Cátedra de Sistemas operativos
UTN - FRBA

Título de documento	Shell scripting en Linux – Especificación de Trabajo práctico.
Contenido	El presente documento contiene la especificación formal del trabajo práctico número 2 de scripting en Linux.

Trabajo práctico Nro. 2

Cátedra de sistemas operativos

Shell scripting en Linux

Michelle

Versión 1.7

Imprima a doble faz

Índice

Introducción

El trabajo práctico Nº2 consiste en el desarrollo de una serie de scripts en lenguaje BASH sobre Linux. La estructura del mismo consta de las siguientes dos partes:
La primera de ellas, presenta diversos consejos y ejercicios de baja complejidad orientados a ofrecer un primer acercamiento del alumno al lenguaje.
La segunda, consiste en el desarrollo de scripts de mayor complejidad que atacarán problemas específicos de la administración de sistemas operativos.

Objetivos del trabajo práctico

Que el alumno:

- Conozca y comprenda la utilidad de los shell scripts para llevar a cabo la administración de Sistemas operativos.
- Conozca la estructura, sintáxis y semántica de un shell script en Linux.
- Desarrolle la capacidad para resolver problemas específicos de administración de sistemas UNIXes.

Características

- Modalidad de desarrollo: Grupal.
- Modalidad de entrega: Obligatoria.
- Duración estimada para su desarrollo: 5 semanas.
- Fecha de comienzo: Sábado 12/04.
- Fecha de finalización: Sábado 17/05.
- Fecha de entrega opcional: Sábado 17/05.
- Lugar de corrección: Designado por el ayudante. (Laboratorio o corrección vía mail).

Nomenclatura del documento

- Los términos que se encuentren en estilo subrayado figurarán en el glosario.
- Los párrafos que se encuentren en estilo *itálica* corresponderán a aclaraciones y/o ejemplos que faciliten la comprensión del texto previo.

Desarrollo

Primera parte – Introducción al shell scripting

A continuación se presentan una serie de ejercicios y consejos básicos que intentarán introducir al alumno en el mundo del shell scripting. Los conceptos aquí adquiridos serán de utilidad para el desarrollo de la segunda parte del TP 2.

- Modularización de programas: Recordar la filosofía de Linux, `Desarrollar componentes simples para luego integrarlos y producir componentes más complejos´. Se recomienda modularizar los scripts de manera tal de lograr mayor legibilidad y mantenibilidad del código. Investigar sobre la creación de funciones en bash y su agrupación en bibliotecas. Investigar sobre el built-in "." (punto) para `embeber´ scripts dentro de otros.

- ¿Qué es una variable de entorno? ¿Qué significa exportar una variable? ¿Cómo se la exporta? Cree un script llamado environ.sh que exporte una variable y luego imprima su valor en pantalla. Después ejecute dicho script en la forma "*./environ.sh"*. Al finalizar el script verifique si dicha variable se encuentra en el entorno del shell actual. Investigue qué ocurrió.

  Ejecute el mismo script en la forma *". ./environ.sh"* (notar el punto separado por un espacio). ¿Qué ocurrió ahora? Investigue sobre el built-in "." (punto).

- Grep: ¿Para qué es utilizado el comando grep?. ¿Cómo encontraría dentro del archivo */proc/cpuinfo* todas las líneas que comienzan con el carácter "a"?. Investigar sobre expresiones regulares.

- ¿Cómo se ejecutan comandos en segundo plano (background) en Linux? ¿Cómo hace para conocer qué procesos corren en segundo plano? ¿Cómo trae un proceso a primer plano nuevamente?.

- /proc: ¿Qué información contiene este directorio? Investigue exhaustivamente su contenido. ¿Qué representan los directorios numéricos?.

- /proc/net: ¿Qué información contiene este directorio? A medida que envía datos a través de la red (por ejemplo al navegar con firefox), investigue qué archivos son modificados y qué representan esos valores cambiantes.

- ¿Qué información contiene el archivo /etc/passwd?.
  ¿Para qué se utiliza el comando chsh? ¿Qué diferencia nota en sus permisos respecto a otros ejecutables?. ¿A qué cree se debe esa diferencia?

- SSH: Herramienta de logueo y ejecución de comandos remoto. Utiliza diversos algoritmos de encripción para proveer seguridad a la comunicación. Investigue su uso y como automatizar el logueo de un usuario en una máquina remota (es decir, sin requerir el ingreso de password), leer *man ssh-keygen* y analizar el uso del archivo *authorized_keys.*

- Procesamiento de textos (awk): awk es un lenguaje de programación destinado al procesamiento de textos. ¿Qué realizan las siguientes operaciones?:

  ls -l | awk '{print $6}'

  cat /etc/passwd | awk 'BEGIN {FS=":"} {print $1,$7}'

- Lecturas de **man** recomendadas:

  bash, chsh, ssh, ssh-keygen, netstat, lsof, proc, stat, ps, grep, make, chmod, cut, awk, pidof, renice, nice, ifconfig, tar, gzip, kill, cat, echo, install, bc, tty, who, w, wc.

- Comandos **built-in** a investigar:

  read, "." (punto) o su alias source, fg, bg, jobs.

Segunda parte – Resolución de problemas complejos de administración

Introducción

El proyecto consiste en el desarrollo de un shell (de ahora en más `Michelle´) íntegramente realizado mediante shell scripting. Dicho script será ejecutado sobre bash, siendo este último el shell por defecto asociado a todos los usuarios en sistemas Linux.
Michelle tendrá como principales características la posibilidad de incorporar/quitar módulos en forma dinámica, establecer configuraciones particulares a cada usuario y por último, al estar montado sobre bash, explotar todas las funcionalidades que este ofrece. De esta manera se puede pensar a Michelle como un wrapper (envoltorio) de bash al cual le adiciona diversas funcionalidades.

Flujo de ejecución ejemplificador

A continuación se presenta un ejemplo de uso básico de Michelle:

El usuario root descomprime el archivo michelle.tar.gz y ejecuta make instalar. Luego ejecuta make configurar y configura el uso de Michelle al usuario pepe.
Pepe se loguea en el sistema, se le presenta como shell a Michelle. En él se registra únicamente el módulo de auditoria.
Pepe ejecuta el comando *"ls /etc/michelle | grep michelle"*, entonces el núcleo lo procesa con el módulo de auditoria. Este retorna con código de salida exitoso, por lo tanto el núcleo deriva su ejecución a bash.

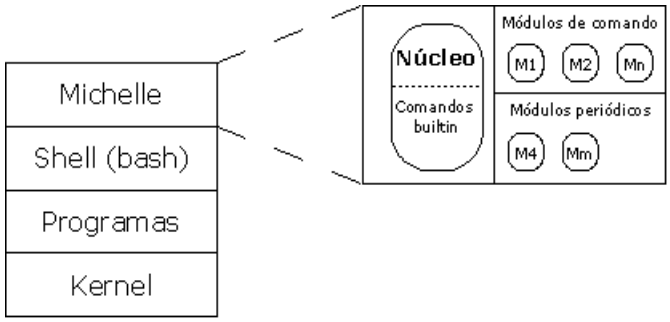
En otra terminal se loguea el usuario root y modifica el archivo de configuración de módulos de comando, agregando al usuario pepe el módulo de seguridad.
Luego, el root, crea el archivo de configuración del módulo de seguridad para el usuario pepe y le agrega el comando *"ls"* como comando restringido.
El núcleo luego de un tiempo detecta dicho cambio, entonces decide actualizar los módulos registrados.

Pepe ejecuta nuevamente el comando *"ls /etc/michelle | grep michelle"*, pero resulta que el módulo de seguridad detecta que es un comando restringido entonces retorna un código de error. Dicha situación hace que el núcleo informe en pantalla el error y no permita al usuario ejecutar el comando.

Componentes de Michelle

Núcleo o script principal

Este componente es el centro o corazón de Michelle, sobre él se registrarán *módulos periódicos* y *módulos de comando*. Será el encargado de leer los comandos tipeados por el usuario y procesarlos a través de los módulos registrados.



Este componente será invocado automáticamente por el sistema operativo al inicio de la sesión del usuario[1]. Luego de iniciado, llevará a cabo la función ***Registrar e inicializar módulos*** descripta más adelante. Por último, permanecerá en ejecución a la espera de comandos del usuario.

El núcleo, al igual que bash, ofrecerá una serie de comandos built-in. Para conocer cuáles son y una descripción detallada de los mismos dirigirse a ***Comandos Built-in de Michelle***.

[1] Sólo a aquellos usuarios a los cuáles se les activó el uso de Michelle. Para más detalle ver ***Makefile***, el target instalar.

Las principales funciones de este script[2] serán:

- **Registrar e inicializar módulos:**
  Como primera acción, el núcleo deberá verificar si existen actualmente módulos registrados[3], en caso de haberlos, los detendrá.
  Luego, realizará la lectura de los archivos de configuración de los módulos[4] y procederá a registrar[5] e inicializar a aquellos que se encuentran asociados al usuario logueado.

  *Aclaración importante: Si al iniciar un módulo, éste retorna un código de salida de **error** deberá informarse tanto en pantalla como en un archivo log[6] dicha situación y luego desloguear al usuario, finalizando de esta manera el shell asociado a esa sesión.*

  Para más información acerca de los archivos de configuración leer la sección ***Archivos de configuración***.

  Para más información acerca de las operaciones factibles sobre los módulos leer la sección ***Módulos***.

- **Verificar periódicamente el cambio de los archivos de configuración de módulos:**
  El núcleo, cada cierto tiempo[7], deberá detectar si se produjo un cambio en los archivos de configuración de los módulos[4] desde la última vez que se accedió a ellos.
  En caso de haberse producido, deberá informarlo por pantalla y llevar a cabo la función ***Registrar e inicializar módulos***.

  Esto permite que un usuario con permisos de escritura sobre dichos archivos pueda modificar en tiempo de ejecución los módulos registrados en Michelle.

- **Administrar la ejecución de los módulos periódicos:**
  El núcleo cada cierto período de tiempo[7] deberá ejecutar en forma secuencial los denominados *módulos periódicos*.
  Aclaración: Secuencial hace referencia a que deben ejecutarse en el mismo orden en que se encuentran registrados, que a su vez es el orden en el que figuran en el archivo de configuración de módulos periódicos.

  En caso que la ejecución de alguno de ellos genere como resultado un código de error, el núcleo deberá informarlo tanto por pantalla como en un archivo de log y desloguear al usuario actual, finalizando de esta manera el shell asociado a esa sesión.

- **Leer y procesar los comandos tipeados por el usuario:**
  El núcleo, al igual que cualquier shell, se encontrará permanentemente a la espera de comandos provenientes del usuario.
  Ante el ingreso de uno, el núcleo ejecutará en forma secuencial los denominados *módulos de comando* enviándoles el `string´ tipeado por el usuario para que estos lo procesen.

  Cómo se detalla más adelante, los módulos retornan un código de salida. Para el núcleo son de particular interés los siguientes:

  - Código de salida de error: Este le indica que el comando tipeado por el usuario produjo resultados negativos en el módulo. En ese caso, el núcleo cortará la ejecución de los siguientes e informará por pantalla dicha situación.
    Esto significa que si por ejemplo en el núcleo están registrados los módulos de comando m1, m2 y m3, al retornar con código de error el m2, no se continuará ejecutando al restante m3.
  - Código de salida exitoso: El procesamiento del comando fue correcto. Entonces se continúa ejecutando el siguiente módulo.

[2] No necesariamente debe ser implementado como un único script. Esto significa que puede ser desagregado en varios scripts y/o bibliotecas de funciones.
[3] Sólo podrán existir módulos registrados cuando se acceda a ésta mediante la función ***Verificar periódicamente el cambio de los archivos de configuración de módulos***.
[4] Por archivos de configuración de los módulos se hace referencia a aquellos que definen la lista de módulos de Michelle, tanto periódicos como de comando.
[5] La registración de los módulos se realizará en el mismo orden en el que figura en el archivo de configuración.
[6] El archivo de log residirá en el home del usuario, ejemplo: /home/pepe/michelle.log
[7] Deberá poder ser configurable mediante algún archivo de configuración definido por el grupo o dentro del mismo script a través de una variable.

Cuando todos los módulos se ejecutaron exitosamente, el núcleo procederá a invocar el comando del usuario en el shell (bash). En caso de tratarse de un built-in, lo procesará internamente en el script.

Ejemplo: El usuario tipea "ls", dicho comando produce resultados exitosos en cada uno de los módulos registrados, entonces michelle ejecuta "ls" sobre bash. Esto hace que se imprima en pantalla el listado de archivos en el directorio actual.

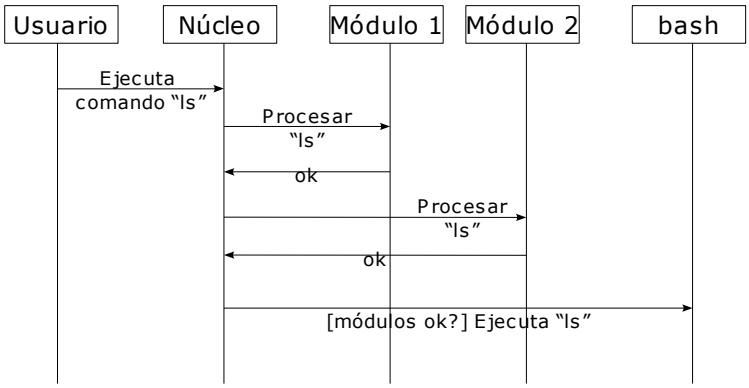
Aclaración importante: Recordar que Linux es un sistema operativo multiprogramado, por lo tanto Michelle deberá permitir ejecutar comandos en modo background, tal como lo hace posible bash.

Estas operaciones deberían ser factibles en Michelle:

*[Michelle] guest@localhost:~# find / -name "*michelle*"* ***&***
e inmediatamente
[Michelle] guest@localhost:~# ls */home/guest*

Diagrama de secuencia de procesamiento de comandos

A continuación se presenta un diagrama que expresa mediante un ejemplo la interacción entre los componentes de Michelle ante el ingreso de un comando del usuario.



Detalle de archivo asociados al núcleo	
Path de los scripts ejecutables del núcleo	/usr/bin
Permisos de acceso de los scripts ejecutables	rwxr-xr-x

Módulos

Los módulos son componentes implementados como shell scripts que proveen las funcionalidades específicas de Michelle. Estos pueden ser incorporados/quitados del sistema en forma dinámica mediante la simple edición de los archivos de configuración correspondientes (Para más detalle leer ***Archivos de configuración***).

Una característica importante de los módulos es que respetan una cierta interfaz[8]. Esto le permite al núcleo el poder tratarlos a todos por igual sin necesidad de conocer qué módulo específico tiene registrado.

Por lo tanto, la forma de invocar a cualquier módulo es la siguiente:

<nombreModulo> <tipoOperacion> [<parametro>] [9]

Donde <nombreModulo> es el path absoluto del script ejecutable del módulo, <tipoOperacion> es una cadena que indica el tipo de operación que se quiere llevar a cabo sobre el módulo y <parametro> representa a los argumentos que recibirá el módulo.

A continuación se especifica la interfaz mínima[10] de un módulo:

[8] Para investigar en mayor detalle cómo se implementa una interfaz común entre diversos scripts, analizar **exhaustivamente** los scripts ubicados en el path "/etc/init.d". ¿Qué representan estos scripts?, ¿Para qué y cómo se los utiliza?.
[9] Los corchetes indican que el parámetro puede o no ser enviado, es decir, es opcional. Esto dependerá del tipo de módulo que se haya ejecutado. Para más información leer las secciones ***Módulos de comando*** y ***Módulos periódicos***.
[10] Se indica mínima porque el grupo puede agregar otros tipos de operaciones.

Tipo de operación	Descripción
Información	Cuando a un módulo se lo invoque con el parámetro tipoOperacion igual a información**,** éste deberá imprimir en la salida estándar los datos específicos del mismo. Para conocer qué información debe presentar cada módulo, leer la sección ***Especificación detallada de módulos***.
Iniciar	Cuando a un módulo se lo invoque con el parámetro tipoOperacion igual a **iniciar**, éste deberá leer su archivo de configuración y exportar las variables de entorno que se desprendan del config con su correspondiente valor. Esto significa que el módulo no leerá el archivo de configuración cada vez que ejecute la operación "procesar", sino que los valores de configuración los tomará de las variables de entorno previamente exportadas. ***Consejo**: Investigar sobre el comando built-in "." (punto) para conocer cómo modificar en un script hijo las variables de entorno del script invocador o padre.*
Detener	Cuando a un módulo se lo invoque con el parámetro tipoOperacion igual a **detener**, éste eliminará todas las variables de entorno exportadas al ejecutar la operación "iniciar". También, si el grupo lo considera necesario, en esta operación se podrá hacer algún tipo de limpieza de archivos u otros.
Procesar	Cuando a un módulo se lo invoque con el parámetro tipoOperacion igual a **procesar**, éste realizará el procesamiento específico del módulo. Para conocer qué proceso debe hacer cada módulo leer ***Especificación detallada de módulos***.

Una característica común a todos los tipos de operación es que retornan un código de salida[11]. Este código podrá tomar diversos valores que dependerán básicamente del éxito o fracaso del procesamiento realizado.

*Por ejemplo, si existe un módulo que no permite al usuario ejecutar el comando "ls" y el usuario ingresa dicho comando en Michelle, el módulo, al ser invocado con el tipoOperacion igual a procesar, generará un código de salida de **error**, en caso contrario de **éxito.***

Detalle de archivo asociados a un módulo	
Path de los scripts ejecutables de un módulo	/etc/michelle/modulos/<nombreModulo>/ Donde <nombreModulo> es el nombre que el grupo haya asignado al módulo. *Ejemplo: El script del módulo de seguridad residirá en: /etc/michelle/modulos/seguridad/**seguridad.sh***
Permisos de acceso de los scripts ejecutables de un módulo	rwxr-xr-x

Existen dos tipos básicos de módulo, los de comando y los periódicos. A continuación se explican ambos en detalle:

Módulos de comando

Son aquellos que se encargan de procesar comandos tipeados por el usuario. Estos módulos serán invocados por el núcleo ante el ingreso de cada comando.

Como se aclaró anteriormente, la forma de invocar a un módulo es la siguiente:

<nombreModulo> <tipoOperacion> [<parametro>]

En estos módulos, <parametro> hace referencia al comando tipeado por el usuario. De esta manera, si se ingresa "ls | grep pepe" la invocación al módulo sería por ejemplo:

/etc/michelle/modulos/moduloX/moduloX.sh procesar "ls | grep pepe".

Módulos periódicos

Se denominan así a los módulos que serán invocados periódicamente por el núcleo. A estos módulos no necesariamente se les deberá enviar <parametro>[12].

Archivos de configuración (config)

En esta sección se presentarán los archivos de configuración asociados a los componentes de Michelle[13].

Configs del núcleo o script principal

[11] Los valores de los códigos de salida de un módulo podrán ser elegidos a gusto del grupo (Siempre y cuando permitan distinguir una condición de otra). Adicionalmente, el grupo, si así lo considera, podrá agregar otros tipos de código de salida aparte del de *éxito* y *error*.
[12] Si el grupo lo considera necesario podrá hacerlo.
[13] Si el grupo lo desea puede agregar más archivos de configuración que los presentados en este documento.

El núcleo tendrá dos archivos de configuración, uno con la lista de módulos de comando y otro con la lista de módulos periódicos. Dichos archivos residirán en el path *"/etc/michelle/modulos"* y tendrán los permisos "rw-r--r--".

La idea de estos archivos es que declaren los módulos de los cuales dispone Michelle y a su vez asociarlos a usuarios del sistema.

Ambos archivos tendrán la misma sintaxis, a saber:
Por cada módulo se creará una línea con el path absoluto del mismo, luego un ":" (dos puntos) como separador y luego la lista de usuarios[14] a los cuales se les activa dicho módulo (separados por coma).

Ejemplo: Dados dos módulos de comando y uno periódico, el contenido de los archivos puede ser el siguiente:

Config de módulos de comando
/etc/michelle/modulos/seguridad/seguridad.sh:pepe,jose
/etc/michelle/modulos/auditoria/auditoria.sh:

Config de módulos periódicos
/etc/michelle/modulos/limitacion/limitacion.sh:jose

Aclaración: En el ejemplo el módulo de auditoria no se encuentra asociado a ningún usuario.

Configs de los módulos

Cada módulo de Michelle tendrá por cada usuario un archivo de configuración. Estos archivos residirán en la ruta *"/etc/michelle/modulos/<nombreModulo>/config"* y serán creados manualmente por el usuario root del sistema. Tendrán los permisos "rw-r--r--".

Donde <nombreModulo> es el nombre que el grupo haya asignado al módulo.

El nombre del archivo de configuración se corresponderá con el nombre del usuario al cual hace referencia.

Ejemplo:
Dado un módulo de nombre "seguridad" y dos usuarios, pepe y josé. Existirán entonces los siguientes archivos:
/etc/michelle/modulos/seguridad/config/pepe
/etc/michelle/modulos/seguridad/config/jose
Y su contenido dependerá de cada módulo y sus requerimientos. Para más detalle ver ***Especificación detallada de módulos****.*

Makefile

Michelle utilizará como herramienta de instalación/desinstalación y configuración de usuarios, la aplicación make.
El grupo, por lo tanto, deberá desarrollar un archivo makefile que contenga mínimamente los siguientes targets:

Target	Descripción
Instalar	Al ejecutar *make instalar*, se distribuirán todos los shell scripts y archivos de configuración a sus respectivos directorios con los permisos indicados previamente. *Aclaración: Para el caso de los configs de cada módulo, se debe copiar un archivo de configuración plantilla para que pueda ser utilizado por el usuario root como base para crear los configs propios de cada usuario.* Luego de esto, se deben cambiar los permisos del programa */usr/bin/chsh* a "rwxr-xr-x".
Desinstalar	Al ejecutar *make desinstalar*, se eliminarán todos los archivos asociados a Michelle y luego se restablecerán los permisos del programa */usr/bin/chsh.* *Tener en cuenta que esto será utilizado al finalizar la evaluación del grupo para limpiar todo rastro de Michelle en el sistema, con lo cual no debería permanecer nada que pueda comprometer la evaluación de los próximos grupos.*
Configurar	Este target permite configurar el uso de Michelle a un usuario específico. De esta manera cuando dicho usuario se loguee, el sistema operativo le presentará automáticamente a Michelle en lugar de bash. Al ejecutar *make configurar*, se solicitará por línea de comando un nombre de usuario al que luego se le asignará Michelle como shell por defecto.

[14] Los nombres de los usuarios se corresponderán con los nombres de la cuenta en Linux.

Aclaración: Sólo el usuario root podrá ejecutar los targets del makefile. En caso de intentar acceder con otro usuario, informar por pantalla dicha situación y salir.

Comandos Built-in de Michelle

A continuación se presenta una lista de comandos embebidos en Michelle:

Comando	Descripción
Ayuda	Presenta en pantalla una breve ayuda sobre los comandos built-in y su uso.
Información	A cada módulo registrado en Michelle le indica que imprima en pantalla información sobre sí mismo. *Para más detalle sobre las operaciones factibles sobre un módulo leer* ***Módulos****.* Si al built-in se lo invoca en la forma *informacion <nombreModulo>*, sólo imprimirá la información del módulo indicado.
ListarModulos	Presenta en pantalla los paths absolutos de los módulos registrados en Michelle, es decir, aquellos que tiene activos el usuario.
ActualizarModulos	Invoca a la función del núcleo ***Registrar e inicializar módulos***.
Salir	Termina la sesión actual del usuario.
Apagar	Apaga la pc.

Especificación detallada de módulos

En esta sección se explicarán en forma detallada cada uno de los módulos que podrán registrarse en Michelle.[15]

Módulos de comando

Módulo de seguridad

Se encargará de restringir la ejecución de ciertos comandos al usuario. Dichos comandos figurarán en el archivo de configuración del módulo (Recordar que hay uno por usuario y son creados manualmente por el root en base a un config plantilla que debe crear el grupo).

Ejemplo: Dado un usuario pepe y el siguiente contenido de su config para el módulo de seguridad:
Archivo /etc/michelle/modulos/seguridad/config/pepe:
ssh
scp
telnet

El módulo producirá los siguientes códigos de salida para los comandos tipeados:
"ls" → éxito
"ls | grep hola" → éxito
"ssh 192.168.2.10" → error
"echo Hola | telnet 192.168.2.10" → error

Al invocar al módulo con el <tipoOperacion> igual a información, deberá imprimir en pantalla los comandos restringidos al usuario.

Módulo de auditoria

Se encargará de loguear en un archivo los comandos tipeados por el usuario. El archivo residirá en el home del usuario.

Parámetros configurables del módulo	
Tamaño máximo de archivo local de log	Indica el tamaño máximo admitido para el archivo de log local de Michelle. Cuando el mismo supere dicho valor, todo logueo deberá realizarse en forma remota sobre el servidor indicado por el parámetro *IP del servidor de log.* *Aclaración I: La unidad de tamaño será en bytes.* *Aclaración II: El logueo remoto deberá ser realizado sobre un archivo que resida en el home del usuario de mismo nombre al usuario local. Ejemplo, si el usuario local es pepe, deberá loguear en el path /home/pepe del servidor remoto. (Se asume la existencia de dicho usuario en el servidor de log, en caso de no existir se creará manualmente).*

[15] Adicionalmente, durante la evaluación del TP, podrán proporcionarse al grupo nuevos módulos desarrollados por la cátedra que deberán integrarse sin problemas a Michelle.

IP del servidor de log	IP del servidor sobre el cual se realizará el logueo en caso de superar el tamaño máximo de archivo de log local.

El módulo retornará siempre un código de salida exitoso (Salvo que el grupo decida crear otros códigos para diferenciar situaciones).

Al invocar al módulo con el <tipoOperacion> igual a *información*, deberá imprimir en pantalla los valores de los parámetros de configuración y el tamaño actual del archivo local de log.

Módulo de control de sesiones

Se encargará de limitar la cantidad de sesiones de Michelle abiertas por el usuario.
Este módulo, a diferencia de los explicados anteriormente, no realizará ninguna operación cuando se lo invoque con el <tipoOperacion> igual a *procesar*.
El control de sesiones lo realizará cuando se inicie el módulo, es decir, al invocarse con el <tipoOperacion> igual a *iniciar*.
De esta manera, se controlará la cantidad de sesiones abiertas por el usuario al momento de cargar el módulo en Michelle.

Como se explicó en la sección ***Módulos***, se retornará un código de salida de error si la cantidad de sesiones supera la máxima establecida. Dicho valor deberá ser leído del archivo de configuración.

Al invocar al módulo con el <tipoOperacion> igual a *información*, deberá imprimir en pantalla los valores de los parámetros de configuración y la cantidad actual de sesiones abiertas por el usuario.

Módulos periódicos

Módulo de limitaciones

Establecerá limitaciones respecto al consumo de recursos dentro de una sesión de Michelle.[16] A continuación se presentan los parámetros de configuración y un detalle de las limitaciones que deben realizarse.

Parámetros configurables del módulo	
Máximo uso de CPU por sesión	Es un valor porcentual que representa el máximo consumo de CPU que puede realizar el usuario en una terminal de Michelle. El uso de CPU por sesión se obtiene como sumatoria de todos los consumos de CPU de los procesos activos[17] iniciados en la sesión.
Máximo uso de Memoria por sesión	Al igual que con el máximo uso de CPU, con la diferencia que tiene en cuenta el consumo de memoria de todos los procesos activos iniciados en la sesión.
Máximo de procesos por sesión	Valor entero que indica la cantidad máxima de procesos activos que puede tener un usuario en una sesión de Michelle.
Máximo de sockets por sesión	Valor entero que indica la cantidad máxima de sockets abiertos por procesos activos iniciados en la sesión. Esta operación deberá ser llevada a cabo mediante el acceso al /proc. Para más información *man proc*.
Máximo de archivos abiertos por sesión	Valor entero que indica la cantidad máxima de archivos abiertos por procesos activos iniciados en la sesión.

El módulo producirá un código de salida erróneo si alguna de las limitaciones es excedida.

Al invocar al módulo con el <tipoOperacion> igual a *información*, deberá imprimir en pantalla los valores de los parámetros de configuración y el valor actual de cada una de las restricciones.

Módulo de control de carga

Realizará un seguimiento del consumo de CPU de los procesos asociados a una sesión de Michelle, intentando mantenerlo debajo de un determinado nivel.

[16] Por sesión se hace referencia a la relación que se establece entre el usuario y el sistema operativo tras el logueo. En una misma PC pueden existir múltiples sesiones.
[17] Procesos actualmente en ejecución, bloqueados o listos para ejecutar.

Más en detalle, dado un nivel máximo de consumo de CPU por proceso obtenido del config, el módulo seleccionará al primer proceso que supere (en mayor medida) dicho consumo y luego le incrementará en 5 unidades su valor de nice.

Cuando un proceso sufra de manera continua 4 incrementos del nice, el módulo lo eliminará del sistema e imprimirá en pantalla dicha acción.

El módulo producirá siempre un código de salida exitoso.

Al invocar al módulo con el <tipoOperacion> igual a *información*, deberá imprimir en pantalla los valores de los parámetros de configuración, el proceso, su nice actual y la cantidad de incrementos de nice del mismo (en caso de existir).

Módulo de limitación de tráfico de red

Intentará establecer una limitación a la cantidad de paquetes IP salientes del nodo de red.

Más en detalle, cuando el módulo detecte que la cantidad de paquetes IP salientes del nodo supera el valor máximo (obtenido del config), se realizará lo siguiente:
Por cada proceso activo iniciado en la sesión actual que posea uno o más sockets con conexiones externas[18], se informarán por pantalla los sockets y luego se procederá a eliminar al proceso del sistema.
De esta manera se penalizan a aquellos procesos del usuario de la sesión actual que puedan llegar a producir tráfico en la red.

El módulo producirá siempre un código de salida exitoso.

Al invocar al módulo con el <tipoOperacion> igual a *información*, deberá imprimir en pantalla los valores de los parámetros de configuración y la cantidad actual de paquetes IP salientes.

Distribución de Michelle

La aplicación será distribuida en un archivo del tipo tar.gz. Este contendrá todos los archivos necesarios para poder instalar, ejecutar y desinstalar Michelle en una pc con Linux.

Glosario

- Shell: Sistema que actúa como interfaz entre el usuario y la aplicación (generalmente el sistema operativo). Traduce los eventos del usuario a eventos comprensibles por la aplicación destino y viceversa.
- Shell script: Programa interpretado por el shell del sistema operativo. Su principal uso está asociado a la administración de sistemas.
- BASH: Bourne again Shell. Shell por defecto utilizado en los sistemas operativos actuales del tipo Linux.
- Comando built-in: Comando embebido en el propio shell. Esto significa que para ejecutarlo no debe invocarse a un programa externo.
- Módulo de Michelle: Componente que provee un servicio específico. Un módulo puede ser incorporado/quitado en forma dinámica, es decir, en tiempo de ejecución. *Una analogía a los módulos de Michelle son las extensiones que pueden agregarse dinámicamente a Firefox.*
- Interfaz: Servicios que expone un componente.
- Make: Herramienta utilizada para automatizar el proceso de compilación, instalación de aplicaciones.
- Nodo de red: Computadora conectada físicamente a la red, se la identifica de manera unívoca mediante un número denominado IP.

[18] Es decir, sockets conectados con otros nodos de red, no con localhost.