



**UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA**

Año 2018 - 2^{do} Cuatrimestre

ALGORITMOS Y PROGRAMACIÓN I (95.11)

TRABAJO PRÁCTICO FINAL

INTEGRANTES:

Landrobe, Natalia. <natylandrobe@gmail.com> Padrón: 102832

Marca Lima, Renán Michael. <michael9763@hotmail.com> Padrón: 103041

1. Enunciado

Implementar un programa ejecutable por línea de comandos que lea de un archivo de entrada una secuencia de datos provenientes de un sensor (GNSS) e imprima por un flujo de salida el contenido que se genera para un archivo GPX.

Las sentencias a procesar serán: ZDA, RMC y GGA en el caso de NMEA, y UBX-TIM-TOS, UBX-NAV-PVT y UBX-NAV-POSLLH en el caso de UBX.

Para realizar esta tarea, el programa debe ir analizando los datos recibidos, interpretar aquellos que tienen información útil, procesarlos e imprimirlo en el formato especificado.

El programa deberá ser capaz de almacenar una cierta cantidad de mensajes en una lista de mensajes, en espera a ser procesados.

El programa deberá contener un ciclo de procesamiento donde se almacenan los mensajes y luego se imprime una cantidad de mensajes a calcular por el programa. Los mensajes ZDA, en el caso de NMEA, y UBX-TIM-TOS, en el caso de UBX, se utilizarán para actualizar la fecha almacenada, para ser utilizado en caso de que algún mensaje la requiera. Los mensajes GGA y RMC, en el caso de NMEA y UBX-NAV-PVT y UBX-NAV-POSLLH, en el caso de UBX, se utilizaran para obtener datos de posicionamiento. Los mensajes GGA y UBX-NAV-PVT deben descartarse si, siendo válido el mensaje, el fix no lo es (la calidad del fix es inválida en NMEA o bit gnssFixOK del registro flags del mensaje UBX-NAV-PVT no es 1).

Estos datos serán almacenados en un contenedor de tipo lista. La lista siempre contendrá como primer elemento el mensaje más antiguo.

Luego de cada carga de mensaje en la lista, se analizará si se procesan datos de la lista o no. Para tal fin, se generará un número aleatorio entero, el cual será utilizado como cantidad de mensajes a procesar. Al procesar los mensajes, se generan los Track Points correspondientes al formato GPX.

El programa deberá utilizar un archivo de logs donde se almacenarán incidentes en el programa, estos incidentes no serán terminantes y el programa deberá continuar. Ejemplos de incidentes con diferentes niveles de "log".

ERROR: Son errores graves que ocurren durante la ejecución. Ejemplos:

- No se pudo abrir un archivo.
- Un checksum no concuerda.
- Un mensaje UBX debería tener un cierto largo y tiene otro.
- Un mensaje NMEA estaba mal formulado, etc.

WARN: Son avisos que ocurren durante la ejecución. Ejemplos:

- No se reconoce un ID.
- Un mensaje contiene un fix inválido.
- Se descarta un mensaje por lista llena, etc.

DEBUG: Son mensajes de información de sobre lo que está haciendo el programa. Ejemplos:

- Buscando los 2 bytes de sincronismo.
- Detecta un determinado ID.
- Recolectó un mensaje.
- Cargo un mensaje en la lista.
- Imprimió un mensaje, etc.

A continuación se describen los argumentos que debe recibir el programa:

-h, --help

Muestra una ayuda.

-n nombre , --name nombre

Indica el metadata nombre(name).

-p protocolo , --protocol protocolo

Indica el protocolo a leer, puede tomar los valores "nmea" o "ubx".

-i archivo , --infile archivo

Indica el nombre del archivo a utilizar como entrada de datos.

-o archivo , --outfile archivo

Indica el nombre del archivo a utilizar para el archivo gpx.

-l archivo , --logfile archivo

Indica el nombre del archivo a utilizar para el archivo log.

-m cantidad , --maxlen cantidad

Indica la máxima cantidad de mensajes que se pueden almacenar en la lista.

2. Estructura funcional del programa desarrollado:

Al comenzar el programa se cargan la fecha universal con la hora del sistema, y se carga la estructura con los argumentos default. Luego se verifican los argumentos, que son ingresados por línea de comandos, con la función `takeArgs`. Esta misma es la que retorna un `status_t` que lo guardamos en una variable llamada `st`. Si `st == ST_HELP` (es decir, el usuario usó, por ejemplo, el argumento `-h`) imprimimos un mensaje de ayuda sobre el programa, si `st == ST_INV`, hubo un error en la escritura del argumento, por lo tanto, imprimimos un mensaje de error por `stderr`. Si `st == ST_EPTNULL` hubo un problema del programa al pasarle los parámetros a la función, por lo que este termina.

A continuación se llama a la función para abrir los archivos que correspondan según los argumentos recibidos. Esta retorna un `status_t` que informa si pudo realizarse correctamente, y de lo contrario, el programa termina. Después enviamos el nombre, la dirección de memoria de la fecha universal y el puntero al archivo de salida a la función `printMetadata` que imprime en formato XML la metadata si recibió correctamente los argumentos. Se crea la lista con una función diseñada para tal fin y luego, se procede al procesamiento de sentencias según el protocolo ingresado.

En el caso del NMEA, primero encontramos un ciclo el cual parará cuando se haya alcanzado la cantidad máxima de mensajes a imprimir o cuando no existan más mensajes y `fgets` retorne `NULL`. Con la función `checkline` se verifica que la sentencia corresponda a uno de los tipos a leer y que el checksum sea correcto, y luego esta devuelve el tipo de sentencia, el cual será almacenado en la variable `t`. Se utiliza esta variable para realizar un switch entre los tipos de sentencias NMEA, y dentro del mismo se llama a las funciones para cargar las estructuras, y agregar nodos a la lista. Las que cargan las estructuras devuelven por el nombre un `status_t`

con el que verificamos que los datos hayan sido correctos, y de ser una sentencia con datos completos de fecha, se modifica la fecha universal por interfaz. Luego, si se trata de un mensaje de navegación, se agrega un nodo a la lista con la información del mensaje y de realizarse correctamente, se incrementa el valor de la variable de iteración.

Para el caso de UBX, se utiliza otro ciclo cuya condición de corte será que se haya alcanzado el número máximo de sentencias a leer, o que se hayan terminado, en cuyo caso la función `procesar_ubx` retornará `S_EREAD`. Todo el procesamiento de las sentencias UBX ocurre dentro de esta función. La misma comienza con una iteración leyendo byte a byte la información hasta encontrar los bytes de sincronismo. Dependiendo de si el archivo de entrada es `stdin` o no, se llamará a `procesar_standard`. Si no lo es, continúa leyendo el byte correspondiente a la clase y se verifica que se trate de un tipo de sentencia válido, de lo contrario se la descartará imprimiendo un mensaje de warning. Se leen los dos bytes del largo y se llama a una función que lo calcula, luego se verifica que el largo se corresponda con el largo esperado para ese tipo de sentencia, de lo contrario se descartará imprimiendo un mensaje de error. Se utiliza la función `fseek` para mover el cursor hacia el byte desde donde se toma el checksum y se lee la sentencia guardándola en un puntero a `unsigned char` con memoria alocada. Si el checksum es correcto, se llama a la función para cargar la estructura correspondiente que además actualiza - si corresponde - la fecha universal y verifica que los datos sean válidos. Luego si es una sentencia con datos de navegación, se guarda un nodo con los datos en la lista y se incrementa por interfaz la variable de iteración y termina la función. De ser `stdin` el archivo de entrada, la única diferencia es que cada byte que se lee, se guarda en un puntero a `unsigned char` con memoria alocada, y luego para cargarle el payload se utiliza un ciclo que comienza desde la posición correspondiente.

A continuación se llama a las funciones de impresión de los mensajes de la lista y de los cierres de tags, las cuales retornan un `status_t` para corroborar que hayan impreso correctamente. Luego se libera la memoria de la lista a través de a función `destruir_lista`, se cierran los archivos abiertos a través de `cerrar_archivos`, y se libera la memoria pedida para el almacenamiento de los argumentos. Estas funciones también retornan un `status_t` con el que verificamos que todo haya funcionado correctamente, y luego el programa termina.

A continuación presentamos un listado de las funciones del programa:

- `bool checkDia(int dia);`
La función recibe un número entero (`dia`) y verifica que el valor de día sea válido (es decir que se encuentre dentro del rango 1-31). Esta función retorna por el nombre *false* si el número de día cargado no se encuentra en el intervalo válido y *true* si este es válido.

- `bool checkMes(int mes);`
La función recibe un número entero (mes) y verifica que el valor de mes sea válido (es decir que se encuentre dentro del rango 1-12). Esta función retorna por el nombre *false* si el número de mes se encuentra por fuera del intervalo válido y *true* si este es válido.
- `bool checkAnio(int anio);`
La función recibe un número entero (anio) y verifica que el valor de año sea válido (es decir que se encuentre dentro del rango 1-9999). Esta función retorna por el nombre *false* si el número de año no se encuentra en el intervalo válido y *true* si este es válido.
- `cal_t convertirCal(long int cal);`
La función recibe un número entero (cal) y retorna por nombre la *calidad del Fix* como un tipo enumerativo *cal_t*.
- `double convertirLon(const char lon[], char * cardinal);`
La función recibe una cadena constante (lon), y un puntero a *char* (cardinal). Convierte la cadena constante en un número, teniendo en cuenta si el indicador del punto cardinal es E u O.
Retorna un double por el nombre:
 - *ERR_LATLON* (número fraccionario 181) si la longitud interpretada no es válida.
 - La longitud como número fraccionario, positivo si es este, negativo si es oeste.
- `double convertirLat(const char lat[], char * cardinal);`
La función recibe una cadena constante (lat) y un puntero a *char* (cardinal). Convierte la cadena constante en un número, teniendo en cuenta si el indicador del punto cardinal es N o S.
Retorna un double por el nombre:
 - *ERR_LATLON* (número fraccionario 181) si la latitud interpretada no es válida.
 - La latitud como número fraccionario, positivo si es norte, negativo si es sur.
- `void printHelp(void);`
La función no toma argumentos, imprime un mensaje de ayuda y devuelve void.
- `unsigned char nmea_checksum(const char * s);`
La función recibe un puntero a una constante *char*, realiza el cálculo de la suma de verificación para las sentencias NMEA(ZDA,RMC y GGA) y retorna un unsigned char.
- `bool checkMembers(double lat, double lon, cal_t cal, long int cant);`

La función recibe dos números enteros (lat y lon), un dato de tipo *cal_t* (cal) y un número entero (cant), provenientes de la sentencia GGA.

Esta función verifica que los datos se encuentren dentro del rango de validez y retorna por el nombre, *false* si alguno de los datos no es válido y *true* si todos son válidos.

- `bool checkMembersrmc(double lat, double lon);`
La función recibe dos números del tipo double (lat y lon), provenientes de la sentencia RMC.
Esta función verifica que los datos se encuentren dentro del rango de validez y retorna por el nombre, *false* si alguno de los datos no es válido y *true* si los dos son válidos.
- `bool checkNum(char *s);`
La función recibe un puntero a char, y verifica que la cadena recibida representa un número decimal. Esta función retorna por el nombre *false* si la cadena contiene caracteres que no sean dígitos del 0 al 9 y *true* si la cadena representa un número válido.
- `sent_t checkLine(char *s, FILE *flog);`
La función recibe un puntero a un char, un puntero a FILE . Verifica que la línea sea de tipo \$GPGGA, \$GPZDA o \$GPRMC y que la suma de verificación sea correcta, y retorna por el nombre el tipo de sentencia que es: ZDA, RMC, GGA o NING (ninguno).
El puntero a FILE floges para saber en qué archivo se debe imprimir los log en ese caso (ejemplo: se identificó el ID).
- `status_t printMetadata(char *name, struct fecha *fecha, FILE *fout)`
La función recibe un puntero a char, un puntero a una estructura de tipo fecha y un puntero a FILE. Imprime el tag Metadata en el formato basado en XML y devuelve un valor del tipo enumerativo *status_t* :
 - *ST_EPTNULL*: si fout, fecha o name son NULL.
 - *ST_OK*: si se imprimieron en el formato XML en el archivo de salida fout.
- `status_t printTrkC(FILE *fout);`
La función recibe un puntero a FILE. Imprime cierres de los tags Trackseg, Track y gpx en el archivo de salida fout y devuelve un valor de tipo enumerativo *status_t* (*ST_EPTNULL* si fout es NULL o *ST_OK* si se imprimieron en el formato XML en el archivo de salida).
- `status_t printStruct(struct trkpt *track, FILE *fout);`
La función recibe un puntero a FILE y un puntero a una estructura del tipo trkpt. Imprime la estructura trkpt en formato XML en el archivo de salida fout y devuelve un valor de tipo enumerativo *status_t* (*ST_EPTNULL* si fout o track

es NULL o ST_OK si se imprimió track en el formato XML en el archivo de salida).

- void imp_log(FILE *flog, status_t * status, ubxst_t *ubx_st, debug_t *deb);
La función recibe un puntero a FILE, un puntero a un status_t , un puntero a ubxst_t y un puntero a unx_st. De acuerdo al tipo de pointer ingresado imprime mensajes de error, warning y debug en el archivo log.
- status_t defaultFecha(struct fecha *def);
La función recibe un puntero a una estructura de tipo *fecha* (def) y carga por la interfaz la fecha del sistema a la estructura de tipo *fecha*.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL : si el puntero def es nulo.
 - ST_OK: si se cargó satisfactoriamente la fecha del sistema a la estructura.
- status_t defaultArgs(struct args *arg);
La función recibe un puntero a una estructura de tipo args, carga la estructura args por la interfaz con los argumentos default.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL : si el puntero arg es nulo.
 - ST_OK: si se cargó satisfactoriamente la los argumentos por default a la estructura.
 - ST_ENOMEM: si no se pudo obtener memoria para el argumento name, infile, outfile, logfile.
- status_t cargar_struct_zda(char *s, struct s_ZDA *Zda, struct fecha *date);
La función recibe un puntero a char, un puntero a la estructura s_ZDA y un puntero a la estructura fecha. Procesa la sentencia ZDA, carga la estructura s_ZDA por la interfaz y asigna su fecha a la estructura de fecha date.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL: si el puntero ZDA, s o date son nulos.
 - ST_OK: si se procesaron y cargaron satisfactoriamente los datos a la estructura.
 - ST_SENTINV: si uno de los datos procesados es inválido.
- status_t cargar_struct_rmc(char *s, struct s_RMC *Rmc, struct fecha *date);
La función recibe un puntero a char, un puntero a la estructura s_RMC y un puntero a la estructura fecha. Procesa la sentencia RMC, carga la estructura s_RMC por la interfaz y asigna su fecha a la estructura de fecha date.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL: si el puntero Rmc, s o date son nulos.
 - ST_OK: si se procesaron y cargaron satisfactoriamente los argumentos a la estructura.

- ST_SENTINV: si uno de los datos procesados es inválido.
- status_t cargar_struct_gga(char *s, struct s_GGA *Gga, struct fecha *date);
La función recibe un puntero a char, un puntero a la estructura s_GGA y un puntero a la estructura fecha. Procesa la sentencia GGA, carga la estructura s_GGA por la interfaz asignándole los valores de la fecha universal para la fecha.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL: si el puntero Gga, s o date son nulo.
 - ST_OK: si se procesaron y cargaron satisfactoriamente los argumentos a la estructura.
 - ST_SENTINV: si uno de los datos procesados es inválido.
- status_t takeArgs(int argc, char *argv[], struct args *arg);
La función recibe un entero, un vector de puntero a char y un puntero a la estructura args. Lee los argumentos recibidos, verifica que sean correctos y decide que acción tomar según el argumento.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL : si el puntero arg, argv o argc son nulos.
 - ST_OK: si se cargaron satisfactoriamente los argumentos a la estructura args.
 - ST_HELP: si el argumento ingresado es para pedir ayuda.
 - ST_ENOMEM: si no se pudo obtener memoria para los argumentos name, logfile, outfile, infile, protocol.
 - ST_INV: si alguno de los argumentos ingresados es inválido.
- status_t procesar_arg(int argum, char *argv[], args_t *argp, size_t index);

La función recibe un entero, un vector de puntero a char, un puntero a un tipo enumerativo args_t, y un size_t. Toma el argumento y por interfaz devuelve el tipo de argumento correspondiente.
Retorna por el nombre un valor de tipo enumerativo status_t:
 - ST_EPTNULL : si el puntero argv o argp son nulos.
 - ST_OK: si se cargó satisfactoriamente el tipo de argumento correspondiente.
 - ST_HELP: si el argumento ingresado es para pedir ayuda.
 - ST_ENOMEM: si no se pudo obtener memoria para los argumentos name, logfile
 - ST_INV: si el argumento ingresado es inválido.
- status_t liberar_args(struct args *arg);
La función recibe un puntero a la estructura args. Libera la memoria pedida para los campos de la estructura args. Devuelve un valor de tipo enumerativo status_t (ST_EPTNULL si arg es NULL o ST_OK si se libera la memoria pedida para cada campo de args).
- status_t abrir_archivos(FILE **fin, FILE **fout, FILE **flog, struct args *arg);

La función recibe tres doble puntero a FILE(fin , fout y flog) y un puntero a la estructura args. Abre los archivos correspondientes según los argumentos ingresados.

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si fin, fout, flog o arg son nulo.
- ST_OK: si los archivos se abrieron correctamente o se les asignaron los de default.
- ST_EFILEO: si no se pudo abrir el archivo para los logs, el archivo de entrada de datos o el archivo de salida.
- ST_EFILEC: si no se pudo cerrar los archivos para los logs, el archivo de entrada o el de salida en caso de que haya un error abriendo un archivo luego de haber abierto otro.

- status_t cerrar_archivos(FILE **fin, FILE **fout, FILE **flog, struct args *arg);
La función recibe tres doble puntero a FILE(fin , fout y flog) y un puntero a la estructura args. Cierra los archivos abiertos

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si fin, fout, flog o arg son nulos.
- ST_EFILEC: si no se pudo cerrar los archivos para los logs, el archivo de entrada de datos o el de salida.

- status_t crear_lista(lista_t *l);

La función recibe un puntero a tipo de dato lista_t. Le asigna NULL a la lista apuntada.

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si l es nulo.
- ST_OK: si se creó la lista .

- status_t destruir_nodo(lista_t *l);

La función recibe un puntero a tipo de dato lista_t. Libera la memoria del nodo.

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si l es nulo.
- ST_OK: si se pudo liberar la memoria del nodo.

- status_t destruir_lista(lista_t *l);

La función recibe un puntero a tipo de dato lista_t. Libera la memoria de la lista.

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si l es nulo o no se pudo destruir un nodo.
- ST_OK: si se pudo liberar la memoria de la lista.

- status_t imprimir_lista(lista_t l, FILE *fout, FILE *flog);

La función recibe un tipo de dato lista_t , dos punteros a FILE(fout y flog). Imprime nodo por nodo la lista.

Retorna por el nombre un valor de tipo enumerativo status_t:

- ST_EPTNULL : si l es nulo, fout o flog es nulo.
- ST_OK: si se pudo imprimir la lista.

- `struct trkpt *cargar_trkpt(const struct fecha *fecha, const double lat, const double lon, const double ele);`
 La función recibe un puntero a una estructura fecha, tres double (lat, lon y ele).
 Toma los datos de una estructura y carga una del tipo trkpt.
 Retorna NULL si fecha es NULL o si no se pudo obtener memoria para un puntero auxiliar a la estructura trkpt. En caso contrario retorna un puntero a una estructura trkpt cargada con los datos de los argumentos.
- `struct trkpt *proc_sentencias(void * dato, sent_t tipo);`
 La función recibe un puntero a void y un tipo de dato sent_t. Toma una estructura cualquiera de las sentencias y pasa los datos correspondientes para crear la trkpt de acuerdo al tipo de sentencia. Retorna por el nombre NULL si dato es NULL o si no se cargó la estructura correctamente, de lo contrario, devuelve un puntero a una estructura trkpt.
- `nodo_t *crear_nodo(void *dato, struct trkpt *(*procesar)(void *, sent_t), sent_t tipo);`
 La función recibe un puntero a void del cual tomar los datos, un puntero a una función para procesar los datos, y un sent_t que indica el tipo de sentencia de la cual tomar los datos para el nodo. Retorna NULL si alguno de los punteros de los argumentos es NULL, si no había memoria, o si no se procesaron los datos de manera correcta. De lo contrario, retorna un puntero a nodo_t.
- `status_t agregar_nodo(void * dato, lista_t *l, sent_t tipo);`
 Esta función recibe un puntero a void del cual toma los datos, un puntero a lista y un sent_t para indicar el tipo de sentencia de la cual toma los datos.
 La función retorna un status_t:
 - ST_EPTNULL si alguno de los punteros de los argumentos es NULL
 - ST_EAGR si hubo un error al crear el nodo a agregar.
 - ST_OK si se agregó correctamente el nodo a la lista
- `unsigned int calc_largo(unsigned char info[]);`
 La función recibe un vector de unsigned char y retorna por el nombre el largo del payload de una estructura, ya que el mismo viene en formato little endian.
- `ubxst_t procesar_ubx(FILE *fin, struct fecha *fecha, lista_t *lista, size_t *index, status_t (*add_nodo)(void *, lista_t *, sent_t), FILE *flog, struct args`

*arg);

Esta función toma dos punteros a FILE para los archivos de entrada (fin) y log (flog), un puntero a una estructura de tipo fecha, un puntero a lista, un puntero a size_t, un puntero a una función para agregar nodos y un puntero a una estructura del tipo args.

Esta función lee la sentencia ubx, procesa los datos y de ser correctos, carga las estructuras, y las agrega a la lista si corresponde y aumenta la variable de iteración por la interfaz. Además, al procesar una sentencia con datos completos de fecha, actualiza la fecha universal por la interfaz. Retorna un ubxst_t:

- S_EPTNULL si alguno de los punteros de los argumentos es NULL o si en el procesamiento de datos algo falla por un puntero nulo.
- S_ENOMEM si no se pudo alocar correctamente la memoria.
- S_EREOF si no se puede realizar la lectura, dado que terminó el archivo.
- S_CLASS_INV si la clase de la sentencia es inválida.
- S_ID_INV si el id de la sentencia es inválido.
- S_CKS_INV si el checksum de la sentencia no concuerda.
- S_LARGO_INV si el largo de una sentencia no es el que corresponde.
- S_FIX_INV si el gnssFix no es 1.
- S_EAGR si no se pudo agregar correctamente el nodo a la lista.
- S_OK si pudo realizar todo el procesamiento correctamente.

- ubxst_t procesar_standard(struct fecha *fecha, lista_t *lista, size_t *index, status_t (*add_nodo)(void *, lista_t *, sent_t), FILE *flog);

La función toma como argumentos un puntero a una estructura de tipo fecha, un puntero a lista, un puntero a size_t, un puntero a una función para agregar nodos y un puntero a FILE para el log. Esta función realiza los mismos procedimientos que procesar_ubx, leyendo de stdin y guardando en un vector a medida que lee. Al procesar una sentencia con datos de fecha completos, actualiza la fecha universal por la interfaz. Por el nombre retorna un ubxst_t:

- S_EPTNULL si alguno de los punteros de los argumentos es NULL o si en el procesamiento de datos algo falla por un puntero nulo.
- S_ENOMEM si no se pudo alocar correctamente la memoria.
- S_EREOF si no se puede realizar la lectura, dado que terminó el archivo.
- S_CLASS_INV si la clase de la sentencia es inválida.

- S_ID_INV si el id de la sentencia es inválido.
- S_CKS_INV si el checksum de la sentencia no concuerda.
- S_LARGO_INV si el largo de una sentencia no es el que corresponde.
- S_FIX_INV si el gnssFix no es 1.
- S_EAGR si no se pudo agregar correctamente el nodo a la lista.
- S_OK si pudo realizar todo el procesamiento correctamente.
- `ubxst_t ubx_cksum(unsigned char *ckBuff, int n, FILE *fin);`
 Esta función recibe un puntero a unsigned char, un entero n que será la cantidad de elementos en ckBuff, y un puntero a FILE para el archivo de entrada. La función verifica que el checksum de la sentencia sea correcto, y retorna por el nombre un ubxst_t:
 - S_EPTNULL si alguno de los punteros de los argumentos es nulo.
 - S_EREAD si no pudo leer correctamente del archivo de entrada.
 - S_CK_INV si el checksum no concuerda.
 - S_OK si el checksum es correcto.
- `ubxst_t calc_fecha(unsigned char *buff, struct fecha *fecha, unsigned char id);`
 La función recibe un puntero a unsigned char, un puntero a una estructura de tipo fecha y un unsigned char que corresponde al id de la sentencia de la cual se toman los datos a procesar. Esta función procesa los datos de las sentencias que poseen fecha y los carga por interfaz a la estructura fecha de la estructura de su tipo. Retorna un ubxst_t:
 - S_EPTNULL si alguno de los punteros de los argumentos es NULL.
 - S_OK si pudo cargar los datos correctamente.
- `ubxst_t cargar_fecha(void *dato, struct fecha *funi, unsigned char id, unsigned char *buff, ubxst_t (*proc_fecha)(unsigned char *, struct fecha *, unsigned char));`
 Esta función recibe un puntero a void de donde se tomarán los datos a procesar, un puntero a una estructura de tipo fecha para la fecha universal, un unsigned char para el id de la sentencia, un puntero a unsigned char y un puntero a una función para procesar los datos de la fecha. La función procesa los datos de la fecha y los carga a la estructura del tipo correcto, actualizando si corresponde la fecha universal. Retorna por el nombre S_EPTNULL si alguno de los punteros de los argumentos es nulo o si no se realiza

correctamente la carga de los datos, o S_OK si se realizó.

- `ubxst_t cargar_precision(struct s_POSLLH *dato, unsigned char *buff);`
La función toma como argumento un puntero a una estructura de tipo `struct s_POSLLH` y un puntero a `unsigned char`. La función procesa los bytes de precisión de la sentencia POSLLH y los carga por interfaz a la estructura. Retorna un `ubxst_t` S_EPTNULL si alguno de los argumentos es nulo o S_OK si se realizó correctamente la carga.
- `ubxst_t cargar_pos(void *dato, unsigned char id, unsigned char *buff);`
Esta función toma como argumentos un puntero a void de donde tomará los datos, un `unsigned char` para el id de la sentencia a procesar, y un puntero a `unsigned char`. La función procesa los datos según el tipo de sentencia y los carga a la estructura correspondiente. Retorna un `ubxst_t` S_EPTNULL si alguno de los punteros de los argumentos es NULL o S_OK si pudo realizar correctamente la carga.
- `ubxst_t cargar_sPVT(struct s_PVT * dato, struct fecha *funi, unsigned char *buff);`
La función toma un puntero a una estructura de tipo `s_PVT`, un puntero a una estructura de tipo fecha para la fecha universal y un puntero a `unsigned char`. La función procesa los datos de la sentencia y los carga por la interfaz a la estructura. Además actualiza la fecha universal. Retorna por el nombre S_EPTNULL si alguno de los argumentos es NULL o si no pudo realizarse la carga, de lo contrario retorna S_OK.
- `ubxst_t cargar_sPOSLLH(struct s_POSLLH *dato, struct fecha *funi, unsigned char *buff);`
Esta función toma un puntero a una estructura de tipo `s_POSLLH`, un puntero a una estructura de tipo fecha para la fecha universal y un puntero a `unsigned char`. La función procesa los datos de la sentencia y los carga por la interfaz a la estructura, tomando como datos de fecha los de la fecha universal. Retorna por el nombre S_EPTNULL si alguno de los argumentos es NULL o si no pudo realizarse la carga, si no retorna S_OK.

- `ubxst_t cargar_sTIMTOS(struct s_TIM_TOS *dato, struct fecha *funi, unsigned char *buff);`

La función toma un puntero a una estructura de tipo `s_TIM_TOS`, un puntero a una estructura de tipo `fecha` para la fecha universal y un puntero a `unsigned char`. La función procesa los datos de la sentencia `TIM_TOS`, los carga por la interfaz a la estructura, y actualiza por interfaz la fecha universal. Retorna por el nombre `S_EPTNULL` si alguno de los argumentos es `NULL` o si no pudo realizarse la carga, si no retorna `S_OK`.

3. Explicación de alternativas consideradas y estrategias adoptadas:

- Si no se ingresan los argumentos de `infile`, `outfile`, `logfile`, se tomarán los estandar (`stdin`, `stdout`, `stderr`).
- Los argumentos pueden ingresarse en cualquier orden, y si son ingresados en el formato de caracter (ejemplo: `-n`), puede hacerse en mayúscula o minúscula.
- El único argumento obligatorio, a menos que se corra el programa con el argumento `help`, es el protocolo. Si no se ingresa el resto, se tomarán los default (nombre default, cantidad aleatoria de mensajes máximos a leer, y los archivos ya mencionados).
- Puede ingresarse como nombre de los archivos “-” para utilizar los estándar.
- Para la elevación se toman los datos de elevación de la GGA, 0 para la RMC y la altura sobre el nivel del mar para las UBX.
- Los errores de puntero nulo o de que no hay memoria se imprimen por `stderr`.
- Si el archivo log no se encuentra abierto, los mensajes se imprimen por `stderr`.
- Asumimos que todas las sentencias nmea traen *.
- Las sentencias que traigan datos completos acerca de la fecha y hora, actualizan una estructura de “fecha universal”, y las que necesiten tomar los datos de la fecha, los tomarán de esta estructura.

4. Resultados de ejecución del programa:

File Edit View Search Terminal Help

```
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ gcc -Wall -Wpedantic -o tpf *.c -lm -g
args.c: In function 'takeArgs':
args.c:26:5: warning: enumeration value 'HELP_A' not handled in switch [-Wswitch]
    switch(argp){
    ^~~~~~
main.c: In function 'main':
main.c:118:4: warning: enumeration value 'NAV_PVT' not handled in switch [-Wswitch]
    switch (t){
    ^~~~~~
main.c:118:4: warning: enumeration value 'TIM_TOS' not handled in switch [-Wswitch]
main.c:118:4: warning: enumeration value 'NAV_POSLLH' not handled in switch [-Wswitch]
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$
```

Imagen [1]: se muestra la compilación del programa utilizando el gcc. El programa no tiene ni errores, pero presenta warnings que trataremos luego.

File Edit View Search Terminal Help

```
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ ./tpf
[ERROR] Ingrese un argumento valido
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$
```

Imagen [2]: se corre el programa sin pasarle argumentos, por lo que el mismo termina, ya que el argumento "protocol" es obligatorio para la ejecución.

```
File Edit View Search Terminal Help
```

```
imestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ cat ejemplo.nmea | ./tpf -p nmea  
<?xml version="1.0" encoding="UTF-8"?>
```

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

```
  <metadata>
```

```
    <name>Default Name</name>
```

```
    <time>2018-12-5T21:31:2Z</time>
```

```
  </metadata>
```

```
  <trk>
```

```
    <trkseg>
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```



```

File Edit View Search Terminal Help
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-12-5T11:28:36.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:36.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="20.115100" lon="-76.756150">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:37.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="22.408317" lon="-79.949267">
        <ele>0.000000</ele>
        <time>2018-11-13T11:28:38.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="20.115100" lon="-76.756150">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:37.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="22.408317" lon="-79.949267">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:38.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      </trkseg>
    </trk>
  </gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ █

```

Imagen [3,4]: se corre el programa en modo Default para el protocolo NMEA.

```

File Edit View Search Terminal Help
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ cat test.ubx | ./tpf -p ubx
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="ubxmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>Default Name</name>
    <time>2018-12-5T21:31:35Z</time>
  </metadata>
  <trk>
    <trkseg>
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2014-1-2T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        ..
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-67.070544" lon="-73.774929">
        <ele>0.000000</ele>
        <time>2020-5-12T3:15:13.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
    </trkseg>
  </trk>
</gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ 

```

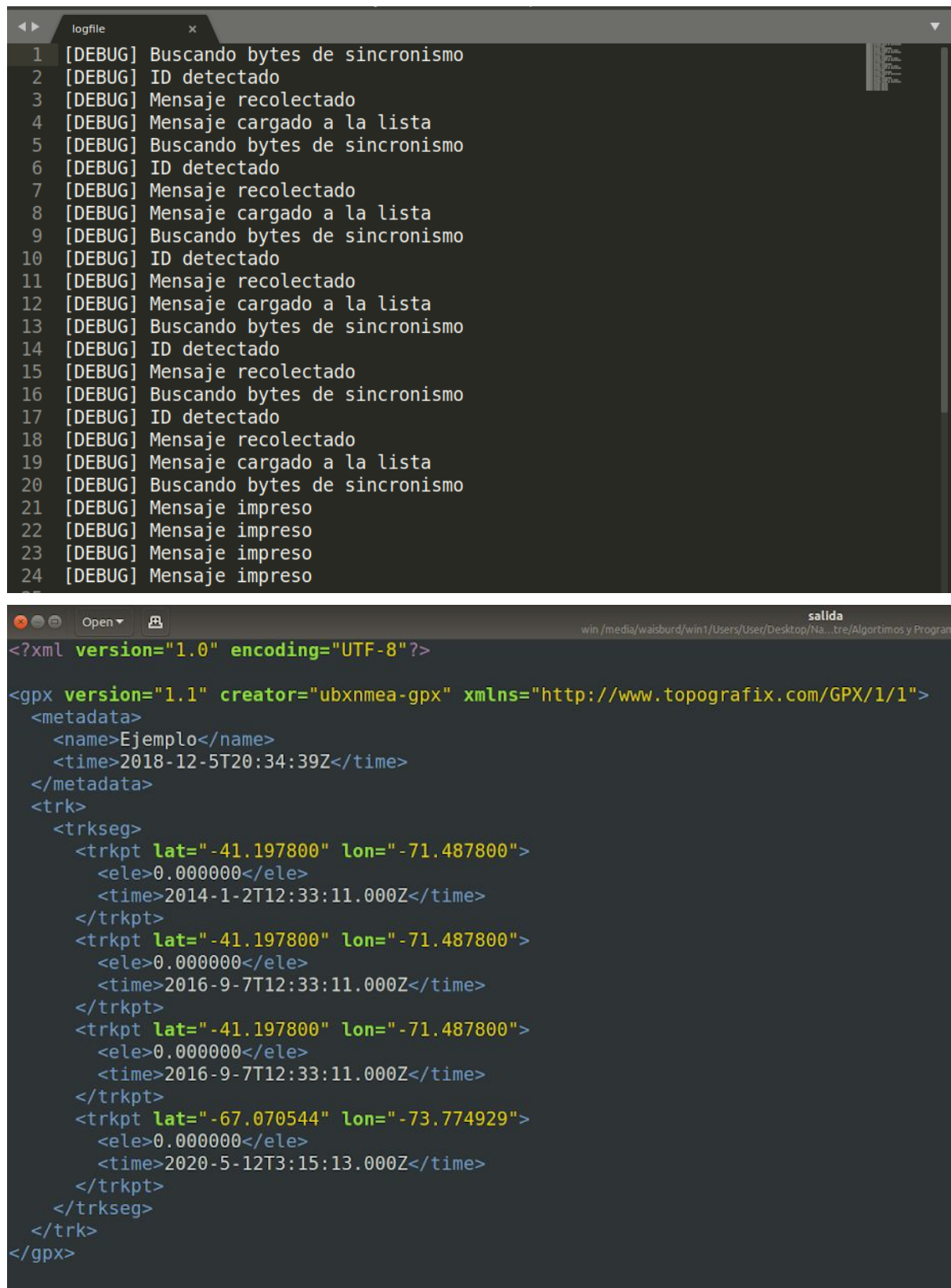
Imagen [5,6]: se corre el programa en modo Default para el protocolo UBX.

```

File Edit View Search Terminal Help
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ ./tpf -p ubx --name Ejemplo -i test
.ubx --outfile salida -L logfile
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ ./tpf -p nmea -N Ejemplo --infile e
jemplo.nmea -O salida --logfile logfile
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat

```

Imagen[7]: se corre el programa pasándole como argumento los nombres de los archivos y el protocolo.



```
1 [DEBUG] Buscando bytes de sincronismo
2 [DEBUG] ID detectado
3 [DEBUG] Mensaje recolectado
4 [DEBUG] Mensaje cargado a la lista
5 [DEBUG] Buscando bytes de sincronismo
6 [DEBUG] ID detectado
7 [DEBUG] Mensaje recolectado
8 [DEBUG] Mensaje cargado a la lista
9 [DEBUG] Buscando bytes de sincronismo
10 [DEBUG] ID detectado
11 [DEBUG] Mensaje recolectado
12 [DEBUG] Mensaje cargado a la lista
13 [DEBUG] Buscando bytes de sincronismo
14 [DEBUG] ID detectado
15 [DEBUG] Mensaje recolectado
16 [DEBUG] Buscando bytes de sincronismo
17 [DEBUG] ID detectado
18 [DEBUG] Mensaje recolectado
19 [DEBUG] Mensaje cargado a la lista
20 [DEBUG] Buscando bytes de sincronismo
21 [DEBUG] Mensaje impreso
22 [DEBUG] Mensaje impreso
23 [DEBUG] Mensaje impreso
24 [DEBUG] Mensaje impreso
```

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.1" creator="ubxntea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>Ejemplo</name>
    <time>2018-12-5T20:34:39Z</time>
  </metadata>
  <trk>
    <trkseg>
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2014-1-2T12:33:11.000Z</time>
      </trkpt>
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
      <trkpt lat="-67.070544" lon="-73.774929">
        <ele>0.000000</ele>
        <time>2020-5-12T3:15:13.000Z</time>
      </trkpt>
    </trkseg>
  </trk>
</gpx>
```

Imagen [8,9]: resultados de la ejecución anterior para el protocolo UBX.

```

1 [DEBUG] Buscando bytes de sincronismo
2 [ERROR] No concuerda el checksum
3 [DEBUG] Buscando bytes de sincronismo
4 [DEBUG] ID detectado
5 [DEBUG] Mensaje recolectado
6 [DEBUG] Mensaje cargado a la lista
7 [DEBUG] Buscando bytes de sincronismo
8 [ERROR] No concuerda el checksum
9 [DEBUG] Buscando bytes de sincronismo
10 [DEBUG] ID detectado
11 [DEBUG] Mensaje recolectado
12 [DEBUG] Mensaje cargado a la lista
13 [DEBUG] Buscando bytes de sincronismo
14 [DEBUG] ID detectado
15 [DEBUG] Mensaje recolectado
16 [DEBUG] Mensaje cargado a la lista
17 [DEBUG] Buscando bytes de sincronismo
18 [DEBUG] ID detectado
19 [DEBUG] Mensaje recolectado
20 [DEBUG] Buscando bytes de sincronismo
21 [DEBUG] ID detectado
22 [DEBUG] Mensaje recolectado
23 [DEBUG] Mensaje cargado a la lista
24 [DEBUG] Buscando bytes de sincronismo
25 [DEBUG] ID detectado
26 [DEBUG] Mensaje recolectado
27 [DEBUG] Mensaje cargado a la lista
28 [DEBUG] Buscando bytes de sincronismo
29 [ERROR] No concuerda el checksum
30 [DEBUG] Buscando bytes de sincronismo
31 [DEBUG] ID detectado
32 [DEBUG] Mensaje recolectado
33 [DEBUG] Mensaje cargado a la lista
34 [DEBUG] Mensaje impreso
35 [DEBUG] Mensaje impreso
36 [DEBUG] Mensaje impreso
37 [DEBUG] Mensaje impreso
38 [DEBUG] Mensaje impreso
39 [DEBUG] Mensaje impreso
40

```

```

salida
win /media/waisburd/win1/Users/User/Desktop/Na...tre/Algoritmos y Program
<?xml version="1.0" encoding="UTF-8" ?>
<gpx version="1.1" creator="ubxnmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>tpf</name>
    <time>2018-12-5T20:36:52Z</time>
  </metadata>
  <trk>
    <trkseg>
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-12-5T11:28:36.854Z</time>
      </trkpt>
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:36.854Z</time>
      </trkpt>
      <trkpt lat="20.115100" lon="-76.756150">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:37.854Z</time>
      </trkpt>
      <trkpt lat="22.408317" lon="-79.949267">
        <ele>0.000000</ele>
        <time>2018-11-13T11:28:38.854Z</time>
      </trkpt>
      <trkpt lat="20.115100" lon="-76.756150">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:37.854Z</time>
      </trkpt>
      <trkpt lat="22.408317" lon="-79.949267">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:38.854Z</time>
      </trkpt>
    </trkseg>
  </trk>
</gpx>

```

Imagen [10, 11]: resultados de la ejecución anterior para el protocolo NMEA.

File Edit View Search Terminal Help

```
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat  
imestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ cat ejemplo.nmea | ./tpf -p nmea -N  
tpf -I - -o - -l -
```

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

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

```
  <metadata>
```

```
    <name>tpf</name>
```

```
    <time>2018-12-5T21:35:58Z</time>
```

```
  </metadata>
```

```
  <trk>
```

```
    <trkseg>
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```



```

[DEBUG] Mensaje cargado a la lista
<trkpt lat="19.918633" lon="-77.709017">
  <ele>0.000000</ele>
  <time>2018-12-5T11:28:36.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
<trkpt lat="19.918633" lon="-77.709017">
  <ele>0.000000</ele>
  <time>2018-9-30T11:28:36.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
<trkpt lat="20.115100" lon="-76.756150">
  <ele>0.000000</ele>
  <time>2018-9-30T11:28:37.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
<trkpt lat="22.408317" lon="-79.949267">
  <ele>0.000000</ele>
  <time>2018-11-13T11:28:38.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
<trkpt lat="20.115100" lon="-76.756150">
  <ele>0.000000</ele>
  <time>2018-9-30T11:28:37.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
<trkpt lat="22.408317" lon="-79.949267">
  <ele>0.000000</ele>
  <time>2018-9-30T11:28:38.854Z</time>
</trkpt>
[DEBUG] Mensaje impreso
</trkseg>
</trk>
</gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat

```

Imagen [12,13]: resultados de la ejecución del programa indicando “-” como nombre de archivo para los archivos de entrada, salida y log para el protocolo NMEA.

```

waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ cat test.ubx | ./tpf -p ubx -N tpf
-I - -o - -l -
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="ubxmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>tpf</name>
    <time>2018-12-5T21:37:2Z</time>
  </metadata>
  <trk>
    <trkseg>
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2014-1-2T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
-      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-67.070544" lon="-73.774929">
        <ele>0.000000</ele>
        <time>2020-5-12T3:15:13.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
    </trkseg>
  </trk>
</gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxmea-gpx$ █

```

Imagen [14,15]: resultados de la ejecución del programa indicando “-” como nombre de archivo para los archivos de entrada, salida y log para el protocolo UBX.

```

File Edit View Search Terminal Help
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algortimos y Programación I/tpf/ubxnmea-gpx$ ./tpf -p ubx -N tpf -I test.ubx -o
- -l - -m 3
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="ubxnmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>tpf</name>
    <time>2018-12-5T21:40:40Z</time>
  </metadata>
  <trk>
    <trkseg>
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[WARNING] Se descartan mensajes por lista llena
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2014-1-2T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
    </trkseg>
  </trk>
</gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algortimos y Programación I/tpf/ubxnmea-gpx$ █

```

Imagen [16]: resultados de la ejecución del programa cuando la cantidad de mensajes a imprimir es menor que la cantidad de mensajes disponibles para el protocolo UBX.


```

waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatri...
File Edit View Search Terminal Help
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ ./tpf -p nmea -N tpf -I ejemplo.nme
a -o - -l - -m 3
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="ubxnmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>tpf</name>
    <time>2018-12-5T21:40:13Z</time>
  </metadata>
  <trk>
    <trkseg>
[DEBUG] Buscando bytes de sincronismo
[ERROR] No concuerda el checksum
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[ERROR] No concuerda el checksum
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[WARNING] Se descartan mensajes por lista llena
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-12-5T11:28:36.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="19.918633" lon="-77.709017">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:36.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="20.115100" lon="-76.756150">
        <ele>0.000000</ele>
        <time>2018-9-30T11:28:37.854Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
    </trkseg>
  </trk>
</gpx>
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuat
imestre/Algoritmos y Programación I/tpf/ubxnmea-gpx$ █

```

Imagen [17]: resultados de la ejecución del programa cuando la cantidad de mensajes a imprimir es menor que la cantidad de mensajes disponibles para el protocolo NMEA.

```
File Edit View Search Terminal Help
```

```
==8602== Command: ./tpf -p nmea -N tpf -I ejemplo.nmea -o - -l - -m 3
```

```
==8602==
```

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

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

```
  <metadata>
```

```
    <name>tpf</name>
```

```
    <time>2018-12-5T21:42:5Z</time>
```

```
  </metadata>
```

```
  <trk>
```

```
    <trkseg>
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[ERROR] No concuerda el checksum
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[DEBUG] Buscando bytes de sincronismo
```

```
[DEBUG] ID detectado
```

```
[DEBUG] Mensaje recolectado
```

```
[DEBUG] Mensaje cargado a la lista
```

```
[WARNING] Se descartan mensajes por lista llena
```

```
  <trkpt lat="19.918633" lon="-77.709017">
```

```
    <ele>0.000000</ele>
```

```
    <time>2018-12-5T11:28:36.854Z</time>
```

```
  </trkpt>
```

```
[DEBUG] Mensaje impreso
```

```
  <trkpt lat="19.918633" lon="-77.709017">
```

```
    <ele>0.000000</ele>
```

```
    <time>2018-9-30T11:28:36.854Z</time>
```

```
  </trkpt>
```

```
[DEBUG] Mensaje impreso
```

```
  <trkpt lat="20.115100" lon="-76.756150">
```

```
    <ele>0.000000</ele>
```

```
    <time>2018-9-30T11:28:37.854Z</time>
```

```
  </trkpt>
```

```
[DEBUG] Mensaje impreso
```

```
  </trkseg>
```

```
  </trk>
```

```
</gpx>
```

```
==8602==
```

```
==8602== HEAP SUMMARY:
```

```
==8602==      in use at exit: 0 bytes in 0 blocks
```

```
==8602==    total heap usage: 24 allocs, 24 frees, 11,372 bytes allocated
```

```
==8602==
```

```
==8602== All heap blocks were freed -- no leaks are possible
```

```
==8602==
```

```
==8602== For counts of detected and suppressed errors, rerun with: -v
```

```
==8602== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```

File Edit View Search Terminal Help
==8633== Memcheck, a memory error detector
==8633== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8633== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==8633== Command: ./tpf -p ubx -N tpf -I test.ubx -o - -l - -m 3
==8633==
<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1" creator="ubxnmea-gpx" xmlns="http://www.topografix.com/GPX/1/1">
  <metadata>
    <name>tpf</name>
    <time>2018-12-5T21:43:9Z</time>
  </metadata>
  <trk>
    <trkseg>
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[DEBUG] Buscando bytes de sincronismo
[DEBUG] ID detectado
[DEBUG] Mensaje recolectado
[DEBUG] Mensaje cargado a la lista
[WARNING] Se descartan mensajes por lista llena
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2014-1-2T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
      <trkpt lat="-41.197800" lon="-71.487800">
        <ele>0.000000</ele>
        <time>2016-9-7T12:33:11.000Z</time>
      </trkpt>
[DEBUG] Mensaje impreso
    </trkseg>
  </trk>
</gpx>
==8633==
==8633== HEAP SUMMARY:
==8633==      in use at exit: 0 bytes in 0 blocks
==8633==    total heap usage: 30 allocs, 30 frees, 11,848 bytes allocated
==8633==
==8633== All heap blocks were freed -- no leaks are possible
==8633==
==8633== For counts of detected and suppressed errors, rerun with: -v
==8633== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo (

```

Imagen [18, 19]: resultados de la ejecución del programa con el valgrind, para ambos protocolos no hay errores ni fugas de memoria.

```

waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
File Edit View Search Terminal Help
o - -l - -m 3
==8735== Memcheck, a memory error detector
==8735== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8735== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==8735== Command: ./tpf -p ubx -N tpf -Infi test.ubx -o - -l - -m 3
==8735==
[ERROR] Ingrese un argumento valido
==8735==
==8735== HEAP SUMMARY:
==8735==      in use at exit: 0 bytes in 0 blocks
==8735==    total heap usage: 14 allocs, 14 frees, 5,495 bytes allocated
==8735==
==8735== All heap blocks were freed -- no leaks are possible
==8735==
==8735== For counts of detected and suppressed errors, rerun with: -v
==8735== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$ valgrind ./tpf -p ubx -N tpf -Infi nmea -o -
-l - -m 3
==8739== Memcheck, a memory error detector
==8739== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8739== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==8739== Command: ./tpf -p ubx -N tpf -Infi nmea -o - -l - -m 3
==8739==
[ERROR] Ingrese un argumento valido
==8739==
==8739== HEAP SUMMARY:
==8739==      in use at exit: 0 bytes in 0 blocks
==8739==    total heap usage: 14 allocs, 14 frees, 5,495 bytes allocated
==8739==
==8739== All heap blocks were freed -- no leaks are possible
==8739==
==8739== For counts of detected and suppressed errors, rerun with: -v
==8739== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$ valgrind ./tpf -p ubx -N tpf -Infi nmea -o -
-l - -m a
==8741== Memcheck, a memory error detector
==8741== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8741== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==8741== Command: ./tpf -p ubx -N tpf -Infi nmea -o - -l - -m a
==8741==
[ERROR] Ingrese un argumento valido
==8741==
==8741== HEAP SUMMARY:
==8741==      in use at exit: 0 bytes in 0 blocks
==8741==    total heap usage: 14 allocs, 14 frees, 5,495 bytes allocated
==8741==
==8741== All heap blocks were freed -- no leaks are possible
==8741==
==8741== For counts of detected and suppressed errors, rerun with: -v
==8741== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$

```

Imagen [20]: resultados de la ejecución del programa con valgrind para casos de error, no hay errores ni fugas de memoria.

```
waisburd@waisburd-desktop: /media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
File Edit View Search Terminal Help
waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$ ./tpf -h
Utilice en cualquier orden los siguientes argumentos:
-n o --name para ingresar el nombre de la ruta
-i o --infile para ingresar el nombre del archivo de entrada (utilice - para stdin)
-o o --outfile para ingresar el nombre del archivo de salida (utilice - para stdout)
-l o --logfile para ingresar el nombre del archivo log (utilice - para stderr)
-p o --protocol para indicar el protocolo a leer
-m o --maxlen para indicar la cantidad maxima de mensajes a leer

waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$ ./tpf -H
Utilice en cualquier orden los siguientes argumentos:
-n o --name para ingresar el nombre de la ruta
-i o --infile para ingresar el nombre del archivo de entrada (utilice - para stdin)
-o o --outfile para ingresar el nombre del archivo de salida (utilice - para stdout)
-l o --logfile para ingresar el nombre del archivo log (utilice - para stderr)
-p o --protocol para indicar el protocolo a leer
-m o --maxlen para indicar la cantidad maxima de mensajes a leer

waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$ ./tpf --help
Utilice en cualquier orden los siguientes argumentos:
-n o --name para ingresar el nombre de la ruta
-i o --infile para ingresar el nombre del archivo de entrada (utilice - para stdin)
-o o --outfile para ingresar el nombre del archivo de salida (utilice - para stdout)
-l o --logfile para ingresar el nombre del archivo log (utilice - para stderr)
-p o --protocol para indicar el protocolo a leer
-m o --maxlen para indicar la cantidad maxima de mensajes a leer

waisburd@waisburd-desktop:/media/waisburd/win1/Users/User/Desktop/Naty/FIUBA/Segundo Cuatrimestr...
re/Algoritmos y Programación I/tpf/ubxnmea-gpx$
```

Imagen [21]: resultados de la ejecución del programa utilizando el argumento “help”.

5. Archivos de prueba utilizados:

- ejemplo.nmea
- test.ubx (realizado por Natalia)

6. Reseña de problemas encontrados en el desarrollo del programa y soluciones:

Al hacer la función que carga los argumentos default, la cantidad de mensajes a imprimir debía ser un número aleatorio, por lo que utilizamos la función `rand()`, pero nos encontramos con el problema de que siempre obteníamos el mismo número aleatorio. Es por esto que utilizamos como seed de la función `rand` la función `time()` que devuelve los segundos desde el EPOCH, y dado que los segundos varían, también lo hace nuestra cantidad máxima.

Otro problema con el que nos encontramos fue que, al hacer que la función `procesar_ubx` fuese la que se encargara de tanto la lectura como el procesamiento y carga de los valores de las sentencias, tuvimos que agregar como argumento de la misma un puntero al index del ciclo en el cual se llama a la función para poder cortarlo si se alcanzara el número máximo.

Un segundo problema que tuvimos con esta función es que cuando el archivo `infile` era `stdin`, no podíamos utilizar la función `fseek` para mover el cursor hacia la

posición desde la que tomamos el checksum. Es por esto que hicimos la función procesar_stdin que a medida que lee los bytes de las sentencias los va guardando en un array de unsigned chars para luego realizar el checksum.

Por último, al hacer la función imp_log la cual imprime el mensaje correspondiente en el archivo de log según el parámetro pasado, tuvimos que definir el tipo debug_t para los mensajes de DEBUG. Esto sucedió ya que cuando se imprimen esos mensajes la función no necesariamente retorna algo (que luego será utilizado por la función), por lo que utilizamos una variable de este tipo para pasarle como argumento.

7. Indicaciones sobre la compilación:

Se debe compilar todos los archivos .c juntos, incluyendo el parámetro -lm al momento de hacerlo, para que el compilador enlace las librerías necesarias:

```
gcc -Wall -Wpedantic -o tpf *.c -lm -g
```

Al momento de compilarlo aparecen los siguientes warnings:

args.c: In function 'takeArgs':

```
args.c:26:5: warning: enumeration value 'HELP_A' not handled in switch [-Wswitch]
    switch(argp){
    ^~~~~~
```

Esto se debe a que evaluamos previamente la posibilidad de que el argumento sea "help", ya que este argumento no necesita que el usuario ingrese nada más (a diferencia de todos los otros que necesitan el ingreso de un dato, como nombres o números).

```
main.c:118:4: warning: enumeration value 'NAV_PVT' not handled in switch
[-Wswitch]
    switch (t){
    ^~~~~~
```

```
main.c:118:4: warning: enumeration value 'TIM_TOS' not handled in switch
[-Wswitch]
```

```
main.c:118:4: warning: enumeration value 'NAV_POSLLH' not handled in switch
[-Wswitch]
```

Esto es porque en ese switch lo que estamos analizando es qué tipo de sentencia nmea es, GGA, RMC o ZDA, por lo que los tipos UBX no son considerados.

Nota:

No lo compilamos con std=c99 porque la función strdup no está en c99

8. Bibliografía:

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

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

[3] https://www.tutorialspoint.com/c_standard_library/c_function_time.htm

[4] https://www.tutorialspoint.com/c_standard_library/stdio_h.htm

[5] https://www.tutorialspoint.com/c_standard_library/c_function_fseek.htm