**Listado de Comandos y Funciones**

a. Título: StartX

b. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.

Centralizar el inicio de procesos desde una función general, la cual verifica si no está actualmente en ejecución y en caso que no lo esté, lo lanza en background.

c. Archivos de Input, Intermedios y de Output

No posee.

d. Parámetros y Opciones

Recibe 2 parámetros, el primero, indica el comando que lo invocó y su única finalidad es que pueda ser registrado en el log. El segundo, es el comando cuya invocación se debe realizar.

e. Invocación Manual y/o automática con ejemplos:

Ejemplo 1:

./StartX.sh "Console" "./InstalarX.sh"

Ejecutará el script InstalarX.sh y dejará grabado en el log que el comando invocante es "Console"

Ejemplo 2:

./StartX.sh "InicioX" "./DetectaX.sh"

Ejecutará el script DetectaX.sh y dejará grabado en el log que el comando invocante es "InicioX"

f. Hipótesis y Aclaraciones especificas del comando

- Los comandos se ejecutan siempre en background, a menos que el comando a ejecutar sea DetectaX.

g. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)

#! /bin/bash

# Parametro 1: comando invocante

# Parametro 2: comando a ejecutar

function StartX {

source GlogX.sh;

if [ $# -eq 3 ]

then

return 1

fi

comando=$1

comandoAEjecutar=$2

nombreComando=$( echo "$comandoAEjecutar" | cut -f1 -d" ")

var=`ps -fea | grep -v "grep" | grep -v "StartX" | grep "$nombreComando" | wc -l`

if [ "$var" -ne 0 ]

then

GlogX "$comando.sh" "SE" "StartX.sh encuentra un proceso $nombreComando en ejecución" "$comando"

return 1

fi

GlogX "$comando.sh" "I" "StartX.sh invoca a $nombreComando" "$comando"

if [ $nombreComando == "DetectaX.sh" ]; then

$comandoAEjecutar &

else

$comandoAEjecutar

fi

return $!

}

StartX "$1" "$2"

a. Título: StopX

b. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.

Centralizar la finalización de procesos desde una función general, la cual verifica si no está actualmente en ejecución y en caso que no lo esté, lo lanza en background.

c. Archivos de Input, Intermedios y de Output

No posee.

d. Parámetros y Opciones

Recibe un único parámetro: el id del proceso a finalizar.

e. Invocación Manual y/o automática con ejemplos

./StopX 2368

Finaliza el proceso de id 2368, si es que existe.

f. Hipótesis y Aclaraciones especificas del comando

No posee

g. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)

#! /bin/bash

# Parametro: id del proceso a finalizar

function StopX {

if [ $# -eq 2 ]

then

return 1

fi

proceso=$1

# Chequeo que el proceso exista

if [ `ps | awk '{ print $1 }' | grep $proceso | wc -l` -eq 0 ]

then

return 1

fi

kill $proceso

return 0

}

StopX $1

a. Título: MoverX

b. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.

Centralizar el movimiento de archivos de un directorio a otro, verificando que no exista dicho archivo en el directorio destino y guardando los duplicados con un número de secuencia que indique el número de veces que se intentó copiar dicho archivo.

c. Archivos de Input, Intermedios y de Output

No posee.

d. Parámetros y Opciones

Recibe 3 parámetros:

- El archivo que se desea mover.

- El directorio al que se lo desea mover.

- El comando invocante.

e. Invocación Manual y/o automática con ejemplos

Ejemplo1:

./MoverX /home/archivo1.txt /bin "Console"

Mueve el archivo "archivo1.txt" del directorio /home al directorio /bin.

Ejemplo1:

./MoverX /home/archivo1.txt /bin "Console"

Mueve el archivo "archivo1.txt" del directorio /home al directorio /bin. Como dicho archivo ya existe, creará una carpeta llamada dup, donde almacenará "archivo1.txt" agregandole ".1" al final, para indicar que es el duplicado N° 1.

f. Hipótesis y Aclaraciones especificas del comando

- El primer parámetro contiene el directorio inicial y el nombre del archivo.

- El segundo parámetro sólo contiene el directorio destino.

g. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)

#! /bin/bash

function MoverX () {

# Si no se pasan los 3 parametros solicitados, es un error

if [ $# -eq 4 ]

then

return 1

fi

# Tomo los parametros

origen=$1

destino=$2

comando=$3

carpetaOrigen=${origen%/\*}

archivo=${origen##\*/}

# 1.1 Verifico si la carpeta de origen y de destino son iguales

if [ $carpetaOrigen = $destino ]

then

return 1

fi

# 1.2 Verifico que existan el archivo origen y la carpeta destino

if [ ! -f $origen ] || [ ! -d $destino ]

then

return 1

fi

# 1.3 Verifico si es un archivo duplicado

if [ -f "$destino/$archivo" ]

then

# Verifico si existe la carpeta dup, si no, la creo y genero el numero de secuencia

if [ ! -d "$destino/dup" ] || [ $(ls -1 $destino/dup | wc -l) = 0 ]

then

mkdir -p "$destino/dup"

nroSecuencia="0"

else

maxNroArchivo=`ls -l $destino/dup/$archivo\* | tail -1 | awk '{ print $NF }' | awk -F. '{ print $NF }'`

let "nroSecuencia = maxNroArchivo + 1"

fi

# Copio el archivo a la carpeta de duplicados

mv $origen "$destino/dup/$archivo.$nroSecuencia"

return 1

fi

mv $origen $destino

return 0

}

a. Título: GlogX

b. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.

Centralizar el registro de mensajes en logs, respetando el formato W5 y controlando el espacio utilizado por dicho log.

c. Archivos de Input, Intermedios y de Output

No posee.

d. Parámetros y Opciones

Recibe 4 parámetros:

- El comando que desea grabar un mensaje en un log.

- El Tipo de mensaje: I 🡪 INFORMATIVO, W 🡪 WARNING, E 🡪 ERROR, SE 🡪 SEVERAL ERROR.

- El mensaje a grabar.

- El nombre del archivo de log.

e. Invocación Manual y/o automática con ejemplos

Ejemplo1:

./GlogX "InterpreteX" "SE" "No se encontró el archivo ps-mae" "InterpreteX"

Registrará el mensaje: "No se encotró el archivo ps-mae" en el log InterpreteX.log (asumiendo que es la extensión definida por el usuario) con el siguiente formato:

Fecha Usuario InterpreteX SE "No se encontró el archivo ps-mae"

f. Hipótesis y Aclaraciones especificas del comando

- El cuarto parámetro tiene como intención facilitar al usuario el uso de otro nombre, distinto del del comando, para el log. Por ejemplo, podría querer almacenar los mensajes informativos y los warnings en un log y en otro los errores normales y severos.

- Para evitar repetir código, se verifica si el comando invocante es InstalarX, en cuyo caso se setean las variables correspondientes para dicho comando.

g. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)

#! /bin/bash

# 4to parametro: nombre del archivo de log sin extension

function GlogX {

# Cantidad de lineas a dejar

nroLineas=50

# Si no se pasan los 4 parametros solicitados, es un error

if [ $# -eq 5 ]

then

return 1

fi

# Chequeo si están seteadas las variables de ambiente

if [ -z $LOGEXT ] && [ -z $LOGDIR ] && [ -z $LOGSIZE ] && [ -z $CONFDIR ]

then

return 1

fi

comando=$1

tipoMensaje=$2

mensaje=$3

archivoLog=$4

directorioLog=$LOGDIR

fecha=$(date)

usuario=$(whoami)

if [ "$comando" = "InstalarX.sh" ]

then

directorioLog=$CONFDIR

LOGEXT=".log"

else

# Si no es el log de instalacion, veo el tamaño

if [ -f $directorioLog/$archivoLog$LOGEXT ] && [ `du -k $directorioLog/$archivoLog$LOGEXT | awk '{ print $1 }'` -gt $LOGSIZE ]

then

tail -50 $directorioLog/$archivoLog$LOGEXT > $directorioLog/$archivoLog$LOGEXT

echo $fecha $usuario $comando $tipoMensaje "Log Excedido" >> $directorioLog/$archivoLog$LOGEXT

fi

fi

# Escribo el log

# When Who Where What Why

echo $fecha $usuario $comando $tipoMensaje $mensaje >> $directorioLog/$archivoLog$LOGEXT

return 0

}

a. Título: VlogX

b. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.

Permitir visualizar los logs en forma simple y ordenada, sin tener que buscar en todo el archivo.

c. Archivos de Input, Intermedios y de Output

No posee.

d. Parámetros y Opciones

Recibe 3 parámetros

- El comando invocante.

- El filtro que puede ser:

- Un número que indica la cantidad de líneas que se desean ver.

- Un string que se desea buscar en el log.

- El nombre del archivo de log.

e. Invocación Manual y/o automática con ejemplos:

Ejemplo 1:

./VlogX.sh "Console" -10 "InicioX"

Mostrará las últimas 10 líneas del log de InicioX.

Ejemplo 2:

./VlogX.sh "Console" "no se encontró" "InicioX"

Mostrará las líneas del log de InicioX que contengan la frase "no se encontró".

f. Hipótesis y Aclaraciones especificas del comando

No aplica.

g. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)

#! /bin/bash

# Filtros:

# -n: muestra las ultimas n lineas

# string: filtra las lineas que tienen el string

function VlogX {

# Si no se pasan los 4 parametros solicitados, es un error

if [ $# -eq 4 ]

then

return 1

fi

comando=$1

filtro=$2

archivoLog=$3

directorioLog=$LOGDIR

usuario=$(whoami)

if [ comando = "InstalarX.sh" ]

then

directorioLog=$CONFDIR

fi

# Leo el log segun el filtro

if [[ $filtro =~ \-[0-9]+ ]]

then

tail $filtro $directorioLog/$archivoLog$LOGEXT

else

cat $directorioLog/$archivoLog$LOGEXT | grep -i $filtro

fi

return 0

}

VlogX $1 $2 $3