SISTEMAS OPERATIVOS

TALLER No. 3: Conceptos básicos de procesos en Linux y herramientas complementarias

OBJETIVOS

1. Comprender el concepto de proceso en Linux.
2. Aprender a usar e interpretar las herramientas básicas del sistema operativo para el listado y el monitoreo de procesos en ejecución.
3. Aprender el uso básico de las páginas del manual de Linux y las ayudas de los comandos.
4. Aprender a instalar software de terceros en el sistema operativo Linux usando el código fuente del software.
5. Aprender a recibir parámetros desde el sistema operativo en un programa de C al momento de crear el proceso asociado al programa.
6. Aprender a retornar un valor de estado de terminación de un programa en C como resultado de la ejecución del proceso.
7. Aprender a reconocer un proceso en ejecución y a enviarle una señal de terminación.

CONTENIDO DEL TALLER

1. Listar procesos en Linux.
2. Páginas del manual de Linux.
3. TL;DR: Consulta rápida del funcionamiento de un comando.
4. Monitoreo de procesos en Linux.
5. Descarga e instalación de las herramientas de desarrollo en Linux CentOS.
6. Comprobar la instalación del compilador de C/C++ y de la utilidad **`make`**.
7. Descarga, compilación e instalación de software de terceros.
8. Trivia.
9. Compilación y ejecución de un programa en lenguaje C en Linux.
10. Argumentos de la función **`main`**.
11. Valor retornado de un proceso al sistema operativo.
12. Trivia.
13. Uso del comando **`size`**.
14. Identificación de su proceso en ejecución y señalización de terminación.

DESARROLLO DEL TALLER

1. Listar procesos en Linux.

   Para listar procesos en Linux se usa el comando **`ps`**, la información básica que entrega este comando (sin opciones/parámetros adicionales) es la siguiente. Ejecute el comando **`ps`** desde una terminal conectada por PuTTy al sistema operativo.

   - **PID:** Identificador del proceso o Process ID.
   - **TTY:** El dispositivo terminal en donde el proceso está ejecutando. Pregunte sobre esto.
   - **TIME:** Cantidad total de tiempo que el proceso ha pasado ejecutando instrucciones en el procesador.
   - **CMD:** Comando ejecutado por el proceso con todos sus argumentos/parámetros pasados al proceso.

   Algunas opciones de uso común con el comando **`ps`** son las siguientes. Pruébelas usted.

   - **`ps x`**: Muestra todos los procesos en ejecución.
   - **`ps ax`**: Muestra todos los procesos en el sistema, no solo los suyos.
   - **`ps u`**: incluye más información detallada en el listado de procesos.
   - **`ps w`**: Muestra la línea de comandos (columna **CMD**) completa, no solo la que cabe en una línea.

2. Páginas del manual de Linux.

   La fuente original de la documentación de Linux está instalada con el sistema operativo. Por ejemplo, si usted quiere consultar la documentación del comando **`ps`**, lo puede hacer de la siguiente manera.

   ```
   # man ps
   ```

   Las páginas del manual de Linux se pueden navegar con las flechas de desplazamiento en el teclado y con las teclas **PágArriba** y **PágAbajo**. Para salir de la página del manual de Linux simplemente presione la tecla **`q`**.

   De la misma manera que se consulta la página del manual de **`ps`**, puede usted consultar la página del manual de comandos como **`ls`**, **`mv`**, **`cp`**, **`pushd`**, entre otros.

3. TL;DR: Consulta rápida del funcionamiento de un comando.

   Si las páginas del manual de Linux le parecen muy agobiantes, la mayoría de los comandos, herramientas y llamadas al sistema (*system calls*) en el sistema operativo Linux, soportan la versión resumida de su página del manual. Por ejemplo, si queremos saber qué hace el comando **`ls`** y las opciones o parámetros que soporta al momento de su ejecución, podemos ejecutar lo siguiente.

   ```
   # ls --help
   ```

   Si la versión resumida de la documentación no resuelve sus preguntas, en los foros de Linux se suele usar una expresión muy común para ello: **RTFM.**

4. Monitoreo de procesos en Linux.

   Para el monitoreo de los procesos en Linux y los recursos que están usando, se usa el comando **`top`**. Esta herramienta muestra de manera interactiva el comportamiento de los procesos y los recursos en el sistema operativo. Consulte la página del manual de **`top`** para comprender la información que éste muestra sobre los procesos.

   Para cerrar el comando **`top`**, simplemente pulse la tecla **`q`**.

   De acuerdo con la documentación del comando **`top`**, responda las siguientes preguntas.

   4.1 ¿Con qué letras se identifica el estado de un proceso?
   4.2 ¿Cuál es el nombre de los diferentes estados de un proceso en Linux?
   4.3 ¿Cuál es la columna, en la salida del comando **`top`**, que indica el estado en el que se encuentra un proceso?
   4.4 ¿Qué indica la columna **PPID** en la salida del comando **`top`**? Si no ve la columna **PPID**, mientras se está ejecutando **`top`**, pulse la tecla **`g`**, seguida del número 2. La utilidad **`top`** tiene 4 vistas de la información que usted puede explorar pulsando la tecla **`g`** seguida de un número del 1 al 4.
   4.5 ¿Qué significan las unidades de medida KiB o MiB en la información de **`top`** (consulte la página del manual de **`top`**)?

5. Descarga e instalación de las herramientas de desarrollo en Linux CentOS.

   Ejecute la siguiente orden sobre el shell del sistema operativo:

   ```
   # yum group install “Development Tools”
   ```

NOTA: Si copia y pega la línea anterior asegúrese de que las comillas dobles peguen correctamente o reescríbalas para evitar errores en la ejecución del comando.

A las preguntas que se le hagan en el proceso de descarga e instalación de las herramientas de desarrollo, responda escribiendo **y** seguido de un **ENTER**.

6. Comprobar la instalación del compilador de C/C++ y de la utilidad **make**.

   Una vez finalizado el proceso de instalación de las herramientas de desarrollo, compruebe que el compilador de C/C++ se encuentra correctamente instalado.

```
# gcc --version
```

   Debería obtener una salida similar a la siguiente.

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

   Compruebe que tiene disponible la utilidad **make** de la siguiente manera.

```
# make --version
```

   Debería obtener una salida similar a la siguiente.

```
GNU Make 3.82
Built for x86_64-redhat-linux-gnu
Copyright (C) 2010  Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

7. Descarga, compilación e instalación de software de terceros.

   En Linux es común disponer de software de terceros que no están incluidos en la distribución instalada pero que pueden compilarse e instalarse. La instalación típica de software de terceros en Linux incluye lo siguiente, pero podría variar dependiendo del software que se proponga instalar en el sistema.

a. Descargar los archivos fuente. Usualmente comprimidos en formato `.tar.gz.` u otros formatos de compresión y empaquetamiento de archivos.
b. Ubicarse en el directorio en donde se encuentran los archivos descargados.
c. Descomprimir los archivos fuente descargados con el siguiente comando para archivos comprimidos o empaquetados en formato `.tar.gz`.

```
# tar -zxvf <nombre_archivo.tar.gz>
```

d. El proceso de descomprimir el archivo .tar.gz, crea usualmente un directorio con el nombre del archivo, pero sin la extensión .tar.gz. A continuación, se entra al directorio de los archivos descomprimidos. Por ejemplo.

```
# cd nombre_archivo
```

e. Leer el archivo **README** que casi siempre se incluye con los archivos fuente. En este archivo se suelen ampliar detalles del proceso de compilación e instalación y por esta razón es importante leerlo antes. El archivo **README** se puede leer con el comando **`more`**. Por ejemplo.

```
# more README
```

f. Configurar la compilación ejecutando el comando **`./configure`**
g. Compilar el software con el comando **`make`**.
h. Instalar el software con el comando **`make install`**.

De acuerdo con lo indicado anteriormente, realice lo siguiente:

7.1 descargue los archivos fuente de la utilidad **`htop`** (p. ej.: `htop-3.1.2.tar.xz`) desde el siguiente URL: https://github.com/htop-dev/htop/releases.

7.2 Copie el archivo `.tar.xz` descargado a su sistema operativo Linux a un directorio llamado **`downloads`** dentro del directorio **`/root`**. Si el directorio **`downloads`** no existe dentro del directorio **`/root`**, por favor crearlo con el comando **`mkdir`**.

7.3 Compile e instale el software descargado, pero tenga en cuenta que para desempaquetar el archivo **`.tar.xz`**, debe usar las opciones **`-xz`** del comando **`tar`**. Por ejemplo.

```
tar -xf htop-3.1.2.tar.xz
```

Es posible que, durante la fase inicial de configuración de la compilación, se le presente el siguiente mensaje de error (sic).

```
configure: error: can not find required library libncursesw; you may want to use --
disable-unicode
```

No se asuste. El error está indicando que para poder compilar el software en el sistema hace falta una librería llamada **`ncurses`** que no está instalada. Para instalar este componente faltante lo podemos hacer de dos formas:

a. Buscamos los archivos fuentes, compilamos e instalamos a mano, o
b. usamos el gestor de paquetes YUM de Linux CentOS para instalar este elemento faltante.

Para simplificar el proceso nos iremos por la opción **b** usando el siguiente comando.

```
# yum install ncurses*
```

A las preguntas que se hagan en el proceso, responda pulsando la tecla **y**.

Con la librería faltante instalada, repita el proceso de configuración de la compilación del software **`htop`**, compile e instale.

8. Trivia

8.1 ¿Dónde quedó instalado **`htop`**? en carpeta htop-3.1.2
8.2 ¿Cómo puede lanzar la ejecución de la utilidad **`htop`**? en esta carpeta ejecutar comando "htop"
8.3 ¿Qué es **`htop`** y para qué sirve? es como un admin de tareas

9. Compilación y ejecución de un programa en lenguaje C en Linux.

Escriba su primer programa “hola mundo” en lenguaje C en Linux y compílelo con el siguiente comando.

```
# gcc -o holamundo holamundo.c
```

9.1 ¿Cómo reconocer el binario producido?

El anterior comando produce el archivo **holamundo**. La opción **-o** indica al compilador de C/C++ que al compilar el archivo fuente **holamundo.c**, se produzca el binario con el nombre **holamundo**.

9.2 ¿Dónde está el binario producido?

En la misma ubicación en donde se ejecutó el comando **gcc**. Si usted ejecuta el comando **ls -l** para listar el contenido del directorio del actual, podrá ver los dos archivos: fuente y binario.

9.3 ¿Cómo reconocer qué tipo de archivo es cualquier archivo en Linux?

A diferencia de Windows, en Linux no se maneja el concepto de extensión del archivo como una manera de tipificar los archivos. Cuando exista alguna duda con relación a un archivo, se puede usar el comando **file** seguido del nombre del archivo. Por ejemplo, para saber qué tipo de archivo es holamundo, basta ejecutar el siguiente comando.

```
# file holamundo
```

El resultado debería ser similar al siguiente.

```
holamundo: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically link
ed (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=fc07b077be894c7a3886f
2d12dac66edf09182a7, not stripped
```

9.4 ¿Cómo crear el proceso a partir del archivo ejecutable?

Suponiendo que usted se encuentra en el directorio en donde está el ejecutable producido, para crear el proceso ejecute el siguiente comando.

```
# ./holamundo
```

10. Argumentos de la función **main**.

Un proceso se puede crear a partir de una orden del usuario en la línea de comandos, en la interfaz gráfica de usuario, o de manera automática por el sistema operativo. Por ejemplo, cuando indicamos la orden **./holamundo** en el caso anterior, estamos indicándole al sistema operativo que cree el proceso asociado a este sencillo programa.

Podemos usar los argumentos de la función **main** para pasar información desde el sistema operativo al proceso que se desea crear. Para ello se usan los argumentos **argc** y **argv** de la función **main** de nuestro programa.

- `int argc`: indica el número de argumentos que se le han pasado al programa en el momento de la ejecución.
- `char *argv[]`: un arreglo de apuntadores a los parámetros pasados al programa en el momento de su ejecución.

Escriba y compile el siguiente programa.

```
1  #include <stdio.h>
2
3  int main (int argc, char *argv[])
4  {
5      printf("argc = %d\n", argc);
6      printf("argv[0] = %s\n", argv[0]);
7      printf("argv[1] = %s\n", argv[1]);
8      printf("argv[2] = %s\n", argv[2]);
9
10     return 0;
11 }
```

- Ejecute el programa anterior sin pasar parámetros al proceso y observe el resultado. Ejecútelo como hasta ahora ha ejecutado su **holamundo** en Linux.
- ¿Cómo puede pasarle opciones/parámetros a este programa al momento de ejecutarlo?
- Ejecute el programa pasando opciones/parámetros. Los que usted quiera y observe el resultado.

11. Valor retornado de un proceso al sistema operativo.

    De la misma manera que se le pasa información a un proceso al momento de crearlo a través de los argumentos de la función **main**, el proceso una vez pasa al estado de terminado, puede entregar información al sistema operativo mediante la instrucción **return** como aparece en la línea 10 del programa anterior.

    Escriba un programa en lenguaje C que, dependiendo del valor de los parámetros pasados al proceso, retorne **0** o retorne **1**. Define usted la lógica de su algoritmo y de los parámetros recibidos que usted validará. Valide que se reciba el número de parámetros esperado por la lógica de su programa. En cada caso el retorno del valor se hace con la instrucción **return 0** o **return 1**, respectivamente.

12. Trivia.

    12.1 ¿Cómo podría en Linux saber cuál fue el valor retornado al finalizar la ejecución de un proceso?

13. Uso del comando **size**

    Sobre el archivo ejecutable de cualquiera de los dos programas anteriormente compilados por usted, ejecute el comando **size** de la siguiente manera.

```
# size <nombre archivo ejecutable>
```

13.1 ¿Cuál es el resultado de la utilidad **`size`** sobre el archivo ejecutable?
13.2 ¿Qué mide la utilidad **`size`** sobre el archivo ejecutable?
13.3 ¿Qué estaría indicando la utilidad **`size`** sobre el modelo de memoria del programa cuando se convierte en proceso?

14. Identificación de su proceso en ejecución y señalización de terminación.

    a. Escriba un programa en C que en un ciclo infinito imprima el mensaje que usted quiera.
    b. Compile el programa y ejecútelo. Como es de esperarse, su programa se quedará ejecutando en un ciclo infinito.
    c. Abra una nueva terminal de conexión al sistema operativo y con las herramientas de gestión básica de procesos, intente ubicar el proceso asociado al programa del ciclo infinito. Identifique el PID del proceso y el PID del proceso padre.
    d. De regreso en la terminal donde tiene el proceso ejecutando el ciclo infinito, pulse **CRTL+C** para detener el proceso. Si no se detiene, desde una segunda terminal ejecute el comando **`kill -9 <PID>`** donde **<PID>** corresponde al PID obtenido por usted.