

Fake Data Studies

- fake_seb has successfully read a mBooNE style uncompressed data file and shipped events over sockets to fake_assembler. Glenn and Eric each have done this. Glenn did it on 2 machines: traffic across network for real, demonstrating faster-than-required data movement to an output file.
- Now, I can report I've done this with a compressed uBooNE data file format. So, I altered Georgia's Root Macro to write fake-Huff coding of events of varying size (in principle). Created file. Fired up fake_seb, fake_assembler and read up evts, shipped, wrote data to output file.
- => Will cross this small task off the list.

uBooNE DAQ SEB code design

- There is nice legacy mBooNE C code that looks suitable, elegant, not out-of-date. I propose to re-use it. Main difference being at the lowest level where the events come into the process memory.
- We will run 2 parent seb processes: one for SN data, one for triggered data. They look the same in almost every respect.
- 2 threads per seb process: getter and sender.

uBooNE DAQ SEB code (2)

- getter in a while(FOREVER) walks through the memory as new memory is filled, increments a newEvent semaphore and packages up digitized signals into a buffer of events.
- sender in its own while(FOREVER) decrements newEvent semaphore when it's available, grabs pointer to latest event, sends it over sockets to assembler (file) for triggered (supernova) data. Decrements sentEvent semaphore.
- I'm inclined to add a third thread that fills the memory: Nevis's code bundled up in its own thread.

uBooNE DAQ SEB code (3)

sem_post,wait
=> give,take a semaphore.

seb.c

```
{
extern sem_t newData,eventToAssembler,newDMA;
extern Event, p_rec;

...
pthread_create(&sender,...)
pthread_create(&getter,...)
pthread_create(&poller,...)
}
```

sender

```
{
while()
{
sem_wait(&newData);

fd=disk OR sockets

write(fd, Event, ...)
sem_wait(&eventToAssembler);
}
}
```

getter

```
{
while()
{

sem_wait(&newDMA);
while(ind<DMA.size())
{
sem_post(&newData);
ind++;
while( t<9600)
{
Event+t->ADC = p_rec+ind+t->ADC;
}
}

sem_post(&eventToAssembler);
}
}
```

poller

```
{
while()
{

// Reserve space, DMAlock p_rec
DMA(p_rec, ...);
sem_post(&newDMA);

}
}
```