Ejercicios parciales SOA- Introducción a los sistemas operativos

1) Realice un script que verifique cada 5 segundos si el proceso **apache** se está ejecutando en el sistema. El script debe correr indefinidamente hasta que verifique que en 10 ocasiones ha encontrado el proceso **apache** corriendo, lo cual debe causar la terminación del proceso. Informar dicha acción en pantalla y posteriormente causar la terminación del script con condición de retorno 50
2) Realice un script que agregue a un arreglo los nombres de los archivos del directorio /etc cuya terminación sea .conf e implemente las siguientes funciones
    a. cantidad: imprime la cantidad de archivos del /etc con terminación .conf
    b. verArchivos: imprime los nombres de todos los archivos del /etc con terminación .conf
    c. existe: recibe como parámetro el nombre de un archivo con terminación .conf e imprime en pantalla si existe en el /etc o no
    d. eliminar: recibe como primer parámetro el nombre de un archivo con terminación .conf y como segundo parámetro la cadena lógico o físico. Si el segundo parámetro es lógico, solo borra la entrada en el arreglo, si es físico borra la entrada en el arreglo y en el FileSystem informando en cada caso la acción realizada
3) Escriba un script que recibe una cantidad desconocida de parámetros al momento de su invocación (debe validad que al menos se reciba uno). Cada parámetro representa la ruta absoluta de un archivo o directorio del sistema. El script deberá iterar por todos los parámetros recibidos y solo para aquellos parámetros que se encuentran en posiciones impares verificar si el archivo o directorio existen en el sistema, imprimiendo en pantalla qué tipo de objeto es (si archivo o directorio). Además deberá informar la cantidad de archivos o directorios inexistentes en el sistema
4) Realice un script que implemente a través del uso de funciones las operaciones básicas sobre arreglos
    a. Inicializar: crea un arreglo llamado array_vacio
    b. Agregar_elem <parametro1> elimina del arreglo el elemento que se encuentra en la posición recibida como parámetro. Debe validad que se reciba una posición válida
    c. Longitud: imprime la longitud del arreglo en pantalla
    d. Imprimir: imprime todos los elementos del arreglo en pantalla
    e. inicializarConValores<parametro1> crea un arreglo con longitud <parametro1>
5) Crear un script que reciba como parámetro el nombre de un directorio. Deben validad que el mismo exista y de no existir causar la terminación del script con código de error 4. Si el directorio existe, deberá contar por separado la cantidad de archivos que en él se encuentran para los cuales el usuario que ejecuta el script tiene permiso de lectura y escritura e informar dichos valores en pantalla. En caso de encontrar subdirectorios, no deberán procesarse y tampoco deberán ser tenidos en cuenta para la suma a informar
6) Implemente un script que agregue a un arreglo todos los archivos del directorio /home cuya terminación sea .doc. Adicionalmente, implemente las siguientes funciones que le permitan acceder a la estructura creada
    a. verArchivo <nombre_de_archivo> imprime el archivo en pantalla si el mismo se encuentra en el arreglo. Caso contrario imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 5
    b. cantidadArchivos: imprime la cantidad de archivos del /home con terminación .doc
    c. borrarArchivo <nombre_de_archivo> consulta al usuario si quiere eliminar el archivo lógicamente. Si el usuario responde Si, elimina el elemento solo del arreglo. Si el usuario responde No, elimina el archivo del arreglo y también del FileSystem. Debe validad que el archivo exista en el arreglo. En caso de no existir, imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 10
7) Escribir un script de bash que reciba un nombre de usuario como parámetro. Si el usuario ingresado como parámetro NO es un usuario del sistema el script debe finalizar con el código de error 1. En caso contrario, debe verificar cada 30 segundos si el usuario está logueado. Al detectar que está logueado debe registrar en un archivo access<NOMBRE DE USUARIO>.log el nombre del usuario junto con la fecha en que se detectó. NO debe borrar el contenido previo en este archivo y este archivo debe escribirlo en una ubicación apropiada del file system. Al detectar 30 veces que el usuario está logueado el script debe finalizar retornando el código de error que indica que el proceso es exitoso

8) Escriba un script que imprima en pantalla la cantidad de archivos del directorio /var/log/$SERVICE que contiene un patron de texto que el usuario pasa como parámetro, $SERVICE es una variable de entorno, si la misma no posee ningún valor debe tomar el valor por defecto “local_service”. El script debe finalizar retornando el valor que indica que le proceso se ejecutó correctamente
9) Se desea saber el consumo de recursos del sistema por parte de los usuarios. Desarrolle un script que se correrá ingresando como argumento los nombres de los usuarios de los cuales se necesita obtener dicha información, no se sabe cuántos son con anticipación. Para la realización del script debe definir las siguientes funciones
    a. esta_logueado(): devuelve true si el usuario pasado como argumento está logueado
    b. cant_procesos(): devuelve la cantidad de procesos que está corriendo el usuario
    c. uso_de_procesos(): devuelve true si el usuario pasado como argumento, usando la función cant_procesos() está corriendo más de 100 procesos

    Usando la funcionalidad anterior, el script debe guardar en un archivo de log, en una ubicación apropiada del file system, el nombre de los usuarios que estén corriendo más de 100 procesos junto con la fecha de ejecución del script

10) Escriba un script que reicba una cantidad desconocida de parámetros al momento de su invocación (debe validad que al menos se reciba uno). Cada parámetro representa la ruta absoluta de un archivo o directorio en el sistema. El script deberá iterar por todos los parámetros recibidos y
    a. Si el parámetro es un archivo existente en el file system deberá comprimirlo dejando el archivo comprimido en lugar del original
    b. Si el parámetro es un directorio existente en el file system
        i. Si tiene permiso de lectura deberá empaquetarlo y comprimirlo
        ii. Si tiene permiso de escritura deberá eliminarlo junto con todo su contenido
    c. Si el parámetro no es un elemento del file system deberá contarlo e indicar la cantidad total al finalizar el script
11) Realice un script que agregue todos los nombres de usuario del sistema a un arreglo e implemente las siguientes funciones:
    a. existe<parametro1>: devuelve 0 si el usuario existe en el arreglo, 1 en caso contrario
    b. eliminar_usuario<parametro1>: si el usuario pasado como parámetro existe en el arreglo, lo elimina del mismo. Caso contrario devuelve código de error 2
    c. usuarios_con_patron<parametro1>: recorre el arreglo e imprime en pantalla los nombres de los usuarios en cuyos caracteres aparece el patrón pasado como parámetro
    d. cantidad: imprime la cantidad total de usuarios en el arreglo
    e. usuarios: imprime todos los nombres de los usuarios que están en el arreglo
12) Realice un script que implemente a través de la utilización de arreglos las siguientes funciones
    a. Insert_elemento: la función Insert recibe un parámetro (validad que así sea), agrega al final del arreglo el elemento recibido como parámetro
    b. Rellenar_n: esta función itera n veces solicitando al usuario que ingrese un patrón de texto y lo agrega al final del arreglo. Debe validad que se reciba 1 parámetro
    c. Select_elemento: la función select recibe un parámetro
        i. Si el parámetro es * entonces imprimirá todos los elementos del arreglo
        ii. Si el parámetro es distinto de *, deberá verificar si en el arreglo existe un elemento que sea igual al parámetro. Si existe deberá imprimirlo en pantalla, caso contrario imprimirá “Elemento no encontrado”
    d. Delete_elemento: la función delete recibe un parámetro
        i. Si el parámetro es * entonces eliminará todos los elementos del arreglo
        ii. Si el parámetro es distinto a * deberá verificar si en el arreglo existe un elemento que sea igual al parámetro. Si existe deberá eliminarlo, caso contrario imprimirá “Elemento no encontrado”
13) Realice un script que verifique cada 3 minutos si existe el archivo /var/log/any-service/error.log. Si el archivo existe y dentro de él hay al menos una línea con el texto “FATAL ERROR” debemos comprimir y

empaquetar el contenido del directorio /var/log/usr-service, guardarlo en un directorio que es pasado como parámetro al script y además causar la terminación del script apropiadamente. Cuando el script termina debemos informar por la salida estándar la cantidad de líneas del archivo que contenían el texto “FATAL ERROR”