Número: Nome:

LEIC/LERC – 2008/09

Primeiro Teste de Sistemas Distribuídos

24 de Abril de 2009

Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas.

Duração: 1h30m

Grupo I [7,2 valores]

Considere uma aplicação cliente-servidor desenvolvida usando SUN RPC. Os extratos de alguns dos seus ficheiros de código são apresentados de seguida:

date.h

```
1 #define DATE_PROG ((u_long)0x31234567)
2 #define DATE_VERS ((u_long)1)
3 #define BIN_DATE ((u_long)1)
4 #define STR_DATE ((u_long)2)
```

client.c

```
1 #include "date.h"
2
3 int main(int argc, char **argv)
4 {
5   CLIENT *cl;
6   long *result;
7   char* server;
8   if (argc < 2) {
9       fprintf(stderr, "Server address missing\n", argv[0]);
10      exit(1);
11  }
12  server = argv[1];
13
14  cl = clnt_create(server, DATE_PROG, DATE_VERS, "tcp");
15  result = bin_date_1(NULL, cl);
16  if (result==NULL) {
17     printf("An RPC error has occurred while trying to call the remote procedure.");
18     exit(1);
19  }
20
21  printf("Current binary time is %d.\n", *result);
22  exit(0);
23}
```

date_clnt.c

```
1 #include "date.h"
2
3 long *bin_date_1(argp, clnt)
4       void *argp;
5       CLIENT *clnt;
6 {
7       static long res;
8
9       bzero((char *)&res, sizeof(res));
10      if (clnt_call(clnt, BIN_DATE, (xdrproc_t)xdr_void, argp, (xdrproc_t)xdr_long, &res, TIMEOUT) !=
11 RPC_SUCCESS) {
12              return (NULL);
13      }
14      return (&res);
15 }
16
17 char **str_date_1(argp, clnt)
18      long *argp;
19      CLIENT *clnt;
20 {
```

```
21      static char *res;
22
23      bzero((char *)&res, sizeof(res));
24      if (clnt_call(clnt, STR_DATE, (xdrproc_t)xdr_long, argp, (xdrproc_t)xdr_wrapstring, &res,
25   TIMEOUT) != RPC_SUCCESS) {
26              return (NULL);
27      }
28      return (&res);
29  }
```

date_proc.c

```
1  #include "date.h"
2
3  /* Return the binary date and time. */
4  long *
5  bin_date_1(){
6       static long    timeval;       /* must be static */
7       timeval = time((long *) 0); /* Unix function that returns current time */
8       return(&timeval);
9  }
10
11 /* Convert a binary time and return a human readable string. */
12 char **
13 str_date_1(long *bintime) {
14      static char    *ptr;          /* must be static */
15      ptr = ctime(bintime);         /* convert to local time */
16      return(&ptr);          /* return the address of pointer */
}
```

date_svc.c

```
1  #include "date.h"
2
3  static void date_prog_1();
4
5  main()
6  {
7       register SVCXPRT *transp;
8
9       transp = svcudp_create(RPC_ANYSOCK);
10      if (transp == NULL) {
11              fprintf(stderr, "cannot create udp service.");
12              exit(1);
13      }
14      if (!svc_register(transp, DATE_PROG, DATE_VERS, date_prog_1, IPPROTO_UDP)) {
15              fprintf(stderr, "unable to register (DATE_PROG, DATE_VERS, udp).");
16              exit(1);
17      }
18
19      transp = svctcp_create(RPC_ANYSOCK, 0, 0);
20      if (transp == NULL) {
21              fprintf(stderr, "cannot create tcp service.");
22              exit(1);
23      }
24      if (!svc_register(transp, DATE_PROG, DATE_VERS, date_prog_1, IPPROTO_TCP)) {
25              fprintf(stderr, "unable to register (DATE_PROG, DATE_VERS, tcp).");
26              exit(1);
27      }
28
29      svc_run();
30 }
31
32 static void
33 date_prog_1(rqstp, transp)
34      struct svc_req *rqstp;
35      SVCXPRT *transp;
36 {
37      union __svcargun {
38              long str_date_1_arg;
39      } argument;
40      char *result;
41      bool_t (*xdr_argument)(), (*xdr_result)();
42      char *(*local)();
43
44      switch (rqstp->rq_proc) {
45      case BIN_DATE:
```

Número: Nome:

```
46              xdr_argument = (xdrproc_t)xdr_void;
47              xdr_result = (xdrproc_t)xdr_long;
48              local = (char *(*)()) bin_date_1;
49              break;
50
51      case STR_DATE:
52              xdr_argument = (xdrproc_t)xdr_long;
53              xdr_result = (xdrproc_t)xdr_wrapstring;
54              local = (char *(*)()) str_date_1;
55              break;
56      default:
57              svcerr_noproc(transp);
58              return;
59      }
60      memset((char *)&argument,0, sizeof(argument));
61      if (!svc_getargs(transp, xdr_argument, &argument)) {
62              svcerr_decode(transp);
63              return;
64      }
65      result = (*local)(&argument, rqstp);
66      if (result != NULL && !svc_sendreply(transp, xdr_result, result)) {
67              svcerr_systemerr(transp);
68      }
69      if (!svc_freeargs(transp, xdr_argument, &argument)) {
70              fprintf(stderr, "unable to free arguments");
71              exit(1);
72      }
73      return;
74 }
```

1) [0,5 valores] Alguns dos ficheiros acima apresentados são totalmente gerados pelo compilador *rpcgen*, enquanto que outros precisam de ser codificados pelo programador da aplicação distribuída. Indique, de entre os ficheiros acima apresentado, quais estão no segundo caso.

2) [1,0 valores] Esboce, da forma mais correcta possível, o ficheiro exemplo.x, que define a IDL na qual este projecto se baseia.

3) [1,2 valores] Para cada um dos seguintes componentes do cliente, identifique em que parte dos excertos de código ela é implementada (indicando o nome do ficheiro e o número das linhas correspondentes):
   a. Binding do cliente ao servidor
   b. Chamada do procedimento remoto no programa cliente
   c. Stubs do cliente

4) [1,2 valores] Para cada um dos seguintes componentes do servidor, identifique em que parte dos excertos de código ela é implementada:
    a. Registo da interface no servidor de nomes

    b. Implementação dos procedimentos que o servidor oferece remotamente

    c. Invocação local do procedimento remoto pedido pelo cliente

5) [1,2 valores] "O SUN RPC permite distribuir uma aplicação anteriormente centralizada com total transparência." Rebata esta afirmação apontando **3 entraves** à transparência do RPC **que sejam evidentes no código apresentado** (incluindo o IDL por si esboçado). Ilustre cada entrave com exemplos do código. *(Atenção: não serão considerados entraves que não sejam evidentes no código apresentado acima.)*

6) Considere apenas o caso em que os procedimentos do servidor são invocados por TCP/IP.
Assuma que, o SUN RPC se limita a enviar cada pedido uma vez para o socket TCP e bloqueia-se à espera da resposta. Caso receba um erro na escrita/leitura do socket, o SUN RPC retorna erro à aplicação cliente.
    a. [0,7 valores] Qual a semântica de execução oferecida?

    b. [0,5 valores] Das seguintes faltas, assinale aquelas que, se ocorrerem, podem levar o RPC a oferecer uma semântica que é diferente da semântica de uma chamada local.
        i. Uma mensagem (pedido ou resposta) perdeu-se na rede.
        ii. Uma mensagem (pedido ou resposta) chegou em duplicado através da rede.
        iii. Servidor falhou temporariamente.

    c. [0,9 valores] Para cada falta assinalada na alínea anterior: justifique com um exemplo e proponha uma solução que solucione o caso que descreveu.

Número: Nome:

Grupo II [6,5 valores]

Considere o seguinte definição de um programa remoto:

```
import java.rmi*;
public interface Account extends Remote {
      float debit(float amount) throws RemoteException,
      InsufficientFundsException;
       float credit(float amount) throws RemoteException;
}
public interface AccountList extends Remote {
       Account getAccount(int id) throws RemoteException;
}
```

1) A definição da interface herda da classe **Remote**. Indique se concorda ou discorda das seguintes afirmações e justifique.
   a. [0,4 valores] Esta interface só pode ser usada por clientes em máquina remotas. Não pode ser invocada localmente.

   b. [0,4 valores] A única notação que é necessário para um método ser invocável remotamente é a sua interface herdar de **Remote**.

   c. [0,4 valores] **RemoteException** permite tratar as falhas decorrentes da distribuição.

Considere agora o programa de um cliente remoto que invoca objectos definidos de acordo com a interface anterior.

```
import java.rmi.*;
import java.rmi.server.*;
public class AccountListClient{
      public static void main(String args[]){
1      System.setSecurityManager(new RMISecurityManager());
2         AccountList acList = null;
3         try{
4          acList = (AccountList)Naming.lookup("//host/AccountList");
5          Account a = acList.getAccount(2);
6          a.debit(1000);
7       }catch(RemoteException e) {System.out.println(e.getMessage());
8      }catch(Exception e){System.out.println("Client:"+e.getMessage());}
```

```
        }
    }
```

Considere o cliente e em particular as estruturas de suporte ao programa

- Tabela de referências remotas
- Classes carregadas

2) Considere a linha número 4
   a. [0,5 valores] Descreva a operação executada nessa linha.

   b. [0,6 valores] Que modificação se efectua na tabela de referências remotas? Justifique.

   c. [0,4 valores] Que classe deve ser carregada como resultado dessa operação?

3) Considere a linha número 5
   a. [0,4 valores] Em que classe é invocado o método no cliente?

   b. [0,4 valores] Em que classe é realmente executado e em que máquina se executa essa classe?

4) Admita que o objecto número 2 do tipo Account era **um objecto local do servidor** (isto é, que não implementa a interface Remote, apenas implementa Serializable).
   a. [0,6 valores] O que deveria ser o resultado da invocação?

   b. [0,6 valores] Nesta situação que classe era carregada no cliente?

   c. [0,8 valores] Como pode o cliente saber a referência à classe?

   d. [0,4 valores] Em que máquina (cliente ou servidor) será executado o método debit (linha 6)?

Número: Nome:

5) O mecanismo de Java RMI tem inerentemente um recuperador de memória (Garbage Collector).
   a. [0,6 valores] Na linha 4 o que deve a máquina do cliente indicar ao GC local do servidor para garantir o correcto funcionamento deste?

Grupo III [6,3 valores]

Considere o documento WSDL que se apresenta fraccionado nas diversas perguntas:

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
              <complexType>
                  <all>
                      <element name="tickerSymbol" type="string"/>
                  </all>
              </complexType>
           </element>
           <element name="TradePrice">
              <complexType>
                  <all>
                      <element name="price" type="float"/>
                  </all>
              </complexType>
           </element>
       </schema>
    </types>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
```

1) [0,6 valores] Construa em Java a função cuja assinatura corresponde ao serviço definido do documento.

2) Considere que o serviço deveria retornar uma estrutura que para além do preço indica a quantidade de itens disponíveis para venda.
   a. [0,3 valores] Que secção deveria modificar?
   b. [0,4 valores] Reescreva a parte estritamente necessária

3) Considere que existe a possibilidade de a invocação falhar porque o item não existe e que pretende ter uma forma de o assinalar ao cliente
   a. [0,3 valores] Que secção deveria modificar?
   b. [0,4 valores] Reescreva a parte estritamente necessária

4) Namespaces – considere a linha input message="tns:GetLastTradePriceInput"
   a. [0,7 valores] Que função tem o namespace neste exemplo? Exemplifique um potencial erro que poderia surgir na interpretação de um documento semelhante a este, caso o namespace não fosse usado.
   b. [0,4 valores] Concretamente o namesapce `tns:` está associado a um url do documento wsdl. Explique porquê a associação a este url e se este url será para ser usado para obter o wsdl por parte dos clientes.

Número: Nome:

5) Se a base do desenvolvimento do servidor e do cliente for este documento, está a usar uma aproximação *contract-first.*

   a. [0,5 valores] O que seria uma aproximação *implementation first*?

   b. [1,0 valores] Construa em JAX-WS a definição de uma interface equivalente, usando a aproximação *implementation-first*.

Considere a continuação do documento anterior:

```
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style ="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>

</definitions>
```

6) [0,4 valores] Esta parte do documento corresponde à interface concreta. Porquê?

7) [0,4 valores] Se pretendesse que a invocação se pudesse efectuar em SMTP, que secções deveria modificar?

8) [0,9 valores] A partir da informação acima descreva a execução de uma invocação completa indicando explicitamente

a. Em que máquina se executa

b. Qual o endpoint

c. Qual a função remota

d. Parâmetro de entrada

e. Parâmetro de saída