LABORATORIO 3

1. Introducción

En esta tarea se desea implementar una agenda telefónica compartida denominada **SoAgenda**. Dicha agenda permitirá ingresar y consultar información telefónica sobre personas en distintas categorías. La agenda se estructura como un directorio jerárquico de dos tipos de datos. Por un lado estarán las **categorías**, que agruparán tanto contactos como otras sub categorías. Por su parte, los **contactos** tendrán información sobre una persona (*nombre y teléfono*). Tanto los contactos como las categorías tendrán información sobre el usuario que creo dicha entrada en la agenda.

Los clientes del sistema podrán navegar dentro del directorio y listar datos sobre la categoría en la que se encuentran (contactos y sub categorías). Asimismo, podrán crear y eliminar tanto categorías como contactos.

El sistema se separará en dos componentes. Por un lado se encuentra el servidor de agenda **serviagenda**, Éste será el encargado de mantener las estructuras de datos para albergar la agenda. Solo dispone de dos comandos:

- **listar**: Este comando listara todas entradas de la agenda indicando para cada entrada su nombre, tipo y la categoría en la que se encuentra. Opcionalmente podrá incluir el usuario que creo dicha entrada.
- **salir** para salir del sistema. Este comando solamente podrá ser ejecutado con éxito si no hay usuarios en el sistema.

Además el servidor deberá ir mostrando en pantalla los comandos recibidos de los usuarios (solo los comandos, no los resultados del comando), junto con el nombre del usuario que ejecuta dicho comando.

Por otro lado se encuentra el cliente de agenda **cliagenda** en el cual el usuario podrá interactuar con el sistema. Para entrar al sistema se debería ejecutar **cliagenda <nombre>** donde nombre será el nombre del usuario que entra al sistema. Una vez que un usuario entra al sistema se encontrará ubicado en la categoría **principal**. Asimismo tendrá disponibles los siguientes comandos:

- **dir**: despliega una lista de las subcategorías y contactos que se encuentren en la categoría actual. El formato será el siguiente:

  <nombre> <tipo> creado por <nombre2>, donde nombre será el *nombre* de la categoría o contacto, *tipo* será **categoría** o **contacto** y *nombre2* será el nombre del usuario que creo el contacto/categoria.
- **ver <contacto>**: que listará información sobre el contacto con el siguiente formato

  <nombre> <teléfono> creado por <nombre2>
- **creacat <nombre>** que creará una categoría **nombre** bajo la categoría actual.
- **creacon <nombre> <teléfono>** que creara un contacto en la categoría actual.
- **cd <nombre>** para acceder a una categoría bajo la categoría actual
- **cd ..** para ir a la categoría superior (se asume que la categoría **principal** es padre de si misma)
- **borrar <nombre>**: que borra el *contacto* con dicho *nombre*. No se permite borrar categorías.
- **salir**: para salir del sistema

Además el cliente deberá ir mostrando en pantalla los comandos recibidos de los demás usuarios que se encuentren actualmente en la misma categoría que el usuario (solo los comandos, no los resultados del comando), junto con el nombre del usuario que ejecuta dicho comando.

1.2. Tarea

La tarea consiste en **diseñar, implementar y documentar el módulo *ServiAgenda*** (programas **serviagenda** y **cliagenda**) asegurando el correcto funcionamiento de los mismos.

1.3. Ejemplos

A continuación se presenta un ejemplo para aclarar el funcionamiento del sistema.

cliagenda juan			
<El servidor no esta corriendo>			
serviagenda			
<Se ha iniciado el servicio>	**cliagenda juan**	**cliagenda maria**	**cliagenda pedro**
<juan entra al sistema> <maria entra al sistema> <pedro entra al sistema>	<Bienvenido al sistema>	<Bienvenido al sistema>	<Bienvenido al sistema>
	dir	**dir**	
<juan dir> <maria dir>	<No hay entradas> <maria dir>	<No hay entradas> <juan dir>	**creacon lucas 123** <juan dir> <maria dir>
<pedro creacon lucas 123>	<pedro creacon lucas 123>	<pedro creacon lucas 123> **creacat lucas**	<se ha creado el contacto>
<maria creacat lucas>	<maria creacat lucas> **creacat amigos**	<Ya existe una entrada con ese nombre>	<maria creacat lucas>
<juan creacat amigos>	<se ha creado la categoria>	<juan creacat amigos>	<juan creacat amigos> **dir**
<pedro dir>	<pedro dir>	<pedro dir>	<lucas contacto creado por pedro> <amigos categoría creado por juan>
			cd amigos
<pedro cd amigos> **salir**	<pedro cd amigos>	<pedro cd amigos>	<has cambiado a la categoría amigos>
<No puedes salir del sistema en este momento>	**creacon ana 112233**	**creacat parientes**	**creacon hector 321**
<juan creacon ana 112233> <pedro creacon hector 321>	<se ha creado el contacto>	<juan creacon ana 112233>	<se ha creado el contacto>
<maria creacat parientes>	<maria creacat parientes>	<se ha creado la categoría>	
			cd ..
<pedro cd ..>			<has cambiado a la categoría principal>

		salir	**dir**
<maria salir> <pedro dir>	<pedro dir> <maria salir>	<Has salido del sistema>	<maria salir> <lucas contacto creado por pedro> <amigos categoría creado por juan> <ana contacto creado por juan> <parientes categoría creado por maria>
			borrar ana
<pedro borrar ana>	<pedro borrar ana>		<Has borrado el contacto>
	salir		
<juan salir>	<Has salido del sistema>		<juan salir> **salir**
<pedro salir> **salir**			<Has salido del sistema>
<Has salido del sistema>			

2. Aclaraciones

- Las estructuras de datos que albergan la agenda deben mantenerse únicamente en **serviagenda**.
- No es necesario chequear unicidad en los nombres de usuario del sistema.
- En la consola del servidor se deberá ir mostrando los comandos que le llegan desde los usuarios, junto con el nombre del usuario que ejecuta dicho comando.
- Se deberá esperar un tiempo aleatorio entre 1 y 5 segundos luego de ejecutar un comando para que el comando se considere completado.
- **Los comandos de los usuarios de crear entradas (contactos y subcategorías) para una misma categoría se deberán serializar.** Es decir que mientras que se ejecuten dichos comandos ningún otro usuario podrá estar ejecutando un comando sobre esa categoría.
- Se deberá maximizar la ejecución concurrente de comandos que no impidan el correcto funcionamiento del sistema. Es decir que por ejemplo dos usuarios podrán crear entradas a la vez en distintas categorías, etc.
- El servidor y los usuarios del sistema deben recibir los comandos a través de la entrada estándar.
- La cantidad máxima de usuarios está acotada por la constante **MAX_USUARIOS**.
- La cantidad máxima de entradas en cada categoría (sumando contactos y subcategorías) está acotada por **MAX_DIR_ENTRADAS**.
- La cantidad máxima de entradas en total en el sistema está acotada por la constante **MAX_ENTRADAS**.
- El largo máximo de un nombre de contacto o categoría o número telefónico está acotado por la constante **MAX_NOMBRE**.
- Los nombres **en cada** categoría deben ser únicos (no puede haber contactos ni categorías repetidas, ni un nombre de contacto con el mismo nombre que el de una categoría). Si podrán repetirse nombres en distintas categorías.
- Si se intentan superar los topes definidos se deberá mostrar un mensaje de error pertinente.
- Cualquier situación de error no considerada en los comandos antes presentados, deberá manejarse desplegando un mensaje de error apropiado.

3. Consideraciones generales

3.1. Implementación

El trabajo debe realizarse utilizando el lenguaje de programación C y debe correr sobre el sistema operativo distribuido oportunamente para la realización de los laboratorios.

- Puede utilizarse sintaxis o funciones de biblioteca de C++, pero no clases.
- Compile utilizando make, con un archivo Makefile (sin extensión). No se aceptarán entregas sin Makefiles, o que compilan con scripts o similares.
- Deberá compilarse con la opción –Wall (un parámetro de g++). Esta opción permite ver errores y advertencias (warnings) de compilación más detalladas. El trabajo entregado **NO** debe producir warnings al compilarlo.

A continuación se enumeran algunos de los comandos con los cuales se deben familiarizar para resolver el problema planteado:

- gcc
- make
- man
- ps
- kill
- gdb
- ipcs
- ipcrm

La siguiente es una lista con algunas de las llamadas al sistema que se deben conocer para entender la sincronización entre procesos en el sistema y el mecanismo de memoria compartida.

- fork
- wait
- waitpid
- exit
- shmctl
- shmget
- shmat
- shmdt
- semget
- semctl
- semop

Otras funciones útiles:

- execv
- signal
- sleep

3.2. Documentación

Debe entregarse toda la documentación impresa y en versión electrónica. En este caso se requiere el diseño de la solución implementada (las estructuras de datos, el pseudocódigo, etc.).

No incluir la letra ni parte de ella en la documentación. Se penalizarán documentaciones que repitan los requerimientos o no describan lo implementado. Los archivos de código fuente deben estar impresos al final de la documentación, en un anexo.

Debe especificarse en el sobre de la entrega el número de grupo e integrantes.

3.3. Plazo y Entrega

La fecha tope de entrega es el día **24 de junio de 2011 a las 23:30** a través de la página web del curso.

De ninguna forma se aceptarán entregas fuera de la fecha de entrega estipulada.

4. Condiciones del Laboratorio

El trabajo es obligatorio, la no entrega del trabajo o el mal funcionamiento implican la pérdida del mismo y por lo tanto la pérdida del curso.

Cada grupo debe realizar su trabajo en forma individual, y la detección de trabajos hechos en forma no individual (entre los grupos) implica la pérdida del mismo para los grupos involucrados. En este caso también implica la pérdida del curso para todos los participantes de los grupos.

Cada grupo es responsable de proteger su trabajo contra copia. No olvide diskettes/pendrives en la sala de máquinas, ni envíe correos o mensajes al newsgroup que permitan a otros copiar su código o parte de él.

Luego de finalizada la tarea, se realizará una prueba. Los detalles de la misma se informarán a través de la cartelera del curso y el grupo de noticias.

5. Consultas

Se atenderán consultas en el newsgroup o en las clases de consulta específicas para dicho fin.

6. Bibliografía

[1] El entorno de programación Unix. (Kernighan/Pike)

[2] El lenguaje de programación C (Kernighan / Ritchie)

[3] Red Hat inc, The Official Red Hat Linux Getting Started Guide

https://www.redhat.com/docs/manuals/linux/RHL-9-Manual/getting-started-guide/

[4] Home Page del Fedora Project (http://fedora.redhat.com/)

[5] Ayuda de los comandos Unix. (en el archivo Lab1/Documentacion/ComandosUnix.txt)

[6] Cualquier tutorial básico de Unix/Linux

(por ejemplo - http://iie.fing.edu.uy/~vagonbar/unixbas/tutorial.htm)

[7] Páginas del manual de Unix on-line

(por ejemplo - http://man.he.net/)

[8] Advanced Linux Programming (http://www.advancedlinuxprogramming.com)