



Parcial 2

Temas

Administracion de procesos

Proceso en linux

Comandos

Directorio importante

Como se pueden ejecutar los procesos

Señales

Prioridades de procesos

Planificación de tareas

Informe sobre el desempeño del sistema.

Ejercicios

ShellScript

Variables

Grupo de ordenes

Estructuras Control

Comando

Ejercicios

Administracion de memoria

Mostrar informacion de memoria física

Mostrar Información de los memoria virtual:

Diretorios

Pasos para crear una rea de intercambio tipo file

Swaping

Entendimiento de Memoria/ usando top:

Ejercicios

Tp 3

Administracion de memoria Linux

Punto 1

Punto 2

Punto 3

Punto 4

Punto 5

ShellScript

[Punto 1](#)
[Punto 2](#)
[Punto 3](#)
[Punto4](#)
[Todo el script](#)
[Errores que tuve](#)

REPASR EL GREP xd

GREP

Tutorial del comando Grep - Cómo buscar un archivo en Linux y Unix con búsqueda recursiva
grep significa Globally Search For Regular Expression and Print out (Búsqueda global de expresiones regulares). Es una herramienta de línea de comando usada en sistemas Linux y Unix para buscar un patrón específico en un archivo o grupo de archivos. grep tiene muchas opciones que permiten realizar
 <https://www.freecodecamp.org/espanol/news/grep-command-tutorial-how-to-search-for-a-file-in-linux-and-unix-with-recursive-find/>



CUT

<https://geekland.eu/uso-del-comando-cut-en-linux-y-unix-con-ejemplos/>

Temas

- Administración de Proceso
- ShellScript
 - Condicionales
 - Est. Repetitivas
- Administración de memoria

Administración de procesos

Proceso en linux

- ▼ Que tipos de procesos tenemos en linux?
 - ▼ Interactivos
 - Iniciado y controlados por un shell, se pueden ejecutar en primer plano/foreground o segundo plano/background
 - ▼ En cola

No estan asociados a un terminal. Esperan en una cola ejecutarse secuencialmente

▼ Demonios

Lanzados al iniciar el sistema. Se ejecutan en background. Por ejemplo: el planificador de tareas cron

▼ Que estados tiene un proceso?

▼ Sleep (S)

Suspendido. Con espera interrumpible (esperando que se complete un evento). Puede ser interrumpido por alguna señal.

▼ Sleep (D)

Suspendido. Con espera No interrumpible (generalmente esperando E/S)

▼ Stopped (T)

Detenido) porque ha recibido una señal de detención

▼ Running (R)

▼ Zombie (Z)

El proceso ha terminado pero no está muerto porque su estructura task_struct permanece referenciada desde el vector task. Es un proceso que aún tiene dependencias.

▼ Dead (X)

Muerto (nunca debe ser visto).

Comandos

[Ver proceso en primer plano]

▼ **pstree** [-opciones] [pid|user]

opciones:

-p muestra los PID de los procesos

-a muestra argumentos de la linea de comando

-h Resalta el proceso actual y a los que encubre sobre él

El programa **pstree** facilita información sobre la finalización de una serie de procesos relacionados entre sí, esto es, todos los descendientes de un proceso particular. El programa deja claro desde un principio que proceso es el primario y cuáles son los secundarios.

```
mserna@mserna-VirtualBox:~$ pstree -hp
init(1) ┌─ ModemManager(726) ┌─ {ModemManager}(809)
         └─ {ModemManager}(811)
          └─ NetworkManager(813) ┌─ dhclient(881)
                           ┌─ dnsmasq(942)
                           ┌─ {NetworkManager}(815)
                           ┌─ {NetworkManager}(817)
                           └─ {NetworkManager}(818)
```

▼ ps [-opciones]

¿Qué es comando ps?

El **comando ps** te ayuda a realizar un seguimiento de tu memoria y uso de la CPU. Por ejemplo, si estás usando una máquina VPS y comienzas a notar que tu uso de memoria pasó de 60MB a 250MB por día, probablemente deberías observar de cerca lo que sucede detrás de escena

opciones:

- l Listado extendido de los procesos
- e Muestra todos los proceso del sistema
- x Lista porcesos que no estan contralods por ningun terminal
- h Muestra estado de los procesos
- u usuario Muestra porceso de un usuario en particular
- L Muestra en la columna LWP los hilos
- C Muestra hilos de un comando
- a obtiene todos los procesos que estén asociados a una terminal
- aux muestra información adicional sobre todos los procesos que se están ejecutando en el sistema y no solo los de una terminal.

```
mserna@mserna-VirtualBox:~$ ps -h
2756 pts/9    Ss      0:00 bash
3008 pts/9    R+      0:00 ps -h
mserna@mserna-VirtualBox:~$
```

Estado STAT y código:

- < prioridad mayor a la normal
- N prioridad menor que lo normal.
- L tiene páginas bloqueadas en memoria.
- s es un líder de sesión.
- L es multihilado.
- + está en el grupo de procesos en 1º plano.

```
mserna@mserna-VirtualBox:~$ ps -L -C rsyslogd
PID  LWP TTY          TIME CMD
571   571 ?        00:00:00 rsyslogd
571   621 ?        00:00:00 in:imuxsock
571   622 ?        00:00:00 in:imklog
571   623 ?        00:00:00 rs:main Q:Reg
mserna@mserna-VirtualBox:~$
```

rsyslogd es un demonio encargado de recolectar los mensajes de servicios que provienen de aplicaciones y el núcleo para luego distribuirlos en archivos de registros . Se muestra el PID del proceso y los hilos asociados LWP.

▼ top [-opciones]

El comando top

te permite ver las tareas del sistema que se ejecutan en tiempo real. Proporciona un buen resumen de tu sistema para verificar rápidamente si algo se destaca que pueda estar causando problemas con tu sitio web o servidor.

Opciones:

- p PID Muestra informacion sobre un proceso
- H Modo hilo
- d sec Intervalo de refresco en segundos

[Ver proceso en segundo plano]

▼ **jobs**

Directorio importante

▼ /proc

El directorio **/proc/** — también llamado el sistema de archivos **proc** — contiene una jerarquía de archivos especiales que representan el estado actual del kernel — permitiendo a las aplicaciones y usuarios mirar detenidamente en la vista del kernel del sistema.

Características

- Sistema de archivo virtual
- la mayoría tienen un tamaño de 0 bytes
- por cada proceso que se ejecute en el sistema operativo se genera un subdirectorio con el nro de PID

Comunicación entre procesos

Pipeline |



Como se pueden ejecutar los procesos

▼ Primer Plano

Lanzar un proceso como venimos trabajando	Ejemplo <code>ls -l</code>
Reanuda un trabajos suspendidos poniéndolos en foreground.	<code>fg nro tarea.</code>
Ver los procesos en 1º plano	<code>ps psfree top</code>

`sleep 500 &`
`jobs`
`fg %1`

Nota: El comando **fg**

(foreground o primer plano) traerá a primer plano un trabajo que está ejecutándose en segundo plano.
También se puede usar para reanudar en primer plano un trabajo que está suspendido o detenido.

▼ Segundo Plano

Lanzar un proceso en segundo plano.	Comando & [sleep 500 &] nohup Comando &
Reanuda un trabajos	<code>bg nro tarea</code>

`sleep 500 &`

suspendidos poniéndolos en background.	
Ver los procesos en 2º plano	jobs

El comando **jobs**

solo nos muestra los trabajo vinculados a la terminal desde donde se ejecuta **jobs**, es decir, no nos mostrará aquellos trabajos que hayan sido lanzados desde otras terminales aunque se estén ejecutando en ese momento.

Señales

▼ **kill** [opciones] [-señal PID]

opciones:

-l Lista el nombre de las señalesPrioridad de los procesos

```
clono@DESKTOP-LTP08DP MINGW64 ~
$ sleep 500 &
[1] 1351

clono@DESKTOP-LTP08DP MINGW64 ~
$ ps
   PID  PPID  PGID  WINPID  TTY      UID      STIME COMMAND
 1351  1323  1351  13372  pty0    197609 19:54:04 /usr/bin/sleep
 1355  1323  1355  5492   pty0    197609 19:54:07 /usr/bin/ps
 1322      1  1322   1192   ?      197609 19:53:47 /usr/bin/mintty
 1323  1322  1323   5528  pty0    197609 19:53:47 /usr/bin/bash

clono@DESKTOP-LTP08DP MINGW64 ~
$ kill 1351
[1]+  Terminated                  sleep 500

clono@DESKTOP-LTP08DP MINGW64 ~
$ jobs
```

Algunas Señales

- 2-SIGINT, , ctrol+c o ctrl+d termina el proceso
- 19-SIGSTOP, ctrol+z suspende el proceso
- 9-SIGKILL, mata/termina el proceso sin esperar el cierre de dependencias

Cómo cancelar un proceso con el comando kill en Linux

Las tareas en Linux se denominan procesos. Cada proceso tiene una ID de proceso única. Para finalizar un proceso en Linux podemos usar el comando kill. En este tutorial, te mostraremos cómo cancelar un proceso en Linux, para mejorar tus habilidades de

 <https://www.hostinger.com.ar/tutoriales/cancelar-proceso-comando-kill-linux>



Prioridades de procesos

▼ **nice** -NI comandos

Antes de ejecutarlo

- o Valor negativo aumenta la prioridad. --NI
- o Valor positivo disminuye la prioridad. -NI
- NI (resta NI a la prioridad base)

```
# nice -n-20 ./prueba.pl
```

Asignar prioridad de CPU a procesos en Linux con nice | rm-rf.es

El comando nice en Linux nos permite modificar la prioridad de un proceso frente al resto dentro del sistema. El kernel Linux es el encargado de planificar y asignar tiempo de CPU a cada uno de los procesos que corren en el sistema.

 <https://rm-rf.es/asignar-prioridad-de-cpu-a-procesos-en-linux-con-nice/#:-:text=El%20comando%20nice%20en%20Linux,que%20corre%20en%20el%20sistema>

▼ renice -/+NI PID

- Cambiar la prioridad minetras el proceso se esta ejecutadno
- o +NI suma a PRI por lo que el proceso tendrá menor prioridad.
- o -NI resta a PRI por lo que el proceso tendrá mayor prioridad.

PRI	valor de la prioridad	cercano a 1 (mayor prioridad)
		cercano a 100 (menor prioridad)

NI margen en el que se modifica la prioridad base.

Planificación de tareas

▼ at

now
8:30am jul 23, 2020

How to Use the Linux at Command {9 Examples}

Introduction The command is a Linux command-line utility used to schedule a job for later execution. The utility reads commands from standard input and groups them into an job, which executes only once. The alternative for is a cron job. However, while jobs execute

 <https://phoenixnap.com/kb/linux-at-command>



▼ atq

muestra los trabajos pendientes

▼ atrm n.^o

borra un trabajo pendiente

▼ crontab

En el archivo donde se programa la tarea se debe especificar:

minuto hora dia del mes mes año comando

- minutos (0-59)
 - horas (0-23)
 - días del mes (1-31)
 - mes del año (1-12)
 - días de la semana (0-6:domingo a sábado). O bien: (1-7: lunes a domingo)

cada del archivo tiene el siguiente formato:

minutos (0-59) horas (0-23) dia del mes (1-31) mes del año (1-12) dia (0-6 dom a sab)

ej: 30 20 30 * * tar cvf ~/respaldo.tar ~/Documentos

<https://www.ochobitshacenunbyte.com/2019/05/17/uso-del-comando-uptime-en-linux/>

Informe sobre el desempeño del sistema.

▼ uptime

```
juli@juli-VirtualBox:~$ uptime  
17:23:58 up 0 min, 1 user, load average: 3,22, 1,05, 0,37
```

- la cantidad de tiempo que el sistema lleva funcionando
 - la cantidad de usuarios que están actualmente en el sistema.
 - Visualiza el número medio de trabajos esperando a ejecutarse desde el último, los cinco y los diez últimos minutos. Si ese número se acerca a 0 significará que el sistema está bastante desocupado, mientras que un valor cercano a 1 indica que el sistema está bastante cargado.

Ejercicios

- ▼ 1. Cree un proceso en segundo plano.
 - ▼ 2. Vea que el proceso está en segundo plano.
 - ▼ 3. Traiga el proceso a primer plano
 - ▼ 4. Vea todos los procesos del sistema.

- ▼ 5. Cambie la prioridad a un proceso nuevo.
- ▼ 6. Cambie la prioridad a un proceso en ejecución.
- ▼ 7. Programar una tarea que realice un backup comprimido de su home y se ejecute todos los Viernes a las 23:30.
- ▼ 8. Crear una tarea que se ejecute a las 20 hs y registre en un archivo el informe de desempeño del sistema.

ShellScript

Variables

- ▼ Creacion
 - o Asignación directa `a=holo`
 - o Sustitución de comandos usuarios= ``who``
 - o Comando `read` read a
 - o Variable **Local**: es propia del shell actual. Reside en la zona de datos local de un proceso y será ignorada por los procesos hijos.
 - o Variable de **Entorno**: su valor es reconocido por todos los shell. Ej: PS1
Con el comando `export` exporta el valor de una variable para que pueda ser accesible en un subshell
- ▼ Especiales
 - ▼ `$?`
Almacena el estado de salida del último comando ejecutado
 - ▼ `$!`
Contiene el identificador del último proceso que se ejecuto en Background
 - ▼ `$#`
Almacena la cantidad de argumentos posicionales que fueron pasados al script
 - ▼ `$0`
Nombre del shellscript
 - ▼ `$1,$2,..`
Argumentos posicionales
 - ▼ `$*`
Lista los argumentos posicionales
 - ▼ `$$`
Almacena el identificador del proceso actual

▼ Entorno

▼ HOME=/home/login

Configura su directorio de usuario, es decir, la localización desde donde inicia la sesión. Sustituya *login* por su identificador de entrada al sistema, por ejemplo */home/mgarcia*.

▼ PATH=path

Contiene una lista de directorios que el shell examina al buscar archivos ejecutables.

▼ PS1=prompt

Contiene el identificador primario de shell, por defecto es el signo \$.

▼ SHELL=shell

Indica la ruta absoluta al programa intérprete de comandos. Por ejemplo, */bin/bash*.

▼ TERM=termtyp

Especifica el terminal por defecto. Suele ser xterm.

▼ PWD=root

Especifica la ruta por defecto del usuario.

Grupo de ordenes

orden && orden2 AND

orden1 || orden2 OR

COMANDOS SECUENCIALES

comando1; comando2; comando3

Estructuras Control

▼ Estructura condicional

```
if condicion
then
    comando
else
    comando
fi
```

▼ Estructura case

```
case $variable in
  patron1) comando;;
  patron2) comando;;
  *) comando;;
esac
```

▼ Estructura repetitiva

▼ while

```
#estructura while
n=0
while test $n -lt 5
do
  echo $n
  read a
  n=`expr $n + 1
done
```

while condición

do

comando1
comando2

done

Cuerpo de acciones

Condición

Es un comando

true
test -a archivo
test \$v1 -eq 3

▼ for

```
#uso de for
#!/bin/bash
for i in sol huerto puerta
do
  echo $i
  read a
done

#uso de for con parametros posicionales
#!/bin/bash
for i in `echo $*`
do
  echo $i
```

```

read a
done

#uso de for
#!/bin/bash
for i in `ls `
do
  if test -f $i
  then
    echo $i es un archivo regular
    read a
  elif [ -d $i ]
  then
    echo $i es un directorio
    read a
  fi
done

```

**for variable in lista_de
argumentos
do**

comando1 \$variable

comando2

done

```

#!/bin/bash
acum=0
xd=$(grep -Eo '[0-9]{1,2}'$ DB)

for i in $xd
do
  acum=`expr $i + $acum`
done

echo $acum

```

▼ until

```
#estructura until
n=0
```

```
until test $n -eq 5
do
    echo $n
    read a
    n=`expr $n + 1
done
```

until condición

do

comando1

comando1

done

Comando

▼ **test**

Permite comprobar el valor de cualquier expresión. Devuelve un 0, si la expresión es verdadera (true) y 1 si es falsa.

▼ Verificacion de archivos

- a Si existe el archivo
- b si es especial de bloque
- c si es especial de carácter
- d si es un directorio
- f si es regular
- r si tiene permiso de lectura
- w si tiene permiso de escritura
- x si tiene permiso de ejecucion

▼ Operadores Aritmeticos

n1 **-eq** n2 son iguales
n1 **-ne** n2 son distintos
n1 **-gt** n2 n1 mayor a n2

```
#punto 3
echo "La cantidad de participantes nacionales es de:"
#Grep -c me sirve para contar la cantidad de archivos que tienen "si"
#Variable por asignacion
countNac=$(grep 'si' DB -c)
echo $countNac
echo ""

echo "La cantidad de participantes extranjeros es de:"
countExt=$(grep 'no' DB -c)
echo $countExt
echo ""

if test $countNac -gt $countExt
then
    echo "En su mayoria los participantes del congreso fueron de origen naci>
else
    echo "En su mayoria los participantes del congreso fueron de origen ex>
fi
```

n1 **-ge** n2 n1 mayor o igual n2
n1 **-lt** n2 n1 menor a n2
n1 **-le** n2 n1 menor o igual a n2

▼ De comprobación de cadenas

test [opción cadena]
test [cadena operador cadena]

-z La cadena es nula
-n Existe la cadena

cadena1 = cadena2 cadena1 es igual a la cadena2
cadena1 != cadena2 cadena1 es distinta a la cadena2

|| (OR) devuelve true si cualquiera de las expresiones es verdadera.
&& (AND) devuelve true si ambas expresiones son verdaderas.
! (NOT) devuelve el valor opuesto de la expresión.

▼ Ejercicios

1.Crear el scrpt que reciba 2 parámetros posicionales (el archivo a enlazar y el nombre del enlace), sino recibe estos datos, deberá pedírselos al operador.

```
#Ejercicio 1

if test $# -eq 2 && test -f $1
then
    ln $1 $2
```

```

else
echo "ingrese el archivo a enlazar"
read archivo
echo "ingrese el nombre del enlace"
read enlace
ln $archivo $enlace
fi

```

2.Crear un script que permita agregar su nombre al final de un archivo ordinario.

```

#Ejercicio 2

echo "ingrese un archivo"
read archivo
if test -f $archivo
then
echo "ingrese su nombre"
read nombre
echo $nombre >> $archivo
else
echo "el archivo ingresado no existe"
fi

```

3.Cree el siguiente menú de opciones.

- a- Mostrar directorio
- b- listar procesos
- c- versión del sistema
- s- Salir

```

#Ejercicio 3

while true
do
echo "Ingrese una opcion
      a- Mostrar directorio
      b- Listar procesos
      c- Versión del sistema
      s- Salir"
read i
case $i in
a|A) echo "Los directorios son:"
      ls -l | grep '^d';
b|B) echo "Mostramos los procesos:"
      ps -aux;;
c|C) echo "Versión del sistema operativo" uname -v;;
s|S) break;;
esac
done

```

4.Crear un script que reciba como parámetro un nombre. Verificar si es archivo ordinario o directorio e informar.

```

#Ejercicio 4

if test -f $1
then
echo "El archivo es un archivo regular"
else

```

```
if test -d $1
then
    echo "El archivo es un directorio"
else
    echo "No ingreso un archivo ni un directorio"
fi
fi
```

Esto estan hechos por las profes

▼ exit

Se utiliza para salir de un bucle y se ejecuta la orden que sigue inmediatamente a éste.

▼ break[n]

Se utiliza para salir de un bucle y se ejecuta la orden que sigue inmediatamente a éste. La opción n indica el número de niveles o bucles de los que hay que salir.

```
#EJEMPLO
while true
do
    echo nivel 1
    while true
    do
        echo nivel 2
        while true
        do
            echo nivel 3
            while true
            do
                echo nivel 4
                read a
                break # sale de un bucle
            # exit      #sale del programa
            done
            echo regrese a nivel 3
            read a
            break 2  #sale de 2 bucles
        done
        echo regrese a nivel 2
        read a
    done
    echo regrese a nivel 1
    read a
done
```

▼ continue

Salta a la siguiente iteración, sin salir del bucle. Esto permite evitar la ejecución del cuerpo de acciones para ciertos valores de la variable que controla el bucle

▼ EJERCICIOS

- 1.Crear el script que acepte números y los guarde en un archivo, hasta que el numero ingresado sea mayor a

```
#uso de while mientras el numero sea menor a 20
clear
echo -n "Ingrese un numero: "
read n
while test $n -le 20
do
    echo $n >> ~/num-ejer1
    echo -n "Ingrese un numero: "
    read n
done
```

2.Crear un script que permita trabajar con un directorio en particular. Deberá mostrar nombre y cantidad de líneas de cada archivo ordinario. Y listar en formato extendido el contenido de los subdirectorios. Almacene toda la información de salida en un archivo.

```
#uso de for contar lineas de archivos
clear
echo -n "Ingrese nombre del directorio: "
read v1
if test -d $v1
then
    for i in `ls $v1`
    do
        if test -f $v1/$i
        then
            wc -l $v1/$i >> ~/lineas-ejer2
        elif test -d $v1/$i
        then
            echo subdirectorio $i
            ls -l $v1/$i
        fi
    done
fi
```

3.Crear un script que permita sumar la cantidad de números indicada por el usuario. Mostrar el resultado parcial de la suma luego de cada iteración.

```
#uso de until sumar numeros
clear
echo -n "Indique la cantidad de numeros a sumar: "
read n      #cantidad de numeros a sumar
c=0        #lleva la cuenta de cuantos numeros hemos sumado ya
suma=0
until test $c -eq $n
do
    clear
    echo -n "Ingrese un numero a sumar: "
    read num
    suma=`expr $suma + $num`
    c=`expr $c + 1`
    echo "Resultado de la suma : $suma      cantidad: $c"
    read a
done
```

1) Guardar en el archivo foto1 la salida del comando ps -le

2.Crear el archivo foto2 con el contenido de foto1 pero sin cabecera.

3.El administrador necesita conocer las prioridades de procesos más utilizadas. Mostrar el contenido del archivo foto2 ordenado por la columna prioridad.

4.En qué valores fueron modificadas esas prioridades? Mostrar el contenido del archivo foto2 ordenado por la columna NI.

5.Mostrar sólo las líneas del archivo foto2 que hagan referencia al proceso getty. Indicar también cuantos procesos getty están presentes en el sistema.

6.Crear un menú que solicite un número y devuelva el día de la semana para ese número siendo el 0, 7 y todo número mayor a 7 igual a Domingo.

7.Cree un script que acepte 2 cadenas e informe si son o no iguales.

HECHO EN CLASE

```
for i in `sort -n numeros`  
do  
    if [ $i -gt 6 ] && [ $i -le 10 ]  
        then  
            continue  
    fi  
    echo $i  
done
```

```
for i in `sort -n numeros`  
do  
    if [ $i -eq 9 ]  
        then  
            continue 3  
    fi  
    echo $i  
done
```

8.Diseñar un menú con las siguientes opciones:

C – Buscar una cadena de caracteres en un archivo

D - Guardar i-nodos

S - salir

Consideraciones:

En la opción C, pedir al operador que ingrese la cadena a buscar y el nombre del archivo donde se realizará la búsqueda. Verificar si el archivo existe y no es de tipo directorio, en cuyo caso mostrar un mensaje de error. En la opción D, pedir al operador que ingrese el nombre del directorio a listar y el nombre del archivo a crear, no olvide verificar que el directorio ingresado exista y sea de tipo directorio. En el archivo deberá guardar el nombre y número de i-nodo de los archivos del directorio. Realice las verificaciones necesarias

TENGO EJERCICIOS HECHO EN CLASES DESP PASARLOS

Ejercicios

- ▼ 1.Desarrolle un script que acepte las edades de 2 personas y muestre el promedio. Los datos deben ingresarse como parámetro posicional, en caso contrario pedirlos por pantalla.
- ▼ 2.Crear un script que reciba como parámetro un directorio (usar parámetros posicionales) que contenga archivos. Calcular y mostrar la cantidad de archivos regulares existentes y mostrar el nombre de esos archivos. Validar que la cantidad ingresada de parámetros sea una.

Administracion de memoria

Mostrar informacion de memoria física

- ▼ **free** [-opciones] [-s demora] [-t]

Muestra la cantidad total de **memoria física** y de intercambio presente en el sistema, así como la memoria compartida y los buffers utilizados por el kernel.

Opciones:

- b Bytes muestra los valores en bytes.
- k Kilo muestra los valores en KBytes.
- m Mega muestra los valores en MBytes.
- g Giga muestra los valores en Gbytes.
- t Tera muestra los valores en TBytes.
- h Human muestra los valores con un máximo de tres dígitos y le agrega la unidad en el que esta expresado. B (bytes), K (kilos), M (megas), G (gigas), T (teras).
- s Seconds muestra la salida del comando cada n segundos.
- t Total muestra valores totales de cada columna.

Here's what each column mean:

- **total** - This number represents the total amount of memory that can be used by the applications.

- **used** - Used memory. It is calculated as: `used = total - free - buffers - cache`
- **free** - Free / Unused memory.
- **shared** - This column can be ignored as it has no meaning. It is here only for backward compatibility.
- **buff/cache** - The combined memory used by the kernel buffers and page cache and slabs. This memory can be reclaimed at any time if needed by the applications. If you want buffers and cache to be displayed in two separate columns, use the `w` option.
- **available** - An estimate of the amount of memory that is available for starting new applications, without swapping.

<https://linuxize.com/post/free-command-in-linux/>

Mostrar Información de los memoria **virtual**:

▼ **vmstat** [periodicidad] [cantidad de informes]

muestra informes estadísticos sobre el uso de la **memoria virtual**. Da información sobre procesos, memoria, paginación, E/S y actividades de la CPU. El primer informe solicitado, da valores medios desde el último arranque.

Opciones:

intervalo tiempo en segundos que trascurre entre refrescos. Si no se especifica muestra sólo un informe.

número cantidad de informes a emitir, siempre y cuando esté definido el intervalo.

-a Muestra la cantidad de memoria activa e inactiva.

-s Muestra una tabla con estadísticas de varios contadores de eventos.

-d Informe de estadísticas de disco

-D Resumen estadístico sobre la actividad del disco.

-p Dispositivo estadísticas detalladas sobre la partición.

-Scaracter Muestra la información según el carácter indicado, en las siguientes unidades:

1000 (k), 1024 (K), 1000000 (m), 1048576 (M).

el ultimo comando para en la unidad que quieras seria -SM

Directorio

▼ **cat /proc/swaps**

Este archivo muestra las áreas de swap activas en el sistema.

```
mserna@mserna-VirtualBox:~$ cat /proc/swaps
Filename                Type      Size   Used  Priority
/dev/sda5                partition 1490940  0       -1
mserna@mserna-VirtualBox:~$
```

Es una partición del disco /dev/sda creada por defecto en el momento de la instalación.

▼ cat /etc/fstab

se suelen añadir las particiones swap que se activan al inicio del sistema. Este archivo almacena información de cómo serán montadas e integradas al sistema las particiones de disco y sistemas de archivos.

/etc/fstab Este archivo almacena información de cómo serán montadas e integradas al sistema las particiones de disco y sistemas de archivos. Por lo que, una vez creada el área de intercambio, se deberá agregar una línea a este archivo como sigue:

partition/file swap swap defaults o o

Para que pueda utilizarse automáticamente desde el siguiente arranque del sistema. (este tema se trata en el capítulo 10 del libro de Linux)

▼ cat /proc/meminfo

brinda información sobre la memoria.

Pasos para crear una rea de intercambio tipo file

▼ 1- Crear el dispositivo/archivo de intercambio (comando dd)

dd if=entrada of=salida bs=n count=bloques

convierte y copia un archivo.

if=fichero es de donde se tomarán los datos, por ejemplo: /dev/zero (todos ceros), /dev/random (de forma aleatoria), /dev/sdb1 (una partición, para ser clonada).

of=salida nombre del archivo de salida.

bs=n cantidad de bytes que se lee y escribe n bytes de una vez, puede utilizarse K,M,G (Kbyte, Mbytes y Gbytes respectivamente).

count=bloques cantidad de bloques a copiar de tamaño determinados por bs, del archivo de entrada.

conv=CONVERSION convierte el archivo según se haya especificado en el o los argumentos.

```
mserna@mserna-VirtualBox:~$ sudo dd if=/dev/zero of=file-swap bs=2048 count=1024
[sudo] password for mserna:
1024+0 registros leídos
1024+0 registros escritos
2097152 bytes (2,1 MB) copiados, 0,0187364 s, 112 MB/s
```

▼ 2-Preparar el dispositivo como swap (comando mkswap)

mkswap [opciones] dispositivo [tamaño]

prepara un dispositivo o archivo como área de intercambio. El del Sistema de archivos se puede especificar en bloques. El argumento dispositivo puede ser un archivo o una partición.

```
2097152 bytes (2,1 MB) copiados, 0,0107501 s, 112 MB/s
mserna@mserna-VirtualBox:~$ sudo mkswap file-swap 2048
Configurando la versión swapspace 1, tamaño = 2044 KiB
sin etiqueta, UUID=1e9f0efe-2fa8-4768-bcd6-806ffd3569c8
```

▼ 3- Escribir el archivo en el disco (comando sync)

sync guarda el contenido del caché de disco dentro del disco físico. De esta forma se fuerza la escritura en disco de la información que ha cambiado.

```
mserna@mserna-VirtualBox:~$ sync
```

▼ 4- Activar el área de intercambio para poder utilizarla (comando swapon)

swapoff/on des/habilita dispositivos o archivos para el paginado e intercambio.

Sintaxis:

swapon nombreArchivo habilita

swapoff nombreArchivo deshabilita

swapon -s muestra información de las áreas de intercambio activadas

Opciones:

-p Especifica la prioridad del dispositivo de intercambio. Cuando la prioridad no está definida, el valor predeterminado es -1.

-s, --summary Muestra el resumen de uso del swap por dispositivo.

```
mserna@mserna-VirtualBox:~$ sudo swapon file-swap
mserna@mserna-VirtualBox:~$ sudo swapon -s
Filename                                Type      Size    Used   Priority
/dev/sda5                               partition 1490940  0       -1
/home/mserna/file-swap                  file        2044   0       -2
```

Swaping

▼ Que es el swaping?

Si un sistema operativo funciona lento, una de las causas más comunes de ello es que la memoria **física** (o memoria RAM) disponible se esté agotando. Es decir, los procesos están consumiendo más memoria de la disponible realmente. Esto conlleva a que el sistema empiece a utilizar el disco duro para almacenar aquellos datos (o páginas) que no entran en la memoria física. Este proceso se denomina **swapping**. El problema de esta **técnica** es que el acceso al disco duro es muy lento respecto al acceso a la memoria física.

Entendimiento de Memoria/ usando top:

- ▼ Como funciona la memoria física y la memoria swap? y como la vemos en comando como top,free?

La memoria física, a la cual, la podemos dividir en dos:



Sin embargo, cuando uno utiliza las herramientas de monitoreo de memoria en Linux, como ser: **top**, **free** y **vmstat**, ellas brindan un poco más de información que a veces suele ser confuso entenderlas. Para comprenderlas, vamos a dividir a la memoria física en lo siguiente:



buff/cache son aquellos datos (o páginas) que pueden estar en la swap o que podrían estar próximamente. Se la considera **libre** pues puede asignarse a otro proceso en caso que se necesite

A continuación, se presenta una captura parcial de la salida del programa top

```
top - 16:05:21 up 4:28, 1 user, load average: 0,73, 0,49, 0,38
Tareas: 287 total, 1 ejecutar, 286 hibernar, 0 detener, 0 zombie
%Cpu(s): 0,6 usuario, 0,7 sist, 0,0 adecuado, 98,6 inact, 0,2 en espera, 0,0 hardw int, 0,0 s
KiB Mem : 8101348 total, 763132 free, 4637492 used, 2700724 buff/cache
KiB Swap: 8312828 total, 8310276 free, 2552 used. 2474288 avail Mem
```

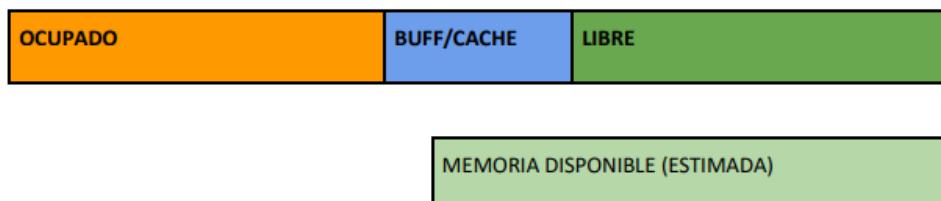
Si analizamos la cuarta línea:

KiB Mem: 8101348 total, 763132 free, 4637492 used, 2700724 buff/cache.

Luego, en la quinta línea vienen los valores de la memoria swap, es decir, aquella porción de la memoria que está en el disco duro:

KiB Swap: 8312828 total, 8310276 free, 2552 user. 2474288 avail Mem

Al final de esta línease encuentra la cantidad de memoria disponible: avail Mem. Este valor es una estimación de cuánta memoria disponible hay para que empiece un nuevo proceso sin swapping. A continuación un gráfico para entender el concepto



Uno podría pensar que la memoria disponible siempre debería ser la suma entre la memoria libre más la memoria buff/cache, sin embargo, esto no es así, de hecho el kernel utiliza un algoritmo bastante complejo que tiene en cuenta varios factores para lograr una mejor estimación.

Ejercicios

- ▼ 1.Crear un menú de opciones:
 - a- Muestre infor. del uso de la memoria
 - b- Muestre infor. de memoria (incluya memoria virtual)
 - c – Muestre áreas de intercambio activas
 - s- Salir
- a- Pida como dato de entrada la unidad en que se mostrará las cantidades de memoria, incluya totales.
- b- Pida como dato de entrada intervalo de refresco y cantidad de informes.
Considere que el usuario puede ingresar una opción incorrecta.
- ▼ 2.Generar un informe estadístico de la memoria virtual que se ejecute cada 2 segundos y se genere un total de 6 informes.
- ▼ 3.Mostrar informacion de memoria fisica y de intercambio en Mbytes. Mostrar la linea de totales y un total de 4 infomres
- ▼ 4.Qué archivo muestra información de memoria ycuál información de las áreas de intercambio?
- ▼ 5. Generar un área de intercambio de 2 Mega.

Tp 3

Administracion de memoria Linux

Punto 1

- Memoria física (expresar los resultados en MiB).

 - a. Cantidad total.
 - b. Cantidad usada.

- c. Cantidad libre.
- d. Cantidad en buff/cache.
- e. Cantidad disponible.

```
emily@emily:~$ free -mt
              total        used      free      shared  buff/cache   available
Mem:       3931         558      2713          14       659       3140
Swap:      974           0      974
Total:    4906         558      3688
```

a. 3931



- **total** - This number represents the total amount of memory that can be used by the applications.

b. 558



- **used** - Used memory. It is calculated as: $\text{used} = \text{total} - \text{free} - \text{buffers} - \text{cache}$

c. 2713



- **free** - Free / Unused memory.

d. 659



buff/cache

- The combined memory used by the kernel buffers and page cache and slabs. This memory can be reclaimed at any time if needed by the applications. If you want buffers and cache to be displayed in two separate columns, use the `-w` option

e. 3140



- **available** - An estimate of the amount of memory that is available for starting new applications, without swapping.

<https://linuxize.com/post/free-command-in-linux/>

- **shared** - This column can be ignored as it has no meaning. It is here only for backward compatibility.

Punto 2

Memoria intercambiada a disco (expresar los resultados en MiB).

- Cantidad total.
- Cantidad usada.
- Cantidad libre.

 Memoria intercambiada a disco = **Swapping** es el **proceso por el cual una página de memoria es copiada a un espacio en el disco duro, llamado memoria swap, para liberar esa página en memoria**. La **combinación** de los tamaños de memoria física y la memoria swap recibe el nombre de memoria virtual disponible

```
emily@emily:~$ vmstat 1 1 -SM
procs -----memory----- swap-----io----- system-----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 0 2693 31 628 0 0 233 55 514 137 2 0 96 2 0
```

- NI IDEA CUAL ES

- 0



- **swpd** Cantidad de memoria virtual utilizada.

- 2693



- **free** Cantidad de memoria inactiva (**free**).

Comando vmstat

En sistemas operativos Debian y similares a Unix (Linux), el comando vmstat reporta estadísticas del sistema, permitiendo obtener un detalle general de los procesos, E/S, uso de memoria/swap, actividad de CPU y estados del sistema. Esto puede ser utilizado para ayudar a identificar cuellos de botella en el rendimiento.

 <https://blog.carrerinux.com.ar/2019/08/comando-vmstat/>

Punto 3

Mencionar el o los archivos de los cuales proviene la información que se observa al ejecutar los comandos vmstat y free. Identificar uno de los valores que se visualizan tanto al ejecutar el comando vmstat como al ejecutar el comando free y expresar su significado.

PARA EL VMSTAT NI NI idea CREO QUE ES EL MISMO DIRECTORIO

Para free viene del directorio /proc/meminfo

```
emily@emily:~$ cat /proc/meminfo
MemTotal:        4026148 kB
MemFree:         2756684 kB
MemAvailable:   3194276 kB
Buffers:          32480 kB
Cached:          602420 kB
SwapCached:       0 kB
Active:           237640 kB
Inactive:         890324 kB
Active(anon):    1340 kB
Inactive(anon):  506532 kB
Active(file):    236300 kB
Inactive(file):  383792 kB
Unevictable:      0 kB
Mlocked:          0 kB
SwapTotal:        998396 kB
SwapFree:         998396 kB
Dirty:            404 kB
Writeback:         0 kB
AnonPages:        489016 kB
Mapped:           242944 kB
Shmem:            14808 kB
KReclaimable:    41132 kB
Slab:             79756 kB

emily@emily:~$ free -kt
              total        used        free      shared  buff/cache   available
Mem:      4026148          593500    2756568          14808    676080     3194208
Swap:     998396           0        998396
Total:   5024544          593500    3754964
```

Punto 4

Averiguar con qué comando se puede conocer el tamaño de página.

¿Qué tamaño tiene cada página de su sistema operativo Linux?



La paginación es una estrategia de organización de la memoria física que consiste en dividir la memoria en porciones de igual tamaño. A dichas porciones se las conoce como páginas físicas o marcos. La división de la memoria en páginas facilita la gestión de la memoria física

```
emily@emily:~$ getconf PAGESIZE  
4096
```

```
getconf PAGESIZE
```

Punto 5

Si su máquina tiene 800 MiB de memoria disponible y 300 MiB de memoria libre, de repente necesita ejecutar un programa que ya se sabe que en promedio ocupa 600 MiB al iniciar. ¿El sistema operativo empezará a swappear?

NOOOOOOOOOOOOOOOOO ashe



- **available** - An estimate of the amount of memory that is available for starting new applications, without swapping.

ShellScript

ANTES CREAR EL ARCHIVO DB

```
Jaime,Fernanda-50-si-3  
Guzmán,Eduardo-32-si-4  
López,Santiago-29-no-1  
Marín,Gonzalo-45-si-3  
Romo,Julieta-47-si-5  
Tulián,Lucas-42-no-2  
Bonfigli,Jorge-24-si-1  
Arrufat,Érica-30-si-2  
Cortés,Jacinto-32-no-2  
Figueroa,Flor-35-si-1
```

Punto 1

Cantidad total de exposiciones que suman los 10 participantes incluidos en el archivo.

creo que el tema del grep, tambien se puede solucionar con un cut

USO de expr para SUMAR, porque si queda solo + concatena

```
#!/bin/bash
acum=0
#-o (--only-matching) - imprimir solo el patrón coincidente
#-E
#[0-9]{1,2}' numeros [0-9] decimales, {1,2} Para mostrar de dos digitos
# $ para mostrar el final de las lineas
xd=$(grep -E '[0-9]{1,2}'$ DB)

#Cada palabra esta separada por un -. Por lo tanto tendremos que fijar el - como
#delimitador. Para fijar el espacio como delimitador lo haré con la opción -d '-'.

#A continuación hay que definir la palabra que queremos mostrar. si queremos mostrar
#la cantidad de exposicion -f4. La opción -f4 hace que se muestre
#los numeros que hay entre el cuarto y quinto (que no hay) delimitador.
probando=$(cat DB | cut -d '-' -f4)

for i in $xd
do
    acum=`expr $i + $acum`
done

echo $acum
```

Punto 2

¿Hubo algún participante que realizó más de 3 exposiciones en el congreso?

```
#punto 2
cont=0
for i in $numExp
do
    if test $i -gt 3
    then
        #No olvidarse del $ para las variables
        #hice un contador para ver las cantidad de personas que hizo mas de 3 presentaciones
        cont=`expr $cont + 1`
    fi
done
#Si el contador es mayor a cero es porque si hubo participantes con > 3 participaciones
if test $cont -gt 0
then
    echo "Hubo participantes que realizar mas de 3 exposiciones en el congr"
```

```
else
    echo "No hubo participantes que realizaron mas de 3 exposiciones"
fi
```

Punto 3

¿Los participantes fueron en su mayoría de origen nacional o no?

Este tambien lo resolví usando el if [] que remplaza al test, pero es mas facil el test, porque con corchetes deben respetarse los espacios y es mas facil cofundirce

-gt

```
#punto 3
echo "La cantidad de participantes nacionales es de:"
#Grep -c me sirve para contar la cantidad de archivos que tienen "si"
#Variable por asignacion
countNac=$(grep 'si' DB -c);
echo $countNac
echo ""
echo "La cantidad de participantes extranjeros es de:"
countExt=$(grep 'no' DB -c)
echo $countExt
echo ""

#num1 -gt num2 Gt es que el num1 es mayor al 2
if test $countNac -gt $countExt
then
    echo "En su mayoria los participantes del congreso fueron de origen nacionales"
else
    echo "En su mayoria los participantes del congreso fueron de origen extranjeros"
fi
```

Punto4

Cantidad de participantes mayores de 40 años.

En este caso si usaste el cut al principio, directamente usa la misma variable

```
#punto 4
#para este si o si necesito el cut para sacar por columna
base=$(cat DB | cut -d '-' -f2)
cant=0
for i in $base
do
    if test $i -gt 40
    then
        cant=`expr $cant + 1`
    fi
done

echo "La cantidad de personas mayores a 40 es de $cant"
```

Todo el script

```
#!/bin/bash
clear
#punto 1
acum=0
numExp=$(grep -Eo '[0-9]{1,2}' $ DB)

for i in $numExp
do
    acum=`expr $i + $acum`
done

cantParticipantes=$(grep '' DB -c)
echo "Los $cantParticipantes totalizaron $acum exposiciones en el congreso"
echo ""

#punto 2
cont=0
for i in $numExp
do
    if test $i -gt 3
    then
        cont=`expr $cont + 1`
    fi
done

if test $cont -gt 0
then
    echo "Hubo participantes que realizar mas de 3 exposiciones en el congreso"
else
    echo "No hubo participantes que realizaron mas de 3 exposiciones"
fi
echo ""
#punto 3
echo "La cantidad de participantes nacionales es de:"
#Grep -c me sirve para contar la cantidad de archivos que tienen "si"
#Variable por asignacion
countNac=$(grep 'si' DB -c);
echo $countNac
echo "La cantidad de participantes extranjeros es de:"
countExt=$(grep 'no' DB -c)
echo $countExt
echo ""

if test $countNac -gt $countExt
then
    echo "En su mayoria los participantes del congreso fueron de origen nacional"
else
    echo "En su mayoria los participantes del congreso fueron de origen extranjero"
fi

#El es lo mismo test que [] Tener en cuenta que [ variab;e ] debe tener un espacio
#if [ $countNac -gt $countExt ]
#then
#    echo "mayor $countNac"
#else
#    echo "menor $countExt"
#fi

#punto 4
```

```
#para este si o si necesito el cut para sacar por columna
base=$(cat DB | cut -d '-' -f2)
cant=0
for i in $base
do
    if test $i -gt 40
    then
        cant=`expr $cant + 1`
    fi
done

echo "La cantidad de personas mayores a 40 es de $cant"
```

Errores que tuve

expr: non-integer argument



cant=`expr variable + 1`



cant=`expr \$variable + 1`

```
num=`expr $num1 + $num2` # para que funcione, mantén los espacios en blanco
```

obtener cant de lineas

grep "" DB -c

Para sacar numeros con el grep

You can use `grep -E` to access the extended regular expression syntax(Same as **egrep**)

I have created a testfile with below contents:

```
>cat testfile
this is some text
with some random lines

again some text
ok now going for numbers (:32)
ok now going for numbers (:12)
ok now going for numbers (:132)
ok now going for numbers (:1324)
```

Now to grep the numbers alone from the text you can use

```
>grep -Eo '[0-9]{1,4}' testfile
32
12
132
1324
```

will be output.

Here "`-o`" is used to only output the matching segment of the line, rather than the full contents of the line.

The squiggly brackets (e.g. { and }) indicate the number of instances of the match. `{1,4}` requires that the previous character or character class must occur at least once, but no more than four times.

Hope this helps

Tener en cuenta siempre el tema de las variables como

for i in \$listado

no olvidarse del done

done

no olvidarese del fi

if test \$variable -gt \$variable2

then

else

fi

Nose si funcionaran bien el
echo "\$variable"
en su version de debian