Tutorial Básico para el Manejo de los Servicios para la Investigación de la DSIT y la Vicerrectoría de Investigación de la Universidad de los Andes Colombia.

Contenido

1. Bash 2
   1.1 Accediendo al Clúster 2
   1.1.1 Sistema operativo Windows 2
      1.1.2 Sistema operativo Linux / iOSX (Mac) 2
   1.2. Directorios 3
      1.2.1 Crear Directorios 4
      1.2.2 Cambiar de directorio 4
   1.3. Archivos 5
      1.3.1 Crear y editar archivos 5
      1.3.2 Listar contenido 5
      1.3.3 Copiar archivos 6
      1.3.4 Mover archivos 7
      1.3.5 Borrar archivos y directorios 7
      1.3.6 Visualizar el contenido de un archivo 7
      1.3.7 Copiar archivos remotamente 8
2. HPC (High-Performance Computing) 9
   2.1 Estructura del clúster 9
   2.2 Solicitar trabajos 9
      2.2.1 Slurm 9
      2.2.2 Módulos 10
      2.2.3 Trabajos 10
      2.2.4 Sistema de colas 11
      2.2.5 Solicitud de recursos 11

1. Bash

1.1 Accediendo al Clúster

Para hacer uso del clúster, se debe establecer una conexión usando el protocolo SSH (Secure SHell) al servidor: **hypatia.uniandes.edu.co**. Esta conexión varía según el sistema operativo desde el que se realizará la conexión:

1.1.1 Sistema operativo Windows

Existen software clientes para Windows que permiten iniciar conexión remota (SSH) y acceder a ambientes Unix desde su escritorio. Entre las más conocidas se encuentra **PuTTY,** que puede descargar del sitio oficial del software http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, también existe el software cliente de ssh **MobaXterm,** que puede descargar del sitio https://mobaxterm.mobatek.net/download.html y finalmente el cliente de transferencia de archivos **FileZilla,** que puede descargar del sitio oficial https://filezilla-project.org/download.php?type=server.

En este tutorial utilizaremos *MobaXterm,* siguiendo las indicaciones dadas a continuación para el ingreso al clúster:

i. En el menú, presione el icono **Session** y seleccione el icono **SSH**. Ingrese la conexión con el nombre: **hypatia.uniandes.edu.co** en el cuadro de texto **Remote Host.** En el cuadro de texto **Specify username** ingrese su usuario de correo de uniandes asignado por la universidad, **sin @uniandes.edu.co**. Asegúrese de que el puerto sea **22**. Presione **OK.**

ii. Ingrese su **contraseña** Uniandes. Aunque no aparecen caracteres cuando la escribe, en realidad está escribiendo su contraseña. Luego presione la tecla "**Enter**".

Ahora hemos accedido al servidor y estamos en nuestro directorio home del ambiente Unix.

1.1.2 Sistema operativo Linux / iOSX (Mac)

En iOSX y en Linux, cuenta con todo lo necesario para comenzar, puede usar SSH desde la terminal, sin necesidad de un software especial. Todo lo que usted necesita para ingresar al servidor, es el software de conexión **SSH** y **nombre del usuario,** presione la tecla "**Enter**" y enseguida ingrese su contraseña Uniandes. Aunque no aparecen caracteres cuando la escribe, en realidad está escribiendo su contraseña

Para realizar la conexión, digite el siguiente comando:

```
$ ssh help.user@hypatia.uniandes.edu.co
```

Sustituya la información ***help.user*** con su nombre de usuario, es decir el usuario de su correo electrónico Uniandes, **sin @uniandes.edu.co**.

`ssh` – es la abreviación de `Secure Shell`, es el nombre de un programa y un protocolo que permite el acceso y la ejecución de comandos en un ordenador remoto. El programa permite que los datos transiten de forma cifrada, ofertando así una mayor seguridad.

Para salir de un ordenador remoto, desconectando la conexión, utilice el comando "`exit`"

```
[help.user @hypatia ~] $ exit
```

Nota: Algunas características importantes para tener en cuenta de la línea de comando del Linux, son:

- El sistema Linux distingue letras mayúsculas y minúsculas. Si el comando a ser ejecutado sea, por ejemplo, `pwd`, los comandos `Pwd`, `PWd`, `PWD`, `PwD`, `pWD`, `pWd` y `pwD` **No** funcionarán.
- Todo comando en Linux debe ser seguido de la digitación de la tecla "`ENTER`".
- El *prompt* del comando varía con el tipo de intérprete de comandos del sistema (bash, csh, etc) y configuración del *shell*. En general, el `bash` presenta el nombre del usuario, el nombre del servidor y el nombre del directorio corriente y, finalmente, un signo de solar (`$`).
  P.ej. `help.user@hypatia proyecto $` - Significa que el usuario `help.user` está logueado en el servidor `hypatia`, y está en el directorio `proyecto`.

1.2. Directorios

Un directorio es un conjunto de archivos. Todos los directorios de un sistema UNIX parten de un directorio principal llamado "***raíz***", que se representa como "***/***". El directorio raíz es la base para todo el árbol de directorios, es allí donde están contenidos todos los directorios del sistema.

Cuando el usuario accede a una sesión, Linux lo ubica en su directorio de trabajo personal (*/home/nombre-usuario*). Allí, tiene la libertad absoluta para hacer lo que requiera con sus archivos y directorios. Sin embargo, no podrá acceder y manipular el directorio de otro usuario, ya que Linux tiene un sistema de permisos que concede o restringe libertades sobre los directorios y ficheros.

NOTA: Es importante destacar que todos los nombres de archivos y comandos son "*case-sensitive*" (hacen diferencia entre mayúsculas y minúsculas).

Una vez, ingresamos con nuestro usuario y contraseña es importante reconocer nuestra ubicación en el sistema para la generación de archivos o movernos a lo largo de nuestra actividad. Para saber dónde estamos ubicados en el sistema ejecutamos el comando "`pwd`":

```
[help.user @hypatia ~] $ pwd
Salida: /hpcfs/home/help.user directorioactual
```

1.2.1 Crear Directorios

El orden de nuestros archivos depende de la metodología de organización deseada por el usuario, por ello **crear y manipular directorios** es importante. Para crear un directorio se utiliza el comando *"*`mkdir`*" seguido por el nombre del directorio que deseamos crear.*

```
[help.user @hypatia ~] $ mkdir prueba
```

donde <*prueba*> es el nombre del directorio que queremos crear.

1.2.2 Cambiar de directorio

Para movernos por la estructura de directorios utilizamos el comando "cd", abreviación de "cambio de directorio".

Como ya vimos, al entrar al sistema comenzamos en el directorio "*~" que hace referencia a nuestro home*. Si queremos cambiarnos al directorio prueba, debemos usar el comando:

```
[help.user @hypatia ~] $ cd prueba/
[help.user @hypatia prueba] $
```

Como podemos ver, el prompt cambia para mostrar el directorio actual de trabajo, ahora ya estamos en el directorio prueba. Para volver al directorio que lo contiene usamos el comando:

```
[help.user @hypatia prueba] $ cd ..
[help.user @hypatia ~] $
```

(Note el espacio entre "cd" y ".."). Cada directorio tiene una entrada de nombre ".." la cual se refiere al directorio que contiene al directorio actual. De igual forma, existe en cada directorio la entrada "." la cual se refiere al directorio mismo en el que nos encontramos. Así que el siguiente comando nos deja donde estamos:

```
[help.user @hypatia prueba] $ cd .
```

Para más información de comandos básicos en Linux, usted puede fijarse en los tutoriales recomendados al final del documento en material de apoyo.

1.3. Archivos

1.3.1 Crear y editar archivos

Para crear un archivo vacío rápidamente, podemos ejecutar el comando "`cat`", "`touch`" o utilizar un editor de texto, como "`nano`", "`vi`" o "`vim`" para crear y editar un archivo.

```
[help.user @hypatia prueba] $ cat > archivo1.txt
[help.user @hypatia prueba] $ touch archivo1.txt
```

Cualquiera de los comandos anteriores crea un archivo vacío llamado `archivo1.txt`

```
[help.user @hypatia prueba] $ nano archivo1.txt
[help.user @hypatia prueba] $ vim archivo1.txt
[help.user @hypatia prueba] $ vi archivo1.txt
```

Cualquiera de los comandos anteriores crea un archivo llamado `archivo1.txt` y lo abre con un editor para manipular su contenido.

1.3.2 Listar contenido

Para ver el contenido de un directorio utilizamos el comando "`ls`"

```
[help.user @hypatia prueba] $ ls
```

El sistema informa la presencia del archivo `archivo1.txt`, creado. vamos a usar el comando "`cat`", ahora para ver el contenido del archivo recién creado:

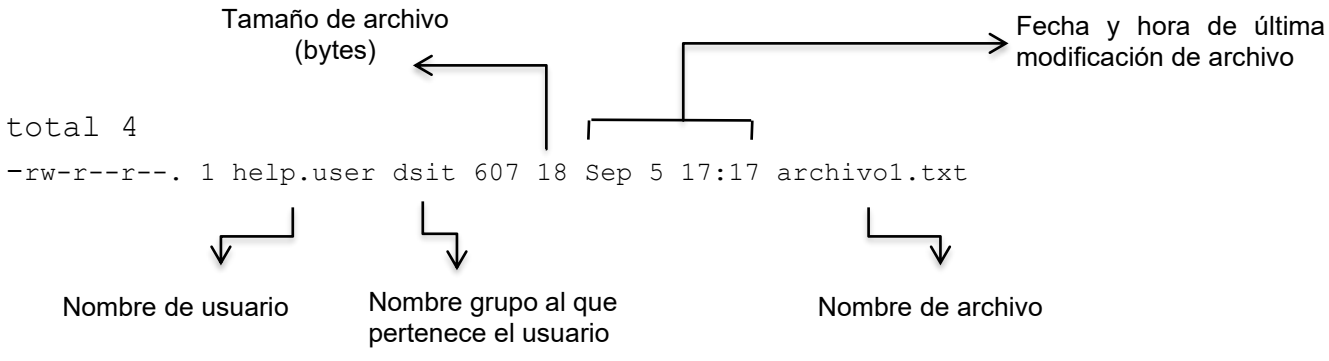
```
[help.user @hypatia prueba] $ cat archivo1.txt
```

Si el archivo tiene algún contenido, este se presentará en pantalla (por ejemplo):

```
Mi primer archivo
```

Para ver un listado más detallado, podemos usar el comando "`ls -l`":

```
[help.user @hypatia prueba] $ ls -l
Salida:
```



Esta información indica que el archivo `archivo1.txt` pertenece al usuario `help.user`, el cual forma parte del grupo `dsit`.

¿Cuál es el significado de las letras antes del nombre del archivo?

Estas letras describen si es un archivo o directorio, y el tipo de permisos que tiene:

```
-rw-r--r-- 1 help.user dsit 18 Sep 5 17:17 archivo.txt
d - Directory
r - Read
w - Write
x – ejecute
```

1.3.3 Copiar archivos

Ahora vamos a crear una copia del archivo `archivo1.txt` creado anteriormente, para esto haremos uso del comando "`cp`":

```
[help.user @hypatia prueba] $ cp archivo1.txt archivo2.txt
```

Teclee el comando `"ls -l"` para confirmar que el archivo haya sido creado:

```
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:17 archivo1.txt
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:44 archivo2.txt
```

Note que el archivo `archivo2.txt`, conforme esperado, tiene exactamente el mismo tamaño en *bytes* que `archivo1.txt`.

Vamos a crear ahora un directorio de nombre `tutorial`. Para eso, teclee el comando: `"mkdir tutorial"`. Si revisamos el contenido del directorio actual con el comando "`ls -l`", verá que ahora existe un directorio `tutorial` (la primera letra `d` indica tratarse de un directorio).

```
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:17 archivo1.txt
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:44 archivo2.txt
drwxr-xr-x 2 help.user dsit 4096 Sep 5 17:45 tutorial
```

Vamos a copiar ahora el archivo `archivo1.txt` en el directorio `tutorial`:

```
[help.user @hypatia prueba] $ cp archivo1.txt tutorial
```

Si revisamos el contenido del directorio actual con el comando "`ls -l tutorial`", podrá ver el listado del directorio `tutorial`:

```
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:46 archivo1.txt
```

1.3.4 Mover archivos

Vamos a mover ahora el archivo `archivo2.txt` para el directorio `tutorial` usando el comando "`mv`". Mientras el comando "`cp`", simplemente copia el archivo deseado manteniendo la copia original, el comando "`mv`" hace el mismo procedimiento, pero borra la copia original, en otras palabras, mueve el archivo.

```
[help.user @hypatia prueba] $ mv archivo2.txt tutorial
```

Revisamos el contenido del directorio actual "`ls -l`". El listado muestra que el archivo `archivo2.txt` ya no se encuentra más en el directorio raíz.

```
-rwxr-xr-- 1 help.user dsit 18 Sep 5 17:17 archivo1.txt
drwxr-xr-x 2 help.user dsit 4096 Sep 5 17:46 tutorial
```

Teclee ahora "`ls -l tutorial`" para ver el contenido de la carpeta.

1.3.5 Borrar archivos y directorios

El comando "`rm`**"** nos permite **borrar** archivos y directorios en Linux, al usar `rm` se eliminará(n) los archivos que indique, pero por defecto, rm no elimina directorios en el sistema; para eliminar un directorio entonces añada el flag "`-r`", al usar esta opción se procede a eliminar los directorios, así como su contenido de forma recursiva, debe usarse de manera responsable pues puede perder información si utiliza ese comando de manera descuidada.

La sintaxis de uso de este comando es:

```
[help.user @hypatia prueba] $ rm -r <directorio>
[help.user @hypatia prueba] $ rm <archivo1> <archivo2> <archivo3>
```

NOTA: Tenga en cuenta que los archivos borrados con la función rm no pueden ser rescatados. En el caso de que añadan a su código el flag "`-i`"**,** esta opción pregunta una vez antes de eliminar más de tres archivos, o bien al eliminar de forma recursiva los archivos. Se recomienda ver la documentación de la función -rm.

1.3.6 Visualizar el contenido de un archivo

Para visualizar el contenido de un archivo en la consola, los comandos más útiles en este sentido son "`cat`"**,** **"**`more`" y "`less`", cada uno con ciertas particularidades, como se especificará a continuación:

- Cat

El comando `cat` es uno de los comandos más utilizados para manejar archivos de texto (.txt). Con `cat` usted puede **visualizar** contenidos de un fichero entero en la consola, en este caso, si el archivo es largo, se desplazará rápidamente y por completo por la consola.

- More

El comando `more` es otro comando útil para **visualizar** por pantalla el contenido de un archivo de texto. Esencialmente es igual que el comando `cat`, con la diferencia que el comando `more` le permitirá paginar el contenido del archivo. Así, primero se mostrarán las líneas que quepan en una pantalla sin hacer scroll. El resto serán accesibles mediante la tecla espacio donde visualizará el porcentaje del total del texto que usted está visualizando.

- Less

El comando `less`, al igual que los comandos `cat` y `more`, le permitirá leer el contenido de un archivo de texto. A diferencia de los otros dos, éste le mostrará el contenido en modo editor de texto, y podrá desplazarse con los controles de desplazamiento del teclado. Otra opción es utilizar la tecla *g* y luego *Enter*. Esto avanzará, por defecto, una línea adelante, pero puedes avanzar cualquier número de líneas que deseas, introduciendo el número justo después de marcar la *g*. Por último, para salir de la visualización, presione la tecla *q*

NOTA En linux usted puede ver el historial de todos los comandos ejecutados en su consola use la orden "`history`"**.**

1.3.7 Copiar archivos remotamente

Para copiar archivos remotamente, es decir, **transferir los archivos** de su computador al clúster, y viceversa, usted debe utilizar el comando **"**`scp`" (secure copy)**.** Para ejecutar la función `scp` es importante tener claros los siguientes parámetros:

- **Usuario:** el nombre de usuario en el servidor.
- **Host:** dominio del servidor remoto.
- **Archivo origen:** ruta del archivo que queremos copiar.
- **Directorio destino:** ruta donde queremos copiar el archivo.

Para copiar archivos del clúster a su computador, ejecute el siguiente comando desde su computador:

```
$ scp usuario@host:Archivo_origen  directorio_destino
```

Ejemplo:

```
$ scp help.user@hypatia.uniandes.edu.co:~/prueba/archivo1.txt  .
```

Para copiar archivos de su computador al clúster, ejecute el siguiente comando desde su computador:

```
$ scp archivo_origen usuario@host:directorio_destino
```

Ejemplo:

```
$ scp archivo.txt help.user@hypatia.uniandes.edu.co:~/prueba
```

2. HPC (High-Performance Computing)

2.1 Estructura del clúster

Hypatia es el servicio de clúster de la Universidad de los Andes. Nuestro sistema está montado en Linux Centos. El manejador de trabajos es SLURM y el sistema de almacenamiento es BeeGFS. EL software está montado y disponible con Módulos.

Al conectarse con su usuario, usted se encuentra en la puerta del clúster (hypatia). Esta puerta es **general** para todos los usuarios con el **objetivo** principal de dar **acceso** al clúster, navegar sus carpetas y **realizar tareas menores** como la creación de archivos de texto y ejecutar comandos de bash. Para realizar cualquier tipo de procesamiento, es adecuado realizarlo en los nodos disponibles, esto con el objetivo de gestionar los recursos para todos los usuarios.

2.2 Solicitar trabajos

2.2.1 Slurm

¿Qué es Slurm? Es uno de los gestores de colas más utilizados (60% del Top500), de libre distribución (gratuito y open source), tiene una gran comunidad y grupo de usuarios. Entre sus características permite una gestión de trabajos completa, ofreciendo calidad de servicio, configurar y categorizar Colas (particiones), límites de recursos por usuario/grupo y reservas dinámicas

Cuando hay más trabajos que nodos hay que establecer un método justo que determine cuál es el próximo trabajo en ser despachado.

Slurm realiza cálculos periódicamente basándose en un algoritmo de gestión de prioridades con varios parámetros configurables (por el administrador) por ejemplo:

- Tamaño del trabajo (número de nodos/cores).

- Duración de los Jobs (se especifica a la hora de encolarlo).
- Edad del trabajo en la cola
- Número de trabajos en cola

Toda la documentación de Slurm, está disponible en https://slurm.schedmd.com

2.2.2 Módulos

Slurm permite cargar librerías mediante módulos, una vez se carga una librería inmediatamente carga sus dependencias, variables de entorno y todo lo necesario para ejecutar la librería.

A continuación, se listan los comandos principales para hacer uso de los módulos:

- `[help.user @hypatia prueba] $ module avail`
- `[help.user @hypatia prueba] $ module load <nombre_modulo>`
- `[help.user @hypatia prueba] $ module list`
- `[help.user @hypatia prueba] $ module unload <nombre_modulo>`
- `[help.user @hypatia prueba] $ module purge`

El primer comando lista los módulos o software disponibles actualmente en el clúster, el segundo carga el módulo indicado, el tercero lista todos los módulos cargados, el cuarto quita el módulo indicado, y el sexto y último comando quita todos los módulos cargados.

Ejemplo para la carga de un módulo:

```
[help.user @hypatia prueba] $ module load R/3.1.2
```

El anterior comando carga las librerías y variables de entorno para ejecutar R 3.1.2.

2.2.3 Trabajos

Para ejecutar procesamiento en el clúster, es necesario enviar trabajos al sistema de colas, allí Slurm asignará un nodo de computación y recursos (como CPU, RAM y tiempo máximo de ejecución), según lo requiera el usuario y de acuerdo con los esquemas de cola.

El conjunto básico de comandos que el usuario requiere utilizar para el sistema de gestión de trabajos son los siguientes:

srun: se utiliza para enviar un trabajo para su ejecución o iniciar los pasos del trabajo en tiempo real. srun tiene una amplia variedad de opciones para especificar los requisitos de recursos.

sbatch: se utiliza para enviar un script de trabajo desatendido (sin que tenga que estar presente). El script generalmente contendrá uno o más parámetros para iniciar tareas paralelas, básicamente los recursos a utilizar.

scancel: se utiliza para cancelar un trabajo en ejecución o en lista de espera para ejecución

squeue: informa el estado de los trabajos o los pasos del trabajo. Tiene una amplia variedad de opciones de filtrado, clasificación y formato. Por defecto, informa los trabajos en ejecución en orden de prioridad y luego los trabajos pendientes en orden de prioridad.

salloc: se usa para asignar recursos para un trabajo en tiempo real. Por lo general, esto se usa para asignar recursos y generar un shell. El shell se utiliza para ejecutar comandos srun para iniciar tareas paralelas.

2.2.4 Sistema de colas

Los trabajos son enviados a un sistema de colas que se asignan dependiendo de los recursos requeridos. La configuración actual se muestra en la siguiente tabla, donde se describe: el nombre de la cola, número de nodos configurados, máximo de nodos por usuario y demás características de cada una de las colas.

Partición (colas de ejecución)	Nodos	Máximo número de nodos por usuario	Memoria predeterminada CPU (Mb)	Máxima memoria CPU (Mb)	Tiempo Máximo	Tiempo predeterminado	Máximo CPUs por usuario	Máxima memoria por usuario (Mb)	Máximo CPUs por trabajo	Máximo GPUs por usuario
bigmem	a-[4-6],i-[1-5]	2	4096	32768	15 días	2 horas	48	578845	48	-
short	todos	16	2048	4096	2 días	2 horas	120	257275	120	-
medium	todos	8	2048	22528	15 días	2 horas	48	368640	48	-
long	a-[1-3],i-[6-10	2	2048	16384	30 días	2 horas	48	262144	48	-
gpu	i-gpu[1,2]	2	4096	16384	15 días	2 horas	32	191776	32	2

2.2.5 Solicitud de recursos

No todos los trabajos necesitan los mismos recursos de computación. Los procesadores, la memoria RAM y el tiempo de ejecución debe ser apropiada para el tipo de trabajo que se desee lograr. Asimismo, dependiendo de los recursos pedidos y el nivel general del uso del clúster, es posible que el tiempo de asignación de estos sea más largo.

Para asignar recursos a un trabajo, Slurm ofrece los siguientes parámetros:

--job-name=<nombre>	Nombre del trabajo
-p <cola>	Cola a usar
-N<número>	Número de nodos requeridos por trabajo
-n<número>	Número de tareas a lanzarse
--cpu-per-task=<número>	Número de CPUs requeridos por tarea
--mem=<MB>	Memoria requerida por nodo
--time=< D-HH:MM:SS >	Tiempo de ejecución
--mem-per-cpu=<MB>	Memoria requerida por CPU asignada
--mail-user=user@mail	Correo electrónico para notificar estado del job
--output=<nombre>	Archivo en donde guardar el output del trabajo

Es importante tener en cuenta que Slurm requiere de manera obligatoria especificar una cola, de no hacerlo asignará el job a la cola "*short*" por defecto.

Para hacer uso de las colas **gpu** es necesario hacer parte del grupo autorizado, esto se solicita a través de soportehpc@uniandes.edu.co, e indicar el flag "`--account=gpu`"

Hay dos maneras en que podemos trabajar con slurm:

i) Creando una sesión interactiva

ii) Ejecutar trabajos de manera desatendida.

Sesión Interactiva

Una sesión interactiva permite al usuario configurar y comprobar la ejecución de un scipt o software en particular, de manera que siempre podrá ver el resultado de la ejecución en su momento, en otras palabras, requiere de la presencia del usuario frente al computador logueado. Para trabajos cortos es una buena idea usar una sesión interactiva. Esta es creada usando los comandos de SLURM: **srun** o **salloc,** siempre especificando los parámetros de los recursos a utilizar, de otra manera se le asignarán los recursos mínimos.

La diferencia entre las dos es que salloc dispone o reserva primero los recursos pedidos en un nodo, por lo que es necesario conectarse a éste por medio de ssh:

```
[help.user@hypatia ~]$ salloc -n 1 --mem=1000

[help.user@hypatia ~]$ ssh node-19

[help.user@node-19 ~]$
```

Por el contrario, el comando srun inmediatamente nos conecta con el nodo dispuesto para la sesión interactiva.

```
[help.user@hypatia ~]$ srun -n 1 --mem=1000 --pty bash

[help.user@node-19 ~]$
```

En caso de requerir recursos específicos, la sintaxis es:

```
[help.user@hypatia ~]$ srun -N 2 –ntasks-per-node=17 --pty bash
```

La opción **--pty** es importante. Esto proporciona una solicitud de inicio de sesión en los nodos de proceso que asemeja a una sesión normal del terminal. Nótese que ahora su ubicación le dice en qué nodo está trabajando

Trabajos automatizados

Usando un editor de texto es posible crear un archivo, conocido como script, con instrucciones para ejecutar un cálculo de forma automática. Un script es un archivo de texto plano con una secuencia de instrucciones que los ejecuta de una manera ordenada.

Se compone de un encabezado y un cuerpo.

Primero usamos un editor de texto para crear el archivo

```
[help.user@hypatia ~]$ vi script_name.sh
```

En el encabezado primero se determina en qué lenguaje se quiere interprete el script

```
#!/bin/bash
```

Después se estipula los recursos que se piensa utilizar. Si no se especifica el valor, se tomarán valores por defecto. Para cada línea es necesario empezar con #SBATCH:

```
#SBATCH --job-name=test
#SBATCH -p short
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2048
#SBATCH --time=24:00:00
#SBATCH -o test.%o
```

En el cuerpo del script se escriben los comandos que se quiere ejecutar. El script ejecutará estas instrucciones en el orden en que se encuentren.

Nota: Disponemos del siguiente job de ejemplo en: /hpcfs/shared/README/testcpu.sh, el cual puede copiar en su home y editarlo para que realice lo que necesita.

Si se desea utilizar los programas disponibles en el clúster es necesario utilizar la herramienta de módulos para cargarlos y utilizarlos.