SISTEMAS DISTRIBUÍDOS

Atividade 1 – RPC (Remote Procedure Call)

**Grupo:**

Diogo Christovão Cortez R.A: 12049102

Luiz Henrique Cruz Junior R.A: 12110797

Matheus F. Nishida R.A: 12212692

Vinicius A. Marques R.A: 12643748

Professor: Edmar Rezende

Data: 23/10/2015

**INTRODUÇÃO**

O RPC (Remote Procedure Call) define um protocolo para execução remota de procedures em computadores ligados em rede. O protocolo RPC pode ser implementado sobre diferentes protocolos de transporte (Na atividade 1 foi utilizado o UDP). Não cabe ao RPC especificar *como*a mensagem é enviada de um processo para outro, mas somente especificá-la (com XDR) e interpretá-la. A sua implementação depende, portanto, de sobre qual protocolo de transporte vai operar.

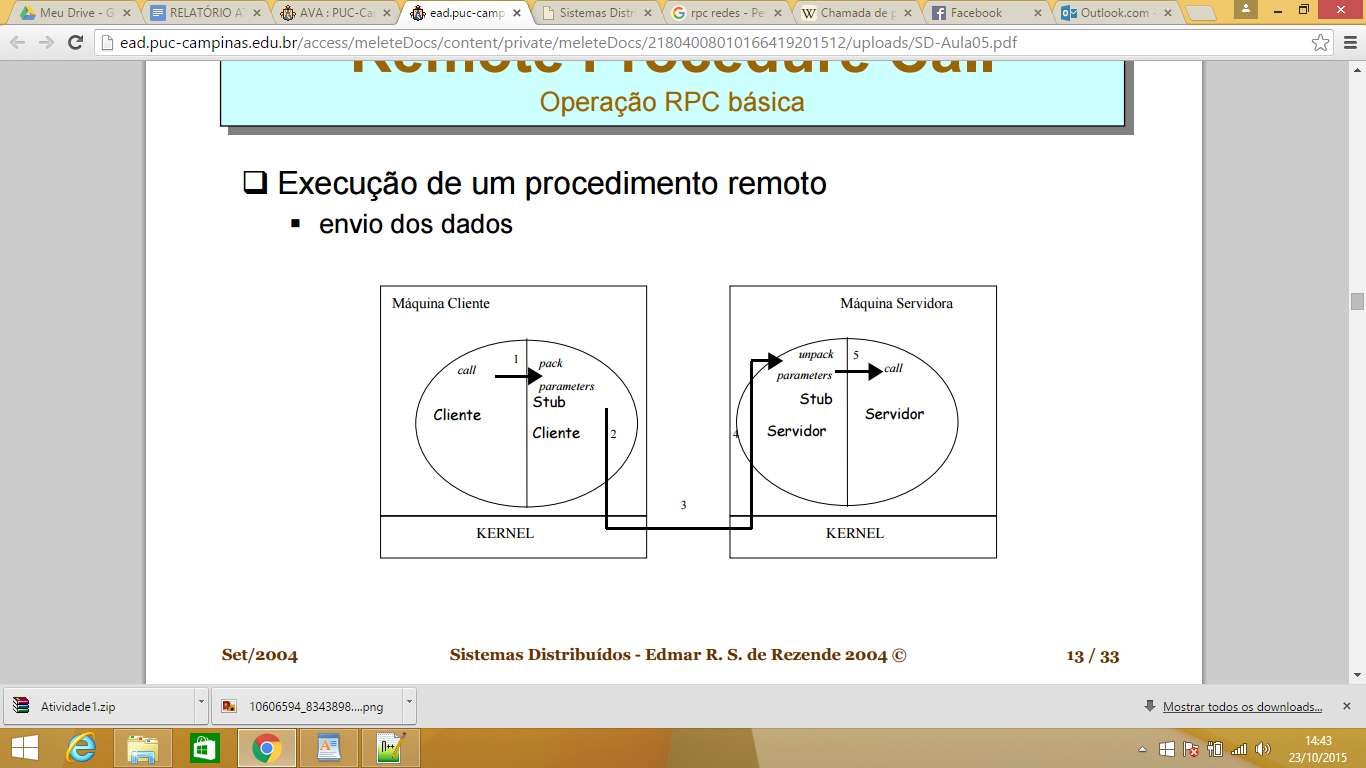
* sobre TCP Não é necessário preocupar-se com time-outs, retransmissões, duplicatas.
* sobre UDP é necessário preocupar-se com time-outs, retransmissões, duplicatas.

Uma mensagem RPC tem três campos inteiros:

* Remote Program Number;
* Remote Program Version Number;
* Remote Procedure Number.

Além, é claro, dos parâmetros específicos à chamada. A operação do RPC pode ser descrita nos seguintes passos:

* Coleta os dados dos parâmetros;
* Forma a mensagem;
* Envia a mensagem;
* Espera a resposta;
* Devolve a resposta através dos parâmetros.



**[Representação Remote Procedure Call (RPC)]**

**DETALHES DE IMPLEMENTAÇÃO RELEVANTES**

- As informações de cadastro são armazenadas na seguinte struct:

struct cadastro {

char user[10];

char mens[50];

in top, p\_livre\_ocupado, msg\_apagada, qtde\_mensagem;

};

- O menu é exibido dentro da *main* do programa *atividade\_client*, dentro de um loop infinito, que será fechado apenas se o usuário selecionar a opção 4 (*Sair da Aplicação*);

- As comparações com os retornos da Chamada de Procedimento Remoto são feitos todos através de uma função *void atividade\_1*, no programa *atividade\_client*;

- No programa *atividade\_server* recebemos a opção selecionada pelo usuário, que pode variar entre cadastrar, ler e apagar mensagem, além de também possibilitar a saída da aplicação. Neste ponto é importante ressaltar que, dependendo da opção selecionada, o retorno da função *cad\_mens\_1\_svc* pode variar.

**DESCRIÇÃO DO PROCESSO DE COMPILAÇÃO**

Primeiramente foi necessário entender como funcionava todo o protocolo RPC, sua funcionalidades e particularidades, para depois podermos compila-lo.

Após estuda-lo, instalamos o pacote “*rpcbind*” através do comando “*sudo apt-get install rpcbind*” no Terminal e, utilizando o arquivo com extensão *.x*, criamos todo o esqueleto que seria utilizado posteriormente para a atividade com o comando “*rpcgen atividade.x*”.

Após a execução deste comando, vários arquivos foram criados, são eles:

*- Makefile.atividade;*

*- atividade.h;*

*- atividade\_client.c;*

*- atividade\_clnt.c;*

*- atividade\_server.c;*

*- atividade\_svc.c ;*

*- atividade\_xdr.c.*

Após a implementação dos respectivos códigos, passamos para a próxima etapa, que seria a de compilar os arquivos gerados pelo comando do *rpcgen*. Segue abaixo os comandos (em ordem) que foram utilizados no Terminal para a compilação da Atividade proposta:

*- cc –c atividade\_xdr.c*

Compilando o servidor (modo root [*sudo su*]):

*- cc –c atividade\_server.c*

*- cc –c atividade\_svc.c*

*- cc –o servidor atividade\_server.o atividade\_svc.o atividade\_xdr.c*

*- ./servidor*

Compilando o cliente:

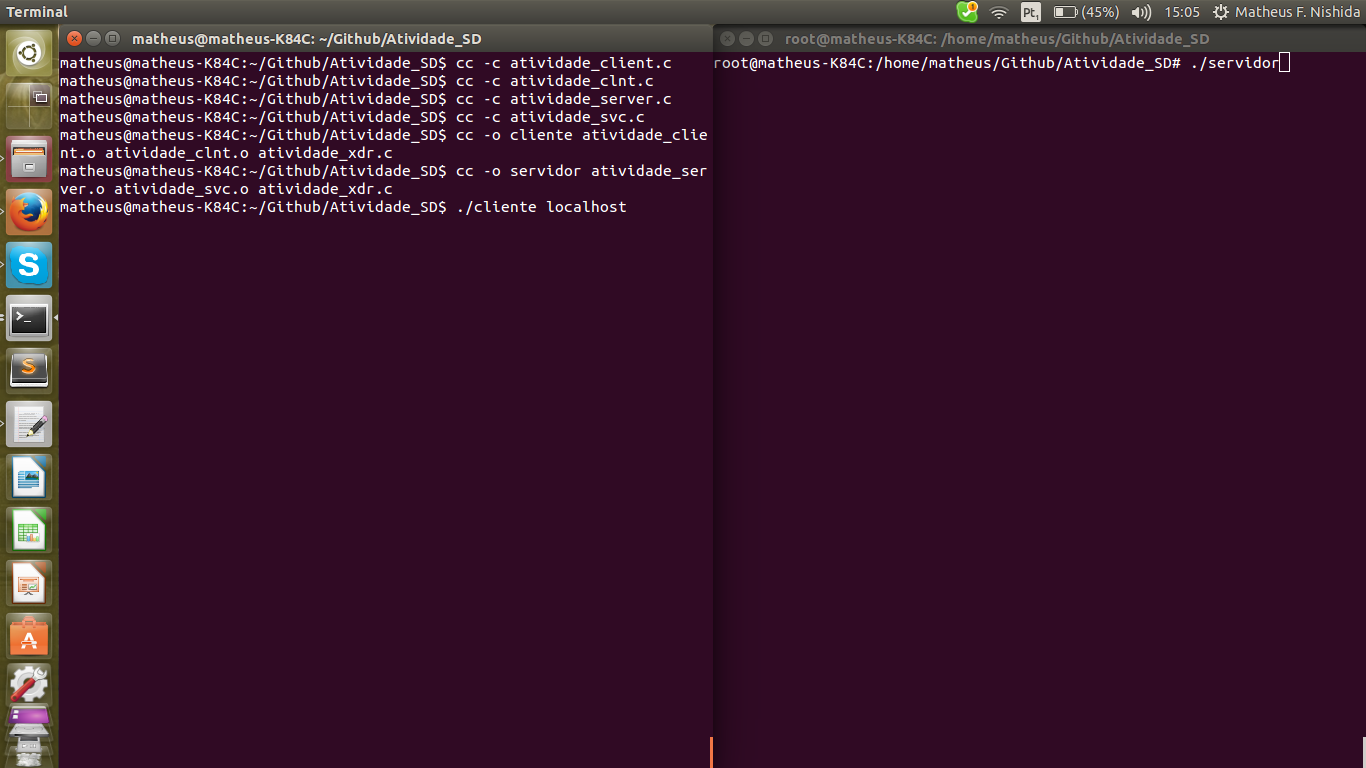
*- cc –c atividade\_client.c*

*- cc –c atividade\_clnt.c*

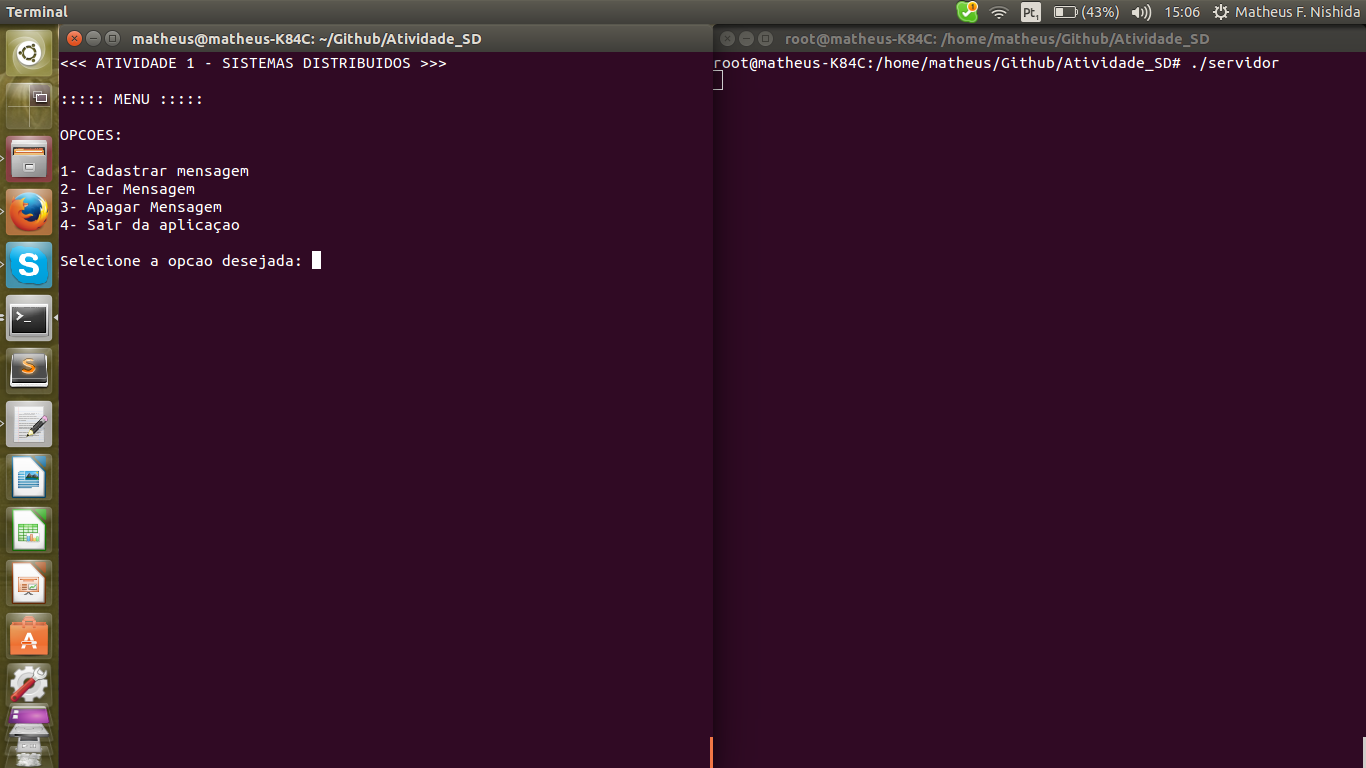
*- cc –o cliente atividade\_cliente.o atividade\_clnt.o atividade\_xdr.c*

*- ./cliente localhost*

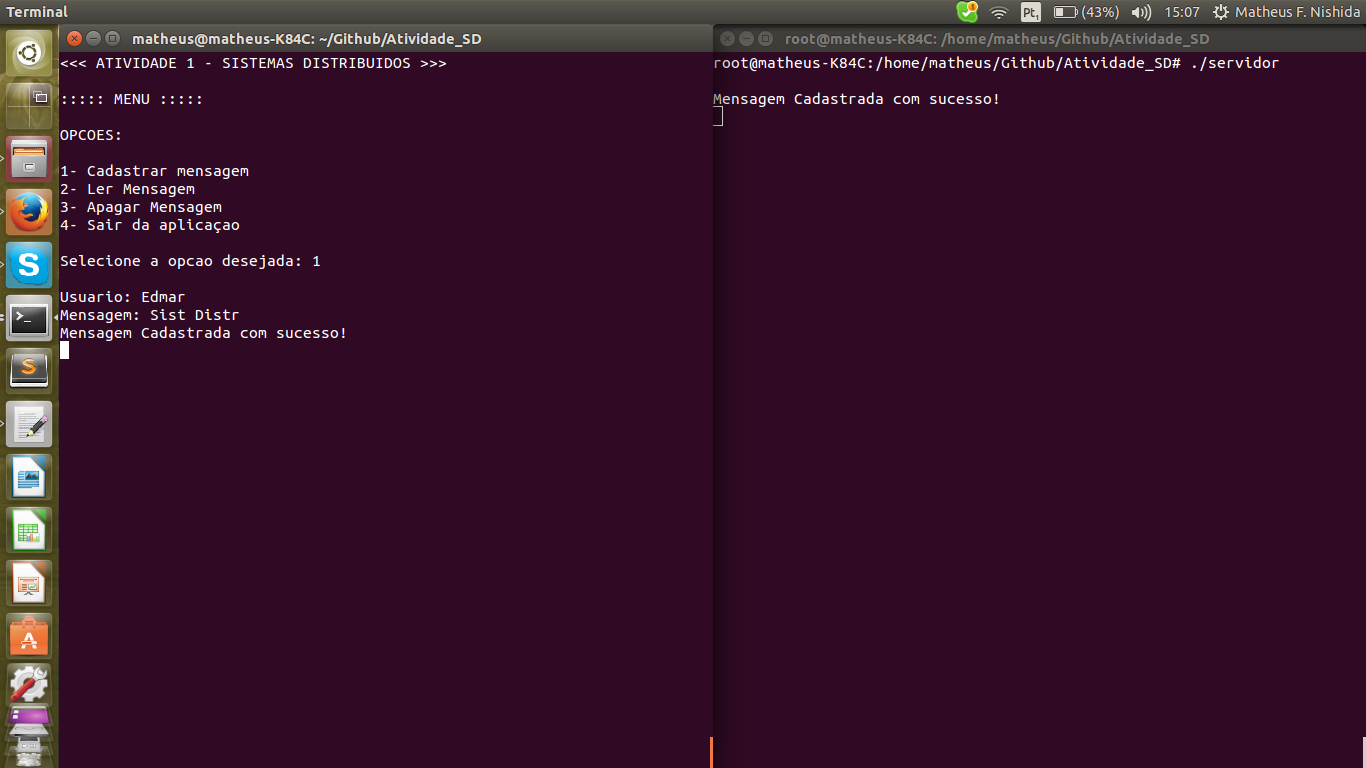
**DESCRIÇÃO DOS TESTES REALIZADOS**



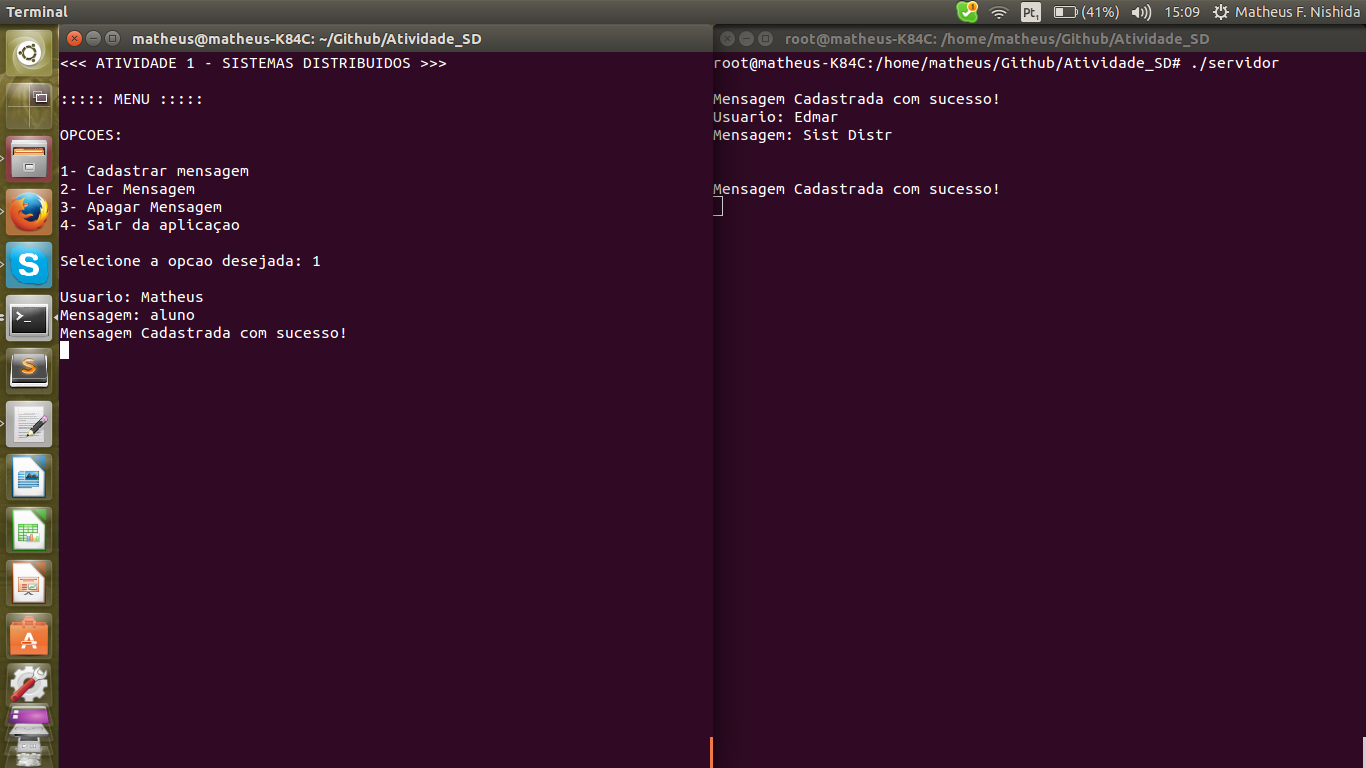
COMPILAÇÃO DOS COMANDOS



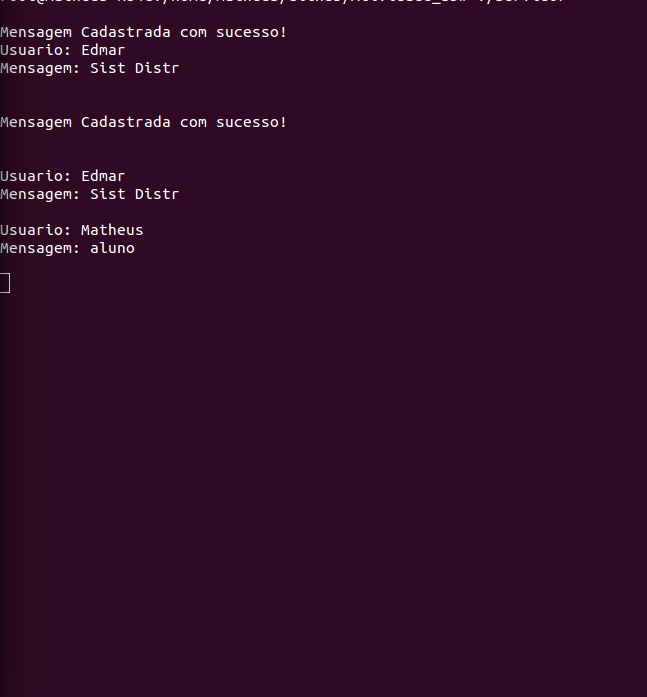
MENU



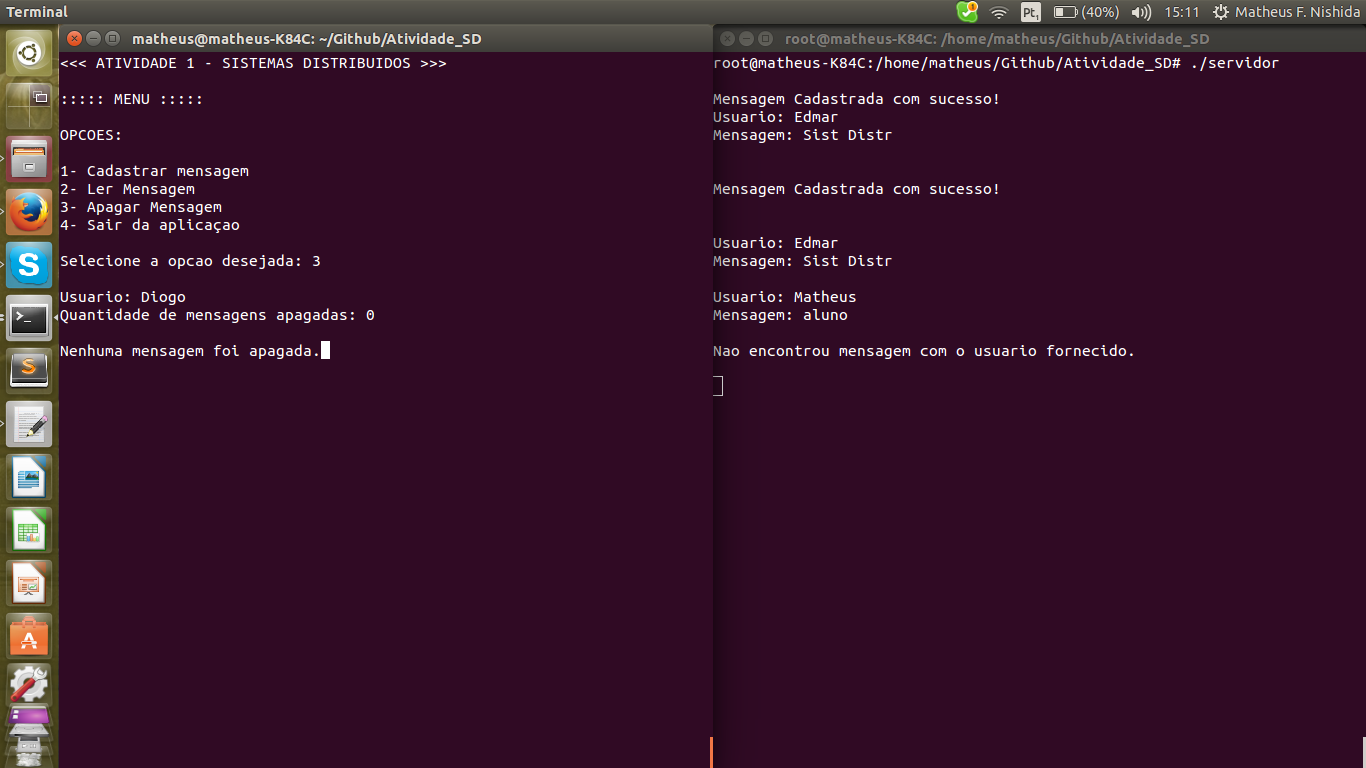
CADASTRAR-1



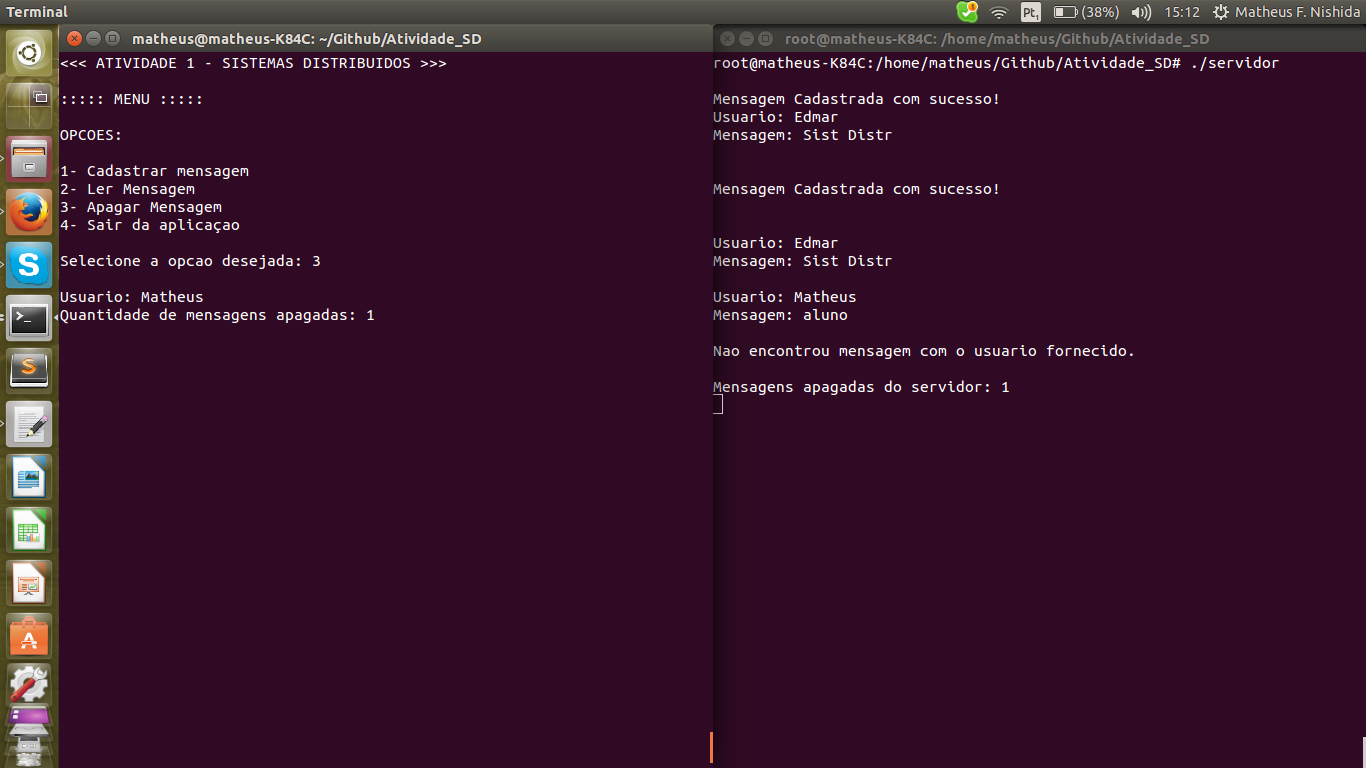
CADASTRAR-2



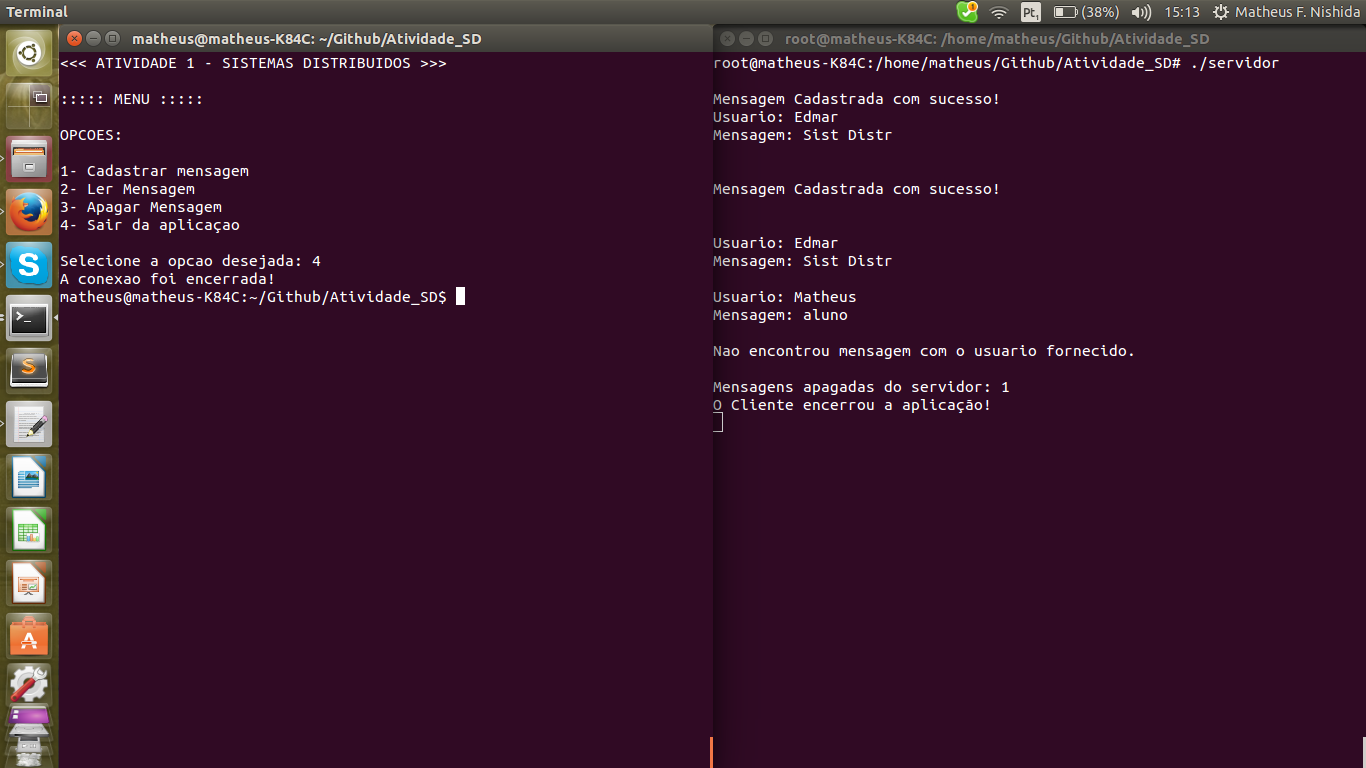
LER MENSAGENS



APAGAR\_COM\_ERRO



APAGAR\_ACERTO



SAIR\_APLICACAO