

Exemplo usando RPC

- ◆ Envia e recebe um inteiro
 - ❖ Server - PORTMAPPER must be running
 - ❖ Client - PORTMAPPER and REMOTE SERVER must be running
- ◆ Envia e recebe um float
 - ❖ Server - PORTMAPPER must be running
 - ❖ Client - PORTMAPPER and REMOTE SERVER must be running
- ◆ Envia e recebe uma string
 - ❖ Server - PORTMAPPER must be running
 - ❖ Client - PORTMAPPER and REMOTE SERVER must be running

Pag. 1

Servidor

```
/* Server RPC - SRV-RPC.C File - Receive an integer, a float and a
   string and return them respectively */
/* PORTMAPPER MUST BE RUNNING */

/* Include Files */
#include <rpc/rpc.h>
#include <stdio.h>

/* Defines for INTEGER Function */
#define intrcvprog ((u_long)150000)
#define intvers    ((u_long)1)
#define intrcvproc ((u_long)1)

/* Defines for FLOAT Function */
#define fltrcvprog ((u_long)150102)
#define fltvrs    ((u_long)1)
#define fltrcvproc ((u_long)1)
```

Pag. 2

... Servidor

```
/* Defines for STRING Function */
#define strrcvprog ((u_long)150204)
#define strvers    ((u_long)1)
#define strrcvproc ((u_long)1)

/* Program Main */
main()
{
    /* Declaration of Functions */
    int *intrcv(); /* declaration of "intrcv" function */
    float *floatrecv( ); /* declaration of "floatrecv" function */
    char **strrcv( char** ); /* declaration of "strrcv" function */
```

Pag. 3

... Servidor

```
/* REGISTER PROG, VERS AND PROC WITH THE PORTMAPPER */

/* Program that sends an INTEGER to the remote host */
registerrpc( intrcvprog, intvers, intrcvproc,
             intrcv, xdr_int, xdr_int );
printf("INTRCV Registration with Port Mapper COMPLETED.\n\n");

/* Program that sends an FLOAT to the remote host */
registerrpc( fltrecvprog, fltvrs, fltrecvproc,
             floatrecv, xdr_float, xdr_float );
printf("FLOATRCV Registration with Port Mapper COMPLETED.\n\n");

/* Program that sends an STRING to the remote host */
registerrpc( strrcvprog, strvers, strrcvproc,
             strrcv, xdr_wrapstring, xdr_wrapstring );
printf("STRRCV Registration with Port Mapper COMPLETED.\n\n");
```

Pag. 4

... Servidor

```
/* Espera por chamadas aos procedimento registrados */
svc_run();

/* If "svc_run()" return error, print a message and exit */
printf("Error: SVC_RUN returned!\n");
exit(1);
}
/* End of Program Main */
```

Pag. 5

... Servidor

```
/* Function "intrcv()" */
int *intrcv(int *in)
{
    int *out;

    printf("Integer received: %d    ",*in);
    out = in;
    printf("Integer being returned: %d\n\n",*out);
    return (out);
}
```

Pag. 6

... Servidor

```
/* Function "floatrcv()" */
float *floatrcv(float *in)
{
    float *out;

    printf("Float received: %e    ", *in);
    out = in;
    printf("Float being returned: %e\n\n", *out);
    return(out);
}
```

Pag. 7

... Servidor

```
/* Function "strrcv()" */
char **strrcv(char **in)
{
    char **out; /* char *out = malloc(64); */

    printf("String received: %s    ", *in);
    /* strcpy(out, *in); */
    out = in;
    printf("String being returned: %s\n\n", *out);
    return (out);
}
```

Pag. 8

Cliente

```
/* Client RPC - CLT-RPC.C File */
/* Send an integer, a float and a string to the remote host and
   receive the integer, float and string back */
/* PORTMAPPER AND REMOTE SERVER MUST BE RUNNING */

/* Include Files */
#include <rpc/rpc.h>
#include <stdio.h>

/* Defines for INTEGER Function */
#define intrcvprog ((u_long)150000)
#define intvers    ((u_long)1)
#define intrcvproc ((u_long)1)
/* Defines for FLOAT Function */
#define fltrcvprog ((u_long)150102)
#define fltvrs    ((u_long)1)
#define fltrcvproc ((u_long)1)
```

Pag. 9

... Cliente

```
/* Defines for STRING Function */
#define strrcvprog ((u_long)150204)
#define strvers    ((u_long)1)
#define strrcvproc ((u_long)1)

/* Main Program */
main(int argc, char *argv[])
{
    int in_int, out_int; /* parameters for "intrcv" function */
    float in_flt, out_flt; /* parameters for "fltrcv" function */

    /* parameters for "strrcv" function */
    char *in_str = malloc(64);
    char *out_str = malloc(64);

    int error;
```

Pag. 10

... Cliente

```
/* how enter the arguments in a command line */
if( argc < 5 ) {
    fprintf(stderr,"Usage: clt-srv <hostname> <integer>
                <float> <string>\n");
    exit (-1);
} /* endif */

/* assign the arguments to the parameters */
in_int = atoi(argv[2]); /* ascii to integer */
in_flt = atof(argv[3]); /* ascii to float */
strcpy(in_str, argv[4]); /* copy argv[4] to in_str */
```

Pag. 11

... Cliente

```
/* call for "intrcv" function */
error = callrpc( argv[1], intrcvprog, intvers, intrcvproc,
                 xdr_int, (char *)&in_int,
                 xdr_int, (char *)&out_int);
if (error != 0) {
    fprintf(stderr,"ERROR: callrpc failed: %d \n",error);
    fprintf(stderr,"Program: %d Version: %d Procedure: %d",
            intrcvprog, intvers, intrcvproc);
    exit(1);
} /* endif */

printf("Integer sent: %d      Integer received: %d\n\n",
       in_int, out_int);
```

Pag. 12

... Cliente

```
/* call for "floatrcv" function */
error = callrpc(argv[1], fltrcvprog, fltvers, fltrcvproc,
                xdr_float, (char *)&in_flt,
                xdr_float, (char *)&out_flt);
if (error != 0) {
    fprintf(stderr,"ERROR: callrpc failed: %d \n",error);
    fprintf(stderr,"Program: %d Version: %d Procedure: %d",
            fltrcvprog, fltvers, fltrcvproc);
    exit(1);
} /* endif */

printf("Float sent: %e    Float received: %e\n\n",
       in_flt, out_flt);
```

Pag. 13

... Cliente

```
/* call for "strrcv" function */
error = callrpc( argv[1], strrcvprog, strvers, strrcvproc,
                xdr_wrapstring, (char *)&in_str,
                xdr_wrapstring, (char *)&out_str);
if (error != 0) {
    fprintf(stderr,"ERROR: callrpc failed: %d \n",error);
    fprintf(stderr,"Program: %d Version: %d Procedure: %d",
            strrcvprog, strvers, strrcvproc);
    exit(1);
} /* endif */

printf("String sent: %s    String received: %s\n\n",
       in_str, out_str);

exit(0);
}
/* end of Main Program */
```

Pag. 14