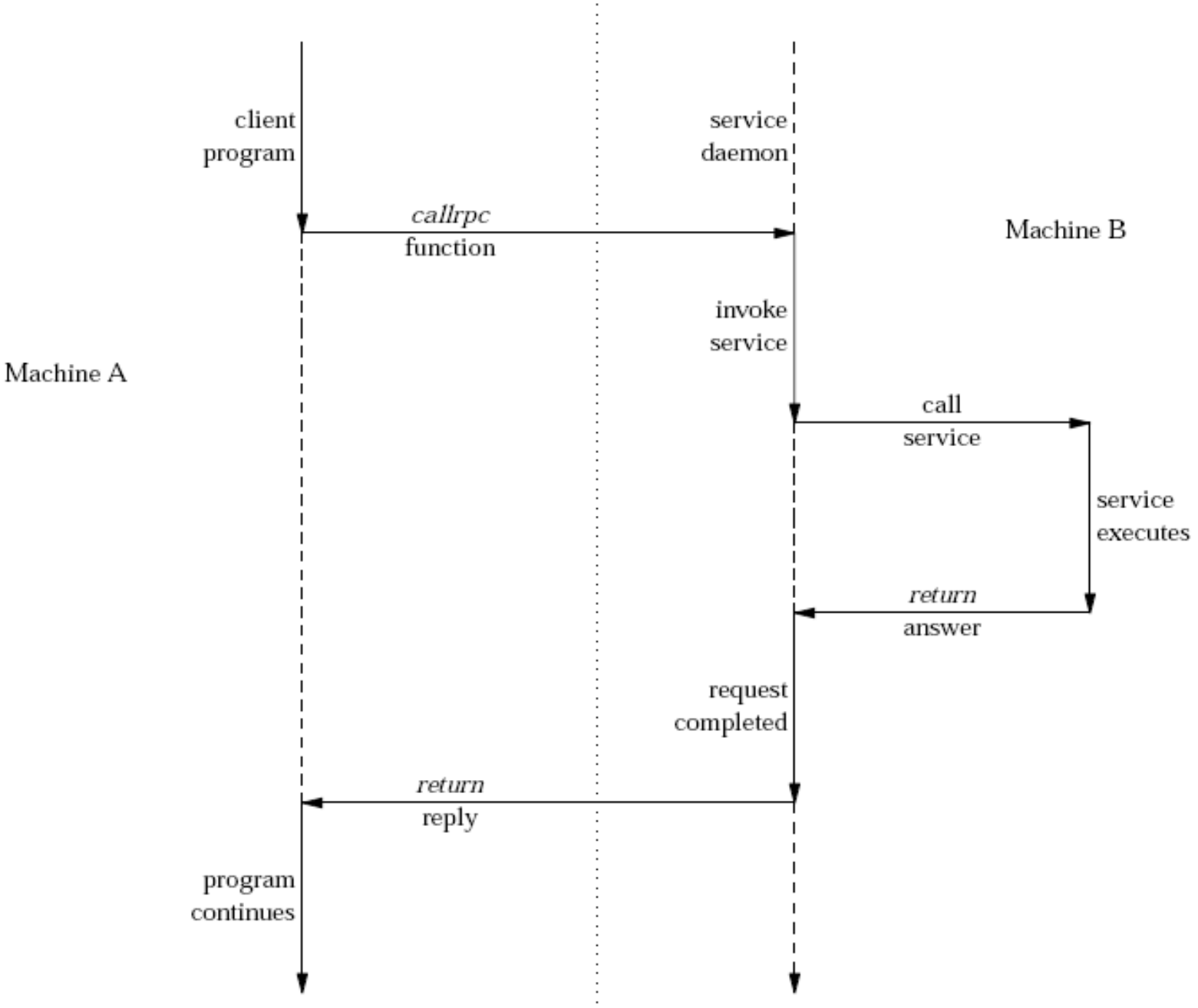
Exemplos de aplicações

RPC / RMI

RPC


client
program
service
daemon
callrpc
function
Machine B
invoke
service
Machine A
call
service
service
executes
return
answer
request
completed
return
reply
program
continues

RPC

- 3 níveis
- Alto (rnusers)
- Intermediário
- Baixo

Exemplos:

rnusers
rusers
havedisk
rstats
rwall
yppasswd

```
#include <stdio.h>

main(int argc,char  *argv[])
{
	int num;

	if ((num = rnusers(argv[1])) < 0) {
		fprintf(stderr, "Error: rnusers\n");
		exit(-1);
	}

	printf("%d users on %s\n", num, argv[1]);
	exit(0);
}
```

RPC

```
#include <stdio.h>
#include <rpc/rpc.h>
#include <utmp.h>
#include <rpcsvc/rusers.h>

main(int argc, char *argv[])
{
	unsigned long nusers;
	int stat;
	if (argc != 2) {
		fprintf(stderr, "usage: nusers hostname\n");
		exit(-1);
	}
	if (stat = callrpc(argv[1],RUSERSPROG,
				RUSERSVERS, RUSERSPROC_NUM,
				xdr_void, 0, xdr_u_long, &nusers) != 0)
	{
		clnt_perrno(stat);
		exit(1);
	}
	printf("%d users on %s\n", nusers, argv[1]);
	exit(0);
}
```

RPCGEN

- Linguagem RPC
- Diminuir o trabalho de depurar código referente à rede
- Código relacionado com XDR

```
program-definition:
"program" program-ident "{"
        version-list
"}" "=" value

version-list:
version ";"
version ";" version-list

version:
"version" version-ident "{"
        procedure-list
"}" "=" value
```

```
procedure-list:
procedure ";"
procedure ";" procedure-list

procedure:
type-ident procedure-ident "(" type-ident
")" "=" value
```

RPCGEN

```
struct info_arq{
        string nome<256>;
        int tamanho;
         struct info_arq *proximo;
};

program DIRPRO{
        version DIRVER{
                struct info_arq LERDIR(string) = 1;
        } = 1;
} = 2;
```

- rpcgen dir.x
  - dir_xdr.c
  - dir_svc.c
  - dir_clnt.c
  - dir.h

Servidor

```
#include <rpc/rpc.h>
#include <sys/dir.h>
#include "dir.h"

struct info_arq * lerdir_1_svc(
                        char **nome, struct svc_req *rq)
{
        DIR *ptr;
        struct direct *d;
        static info_arq informacao;
        info_arq *info_ptr;

        ptr = opendir(*nome);

        if (ptr == NULL) {
                strcpy(informacao.nome,"Sem arquivos");
                informacao.proximo=NULL;
                informacao.tamanho=0;
                return (&informacao);
        }

        xdr_free(xdr_info_arq, &informacao);
```

```
        if(d = readdir(ptr))
        {
                informacao.nome = malloc(sizeof(char)*256);
                strncpy(informacao.nome, d->d_name,256);
                informacao.tamanho = d->d_off;
                informacao.proximo = NULL;
                info_ptr = &informacao;

                while (d = readdir(ptr)) {
                        info_ptr->proximo =
                                malloc(sizeof(info_arq));
                        info_ptr->proximo->nome =
                                malloc(sizeof(char)*256);
                        strncpy(info_ptr->proximo->nome,
                                d->d_name,256);
                        info_ptr->proximo->proximo = NULL;
                        info_ptr->proximo->tamanho = d->d_off;
                        info_ptr = info_ptr->proximo;
                }
                closedir(ptr);
                return (&informacao);
        }
}
```

Interface

```
#ifndef _DIR_H_RPCGEN
#define _DIR_H_RPCGEN

#include <rpc/rpc.h>


#ifdef __cplusplus
extern "C" {
#endif


struct info_arq {
        char *nome;
        int tamanho;
        struct info_arq *proximo;
};
typedef struct info_arq info_arq;

#define DIRPRO 2
#define DIRVER 1

#if defined(__STDC__) || defined(__cplusplus)
#define LERDIR 1
extern  struct info_arq * lerdir_1(char **, CLIENT *);
extern  struct info_arq * lerdir_1_svc(char **, struct svc_req *);
extern int dirpro_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);
```

```
#else /* K&R C */
#define LERDIR 1
extern  struct info_arq * lerdir_1();
extern  struct info_arq * lerdir_1_svc();
extern int dirpro_1_freeresult ();
#endif /* K&R C */

/* the xdr functions */

#if defined(__STDC__) || defined(__cplusplus)
extern  bool_t xdr_info_arq (XDR *, info_arq*);

#else /* K&R C */
extern bool_t xdr_info_arq ();

#endif /* K&R C */

#ifdef __cplusplus
}
#endif

#endif /* !_DIR_H_RPCGEN */
```

Cliente

```
#include <stdio.h>
#include <rpc/rpc.h>
#include "dir.h"

int main(int argc,char *argv[])
{
	CLIENT *cl;
	info_arq *resultado;

	cl = clnt_create(argv[1], DIRPRO, DIRVER, "tcp");
	if (cl == NULL) {
		clnt_pcreateerror(argv[1]);
		exit(1);
	}

	resultado = lerdir_1(&argv[2], cl);
	if (resultado == NULL) {
		clnt_perror(cl, argv[1]);
		exit(1);
	}

	if (resultado->proximo == NULL) {
		printf("Erro: %s\n",resultado->nome);
		exit(1);
	}

	while(resultado!=NULL)
	{
		printf("Entrada: %s offset: %d\n",resultado->nome, resultado->tamanho);
		resultado = resultado->proximo;
	}

	exit(0);
}
```

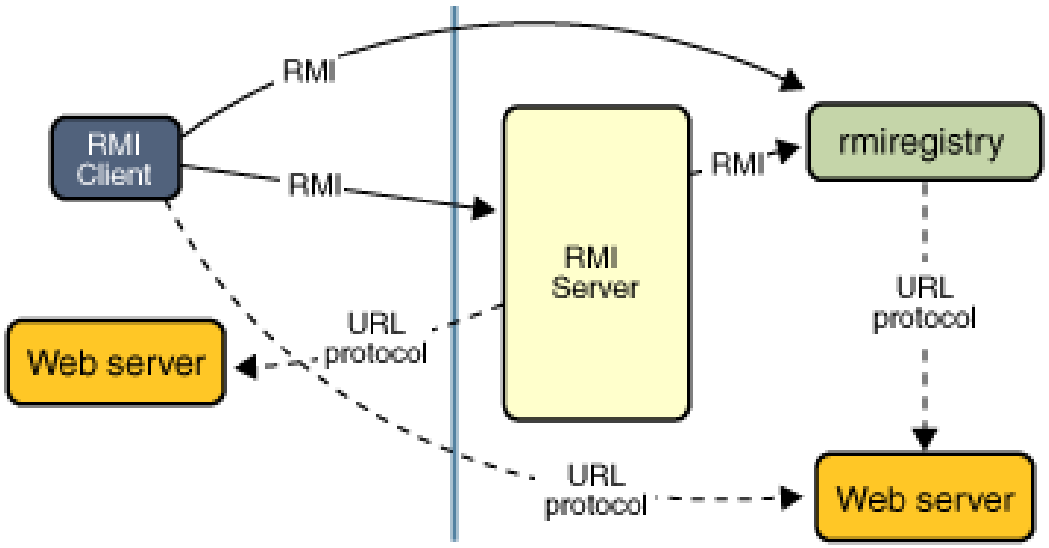
Compilação

- Servidor
  - gcc -o dir_servidor dir_servidor.c dir_xdr.c dir_svc.c
- Cliente
  - gcc -o dir_cliente dir_cliente.c dir_xdr.c dir_clnt.c

Execução

RMI

- Servidor
  - Implementação
- Interface
  - “Prototipação”
- Cliente
  - Instância do objeto



RMI

- Primeira aplicação
  - eco
- Compilador

```
package srvd;

import java.rmi.*;

public interface Imprimir extends Remote
{
	public int imprimir(String mensagem, int i) throws RemoteException;
	public void atribuir_valor(int i) throws RemoteException;
}
```

Servidor

- Implementação

```
package srvd;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ImprimirImplementado extends
                UnicastRemoteObject implements Imprimir
{
        private int valor=0;

        public ImprimirImplementado() throws RemoteException
        {
                super();
        }

        public int imprimir(String mensagem, int vezes)
        {
                try
                {
                        FileWriter fw = new FileWriter("/tmp/imprimir.txt");
                        PrintWriter pw = new PrintWriter(fw);

                        int i;
                        int fim = (vezes>valor)?vezes:valor;
                        for(i=0;i<fim;i++)
                        {
                                pw.write(mensagem);
                        }

                        pw.close();
                        fw.close();

                        return i;
                }catch(Exception e)
                {
                        System.out.println(e);
                }
                return 0;
        }

        public void atribuir_valor(int i)
                        throws RemoteException
        {
                valor=i;
        }
}
```

Cliente

- Cliente

```
package clnt;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.NotBoundException;
import java.net.MalformedURLException;
import srvd.Imprimir;

public class ImprimirCliente {

	public static void main( String args[] )
	{
		try {
			Imprimir i = (Imprimir) Naming.
					lookup( "rmi://localhost/Imprimir" );
			i.atribuir_valor(10000);
			System.out.println( i.imprimir("Oi",1000) );

			Imprimir i2 = (Imprimir) Naming.
					lookup( "rmi://localhost/Imprimir" );
			i2.atribuir_valor(10);
			System.out.println( i2.imprimir("Oi",200) );

			System.out.println( i.imprimir("Oi",500) );
		}
		catch( MalformedURLException e ) {
			System.out.println();
			System.out.println(
			 "MalformedURLException: " + e.toString() );
		}
		catch( RemoteException e ) {
			System.out.println();
			System.out.println(
				"RemoteException: " + e.toString() );
		}
		catch( NotBoundException e ) {
			System.out.println();
			System.out.println(
				"NotBoundException: " + e.toString() );
		}
		catch( Exception e ) {
			System.out.println();
			System.out.println(
				"Exception: " + e.toString() );
		}
	}
}
```

Compilação

- Servidor
  - javac srvd/*.java
- Cliente
  - Javac clnt/*.java

Execução