API de Sockets

Exemplo de uma calculadora

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007

Conteúdo

Exercício

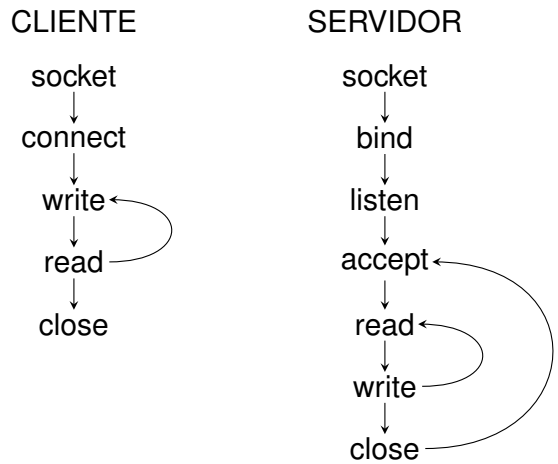
Enunciado

Implementar uma aplicação cliente-servidor que permita simular as operações de uma calculadora.

Objectivos

- implementar um cliente TCP
- implementar um servidor TCP
- trocar informação em formato binário
- projectar um protocolo binário

Sequência de chamadas



Cliente TCP (1/2)

```c
// Header files: <stdio.h>, <stdlib.h>, <string.h>, <stdint.h>
// Header files: <unistd.h>, <sys/types.h>
// Header files: <netinet/in.h>, <arpa/inet.h>

int main(int argc, char *argv[])
{
    int cli_sock;
    struct sockaddr_in srv_addr;
    uint16_t x, y, z, tmp;

    if (argc != 3) {
        printf("uso: tcpcli <endereco_ip> <porta>\n");
        exit(EXIT_FAILURE);
    }

    cli_sock = socket(PF_INET, SOCK_STREAM, 0);

    memset(&srv_addr, 0, sizeof(struct sockaddr_in));
    srv_addr.sin_family = AF_INET;
    srv_addr.sin_port = htons(atoi(argv[2]));
    inet_pton(AF_INET, argv[1], &srv_addr.sin_addr);
```

Cliente TCP (2/2)

```c
    connect(cli_sock, (struct sockaddr *) &srv_addr,
            sizeof(srv_addr));

    x = 1000; y = 2000; z = 0;

    tmp = htons(x);
    write(cli_sock, &tmp, sizeof(uint16_t));
    tmp = htons(y);
    write(cli_sock, &tmp, sizeof(uint16_t));

    read(cli_sock, &tmp, sizeof(uint16_t));
    z = ntohs(tmp);

    printf("%d + %d = %d\n", x, y, z);

    close(cli_sock);

    return 0;
}
```

Servidor TCP (1/3)

```c
// Header files: <stdio.h>, <stdlib.h>, <string.h>, <stdint.h>
// Header files: <unistd.h>, <sys/types.h>, <netinet/in.h>

int main(int argc, char *argv[])
{
    int srv_sock, cli_sock;
    struct sockaddr_in srv_addr, cli_addr;
    socklen_t cli_addr_size;
    uint16_t op1, op2, res, tmp;

    if (argc != 2) {
        printf("uso: tcpsrv <porta>\n");
        exit(EXIT_FAILURE);
    }

    srv_sock = socket(PF_INET, SOCK_STREAM, 0);
```

Servidor TCP (2/3)

```c
    memset(&srv_addr, 0, sizeof(struct sockaddr_in));
    srv_addr.sin_family = AF_INET;
    srv_addr.sin_port = htons(atoi(argv[1]));
    srv_addr.sin_addr.s_addr = INADDR_ANY;

    bind(srv_sock, (struct sockaddr *) &srv_addr,
            sizeof(struct sockaddr_in));

    listen(srv_sock, 5);

    cli_addr_size = sizeof(struct sockaddr_in);

    while((cli_sock = accept(srv_sock,
                (struct sockaddr *) &cli_addr, &cli_addr_size))) {
```

Servidor TCP (3/3)

```
        read(cli_sock, &tmp, sizeof(uint16_t));
        op1 = ntohs(tmp);
        read(cli_sock, &tmp, sizeof(uint16_t));
        op2 = ntohs(tmp);

        res = op1 + op2;

        tmp = htons(res);
        write(cli_sock, &tmp, sizeof(uint16_t));

        shutdown(cli_sock, 2);
        close(cli_sock);
    }

    close(srv_sock);
    return 0;
}
```

Exercícios

Enunciados

Modificar o exemplo fornecido de modo a:

- tratar todos os valores de retorno das chamadas ao sistema (as chamadas ao sistema retornam o valor -1 em caso de erro; a variável **errno** contem o código do erro);
- adicionar mais operações tais como a subtração, multiplicação, . . .);
- permitir efectuar operações aritméticas com dois ou mais operandos.