

Exercício 2.2 – Servidor TCP Concorrente

# **Aluna:** Naomi Takemoto

# **RA:** 184849

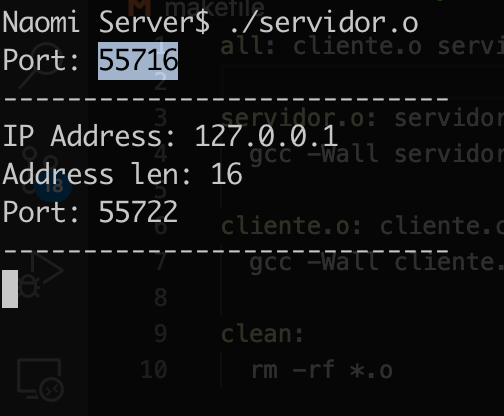
# Instituto de Computação

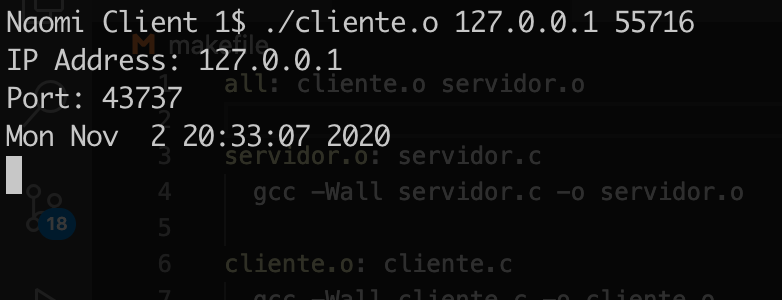
# Universidade Estadual de Campinas

# Novembro de 2020

# Excercício 1

Adicione a função ​sleep no ​servidor.c da atividade prática anterior antes do socket ser fechado close(connfd) de modo que o servidor ''segure'' a conexão do primeiro cliente que se conectar. Com essa modificação, o servidor aceita a conexão de dois clientes de forma concorrente ? Comprove sua resposta através de testes.







# Exercício 2

Escreva, utilizando sockets TCP, um programa cliente e um programa servidor de echo que possibilitem a execução remota de comandos enviados pelo cliente. ​**Lembre-se que o servidor deve atender a vários clientes de forma concorrente.** O servidor deve receber como argumento na linha

de comando a porta na qual irá escutar. O cliente deve receber como argumento na linha de comando o endereço IP do servidor e a porta na qual irá conectar.

**Detalhes do funcionamento:**

* ○  O ​**cliente**​ faz continuamente o seguinte:
  + Estabelece conexão com o servidor
  + Recebe uma cadeia de caracteres do servidor
  + Executa uma cadeia de caracteres
  + Envia o resultado para o servidor
* ○  O ​**servidor**​ faz continuamente o seguinte:
  + Recebe o resultado do cliente
  + Escreve em um arquivo o resultado IP e porta dos clientes

O ​**cliente**​ deverá exibir na saída padrão:

* + dados do host servidor ao qual está se conectando (IP e PORTA)
  + dados de IP e PORTA locais utilizados na conexão O ​**servidor**​ deverá exibir na saída padrão:

■ Cadeia de caracteres enviadas pelo cliente juntamente com dados de IP e porta do cliente.

**\*\*Devem\*\*** ser escritas e usadas "funções envelopadoras" (wrapper functions) para as chamadas da API de sockets, a fim de tornar o seu código mais limpo e poderem ser reutilizadas nos próximos trabalhos. Utilize a convenção do livro texto, dando o mesmo nome da função, com a 1a letra maiúscula. Veja abaixo um exemplo:

int Socket(int family, int type, int flags) {

int sockfd;

if ((sockfd = socket(family, type, flags)) < 0) {

perror("socket");

exit(1);

} else

return sockfd;

}

○ Dica: para desenvolver esta atividade poderá ser encontrada nos exemplos do livro-texto da disciplina e nos programas utilizados no exercício anterior.

# Exercício 3

Modifique o servidor para este gravar em um arquivo as informações referentes ao instante em que cada cliente conecta e desconecta, IP, e porta. O servidor não deverá mostrar nenhum mensagem na saída padrão. OBS: Comente o código onde era exibibido mensagens pois fará parte da avaliação.

# Exercício 4

**Detalhes das modificações:**

* + ○  O cliente deve ser modificado de modo que, quando uma certa string for digitada na entrada padrão (por exemplo: ​exit,​ ​quit​, ​bye​, ​sair​, ...), a sua execução seja finalizada (todas as conexões abertas devem ser corretamente fechadas antes).
  + ○  O cliente exibirá, no lugar do "echo" do servidor:

■ cadeias de caracteres enviadas pelo servidor invertidas

○ O servidor exibirá, no lugar da cadeia de caracteres:

■ os dados de IP e PORTA seguidos da string que foi enviada por aquele cliente, de modo a identificar qual comando foi enviado por cada cliente.

■ O IP e PORTA dos clientes que se desconectem, no momento da desconexão. ○ O servidor irá escrever em um arquivo texto o endereço IP, porta, instante de conexão

e de desconexão para cada cliente.

# Exercício 5

Com base ainda no seu código, é correto afirmar que os clientes nunca receberão FIN neste caso já que o servidor sempre ficará escutando (LISTEN)? Justifique.

# Exercício 6

Comprove, utilizando ferramentas do sistema operacional, que os processos criados para manipular cada conexão individual do servidor aos clientes são filhos do processo original que foi executado.