

# Tutorial de PPSS (Parallel Processing Shell Script)

2012

| P | P | S | S |



# Contenido

**Objetivo** .....2

**Introducción** .....2

**PPSS distribuidos** .....3

**Práctica** .....4

**Instalando PPSS** .....4

**Ejercicio** .....5

**Bibliografía** .....9

## Tutorial introductorio de PPSS (Parallel Processing Shell Script)

**Objetivo:** Conocer PPSS (Parallel Processing Shell Script) así como poder comparar las características con un proceso serial.

### Introducción:

PPSS es un script de bash shell que ejecuta los comandos, scripts o programas en paralelo. Está diseñado para hacer un uso completo de las actuales CPUs multi-core.

Detecta un número de CPUs disponibles e inicia trabajo separados por cada núcleo del CPU.

PPSS se puede ejecutar en varios sistemas, el procesamiento de un solo grupo de artículos, como un clúster.

PPSS tendrá una lista de elementos como entrada. Los elementos pueden ser archivos de un directorio o de las entradas en un archivo de texto. PPSS ejecuta un comando especificado por el usuario para cada elemento de la lista. El artículo está suministrado como argumento para este comando. En cualquier punto en el tiempo, nunca hay más artículos procesados en paralelo, ya que hay núcleos disponibles.

Al escribir PPSS desde la terminal de comandos Linux ésta le proporciona ejemplos que harán que esté claro cómo se utiliza:

```
pc@ubuntu:~$ ppss
|P|P|S|S| Distributed Parallel Processing Shell Script 2.97
usage: /usr/local/bin/ppss [[ -d < sourcedir > | -f < sourcefile > ]] [[ -c '<command> "$ITEM"' ]]
      [[ -C < configfile > ]] [[ -j ]] [[ -l < logfile > ]] [[ -p < # jobs > ]]
      [[ -q ]] [[ -D < delay > ]] [[ -h ]] [[ --help ]] [[ -r ]] [[ --daemon ]]
Examples:
      /usr/local/bin/ppss -d /dir/with/some/files -c 'gzip '
      /usr/local/bin/ppss -d /dir/with/some/files -c 'cp "$ITEM" /tmp
      ' -p 2
      /usr/local/bin/ppss -f <file> -c 'wget -q -P /destination/directory "$ITEM"' -p 10
```

Como se puede ver, una gran cantidad de información se registra por PPSS sobre el elemento procesado, incluyendo el tiempo que se tardó en procesar. Es de particular interés la línea de estado: se basa en el estado de salida del comando ejecutado, por lo que la detección de errores es constante.

Este programa se ejecuta en script y esto se construye con el objetivo de ser muy fácil de usar. Se ejecuta en Linux y Mac OS X. Debería funcionar en otros sistemas operativos tipo Unix, como Solaris, que apoyan el shell Bash.

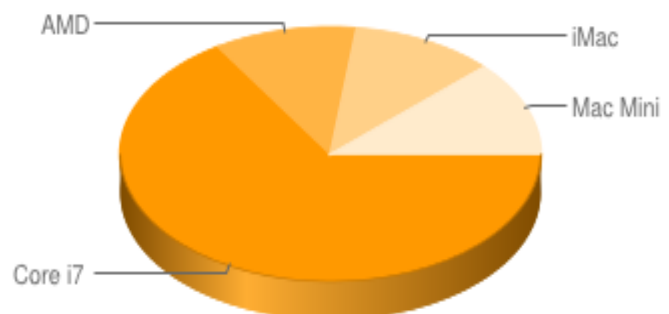
Este script es (sólo) útil para trabajos que pueden ser fácilmente clasificadas en distintas tareas que se pueden ejecutar en paralelo. Por ejemplo, la codificación de muchos archivos wav a formato mp3, descargar un gran número de archivos, tamaño de las imágenes, todo lo que se pueda imaginar.

Tenga en cuenta que este script es *útil incluso en una máquina de un solo núcleo*. Ciertos trabajos, tales como la descarga de archivos y el procesamiento de estos archivos descargados a menudo se pueden optimizar mediante la ejecución de estos procesos en paralelo.

### PPSS distribuidos

A partir de la versión 2.0 en adelante, PPSS soporta la computación distribuida. Con esta versión, es posible ejecutar en PPSS de hos múltiple esto quiere decir que cada proceso de una parte de la misma cola de elementos. Los nodos se comunican entre sí a través de un único servidor de SSH.

Este guion ya ha sido utilizado para convertir 400 GB de archivos WAV a MP3 con 4 anfitriones, un Core i7 con Ubuntu, dos Macs basados en 1,8 y 2 GHz Core Duo corriendo Leopard, y un 2,2 Ghz AMD funcionando con el sistema Debian.



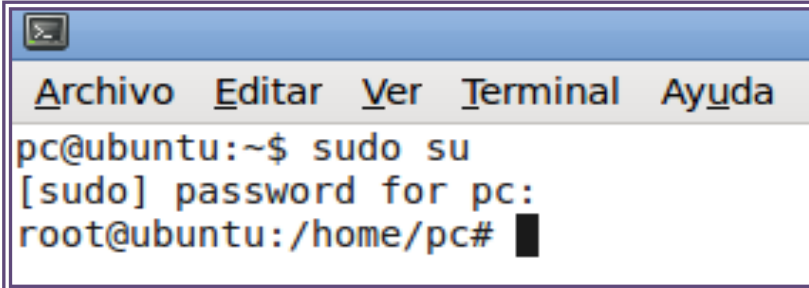
Lo notable es que el Core 7i a 3,6 Ghz procesa 380 archivos, mientras que los otros tres sistemas *combinados* sólo procesan 199. Tenga en cuenta que el Core 7i solo tiene 4 núcleos físicos.

A continuación se explicará paso a paso con un ejemplo el uso de PPSS y su diferencia con su solución serial.

### Práctica:

#### Instalando PPSS:

- Lo primero que se debe de hacer para poder instalar el PPSS en Linux, es identificarse como usuario root



```
Archivo  Editar  Ver  Terminal  Ayuda
pc@ubuntu:~$ sudo su
[sudo] password for pc:
root@ubuntu:/home/pc#
```

Se recomienda tener el archivo en la dirección /usr/local/

- Se descomprimirá con los siguientes comandos para otorgarle algunos permisos:

```
root@ubuntu:/usr/local/bin# tar -xvzf ppss-2.97.tgz -C /usr/local/bin/
ppss
root@ubuntu:/usr/local/bin#
root@ubuntu:/usr/local/bin# chown root:root /usr/local/bin/ppss ; chmod a+rx /usr/local/bin/ppss
root@ubuntu:/usr/local/bin#
```

- Ejecutamos el comando PPSS, para asegurarnos que ya ha sido instalado

```

root@ubuntu:/usr/local/bin# ls
packettracer ppss ppss-2.97.tgz schedulebuilder
root@ubuntu:/usr/local/bin# cd ppss
bash: cd: ppss: No es un directorio
root@ubuntu:/usr/local/bin# ppss

|P|P|S|S| Distributed Parallel Processing Shell Script 2.97

usage: /usr/local/bin/ppss [[ -d <sourceidir> | -f <sourcefile> ]] [[ -c '<command> "$ITEM"' ]]
      [[ -C <configfile> ]] [[ -j ]] [[ -l <logfile> ]] [[ -p <# jobs> ]]
      [[ -q ]] [[ -D <delay> ]] [[ -h ]] [[ --help ]] [[ -r ]] [[ --daemon ]]

Examples:
      /usr/local/bin/ppss -d /dir/with/some/files -c 'gzip '
      /usr/local/bin/ppss -d /dir/with/some/files -c 'cp "$ITEM" /tmp' -p 2
      /usr/local/bin/ppss -f <file> -c 'wget -q -P /destination/directory "$ITEM"' -p 10

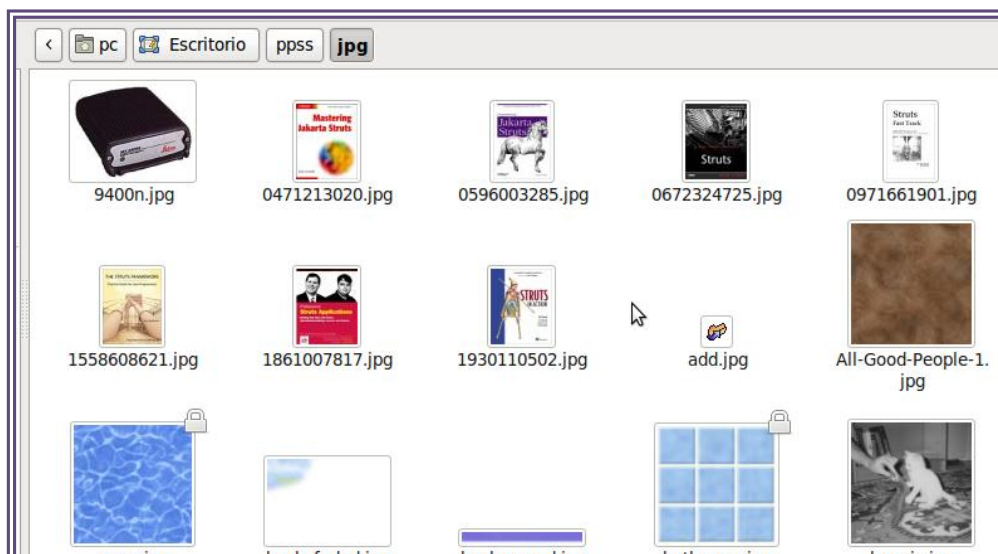
root@ubuntu:/usr/local/bin# █
    
```

Una vez instalado podremos crear algunos scrips para probar el compilador. Para crearlos sugerimos, crear una carpeta en un directorio independiente.

**Ejercicio:**

Primero lo que haremos es hacer un scrip que convierta de imágenes con extensión .jpg a .tiff, ésto de forma serial, para después usar el compilador PPSS que nos hará la misma tarea de forma paralela, y así comparar la eficiencia y tiempo de ejecución.

- Para este ejercicio, tendremos que crear una carpeta llamada “jpg” en el directorio /home/pc/Escritorio/ppss
- Dentro de esta carpeta debe de ir como mínimo 50 imágenes con extensión .jpg para ver el tiempo y compararlo al hacer esto mismo de forma paralela.



- Dentro de la carpeta de jpg creamos el siguiente scrip:

```
1 #!/bin/bash
2 clear
3
4 find / -name *.jpg > /home/pc/Escritorio/ppss/jpg/archivo.txt
5
6 VAR=`cat archivo.txt`
7
8 for a in $VAR;
9 do
10     cp $VAR /home/pc/Escritorio/ppss/jpg
11
12 done
13 echo
```

- Este .sh generara un archivo con las direcciones en forma de texto de cada imagen, que más adelante el archivo lo usaremos, para la conversión de las imágenes a .tiff.
- Ejecutamos el scrip y nos aseguramos que aparezca el archivo dentro de la carpeta “jpg”.

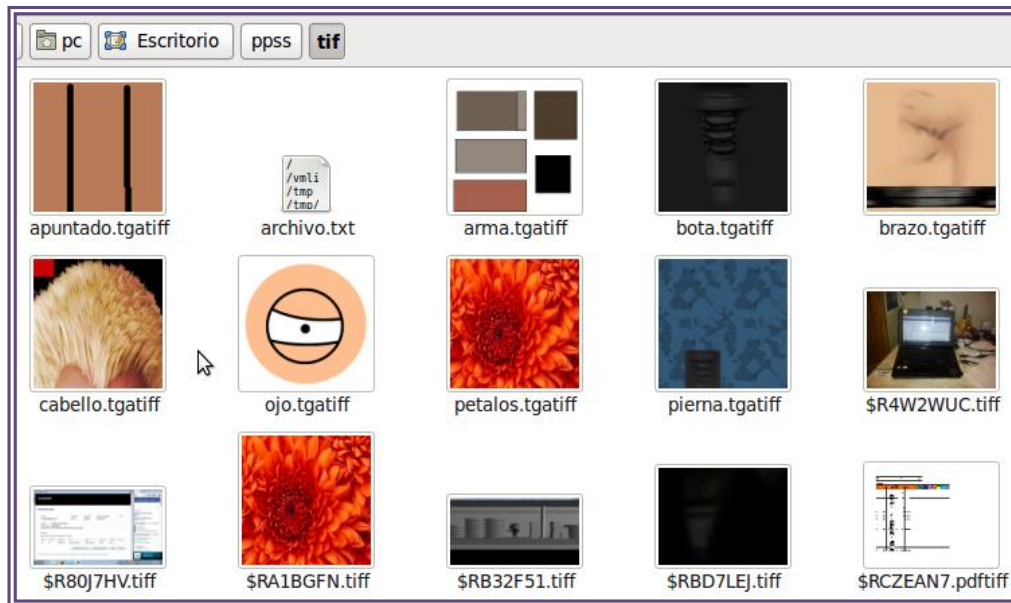
```
find: «/proc/2654/task/2654/fdinfo»: Permiso denegado
find: «/proc/2654/fd»: Permiso denegado
find: «/proc/2654/fdinfo»: Permiso denegado
find: «/proc/2668/task/2668/fd»: Permiso denegado
find: «/proc/2668/task/2668/fdinfo»: Permiso denegado
find: «/proc/2668/fd»: Permiso denegado
find: «/proc/2668/fdinfo»: Permiso denegado
find: «/proc/2683/task/2683/fd»: Permiso denegado
find: «/proc/2683/task/2683/fdinfo»: Permiso denegado
find: «/proc/2683/fd»: Permiso denegado
find: «/proc/2683/fdinfo»: Permiso denegado
find: «/var/run/cups/certs»: Permiso denegado
find: «/var/run/gdm»: Permiso denegado
```

- Posteriormente realizamos el siguiente scrip dentro del directorio */home/pc/Escritorio/ppss*

```

1 #!/bin/bash
2 clear
3
4 mkdir /home/pc/Escritorio/ppss/tif
5
6 cd tif
7
8 cp /home/pc/Escritorio/ppss/jpg/archivo.txt /home/pc/Escritorio/ppss/tif
9
10 VAR=`cat archivo.txt`
11
12 for a in $VAR;
13 do
14     convert -compress lzw "$a" "`basename "$a" jpg`tiff"
15
16 done
17 echo
    
```

Lo ejecutamos en una terminal, y vemos los cambios dentro de la carpeta “tiff” que se creó en el directorio `/home/pc/Escritorio/ppss`



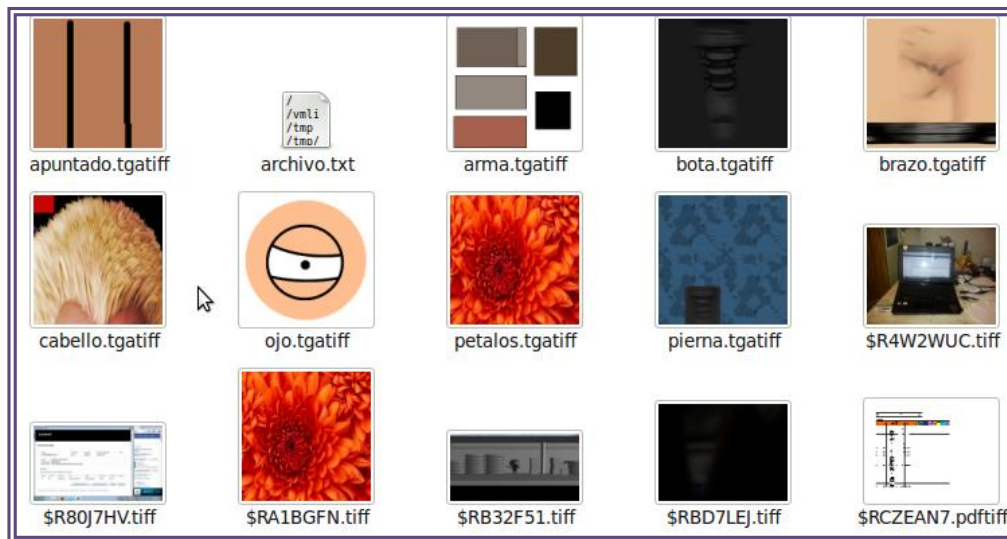
Hasta ahora sabemos como se convierte de forma serial las imágenes jpg a tiff ; ahora lo haremos con el compilador PPSS de forma paralela y observaremos la rapidez con lo que lo hace.



☞ Abrimos una terminal e ingresamos el siguiente comando y lo ejecutamos :

```
pc@ubuntu:~/Escritorio/ppss$ ppss -d /home/pc/Escritorio/ppss/jpg/ -c 'BASENAME=$(basename "$ITEM".jpg); c
onvert -compress lzm "$ITEM" "$BASENAME.tiff"'
ago 03 12:11:37:
ago 03 12:11:38: =====
ago 03 12:11:38: |P|P|S|S|
ago 03 12:11:38: Distributed Parallel Processing Shell Script vers. 2.97
ago 03 12:11:38: =====
ago 03 12:11:38: Hostname:          ubuntu
ago 03 12:11:38: -----
ago 03 12:11:38: CPU: Intel(R) Atom(TM) CPU N455 @ 1.66GHz
ago 03 12:11:38: Found 2 logic processors.
ago 03 12:11:38: Starting 2 parallel workers.
ago 03 12:11:38: -----
ago 03 12:11:38: 0% complete. Processed 1 of 284. Failed 0/284.
ago 03 12:11:39: Total processing time (hh:mm:ss): 00:00:01
ago 03 12:11:39: 0 failed items.
ago 03 12:11:39: Finished. Consult ppss dir/job log for job output.
```

Podemos observar el mismo resultado en diferentes tiempos de procesamiento.



## Bibliografía

<http://code.google.com/p/ppss/>