Actividad Práctica 1

Bash Scripting

Profesor: Ismael Figueroa
Equipo docente: Diego Arias, Fabián Díaz, Andrés Gallardo, Blaz Korecic

Bash es una herramienta súper flexible que facilita la automatización de ejecución de comandos en entornos UNIX-like e incluso puede llegar a ser una pieza fundamental de algunos softwares y distribuciones (como el gestor de paquetes de Arch Linux).

Esta actividad está centrada en que se familiaricen con la sintaxis de Bash y valoren el poder de abstracción y automatización que ofrece, como otros lenguajes de programación, pero enfocado en el sistema.

El objetivo es que desarrollen un sencillo juego de terminal al estilo de «búsqueda del tesoro», utilizando principalmente los comandos vistos en clases y auxiliares:

Los comandos que estarán prohibidos para el juego son **`grep`** y **`find`**, pues le quitarían la gracia.

Adicionalmente, deben asegurarse de tener instalados los siguientes paquetes:

- **gpg**
- **openssl**

pues serán necesarios para implementar los distintos modos (independientes entre sí) del juego:

a) `name`: El «tesoro» es el nombre de un archivo
b) `content`: El «tesoro» es el contenido de un archivo
c) `checksum`: El «tesoro» es el *checksum* de un archivo
d) `encrypted`: El «tesoro» es un archivo encriptado usando **gpg**
e) `signed`: El «tesoro» es un archivo firmado con una llave de **openssl**

Un método para guardar y compartir el estado del programa entre scripts es mediante archivos que pueden crear en `/run`, `/var` o `/tmp`. Pueden usar otros métodos pero deben detallarlos.

P1. Generar el tablero (1 pts)

El primer paso para construir nuestro juego es tener un tablero. En este caso, nuestro tablero será el sistema de archivos de Linux. Deben crear un script llamado **`board.sh`** que se encargará de crear los directorios y archivos entre los que deberán esconder el «tesoro». Además, deben procurar que el script se encargue también de limpiar cualquier rastro que pudo haber dejado una ejecución anterior.

Los parámetros del tablero son:

1. `depth`: Profundidad máxima de directorios (niveles)
2. `width`: Cantidad de directorios por nivel
3. `files`: Cantidad de archivos por directorio

Se debe tener una función **`create_board`** que cree los directorios y archivos (vacíos por ahora), y otra función **`clean_board`** que los limpie. El nombramiento de archivos y directorios es libre.

A continuación se muestra un ejemplo de cómo quedaría un tablero con `depth=3`, `width=2` y `files=4`:

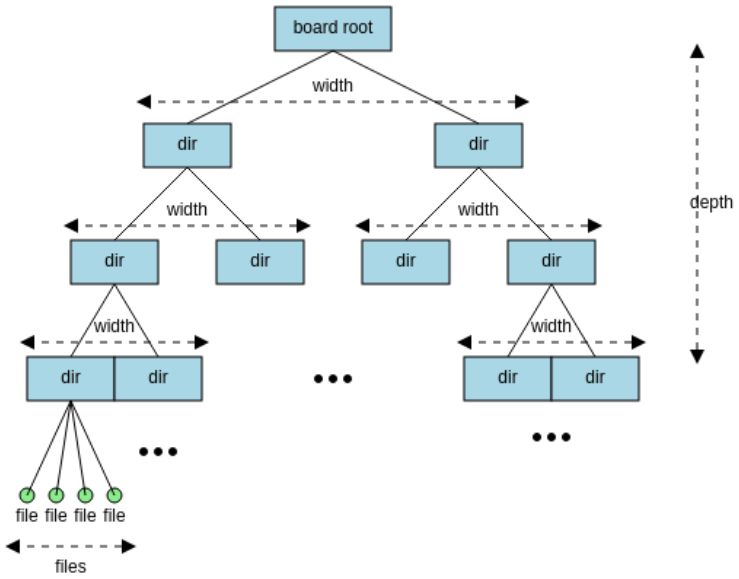

Figura 1: Ejemplo de un tablero con parámetros depth=3, width=2 y files=4.

Notar que los archivos sólo quedan dentro de los directorios del último nivel.

P2. Llenar el tablero (0.4 pts / variante)

Una vez creado el tablero hay que empezar a llenarlo. Creen una función **`fill_board`** en `board.sh` que recorra la estructura de ficheros y escriba contenido a los archivos.

El contenido de los archivos y/o los procesos posteriores que se les aplicarán dependerán del modo de juego:

	`name`	`content`	`checksum`	`encrypted`	`signed`
Contenido de los archivos	Libre, pueden quedar vacíos	Deben contener caracteres aleatorios	Deben contener caracteres aleatorios	Libre pero no pueden quedar vacíos	Libre pero no pueden quedar vacíos
Postprocesado de los archivos	Ninguno	Ninguno	Ninguno	Encriptar todos con la misma passphrase	Firmar todos con la misma llave privada

P3. Escoger el tesoro (0.3 pts / variante)

Después de que se haya generado el tablero, se hayan creado los archivos y se les hayan realizado los procesos posteriores según corresponda. En un script **`controller.sh`** se deberá definir la función **place_treasure** que debe elegir un archivo al azar y:

a) `name:` Retorne el nombre del archivo escogido

b) `content`: Retorne el contenido del archivo escogido
c) `checksum`: Retorne el checksum del archivo escogido
d) `encrypted`: Genere una segunda passphrase, distinta a la anterior, se debe reemplazar el archivo seleccionado por uno nuevo, encriptado con la nueva passphrase escogida y se debe retornar la passphrase nueva.
e) `signed`: Generar un nuevo par de llaves, reemplazar el archivo seleccionado y firmarlo con la nueva llave privada. Se debe retornar la nueva llave publica.

P4. ¿Es el tesoro? (0.3 pts / variante)

Para poder buscar el tesoro se deben tener mecanismos de validación de candidatos, por lo que deben definir la función **verify** en `controller.sh` que, dado un path, revise el archivo *candidato* y responda si corresponde al tesoro o no de acuerdo a los siguientes criterios:

a) `name`: El nombre del candidato es igual la key
b) `content`: El contenido del candidato es igual a la key
c) `checksum`: El checksum del candidato es igual a la key
d) `encrypted`: El candidato se puede desencriptar usando la key (passphrase)
e) `signed`: El candidato se puede verificar usando la key (llave pública)

P5. ¡A buscar el tesoro! (1 pts)

Lo último que queda es unir tanto la generación del tablero como el controlador, para ello creen un nuevo script **`game.sh`** que sirva de menú y usndo los scripts `board.sh` y `controller.sh` se encargue de:

a) Inicializar el juego generando el tablero, llenándolo y escondiendo el tesoro.
b) Ejecutar el *main loop* que consistirá en recibir paths ingresados por el usuario y validarlos.
c) Permitir salir del *main loop*, indicando el path del tesoro.

Entregables

Deberán entregar un archivo `.tar` que contenga los scripts solicitados y cualquier otro que sea necesario para la ejecución de su programa, además de un README.md con las instrucciones para jugar y cualquier decisión de diseño importante que hayan tomado (mecanismo de nombramiento de las carpetas, generación del contenido de los archivos, método de persistencia de estado, etc.)