Facultad de Ingeniería
Universidad de Buenos Aires

75.08 Sistemas Operativos

Prof. Lic. Ing. Osvaldo Clúa
Jefe T.P. Lic. Adrián Muccio

Guía de Ejercicios Shell y ER

Autores

Lic. Sandra Abraham
Lic. Guido Fernandez
Lic. Adrián Muccio

Versión 2.0

Índice

I. Introducción

La presente Guía contiene ejercicios relacionados a temas propios de la Práctica

El listado de ejercicios no contempla el alcance completo de los temas, simplemente tiene el fin de servir como herramienta de apoyo a la cursada regular.

No es obligatoria la resolución de los ejercicios

Se incluye la resolución de algunos ejercicios a modo ejemplificativo sin perjuicio de que existan otras posibles soluciones.

Los ejercicios se encuentran ordenados por tema y nivel de dificultad.

Los ejercicios tipo parcial no tienen orden definido.

II. Shell Scripting

1 Fecha Actual

Obtenga la siguiente salida (tal cual está, en dos líneas):

Son las hh:mm:ss del día: dd

del mes: mm del año: aa.

Solución:

date +"Son las %T del día %e %n del mes %m del año %y."

2 Directorio

Realizar un shell script que dado un directorio pasado por parámetro, cree un archivo tar comprimido con gzip y con nombre igual a la fecha en formato yyyymmdd seguido de guión y seguido del nombre del directorio terminado en .tar.gz.

Ejemplo: aplicado sobre home obtendríamos -? 2004-04-03-home.tar.gz.

3 Tabla de multiplicar

Realizar un shell script que dado un número 'n' pasado por parámetro muestre los diez primeros elementos de su tabla de multiplicar, mostrando el resultado en la forma: i x n = resultado.

4 Divisible

Realizar un shell script que, dado un número pasado por parámetro, indique si es o no divisible entre 101. Si no se proporciona un número debe mostrar como usar el programa.

5 Comprimir

Realizar un shell script que dado una lista de directorios, cree un archivo tar comprimido con gzip con nombre igual a la fecha en formato yyyy-mm-dd.tar.gz. Además se generará un archivo yyyy-mm-dd.lst con los nombres de los directorios contenidos en el archivo tar, UNO POR LINEA usando un bucle. Si el archivo lst existe, mostrar un error y terminar el programa. Si alguno de los elementos no es un directorio, mostrar un error y finalizar el programa.

6 PID

Realizar un shell script que permita adivinar al usuario cual es su PID. El script pide un número al usuario y cada vez que lo haga debe indicar al usuario si el PID es mayor o menor que el número introducido. Cuando se adivina el

valor, se deben mostrar los intentos empleados.

7 Verificación

Realizar un shell script que verifique cada 30 segundos si existe en el directorio actual un archivo prueba.txt. Para probar este guión es necesario ejecutarlo en segundo plano.

8 Directorios

Crear un shell script que liste todos los directorios y subdirectorios recursivamente de uno dado. El directorio será introducido como argumento y el guión lo primero que hará será verificar si es precisamente un directorio.

9 Proceso

Hacer un shell-script llamado 'proceso.sh' para ser lanzado en background que como máximo permanecerá vivo 30 segundos.

Podrá ser lanzado varias veces en background y cada vez generará un shell-script distinto 'stop_proceso.sh.' que al ser ejecutado matara el proceso lo origino y después se borrará a si mismo.

10 Pirámide

Realice un shell script 'piramide.sh' que reciba por entrada std (no por parámetro) un número 'N' y que obtenga a la salida una serie de 'N' filas de forma triangular.

Para ./piramide.sh 8 la salida sería.

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7 7

8 8 8 8 8 8 8 8

11 Igualdad

Realice un shell-script que admita tres palabras como argumentos y que muestre un mensaje informando de las relaciones de igualdad y desigualdad entre esas palabras.

"Las tres son iguales"

"Son iguales primera y segunda"

"Son iguales primera y tercera"

"Son iguales segunda y tercera"

"Son las tres distintas"

12 LOG

Asumiremos que tenemos en un directorio una serie de archivos de log que van creciendo de forma ilimitada con el uso regular de ciertos programas.

Realizar un shell script que actúe sobre los archivos con nombre tipo '*.log' del directorio actual de forma tal, que si alguno de ellos supera en tamaño las 2000 líneas, dejará solo las últimas 1000 líneas del archivo y las restantes serán guardadas en un directorio old_rot en formato comprimido.

En dicho directorio habrá que conservar en formato comprimido no solo la última porción eliminada del original, sino las cuatro últimas porciones eliminadas. Para ello será necesario ir rotando los nombres de estos archivos comprimidos incorporando en los mismos un digito de secuencia.

(parte eliminada) --> *.log.rot1.gz --> *.log.rot2.gz --> *.log.rot3.gz --> *.log.rot4.gz --> eliminado

El proceso durante su ejecución irá mostrando los archivos encontrados y señalará aquellos que por su tamaño sea necesario rotar.

13 Contenido

Escribir un shell-script denominado fileodir que compruebe que el parámetro que se le ha pasado es un archivo y en caso afirmativo muestre su contenido. Si el parámetro es un directorio deberá mostrar los archivos que contiene. También, debe aceptar más de un parámetro.

14 PATH

Hacer un shell-script que busque la presencia del comando pasado como argumento en alguno de los directorios referenciados en la variable $PATH, señalando su localización y una breve descripción del comando caso de existir su página man.

15 Rango

Realizar un shell-script que reciba como argumentos números comprendidos entre 1 y 75. Dará error en caso de que algún argumento no esté dentro del rango y terminará sin hacer nada. En caso contrario generará una línea por cada argumento con tantos asteriscos como indique su argumento.

16 Archivos

Realizar un shell-script que acepte como argumentos nombres de archivos y muestre el contenido de cada uno de ellos precedido de una cabecera con el nombre del archivo

17 Media Aritmética

Hacer un shell-script que calcule la media aritmética de todos los argumentos pasados como parámetros con una precisión de 40 dígitos decimales después de la coma.

III. Expresiones Regulares

Solo se pueden utilizar comandos sed, grep, wc, echo.

1 Capicúas

Se tiene un archivo con números enteros de 3 dígitos, se desea generar otro archivo con los capicúas de cada uno de los números

2 Reformatear Fecha

Se tiene un archivo de texto en el que aparecen fechas con el siguiente formato mm/dd/aaaa se desea cambiarle el formato a dd/mm/aaaa

Solución Ejemplo

```
                    1          2          3
sed 's-\([0-1] [0-9]\)/\([0-3][0-9]\)/\([0-9] \{4\}\)-\2/\1/\3-g'
                 -----mes----  -----día-----  -----año----
```

3 IP

Realizar un shell script que reciba por parámetro una ip (EJ: 192.168.1.1) e indique si es válida o no.

4 FECHA

Realizar un shell script que tome por entrada std una fecha (formato YYYY-MM-DD) e indique si es válida o no.

5 EMAIL

Realizar un shell script que reciba por parámetro un email e indique si es válido o no.

6 DEBITO

Realizar un shell script que busque dentro del archivo debito-automatico.txt que solo contiene números de tarjetas de crédito de Visa (empiezan con 4 y tienen 16 dígitos), Mastercard (empiezan del 51 al 55 inclusive y tienen 16 dígitos), American Express (empiezan del 34 al 37 y tienen 15 dígitos) y muestren por pantalla el número con su respectiva denominación.

Ejemplo:

4312-4311-4311-4123:VISA

5123-1231-1231-1231:MASTERCARD

7 DECIMALES

Realizar un shell script que busque dentro del archivo datos.txt los números decimales mayores a 7.534

ejemplo línea datos.txt

1:Juan Perez:9.321

2:Juan Paso:3.321

3:Matias Perez:7.534

4:Fernando Poso:7.999

8 /ETC/PASSWD

a. Mostrar los nombres de todos los usuarios de la máquina:

b. Mostrar los nombres de los usuarios, pero sólo los que usan bash (/bin/bash):

9 CANCIONES

Dado una duración de canción pasada por parámetro con el siguiente formato (hh:mm:ss) indicar si es mayor o menor al límite establecido:

LIMITE="00:03:12"

10 TEXTO

Dado el siguiente archivo de texto:

lipsum.txt

"Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Duis suscipit dictum urna et luctus. Ut non elementum urna, ac ultricies justo. Nulla a lacus rutrum, sollicitudin neque interdum, scelerisque mauris.

Mauris tempor rhoncus tincidunt. Mauris dignissim venenatis risus, sit amet tincidunt risus laoreet vehicula.

Etiam volutpat libero ac ipsum vestibulum elementum. Morbi eget diam non tellus mattis malesuada.

Sed fermentum felis tempus nisi venenatis dictum.

Nam suscipit lacinia nisl ut pulvinar. Proin id enim condimentum, ultrices leo quis, auctor tellus.

Aliquam porttitor nibh felis. Integer pharetra elementum libero rhoncus egestas.

Etiam at aliquet elit, ac feugiat nunc. Proin id lorem viverra mi consectetur tempus ut sed neque.

Nullam scelerisque congue accumsan. Nunc tincidunt tellus odio, eu fermentum mi tempor at.

Etiam et ullamcorper elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos."

a. Realizar un shell script que busque dentro del archivo de texto palabras en latín pasadas por parámetro.

b. Reemplace en las líneas que empiezan con E, la quinta palabra por el texto "ELIMINADO"

c. Elimine las líneas que terminan tienen la palabra "sed"

d Inserte comentarios con la siguiente notación: "#COMENTARIO: " al final de cada línea

I. Ejercicios Tipo Parcial

ER-Pay Per View

Una empresa operadora de televisión para su servicio de canales con cargo adicional, conocido como “Pay Per View”, desea que sus clientes puedan adquirir este servicio por medio de un mensaje SMS.

Nos pide desarrollar un script que sea invocado por una operadora telefónica y nuestro script registre la venta invocando a su vez a un API de su sistema CRM.

El script debe recibir como parámetros de entrada:

- Número de teléfono origen
- Código del cliente
- Canal

El Número de teléfono origen se envía con el siguiente formato fijo *(nn)(nnnnnn)nnnn*.

Donde ‘*n’:* significa dígito

Los caracteres 6 a 11 contienen el código de área

Los caracteres 13 a 16 contienen el número

El API a invocar es un programa que se llama RegistrarVentaPPV y recibe como parámetros el código de cliente y la señal PPV.

Por restricciones técnicas, en un mismo canal, no todos los clientes ven las mismas señales.

La señal se determina en base al código de área del teléfono y el canal recibidos.

Las relaciones se encuentran en el archivo Signals&Chanels.dat, que posee el siguiente formato:

Nombre de Campo	Descripción	
ID_RELACION	Identificación de Relación (*)	Todos los campos se encuentran separados por el caracter ;

SEÑAL	Señal (*)	El (*) indica que se desconocen tipo y formato del campo. Existe a lo sumo un canal por cada dupla de SEÑAL y CODIGO_AREA
NOMBRE_LARGO	Nombre largo de la señal (*)	
CODIGO_AREA	Código de área. Tipo: numérico de 6 posiciones Formato: de ser necesario se rellena con 0s a la izquierda.	
CANAL	Código del canal (*)	
COMENTARIOS	Comentarios varios (*)	

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, bc, let y echo

ER-Centro de Despacho

En una empresa de servicios, el Centro de Despacho asigna las órdenes de trabajo a los técnicos y controla su ejecución. Este sector requiere un script que reciba como parámetro un identificador de técnico y devuelva por salida std el código de la orden de trabajo que tiene asignada y la hora con minutos y segundos en que fue planificada. A lo sumo existe una orden asignada para un técnico en un día.

Para esto se cuenta con el archivo de asignaciones del día, que posee el siguiente formato:

ID_ASIGNACION	Identificador de Asignación. Tipo Alfanumerico. Se desconoce formato y longitud	Todos los campos se encuentran separados por el caracter ;
COD_OT	Código de Orden de Trabajo (*)	El (*) indica que se desconocen tipo y formato del campo
TIPO_OT	Tipo de Orden de Trabajo (*)	
ID_CLIENTE	Identificador de Cliente (*)	
DIRECCION	Dirección (*)	
ID_TECNICO	Identificador del Técnico (*)	
FECHA_PLAN	Fecha Planificada. Tipo Fecha. Formato: dd/mm/aaaa hh:mi:ss	
ESTADO	Estado de la Orden (*)	
FECHA_REAL	Fecha Real. Tipo Fecha Formato: dd/mm/aaaa hh:mi:ss	
COMENTARIO	Comentarios varios (*)	

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, sort y echo

ER-Juegos Panamericanos

Para evaluar los resultados de cada uno de los países participantes de los Juegos Panamericanos se nos pide desarrollar un script que reciba como parámetros un nombre de país, un nombre de disciplina y muestre por salida std el mensaje:

"<Nombre_de_país> obtuvo una medalla <metal_de_medalla> en <nombre_de_disciplina>" si ganó una medalla.

O el mensaje "<Nombre_de_país> no obtuvo ninguna medalla en <nombre_de_disciplina>"

Para esto se cuenta con el archivo Resultados.dat, que posee un registro por cada combinación de país-disciplina y cuenta con el siguiente formato:

Nombre de Campo	Descripción	
CODIGO_PAIS	Código de país (*)	Todos los campos se encuentran separados por el caracter ;
NOMBRE_PAIS	Nombre de país (*)	El (*) indica que se desconocen tipo y formato del campo
CODIGO_DISCIPLINA	Código de disciplina (*)	
NOMBRE_DISCIPLINA	Nombre de disciplina(*)	
FECHA_FINAL	Fecha de última prueba(*)	
ORO	Caracter S o N	
PLATA	Caracter S o N	
BRONCE	Caracter S o N	
BATIO_RECORD	Caracter S o N	
COMENTARIOS	Comentarios varios (*)	

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, bc, let y echo

ER-Clasificador de Errores

Se nos pide realizar un script para clasificar errores llamado ClasificaErrores.sh que recibe como parámetro un código de error y devuelve por salida std la clasificación que le corresponda.

Para clasificar los códigos se debe aplicar la siguiente lógica según el código enviado.

Buscar en archivo CLASE_DE_ERRORES.dat, ingresando por el campo ERROR_CODE y obteniendo el campo “ERROR_CLASS”

En caso de no encontrar clasificación para el código, debe devolver el código encerrado entre “<>” y seguido del mensaje “ ***- No Clasificado***”

El archivo CLASE_DE_ERRORES.dat posee el siguiente formato:

ID_CLASS	(*)	Todos los campos se encuentran separados por el caracter :
ERROR_CODE	(*) (**)	(*) indica que se desconocen tipo y formato del campo
ERROR_DESCRIPCION	(*)	(**) indica que no hay más de una línea con el mismo valor en ese campo
ERROR_CLASS	(*)	
LAST_UPDATED_BY	(*)	
LAST_UPDATE_DATE	(*)	

Ejemplo de invocación del script:

> ClasificaErrores.sh ”*código de error*”

<*código de error*> - No Clasificado

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, sort y echo

ER-Actualizador de Precios

Un supermercado nos pide realizar la integración para actualizar los precios entre los sistemas ERP y Flejes (que muestra los precios de los productos en góndola).

Para esto desarrollaremos el script ActualizarPrecioERP.sh que recibirá por parámetro el identificador de artículo del ERP, el precio y el id de sucursal destino.

Nuestro script traducirá el identificador de artículo al código standard internacional EAN 13, enviará el id de sucursal destino y cambiará el formato del precio para finalmente invocar al API SetPriceFlejes.

El código de retorno del script debe ser el mismo código que devuelve el API.

El formato de precio de que envía el ERP no tiene separador de decimales, interpreta los últimos dos caracteres como los centavos, pero el formato esperado por Flejes tiene el carácter '.' como separador. Ejemplo '1000' à '10.00'

Para obtener el código EAN13 se cuenta con el archivo ARTICULOS.dat, que posee el siguiente formato:

ID_ARTICULO	Identificador de artículo (*)	Todos los campos se encuentran separados por el caracter ;
NOMBRE	Nombre del artículo (*)	El (*) indica que se desconocen tipo y formato del campo
DESCRIPCION	Descripción del artículo (*)	
COD_EAN13	Código EAN 13 (*)	
COD_EAN7	Código EAN 7 (*)	
OTROS	(*)	

Ejemplo de invocación del script:

ActualizarPrecioERP.sh <id_artículo> <precioERP> <id_sucursal>

Ejemplo de invocación del API:

SetPriceFlejes <*EAN_13*> < *id_sucursal*> <*precioFLEJES*>

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, sort y echo

ER-Medidor de Distancia

La Agencia Nacional de Vialidad nos pide desarrollar un script llamado ***distancia.sh*** que muestre por salida standard la cantidad de kilómetros de distancia entre dos localidades unidas por una ruta.

El script recibe por entrada standard una línea que contiene el nombre de la localidad origen, el nombre de la localidad destino y como tercer valor, el número de ruta.

Los valores se encuentran separados por el caracter “%”

Contamos con el archivo Info_rutas.dat que contiene en que kilómetro se encuentra cada una de todas las localidades según las rutas de las que formen parte.

El archivo Info_rutas.dat posee el siguiente formato de campos separados por el caracter '-''

- ID
- Ruta
- Localidad
- Kilómetro relativo al inicio de la ruta

El script ***distancia.sh*** va a ser invocado en forma automática. Es por eso que todos los valores de la entrada se consideran válidos (cantidad, formato y valores existentes en archivo Info_rutas.dat). Se utiliza el mecanismo de pipe “|” de manera de optimizar los tiempos de procesamiento.

Ejemplo de invocación:

$ Cmd1.exe | ***distancia.sh***

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, bc y echo

ER-Disponibilidad Hotelera

Una agencia de turismo nos pide desarrollar un script que muestre por salida std **SOLO** la cantidad de habitaciones disponibles en los hoteles a la fecha de inicio del próximo carnaval, esto es el 17/02/2013

El script recibe por entrada std la cantidad de huéspedes y el número de estrellas ambos campos separados por el carácter **','**

Para obtener la cantidad de habitaciones disponibles se cuenta con el archivo disponibilidad.dat que posee el siguiente formato de campos separados por el caracter **'-''**

- ID
- Hotel
- Estrellas
- Habitación
- Capacidad de huéspedes de la habitación
- Fecha #formato mm/aa/aaaa
- Estado # el estado disponible tiene como valor **'DISP'**

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, bc y echo

ER-Filtro Musical

Se desea realizar un script llamado FiltroMusical que reciba de la entrada std un género musical y muestre por salida std todos los autores y los nombres de los temas cuya duración se encuentre dentro del rango entre 30 minutos y una hora con 20 minutos inclusive.

La información de los temas musicales se encuentra en un archivo llamado Lista.Musical con el siguiente formato:

FECHA DE CREACION	
ALBUM	
NOMBRE DEL TEMA	
DURACION	Formato: HH:MM:SS
INTERPRETE	
AUTOR	
GENERO	

Todos los campos se encuentran separados por el caracter ;

Ejemplo de invocación del script:

> echo "Opera" | Filtro.Musical

Ejemplo de salida:

Mozart-Las bodas de Figaro

Verdi-La traviata

IMPORTANTE: Solo se permite el uso de los comandos grep, sed, wc, y echo

Shell- Facturas

Se reciben en un directorio `/input` archivos de diversos proveedores con la información

de las facturas por estos emitidas, el nombre de los archivos `<cod.prov>.<aaaammdd>.dat` (ej. `K234.20100910.dat`), con la siguiente información:
`nro factura,cod.producto,cantidad,importe`
Además se cuenta con el archivo de proveedores (`/tablas/provema.txt`)
`cod. proveedor, tipo producto, cuit, razón social`
También tenemos el archivo de productos (`/tablas/productos.txt`)
`cod. producto, tipo producto, descripción`
Se pide realizar un script que valide estos archivos, teniendo las siguientes consideraciones:

- El código de proveedor debe ser valido (debe existir en prevema.txt).
- El producto debe pertenecer al tipo de productos que trabaje ese proveedor, para eso utilizar los archivos de proveedores y productos.

Las facturas validadas deben guardase en el directorio `/ok` con el nombre de archivo `FACTURAS.aaaamm.txt` con los siguientes campos:
`Cod. Proveedor,nro factura,cod. producto,cantidad`
Las facturas erróneas deben guardarse con el mismo formato en `/error` con el nombre `facturas.error.aaaamm.err`

Shell- Validar archivos.

Se tienen n archivos en el directorio seteado en la variable de ambiente DIR_PROC, se pide validar que los archivos tengan todos sus registros con el formato correcto, para esto se deberá validar la cantidad de campos de cada registro, la misma se obtendrá el archivo REGISTROS.DAT que tiene el siguiente formato:
Extensión|separador| cantidad de campos (ej. txt|;|5)

Tomando esta información validar los archivos, solo se debe validar los archivos cuyas extensiones existen en el archivo REGISTROS.DAT.
Por cada archivo mostrar por salida standard nombre del archivo cantidad de registros totales, erróneos y buenos.

Se deben copiar en el directorio de la variable DIR_PROC_OK los registros que cumplen la cantidad de campos, los registros los erróneos copiarlos en DIR_PROC_ERR.

Shell- Pedidos.

Se reciben archivos de Pedidos de diferentes agencias (agencia.sec.dat ej.SOLDATI.003.dat), con el siguiente formato:
Nro pedido, fecha pedido, producto, cantidad, fecha_entrega
Ej. 12154,20111030,A45,152,20111105

Se pide realizar un script que realice las siguientes validaciones:
- La secuencia del archivo debe ser válida (tiene que ser mayor que la secuencia que le corresponde a la agencia Agencia.dat). El formato de AGENCIA.DAT es:

Agencia, descripción, dirección, última secuencia
Ej. SOLDATI,Agencia Soldati, Moreno 456,2

- La fecha pedido debe ser menor que la fecha de entrega

- El producto tiene que tener stock (verificar en Productos.DAT cantidad debe ser mayor a cantidad producto) . El formato de PRODUCTOS.DAT es:

Producto, descripción ,cantidad producto

Ej.A45, Producto A,300

Generar para los registros ok un archivo agregando la extensión ok y para los errores la extensión err (ej. SOLDATI.003.dat.ok y/o SOLDATI.003.dat.err)