

Tarea 1 CC51C

9 de Abril 2007 - Plazo: 2 semanas

1. Objetivos

Esta tarea persigue construir una aplicación simple de tipo cliente/servidor, usando un protocolo de comunicaciones simple, en un medio de comunicación sin errores. El alumno debe escribir el cliente usando un socket TCP directamente. El principal problema de esta tarea radica en adaptar los servicios requeridos por la aplicación a las primitivas provistas por el layer de datos (TCP) y entender bien un protocolo especificado por un documento.

2. Aplicación

La aplicación cliente, será un comando interactivo que leerá comandos de la entrada standard. Los comandos se transformarán en pedidos al servidor que siguen reglas estrictas de formateo. Este nivel usa las primitivas de la capa datos para comunicarse con el servidor.

Los comandos que debe implementar el cliente son los siguientes, listados en orden de dificultad, y acompañados del formato que debe tener el pedido al servidor. Los pedidos siempre son un byte (char) con el código de comando y luego los argumentos, todo en ASCII. Los strings se envían sin el cero al final y sus largos son en número de bytes.

En la lista siguiente, aparece el nombre del comando, entre paréntesis un ejemplo tipeado como comando al cliente, a la izquierda lo que transmite el cliente y a la derecha lo que responde el servidor, en un ejemplo de cada comando.

- `quit ('q')` Termina la comunicación. El servidor muere luego de contestar este pedido con una línea:

cliente	servidor
q	q

- `load ('l')` Pregunta al servidor la carga actual del sistema (en el servidor), que es un string terminado por newline:

cliente	servidor
l	22:00:08 up 12:34, 8 users, load average: 0.08, 0.06, 0.00

- `ping n ('p 3')` Pide al servidor n números desde el 0 hasta $n - 1$ en orden creciente.

cliente	servidor
p3	0
	1
	2

- **fortune ('f')** Pide al servidor la buena fortuna. El servidor contesta con un mensaje de tamaño variable pero de profundo contenido:

cliente	servidor
f	<largo>
	<string>
	<largo>
	<string>
	...
	-1

Los largos son ASCII y vienen uno por línea.

- **cat ('c termcap')** Pide al servidor que le mande un archivo. El cliente debe mostrarlo en la salida standard.

cliente	servidor
ctermcap	<largo_archivo>
	<archivo completo>

El largo es ASCII y viene en una línea.

- **wc ('w /etc/termcap')** Le envía el contenido del archivo `/etc/termcap` al servidor, el servidor retorna el número de palabras (como el comando `wc -w`) que había en él.

cliente	servidor
w3021	
<archivo completo>	<contador de palabras>

Los largos son ASCII, terminados por newline.

3. Ejecución

Para compilar la tarea, recomendamos hacer un Makefile. Separen las funciones de comunicación de la aplicación para que puedan reusar la aplicación para las próximas tareas.

En `dichato.dcc.uchile.cl` hay un servidor de prueba corriendo en el puerto 2000. Para la transferencia de archivos pueden pedir "prueba" que es chico y "termcap" que es grande, ambos son ASCII. También pueden tratar de simular el protocolo a mano, haciendo un telnet al port 2000 de dichato. Por razones

de seguridad, el servidor no puede acceder archivos fuera de su directorio de ejecución, por lo que solo pueden usar esos nombres de archivos.

La tarea será probada contra ese mismo servidor, de modo de poder garantizar que respeta el protocolo estipulado.

La recomendación es que implementen los comandos en orden de dificultad, de modo de garantizar que funcionen antes de pasar al siguiente. Recuerden que no se aceptan tareas atrasadas.

Adhiéranse estrictamente a los comandos y sus respuestas, porque la tarea será evaluada automáticamente...

Cualquier duda o pregunta o reporte de bugs, dirigirse al foro de U-cursos.