Universidade Federal de Roraima
Departamento de Ciência da Computação
Sistemas Operacionais – DCC403

PROJETO FINAL - Prazo de Entrega: 28/07/2022

ALUNO(A):_____NOTA:_____

ATENÇÃO: Descrever as soluções com o máximo de detalhes possível, no caso de programas, inclusive a forma como os testes foram feitos. Todos os artefatos (relatório, código fonte de programas, e outros) gerados para este trabalho devem ser adicionados em um repositório no site `github.com`, com o seguinte formato:

`Aluno1Aluno2 _FinalProject_OS_RR_2022`.

[DESCRIÇÃO] ***Compressed Network Communication***

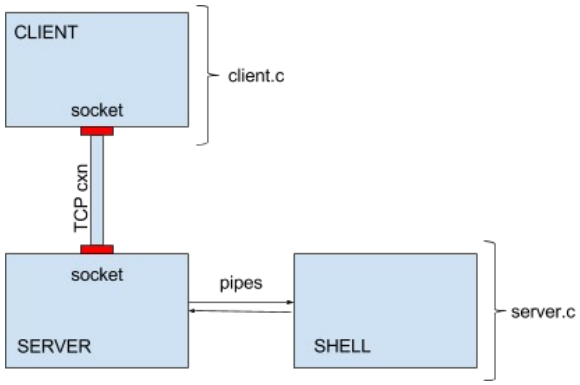
Quando uma aplicação (por exemplo, shell) é para ser usada remotamente, geralmente não é suficiente simplesmente substituir E/S de dispositivo local (por exemplo, entre o terminal do usuário e o shell) com a E/S de rede. Sessões remotas e protocolos de rede adicionam opções e comportamentos que não existiam quando o aplicativo estava sendo usado localmente. Para manipular esse processamento adicional sem fazer nenhuma alteração no aplicativo, é comum criar agentes do lado do cliente e do lado do servidor que manipulem a comunicação de rede e protejam o aplicativo das complexidades dos protocolos de acesso remoto. Esses agentes intermediários podem implementar recursos valiosos (por exemplo, criptografar o tráfego para aumentar a segurança ou compactar o tráfego para melhorar o desempenho e reduzir o custo da comunicação em escala de WAN), de forma totalmente transparente para o usuário e o aplicativo.

Neste projeto, você criará um cliente e um servidor de telnet que pode ser dividido em duas etapas principais:

- Passar a entrada e saída através de um soquete TCP; e
- Compactar a comunicação entre o cliente e o servidor.

Passar a entrada e saída através de um soquete TCP

Seu projeto geral do programa será semelhante a figura abaixo. Você terá um processo de cliente, um processo de servidor e um processo de shell. Os dois primeiros são conectados via conexão TCP, e os dois últimos são conectados via pipes.



O programa cliente

- O programa cliente abrirá uma conexão com um servidor (a porta deverá ser especificada com o parâmetro de linha de comando --port) em vez de enviá-lo diretamente para um shell. O cliente deve então enviar a entrada do teclado para o soquete (enquanto ecoa para o monitor), e a entrada do soquete para o monitor;
- Inclua, no cliente, uma opção --log=filename, que mantém um registro dos dados enviados pelo soquete;
- Adicione uma opção --compress ao cliente;

Para todos os testes, o cliente e o servidor estarão executando no mesmo host (localhost), mas se você quiser adicionar um parâmetro --host=name, poderá acessar uma sessão de shell de outros computadores.

O programa do servidor

- O programa do servidor irá se conectar com o cliente, receber os comandos do cliente e enviá-los para o shell e "servir" ao cliente as saídas desses comandos;
- O programa do servidor escutará em um soquete de rede (porta especificada com o parâmetro de linha de comando --port = obrigatório);
- Aceite uma conexão quando ela for feita;
- Uma vez que uma conexão é feita, crie um fork do processo filho, que executará um shell para processar os comandos recebidos. O processo do servidor deve se comunicar com o shell por meio de pipes;
- Redirecione o stdin/stdout/stderr do processo do shell para as extremidades apropriadas do pipe;
- Entrada recebida através do soquete de rede deve ser encaminhada através do pipe para o shell. Como o servidor cria um fork do shell (e conhece seu ID de processo), o processamento de ^C (transformando-o em um SIGINT no shell) deve ser feito no servidor. Da mesma forma, quando o servidor encontra um ^D, ele deve fechar o lado da gravação do pipe para o shell;
- A entrada recebida dos pipes do shell (que recebem stdout e stderr do shell) deve ser encaminhada para o soquete da rede.

A opção --log

Para garantir que a compactação está sendo feita corretamente, pedimos que você adicione uma opção --log=filename ao seu cliente. Se essa opção for especificada, todos os dados gravados ou lidos do servidor deverão ser registrados no arquivo especificado. Prefixo cada entrada de log com SENT # bytes: ou RECEIVED # bytes: conforme apropriado. (Observe o dois pontos e o espaço entre a palavra bytes e o início dos dados). Cada uma dessas linhas deve ser seguida por um caractere de nova linha (mesmo se o último caractere da string relatada for uma nova linha).

Saída de formato de log de amostra:

```
ENVIADO  35 bytes: sendingsendingsendingsendingsending
RECEBIDO 18 bytes: receivingreceiving
```

Comunicação comprimida/compactada

O objetivo da compactação é reduzir a quantidade de dados de espaço ocupados. Nos cenários de comunicação, uma versão compactada de dados usará menos largura de banda para transmitir os mesmos dados que a forma não compactada usaria. Neste projeto, você só executará compactação, sem criptografia. Você usará uma biblioteca de compactação padrão para melhorar a eficiência de suas comunicações de soquete.

- Dependendo do sistema em que você desenvolve o seu projeto, você pode precisar instalar o pacote **zlib** para obter a biblioteca de compressão e sua documentação;
- Adicione uma opção de linha de comando `--compress` ao seu cliente e servidor que, se incluído, ativará a compactação (de todo o tráfego em ambas as direções);
- Modifique o aplicativo cliente e servidor para compactar o tráfego antes de enviá-lo pela rede e descompactá-lo após recebê-lo;
- A opção `--log` deve registrar os dados de saída após a compactação e a pré-descompactação de dados recebidos; e
- Inclua seus arquivos `--log` no seu envio.

Segue abaixo alguns links:
http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html
https://www.zlib.net/zlib_how.html