# Trabalho Prático 3 – Redes de Computadores

Fernanda Guimarães – 2016058166

## 1 Framework escolhido

Foi utilizado `poll` para o servidor e `threads` no cliente. Escolhi o poll devido a sua excelente documentação. Já as `threads` foram usadas para resolver o problema do cliente de poder digitar e receber mensagens ao mesmo tempo, e por serem mais simples de implementar.

Assim, foram utilizadas threads apenas em duas funções:

```
int get_message(int sock);
int send_input(int sock, const std::string & userName);
```

## 2 Identificadores de Mensagem

O cliente seta alguns delimitadores no início de cada mensagem para o servidor.

| Unicode | Significado |
|---|---|
| PM | Mensagem privada |
| ENQ | Listar usuários |
| STX | Broadcast |
| ACK | Novo usuário |

## 3 Instrucões

Primeiro, compile os arquivos com o comando `make`.

Após isso, execute o servidor com uma porta (no exemplo, 9000):

```
./servidor 9000
```

Agora, basta criar clientes na mesma porta:

```
./cliente localhost 9000
```

O endereço é agnóstico à natureza, aceitando tanto nome de domínios quanto ipv6 ou 4.

### 3.1 Listar usuários

Comando:

```
users
```

A primeira coisa que um cliente deve fazer é digitar um nome de usuário. Antes disso, é mantido como `undefined`.

### 3.2 Mensagem privada

Para mandar uma mensagem privada (`unicast`), bast digitar o comando:

```
uni;usuárioX;msg
```

### 3.3 `Broadcast`

Para mandar uma mensagem a todos usuários, incluindo a si mesmo, basta digitar o comando:

```
all;msg
```

### 3.4 Sair

Comando:

```
exit
```

## 4 Observações

- Clientes infinitos, setados dinamicamente;
- Tamanho máximo de mensagem `80`;
- Servidor usa `poll`, cliente usa `threads`;
- Agnóstico a `IPV6` e `IPV4`.
- Valores de constantes se encontram no arquivo de configuração `lib/config.cpp`.