### Ejercicios

Bajar el dataset de Hits Informativos del Gobierno de la Ciudad de México en la página de Datos Abiertos: `http://datosabiertos.df.gob.mx/sigdata/index.php/Publicacion/busquedaxtema/4`

1. Revisar el archivo. ¿Son únicamente caracteres ASCII?
2. Generar un archivo Hits_ultimos_6_meses con los datos de los últimos seis meses, quitando los caracteres raros
3. ¿Cuántos programas distintos recibieron hits en ese periodo?
4. ¿Hay forma de obtener el programa con más hits (suma de todos los días)?
5. ¿Cómo puedo generar un archivo por cada año, con extensión .csv?

## En Paralelo

Lo visto en la sección anterior presenta una desventaja: El flujo es lineal (sin contar la opción de correr varios programas en background). ¿Qué pasa si se tienen miles de archivos a los que se les tiene que aplicar el mismo proceso de limpieza?
La respuesta: GNU Parallel. (`http://www.gnu.org/software/parallel/`).
Características:

- Permite la ejecución en paralelo de un programa (duh!)
- Levanta workers que ejecutan cada uno una tarea
- Usa todos los cores de la computadora! (Esto también puede ser un riesgo)
- Puede ejecutar en varias máquinas a la vez
- Puede ser un solo comando, o bien ejecutar un pequeño script para cada tarea
- La salida de los comandos ejecutados es la misma que si se ejecutasen secuencialmente, por lo que estas salidas pueden utilizarse sin problemas por otros programas

La sintaxis básica es la siguiente:

Listing 1: Sintaxis de GNU Parallel

```
#Puede ejecutar comandos con:
#parallel [options] [command [arguments]] <
    list_of_arguments
#o bien utilizando datos de linea de comandos:
#parallel [options] [command [arguments]] ( ::: 
    arguments | :::: argfile(s) )
...
parallel echo ::: Hola Mundo
parallel echo ::: Hola Adios ::: Ana Jorge Raul
#Tambien se le puede pasar datos con pipes:
ls | parallel head -3 {}
```

Un tutorial corto con el resto de las opciones se encuentra en: `https://www.gnu.org/software/parallel/parallel_tutorial.html`

### Ejercicios

Usando los datasets generados por año a partir de la base Hits Informativos de Datos Abiertos CDMX:

1. Medir todas las actividades siguientes con date;comando;date; vs. sus contrapartes lineales
2. Quitar las columnas a todos
3. Quitar la extensión que tengan y ponerles .txt
4. Traer el máximo para cada año
5. ¿Hay forma de calcular el total de Hits para cada año?
6. En una sola línea de comando, para todos los años, sustituir el separador de ', 'a '| ', quitarle los caracteres extraños y guardar en un archivo hits.psv
7. ¿Hay forma de mantener el orden en los años? (sin perder la ventaja del paralelismo, obviamente)

# Manejo de Datos

Ya que se tiene la confianza en que los archivos de entrada están limpios y en un formato conveniente para su uso, veremos una breve introducción a la carga y manejo de datos en una base de datos convencional.

## ETL en PostgreSQL

La base a utilizar es PostgreSQL. Una base de datos relacional, Open Source y bastante poderosa. Se puede bajar e instalar siguiendo las instrucciones en `http://www.postgresql.org/download/`[6].
Una vez instalado[7], se puede acceder con el usuario por default:

```
sudo -u postgres psql postgres
```

Una vez adentro, se puede cambiar la contraseña por default del usuario postgres:

```
/*Cambio de Password*/
\password postgres
```

Posteriormente se creará una base y se cargarán los datos[8]:

```
/*Creacion de la base de datos*/
\password postgres
```

A continuación, veremos cómo crear una base de datos y una tabla y se subirán los datos de Hits Informativo:

```
/*Creacion de la base de datos con encoding UTF8*/
create database cdmx encoding='utf-8';
\connect cdmx;
```

[6]Es recomendable estudiar la documentación para referencias específicas: (`http://www.postgresql.org/docs/9.4/interactive/index.html`).

[7]Aunque en este curso no se tocó el tema, la instalación más recomendada en linux es compilar PostgreSQL, definir correctamente el espacio de almacenamiento, y configurar correctamente el usuario postgres desde el sistema operativo.

[8]El lenguaje utilizado es SQL. Si se requiere revisar algún concepto, PostgreSQL ofrece un apartado en su documentación: `http://www.postgresql.org/docs/9.4/interactive/sql.html`

```
--Creacion de dos schemas: upload y datoscdmx
create schema upload;
create schema datoscdmx;
--Creacion de tabla del schema upload
create table upload.datos (hits varchar(20),concepto
    text,fecha varchar(20));
--Validamos:
\l
\dn
```

En el apartado anterior se crearon la base y el schema. En Postgres, se pueden tener varias bases de datos, y cada base puede tener varios schemas. Y en los schemas es donde se alojan las tablas. Al final se validó listando las bases de datos con `\l` y los schemas con `\dn`. Este tipo de instrucciones (que no forman parte de SQL), permiten ejecutar instrucciones generales, de consultas, de entrada / salida e informativas. Para una lista completa, ver `\?`.
*NOTA:Para no teclear siempre schema.tabla, se puede establecer en el path del schema, el orden en el cual se va a buscar una tabla en los schemas existentes.*
Ahora se va a cargar la información a la base. Al momento de cargar, hay que considerar:

- qué tanto tiempo nos puede lleva la carga de grandes volúmenes de datos, y
- qué tantos cambios / limpieza tenemos que realizar con los mismos

. En muchos casos, la velocidad de la red es un factor determinante para el tiempo de carga. Por tanto, nos importa subir los datos lo más rápido posible y procesarlos en la base. Y generalmente, el engine de la base será lo suficientemente rápido para procesar los datos adquiridos, por lo que se puede aprovechar la velocidad para subir una versión de los datos *en sucio* tal cual se encuentran a un esquema separado, y de ahí importar los datos a otro esquema, siendo *limpios* mediante consultas.
Por esta razón se creó la tabla *datos* en el esquema *upload*. Nótese que son únicamente campos de caracteres (incluidos los hits y la fecha) y el concepto es de tipo text, para garantizar que se suba todo lo que trae el archivo. Adicionalmente, la base se creó desde un inicio con encoding UTF-8, para soportar bastantes caracteres adicionales a los estándar en ASCII.

```
--Importacion de los datos
copy upload.datos from
    '<directorio>/Hits_Informativos....csv' delimiter
    ','
csv;
```