segundos
   // t pode ser um valor aleatório (ex=entre 1 e 20 seg.)
}
```

Teste seu programa, executando o servidor na mesma máquina do navegador e acesse a página através do endereço "localhost" (endereço IP 127.0.01). O seu programa será testado assim também. Procure testar seu servidor com diferentes navegadores. Internet Explorer, Mozilla Firefox são os mais recomendados. O navegador acessará o servidor como:

http://127.0.0.1 ou http://localhost

O Linux não permite que usuários comuns utilizem portas abaixo da 1024. Portanto a porta 80 não pode ser usada. Usando por exemplo a porta 8080, o navegador poderá acessar as páginas do seu servidor da seguinte forma:

http://localhost:8080 ou http://<nome da máquina>:8080

O servidor acima atende em 2 portas: a porta 80 ou 8080 e uma outra porta (exemplo 8888) para atender

ao cliente gerador de mensagens.

Para a consulta da tabela o servidor atende a cliente browser. Portanto vai receber um **GET**.
Para retornar a mensagem ao browser basta enviar os parâmetros:

```
HTTP/1.1 200 xxxxx
Content-Type: text
Content-Length: NNN
<linha vazia>
```