Ver o ficheiro adxunto para información sobre as funciones e como crear un servidor e un cliente TCP básicos.

Obxectivo da práctica
Crear un programa servidor e un programa cliente en TCP e entender o seu funcionamento.

1. Cliente/servidor básicos. Sigue os pasos do documento adxunto para crear unha conexión TCP cliente/servidor na que o servidor envíe unha mensaxe de saudo ao cliente. Usar buffers de recepción e envío de 1024. Recoméndase probar que todo funciona en diferentes computadores.

i.-Programa servidor TCP:

Debe escoitar por todas as suas interfaces usando un número de porto indicado como parámetro na liña de comandos (ver exemplo_argv.py da P1).
Debe poder atender múltiples conexións SECUENCIAIS de clientes (non á vez).
Para cada cliente que se conecte, debe imprimir en pantalla a IP do cliente e enviarlle unha mensaxe de saudo (unha cadena de polo menos 20 caracteres), despois pechar a conexión. Chamade ao programa ''servidor-11.py''.

ii.- Facer un programa cliente TCP:

Debe permitir indicar a IP e o porto do servidor como parámetros na liña de comandos (ver exemplo_argv.py da P1).
Debese conectar ao servidor e recibir a mensaxe do mesmo.
Debe imprimir a mensaxe recibida e o número de bytes que se recibiron, despois pechar a conexión. Chamade ao programa ''cliente-11.py''.

iii.- Comproba se e posíbel que o servidor envíe dúas mensaxes con sendas funcións send() e que o cliente reciba (despois dun tempo prudencial, usar a función *sleep()*) ambas mensaxes cunha única sentencia recv(). Describe as modificacións introducidas nos códigos e o resultado obtido (número de bytes recibidos e a mensaxe).

iv.- Proba a usar no cliente un bucle de tipo *while len(mensaxe_recibida) > 0:* e cambiar o seu buffer de recepción a 8. Describe as modificacións introducidas nos códigos e o resultado obtido. (En python igual é máis sinxelo usar *while(true)* e *if len(mensaxe_recibida) == 0: break;*)

2. Servidor de maiúsculas.
v.- Facer un programa servidor e un programa cliente no que o cliente lea un arquivo de texto de entrada (pasándolle o nome en liña de comandos), e se llo envíe liña a liña ao servidor usando a mesma conexión. O servidor debe pasar os caracteres de cada liña a maiúsculas e devolverlle ao cliente as liñas convertidas (ver función toupper() en C e upper() en Python). Chamade aos programas ''cliente-25.py'' e ''servidor-25.py''. Por último, o cliente vai recibindo as liñas e as vai escribindo nun arquivo de saída, que debe ter o mesmo nome que o arquivo de entrada pero engadindolle a terminación 'CAP', e ser igual ao arquivo orixinal pero coas letras en maiúsculas. O funcionamento debe ser o seguinte: unha vez establecida a conexión:

a.- o cliente le unha liña do arquivo de entrada e se lla envía ao servidor (imprimir por pantalla o número de bytes enviados)
b.- o servidor devolve a liña ao cliente pasada a maiúsculas (imprimir por pantalla o número de bytes recibidos e enviados)
c.- o cliente a escribe no arquivo de saída (imprimir por pantalla o número de bytes recibidos)
d.- de volta ao paso a, ata que se termine o arquivo.

Ademáis, o servidor debe aceptar como parámetro de entrada o número de porto polo que atende as solicitudes e o cliente a IP e porto do servidor ademáis do nome do ficheiro de entrada. Usade buffers de 1024

vi.- Cambiade os buffers de recepción a 10 e 20 (para ques sexan diferentes no servidor e cliente). Segue funcionando ben? Modificade o programa para que funcione corectamente con buffers pequenos e diferentes (unha solución é usando *socket.shutdow()*, e xa non ten porque ser liña a liña). Chamade aos programas ''cliente-26.py'' e ''servidor-26.py''.

Requisitos mínimos:
Os programas poden programarse en C ou Python. Se se fai todo en Python (o recomendado).

Débese facer uso das funcións vistas na clase.
Toda chamada a unha función do sistema debe ter o seu correspondente chequeo de erro.
Toda función debe sair co mensaxe de erro e o código apropiado en caso de erro (se o hai).
O código non pode fallar aínda qu se usen datos de entrada incorrectos.
Toda memoria reservada dinámicamente debe liberarse correctamente (so para C).

?

O código debe de estar adecuadamente comentado, indicándose de forma clara qué se fai en todas as funcións definidas, xunto coa explicación dos parámetros de entrada e saída das mesmas.
O código deberá estar correctamente formateado e tabulado.
O código debe compilar nun sistema Linux con gcc (para C). Se se usa Windows, débese convertir o ficheiro co código a formato Unix.
Os programas deben funcionar aínda que o cliente e o servidor se executen en computadores diferentes
Os parámetros de entrada necesarios deben ser proporcionados como argumentos do main.
Penalizarase que aparezan mensaxes de Warning na compilación (coa opción -Wall)(so C)

Entrega. Un arquivo comprimido zip con:
- Os códigos (non os executables) dos puntos 1.1 (servidor básico), 1.2 (cliente básico) e 2 (servidor e cliente de maiúsculas, punto 2.6, se non se ten feito vale o 2.5).
- Informe sobre os puntos 1.3 e 1.4. Chamadeo ''informe.txt''
- (Non obrigatorio) Unha captura de pantalla na que se poida ver que o cliente e servidor de maiúsculas se están executando en distinto ordenador (ou máquina virtual).

Última modificación: luns, 9 de outubro de 2023, 12:03