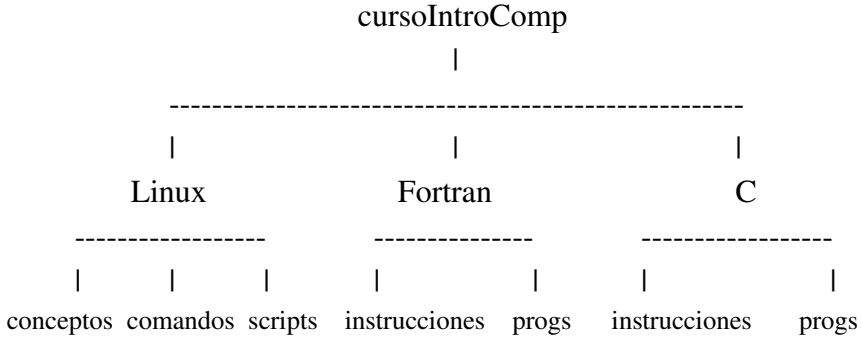
INTRODUCCION A LA COMPUTACION
PRACTICO 2 – LINUX

ARCHIVOS Y DIRECTORIOS

1- a) Logearse con su usuario. Verificar con **pwd**, que está actualmente parado en su home.

b) Sin salir de su directorio home, crear la siguiente estructura de directorios dentro de su home:



c) Ejecute **ls –lR** (R de Recursivo, l para listar info.) para visualizar todos los directorios.

d) Solo con un comando, cambiarse al directorio conceptos.

e) Con el comando **touch nombre_arch**, cree un archivo con nombre_arch = *fileSystem*. Con **ls**, confirme que fue creado. El comando touch usado de esta manera, crea un archivo sin contenido. Verifique esto último, visualizando el archivo.

f) Con touch usando más de un argumento crear dos archivos más también dentro de conceptos con los nombres: *kernel* y *distibuciones*.

g) Sin salir del directorio conceptos, crear tres archivos dentro del directorio comandos, llamados: *cmd_archs cmd_sistema* y *cmd_redes*. Verificar con **ls ../comandos** y en el directorio scripts crear el archivo *script.sh*.

h) Posicionarse en el directorio instrucciones bajo fortran, verificar.

i) Crear dos archivos de nombres: *instrucciones tiposDeDatos*.

j) Con el comando copy (**cp arch1 arch2 … archn dir_destino**), copiar estos dos archivos al directorio instrucciones bajo el directorio C. Verificar.

k) Cambiarse a ~/cursoIntocomp/C/instrucciones. Cambiar el nombre de los dos archivos a los mismos que tienen pero con un prefijo "c-", esto es *arch*, pasa a ser *c-arch*. (comando **mv**).

l) Ir al directorio progs bajo fortran. Crear el archivo prog1.f.
m) Desde el directorio fortran/progs, borrar el directorio progs bajo C.

n) Copiar el directorio fortran/progs con sus contenido entero al directorio C y renombrar el archivo C/progs/prog1.f a C/progs/prog1.c, sin salir de fortran/progs.

o) Desde el directorio fortran progs (donde quedó en la parte n), eliminar el directorio scripts y su contenido (script.sh).

p) Posicionarse en su home. En su home se encuentra el directorio cursoIntroComp. Compactarlo en un único archivo comprimido (con el compresor gzip) en un archivo de nombre *directorios.tar.gz*. Visualizar el contenido de este archivo nuevo (opción t del tar).

q) Ver el tamaño en disco de directorios.tar.gz. Descomprimir directorios.tar.gz con el comando descompresor **gunzip** . ¿Qúe tamaño tiene ahora?. (Cuando se crea un tar con compresión gzip, el comando tar llama implícitamente al comando **gzip**, gunzip descomprime los archivos .gz del gzip).

COMANDOS DEL SISTEMA, USUARIOS Y DE REDES

2 – a) Con el comando **df** visualizar la información del espacio en el disco duro.
¿Cuánto espacio libre hay ?. ¿Cuánto hay ocupado?.

b) Usar el comando **du** para saber cuanto es el espacio total que ocupa del directorio /home (junto con todos sus subdirectorios y archivos) y que el resultado sea en MB. (En el proceso algunos archivos o directorios figuran como inaccesibles, estos son que no tienen permiso de lectura para nuestro usuario).

c) Con el comando **free** visualizar cuanta memoria RAM tiene la máquina.

d) Con **finger** ver quienes están logeados al sistema (al menos usted debe estar).

e) Ejecute el comando **id** se puede ver información del usuario. Cada usuario tiene un grupo asociado (al que pueden pertenecer otros usuarios). El usuario y el grupo llevan asociados un número de identificación (en general solo nos manejamos con los nombres del usuario y el grupo). ¿Cúal es el número de usuario (uid) ?

f) Con el comando **ps** se pueden ver los procesos que están corriendo en memoria. Ejecutar dicho comando. El campo "tty" es la Terminal, pty/0 indica que es la teminal remota cero que se está ejecutando (¡La que están visualizando!). Con la opción "**u**" se muestra más información, incluida el identificador de usuario. La opción "**x**" muestra todos los procesos del usuario (deben ver por ejemplo el proceso *ssh* que permite que se comuniquen por ssh). Al ejecutar **ps aux** ven todos los procesos de todos los usuarios. Cada proceso lleva asociado un *número de proceso* "pid".

g) El comando **kill** (permite entre otras cosas) eliminar o terminar un proceso. Abra otra terminal sin cerrar la actual. Ejecute **ps aux** y fijarse en los procesos y en la columna "tty"; debe haber dos uno con pts/0 y otro con pts/1. El proceso con pts/1 es la segunda terminal, ejecute **kill -9 PID** siendo PID en número de dicho proceso. ¿Qué sucede?.

3 – a) Ingrese al sistema con su usuario. Parese en /home y ejecute **ls –l**. ¿Qué permisos tiene el usuario (usted) y el grupo y otros ?.

b) Cambie el permiso de lectura para "otros" a "no lectura" con el comando **chmod**.

Con ls ver los permisos actuales de su directorio personal.

c) Sin cerrar su sesión con su usuario, logearse con otra Terminal con el usuario: "user", password: "user" en introcomp-server. Intentar ver el contenido del directorio de su usuario (su usuario, no "user"); ¿Qué sucede?.

d) En la primera Terminal (su usuario personal en introcomp-server), volver a cambiar el permiso de lectura para otros a "lectura". Vuelva a intentar el paso anterior en la Terminal de "user", ahora debería poder visualizar el contenido.

e) En la Terminal de su usuario, entre a su home, o sea el directorio con el nombre de su usuario. Crear un archivo de texto cualquiera y cambiarle el permiso de lectura a "no lectura". Intentar visualizarlo con el usuario "user" en la otra Terminal, no debería poder leerlo.

f) Posicionarse en /home. Con el comando **chmod –R *directorioUsuario***, cambie el permiso a "lectura" para otros con sus usuario. Entrar en la Terminal de user a su directorio personal y lista los premisos de todos sus archivos y subdirectorios bajo él, ahora todos deben tener permiso de lectura para otros (la opción R es de recursivo y aplica el cambio a todo o que está debajo de un directorio).

4 – a) El siguiente ejercicio está relacionado con un sitio de la NASA:
http://www.cv.nrao.edu/~rfisher/Ephemerides/ephem_descr.html
Con el comando **ftp -i ssd.jpl.nasa.gov** logearse con el usuario "anonymous" y como password nulo. Cambiarse luego al directorio *pub/eph/export* (ejcutar cd y ls hasta llegar). Bajar con *get* el archivo *usrguide*. Luego desconectarse (exit) y visualizar el archivo ya en su home.

FILTROS DE TEXTO

5 -Comodines.

Los comandos que requieren nombes de archivos como argumento, aceptan la utilización de comodines para construir el argumento. Los comodines son:

- * : implica cualquier cadena de caracteres
- ? : implica un solo carácter (que puede ser cualquiera).

Crear tres archivos con **touch**: *archivo1-1.txt* y *archivo1-2.dat y archivo1,2.dat.*
Listar en el directorio donde están los archivos y ver los resultados:

```
ls archivo*
ls  a*
ls  *.txt
ls  *1-1*
ls  archivo1?2.dat
ls  archivo???.???
```

6 - a) Crear el archivo usuarios.txt con: **less /etc/passwd > usuarios.txt.**
Visualizarlo. Los distintos campos de los usuarios (un usuario por fila) están separados por ":". Extraer el nombre completo de los usuarios junto con sus descripción. (Aparecen algunos usuarios del sistema, los "convencionales" están al final).

b) Crear un archivo de texto llamado *procesos.txt* ejecutando el comando **ps aux > procesos.txt** (La salida de ps aux ls "redirecciona" a un archivo procesos.txt en vez de la salida estándar de consola).

Visualizar el archivo. Extraer (en la salida estándar) el encabezado (la primera linea). Extraer la última linea. Imprimir las 5 primeras y últimas lineas.

7) Expresiones regulares (*regex*), comando GREP.

Una expresión regular es una forma estándar sintética de expresar cualquier cadena de caracteres. Se usa como argumento de los comandos que esperan una cadena.
Los elementos constructivos son:

^cadena	: busca la cadena al comienzo de cada fila.
cadena$	: busca la cadena al final de cada fila.
[c1c2...cn]	: busca alguno de los caracteres dentro del corchete.
[c1..c2]	: busca alguno de los caracteres dentro del rango ordenado c1..c2
.	: iguala cualquier carácter único (similar al comodín ?)
c*	: busca cualquier número de caracteres después de c
*	: busca cero o más ocurrencias de cualquier carácter.

El comando grep es : **grep [opciones] 'regex' archivo.txt**

Pruebe los siguientes casos en el archivo usuarios.txt de la parte b):

```
grep -i 'user'  usuarios.txt
grep -i '^user' usuarios.txt
grep -i 'user$' usuaros.txt
grep -i   *       usuarios.txt
grep -i  'integra*' usuarios.txt (la descripción del usuario “cvoulgaris” no
                contiene la palabra “integrantes” y no figura en el resultado de grep).
grep -i  '[6-9]:[6-9]' usuarios.txt (Esto busca los dos números separados por “:”
            que son el nro. de usuario y de grupo.
```

8) El comando **awk**.

Awk es un lenguaje de programación de por si, que permite patrones (expresiones regulares) en uno o más archivos y condicionalmente modificarlos en base a acciones. La sintaxis de awk es la siguiente:

awk 'script' files

“Files” es una lista de archivos, sobre los que actúa awk. “Scripts”, es una o más parejas del tipo: */patrón/ {acciones}*. “Patrón” es una expresión regular y “acciones” uno o más comandos.

Para los ejemplos deben crear el archivo *planetas* tal como figura (mayúsculas/minúsculas y usen tabulación para que las columnas queden alineadas) o copiarlo a su home desde la carpeta home del usuario *user* en *introcomp-server*:

Planeta	Distancia-sol(Mkm)	Radio(Km)
Mercurio	57.9	4880
Venus	108.2	12103
tierra	149.6	12756
marte	227.9	6794
Jupiter	778.3	142984
Saturno	1429.4	120536
Urano	2871.0	51118
Neptuno	4504.0	49532
Pluton	5913.5	2274

Ejecutar los casos que siguen y analizarlos.

Ejemplos:

a)
```
user@hostname:~$ awk '{print ;}' planetas
Planeta         Distancia-sol(Mkm)      Radio(Km)
Mercurio        57.9                    4880
Venus           108.2                   12103
tierra          149.6                   12756
marte           227.9                   6794
Jupiter         778.3                   142984
Saturno         1429.4                  120536
Urano           2871.0                  51118
Neptuno         4504.0                  49532
Pluton          5913.5                  2274
```

En este caso el patrón está omiso por lo que se aplica el script (print) a cada línea de cada archivo que se le pasa a awk (solo uno en este caso: planetas). No hay diferencia con ejecutar *less planetas.*

b) awk separa en base a un delimitador (que es el espacio en blanco en este caso) cada línea en registros (columnas, similar a cut), cada registro se referencia por $1, $2,.. para el primer registro, el segundo, etc. Ejecutar (nuevamente sin patrón alguno):

```
awk '{print $1 $3}' planetas
```

c) En el caso b) los registros o campos los escribe uno a continuación del otro; para insertar un espacio podemos usar la “,” o insertar la cadena de caracteres “cadena” (con cadena = espacio en blanco, o sea “ “):

```
awk '{print $1,$3}' planetas
awk '{print $1 “ “ $3}' planetas
```

d) Quiero marcar todos los planetas que distan del sol más de 200 Mkm, con un * al final de la línea ($0 en print imprime la línea con el formato exacto que tenía):

```
awk'/[2-9][0-9][0-9]\.[0-9]/ {print $0,"*"}' planetas
```

(nota: el "\" que antecede al "." en el patrón se debe a que el punto "." es un carácter reservado para construir el patrón o expresión regular. O sea, si queremos reconocer un carácter en un archivo que justo coincide con alguno de los que usamos para construir la regex, debe estar antecedido por "\" en la regex. Si deseamos reconocer "[", como este se usa para marcar un rango ([2-9] por ejemplo) entonces debemos escribir "\[". Al "\" bajo estas circunstancias se e lama caracter de escape).

e) Operadores de comparación en awk
Se puede usar en awk, los siguientes operadores de comparación que devuelven verdadero (y por lo tanto se ejecuta la acción asociada) si las comparaciones siguientes se cumplen:

>	: mayor que
<	: menor que
>=	: mayor o igual que
<=	: menor o igual que
==	: igual a
!=	: distinto a
valor ~ /patrón/	: valor está dentro de los valores posibles de patrón
valor !~/patrón/	: valor no está dentro de los valores posibles de patrón

Probar los comandos y analizarlos:

```
awk '$2 > 200 {print $0,"*"}' planetas

awk '$2 > 200 {print $0,"*"}    $2 <= 200 {print $0}' planetas

awk '$2 ~/5[0-9][0-9][0-9]/ {print $0;}' planetas

awk '5438 ~/5[0-9][0-9][0-9]/ {print $0}' planetas

awk '$1 ~/no$/ {print "planetas que terminan en no"; print $0;}' planetas

awk '$1 !~/no$/ {print "planetas que no terminan en no"; print $0;}' planetas
```

```
awk '$1 ~/^m/ {print $0;}' planetas

awk '$1 ~/^M/ {print $0;}' planetas

awk '$1 ~/^[Mm]/ {print $0;}' planetas
```

SECUENCIA, REDIRECCIÓN Y PIPES DE COMANDOS

9) Ejecutar y analizar lo siguiente:

```
a) ls –l ; date ; id

b) ls –l > listaDirectorio ; cat listaDirectorio

c) ls –l | cat

d) ls –l | less

e) cd ; pwd | cat

f) cd ; pwd|cat ; touch file ; ls –l | grep file

g) cd ; pwd|cat

h) touch file ; ls –l | grep file | awk '{print
        "user="$3,"group="$4,"tamano="$5,"archivo="$9}'
```

i) Crear el archivo file1 con el contenido: hola mundo y file2 con el contenido: hola de nuevo. Luego ejecutar:

```
cat file1 file2 > file3
cat file2 >> file1
```

j) Crear el archivo “homedir” con el contenido: /home/miusuario (donde miusuario es su usuario). Ejecutar:

```
ls -l < homedir > homedirAgain ; cat homedirAgain

k) who | wc -l

l) file * | awk '$3~/text/ {print $0}' > textFiles.txt ;
cat textFiles.txt
```

m) Redirección de errores. Cuando se ejecuta un comando de forma errónea (por ejemplo ls de un archivo o dir que no existe), sale un mensaje de error en la consola. La salida por consola habitual se llama *salida estándar* (*stdout*); la salida por

consola de un error se llama *error estándar* (*stderr*). Se puede usar el símbolo de redirección “>&” para redirecciónar la *stdout* y el *stderr* en caso de que se de uno u otro a un archivo. Por ejemplo considere un archivo que existe: *file* y otro que no: *file87.* Ejecute los siguiente:

```
ls file87 > salidaLs
ls file87 >& salidaLs
ls file >& salidaLs
```