

Sistemas Distribuídos - UFRJ



Trabalho Prático 1

Aluno: Paulo Victor Innocencio

DRE: 116213599

Disciplina: Sistemas Distribuídos
(COS470)

Professor: Daniel Ratton Figueiredo

Introdução

O intuito deste relatório é mostrar as decisões feitas para o projeto e o passo a passo das partes cruciais dos programas. A linguagem utilizada para desenvolvimento foi a linguagem ANSI C e o ambiente de trabalho foi o Linux Ubuntu.

Sinais

O objetivo do programa era realizar uma simples troca de mensagens entre dois processos distintos, sendo que a escolha dos sinais a serem enviados era livre. Os sinais escolhidos foram:

- SIGINT
- SIGUSR2
- SIGUSR1
- SIGQUIT

Lista base de sinais com seus respectivos inteiros correspondentes:

A saída do comando corresponde aos nomes de todos os sinais usados no sistema, inclusive os sinais da categoria *realtime*.

1) SIGHUP	2) SIGINT
3) SIGQUIT	4) SIGILL
5) SIGTRAP	6) SIGABRT
7) SIGBUS	8) SIGFPE
9) SIGKILL	10) SIGUSR1
11) SIGSEGV	12) SIGUSR2
13) SIGPIPE	14) SIGALRM
15) SIGTERM	16) SIGSTKFLT
17) SIGCHLD	18) SIGCONT
19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOU
23) SIGURG	24) SIGXCPU
25) SIGXFSZ	26) SIGVTALRM
27) SIGPROF	28) SIGWINCH
29) SIGIO	30) SIGPWR
31) SIGSYS	34) SIGRTMIN
35) SIGRTMIN+1	36) SIGRTMIN+2
37) SIGRTMIN+3	38) SIGRTMIN+4
39) SIGRTMIN+5	40) SIGRTMIN+6
41) SIGRTMIN+7	42) SIGRTMIN+8
43) SIGRTMIN+9	44) SIGRTMIN+10
45) SIGRTMIN+11	46) SIGRTMIN+12
47) SIGRTMIN+13	48) SIGRTMAX-14
49) SIGRTMAX+15	50) SIGRTMAX-14
51) SIGRTMAX-13	52) SIGRTMAX-12
53) SIGRTMAX-11	54) SIGRTMAX-10
55) SIGRTMAX-9	56) SIGRTMAX-8
57) SIGRTMAX-7	58) SIGRTMAX-6
59) SIGRTMAX-5	60) SIGRTMAX-4
61) SIGRTMAX-3	62) SIGRTMAX-2
63) SIGRTMAX-1	63) SIGRTMAX

Foi necessário a criação de um protocolo simples para o envio dos sinais, utilizando a função `switch ()`. Um dos quatro sinais citados acima, eram evocados de maneira aleatória pelo processo ‘Pai’, que o encaminhava para o processo ‘Filho’.

Demonstração do output final do programa:

```
paulo@paulo-VirtualBox: ~/Área de Trabalho/TP 1/1.1
paulo@paulo-VirtualBox:~/Área de Trabalho/TP 1/1.1$ ./sinais
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 12
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 10
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 2
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 10
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 2
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
```

```
paulo@paulo-VirtualBox: ~/Área de Trabalho/TP 1/1.1
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Enviando Sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 12
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 2
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 10
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 12
Processo Pai diz: Enviando Sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 10
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
Processo Pai diz: Sinal enviado!
Processo Filho diz: Recebi o sinal: 3
Processo Pai diz: Enviando Sinais...
Processo Filho diz: Aguardando sinais...
--Fim do Programa
paulo@paulo-VirtualBox:~/Área de Trabalho/TP 1/1.1$
```

Pipes

A troca de informação do clássico programa Produtor-Consumidor deveria ocorrer através de Pipes. Após um número aleatório ser gerado de forma crescente a partir do número 1 pelo programa produtor, o mesmo deveria ser enviado para o programa consumidor que determinaria se ele seria um número primo ou não.

Demonstração da validação do pipe criado:

```
//Criando o Pipe
if(pipe(fd)<0){
    perror("Pipe");
    return -1;
}
```

O processo pai assumia a função do produtor, enquanto o processo filho representava o consumidor. Apenas um escrevia no pipe, enquanto o outro apenas realizava a leitura.

Escrita (Produtor)

```
//Processo Pai
if(pid>0){
    char srt[BUFFER];
    close(fd[READ]);

    if(i==quant_numeros){
        write(fd[1],"0",BUFFER);
        return 1;
    }
    else{
        sprintf(srt,"%i",num);
        write(fd[1],srt,BUFFER);
        printf("\nProcesso Pai diz: enviei pelo pipe -> %s",srt);
    }

    num=num+(rand()%100+1);
}
```

Leitura (Consumidor)

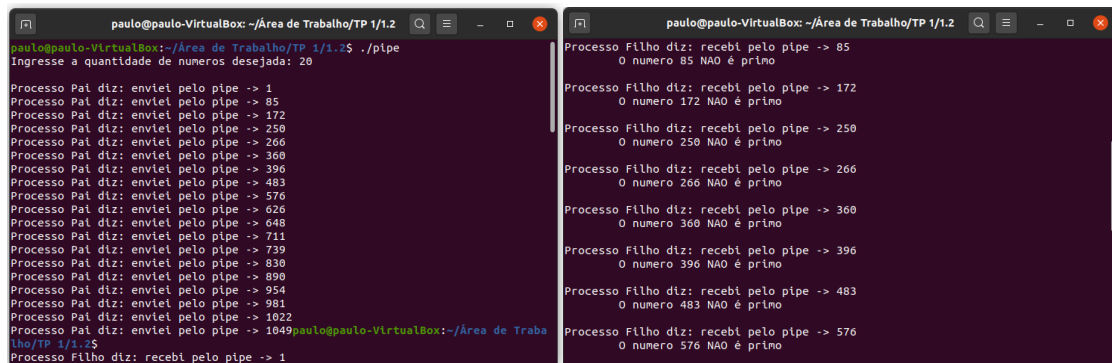
```
//Processo Filho
if(pid==0){
    char srt_recebida[BUFFER];
    int num_recebido;

    close(fd[WRITE]); //Fecha a escrita do pipe no filho
    read(fd[0],srt_recebida,BUFFER); //Leitura do pipe

    num_recebido=atoi(srt_recebida); //Converte para inteiro

    printf("\nProcesso Filho diz: recebi pelo pipe -> %d\n",num_recebido);
    primoTeste(num_recebido); //Verifica se é primo
}
```

Demonstração do output final do programa:



```
paulo@paulo-VirtualBox: ~/Área de Trabalho/TP 1/1.2
paulo@paulo-VirtualBox:~/Área de Trabalho/TP 1/1.2$ ./pipe
Ingresse a quantidade de numeros desejada: 20

Processo Pai diz: enviei pelo pipe -> 1
Processo Pai diz: enviei pelo pipe -> 85
Processo Pai diz: enviei pelo pipe -> 172
Processo Pai diz: enviei pelo pipe -> 250
Processo Pai diz: enviei pelo pipe -> 266
Processo Pai diz: enviei pelo pipe -> 360
Processo Pai diz: enviei pelo pipe -> 396
Processo Pai diz: enviei pelo pipe -> 483
Processo Pai diz: enviei pelo pipe -> 576
Processo Pai diz: enviei pelo pipe -> 626
Processo Pai diz: enviei pelo pipe -> 648
Processo Pai diz: enviei pelo pipe -> 711
Processo Pai diz: enviei pelo pipe -> 739
Processo Pai diz: enviei pelo pipe -> 830
Processo Pai diz: enviei pelo pipe -> 890
Processo Pai diz: enviei pelo pipe -> 954
Processo Pai diz: enviei pelo pipe -> 981
Processo Pai diz: enviei pelo pipe -> 1022
Processo Pai diz: enviei pelo pipe -> 1049
paulo@paulo-VirtualBox:~/Área de Traba
lho/TP 1/1.2$
Processo Filho diz: recebi pelo pipe -> 1
Processo Filho diz: recebi pelo pipe -> 85
    O numero 85 NAO é primo
Processo Filho diz: recebi pelo pipe -> 172
    O numero 172 NAO é primo
Processo Filho diz: recebi pelo pipe -> 250
    O numero 250 NAO é primo
Processo Filho diz: recebi pelo pipe -> 266
    O numero 266 NAO é primo
Processo Filho diz: recebi pelo pipe -> 360
    O numero 360 NAO é primo
Processo Filho diz: recebi pelo pipe -> 396
    O numero 396 NAO é primo
Processo Filho diz: recebi pelo pipe -> 483
    O numero 483 NAO é primo
Processo Filho diz: recebi pelo pipe -> 576
    O numero 576 NAO é primo
```

Como não utilizei nenhum meio de controle entre processos, as saídas no terminal do processo consumidor apareceram após as do processo produtor, devido a sua maior complexidade de instruções a serem realizadas.

Sockets

Seguindo a mesma proposta do exercício anterior, o programa Produtor-Consumidor agora deveria trocar informação utilizando sockets. O método mais complexo em comparação com os anteriores, mas ao mesmo tempo se mostrou mais eficiente.

Foi necessário a criação de dois programas distintos, seguindo a lógica do Cliente/ Servidor

Demonstração da criação do socket no lado do Servidor (produtor):

```
char server_message[256] = "You have reached the server!";

// create the server socket
int server_socket;
server_socket = socket(AF_INET, SOCK_STREAM, 0);

//define the server address
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;
server_address.sin_port = htons(9002);
server_address.sin_addr.s_addr = INADDR_ANY;

// bind the socket to our specified IP and port
bind(server_socket, (struct sockaddr*) &server_address, sizeof(server_address));

listen(server_socket, 5);

int client_socket;
client_socket = accept(server_socket, NULL, NULL);

// send the message
send(client_socket, server_message, sizeof(server_message), 0);
```

Demonstração da conexão ao socket criado no lado do Cliente (consumidor):

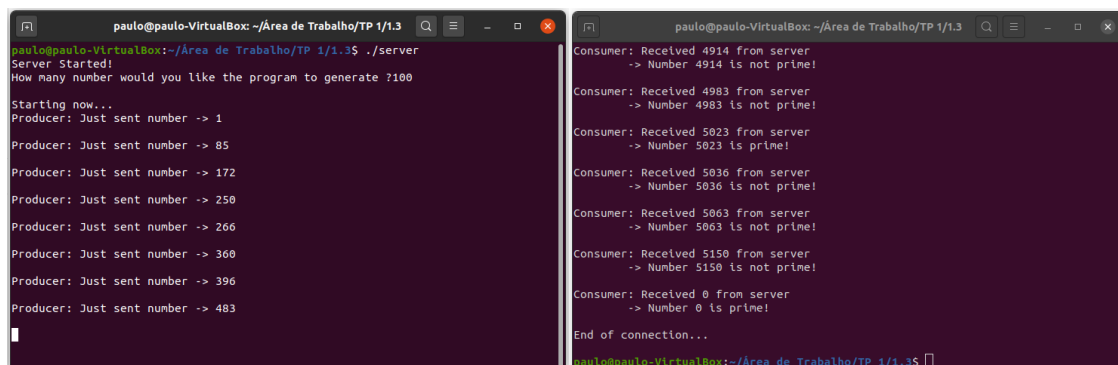
```
// create a socket
int network_socket;
network_socket = socket (AF_INET, SOCK_STREAM, 0);

// specify an address for the socket
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;
server_address.sin_port = htons(9002);
server_address.sin_addr.s_addr = INADDR_ANY;

int connection_status = connect(network_socket, (struct sockaddr *) &server_address, sizeof(server_address));
// check for error with the connection
if(connection_status == -1){
    printf("Theres was an error making a connection to the remote socket \n\n");
}

// receive data from server
char server_response[256];
recv(network_socket, &server_response, sizeof(server_response), 0);
printf("-> %s\n\n", server_response);
```

Demonstração do output final do programa:



```
paulo@paulo-VirtualBox: ~/Área de Trabalho/TP 1/1.3
paulo@paulo-VirtualBox:~/Área de Trabalho/TP 1/1.3$ ./server
Server Started!
How many number would you like the program to generate ?100
Starting now...
Producer: Just sent number -> 1
Producer: Just sent number -> 85
Producer: Just sent number -> 172
Producer: Just sent number -> 250
Producer: Just sent number -> 266
Producer: Just sent number -> 360
Producer: Just sent number -> 396
Producer: Just sent number -> 483

Consumer: Received 4914 from server
-> Number 4914 is not prime!
Consumer: Received 4983 from server
-> Number 4983 is not prime!
Consumer: Received 5023 from server
-> Number 5023 is prime!
Consumer: Received 5036 from server
-> Number 5036 is not prime!
Consumer: Received 5063 from server
-> Number 5063 is not prime!
Consumer: Received 5150 from server
-> Number 5150 is not prime!
Consumer: Received 0 from server
-> Number 0 is prime!
End of connection...
paulo@paulo-VirtualBox:~/Área de Trabalho/TP 1/1.3$
```

Considerações finais

- A imagem de capa não é de minha autoria e foi retirada da internet
- Todos os códigos estão disponíveis em:
<https://drive.google.com/drive/folders/1jU7L9n7SUibS2ftHeR5FVrUIedj3JA9G?usp=sharing>