Questão 1

Completo

Atingiu 0,0 de 1,0

Sobre o RPC (linguagem C), julgue as afirmações abaixo e marque a alternativa correta:

I - O XDR é um formato que permite conversão de tipos de dados, ficando restrito a Little Endian e Big Endian para representação interna entre os processos comunicantes

II - O stub do servidor é a parte do código que contém a função main do lado servidor de uma aplicação RPC

III - O RPC possui funcionalidades para comunicação síncrona e assíncrona em aplicações distribuídas.

a. Apenas a afirmativa I é verdadeira

b. Apenas a afirmativa II é verdadeira

c. Apenas a afirmativa III é verdadeira

d. Apenas as afirmativas II e III são verdadeiras

e. Nenhuma das opções satisfaz as afirmativas apresentadas

Sua resposta está incorreta.

A resposta correta é:
Apenas as afirmativas II e III são verdadeiras

Questão 2

Completo

Atingiu 1,0 de 1,0

Considere um usuário acessando uma página web que envolve 100 requisições entre cliente (browser) e servidor web. Julgue as afirmativas a seguir e marque a alternativa correta:

I - Considerando um diálogo HTTP/1.1, é comum os navegadores (clientes web) abrirem apenas uma conexão persistente com o servidor, para atender as requisições da página solicitada

II - Se o diálogo for HTTP/2, o cliente abre várias conexões com o servidor e distribui as 100 requisições entre as conexões feitas, de modo a reduzir o tempo de carga da página web solicitada.

III - Para um diálogo HTTP/3, essa página carregará muito mais rápido do que com as versões anteriores do HTTP, por fazer uso de multiplexação nas conexões TCP abertas entre o browser e o servidor web.

a. Todas as afirmativas são verdadeiras

b. Nenhuma das opções apresentadas corresponde às afirmativas

c. Apenas as afirmativas I e III são verdadeiras

d. Apenas as afirmativas II e III são verdadeiras

e. Apenas as afirmativas I e II são verdadeiras

Sua resposta está correta.

A resposta correta é:
Nenhuma das opções apresentadas corresponde às afirmativas

Questão 3

Completo

Atingiu 0,4 de 2,0

Microserviço com uso de sockets TCP

Elaborar um microserviço baseado em sockets de rede TCP, no qual o servidor calcula a soma dos N elementos da Série de Fibonacci (1, 1, 2, 3, 5, 8, 13, ...), tendo como base o código exemplo apresentado a seguir:

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>
#define MAX_SIZE        80
int main(int argc,char * argv[]) {
    struct  sockaddr_in lServ; /* dados do servidor     */
    int     sd, n, k;
    char    bufout[MAX_SIZE]; /* buffer de dados enviados  */
    if(argc<3)  {
      printf("Uso: %s <ip_Serv> <porta_Serv>\n", argv[0]); exit(1);}
    memset((char *)&lServ,0,sizeof(lServ)); /* limpa estrutura */
    memset((char *)&bufout,0,sizeof(bufout)); /* limpa buffer */
    lServ.sin_family      = AF_INET;
    lServ.sin_addr.s_addr = inet_addr(argv[1]);
    lServ.sin_port        = htons(atoi(argv[2]));
    sd = socket(AF_INET, SOCK_STREAM, 0);
    if (sd < 0) {
        fprintf(stderr, "Criacao do socket falhou!\n"); exit(1); }
    /* Conecta socket ao servidor definido */
    if (connect(sd, (struct sockaddr *)&lServ, sizeof(lServ)) < 0) {
        fprintf(stderr,"Tentativa de conexao falhou!\n"); exit(1); }
    while (1) {
        printf("> ");
        fgets(bufout, MAX_SIZE, stdin);  /* le dados do teclado */
        send(sd,&bufout,strlen(bufout),0); /* enviando dados ...  */
        if (strncmp(bufout, "FIM",3) == 0)
            break;
    } /* fim while */
    close (sd);
    return (0);
} /* fim do programa */
```

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>
#define QLEN    5  /* tamanho da fila de clientes  */
#define MAX_SIZE  80    /* tamanho do buffer */
int atende_cliente(int descritor, struct sockaddr_in lCli)  {
  char bufin[MAX_SIZE];  int  n;
  while (1) {
    memset(&bufin, 0x0, sizeof(bufin));
    n = recv(descritor, &bufin, sizeof(bufin),0);
    if (strncmp(bufin, "FIM", 3) == 0) break;
    fprintf(stdout,"%s:%u> %s\n",inet_ntoa(lCli.sin_addr),ntohs(lCli.sin_port),bufin);
  } /* fim while */
  close (descritor);
} /* fim atende_cliente */
int main(int argc, char *argv[]) {
  struct sockaddr_in lServ, lCli;
  int    sd, novo_sd, pid, n, alen=sizeof(lCli);
  memset((char *)&lServ,0,sizeof(lServ));
  lServ.sin_family = AF_INET; lServ.sin_addr.s_addr  = inet_addr(argv[1]);
  lServ.sin_port   = htons(atoi(argv[2])); /* PORTA */
  sd = socket(AF_INET, SOCK_STREAM, 0);
  if (bind(sd, (struct sockaddr *)&lServ, sizeof(lServ)) < 0) exit(1);
  if (listen(sd, QLEN) < 0) exit(1);
  printf("Servidor ouvindo no IP %s, na porta %s ...\n\n", argv[1], argv[2]);
  for ( ; ; ) {
    if ( (novo_sd=accept(sd, (struct sockaddr *)&lCli, &alen)) < 0) {
      fprintf(stdout, "Falha na conexao\n"); exit(1); }
    atende_cliente(novo_sd, lCli);
  } /* fim-for */
} /* fim do programa */
```

Entrada

Na linha do arquivo de entrada o número refere-se à quantidade de elementos (N) da Série de Fibonacci a serem considerados.

Entrada

Na linha do arquivo de entrada o número refere-se à quantidade de elementos (N) da Série de Fibonacci a serem considerados.

Saída

O arquivo de saída contém a soma dos N elementos da série de Fibonacci considerando N informado pelo processo cliente.

Restrições

- O socket cliente deve ler de um arquivo de entrada a quantidade de elementos (N) da Série de Fibonacci a serem impressos. Deve ainda solicitar o cálculo para o socket servidor e imprimir o resultado recebido.
- O socket servidor deve receber o valor de N, deve calcular a soma dos elementos da Série e devolver o resultado da soma para o processo cliente.

Exemplo de Entrada 1

```
5
```

Exemplo de Saída 1

```
12
```

Exemplo de Entrada 2

```
8
```

Exemplo de Saída 2

```
54
```

Obs.: Os códigos do cliente e do servidor devem ser compactados, gerando arquivo único nomeado com a matrícula do aluno, por exemplo, 2019000894.zip

Questão 4

Completo

Atingiu 0,6 de 3,0

Gerando série a partir de uma lista de números

Dada uma lista de números de entrada, construa um programa MPI que gere uma série de saída cujos valores referem-se aos elementos de entrada multiplicados pela sua posição N na lista, considerando N > 0.

Entrada

Na linha do arquivo de entrada o primeiro número refere-se à quantidade de números da lista de entrada. Os demais números da linha do arquivo de entrada é a lista de números a serem considerados para geração da série de saída

Saída

O arquivo de saída contém a série de saída obtida pela transformação na lista de entrada

Restrições

- Na parte MPI do programa, usar apenas as primitivas MPI_Send, MPI_Recv para comunicação entre Master e Slaves
- A impressão da série de saída deve ser feita apenas pelos Slaves, sob a coordenação do processo Master.
- O Master, por sua vez, deve escolher aleatoriamente (uso de função randômica) quais Slaves serão requisitados para geração da série de saída.
- O Master é livre para requisitar Slaves para o trabalho e pode ser que não requisite todos os Slaves para geração da série de saída ou ainda requisitar o mesmo Slave mais de uma vez, se for necessário.
- O Slave requisitado deve sempre atuar na geração da série de saída, a partir do trabalho realizado pelo Slave anterior.
- A quantidade (qt) de elementos da série a serem impressos por cada Slave é obtida pela fórmula qt = req*rank. Por exemplo, o slave 2, se for requisitado (req) três vezes, vai imprimir 2, 4 e 6 elementos a cada vez. Já o slave 3, se for requisitado (req) duas vezes, vai imprimir 3 elementos da primeira vez e 6 elementos na segunda vez. Obs.: cuidar para que qt não ultrapasse a quantidade de elementos da série de entrada.

Exemplo de Entrada 1

```
5 9 8 2 6 5
```

Exemplo de Entrada 1

```
5 9 8 2 6 5
```

Exemplo de Saída 1

```
9 16 6 24 25
```

Exemplo de Entrada 2

```
12 126 18 77 82 6 21 11 20 9 10 11 30
```

Exemplo de Saída 2

```
126 36 231 328 30 126 77 160 81 100 121 360
```

Questão 5

Completo

Atingiu 1,0 de 1,0

Analise as afirmativas e, a seguir, marque a alternativa correta:

I - Mecanismos que provêem interoperabilidade entre sistemas distintos pressupõem o uso de um sistema de mensageria e um protocolo de comunicação. Sem esses recursos não há como realizar a interoperabilidade citada

II - No grpc as aplicações que usam protobuf enviam dados em formato binário. Por isso, essas aplicações tem tempo de processamento melhor do que aplicações grpc que fazem uso de formatos como o JSON

III - Num diálogo http/2, se o cliente fizer uma solicitação de recurso para o servidor, este último pode enviar não só o recurso solicitado, mas vários outros associados (sem uma solicitação explícita) na mesma conexão . Essa característica difere o http/2 do http/1.1.

- a. Nenhuma das alternativas satisfaz as afirmativas apresentadas
- b. Apenas II e III estão corretas
- c. Apenas III está correta
- d. Apenas I está correta
- e. Apenas II está correta

Sua resposta está correta.

I - correto. No contexto citado, a interoperabilidade pressupõe um sistema de mensageria e um protocolo de comunicação entre as partes comunicantes.

II - correto.

III - correto.

A resposta correta é:
Nenhuma das alternativas satisfaz as afirmativas apresentadas

Questão 6

Completo

Atingiu 1,0 de 1,0

Analise as afirmações abaixo e, a seguir, marque a alternativa correta

(i) O bit URG é suficiente para garantir o envio de dados urgentes entre entidades TCP

(ii) No TCP/IP, a API-Socket pode ser considerada uma forma de facilitar o acesso às funcionalidades dos protocolos da camada de transporte (TCP e UDP)

(iii) O segmento TCP tem cabeçalho de tamanho variável e payload de tamanho fixo

a. Apenas (iii) é falsa
b. Apenas (ii) e (iii) são falsas
c. Apenas (i) e (iii) são falsas
d. Apenas (i) é falsa
e. Apenas (ii) é falsa

Sua resposta está correta.

A resposta correta é:
Apenas (i) e (iii) são falsas

Questão 7

Completo

Atingiu 1,0 de 1,0

Analise as afirmativas e, a seguir, marque a alternativa correta:

I - Apache Kafka é uma plataforma distribuída de tratamento de streaming de eventos em tempo real cujos tópicos podem ser divididos entre vários nós de um cluster

II - No Apache Kafka, os canais de acesso funcionam como uma fila de entrada/saída, no modelo FIFO - First In, First out, onde cada processo sempre recupera o último elemento do canal

III - O mecanismo publish-subscribe do Apache Kafka funciona como um protocolo de comunicação que equaliza os tempos de processamento dos vários consumidores do broker.

- a. Apenas as afirmativas I e III estão corretas
- b. Todas afirmativas estão corretas
- c. Apenas as afirmativas II e III estão corretas
- d. Apenas as afirmativas I e II estão corretas
- e. Nenhuma das opções corresponde às afirmativas apresentadas

Sua resposta está correta.

I - Correto

II - Incorreto. Os processos podem acessar qualquer elemento da fita de streaming

III - Correto

A resposta correta é:
Apenas as afirmativas I e III estão corretas