PseudoCode:

Telemetry.h

SHM_NAME “/telemetry_shm “

MAX_EVENTS - 2000
MAX_NODES - 4096
MAX_SENSOR - 64

STRUCT EventNode
Event_id : uint64_t
Sensor_id : uint32-t
Next_index :int
Severity : uint8_t
Timestamp : uint64_t
Value : double
END STRUCT

STRUCT Sensor Stats
Count : int
Average Sum : double
Min value : int
Max value : int
MAx severity seen : int
End STRUCT

STRUCT TelemetryStats
Total events
Overall max severity
All sensor stats [MAX Sensors]
END STRUCT

STRUCT Shared Memory:

Read write locks
Allocation mutex
Shutdown_ flag : bool
Head_idx (oldest event)
Tail_idx (newest event)
Free_head
Event count
Stats
Nodes [MAX _NODES]
END STRUCT

Start generator process

DEFINE all header files

Open the file telemetry.bin - in binary mode

Create new event record STRUCT
  Event id
  Sensor id
  Timestamp
   Value
   Severity
END STRUCT

FUNCTION now_ns that return system time to nano seconds
  Get current time in seconds and nanoseconds
  Convert seconds to nanoseconds and add nanoseconds part
  Return total time in nanoseconds

Main program

OPEN file "telemetry.bin" in append + binary mode

```
IF file open fails THEN
    print error message


    Exit the program
END IF


Initialize event id counter to 0

Generate random number with current time ,

Loop forever
 Create a new Event Record    r

 Set
   r-> event id  = current even id
   Event id ++

Generate random sensor id number

  R.sensor id  =  (rand()% 64 )+1

 Set timestamp

R.timestep   = now_ns()

Generate value
   R.value  = generate random value

Generate  severity
 R.severity = generate random severity (range 0 to 5)
     (  0- less severe
      5 - most severe)

Write the event record (R)  to file in binary format

Fflush
```

Generate record and append to the file every 10 millisecond
Sleep (10000)

Close the file

2. Analyzer Process

Create new event record STRUCT
Event id
Sensor id
Timestamp
Value
Severity
END STRUCT

Shared memory * shm (pointer to the shared memory region)

Init memory (create shared memory , allocate size , map it into the address space of the process)

```
FUNCTION init_shared_memory
    fd = shm_open(SHM_NAME, CREATE | READ_WRITE)
    set size of shared memory to sizeof(SharedMem)
    shm = mmap shared memory

    initialize RW lock as PROCESS_SHARED
    initialize mutex as PROCESS_SHARED

    set list head = -1
    set list tail = -1
    set list count = 0
    set shutdown = 0

    // Initialize free-list
    free_head = 0
```

```
    FOR i = 0 TO MAX_NODES - 2
        nodes[i].next_idx = i + 1
    END FOR
 nodes[MAX_NODES - 1].next_idx = -1
END FUNCTION


FUNCTION alloc_node
    lock allocator mutex

    idx = free_head
    IF idx != -1
        free_head = nodes[idx].next_idx
    END IF

    unlock allocator mutex
    RETURN idx
END FUNCTION
```

Function free -

Function free node (index)

Use allocation mutex to lock this

Node[index]. Next_index = free head

Free head = index

Unlock allocation mutex

End function

Begin analyzer
Call init_shm()

Open telemetry file in read mode ,

See to end of the file (ignore the old data)

While shutdown flag is false

Read one event record from the file

If full record not read
    Sleep for 10 ms
 Continue

Acquire write lock on shared memory

Index = alloc_ node()

If index != -1

 Copy event data into nodes[index]

nodes[index] .next_idx = -1

If tails exists
  Tail.next_idx = index
Else
Head_idx = index

Tail_idx = index

Count ++

If count > MAX EVENTS
 Old =head_idx

Head_idx = nodes[old].next_idx

Count - -

free_nodes(old)

UPDATE statistics
Total event ++

Sensor stats [sensor id].count++

Update min / max / sum

Update per sensor max severity

Update overall max severity

Last event id = event id

Last timestamp = timestamp

Release write lock

End while loop

End analyzer

3. Monitor

Open the telemetry file in read mode

Wait until Shm is created
Attach to shared memory

Fd = obtain shm memory open_shm(SHAREDmem name , O_CREAT | Read | Write)

If fd < 0
  perror (“ failed to open)
 exit(1)

Shm = mmap(SHAREDmem)

While (! shutdown)
   Read write lock in read mode

  Print the total events ….

 Print overall max severity ..

Loop from 0 to MAX SENSOR(64) - 1

   If the sensor count is greater than 0
      Calculate the average . which will be sum = sum /count

   Print the sensor ID for each sensor
  Print the event count
  Print the average value , sum

Loop ENds

Release the read write lock .

Sleep for 500 milliseconds

End while loop
End the monitor process