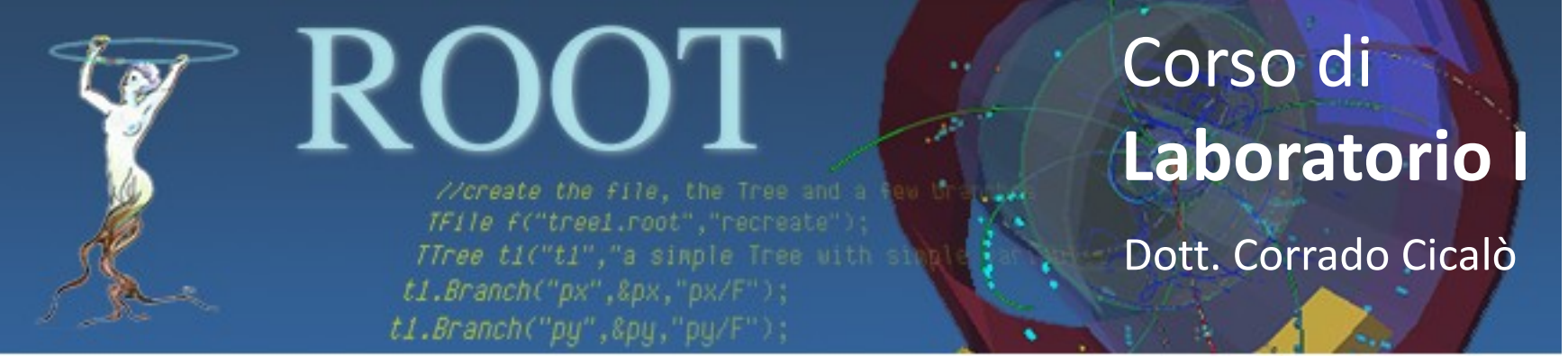


Corso di
Laboratorio I

Dott. Corrado Cicalò

Laurea Specialistica

a.a. 2009/2010

Introduzione all'analisi dei dati sperimentali con il framework ROOT

Lezione 3: File di input, File di output

Antonio Uras

Università di Cagliari & INFN

Sommario

- Aprire e leggere un file di testo
- File ROOT di input
- File ROOT di output

Aprire e leggere un file di testo

Aprire un file di testo in input

Può capitare (capita spesso!!!) che i risultati delle misure vengano scritti su file di testo prima di essere analizzati

Problema: come **aprire** e **leggere** un file di testo all'interno di una macro di ROOT?

Per l'**apertura** del file basta creare una viariabile di tipo ifstream, con il comando:

```
ifstream  myFile("nome-del-file") ;
```

Per la **lettura** dei valori contenuti nel file, occorre conoscere come sono distribuiti all'interno del file!!! Consideriamo solamente il caso più comune: nel file ci sono **N righe**, ciascuna contenente **M valori** separati da spazi

Leggere un file di testo in input

Supponiamo di voler leggere una riga in cui siano scritti due valori decimali, separati da caratteri vuoti (spazi). Allora dobbiamo:

1) Dichiarare due variabili Double_t

```
Double_t  var1,  var2 ;
```

2) Leggere la riga del file (già aperto nella slide precedente) con il comando

```
myFile  >>  var1  >>  var2 ;
```

oppure:

```
myFile  >>  var1 ;
myFile  >>  var2 ;
```

Leggere un file di testo in input

Per leggere un **intero file**, occorre leggere ciascuna riga con il comando visto nella slide precedente

E' possibile usare vari metodi, due sono quelli più semplici. Se il file ha N righe, ciascuna con 2 valori:

METODO 1

```
Double_t  var1,  var2 ;
for (Int_t i=0;  i<N;  i++)  {
      myFile  >>  var1  >>  var2 ;
}
```

METODO 2

```
Double_t  var1,  var2 ;
while ( !myFile.eof() )  {
      myFile  >>  var1  >>  var2 ;
}
```

Ad ogni iterazione, le variabili var1 e var2 assumeranno rispettivamente il primo e il secondo valore riportati nella (i+1)esima riga del file di testo. **All'interno del ciclo**, queste variabili possono essere usate per l'analisi (ad esempio per riempire un istogramma!)

Quante righe ha un file di testo?

Per usare il codice della nella slide precedente (metodo 1) abbiamo bisogno di conoscere il numero di righe di cui è composto il file di testo in input

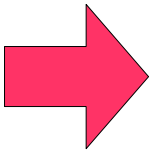
Non c'è un metodo diretto, un possibile modo consiste nell'usare una serie di istruzioni che è conveniente inglobare in un unico metodo:

```
Int_t  GetNLines(Char_t *nameFile)  {
      gSystem  ->  Exec(Form("cat %s | wc -l >> temp.temp", nameFile)) ;
      ifstream myFile("temp.temp") ;
      Int_t nLines ;
      myFile >> nLines ;
      gSystem->Exec("rm  -rf  temp.temp") ;
      return nLines ;
}
```

Input/Output con file di ROOT

Input/Output con file di ROOT

I file in formato .root NON servono per scrivere del testo



Potete vederli come **CONTENITORI** per oggetti appartenenti a classi di ROOT (nel nostro caso, istogrammi e grafici)

La gestione dei file di ROOT avviene tramite la creazione di oggetti di classe TFile
Dovremo imparare come:

- definire un oggetto TFile per operazioni di **output** (scrittura)
- definire un oggetto TFile per operazioni di **input** (lettura)
- scrivere su un oggetto TFile di **output**
- leggere da un oggetto TFile di **input**

Usare i file ROOT di output

Immaginate di aver creato un oggetto **myHist** di classe TH1D (istogramma) all'interno di una macro. Oltre a disegnarlo sul momento, potreste volerlo **salvare** per utilizzarlo successivamente. Come fare?

1) Definite un oggetto TFile per operazioni di **output**:

```
TFile *fileOutput = new TFile("nomeFile.root", "recreate") ;
```

2) Scrivete l'oggetto **myHist** sul file appena aperto:

```
fileOutput -> cd() ;

myHist -> Write() ;
```

Due commenti:

- l'opzione “recreate” fa in modo che il **file fisico**, se esiste, venga **ri-creato**!!!
- fileOutput -> cd() ; serve nel caso siano stati aperti diversi file di output

Usare i file ROOT di input

Immaginate di aver salvato un oggetto di nome **myHist** e di classe TH1D (istogramma) nel file nomeFile.root

Per poter ri-utilizzare l'istogramma all'interno di una macro (per esempio, per modificarne il contenuto o solamente disegnarlo) dobbiamo:

1) Definite un oggetto TFile per operazioni di **input**:

```
TFile *fileInput = new TFile("nomeFile.root", "read") ;
```

2) Leggete l'oggetto **myHist** dal file appena aperto:

```
TH1D  *myHist  =  (TH1D*)  fileInput -> Get("myHisto") ;
```

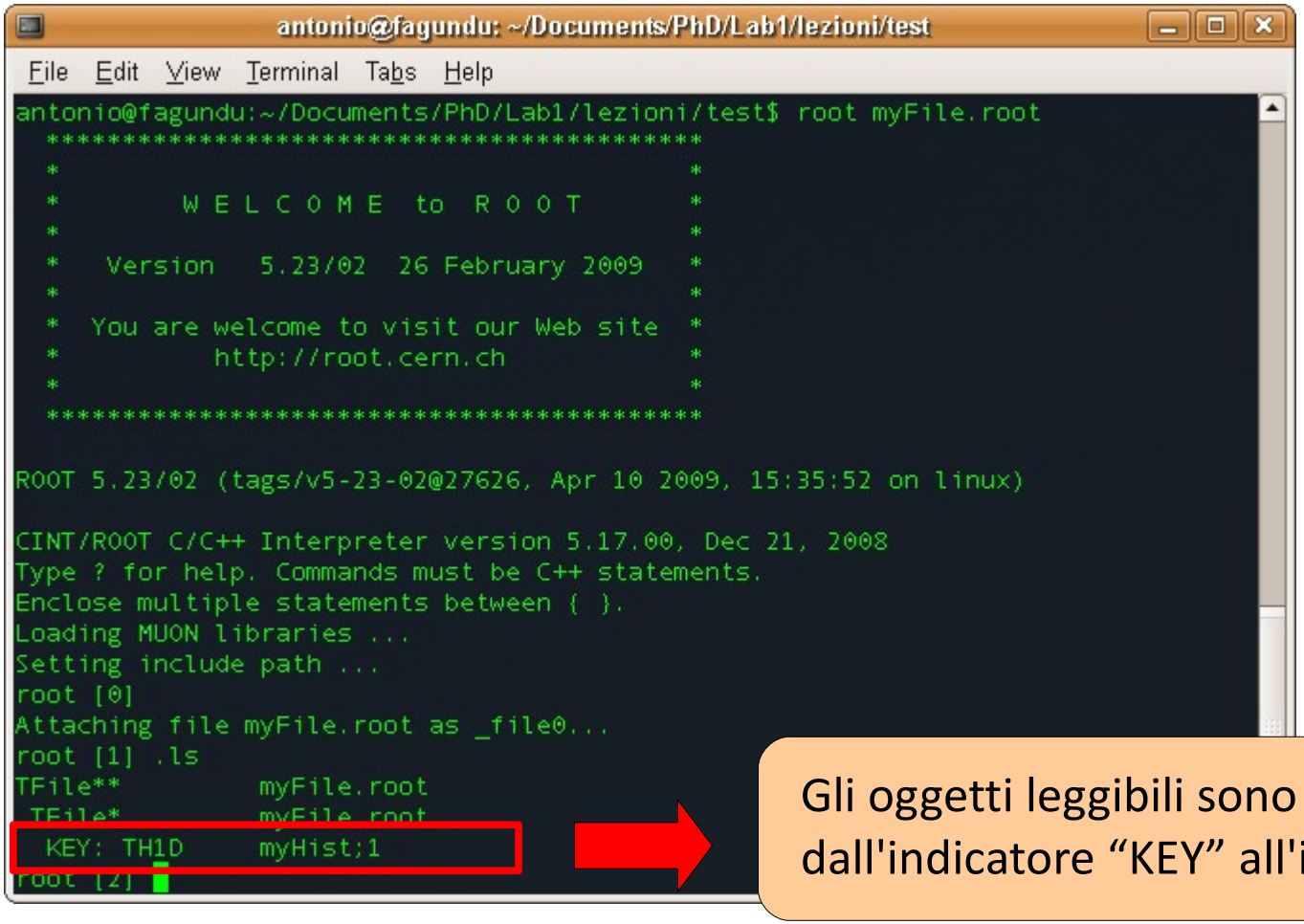
A questo punto, potete utilizzare myHist per qualsiasi operazione all'interno della macro

Input/Output con file di ROOT: commenti sparsi

- L'opzione “**recreate**” è comoda ma potenzialmente pericolosa: se il file esiste già, lo sostituisce con quello nuovo
- L'opzione “**read**” impedisce che il file aperto possa essere modificato
- Se cercate di aprire un file inesistente, avrete errori cercando di utilizzare l'oggetto. Per verificare l'effettiva apertura del file, utilizzate il metodo **IsOpen()** che restituisce un valore Booleano 0 o 1 (falso o vero)
- Se non specificate l'opzione di aprtura, viene usata come **predefinita** l'opzione “**read**”
- Altre opzioni di apertura: “**update**” (se il file esiste, ne aggiorna il contenuto; se non esiste, lo crea), “**create**” o “**new**” (se il esiste NON lo apre; se non esiste, lo crea)

Appendice: visualizzare il contenuto dei file ROOT

Per leggere gli oggetti scritti nei file di input, dovete conoscerne la classe e il nome. Se non li conoscete, potete visualizzare al volo il contenuto del file usando l'interprete C++ di ROOT:



Gli oggetti leggibili sono identificati dall'indicatore “KEY” all'inizio della riga

Backup Slides