**Práctica BJ10 ef**

**Programación del Shell. Scripts y procesos.**

**INDICACIONES SOBRE EL DESARROLLO**

En todos los puntos que sigue tendrá que explicar con detalle cómo logró llevar a cabo lo pedido, indicando trayectorias, comandos con su despliegue y / o acciones realizadas. En el caso de despliegue de comandos explique el significado de lo desplegado. Cuando haya preguntas específicas, no basta con aplicar y desplegar el resultado del comando, deberá usted contestarlas. Tendrá que abrir una terminal-ventana, en Ubuntu, y también puede ayudarse del File Manager.

Las respuestas a los ejercicios 2-10 (6.4 pts), deberán reportarse en un archivo en Comunidad. Las respuestas de los ejercicios 11-15 (3.6 pts) deberán mostrarse, al profesor, en ejecución.

**PREVIO AL INICIO**

* Copie el contenido del fólder *LabBJ10* de Sistemas Operativos (SO) de Comunidad. Este fólder contiene tanto el texto de la práctica como material de soporte.
* Después arranque la máquina virtual de Ubuntu/Linux y entre a la cuenta de “sisops”.

**ACTIVIDADES INICIALES**

Desde su directorio base (*home*), copie el archivo  ***udtar*** del URL [***ftp://ftp.itam.mx/pub/investigadores/rrios/SisOpe/Lab/PracBJ10/***](ftp://ftp.itam.mx/pub/investigadores/rrios/SisOpe/Lab/PracBJ10/) ***.*** Esto lo puede hacer, ya sea desde su browser, o desde su computadora usando el comando *wget*. Una vez en el directorio base aplique el comando ***tar –xvf udtar*** que descomprimirá el archivo ***udtar*** en su directorio base creando el directorio (folder) ***ud*** que contiene a su vez varios directorios y archivos.

**DESARROLLO**

1. Antes de empezar a resolver cualquier ejercicio deberá leer hasta la página de notas 4.13 el archivo *UbuntuProgShell.pptx* (en formato de notas), que contiene el material de soporte de la práctica.
2. Elabore el siguiente script (guión) ***ej1***. cuyo contenido será el siguiente

|  |
| --- |
| #!/bin/tcsh  echo “ps -l”  ps –l  echo “ls -al”  ls –al  echo “nos despedimos *Fulano* y *Mengano*” |

Después de elaborar el archivo ***ej1*** agréguele los permisos u+rx con alguno de los dos siguientes comandos:

|  |
| --- |
| ***prompt>*** chmod u+rx ej1 |

Ó con el script de la página de notas 4.6 (no olvide el permiso sobre “cx”).

|  |
| --- |
| ***prompt>*** cx ej1 |

Ejecute ***ej1*** y muestre lo desplegado \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ¿Por qué *ps –l* despliega a ***ej1*** como un proceso? \_\_\_\_\_\_\_\_\_\_\_\_\_ ¿Cuál proceso es, en realidad, ***ej1***? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

¿Cuál son? su PID \_\_\_\_ y su PPID \_\_\_\_ ¿Cuál es su proceso padre? \_\_\_\_\_\_\_\_\_\_ ¿Padre de *ps*? \_\_\_\_\_

ESTE EJERCICIO DEBERÁ MOSTRARLO AL PROFESOR, AHORA.

1. ¿Cuál es la función de la variable especial *$$*? Ver ejemplo de ***parpos*** en la pág. 4.8 y en el fólder ***ud***.

¿Qué valor arroja el siguiente comando y a que se refiere dicho valor?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| ***prompt>*** echo $$ |

Agregue el comando anterior a ***ej1***. Ejecute ***ej1*** y diga a que se refiere el valor arrojado por *echo $$*. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Ver ejemplo de ***perin*** en la pág. 4.11 y en el fólder ***ud***.

Este script asigna valores a *v1* y *v2* mediante lecturas datos de *stdin*. Nos enteramos de la lectura porque el script se detiene. Agregue los comandos necesarios para que los datos sean pedidos, al usuario, adecuadamente. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (Lea hasta la página 4.19 de *UbuntuProgShell.pptx*). La pág. 4.19 nos muestre un ejemplo de uso tanto del comando *test* como de la variable *$status*.

Según nuestra documentación la variable contiene el código numérico devuelto por la operación del último comando.

Del ejemplo mencionado, diga que significa el valor 0 y que significa el valor 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. Uno de estos dos valores significa que el último comando se realizó correctamente y el otro valor que hubo un error.

Usando los comandos *cp* y *ls*, provoque un acierto y un error en cada caso. Los comandos los aplicará al estilo de

|  |
| --- |
| ***prompt>*** cp x y**;** echo $status |
| ***prompt>*** ls z**;** echo $status |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (Lea hasta la página 4.24). La pág. 4.24 nos muestre un ejemplo de uso de la estructura *switch*.

Ejecútelo un primera vez tal y como está. Modifíquelo de tal manera que al ser ejecutado imprima “Mi Aniversario”, sólo para éste día (debe usar *case …*).\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Del documento de powerpoint sobre programación de scripts, páginas 4.25-4.27, vienen ejemplos del comando estructurado ***foreach***. ¿En que es diferente ***foreach*** , en cuanto a funcionamiento, de la tradicional instrucción repetitiva ***for*** del lenguaje C, C++ o Java? Los scripts ***planta, sol***  y ***árbol*** se encuentran en el fólder ***ud***. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. (Lea hasta la página 4.29). De la página 4.29, donde se muestra el script ***fact***, responder: ¿Cuál es el objetivo de la estructura *if-then-else*?, ¿Cómo se obtuvo el valor que se asignó a *cta*? \_\_\_\_\_\_\_\_\_\_
3. También, de la página 4.30, explique el programa del script ***parpos2***. \_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. En el script ***árbol***, página 4.27, cuál es el objetivo de la variable ***cabeza***. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hasta aquí deberá reportar esta práctica. Del ejercicio 11 en adelante, el profesor los calificará en la misma sesión de laboratorio.

1. a) Ejecute el script ***parpos2*** que se encuentra en su cuenta ***$HOME/ud***. Si a continuación Ud. ejecuta el comando ***set*** ¿porque en su desplegado no aparecen las variables shell ***i*** y ***j*** usadas dentro del script?

b) Ahora ejecute usando el comando *source*, ***source parpos2***. ¿Por qué ahora en su desplegado si aparecen las variables shell ***i*** y ***j***?

1. Elabore un script, que cada vez que se ejecute, pida dos datos (un número y un texto) vía lectura, que se agreguen como un nuevo renglón a un archivo. Si el archivo ya existía, este renglón de datos se debe agregar al final. Nombre del script \_\_\_\_\_\_\_\_
2. Elabore un script que le permita sumar ***n*** números.

Al ejecutar el script se irá pidiendo los ***n*** valores de a uno por uno.

1. Elabore un script que le permita generar los subdirectorios

**hola01, hola02, . . . , hola09, hola10, hola11, . . .**

desde el directorio donde se ejecute el script. Al elaborar este programa en shell piense que bien pudieran ser 10, 20 ó 30 subdirectorios. Siempre son dos dígitos en el nombre de los subdirectorios.

1. Elabore otro script para borrar todos los subdirectorios creados en el inciso anterior.