Cómo corer procesos en CHILDHOOD – PARANOIDE

1.- Introducción:
Los servidores de cálculo del GHS (Paranoide y Childhood) tienen el sistema operativo Linux. Este sistema es un derivado de UNIX y básicamente corren programas en el modo "consola de comandos", muy parecido al sistema DOS. La idea es que, como carecen de interfaz gráfica, son más aptos para ejecutar programas de cálculo. Para aprender cómo conectarnos a los servidores (tarea que se realiza mediante un sistema llamado Secure Shell), se recomienda la lectura de la guía "Ejecución de procesos en forma remota", que se encuentra en la web del Grupo.

En adelante nos referiremos a un *proceso* entendiendo por tal cosa *un programa en ejecución*. Usaremos el símbolo "$" (dólar) para indicar la línea de comandos del servidor. Utilizaremos por generalidad el término "miprograma" para denotar el ejecutable que deseamos correr, y "archivos_de_datos" para referirnos a uno o más archivos que "miprograma" requiera como entradas. Se entenderá que luego de escribir una orden en la línea de comandos, debemos presionar la tecla INTRO para que la orden se ejecute.

2.- Modo interactivo
Un proceso se ejecuta en modo interactivo cuando damos la orden
$ miprograma
y recibimos un mensaje del estilo "Dame el nombre del archivo de datos". Es decir, cuando el proceso en ejecución necesita que tecleemos un valor o una cadena de caracteres. Es la forma básica de ejecutar procesos, pero tiene el inconveniente de que una vez lanzado el proceso, si cerramos la ventana o apagamos el ordenador local (nuestra máquina), el proceso se corta. Vale decir que para correr un programa en interactivo, deberemos dejar abierta la ventana de SecureShell, lo cual no es muy práctico, sobre todo si deseamos enviar a ejecutar un proceso largo. Esto se debe a que la gran mayoría de los programas escriben algún tipo de información en pantalla y por lo tanto esperan que exista una ventana donde escribir. Si no hay tal ventana (o "terminal"), se produce un error y el proceso se corta.

3.- Modo background
Este modo también requiere dejar abierta la ventana de SecureShell, pero tiene la ventaja de devolvernos la línea de comandos para enviar más procesos o controlar la ejecución de alguno que ya está corriendo. Como no nos permitirá teclear nada, si "miprograma" requiere teclear el nombre de un archivo de datos, deberemos dárselo en la propia línea:

$ miprograma < archivos_de_datos&

Donde vemos dos cosas: el operador "<", que introduce el nombre del / los archivos de datos y el operador "&" que envía el procesoa correr en background.

Para controlar si el programa está corriendo correctamente, usamos el comando

$ top

“top” (table of processes) muestra una lista de los procesos que están ejecutándose en tiempo real, el usuario que los ha enviado a ejecutar, el tiempo que lleva ejecutándose, la cantidad de CPU que está consumiendo y un número llamado PID (Process ID). Para “matar” un proceso, podemos utilizar, dentro de “top” la tecla “k”, introducir luego el PID del proceso que queremos cancelar y luego la señal “9”. Ningún usuario sin privilegios de administrador puede matar procesos de otros usuarios.
También podemos usar la línea de comandos para el mismo fin

$ kill -9 PID

donde reemplazaremos PID por el número de proceso que hemos leído en “top”.
Para salir de “top” pulsamos la tecla “q”.

4.- Modo ejecución programada
En este modo sí podremos apagar la terminal local, para lo cual usamos un truco: redireccionamos la salida del programa (la pantalla) a un archivo. En adelante llamaremos “mensaje” a ese archivo. Al archivo “mensaje” no lo creamos nosotros. Lo crea el sistema y lo va llenando con la salida que usualmente vemos en pantalla. A veces resulta muy útil ir mirando la evolución de la ejecución de un programa mediante la consulta del archivo “mensaje”.

4.1.- Si “miprograma” no necesita que se le especifique ningún dato de entrada, bastará con crear un pequeño archivo, que llamaremos “corre.txt” (usando el bloc de notas o cualquier editor de texto sin formato). En ese archivo escribiremos la orden tal y como la teclearíamos en la terminal. Supongamos que deseamos correr “miprograma”, que lee el archivo “datos.dat” por defecto. Creamos el archivo “corre.txt” que tendrá simplemente la orden

miprograma > mensaje

Guardamos y cerramos “corre.txt”.

a) Copiamos a la carpeta de destino en el ordenador de cálculo los archivos “corre.txt”, “miprograma” y “datos.dat”.
b) Ejecutamos

   $ at now -f corre.txt

Nótese que usamos un comando llamado “at” que sirve para programar la ajecución diferida de un proceso. Como usamos el modificador “now”, el proceso se empezará a ejecutar inmediatamente. Y el proceso que se ejecutará es aquel que está especificado en el fichero “corre.txt”. Es decir que la orden anterior se traduce “ahora mismo ejecuta lo que pone en corre.txt”. El fichero “mansaje” contendrá lo que “miprograma” normalmente volcaría en pantalla.

4.2- Si “miprograma” necesita que se le especifique un nombre de archivo o alguna otra variable, deberemos crear un segundo fichero auxiliar al que llamaremos arbitrariamente “raiz”. En “raiz” escribiremos lo que usualmente escribiríamos cuando “miprograma” se lanza. Por ejemplo, TRANSIN necesita que le demos la raíz de los archivos de datos (dim.dat, tim.dat, par.dat, etc.). Supongamos que la raíz es “01” (01DIM.DAT, 01TIM.DAT….), entonces nuestro archivo auxiliar “raiz” quedaría como sigue

01

Y nuestro archivo “corre.txt” quedaría:

miprograma < raiz > mensaje

Donde el operador “<” actúa diciéndole a “miprograma” que utilice “raiz” en vez de la entrada por teclado, y el operador “>” actúa diciéndole al sistema que reemplace la salida por pantalla por la salida al archivo “mensaje”.

4.3- El truco también sirve para concatenar procesos, es decir para que al acabar una pasada se ejcute inmediatamente otra. Supongamos que tenemos los archivos de datos “01DIM.DAT, 01TIM.DAT….” y “02DIM.DAT, 02TIM.DAT…” y queremos enviar las dos pasadas una después de la otra.

a) Creamos dos archivos “raiz1” y “raiz2”. Dentro de “raiz1” pondrá

01
y dentro de “raiz2” pondrá
02
…….

b) El archivo “corre.txt” será así:

miprograma < raiz1 > mensaje1
miprograma < raiz2 > mensaje2
……

c) Luego de copiar **todos** los archivos a la carpeta del servidor remoto, ejecutaremos:

$ at now -f corre.txt

y el sistema se encargará de concatenar los procesos.

Para visualizar rápidamente los archivos “mensaje”, podemos utilizar un editor de textos o bien la orden:
$ cat mensaje
que mostrará por pantalla el contenido del fichero.

4.4- Por último veremos cómo correr procesos de manera concatenada en carpetas separadas. Esto es por si el ejecutable “miprograma” sobreescribe (chafa) los archivos de salida. La idea es la misma que en caso anterior, solo que “corre.txt” tiene la forma:

```
mkdir output1
cp miprograma raiz archivos_de_datos1 output1
cd output1
miprograma < raiz1 > mensaje1
cd ..
```

```
mkdir output2
cp miprograma raiz archivos_de_datos2 output2
cd output2
miprograma < raiz2 > mensaje2
cd ..
……..
```

Donde los puntos suspensivos indican que podemos seguir agregando pasadas. Al ejecutar:
$ at now -f corre.txt
ocurre lo siguiente

- Se crea la carpeta output1 (mkdir output1)
- Se copian a esa carpeta el ejecutable y todos lor archivos necesarios para la pasada. (cp miprograma….)
- Se sitúa al sistema en la carpeta “output1” (cd output1)
- Se ejecuta el proceso
- Se vuelve la carpeta inicial (cd ..)
- Se repite el loop, esta vez con los siguientes archivos de datos.

Cualquier duda o pregunta que sirva para aclarar este mini tutorial, por favor escribirme a adolfodante.castro@gmail.com