Sistemas Operativos
Laboratorio 3 V1

Díaz, Adolfo
adolfo.diaz@usach.cl

González, Pablo
pablo.gonzalezca@usach.cl

October 9, 2014

1 Objetivos

1. Creación y utilización de hebras en linux.
2. Utilización de hebras POSIX.
3. Utilización de mecanismos de sincronización como *mutex* y variables de condición.

2 Información preliminar

2.1 Hebras

Un proceso típico de Unix puede ser visto como un único hilo de control, es decir, cada proceso hace sólo una cosa a la vez.

Con multiples hilos podemos realizar operaciones de forma paralela, entre los beneficios que nos presentan tenemos los siguientes:

- Se puede manejar eventos asíncronos asignando un hilo a cada tipo de evento. Luego cada hilo maneja sus eventos en forma sincrónica.
- Los hilos de un proceso comparten el mismo espacio de direcciones y descriptores de archivos.
- Procesos con múltiples tareas independientes pueden terminar antes si estas tareas se desarrollan traslapadamente en hilos separados. De este modo tiempos de espera de la primera tarea no retrasan la segunda.
- Programas interactivos pueden lograr mejor tiempo de respuesta usando hilos para manejar la entrada y salida. Este es un ejemplo del punto previo.
- La creación de un hilo es mucho más rápida y toma menos recursos que la creación de un proceso.

Fuente: `http://is.gd/3pI95A`

2.2 Algoritmos de reducción criptográficos.

Cuando un usuario se registra en un sitio web, los datos ingresados en el registro pasan a formar parte de una base de datos con la finalidad de que cada vez que un usuario desee ingresar se comprueben los datos de los formularios web con los datos que existen en la base de datos.

Con la finalidad de no guardar las contraseñas en texto plano se utilizan algoritmos de reducción criptográficos y así proteger la privacidad de los usuarios, ejemplos de éstos son MD5, SHA1, SHA2 etc. Estos algoritmos tienen como entrada un conjunto de caracteres y los convierte (mapea) en un rango de salida finito, normalmente otra cadena de caracteres de longitud finita.

MD5("Me gusta trabajar con procesos") = eca167f6bd312535648e66ec2765f8e4

MD5("me gusta trabajar con procesos") = c67fa8f17f601d905efdff0bd8a81487

Cuando un usuario se conecta e ingresa su clave simplemente se comprueba si el hash MD5 de la contraseña ingresada concuerda con el hash MD5 almacenado en la base de datos. Es importante recalcar que **no existe una función que devuelva de un hash md5 su valor en texto plano.**

3 Enunciado

Un usuario malicioso ha conseguido una base de datos con la tabla de usuarios de un Sistema Web, lamentablemente para él las contraseña se encuentran codificadas con algoritmos de reducción criptográficos específicamente con MD5. Ya que no existe ninguna función que retorne la cadena de caracteres inicial una vez aplicado MD5, la base de datos para el no posee utilidad.

3.1 Solución a implementar

Con propósitos netamente educativos los alumnos del curso de sistemas operativos deberán conseguir cada una de las claves que se encuentran en la base de datos, para esto se debe aplicar fuerza bruta, es decir, dado un diccionario de palabras se realiza el algoritmo MD5 a cada una de ellas y comparar si el resultado es el mismo que se encuentra en el archivo de la base de datos, en caso de ser el mismo se debe anotar el usuario y contraseña.

3.2 Especificaciones

Las especificaciones que se piden de la solución a implementar son las siguientes:

- La solución se debe implementar con hebras POSIX.
- El programa recibe por parámetro el numero de hebras que se desean utilizar.

- No deben existir condición de carrera en la ejecución del programa.
- Se debe implementar una cache que siga políticas LFU, el tamaño de la cache se recibe por parámetros.

3.3 Algoritmo

A continuación se presenta un pseudo-código que puede ser considerado para su implementación.

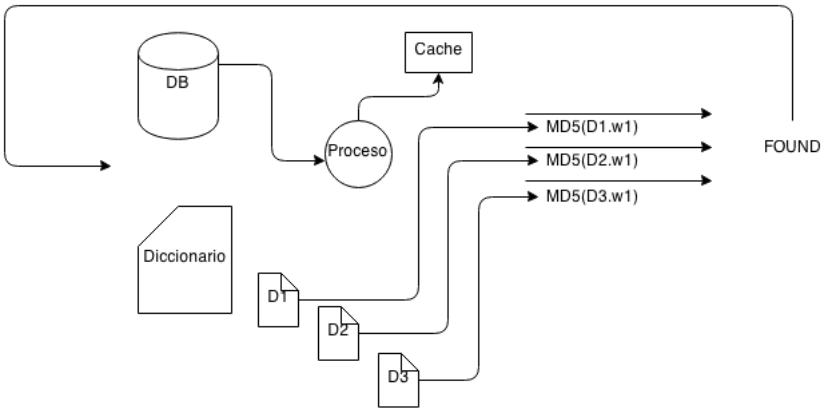

Figure 1: Laboratorio 3.

1. Se lee la base de datos y se obtiene la lista de usuarios y sus contraseñas en MD5.
2. Se obtiene la primera contraseña en MD5 se guarda en una variable.
3. Se verifica si se encuentra en la cache, si se encuentra se guarda el resultado y se pasa a la siguiente.
4. Si no se encuentra, y es la primera ejecución, se obtiene el numero de hebras que se desean utilizar y se procede a dividir el diccionario.
5. Cada hebra empieza a aplicar la función MD5 a cada palabra de su partición del diccionario para tratar de descubrir la contraseña del paso 2.
6. Si el resultado de la función MD5 es igual a la variable que guardamos en el paso 2, significa que se ha encontrado la contraseña y por lo tanto todas las hebras paran de buscar, el resultado se guarda en la cache.
7. Se reinicia la búsqueda con la siguiente contraseña de la base de datos.

8. Cuando el programa termine con todas las contraseñas de la base de datos, debe imprimir el resultado, que consta de los nombres de usuarios y contraseñas que se encontraban en la base de datos. El orden debe ser el mismo con el que se leyeron los datos.

4 Ejemplos de Ejecución

./Laboratorio3 -r database -d diccionario -c tamañoCache -h numeroHebras

En donde la bandera *-r* indica la base de datos y la bandera *-d* el diccionario. Considere que la base de datos se encuentra siempre correcta y que el diccionario se compone de una palabra por linea.

4.1 Ejemplo diccionario

Baalita
Babá
Babada
Babadero
Babador
Babahoyense
Babanca
Babanco
Babatel
⋮

4.2 Ejemplo de archivo de base de datos SQL

El ejemplo de la base de datos se encuentra en el anexo.

5 Consideraciones

- El programa puede recibir las banderas en cualquier orden.
- El binario se debe llamar laboratorio3.
- El programa debe ser subido a Usach Virtual en un archivo .tar con el rut del alumno, se aceptan envíos con hasta 24 horas de atraso con nota máxima un 4, cualquier entrega superior a este umbral sera clasificado con nota mínima.
- Las pruebas se realizaran en los computadores del laboratorio de informática.

6 Evaluación

La correcta implementación de la tarea obtendrá nota 7.0, se realizara una pauta de recorreción la cual sera dada a conocer a la entrega de las notas. Cualquier situación que no esté contemplada en el enunciado será resuelto el ayudante. El plazo de entrega es el martes 28 de octubre antes de las 9:30 hrs.

7 Anexo

7.1 Archivos

A continuación se presenta el diccionario y la base de datos que se tiene que utilizar.

7.1.1 Diccionario

http://is.gd/0aHoBm

Mirror:
http://is.gd/OOleR1
http://is.gd/s5XOAP

7.1.2 Archivo base de datos

http://is.gd/wEcyPh

7.2 Archivo Base de datos

El formato que posee el archivo de la base de datos es el siguiente, para mayor información ver el archivo disponible en http://is.gd/wEcyPh .

```
-- phpMyAdmin SQL Dump
-- version 4.1.12
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generacion: 07-10-2014 a las 13:47:26
-- Version del servidor: 5.6.16
-- Version de PHP: 5.5.11

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
```

```
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de datos: `TodoOferta`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `id_user` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(200) NOT NULL,
  `correo` varchar(200) NOT NULL,
  `pass` varchar(200) NOT NULL,
  PRIMARY KEY (`id_user`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;

--
-- Volcado de datos para la tabla `user`
--

INSERT INTO `user` (`id_user`, `nombre`, `correo`, `pass`) VALUES
(2, 'Tommy', 'tommy@gmail.com', '6074c6aa3488f3c2dddff2a7ca821aab'),
(3, 'Cristian', 'Cristi1313@gmail.com', 'fd99e2a1f4a2c60693a0475ad468a3ea'),
(4, 'Romina', 'romy_xxx@gmail.com', '59e9b88d1cf13e9c057aef3e82a3f6ac'),
(5, 'Ana', 'Anabellio@gmail.com', 'c52665c245f78f8d4edc6d27adb7ec0a');

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```