ROOT TTrees and Branching

Or

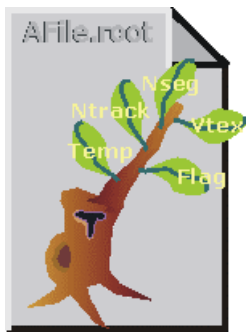
How do we want to organize our ROOT files?

What is Branching?

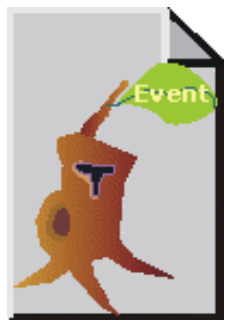
- As the name implies, branching structures a TTree into a hierarchy.
- This is an option – controlled by the split level specified when writing a TTree.
  - 0 denotes no branching
    1 denotes branching applied to the 1st level
    2 denotes branching applied to the 1st and 2nd
    etc up to 99

Why Branch?

- While not required, branching allows the user more control over what is read in at analysis time.



Branching



No Branching

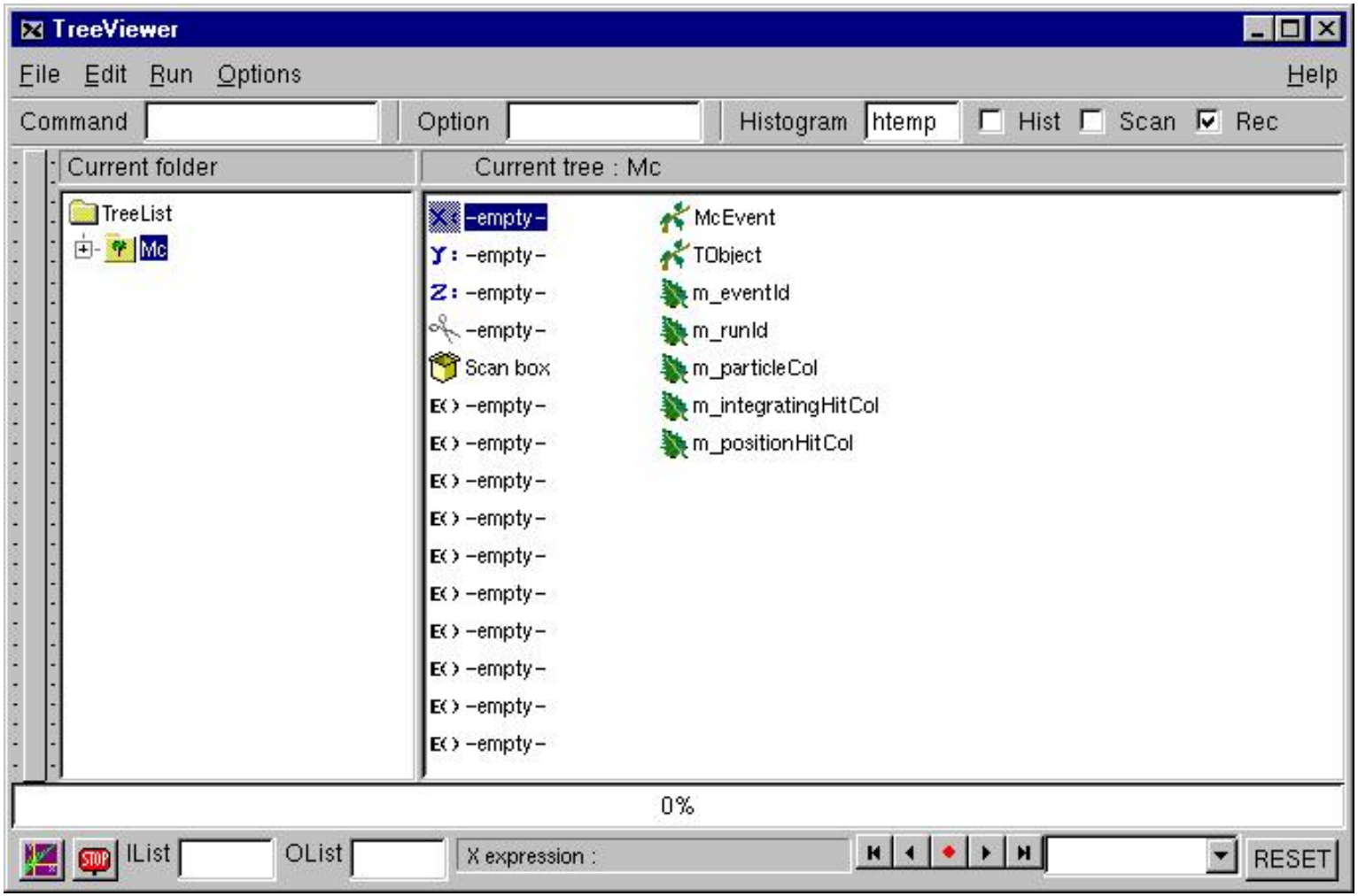
For example:

```
root [16] TFile f("/local/data/ROOT/bfem/nsbf_r000053_20010804_072159_ivte-raw$
root [17] DigiEvent *evt = new DigiEvent()
root [18] TTree *tDigi = (TTree*)f.Get("T")
root [19] tDigi->SetBranchAddress("DigiEvent", &evt)
root [20] tDigi->SetBranchStatus("*", 0)
root [21] tDigi->SetBranchStatus("m_eventId", 1)
root [22] tDigi->GetEvent(0)
(Int_t)4
```

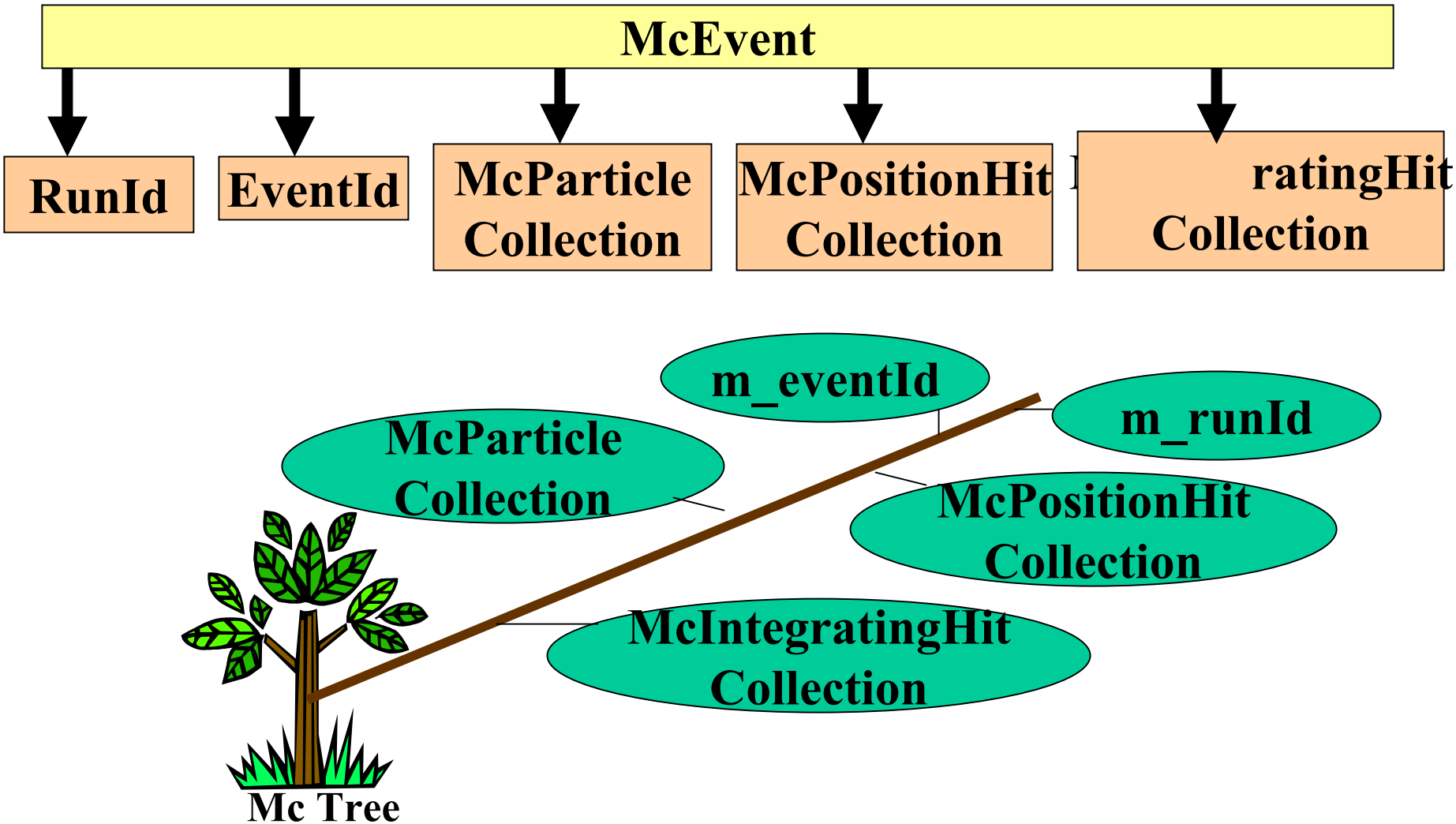
We can specify what parts of the data we want read in.
In this case, we only read in 4 bytes, corresponding to the event id

MC TTreeViewer

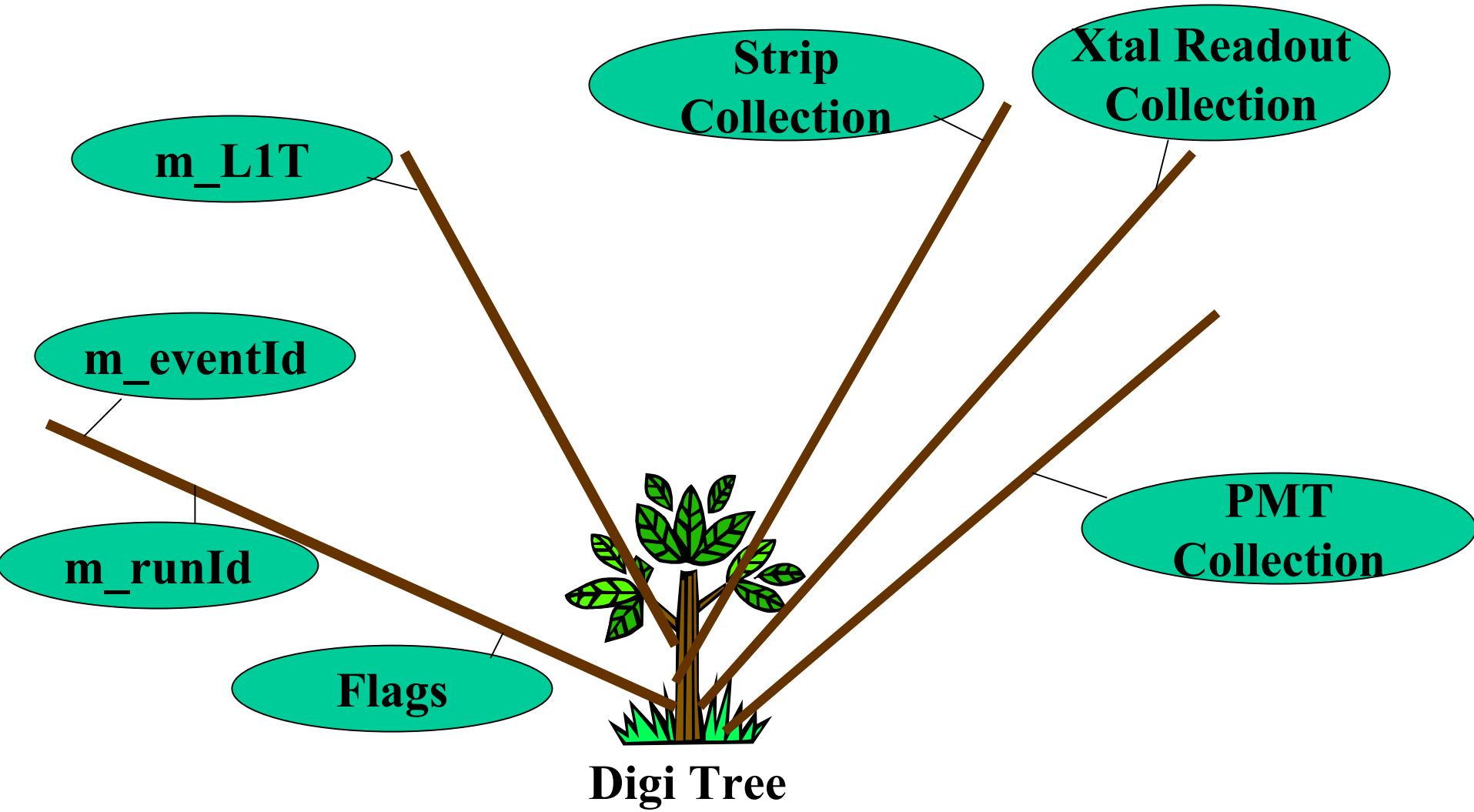
Split level 2



MC Logical Organization



Potential Digi Branching



Potential Recon Branching