Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 49 (2012) 9876-9878

ISSN: 2229-712X

Ranging station log data extraction for IRNSS project

B Prabadevi [a] and Neetha Tirmal[b]

[a]Department of Computer science, Sona College of Technology , Salem-05, India.

[b]Space Navigation Group, ISAC, Bangalore-17, India.

ARTICLE INFO

Article history:

Received: 20 June 2012;

Received in revised form:

23 July 2012;

Accepted: 4 August 2012;

Keywords

Satellite,
Data extraction,
Navigation,
Endianness,
Shared memory, Semaphore.

ABSTRACT

Indian Regional Navigation Satellite System is a regional navigation system for Indian Region This system consists of seven satellites each generating navigational message in binary notation at stipulated time based on the data uploaded periodically from the control segment. Each satellite sends data in two bands viz. S and L5. Ground ranging stations receive Range and navigation data from IRNSS satellites. The data received is in properiately packed binary format and needs to be extracted to engineering units for further processing. The measurement data logs are received for every 1 second and the navigation data logs are received at every 12 seconds from each satellite. Since the logs are stored in the little Endian format and the packet headers are in big Endian format, endianness has to be considered for accurate data extraction. These logs after decoding are sent in UDP packets to the receiver which in turn stores the data in the shared memory for its readers. The reader process fetches the data from the memory and stores it to a file.

Introduction

The Indian Space Research Organization (ISRO) aims at developing an independent satellite based navigation system to serve its user community with reliable and continuous positioning and timing service of accuracy better than 20m over India and a region extending about 1500km around India. IRNSS is to be established using the combination of GEO and GSO spacecrafts. The navigational requirements of IRNSS comprises of providing

1. High accuracy real time position, velocity and time for authorized users on a variety of platforms.
2. Good accuracy for a single frequency user with the help of grid based ionosphere corrections.
3. All weather operation, 24hrs a day [1-3].

It has three segments viz. Space segment, Ground segment and User segment. IRNSS constellation consist of seven satellites three in Geostationary orbit (GEO) and four satellites in Geosynchronous orbit (GSO) inclined at 29° to the equatorial plane in the space segment. The navigational message is received from seven satellites periodically. The Ground segment is responsible for controlling the IRNSS satellites and generating the navigation data. One of the important ground elements for IRNSS is the IRNSS Range and Integrity Monitoring Station (IRIMS). IRIMS are dedicated for continuous one way ranging and monitoring of the IRNSS satellites [2]. The IRNSS User community is classified as (Standard Positioning Service) SPS user and Restricted Service (RS) users. In each, user's community will have single and dual frequency users. In each receiver type, there shall be an option for single and dual frequency operation [1, 3].

The navigational data from the satellites are received in binary notation at IRIMS receiver which is dumped in the binary file. The logs in the binary file are in little Endian format [3]. The extraction process is implemented in LINUX operating system, endianness plays a major role. Endianness describes the format in which the numbers are stored in the system memory i.e. whether the bytes are represented from left to right or right to left. The Internet Protocol (IP) defines big-endian as the standard network byte order used for all numeric values, so data should be converted into the little Endian format for accurate data extraction.

The following tasks are accomplished:

1. Extracting logs data from the binary file using IEEE-754 [4] decoding and union concept in c for floating point values considering the endianness in the client process.
2. Creating a shared memory for storing the logs data received through UDP packet in the shared memory in the server process. Establishing the Inter Process Communication (IPC) using Posix shared memory and Named
3. semaphore between the unrelated server and reader processes [5].
4. The reader process access the data from the shared memory and write it to the text files, one text file for each log.

The paper is organized as follows Proposed work in the section 2, system architecture in section 3, Experimental results in section 4 and conclusion in section 5.

Proposed work and system architecture

This section describes the proposed work and system architecture with the process involved.

Proposed work

The binary file from the IRIMS receiver is given as input to the IRNSS Log Data Extraction (ILDE). The binary file contains the packets identified by the packet Header. Each packet contains a log with its log header and data. The logs contain the data received from all seven satellites for two frequency modes. The binary file is extracted and stored in the local structures, one structure for each log. The log structure is stored in the shared memory structure which can be fetched and accessed by other process by establishing the IPC between the processes using the POSIX shared memory. The accessed logs data are written to output file

These processes are named as

Tele:

E-mail addresses: prabadevi.boopathy@gmail.com

1. UDP Sender/Client
2. UDP Receiver/Server
3. The Reader

The UDP Sender/Client: This process extracts the logs in the binary file with reference to the data formats comply WAAS G-III EICD document [6]. The client performs the following functions depicted in Fig. 1.

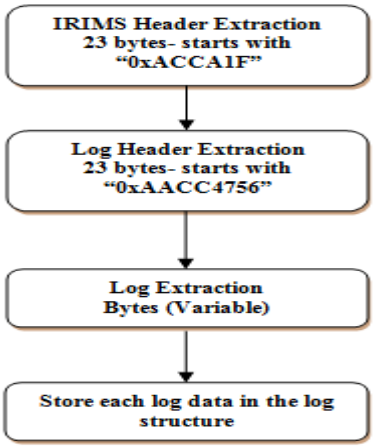

Fig. 1 Activities in the client Process

Each packet is preceded by the IRIMS header. It is identified by 3 byte Hexadecimal synchronization pattern "0XACCA1F" and length of the header is 23 bytes [7].

The IRIMS header is followed by the actual log which is preceded by the Log Header that occupies 23 bytes. The log header can be uniquely identified by the four byte synchronization pattern "0xAACC4756".All logs are protected with a 32-bit CRC at the end of each message [6].

The log is identified by the Message id Field in the Log Header. From the message id the corresponding log is extracted. The extraction of the headers and logs are described in next section. The Table 1gives the extracted logs with their message id and functionality.

The stored structures are written to the UDP socket and then transmitted to the UDP Receiver/client.

Table I

Logs Message id and its frequency of occurrence

Log	Message ID	Frequency of occurrence
Agcinfo	4096	Every second
Almanac	4097	Whenever changed
Cardstatus	4098	Once in every 5 minutes
Corrdata	4099	Every second
Corrlocation	4100	Once in every 5 minutes
Ethstatus	4101	Once every 5 minutes
Measurementdata	4103	Every second
Rawframedata	4104	As soon as it arrives (at present 12 s)
Rxcommands	4105	Whenever changed
Satpos	4106	Every second
Timesolution	4107	Every second
Version	4108	Whenever updated

The UDP Receiver/Server

The UDP Server receives the extracted logs from the UDP Client. The UDP sender creates a POSIX shared memory with 3 objects. The log received from the client is stored to the shared memory object only if the particular block is empty and sets a flag to one indicating that the data is written for the reader process. It checks for the space in the other two blocks if the block one is non-empty block, and returns an error in case of non-empty blocks. The logs received and stored are controlled by POSIX Named Semaphore [5].

The Reader:

The Reader process and UDP server are synchronized using POSIX shared memory and Named semaphores. The reader process locks the shared memory and checks for the log data availability, if the object is full fetches the data sets the flag to zero and unlocks the memory. The fetched logs data is written to the output file.

System Architecture:

The architecture of the proposed work is depicted in the fig 2.

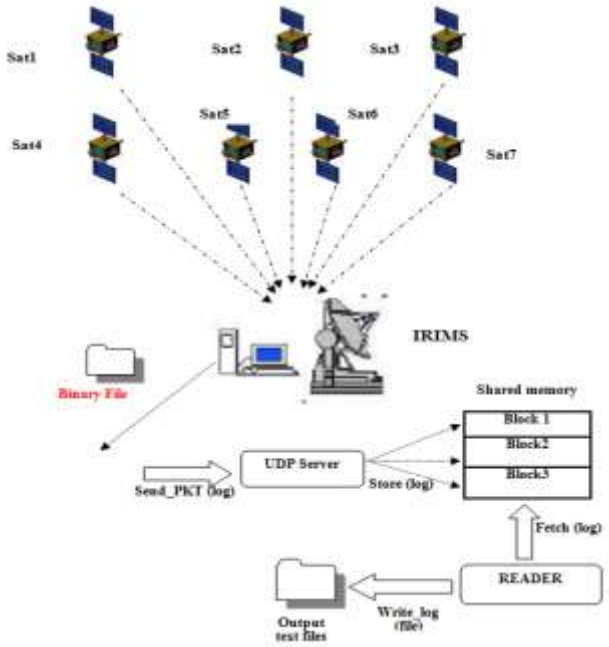

Fig. 2 System Architecture

The processes described in the proposed work are depicted in the fig2.

Implementation

This section describes about the method of extraction of the logs with different data type.

As mentioned earlier the logs are in the little Endian format and the packet headers are in big Endian format, the packet headers can be extracted without changing the endianness.

The little Endian format is changed to standard big Endian by shifting operations and byte reversal. The data types of integer type can be extracted by the byte reversal and shifting operations. The float and double values are extracted using two methods

1. Character arrays and Union
2. IEEE decoding

The character Arrays and Union: The float values and double values are fetched in the character arrays and the Endian conversion is done. This is stored to the character arrays of union and displayed as float and double values of union for float and double values respectively

IEEE decoding: The single and double precision floating point values are decoded as per IEEE-754 format described in the Tables 2 and 3.

Table 2

IEEE Single Precision Format

Fields	Sign	Exponent	Mantissa
No of bits	1	8	23

Table 3

IEEE Double Precision Format

Fields	Sign	Exponent	Mantissa
No of bits	1	11	52

The steps involved in converting the Hexadecimal floating point value to floats and doubles are as follows

1. Convert each Hex value to binary
2. Convert the binary value to Binary Scientific Notation (BSN) i.e. Sign, Exponent, Mantissa fields
3. Convert BSN value to decimal equivalent

These are performed using functions.

Results and discussion

The system is implemented with the binary data from the seven satellites and one reference receiver. The output text files are validated using the actual logs data received from the IRIMS station. The output screen shots for few sample logs are shown in the fig 3 and 4

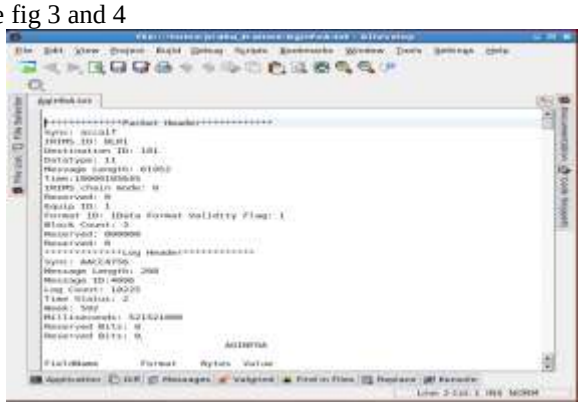

Fig 3. Agcinfo log with packet header and log Header

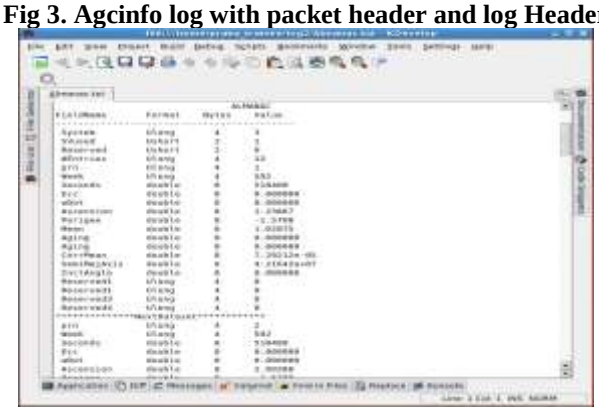

Fig. 4 Almanac log with its log data

The log summary file contains the count of number of each log in the binary file. The sample screen shot for this binary file is shown in the fig. 5.

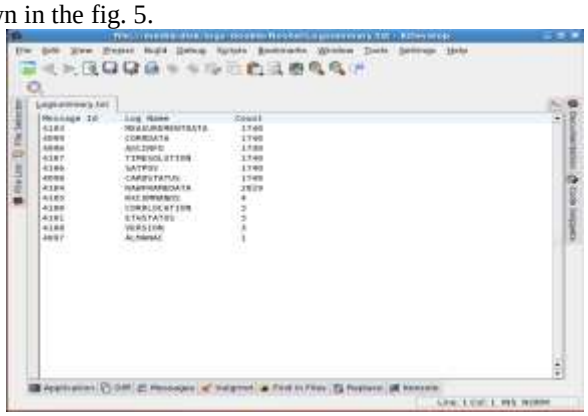

Fig.5 log summary file

Comparison between two methods of implementation: the data from the above two methods are validated manually and found to be same values.

Conclusions and Future work

The architecture for the IRNSS Logs Data Extraction is implemented. The IPC is established between the processes UDP Server and the Reader Process using the POSIX shared memory and POSIX Named semaphores. The result is validated with actual logs data.

The future enhancement is to perform the extraction at server process with more number of blocks and data from 17 IRIMS Stations.

References

[1] IRNSS Project Report, 2004
[2] IRNSS Ground Segment PDR Document, June 2008
[3] IRNSS Navigation Software Design Document, Dec 2011
[4] *Floating point numbers*, IEEE Std. 754, 2008.
[5] *Stevens W.Richard, Unix Network Programming, Vol 2, 2nd* ed., Prentice Hill, 1999.
[6] *WAAS G-III External Interface Control Document, D14799,* Novtel, 23 March 2011.
[7] *IRIMS Reference Receiver Interface Control Document*, 2011