Procesador de datos de un sistema GNSS

Protocolo NMEA-0183 – Formato GPX

Algoritmos y Programación I (95.11/75.02)

1 de octubre de 2018

1. Objetivo

El objetivo del presente trabajo es la integración de los conocimientos adquiridos hasta el momento en el curso, a través del diseño y desarrollo de una aplicación que permita procesar los datos provenientes de un sensor de un sistema global de navegación por satélites, GNSS por sus siglas en inglés.

2. Alcance

Mediante el presente TP se busca que el/la estudiante aplique conocimientos sobre los siguientes temas:

- Directivas al preprocesador C
- Programas en modo consola
- Tipos enumerativos
- Funciones
- Salida de datos con formato
- Modularización
- Arreglos/Vectores
- Arg. en línea de comandos
- Estructuras
- Modularización

Fecha de entrega: 18 de octubre de 2018

3. Introducción

La localización usando sensores que se comunican con un sistema global de navegación por satélites (GNSS) es ampliamente utilizada en todo el mundo. Prácticamente todo celular moderno posee sensores que se conectan a alguno de estos sistemas, siendo los sistemas GPS uno de ellos.

Para que estos sistemas funcionen, es necesario que el dispositivo tenga "visión" de varios satélites, para poder triangular una posición. Según la cantidad de satélites que se "vean", cambia el tipo de posicionamiento y la precisión del dato.

Un protocolo muy común utilizado por los fabricantes de los sensores GNSS es el protocolo NMEA-0183[1][1][2][3] y es el protocolo que se deberá interpretar en este trabajo. Además, los datos recibidos serán procesados e impresos utilizando el formato GPX[2] (Formato de intercambio GPS)[4][5], que es otro formato muy utilizado para generar registros, por ejemplo, de las rutas recorridas usando un GPS.

Como se asume que dichos protocolos no son de conocimiento general, a continuación se hace una revisión mínima de los mismos.

3.1. Protocolo NMEA-0183

El estándar NMEA-0183 fue desarrollado por la *National Marine Electronics Association*, una empresa de EEUU. Este estándar no sólo define el protocolo de comunicación, sino que además define interfaces eléctricas. Actualmente la empresa no ha publicado el estándar libremente, y lo vende por 350 dólares, es decir, es privativo. Sin embargo, haciendo ingeniería inversa de diversos aparatos, se ha logrado decodificar y se publica en diversos lugares. Se asume que estos sitios poseen información sobre estándares más antiguos que el actual, y la versión correcta y actualizada es la de NMEA. Sin embargo, a los fines prácticos no se ven grandes modificaciones que no permitan realizar el trabajo con, prácticamente, cualquier sensor.

El estándar mencionado genera datos compuestos por caracteres ASCII, lo que hace sencilla su lectura. La trama generada por la secuencia de datos está compuesta por elementos llamados *sentencias*. Cada una de estas *sentencias* aparece en una línea distinta a la anterior. Todas las líneas comienzan con el carácter `$` y finalizan con los caracteres `CRLF`. El formato general de una sentencia es el siguiente:

```
$ttsss,c1,c2,....CRLF
```

Las primeras 2 letras después del `$` (`tt`) identifican al dispositivo (*talker*) que genera el dato, las siguientes 3 letras (`sss`) identifican el tipo de sentencia recibido, luego viene una secuencia de campos (`c1`, `c2`, `ci`, etc) con datos que dependen del tipo de sentencia recibido.

[1] https://es.wikipedia.org/wiki/NMEA_0183
[2] https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp
[3] http://www.gpsinformation.org/dale/nmea.htm
[4] https://es.wikipedia.org/wiki/GPX
[5] http://www.topografix.com/gpx.asp

Algunos, de los muchos, *talkers* son:

- **GP - Receptor GPS**
- **GL - Receptor GLONASS**
- AG - Autopiloto general
- CS - Satélite
- CV Radio/Teléfono (VHF)

Los tipos de sentencias son aún más que los *talkers*, y algunas son:

- RMC - datos mínimos recomendados para GPS
- RMB - datos recomendados para navegación GPS
- **GGA - información del *fix* (posicionamiento)**
- GSV - información de los satélites visibles
- GLL - latitud/longitud
- ZDA - fecha y tiempo

Ejemplo de secuencias provenientes de un *talker* GPS. Las frases contenidas en el ejemplo son `GGA`, `GSA` y `RMC`.

```
$GPGGA,112836.854,1955.118,N,07742.541,W,1,12,1.0,0.0,M,0.0,M,,*7C
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,112836.854,A,1955.118,N,07742.541,W,198121.7,078.3,300918,000.0,W*52
$GPGGA,112837.854,2006.906,N,07645.369,W,1,12,1.0,0.0,M,0.0,M,,*7C
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,112837.854,A,2006.906,N,07645.369,W,811970.2,305.7,300918,000.0,W*5E
$GPGGA,112838.854,2224.499,N,07956.956,W,1,12,1.0,0.0,M,0.0,M,,*71
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
$GPRMC,112838.854,A,2224.499,N,07956.956,W,811970.2,305.7,300918,000.0,W*53
```

Ejemplo 1: Secuencia NMEA-0183

3.2. Formato GPX (GPS eXchange Format)

El formato GPX define la estructura que debe tener un archivo de texto para representar datos de un GPS, que pueden ser rutas, trayectos o puntos. En este trabajo, sólo se generarán trayectos. Del mismo modo que en C tenemos que cumplir ciertas reglas para que el código, el texto escrito sea válido, en GPX también. A continuación se muestra un ejemplo. Las indentaciones y los caracteres de nueva línea no son necesarios, sólo facilitan la lectura.

```
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="95.11 TP1 - Tracker" xmlns="http://www.
    topografix.com/GPX/1/1">
  <metadata>
    <name>Ejemplo</name>
```

```
6       <time>2018-09-30T12:27:12Z</time>
7     </metadata>
8     <trk>
9       <trkseg>
10        <trkpt lat="19.918633" lon="-77.709016">
11          <ele>0.000000</ele>
12          <time>2018-09-30T11:28:36.854Z</time>
13        </trkpt>
14        <trkpt lat="20.115100" lon="-76.756150">
15          <ele>0.000000</ele>
16          <time>2018-09-30T11:28:37.854Z</time>
17        </trkpt>
18        <trkpt lat="22.408316" lon="-79.949266">
19          <ele>0.000000</ele>
20          <time>2018-09-30T11:28:38.854Z</time>
21        </trkpt>
22      </trkseg>
23    </trk>
24  </gpx>
```

Ejemplo 2: Archivo GPX correspondiente al ejemplo 1

Dado que no se pide la creación de un conversor gpx genérico, analizaremos el ejemplo 2 para entender qué se debe imprimir. Sin embargo, antes de eso es fundamental entender, primero, que GPX es un formato basado en XML y por ésto, al igual que en C los bloques estás delimitados por llaves `{}` (generalmente), en GPX están delimitados por *tags*. Los tags son estructuras de texto con el siguiente formato:

1. `<tagname ... />`
2. `<tagname ... > ... </tagname>`

donde `...` indica que puede haber más contenido, incluso otros tags *anidados*.

3.2.1. Encabezado

El encabezado corresponde a la línea 1 del archivo gpx, y es la misma siempre:

```
1  <?xml version="1.0" encoding="UTF-8"?>
```

3.2.2. Elemento `gpx`

Es el *tag* que comienza en la línea 3, y tiene anidado todo el resto del documento. Además, tiene algunos metadatos, como la versión, el creador del documento y el esquema utilizado, como se ve a continuación:

```
3  <gpx version="1.1" creator="95.11 TP1 - Tracker" xmlns="http://www.
       topografix.com/GPX/1/1">

24 </gpx>
```

Metadata: es un *tag* que contiene datos que no corresponden a la ruta, a los datos NMEA, sino que son accesorios y pueden ser interpretados por otros programas. En este trabajo, este tag contendrá en su interior dos tags, uno *name* y el otro *time*. El primero debe almacenar el nombre que se le dará la secuencia, y el segundo la hora del sistema en el momento en que se procesa el NMEA, es decir, la hora al momento de ejecución del programa (en hora UTC). Estos tags se corresponden con las líneas 5 y 6 del ejemplo.

```
5     <name>Ejemplo</name>
6     <time>2018-09-30T12:27:12Z</time>
```

El ejemplo completo del *tag metadata* es:

```
4   <metadata>
5     <name>Ejemplo</name>
6     <time>2018-09-30T12:27:12Z</time>
7   </metadata>
```

3.2.3. Track – Ruta

```
8   <trk>
9     <trkseg>

22    </trkseg>
23  </trk>
```

Es el *tag* que comienza en la línea 8 del ejemplo y en su interior tiene uno o más *tags trkseg* (aunque sólo uno en este trabajo) y cada *trkseg* contiene uno o más *trkpt*. Este *tag* define la ruta que se siguió.

3.2.4. Track points – Puntos de la ruta

```
10      <trkpt lat="19.918633" lon="-77.709016">
11        <ele>0.000000</ele>
12        <time>2018-09-30T11:28:36.854Z</time>
13      </trkpt>
```

Cada *tag trkpt* contiene la latitud y longitud de un punto de la ruta, que se corresponde, en la entrada NMEA, con una línea `GGA`. Además, anidados contiene la elevación y la fecha del punto. Se debe notar que las horas, minutos y segundos se obtienen de los datos de entrada, pero no así el año, mes y día.

4. Desarrollo

En este trabajo se pide que implementar un programa ejecutable por línea de comandos que lea de `stdin` una secuencia de datos provenientes de un sensor e imprima por `stdout` el contenido que se genera para un archivo GPX[6], que es un formato típico para almacenar rutas.

Para realizar esta tarea, el programa deberá ir analizando los datos recibidos, interpretar aquellos que tienen información geográfica, procesarlos e imprimir en el formato especificado.

[6] https://es.wikipedia.org/wiki/GPX

4.1. Procesamiento NMEA-0183

En este trabajo sólo se interpretarán las frases de tipo `GGA`. Toda otra línea que aparezca en el flujo de entrada debe ser ignorada silenciosamente. Para ver el formato de esta sentencia, tomemos la primera línea del ejemplo 1:

```
$GPGGA,222310.983,3437.064,S,05822.107,W,1,12,1.0,0.0,M,0.0,M,,*69

Donde:
   $            Caracter inicial de TODAS las sentencias
   GP           ID del talker (GPS)
   GGA          Tipo de la sentencia (Fix Data)
   222310.983   Horario del fix (22:23:10.983 UTC): hhmmss.sss
   3437.064     Latitud (34 deg 37.064'): ddmm.mmm
   S            Indicador de latitud Norte o Sur (S): S/N
   05822.107    Longitud (58 deg 22.107'): dddmm.mmm
   W            Indicador de longitud Este u Oeste (W): E/W
   1            Calidad del Fix (1): 0 a 8
                    0 = invalido
                    1 = fix GPS (SPS)
                    2 = fix DGPS
                    3 = fix PPS
                    4 = Real Time Kinematic
                    5 = Float RTK
                    6 = Estimada (dead reckoning)
                    7 = Manual
                    8 = Simulación
   12           Cantidad de Satélites utilizados (12): 0 a 12.
   1.0          HDoP (1.0)
   0.0          Elevación (0.0)
   M            Unidad (M, metros)
   0.0          Separación del geoide (aproximación)
   M            Unidad (M, metros)
   ,,           Secuencia de campos nulos para el receptor
   *69          suma de verificación (siempre comienza con *)
```

Estos datos deben cargarse en una estructura diseñada para tal fin. La estructura no debe contener todos los datos, ni mantener el formato especificado, sino que debe resumir la información relevante, a saber: fecha del fix, latitud, longitud, calidad, cantidad de satélites, hdop, elevación y separación del geoide.

4.1.1. Conversión latitud y longitud

Los datos de latitud y longitud en NMEA vienen en formato `[d]ddmm.mmmm`, donde `mm.mmmm` son los minutos y `[d]dd` son los grados, que pueden ser 2 o 3 caracteres. La conversión de estos datos es simple, ya que sólo hay que obtener los grados y minutos, y luego hacer una multiplicación y suma. Con el siguiente ejemplo quedará claro. Si tomamos la latitud del ejemplo anterior:

```
$GPGGA,222310.983,3437.064,S,05822.107,W,1,12,1.0,...
latitud en nmea: 3437.064 (ddmm.mmm), S
grados: 34
minutos: 37.064
    -> latitud = 34 + 37.064/60
Además, si el indicador de latitud es S, se debe multiplicar por -1.

longitud en nmea: 05822.107 (dddmm.mmm), W
grados: 58
minutos: 22.107
    -> longitud = 58 + 22.107/60
Además, si el indicador de longitud es W, se debe multiplicar por -1.
```

4.1.2. Cálculo de la suma de verificación

El cálculo de la suma de verificación (*checksum*) se calcula como el XOR entre todos los caracteres de la sentencia, excluyendo el caracter inicial (`$`) y, obviamente, los caracteres de la suma de verificación que corresponden a los últimos 2 caracteres y el signo `*` anterior a ellos.

```c
unsigned char nmea_checksum(const char * sentence) {
    unsigned char sum = 0;

    while (*sentence)
        sum ^= *sentence++;

    return sum;
}
```

Esta suma se utiliza para saber si el mensaje llegó correctamente o debe ser descartado por estar modificado.

4.1.3. Fecha

Dado que el mensaje de `GGA` no posee el año, el día y el mes, la fecha es incompleta. Por ello se especificará esa tupla de datos a través de la línea de comandos. Para ver cómo se especifica este formato, ver la sección 5. Si esta fecha no es especificada, se utilizará la fecha del sistema.

Debe notarse una diferencia entre las fechas contenidas en los metadatos del comienzo (ver 3.2.2) y el contenido de cada *trkpt* (ver 3.2.4). Estos últimos poseen milisegundos—segundos expresados con decimales—mientras que la fecha de los metadatos no.

Opcionalmente, se puede agregar la opción de por procesar las sentencias `RMC` y obtener de ellas el mes, día y año correspondiente a la secuencia de frases.

4.2. Impresión GPX

La salida GPX debe estar compuesta por el encabezado del archivo, y el elemento `gpx`. Este elemento debe contener los metadatos mencionados en la sección 3.2.2

A medida que se van procesando los datos y se encuentran sentencias de tipo `GGA`, se deben ir cargando las estructuras mencionadas en la sección 4.1 e imprimiendo los *tags trkpt*, como se menciona en la sección 3.2.4.

5. Ejecutable

A continuación se describen los argumentos que debe recibir el programa. Esta descripción sirve para armar la ayuda que debe imprimir el programa.

-h, --help

Muestra una ayuda

-n `nombre`, **--name** `nombre`

Indica el *metadato* nombre (*name*). Ver la sección 3.2.2.

-f `fecha`, **--format** `fecha`

Indica la fecha. `fecha` debe ser una secuencia de 8 dígitos que indiquen el año (con la centuria), el mes y el día, por ejemplo: `20181120` (indica el 20 de noviembre de 2018).

-Y `año`, **--year** `año`

Indica el año. `año` debe ser una secuencia de 4 dígitos que indiquen el año (con la centuria). Por ejemplo, `2018` indica el año 2018.

-m `mes`, **--month** `mes`

Indica el mes. `mes` debe ser una secuencia de 1 ó 2 dígitos que indiquen el mes. Por ejemplo, `11` indica el mes noviembre.

-d `día`, **--day** `día`

Indica el día. `día` debe ser una secuencia de 1 ó 2 dígitos que indiquen el día. Por ejemplo, `20` indica el día veinte.

Si no se indica la fecha, se debe tomar la del sistema.

5.1. Ejecución del programa

Algunos ejemplos de ejecución, sin sus resultados son:

```
$ cat ejemplo.nmea | ./programa -h
$ cat ejemplo.nmea | ./programa --help
$ cat ejemplo.nmea | ./programa >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -f 20181120 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa --fecha 20181120 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -Y 2018 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -m 11 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -d 20 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -m 11 -d 20 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa --month 11 --day 20 >> ejemplo.gpx
```

```
$ cat ejemplo.nmea | ./programa -Y 2018 -d 20 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa -Y 2018 -m 11 >> ejemplo.gpx
$ cat ejemplo.nmea | ./programa --year 2018 --month 11 >> ejemplo.gpx
```

6. Testing

Junto con el informe se han agregado algunos archivos nmea y gpx. Los archivos nmea se pueden utilizar para generar una salida en formato gpx y compararla con la que se encuentra en el archivo comprimido.

Además, en la sección 7 se describen algunas herramientas útiles para este trabajo.

7. Herramientas

7.1. Generadores de archivos NMEA

7.1.1. Online

Son webs que permiten generar trayectorias ficticias, ya sea aleatorias o especificándolas sobre un mapa.

- FreeNMEA: http://freenmea.net/
- NMEA Generator: https://nmeagen.org/

7.1.2. Apps Android

Requieren GPS en el teléfono y, potencialmente, permiten almacenar un recorrido real en el teléfono.

- GPSd Client: https://github.com/tiagoshibata/Android-GPSd-Client
- GPS Logger for Android: https://github.com/mendhak/gpslogger/

Ambas aplicaciones están en Google Play o Play Store y son de código libre. Únicamente GPSd Client se encuentra disponible en F-Droid. Ambas aplicaciones se pueden descargar desde sus repositorios.

7.1.3. GNU/Linux

GPSd http://www.catb.org/gpsd/ es un demonio/servicio de GNU/Linux que permite conectar cualquier dispositivo que genere datos GPS, ya sea serie, por USB o por internet y recibirlos en la computadora. Por ejemplo, es posible conectar *GPSd Client* y, utilizando la aplicación *gpspipe*, se puede obtener la salida deseada.

8. Restricciones

La realización de los programas pedidos está sujeta a las siguientes restricciones:

- Debe realizarse en grupos de **3 (tres)** integrantes.
- No está permitida la utilización de `scanf()`, `gets()`[7], `fflush(stdin)`[8],la biblioteca `conio.h`[9] (`#include <conio.h>`), etc.
- Debe recurrirse a la utilización de funciones mediante una adecuada parametrización.
- No está permitido en absoluto tener hard-codings:

```
...
char s[282];
...
if (s[1] == 'G' && s[2] == 'G' && s[3] == 'A') /*¡hard-coded!*/
    printf("%s","xxxxxxxx");          /* ¡¡hard-coded!! */
...
if (!strcmp(argv[1], "--formato"))   /* ¡¡hard-coded!! */
```

  sino que debe recurrirse al uso de `ETIQUETAS`, `CONSTANTES SIMBÓLICAS`, `MACROS`, etc.

  Los ejemplos no son exhaustivos, sino que existen otros hard-codings y tampoco son aceptados.

- Hay ciertas cuestiones que no han sido especificadas intencionalmente en este Requerimiento, para darle al/la desarrollador/a la libertad de elegir implementaciones que, según su criterio, resulten más convenientes en determinadas situaciones. Por lo tanto, se debe explicitar cada una de las decisiones adoptadas, y el o los fundamentos considerados para las mismas.

9. Entrega

Fecha límite de **aprobación**: 1 de noviembre de 2018.

La entrega en papel no es obligatoria, pero debe acordarse con quien corrija el trabajo si la exige o no. Sin embargo, es obligatorio realizar una entrega digital, a través del campus de la materia y por correo electrónico, de un único archivo cuyo nombre debe seguir el siguiente formato:

`YYYYMMDD_apellido1-apellido2-apellido3-N.tar.gz`

donde `YYYY` es el año (2018), `MM` el mes y `DD` el día en que uno de los integrantes sube el archivo, `apellido-1a3` son los apellidos de los integrantes ordenados

[7] obsoleta en C99 [3], eliminada en C11 [4] por fallas de seguridad en su uso.
[8] comportamiento indefinido para flujos de entrada ([3],[4]). Definida en estándar POSIX.
[9] biblioteca no estándar, con diferentes implementaciones y licencias, y no siempre disponible.

alfabéticamente, `N` indica el número de vez que se envía el trabajo (`1`, `2`, etc.), y `.tar.gz` es la extensión, que no necesariamente es `.tar.gz`.

El archivo comprimido debe contener los siguientes elementos:

- La correspondiente documentación de desarrollo del TP (en formato pdf), siguiendo la numeración siguiente, incluyendo:
  1. Carátula del TP. Incluir una dirección de correo electrónico.
  2. Enunciado del TP.
  3. Estructura funcional de los programas desarrollados.
  4. Explicación de cada una de las alternativas consideradas y las estrategias adoptadas.
  5. Resultados de la ejecución (corridas) de los programas, captura de las pantallas, bajo condiciones normales e inesperadas de entrada.
  6. **Archivos de prueba utilizados.**
  7. Reseña sobre los problemas encontrados en el desarrollo de los programas y las soluciones implementadas para subsanarlos.
  8. Bibliografía (ver sección 10).
  9. Indicaciones sobre la compilación de lo entregado para generar la aplicación.
     NOTA: Si la compilación del código fuente presenta mensajes de aviso (warning), notas o errores, los mismos deben ser comentados en un apartado del informe.

  NOTA: El Informe deberá ser redactado en *correcto* idioma castellano.
- Códigos fuentes en formato de texto plano (.c y .h), *debidamente documentados*.

NOTA: Se debe generar y subir un único archivo (comprimido) con todos los elementos de la entrega digital. **NO usar RAR.** La compresión RAR no es un formato libre, en tanto sí se puede utilizar *ZIP*, *GUNZIP*, u otros (soportados, por ejemplo, por la aplicación de archivo *TAR*).

Si no se presenta cada uno de estos ítems, será rechazado el TP.

10. Bibliografía

Debe incluirse la referencia a toda bibliografía consultada para la realización del presente TP: libros, artículos, URLs, etc., citando:

- Denominación completa del material (Título, Autores, Edición, Volumen, etc.).
- Código ISBN si lo tiene.
- URL del sitio consultado. No poner Wikipedia.org o stackexchange.com, sino que debe incluírse un enlace al artículo, hilo, etc. consultado.

Utilizando LaTeX, la inclusión de citas/referencias es trivial. Los editores de texto gráficos de las suites de ofimática, como LibreOffice Write o MS Word, admiten plugins que facilitan la inclusión.

Ejemplo de referencias

[1] National Marine Electronics Association. *NMEA 0183 Interface Standard*. 0183. NMEA, 2008.

[2] *GPS eXchange Format*. Open Standard/Public Domain 1.1. 2004. URL: http://www.topografix.com/gpx.asp.

[3] B.W. Kernighan y D.M. Ritchie. *The C Programming Language*. 2.ª ed. Prentice-Hall software series. Prentice Hall, 1988. ISBN: 9780131103627.

[4] P. Deitel y H. Deitel. *C How to Program*. 7.ª ed. Pearson Education, 2012. ISBN: 9780133061567.

[5] ISO/IEC. *Programming Languages – C*. ISO/IEC 9899:1999(E). ANSI, dic. de 1999, págs. 270-271.

[6] ISO/IEC. *Programming Languages – C*. INCITS/ISO/IEC 9899:2011. INCITS/ISO/IEC, 2012, pág. 305.