Revista Boliviana de Física **14**, 131–138 (2008)

EL USO DEL XML COMO TRADUCTOR ENTRE EL ASCII Y EL Excel

N. Martinic[†], F. Osco[‡]

[†]*Instituto de Investigaciones Físicas, UMSA, La Paz*
[‡]*Visión Asociados, SRL, La Paz*

RESUMEN

Este trabajo tiene por objeto el escribir un código apropiado para traducir archivos en *ascii* –típicos en banco de datos con un número muy grande de archivos, hacia el *Excel*, que es una hoja electrónica que sirve para escribir y representar columnas de datos con gran flexibilidad para hacer cálculos sencillos y gráficos por parte de usuarios de un gran espectro, desde estudiantes de colegio hasta investigadores. Como quiera que desde el punto de vista didáctico no es conveniente hablar en abstracto, se hace la explicación de la traducción de columnas en *ascii* de variables meteorológicas registradas a partir de datos crudos de cientos de estaciones en Bolivia, hacia hojas electrónicas en *Excel* mediante el uso del lenguaje C y del dialecto XML.

Descriptores: bases de datos — métodos computacionales — meteorología

ABSTRACT

The purpose of this work is to write a code that translates files in *ascii* –typical in data banks with a large number of files– to *Excel*, a versatile electronic spreadsheet for recording and representing columns of data used to carry out simple calculations, and present graphics by many users, from high school students to professional researchers. To provide a didactic explanation of our work, a concrete example is used to explain (demonstrate) the code translator; data is taken from hundreds of meteorological stations in Bolivia and converted to *Excel* using C language and dialect XML.

Key words: data bases — computer methods — meteorology

1. INTRODUCCIÓN

El banco de datos de temperaturas y precipitación pluvial de Bolivia generado por N. Martinic et. al [1] consta de más de tres mil archivos, todos en *ascii*. Se trata de archivos bajo tres subtítulos a saber, *datos diarios, datos mensuales y datos anuales*. Donde aparecen archivos ya sea crudos, en hojas electrónicas en *ascii* tal como fueran tomados por los técnicos meteorológicos, o bien los archivos en *ascii* deducidos de aquellos mediante programas en C que también se encuentran a disposición de los usuarios en directorios apropiados.

Los datos diarios, tanto de temperatura como de precipitación pluvial, exhiben valores de la temperatura media, así como de las autocorrelaciones para cada estación meteorológica, los espectros de potencia y gráficos en diferentes formatos, siendo el más popular los documentos en PDF. En dichos gráficos se encuentran además de los archivos indicados ajustes armónicos y ubicación de las estaciones en un mapa de Bolivia.

En la sección 2 se ilustra una discusión somera de los datos diarios de estaciones del Altiplano boliviano con el objeto de familiarizar al lector con la textura de datos que se desean traducir en *Excel*. En la sección 3 se hace una introducción del dialecto *xml* que sirve para traducir automáticamente archivos *ascii* –con formato de columnas, hacia las hojas

TABLA 1

Fragmento del formato para datos de temperatura para la estación de Ayo Ayo. La primera fila posee cuatro valores enteros, los dos primeros son en grados y minutos *geográficos*, la latitud *sur* de la ubicación de la estación, mientras que los dos segundos corresponden a las mismas coordenadas, longitud *oeste* de dicha estación. El quinto valor es la altura, en *m* de Ayo Ayo. Las siguientes filas son, respectivamente, el año, el mes, y las *31* temperaturas en grados centígrados del mes correspondiente. Está claro que cuando dicho mes no posee los *31* días, entonces se llena de valores 999.0. Lo propio ocurre cuando, por razones de fuerza mayor, no se registra un día determinado, se llena con el valor indicado.

17	5	68	0	3856			
73	1	1.0	1.0	3.2	2.5	2.2	...
73	2	3.7	3.9	4.9	3.6	4.0	...
73	3	1.3	...	...	...	...	...

electrónicas que son reconocidas por el entorno del *Excel*. En la sección 4 se indica, mediante un algoritmo positivista los pasos necesarios para –gracias a un programa en C, hacer posible la traducción de una manera inmediata.

Finalmente, en la sección 5 se hace una discusión apropiada en el contexto de plataformas híbridas con el objeto de hacer posible la lectura de miles de datos en formato de columnas a un público relativamente joven que desea "leer" los archivos *ascii* de columnas de datos mediante paquetes con un gran contenido didáctico, como es el *Excel*.

2. DATOS DIARIOS DE TEMPERATURA Y PRECIPITACIÓN PLUVIAL

Se encuentran en los directorios[1] *banco/daily/data/* donde, en archivos *ascii* legibles ya sea con cualquier editor convencional (*wordpad* para la plataforma *Microsoft* por ejemplo) se exhibe las hojas de datos tales como son registradas por los técnicos en el campo. El formato de estos datos crudos se exhibe en la tabla 1, que es un ejemplo para la estación de Ayo Ayo para datos de temperatura.

Las estaciones que se hallan registradas en este banco de datos, se exhiben en el mapa de Bolivia en la figura 1. A la izquierda los datos de temperatura, mientras que a la derecha, los datos de precipitación pluvial. Las unidades son $^\circ C$ y $mm\,d$, respectivamente.

2.1. *Las series temporales deducidas*

En los directorios */banco/daily/prog/* se puede encontrar todo el *software* para reproducir los archivos de este banco de datos diarios. Por un lado, los programas en lenguaje C son relativamente fáciles de comprender con un conocimiento superficial de dicho lenguaje. Para aquellos que tienen un conocimiento más profundo de la plataforma UNIX se suministran *shells* que ejecutan no sólo estos programas sino que hacen uso de los datos crudos y obtienen en forma inmediata todos los archivos obtenidos en los directorios *banco/daily/outdata/*, *banco/daily/figura/* y *banco/daily/xml/*. Se habla en plural debido a que lo que se explica aquí sirve no sólo para los datos de muestreo diario, sino que también para otros muestreos del presente banco de datos, a saber *banco/monthly/... y banco/yearly/...* que no se discute en esta introducción didáctica. El esquema mostrado en la figura 2 ilustra en un diagrama de bloques el *modus operandi* de estas operaciones del software.

Aquí vale la pena una digresión con respecto a la identificación o etiqueta de cada archivo. Ya que se trata de un banco de datos con miles de archivos, es oportuno el definir la lógica de las etiquetas de cada archivo. Por un lado, el nombre de la estación se encuentra a través de un acróstico que no es difícil de conjeturar correctamente, sobre todo para alguien que tiene experiencia de vida en Bolivia[2]. Por otro lado, al final del acróstico aparece un número ya sea *1=temperatura media*, o bien *2=precipitación pluvial*. Las unidades en un caso son grados centígrados por lo que es imposible que hubiesen valores por encima de $50\,^\circ C$ mientras que para el otro caso es en $mm\,d$. Esta manera de etiquetar para los valores diarios puede no ser del agrado de la mayoría, pero, una vez tomada la decisión de así hacerlo, se ha deseado conservar el código. Las salidas, tanto en archivo como en gráfico, se explican detalladamente en las subsecciones 2.2, 2.3, 2.4 y 2.5.

[1] Existe un CD que acompaña al documento [1], donde en la raíz se encuentra el directorio *banco/*

[2] En una segunda edición de este documento se tratará de presentar tablas de traducción del acróstico con los nombres oficiales de cada estación.

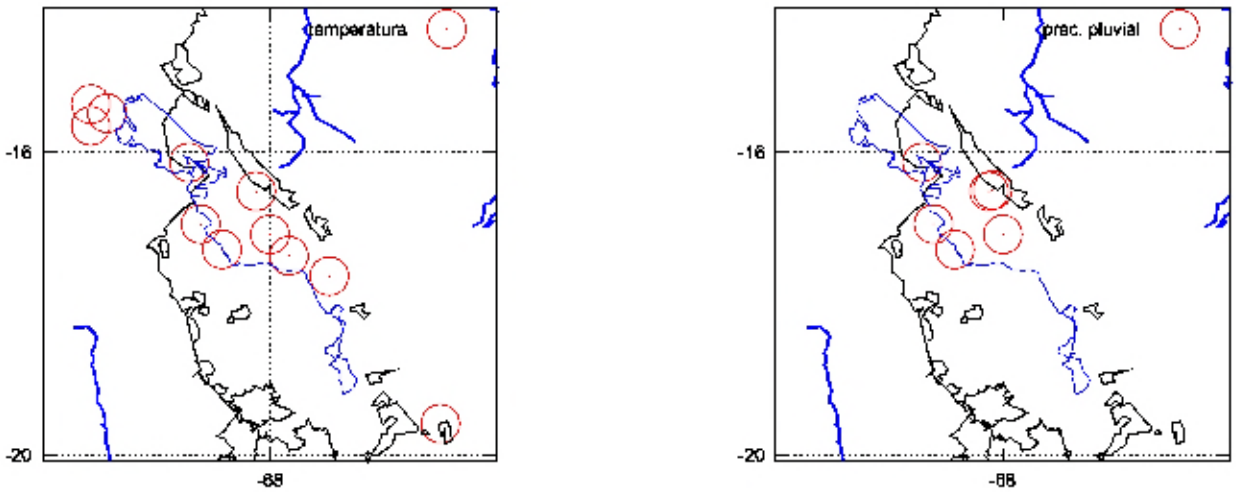


Figura 1. A la izquierda las estaciones en el Altiplano norte que registran las temperaturas medias diarias, mientras que a la derecha las de las precipitaciones pluviales. Además de la costa del Pacífico se puede reconocer el contorno de los grandes lagos del Altiplano. Las otas curvas son los cortes de las cordilleras a una altura de más de 4 km sobre el nivel del mar.

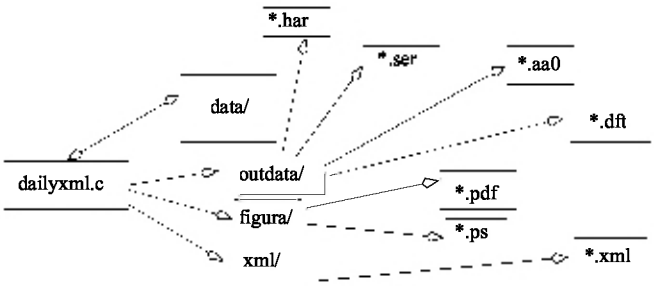


Figura 2. Un análisis de bloques que explica el flujo de la información de acuerdo a las flechas. Los datos crudos se encuentran en el directorio *banco/daily/data/* que el programa lee y produce con esa información las series temporales que se identifican de acuerdo a su extensión. Todos los gráficos se encuentran en el directorio *banco/daily/figura/* mientras que los resultados en *Excel* se encuentra en el directorio *banco/daily/xml/*.

2.2. *La serie temporal (*.ser)*

Se trata de datos en tres columnas, la primera es la secuencia en días entre 1 y 365/366 dependiendo del tipo de año, ya sea normal o bien bisiesto. La segunda columna se encuentra en fracciones del año correspondiente, mientras que la última son los valores de la temperatura media de la estación en cuestión. En la figura 3 se ilustra un ejemplo para la estación de Potosí.

2.3. *Promedios anuales (*.aa0)*

Se obtienen superponiendo los datos diarios durante todo el periodo de registros para una estación y dividiendo sobre el número de temperaturas registradas. Estos archivos poseen tres columnas, la primera corresponde a los días promedios del año, la segunda a los valores promedios menos el promedio de todos los años. En la figura 4 se exhiben los valores promedios obtenidos mediante el archivo de Potosí.

2.4. *Superposiciones anuales (*.har)*

Para poder superponer –sin efectuar promedios totales a lo largo de un año promedio, todos los datos ya sea de la temperatura o bien de la precipitación pluvial, sobre un promedio nulo o bien sobre un promedio real (que no es aconsejable en virtud que los promedios anuales año tras año no son los mismos, evidentemente, obteniéndose colecciones de curvas anuales más o menos dispersas) se pueden utilizar los archivos con extensión *har*. Estos archivos poseen cuatro columnas, en la primera se escribe el año a lo largo de los dias de cada año, que están escritos en la segunda columna. La tercera y cuarta corresponden a los valores reales o bien a los valores superpuestos sobre un cero equivalente al promedio común a todos los años. Este tipo de gráfico se ilustra en las salidas de los directorios *banco/daily/grafica/* donde ilustran sólo las variaciones armónicas. Asimismo, en estos mismos gráficos, que se repiten en la figura 5 para la localidad de Potosí, se ha acomodado un ajuste armónico con cuatro armónicos además de la salida de la ilustración 4 de más arriba. Obsérvese que los datos reales tomados sobre un promedio anual poseen

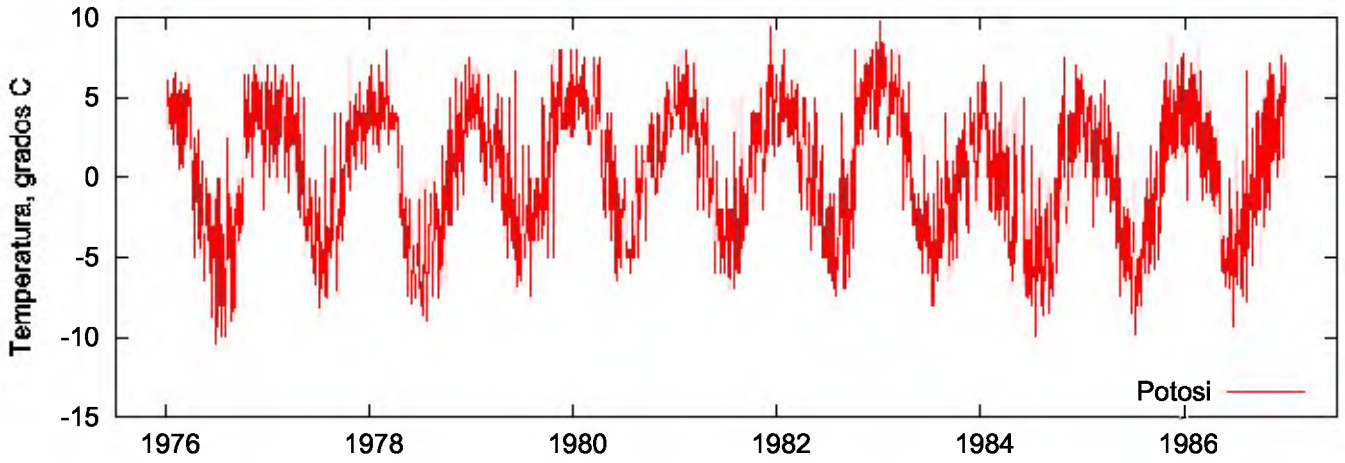


Figura 3. La serie temporal *potosi1.ser* que exhibe la temperatura media diaria de Potosí.

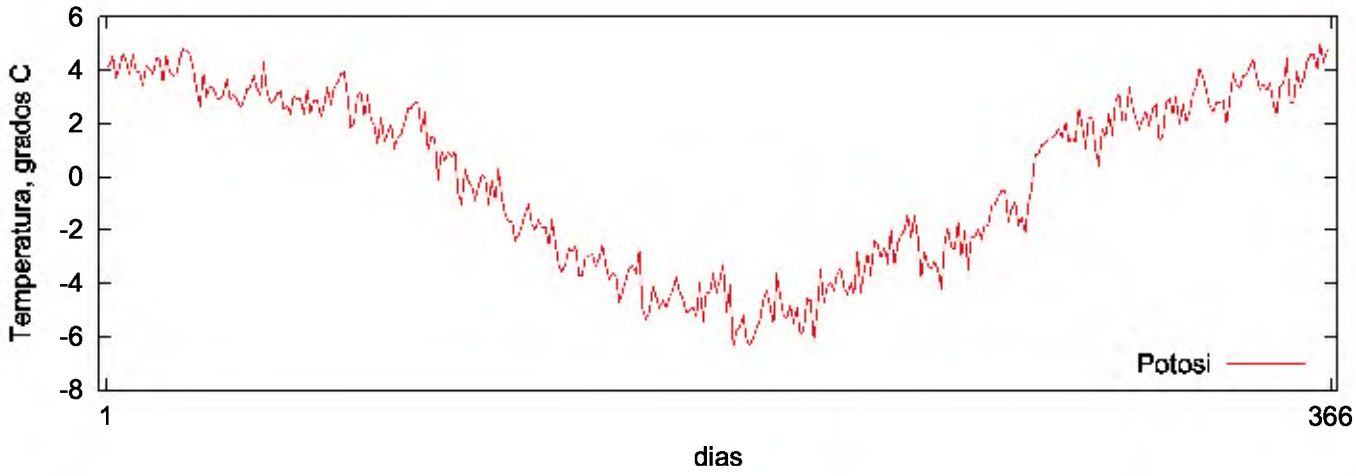


Figura 4. Promedios de la temperatura media diaria de la ciudad de Potosí durante más de 30 años de registros. Estas series temporales poseen una extensión *aa0*.

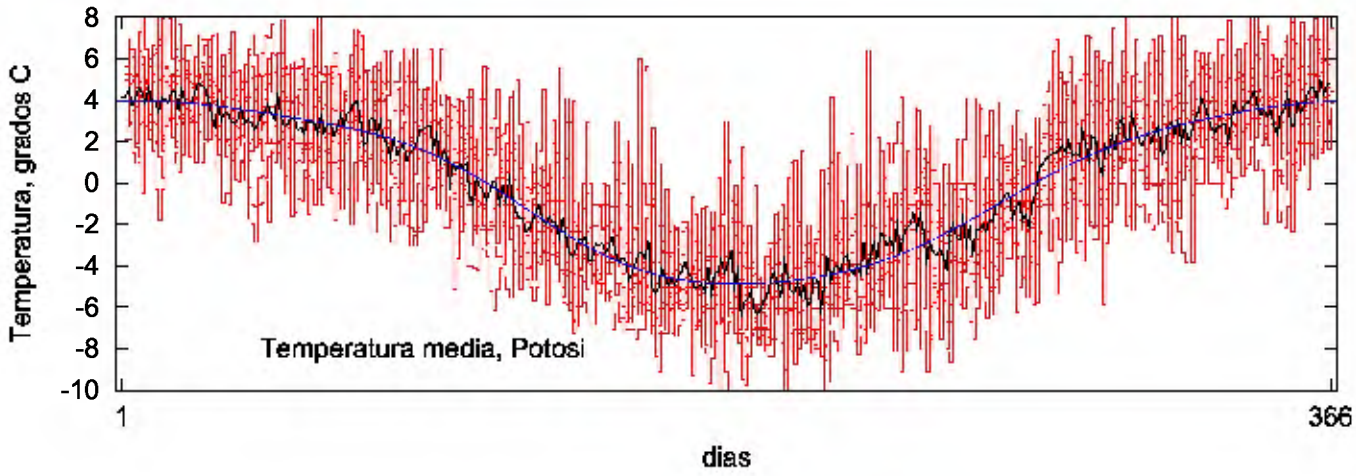


Figura 5. La superposición del archivo **.aa0*, en línea negra, más la superposición de las temperaturas medias de los días superpuestos a un promedio común anual (tomado como $0\,^{\circ}\,C$ en la ilustración) en línea con grandes fluctuaciones. Finalmente, en línea continua el ajuste armónico con cuatro armónicos.

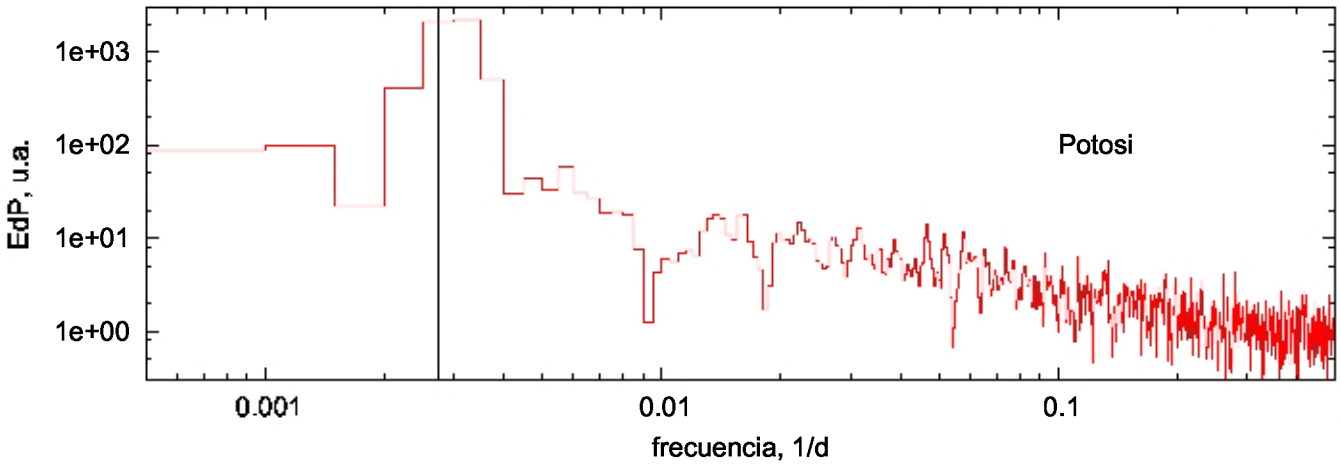


Figura 6. El espectro de frecuencias obtenido a partir de los archivos con extensión *dft*. En las dos primeras columnas aparecen la frecuencia, en las dimensiones indicadas en esta ilustración, y en ordenadas el módulo al cuadrado de la transformada de Fourier de la serie temporal de la localidad de Potosí. La línea vertical indica la frecuencia anual. Las unidades en las ordenadas son arbitrarias, al ser la escala logarítmica. Obsérvese que la máxima frecuencia coincide con la frecuencia de Nyquist, esto es 0,5 en las unidades indicadas.

mayores fluctuaciones que un promedio total de todos los datos a lo largo de un año común. Este tipo de gráfico tiene por objeto familiarizar las variaciones periódicas de los datos de temperatura y precipitación pluvial.

2.5. *La potencia espectral (*.dft)*

Finalmente, se presentan archivos con extensión *dft* (por *digital Fourier Transform*) que suministran series temporales de las potencias espectrales de los datos tanto de la temperatura como de la precipitación pluvial. Se trata de cuatro columnas, la primera es la frecuencia de las oscilaciones en unidades del inverso del muestreo; en el caso de la localidad de Potosí, que se ilustra en la figura 6, se trata de dimensiones físicas de $\frac{1}{d} \equiv \frac{1}{24\,hr}$. La segunda columna es e l espectro de frecuencias y las últimas dos columnas son respectivamente, los días y la autocorrelación.

Si bien este no es el foro apropiado para hablar de las características físicas de la serie temporal tanto del espectro de potencias como de la autocorrelación, se dice sólo que los picos conspícuos que aparecen en los espectros dan las oscilaciones estacionarias de los datos originales y, por otro lado, la autocorrelación es no sólo un paso intermedio para obtener el espectro de potencias, sino que por mérito propio posee información estadística de la serie original.

3. EL DIALECTO *xml*

El obejtivo –a nuestro juicio un tanto ingénuo, de traducir los archivos *ascii* en formato *Excel*, ha sido construido debido a que existe una presión popular, o existen clientes, capaces de "ver" tablas de datos mediante el *Excel* de la plataforma *Microsoft*. Está claro que series temporales largas son bastante pesadas en esta plataforma, y, si se desea efectuar operaciones modernas tales como el espectro cruzado de potencias, o bien *wavelets*, por decir algo, las herramientas de este paquete son insuficientes. Uno debe terminar utilizando paquetes un poco más científicos tales como *Mathematica*, *Matlab* u otro similar. Empero, este ejercicio de llevar archivos *ascii* hacia archivos *Excel* no deja de ser interesante desde el punto de vista de una informática elemental.

El dialecto que manipula archivos sin necesidad de tener un conocimiento técnico de los programas a los que enlaza se denomina *xml*. La forma de los códigos es similar al *html* que se conoce fundamentalmente para la edición de las páginas *web*. En realidad, el explicar la sintáxis de cada proposición de este dialecto no tiene sentido en este tipo de publicación. Sólo se debe añadir que es necesario dar el *modus operandi* de estas traducciones de un modo positivista, ver, ecléctico, sin necesidad de suministrar el por qué de la operación. Sin embargo, en la subsección 3.1 se da un listado de cómo se escribe una fila de datos.

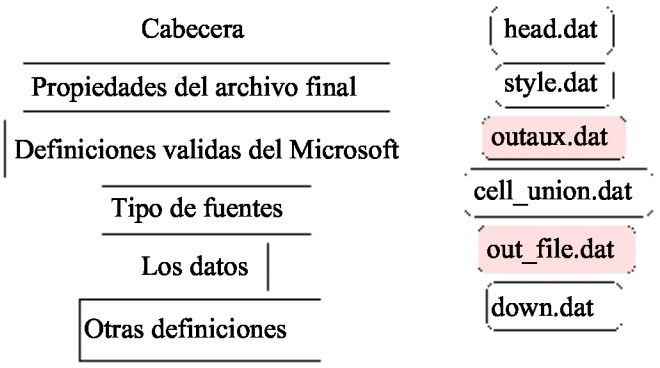


Figura 7. A la izquierda los bloques para la comprensión secuencial del lector y a la derecha la traducción de archivos que se deben construir para tal efecto. Dos de estos archivos (los bloques que tienen un fondo más obscuro) se obtienen a partir de la ejecución de un programa convencional (en C, Fortran, Java,...), en nuestro caso mediante el programa *lastP.c* que lee los datos *ascii* de los archivos de entrada.

3.1. *Un ejemplo de párrafo en el xml*

Con el objeto de dar una impresión superficial de cómo se debe escribir una fila de datos se da a continuación la primera fila de datos (de tres columnas) de la estación de Copacabana

```
</Row>   <Row ss:AutoFitHeight="0">
<Cell ss:StyleID="s23"><Data ss:Type="Number">1.000000</Data></Cell>
<Cell ss:StyleID="s23"><Data ss:Type="Number">1973.000000</Data></Cell>
<Cell ss:StyleID="s23"><Data ss:Type="Number">6.000000</Data></Cell>
</Row>
```

Se puede reconocer que se trata del día 1 del año 1973 cuando la temperatura media diaria alcanzaba a $6\,^{\circ}\,C$.

3.2. *El diagrama de bloques para el archivo legible en el Excel*

Siguiendo con la manera de escribir proposiciones o grupos de proposiciones en el *xml* se ilustra en el gráfico 7 los bloques que se deben construir para poder ejecutar el traspaso de los archivos *ascii* a los *xml*. Algunos archivos, debido a que se deben escribir más de una línea en código *xml* por cada línea en código *ascii*, deben efectuarse mediante un lenguaje superior, digamos el C. Por otro lado, para que no se repita el programa en cuestión sistemáticamente para cada grupo de columnas de los archivos que salen como salida de los programas, se debe suministrar como *argumentos* todos los parámetros necesarios como el número de columnas por archivo, luego los títulos de cada columna así como sus unidades y finalmente el título general de las columnas.

Entonces los ejecutables *a.out* deberán escribirse como

```
./a.out copac1 ser 3 copac1 temp_media
dia año grad C
```

que quiere decir que se lee el archivo *copac1.ser* que tiene 3 columnas cuyas etiquetas son respectivamente *dia*, *año* y *grados C*. Por otro lado el título del archivo se denomina *copac*1 y el subtítulo *temp_media*, esto es *temperatura media*. En forma análoga el ejecutable efecúa una salida para las precipitaciones pluviales, se debe escribir

```
./a.out copac2 ser 3 copac2
prec_pluvial dia año mm/dia
```

y así sucesivamente.

4. EL MECANISMO AUTOMÁTICO DE TRADUCCIÓN

Ahora, sin necesidad de explicar más la textura del banco de datos de la UMSA, se pretende explicar dado un archivo en *ascii* de la estación de Ayo Ayo (esto implica datos de precipitación pluvial como se ha explicado más arriba) cómo encontrar el archivo *Excel* que se exhibe en la figura 8.

Para este efecto, además de la existencia del archivo *ayoay2.har* (como es sabido por el usuario, éste posee cuatro columnas que se hallan explicadas en la leyenda de la figura 8) cuyo formato *Excel* se desea generar, se utiliza el programa *lastP.c*, que se halla listado en la subsección 4.1, y el cual, una vez compilado, se ejecuta mediante el guión que se ilustra en la subsección 4.2, para finalmente obtener el archivo *Excel* buscado y que se puede "levantar" con el *Excel* del *Microsoft*.

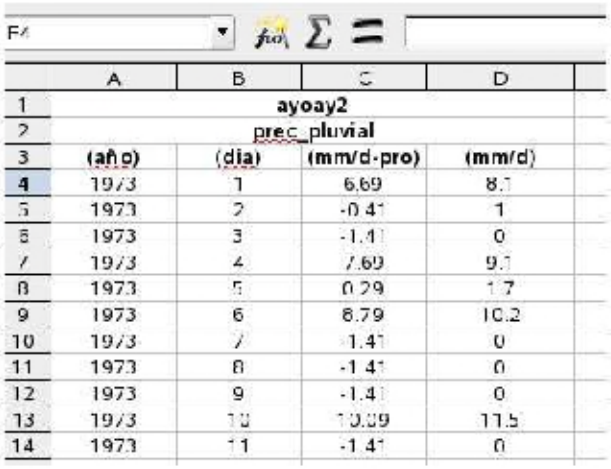
4.1. *El programa que suministra* outaux.dat *y* out_file.dat

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
char *tun[]={"","",""};
//titulos vacios, se pueden llenar
int main(int argc, char *argv[])
{
  int i=0,j;
  int  column,row,aux_1;
  float dat;
  char buffer[30],tit1[20],tit2[20];
  char *xts[]={"dia",argv[7],argv[6],argv[8]};
  //argv[6]="mm/dia"
  FILE *fp_in,*fp_out;
  column=atoi(argv[3]);
  sprintf(buffer,"%s.%s",argv[1],argv[2]);
  aux_1=0;
```

```
  sprintf(tit1,argv[4]);
  sprintf(tit2,argv[5]);
  fp_in=fopen(buffer,"r");
  fp_out=fopen("outfile.dat","w");
  //salida out_file
  while(feof(fp_in)==0){
    fprintf(fp_out,
    "   <Row ss:AutoFitHeight=\"0\">\n");
    for(j=0;j<column;j++){
      fscanf(fp_in,"%f ",&dat);
      fprintf(fp_out,
      "    <Cell ss:StyleID=\"s23\">
       <Data ss:Type=\"Number\">%f</Data>
       </Cell>\n",dat);
    }
    fprintf(fp_out,"   </Row>\n");
    aux_1 += 1;
  }
  fprintf(fp_out," </Table>\n");
  fclose(fp_in);
  fclose(fp_out);
  fp_out=fopen("cell_col_name.dat","w");
  fprintf(fp_out,"<Row ss:AutoFitHeight=\"0\">\n");
  for(i=0; i<column;i++){
    fprintf(fp_out,"   <Cell ss:StyleID=\"s21\">
      <ss:Data ss:Type=\"String\"");
    fprintf(fp_out,"     xmlns=\"http:
      //www.w3.org/TR/REC-html40\"><B>%s </B><Font>
      (%s)</Font></ss:Data></Cell>\n",tun[i],xts[i]);
  }
  fprintf(fp_out,"</Row>");
  aux_1 += 1;
  fclose(fp_out);
  fp_out=fopen("outaux.dat","w");
  //salida outaux.dat
  fprintf(fp_out,"\n<Worksheet
   ss:Name=\"%s.%s\">\n\n",argv[1],argv[2]);
  fprintf(fp_out," <Table ss:ExpandedColumnCount=
   \"%d\" ss:ExpandedRowCount=\"%d\" x:FullColumns=
   \"1\"\n",column,aux_1+2);
  fprintf(fp_out,"   x:FullRows=\"1\" ss:
   DefaultColumnWidth=\"60.400000000000006
   \" ss:DefaultRowHeight=\"13.2\">\n");
  fclose(fp_out);
  fp_out=fopen("cell_union.dat","w");
  //salida cell_union.dat
  fprintf(fp_out,"   <Column ss:Index=
   \"%d\" ss:AutoFitWidth=\"0\" ss:Width=
   \"72\"/>\n",column);
  fprintf(fp_out,"      <Row ss:AutoFitHeight=
   \"0\">\n");
  fprintf(fp_out,"       <Cell ss:MergeAcross=
   \"%d\" ss:StyleID=\"s21\"><Data ss:Type=\"String\">
   %s</Data></Cell>\n",column-1,tit1);
  fprintf(fp_out,"      </Row>\n");
  fprintf(fp_out,"      <Row ss:AutoFitHeight=
   \"0\">\n");
  fprintf(fp_out,"       <Cell ss:MergeAcross=
   \"%d\" ss:StyleID=\"s21\"><Data ss:Type=
   \"String\">%s</Data></Cell>\n",column-1,tit2);
  fprintf(fp_out,"      </Row>\n");
  fclose(fp_out);
  return (0);
}
```

4.2. *El guión para la construcción del archivo Excel*

Una vez compilado el programa *lastP.c* de un modo convencional, se debe concatenar los archivos que éste genera. Esta concatenación se la efectua en una línea como se ilustra más adelante. Por



	A	B	C	D
1	ayoay2			
2	prec_pluvial			
3	(año)	(dia)	(mm/d-pro)	(mm/d)
4	1973	1	6.69	8.7
5	1973	2	-0.41	1
6	1973	3	-1.1	0
7	1973	4	7.69	9.7
8	1973	5	0.29	1.7
9	1973	6	8.79	10.2
10	1973	7	-1.41	0
11	1973	8	-1.41	0
12	1973	9	-1.41	0
13	1973	10	10.09	11.5
14	1973	11	-1.41	0

Figura 8. La hoja *Excel* de la salida del programa *lastP.c* que se halla listado en la subsección 4.1. Por otro lado en la subsección 4.2 se ilustra también la concatenación de diversos archivos –como se muestra en la figura 7, para finalmente obtenerse el archivo *ayoay2har.xml*. A continuación se explica el significado de cada una de las columnas. La primera, se repite el año 365/366 veces, la segunda el día de ese año, la tercera coincide con el valor de la precipitación pluvial para Ayo Ayo pero restándole el promedio de ese año, mientras que la última columna es el valor nominal de la precipitación pluvial de cada uno de los días del año en cuestión. Está claro que la tercera columna esta bajo la estrategia que se pretende encontrar en el análisis armónico año tras año.

otro lado, todos los archivos que se traducen, que son en total unos 3500 archivos *ascii*, deben ser preparados naturalmente con el programa original tal como se ilustra en la figura 2. En conclusión existen decenas de páginas escritas por ese programa que tienen un aspecto similar al de este guión

```
gcc lastP.c
./a.out ayoay2 har 4 ayoay2 prec_pluvial
 mm/dia ano dia mm/dia-pro
cat head.dat style.dat outaux.dat cell_union.dat \
cell_col_name.dat outfile.dat down.dat \
> ayoay2har.xml
```

5. DISCUSIÓN

Independiente de los méritos o deméritos de un banco de datos con la característica delineada en este trabajo, se ha hecho un relato lineal de la manera cómo se traduce archivos *ascii* en archivos *Excel*. Siempre en un contexto de datos escritos en forma de varias columnas.

El problema informático es relativamente simple. Dado un número grande de archivos en formato de columnas, cada uno representando alguna característica de un banco de datos, la traducción

de los archivos *ascii* –típico en una plataforma UNIX de un trabajo científico, necesita de un lenguaje superior, C, Fortran, Java,..etc. para producir otros archivos también en columnas, esto es en formato de *series temporales*, como es habitual en el análisis de datos. La salida de estos programas es de suyo un problema realtivamente complejo, empero, una vez resuelto ese problema, el hecho de escribir un programa con más de una centena de líneas para traducir las columnas de datos en formato *Excel* u otro tradicional no añade gran cosa a la dificultad original de producir series temporales. Parece ser que el formato *Excel*, por el momento, posee muchos seguidores entre los colegiales, secretarias y científicos que desean exhibir rápidamente un primer análisis de esos datos. Pensamos que este tipo de trabajo sin lugar a dudas contribuye a que los paquetes cerrados de *Microsoft* (u otro tipo de paqueterías como *MatLab*, *Mathematica*,...) puedan hacerse transparentes con estos dialectos, apoyados incluso por los creadores de *Excel* debido a que todas las versiones de este paquete a lo largo de decenios deben ser *portables* desde el punto de vista de los creadores del propio *Excel*. Ello sin detrimento de lo misterioso que puede ser el propio paquete de *Excel* o *Word* debido a *copyrights* de estas empresas que no permiten que sus *softwares* sean libres. Desde el punto de vista del *software* libre, esto es los programas en UNIX, hace que se pueda siempre –con un mínimo de esfuerzo, pasar de una plataforma a otra *con el consentimiento de las empresas comerciales.*

Un comentario final. Una vez escrito en el dialecto xml, la plataforma *Excel* "levanta" naturalmente dicho archivo con el formato *Excel*, aunque tarde el doble de tiempo que un archivo escrito originalmente "a mano", esto es, un poco más lento que de costumbre. Sin embargo, por un lado, el usuario seguramente guardará aquellos archivos en el formato original *Excel* de su ordenadora para que la próxima vez la apertura del archivo tenga una extensión xls y así ganar un poco de tiempo. Por otro lado, no es posible que un usuario utilice todos los 3500 archivos a la vez, por lo que una gran cantidad de los archivos xml se quedarán con ese formato, cosa que no es un problema ni para el usuario ni para una ordenadora.

REFERENCIAS

[1] N. Martinic, *Banco de Datos Meteorológico*, Informe Biblioteca Carrera de Física, FCPN, UMSA, 2006.