



# Introduction to Linux

Anton Lechner

**Geant4 Training Course**  
Oak Ridge National Laboratory  
May 19<sup>th</sup>, 2008

# What's Linux?

- **Unix-like operating system**
  - **Open source**
  - Many flavours exist (Fedora, SuSE, Ubuntu, ...)
  - Common to all: Linux Kernel ([www.kernel.org](http://www.kernel.org))
- Where can I get it?
  - Download it from the web
  - Order a CD or DVD from a Linux distributor
- How much does it cost?
  - **It's free**
  - Enterprise versions exist: Pay for user support



# *I. Basic Concepts*

# Users and Groups

- Linux is a **multi-user** system
  - Several users can have an account (each user has a **user name** and a **password** (required at login))
  - Each user has his/her own working environment
  - User with special permissions: root
    - System administration (system software installation, user account management, ...)
- Users can be member of one or more **user groups**
- **File operations**: Users do not necessarily have the same rights
  - **Security and confidentiality**

# Directories and Files (1/2)

- Directory path: directory names separated by “/”
  - E.g.: `/home/alechner/Documents`
- Directory and file names:
  - Case sensitive: File “`MyDocument.tex`” is not “`mydocument.tex`”
  - Hidden files start with a “.”;
    - But: a single “.” indicates the current directory and “..” the parent directory
- Hierarchical directory structure:
  - Root directory: `/`
    - Do not confuse the root user with the root directory

# Directories and Files (2/2)

- Hierarchical directory structure (cont.):
  - All directories appear under the root directory
  - Home directory of user “xyz”: `/home/xyz`
- File permissions:
  - Linux distinguishes **3 types of file permissions**:
    - “**Read**” (r), “**Write**” (w) and “**Execute**” (e)
  - Each type of permission is defined for **3 sets of users**:
    - The **owner** of the file (u)
    - The **group** the owner belongs to (g)
    - **Other** users (o)
  - Since Linux treats directories as files, the same permission schema applies



## *II. Basic Tool: The Shell*

# What is the “Shell”?

- User Interfaces (UI) in Linux:

- Graphical User Interface (GUI)
- Command Line Interface (CLI)

- The shell is a CLI:

- It takes your commands from the keyboard,
- interprets and executes them (**command interpreter**)

- Various different shells exist:

- Most common: **bash** (Bourne Again SHell)
- **csh** (Berkeley UNIX C shell)
- ksh, zsh, ...

```
Dateisystem      Größe Benut  Verf Ben% Eingehängt auf
/dev/sda4        18G   18G  276M  99% /
udev             505M  164K  505M   1% /dev
/dev/sda7        19G   18G  488M  98% /home
/dev/sda2        20G   18G  1,9G  91% /windows/C
/dev/sda5        11G   1,1G  9,3G  11% /windows/D
anlech@linux-gowo:~/geant4/tmp/Linux-g++>
anlech@linux-gowo:~/geant4/tmp/Linux-g++> du -sh
298M
anlech@linux-gowo:~/geant4/tmp/Linux-g++> cd..
anlech@linux-gowo:~/geant4/tmp> cd..
anlech@linux-gowo:~/geant4> cd
bin/ tmp/
anlech@linux-gowo:~/geant4> cd bin/
anlech@linux-gowo:~/geant4/bin> ls
Linux-g++
anlech@linux-gowo:~/geant4/bin> cd Linux-g++/
```



# Terminal emulators

- Terminals allow for the interaction with shells
- “Terminal emulators” in Linux
  - Enable CLI's in a desktop environment
- Commonly used terminal emulators are:
  - **xterm**
  - konsole
  - gnome-terminal
  - ...



# Secure SHell (SSH): Remote access

- A Secure SHell (SSH) provides a convenient way to **access a shell remotely**
  - A user can log into the remote machine and execute commands, access files, ...
  - The session is **encrypted** (high security)
  - Allows **X session forwarding**
- On the remote machine a **SSH deamon** must be running
  - Listens by default on port 22
- The user needs an **SSH client**, and a **X Window server** (if X11 connections are forwarded)



# *III. Using the shell*

# Shell commands (1/2)

- General syntax for executing programs from the command line:
  - **<Program Name> [OPTIONS] [FILES]**
    - **<Program Name>** is the name of the program
      - Example: `ls`
    - **[OPTIONS]** are options specific to the program, that may modify the program's behaviour
      - Options start with a “-” (short form) or a “--” (long form) character:
        - Example: `ls -s` (is equal to `ls --size`)
      - Options may be summarized:
        - Example: `ls -l -h` is equal to `ls -lh`
    - **[FILES]** are files or directories the program is applied on
      - Some programs require files and/or directories as arguments, some don't require them but allow them, some don't allow them
      - Example: `ls -l myfile.dat`

# Shell commands (2/2)

- Each Unix/Linux program has **two output streams** opened for it when execution starts
  - They are: Standard output (called **stdout**) and standard error (called **stderr**)
  - These are usually attached to the terminal, *i.e. output/error messages are printed in the terminal*
- Commands for **output stream redirection**
  - Using “**1>**” and “**2>**” (at the end of the command), one can save the stdout and stderr to files
  - Example: **ls 1> out.txt 2> err.txt** (redirects the stdout to out.txt and the stderr to err.txt)

# Shell commands: Examples (1/6)

- Files and directories
  - Listing files and directories (and their attributes):
    - `ls`, `ls -l`, `ls -a`, `ls -h` (or combinations of options)
  - Changing to subdirectories:
    - `cd destinationdir`
  - Creating new directories:
    - `mkdir mynewdir`
  - Removing files:
    - `rm myfile`
  - Removing (empty) directories:
    - `rmdir mydir` (for not empty directories use: `rm -r mydir`)

# Shell commands: Examples (2/6)

- Files and directories (continued):
  - Copy files and directories
    - `cp myfile myfile2`
    - `cp -r mydir mydir2` (note the “-r” option for copying directories)
  - Move files and directories
    - `mv myfile destination` (“destination” is either a file or a directory)
    - `mv mydir destination` (“destination” is a directory)
    - NOTE: If the destination file or directory “destination” does not exist, the move command basically acts as rename command.

# Shell commands: Examples (3/6)

- File content
  - Concatenate files and print content on stdout:
    - `cat myfile1 myfile2`
  - Show file content in terminal:
    - `less myfile` (scroll with arrows, quit program by typing “q”)
  - Print lines matching a pattern:
    - `grep “my text” myfile` (prints all lines of the file “myfile” to stdout which contain the string “my text”)



# Shell commands: Examples (4/6)

- Archives and compressed files:
  - Archiving files and directories:
    - `tar cvf myarchive.tar myfile1 mydir2 ...` (the files and directories `myfile1`, `mydir2`,... will be archived in the file `myarchive.tar`)
  - Extracting files and directories from archives:
    - `tar xvf myarchive.tar` (extracts files from archive `myarchive.tar`)
  - Compressing and decompressing files
    - `gzip file1` or `bzip2 file1` (both cmd's compress file but in different formats; the resulting files are `file.gz` and `file.bz2`)
    - `gzip -d file1.gz` or `bzip2 -d file1.bz2` (decompresses file)

# Shell commands: Examples (5/6)

- Archives and compressed files (cont.):
  - Archiving files/directories and compressing archive (in **.gz** format) in one command:
    - **tar czvf comprarch.tar.gz myfile1 mydir2 ...** (the files and directories myfile1, mydir2,... will be archived and the archive will be compressed into file comprarch.**.tar.gz**)
      - Append **.tar.gz** to name of compressed archive to specify file type
  - Extracting and decompressing **.tar.gz** files with one command:
    - **tar xzvf comprarch.tar.gz**
  - Similarly, to archive files and (de)compress the archive in one line, but for the **.bz2** format, just use the options “**j**” instead of “**z**”

# Shell commands: Examples (5/6)

- ssh and scp:
  - For logging in remotely, one can use the OpenSSH ssh client:
    - `ssh userXYZ@lxminus.cern.ch` (specify your user name on the remote machine (here: userXYZ) and the hostname (lxminus.cern.ch) to login remotely; **as the next step you are prompted for the user password**)
  - Copying a file from a remote location to your machine:
    - `scp userXYZ@lxminus.cern.ch:/home/userXYZ/file1 file1` (copies file “file1” located in the directory /home/userXYZ on the remote machine lxminus.cern.ch to the current directory)
    - If you change the order of the arguments you can copy the local file to the directory on the remote machine:  
`scp file1 userXYZ@lxminus.cern.ch:/home/userXYZ/file1`

# File permissions (1/2)

- To list the permissions of a file, execute `ls -l`
  - A line of the output typically looks like:
    - `-rwxr--r-- 1 anlech users 3,8K 2007-06-10 9:52 retriever.pl`
  - The 10 initial characters “`-rwxr--r--`” mean:
    - The first character denotes the **file type**:
      - `d` for directory, `l` for link and `-` for regular file
    - The next nine characters are the permissions for **owner**, **group** and **others**:
      - For each, three consecutive symbols indicate the **read**, **write** and **execute** permissions (in this order)
      - A “`-`” means permission denied
      - E.g. “`-rwxr--r--`” means that the owner has all permissions, while members of his/her group and all others only have read permissions

# File permissions (2/2)

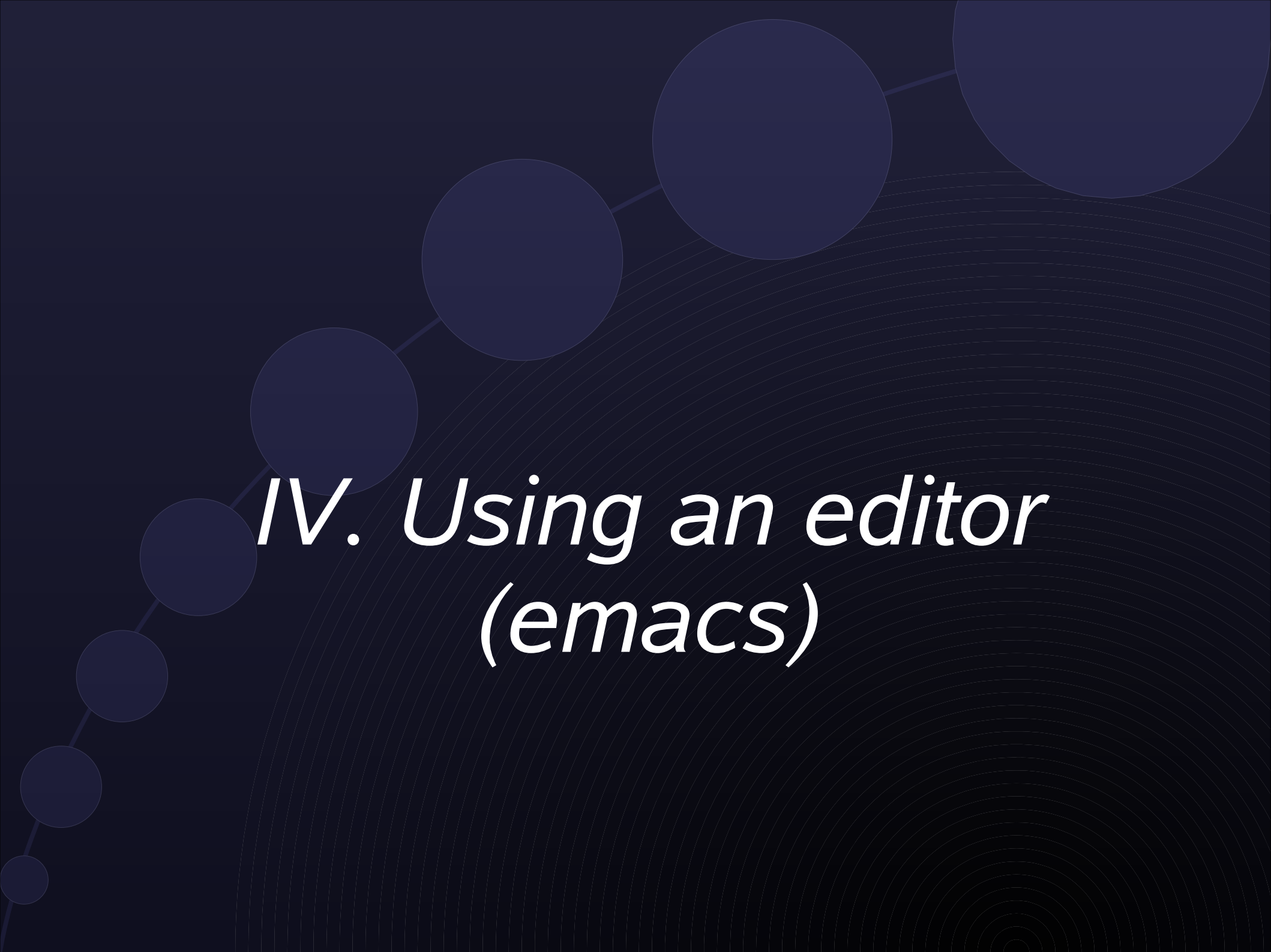
- To change the permissions of a file, use the command **chmod**:
  - Usage: **chmod [options] <file/directory name>**
  - Examples:
    - **chmod g+rx <file name>** allows the group to read and execute the file
    - **chmod a-r <file name>** denies anybody to read the file (incl owner)
    - **chmod go-r <file name>** denies group and others to read the file
    - **chmod o+x <file name>** makes the file executable for owner

# Environment variables (1/2)

- Environment variables (EV) are **named objects**, that can provide **information to applications**
  - EV have a **name and a value**, where names are usually in capital letters
  - Use “\$” as prefix to the variable name to extract the value of an EV
- Listing environment variables
  - Use the shell command **printenv** to list all defined EV
    - The output shows: **<Variable name>=<variable value>**
  - To show the value of a specific variable use: **echo \$<variable name>**
    - E.g. **echo \$PATH** (where “PATH” is the variable name)

# Environment variables (2/2)

- Setting environment variables (commands may differ among types of shells):
  - bash: `export <variable name>=<variable value>`
    - E.g. `export MYHOMEDIR=/home/xyz`
  - c-shell: `setenv <variable name> <variable value>`
    - E.g. `setenv MYHOMEDIR /home/xyz`
- Note:
  - If you set an EV in your shell and you close the shell the variable is not remembered in further sessions

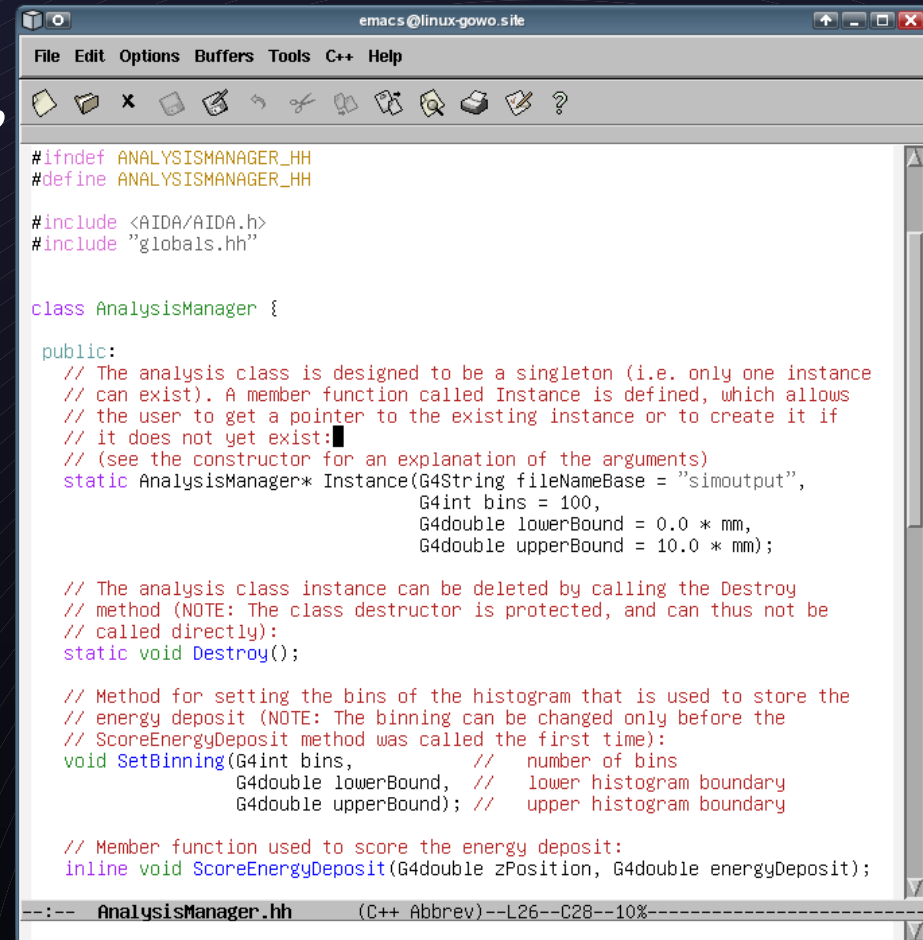


*IV. Using an editor  
(emacs)*



# Manipulating file content

- You can use an **editor**, e.g. **emacs**, to read and manipulate the content of a file.
- To open a file with emacs, type in your shell:  
**emacs <file name>**
- To **copy & paste** text:
  - **Mark text** you want to copy and use the **2<sup>nd</sup> mouse button** (i.e. button in the middle) to paste it at another position



```
emacs@linux-gowo.site
File Edit Options Buffers Tools C++ Help

#ifdef ANALYSISMANAGER_HH
#define ANALYSISMANAGER_HH

#include <AIDA/AIDA.h>
#include "globals.hh"

class AnalysisManager {
public:
    // The analysis class is designed to be a singleton (i.e. only one instance
    // can exist). A member function called Instance is defined, which allows
    // the user to get a pointer to the existing instance or to create it if
    // it does not yet exist:
    // (see the constructor for an explanation of the arguments)
    static AnalysisManager* Instance(G4String fileNameBase = "simoutput",
                                     G4int bins = 100,
                                     G4double lowerBound = 0.0 * mm,
                                     G4double upperBound = 10.0 * mm);

    // The analysis class instance can be deleted by calling the Destroy
    // method (NOTE: The class destructor is protected, and can thus not be
    // called directly):
    static void Destroy();

    // Method for setting the bins of the histogram that is used to store the
    // energy deposit (NOTE: The binning can be changed only before the
    // ScoreEnergyDeposit method was called the first time):
    void SetBinning(G4int bins, // number of bins
                  G4double lowerBound, // lower histogram boundary
                  G4double upperBound); // upper histogram boundary

    // Member function used to score the energy deposit:
    inline void ScoreEnergyDeposit(G4double zPosition, G4double energyDeposit);
};

--:-- AnalysisManager.hh (C++ Abbrev)--L26--C28--10%-----
```

# Using emacs

- You can either use the toolbar on the top to invoke actions or you use **keyboard shortcuts**
- Some shortcuts (hold Ctrl pressed and type letters):
  - Open a file: **Ctrl - x - f**
  - Save a file: **Ctrl - x - s**
  - Save file (and choose filename): **Ctrl - x - w**
  - Quit: **Ctrl - x - c**
  - Delete line content (after cursor position): **Ctrl - k**
  - Delete marked text: **Ctrl - w**
  - Paste text: **Ctrl - y**
  - Incremental search: **Ctrl - s**