Assignment 03: Report

The purpose of this assignment was to design and implement an email system event modeler and intrusion detection system, implemented in C, C++, or Java.

# Initial Input

The program accepts either 3 or 4 arguments when attempting to run it from the command line. The command to run the program is:

IDS <events file> <stats file> <days> <(optional) username>

The events file and stats file are self-explanatory. **Days** is used to generate the appropriate number of entries in the log file, and the **username** is the name of the person the statistics is based on. If the username field is left out, the system will default to using “Peter” as the user. Thus, the completed command, with proper arguments looks like this:

IDS Events.txt Stats.txt 1000 Mark

## Storage

The Events.txt and Stats.txt have specific formats that have to be followed. This is the Events.txt file:

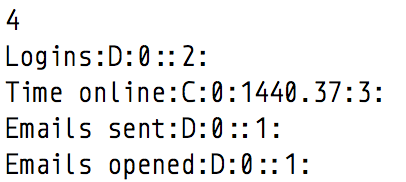


Figure : The Events.txt file

The first line is merely a counter, which shows the number of entries in the file. Each line after that is an event, with the following format:

<Event Name>:<Continuous(C) or Discrete(D)>:<Min>:<Max>:<Weight>:

A continuous event is one where it occurs over a period of time *(i.e. time online)*, whereas a discrete event is one where each occurrence is a separate, or discrete, event *(logging in, sending emails, etc)*.

There is also a minimum and maximum occurrences allowed, and the weight, which is used to determine if there are any anomalies.

The stats file is as follows:

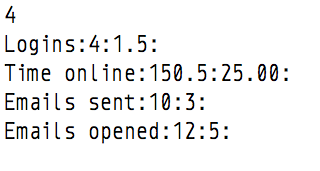


Figure : The Stats.txt file

Similar to the event file, the first line is a counter, with the following lines of the format:

<Event Name>:<Mean>:<Standard Deviation>:

The mean and standard deviation are derived from a set of event files, with the relevant calculations performed on them.

We have created Stat, Event and Log classes to help facilitate the storage of the information. The Event class is the base class for the DiscreteEvent and ContinuousEvent classes. After reading the information from the events and stats files, we will dynamically create instances of those classes and store them into a vector.

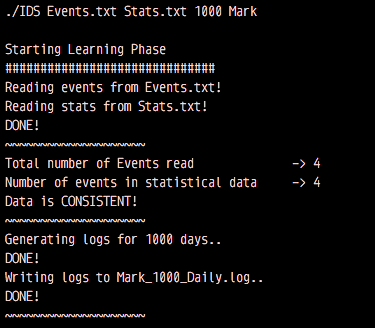


Figure 3: The output after the LEARNING PHASE is completed

## Data Inconsistency or Missing Files

In the event that the statistics and events file are not consistent *(i.e. the number of entries do not tally)*, or any of them are missing, the system will alert the user and will not proceed.

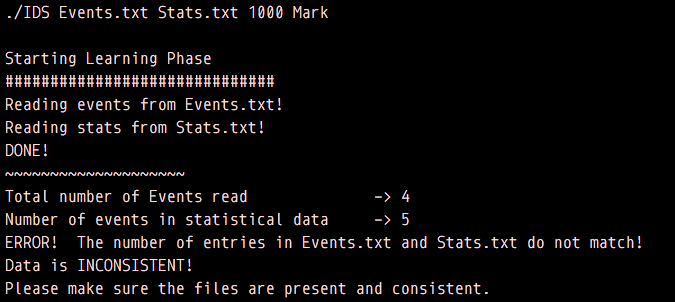


Figure 4: The system will alert the user and exit if the files are missing or inconsistent with each other.

The system will also alert the user and not proceed if it finds that the information in the events and statistics file do not tally *(i.e. different names, or the names do not match)*:

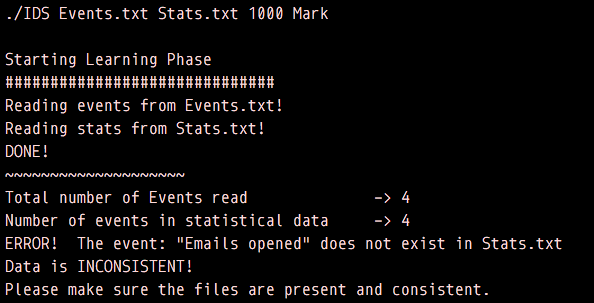
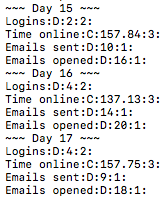


Figure 5: The system will also alert the user if the contents in each file are inconsistent.

After this is done, the program will use the **days** and **username** argument provided in the command used to start the program to generate a series of log entries.

The log entries are of the format:

<Event Name>:<C or D>:<Value>:<Weight>:

These log entries will be used to generate a new statistics file which will serve as the baseline for the comparison of data in the next part. Using these two arguments, it will create a file called <username>\_<days>\_Daily.log, which will contain the aggregated log entries for each day, as shown on the right.

Figure : A sample of the Daily Log's output, generated using the Marsaglia Polar Method.

# Activity Engine and Reading Logs

The activity engine reads the logs generated in the previous step. This is probably not done in a real IDS system, where the information would be captured and stored through actual usage. In this simulation, however, we generate the logs randomly because we don’t have any actual information.

Using these logs, the program can generate a new statistics file, which will be used as our baseline to determine if the entered information is anomalous or not. The new statistics file will be