User Programming in the FLUKA environment

From the FLUKA Advanced Course

Why user routines

- Fluka offers a rich choice of **built-in options** for scoring most quantities and for applying variance reduction techniques, without requiring the users to write a single line of code
- However there are special cases where "ad-hoc" routines are unavoidable, because the needed information cannot be obtained through standard options
- FOOT is one of these cases

What is available for the users

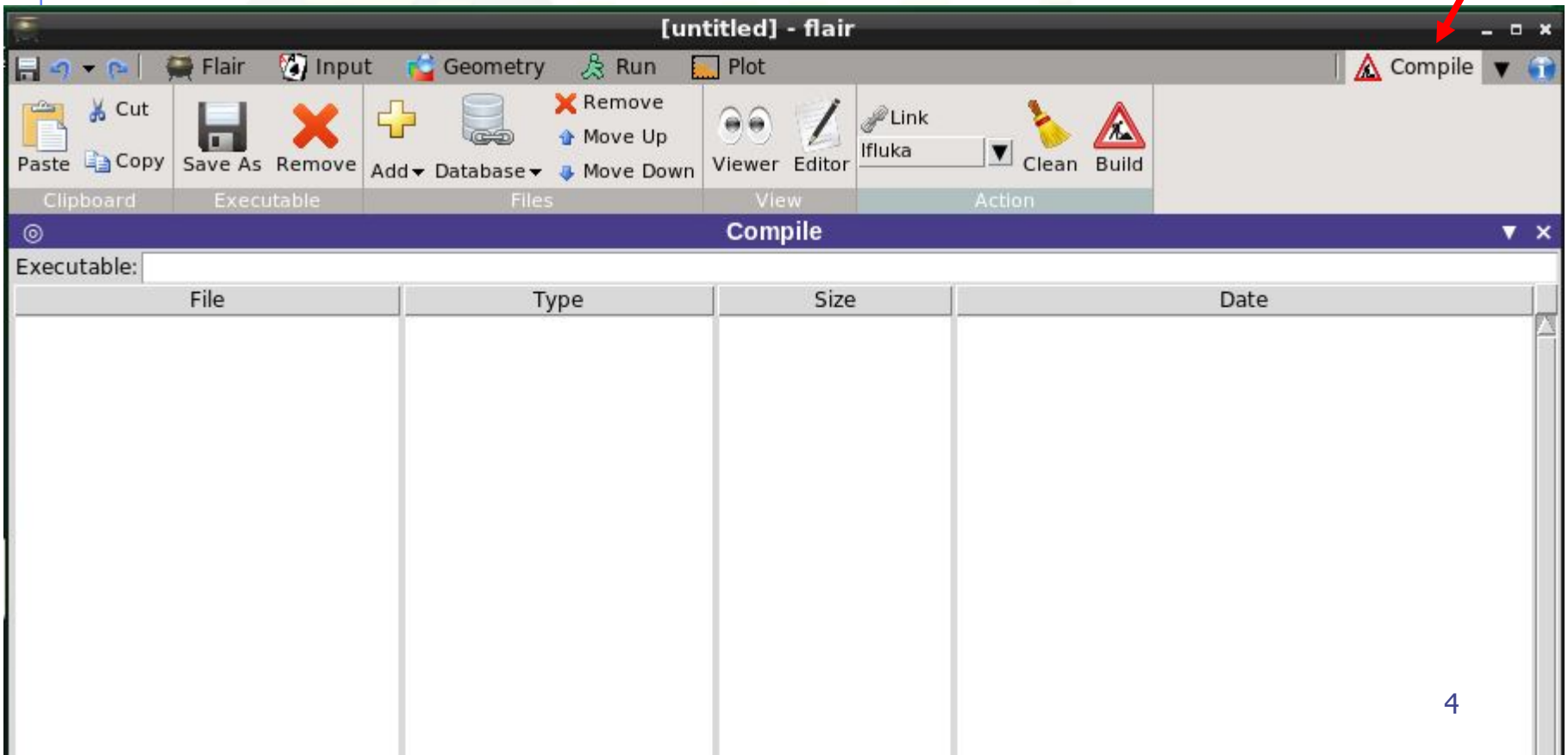
- A number of **user routine** templates are available in the $FLUPRO/usermvax directory and can be modified/activated by the user in order to fulfill non-standard tasks
- The **INCLUDE** files containing the COMMON blocks are in the $FLUPRO/flukapro directory
- An extended **mathematical library** can in principle be exploited by properly calling its members from inside an user routine
- The **compiling and linking scripts** are in the directory $FLUPRO/flutil
- Most user routines need to be **activated** by input directives

Flair can be used to edit, compile and link user routines in order to build a user-specific FLUKA executable

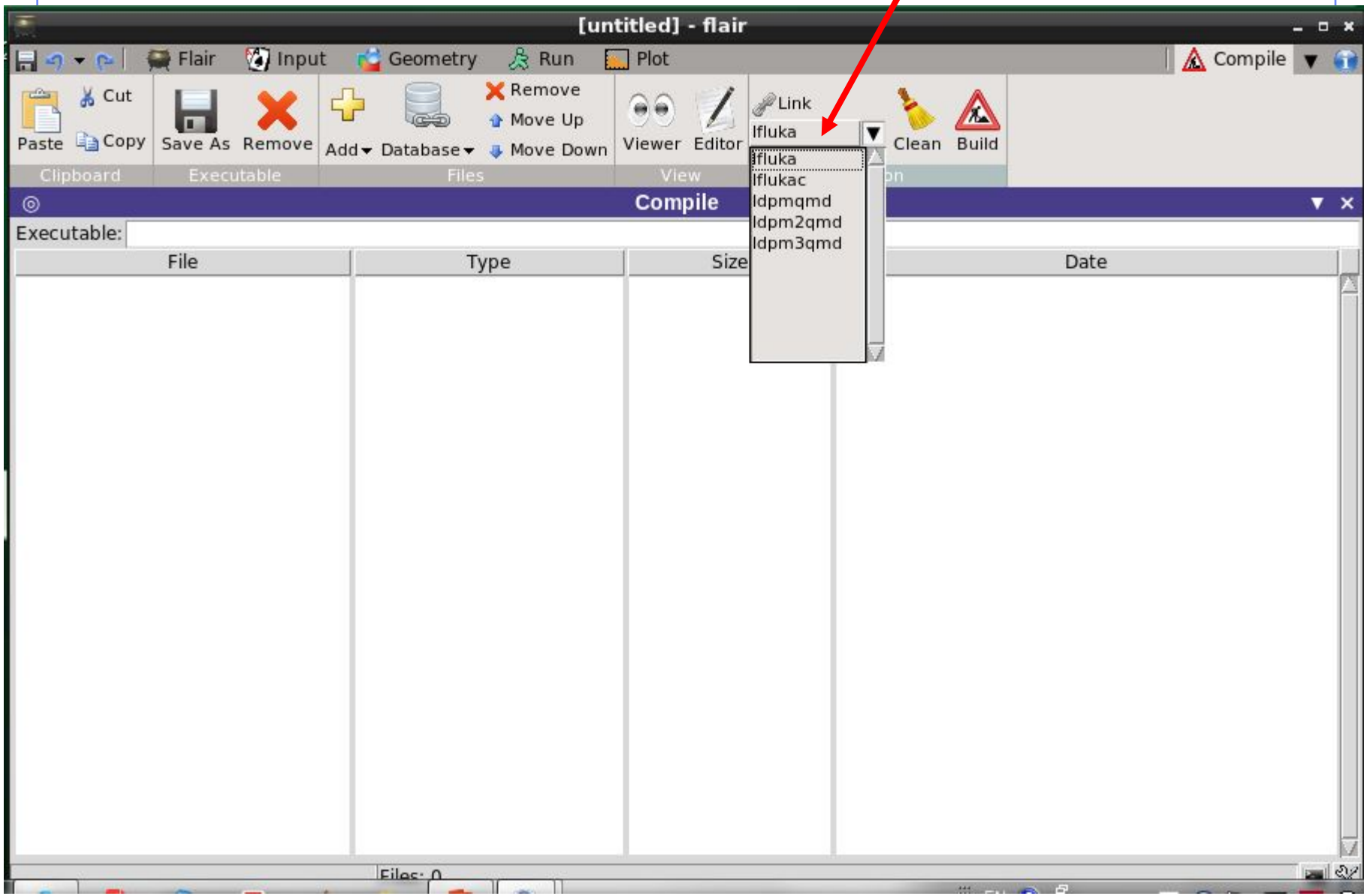
Flair interface (I)

Flair has a button in the Compile frame which scans the input file for possible cards that require an user routine

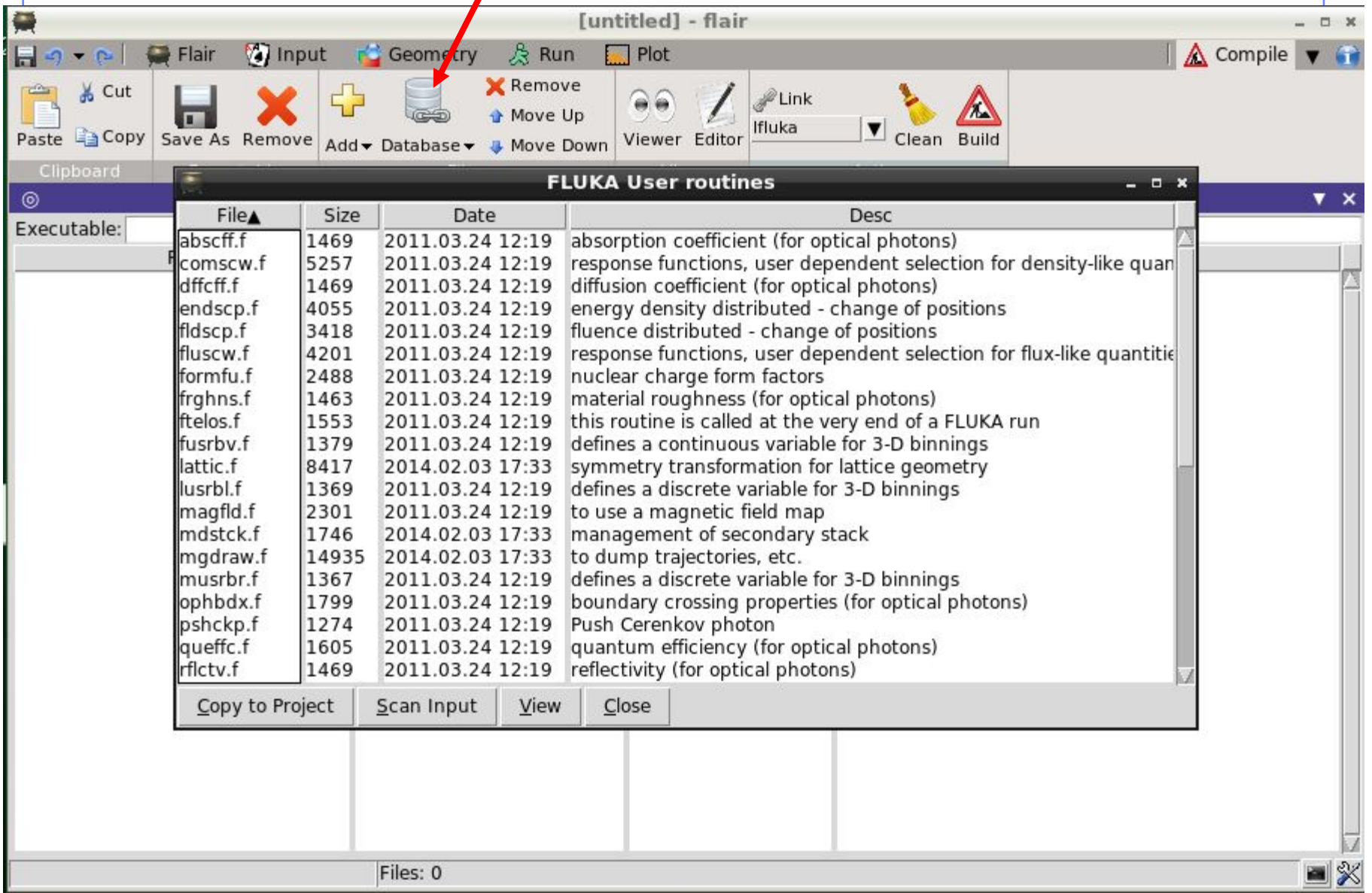
It allows to copy the template routine from $FLUPRO/usermvax to the project directory



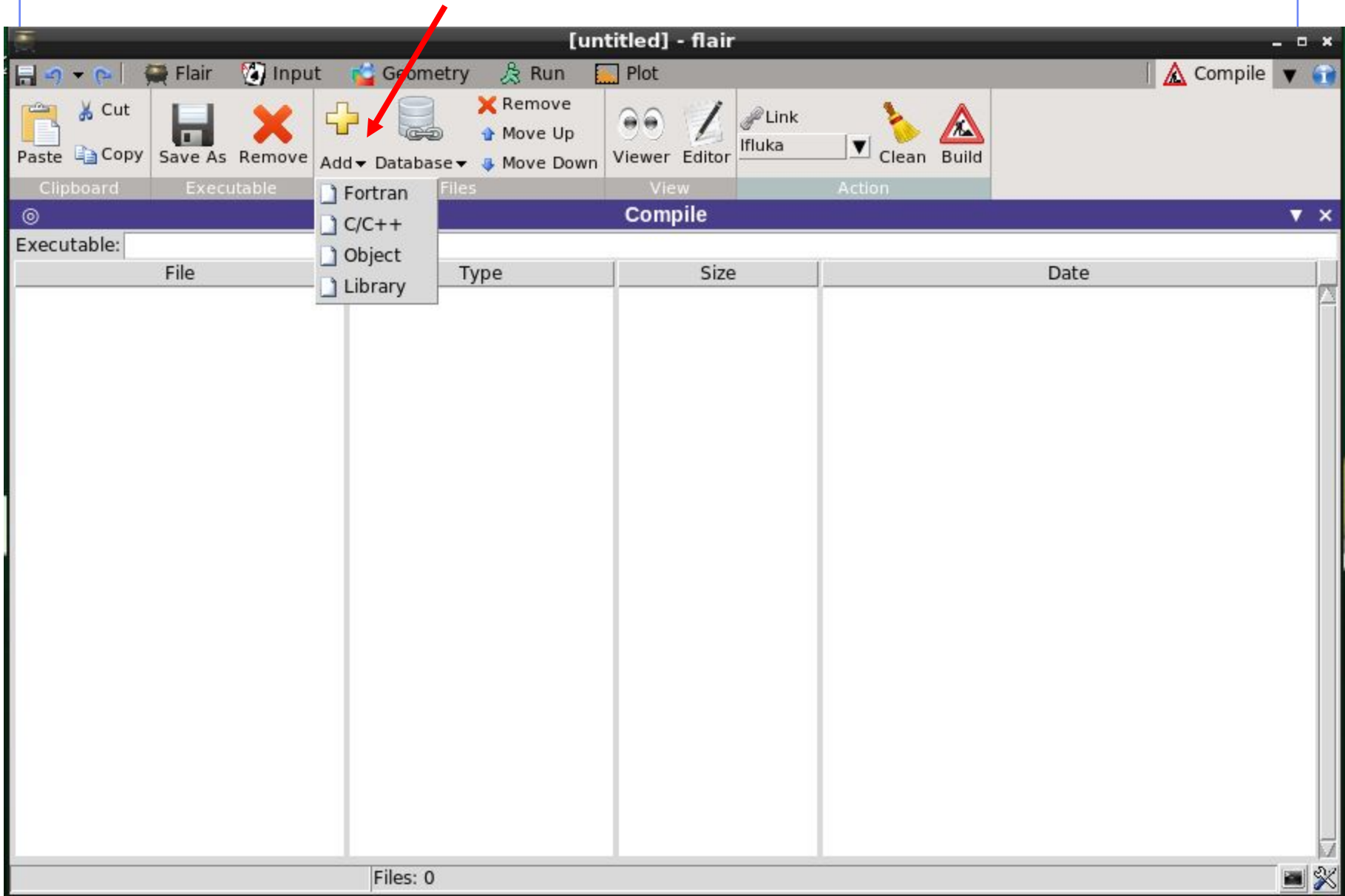
Choosing the builder



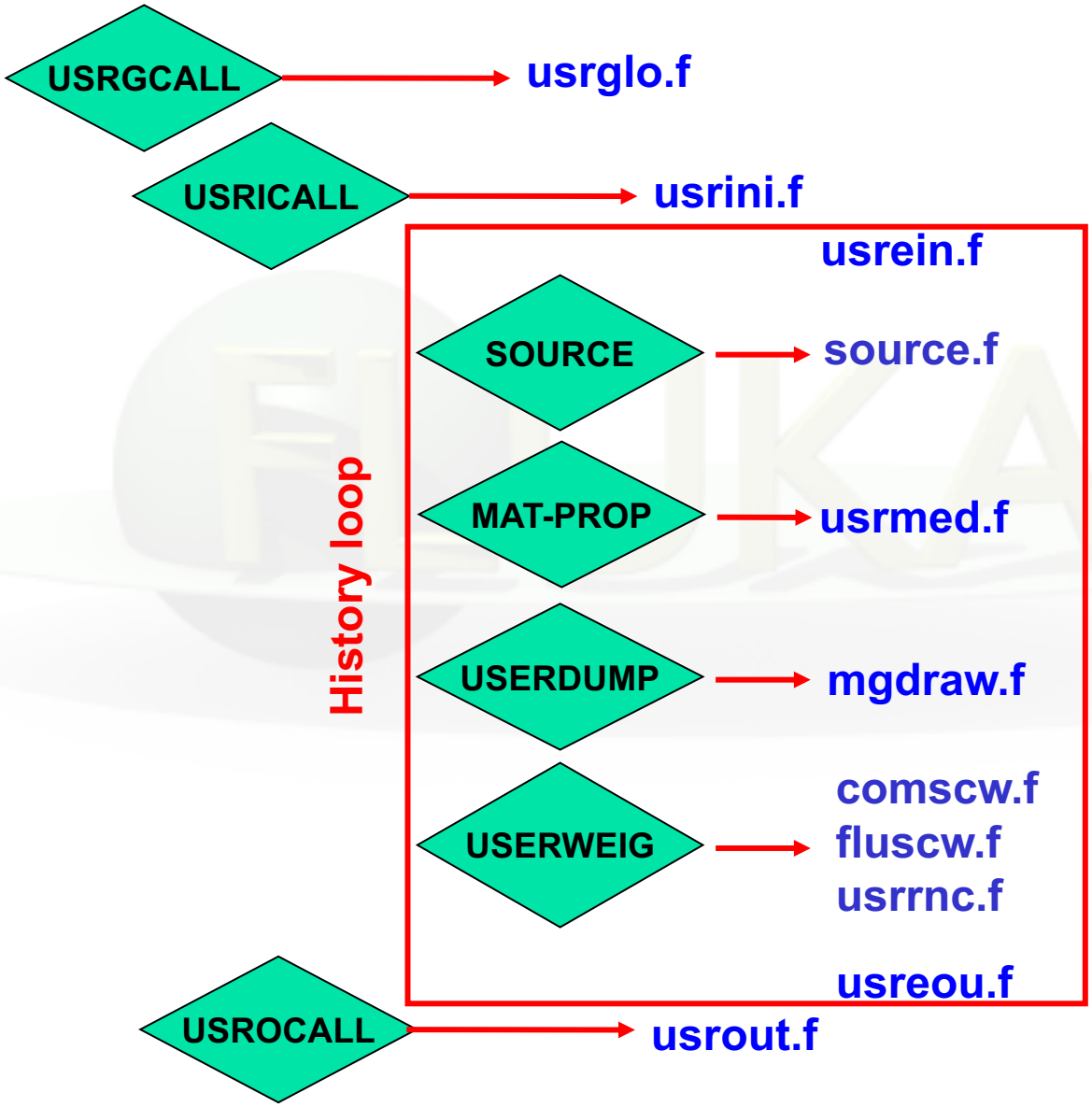
Parsing content of $FLUPRO/usermvax



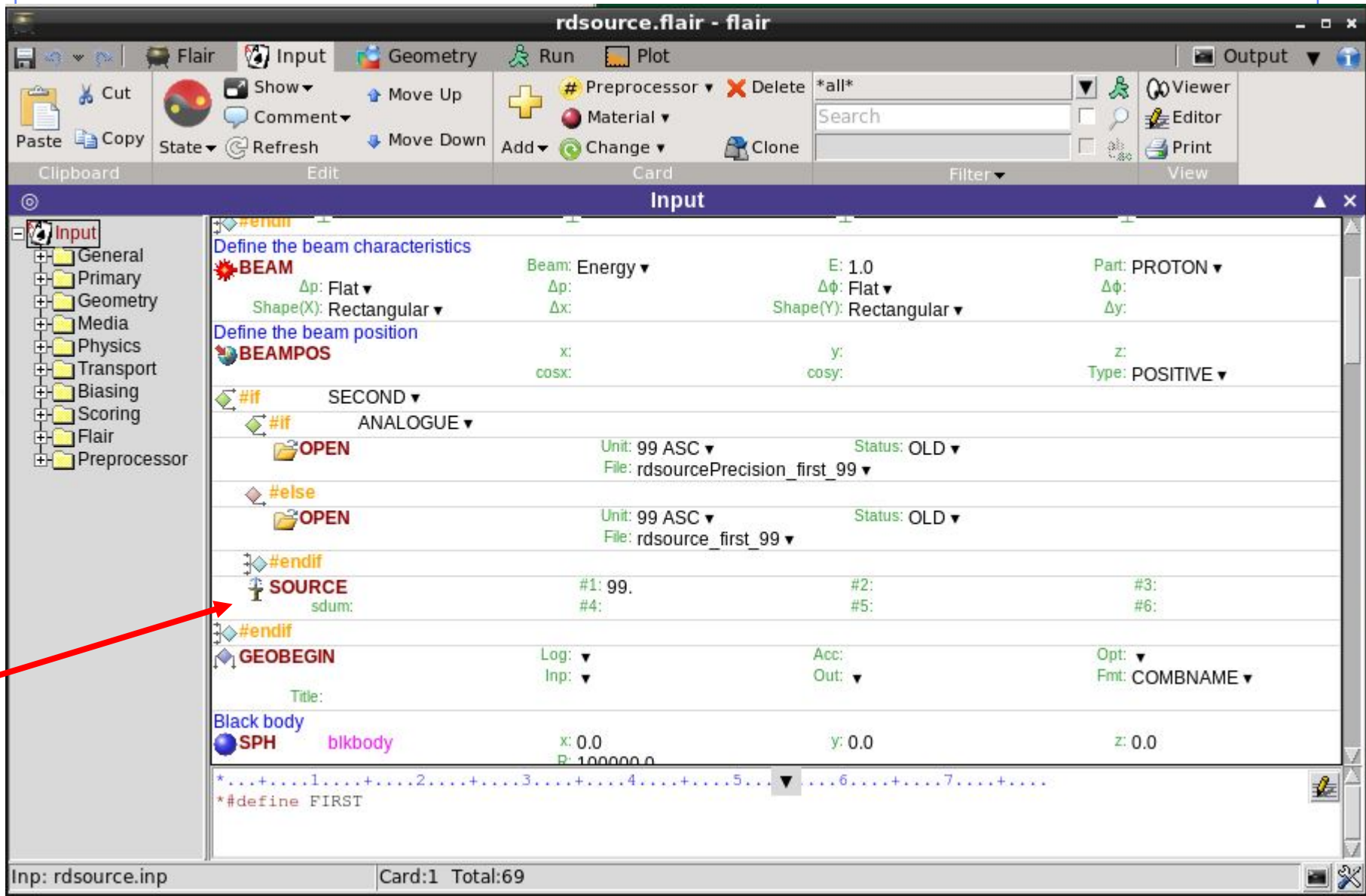
Adding routines from other directories



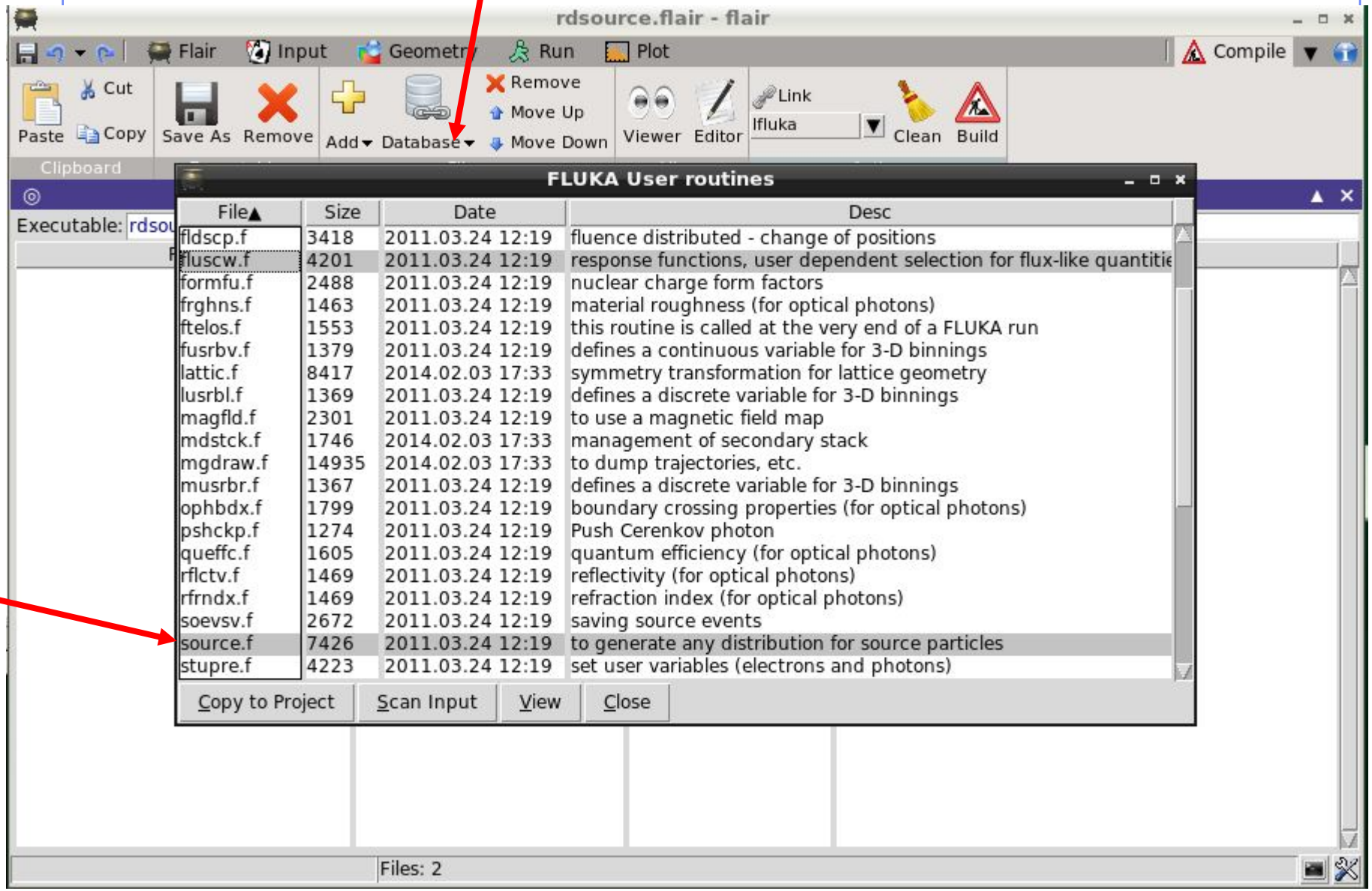
Card – user routine correspondence



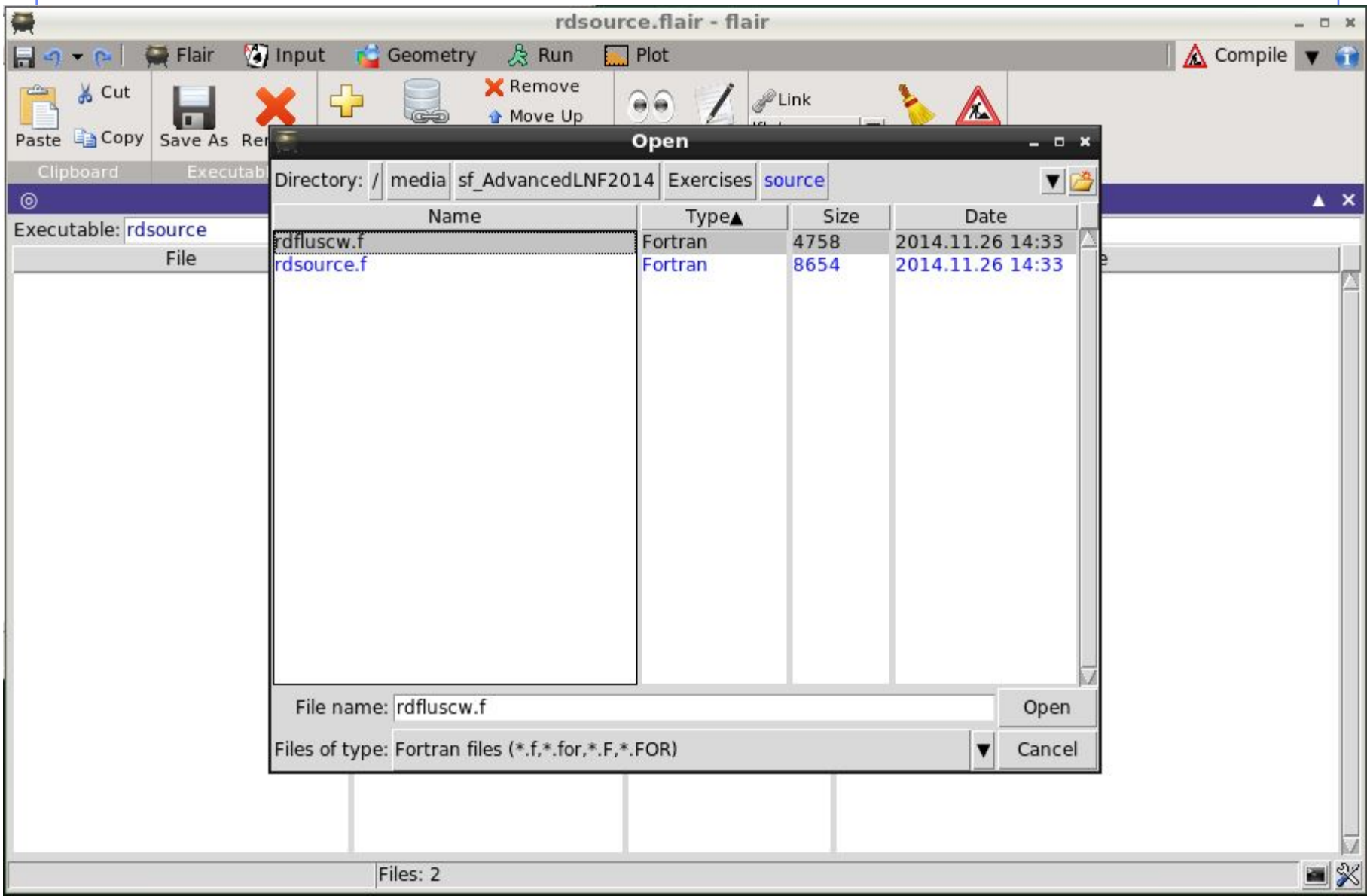
SOURCE card in the input



Automatic recognition by FLAIR



Manual insertion from a user directory



User routine scope (I)

SCORING

- comscw.f
- fluscw.f
- endscp.f
- fldscp.f
- musrbr.f
- lusrbl.f
- fusrbv.f
- usrrnc.f

BIASING

- usbset.f
- usimbs.f
- udcdrl.f

SOURCE GENERATION

- source.f
- (soevsv.f)

OPTICAL PHOTONS

- abscff.f
- dffcff.f
- frghns.f
- ophbdx.f
- queffc.f
- rflctv.f
- rfrndx.f

LATTICE GEOMETRY

- lattic.f

MAGNETIC FIELD

- magfld.f

INITIALIZATION

- usrglo.f
- usrini.f
- usrein.f

OUTPUT

- usreou.f
- usrout.f

User routine scope (II)

accessing
particle stack

- mdstck.f
- stupre.f
- stuprf.f

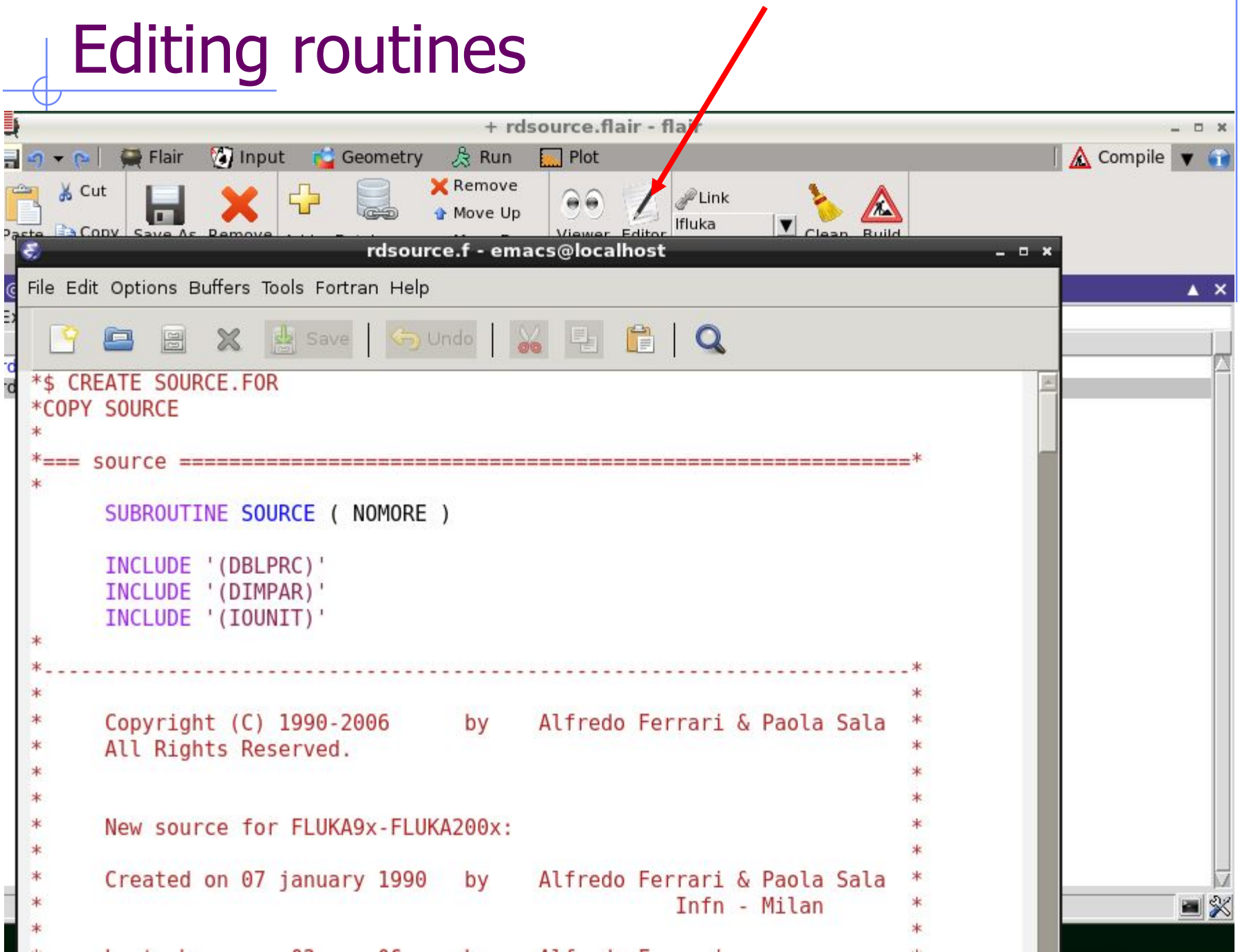
accessing
(almost) everything

- mgdraw.f

multipurpose

- usrmed.f

Editing routines



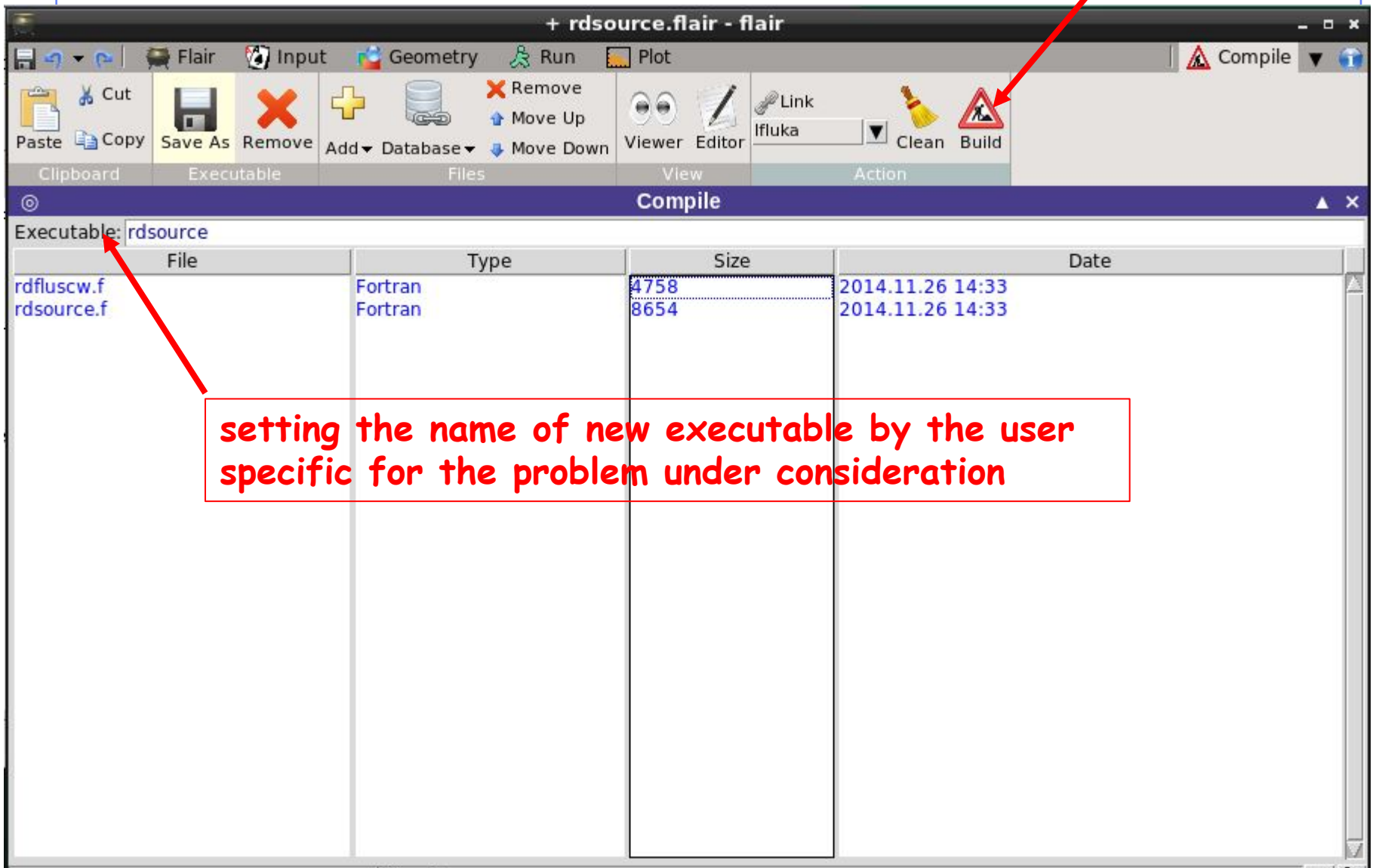
Compiling and linking

- A FLUKA executable with user routines is in general application specific. It must be named and kept separately from the standard FLUKA
- Everything is managed today by FLAIR, however it is important to know the following details (managed automatically inside FLAIR):
- $FLUPRO/flutil/fff is the compiling script with the proper path to the INCLUDE subdirectory and the required compiler (g77 or gfortran) options

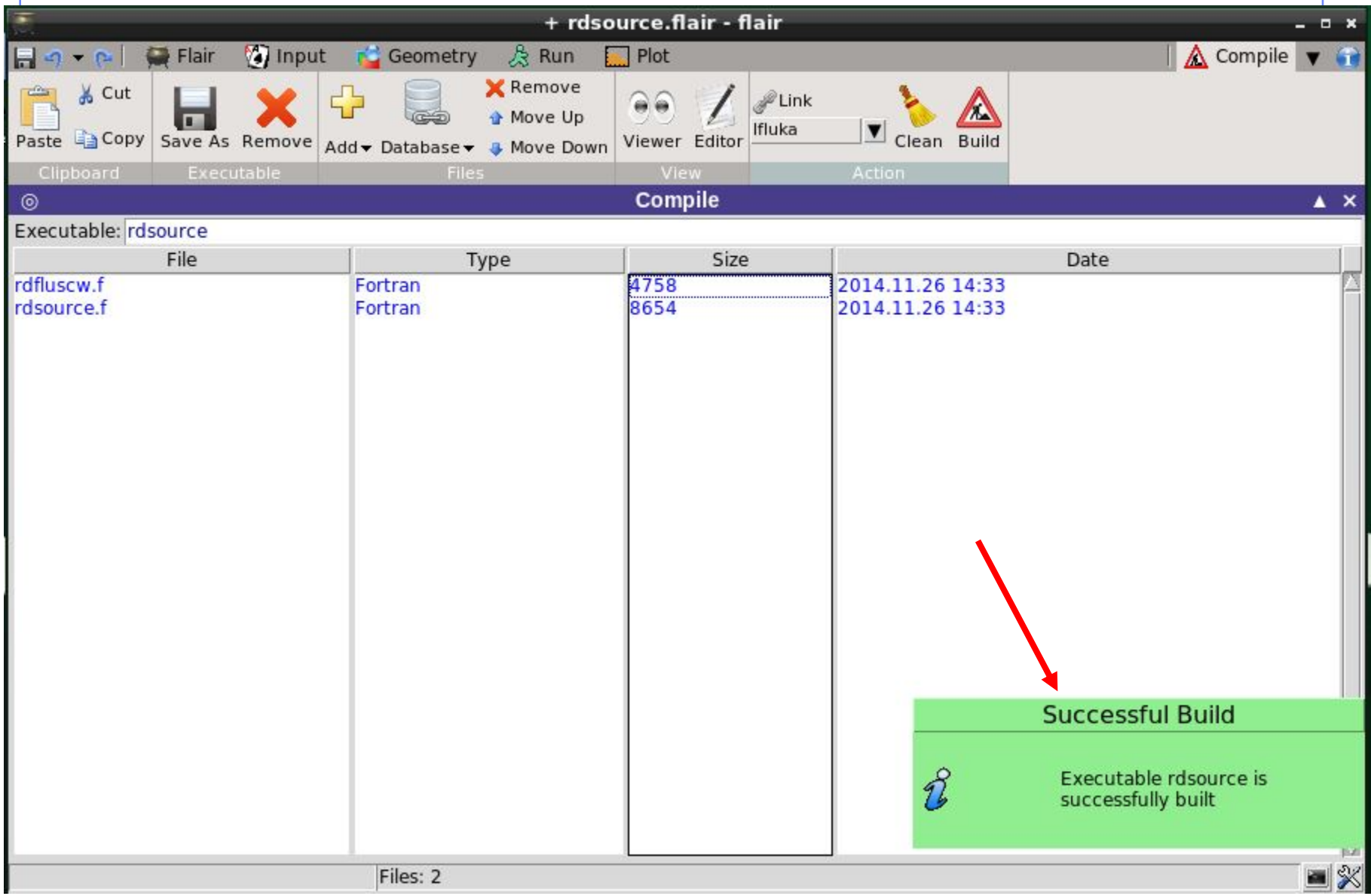
  Example: $FLUPRO/flutil/fff usrini.f generates usrini.o

  then $FLUPRO/flutil/ldpmqmd –m fluka –o flukamy usrini.o will perform the proper linking generating the executable here called flukamy
- Tip: $FLUPRO/flutil/ldpmqmd –m fluka –o flukamy usrini.f will automatically call $FLUPRO/flutil/fff

Compiling and linking (Build) by FLAIR



Successful building



FLUKA programming rules

- Language is Fortran 77 (C routines can be linked)
- Double Precision everywhere, except for integer variables beginning with a letter in the range [i-n]
- Common blocks are in $FLUPRO/flukapro files and are loaded by the INCLUDE statement
- Each routine must start with the following includes/common blocks:

```
INCLUDE '(DBLPRC)'
INCLUDE '(DIMPAR)'
INCLUDE '(IOUNIT)'
```

Note the parentheses which are an integral part of the Fluka INCLUDE file names

- Users may add other FLUKA commons as well as their own commons which may reside in different places

Numerical precision

- Floating point representation

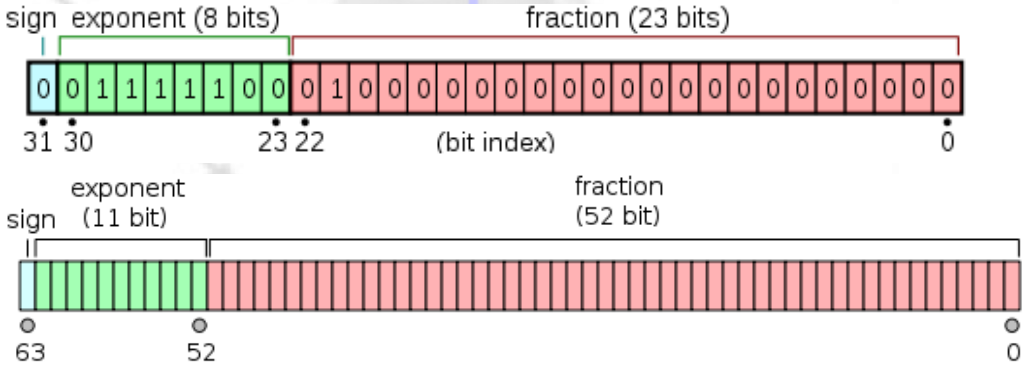
$$\pm d_0 d_1 d_2 \ldots d_{p-1} \times \beta^e$$

where: β=base, $0.dddd$=significant

- Represents the number

$$\pm (d_0 + d_1 \beta^{-1} + \ldots + d_{p-1} \beta^{-(p-1)}) \beta^e, \qquad (0 \le d_i < \beta)$$

- Bits required: $\log_2(e_{max} - e_{min} + 1) + \log_2(\beta^p) + 1$
- Real numbers might not be exactly represented as a floating-point number. Example:
  with β=2 the number 0.1 has an infinite representation and with p=24 will be represented as: 0.100000001490116119384765625
- IEEE representation:
  - Single precision (32bit):
  - Double precision (64bit):



Floating point: Accuracy

- Cancellation: subtraction of nearly equal operands may cause extreme loss of accuracy.
- Conversions to integer are not intuitive:
  converting (63.0/9.0) to integer yields 7,
  but converting (0.63/0.09) may yield 6.
  This is because conversions generally truncate rather than round.
- Limited exponent range: results might overflow yielding infinity, or underflow yielding a denormal value or zero. If a denormal number results, precision will be lost.
- Testing for safe division is problematic: Checking that the divisor is not zero does not guarantee that a division will not overflow and yield infinity.
- Equality test is problematic: Two computational sequences that are mathematically equal may well produce different floating-point values. Programmers often perform comparisons within some tolerance

Minimizing Accuracy Problems

- Use double precision whenever possible.
- Small errors in floating-point arithmetic can grow when mathematical algorithms perform operations an enormous number of times. e.g. matrix inversion, eigenvalues…
- Expectations from mathematics may not be realized in the field of floating-point computation. e.g. $\sin^2\theta+\cos^2\theta = 1$.
- Always replace the $x^2-y^2 = (x+y)(x-y)$
- Equality test should be avoided: replace with "fuzzy" comparisons (if (abs(x-y) < epsilon) ...)
- Adding a large number of numbers can lead to loss of significance, use Kahan algorithm instead
- For the quadratic formula use either

$$\frac{-b \pm \sqrt{b^2-4ac}}{2a} \quad \text{or} \quad \frac{2c}{-b \pm \sqrt{b^2-4ac}}$$

when $b^2>>4ac$, then $\sqrt{(b^2-4ac)} \approx |b|$ therefore will introduce cancelation

Some COMMON blocks in short

BEAMCM:	beam particle properties (from BEAM and BEAMPOS)
SOURCM:	user variables and information for a user-written source
SOUEVT:	recording of the source event
CASLIM:	number of primary particles followed
FLKSTK:	main particle stack of FLUKA
EMFSTK:	particle stack for electrons and photons
GENSTK:	properties of secondaries created in a hadronic event
FHEAVY:	special stack for nuclear fragments
FLKMAT:	material properties
LTCLCM:	LaTtice CeLl CoMmon for lattice cell identification
TRACKR:	properties of the particle currently transported
PAPROP:	intrinsic particle properties (mass, charge, half live…)
SCOHLP:	variables concerning the current estimator type

(DBLPRC) (I)

DouBLe PReCision common
Included in all routines of Fluka, contains the declaration

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

and sets many mathematical and physical constants.
Users are strongly encouraged to adhere to "Fluka style" by

- using systematically double precision (except for very good reasons such as calling external single precision scoring packages)
- and to use constants defined in this file for maximum accuracy.

(DBLPRC) (II)

```
*========== M A T H E M A T I C A L C O N S T A N T S ==========*

* -------- Numerical constants (double precision): --------*
* Zerzer = 0 *
PARAMETER ( ZERZER = 0.D+00 )
* Oneone = 1 *
PARAMETER ( ONEONE = 1.D+00 )
* Twotwo = 2 *
PARAMETER ( TWOTWO = 2.D+00 )
* Pipipi = Circumference / diameter *
PARAMETER ( PIPIPI = 3.14159265358979323846264338327 9D+00 )
* Twopip = 2 x Pipipi *
PARAMETER ( TWOPIP = 6.28318530717958647692528676655 9D+00 )
* Eneper = "e", base of natural logarithm *
PARAMETER ( ENEPER = 2.71828182845904523536028747135 3D+00 )
* Sqrtwo = square root of 2 *
PARAMETER ( SQRTWO = 1.41421356237309504880168872421 0D+00 )
```

(DBLPRC) (III)

```
*========== P H Y S I C A L C O N S T A N T S ==========*
* -------- Primary constants: -------- *
* Clight = speed of light in cm s-1 *
PARAMETER ( CLIGHT = 2.99792458 D+10 )
* Boltzm = k Boltzmann constant (J K-1) *
PARAMETER ( BOLTZM = 1.380658 D-23 )
* Amelgr = electron mass (g) *
PARAMETER ( AMELGR = 9.1093897 D-28 )
* Plckbr = reduced Planck constant (erg s) *
PARAMETER ( PLCKBR = 1.05457266 D-27 )

* -------- Derived constants: -------- *
*Alamb0 = Compton wavelength = 2 pi r0 / fsc , being r0 the classical electron radius *
* and fsc the fine structure constant *
PARAMETER ( ALAMB0 = TWOTWO * PIPIPI * RCLSEL / ALPFSC )

* -------- Astronomical constants: -------- *
* Rearth = Earth equatorial radius (cm) *
PARAMETER ( REARTH = 6.378140 D+08 )

* -------- Conversion constants: -------- *
* GeVMeV = from GeV to MeV *
PARAMETER ( GEVMEV = 1.0 D+03 )
```

(IOUNIT)

Logical input and output unit numbers

The logical units up to 19 (included) are reserved for FLUKA

```
* lunin = standard input unit *
PARAMETER ( LUNIN = 5 )
* lunout = standard output unit *
PARAMETER ( LUNOUT = 11 )
* lunerr = standard error unit *
PARAMETER ( LUNERR = 15 )
…
```

Use the pre-defined output units when you need messages from your user routines:

```
WRITE ( LUNOUT, *)   ' My initialization is active'
WRITE (LUNERR, *)    '  MySource : warning, energy is 0'
```

(CASLIM)

Keeps preset number of histories and current number of histories

```
*   /caslim/ is needed to decide when to stop the run
*  Trnlim = if cpu-time-left<tlim the run will be ended
*  Tpmean =  average time needed to follow one beam particle
*  Tprmax = i maximum time needed to follow  one beam particle
*  Trntot = the cumulative time needed to follow the beam particles
*  Ncases = maximum number of beam particles to be followed
*                modulo 1,000,000,000)
*   Mcases = maximum number of beam particles to be followed
*                in excess of 1,000,000,000, divided by 1,000,000,000
•  Ncase  = current number of beam particles followed (modulo
•                     1,000,000,000)
*   Mcase  = current number of beam particles followed in excess
*                of 1,000,000,000, divided by 1,000,000,000
```

Useful to be included whenever the current event number is needed

(FLKSTK)

At each interaction/decay… etc
new particles are feeding the stack

```
*    /Flkstk/ stack for the primaries                                    *
*       Wtflk  = particle statistical weight                             *
*       Pmoflk = particle (laboratory) momentum (GeV/c)                  *
*       Tkeflk = particle (laboratory) kinetic energy (GeV)              *
*       Xflk   = particle position  x-coordinate                         *
*       Yflk   = particle position  y-coordinate                         *
*       Zflk   = particle position  z-coordinate                         *
*       Txflk  = particle direction x-coordinate                         *
*       Tyflk  = particle direction y-coordinate                         *
*       Tzflk  = particle direction z-coordinate                         *
*       Txpol  = x direction cosine of the particle polarization         *
*       Typol  = y direction cosine of the particle polarization         *
*       Tzpol  = z direction cosine of the particle polarization         *
*       Dfnear = distance to the nearest boundary                        *
*       Agestk = age of the particle (seconds)                           *
*       Cmpath = cumulative path travelled by the particle since it was produced (cm) *
*       Iloflk = particle identity (Paprop numbering)                    *
*       Igroup = energy group for low energy neutrons                    *
*       Loflk  = particle generation                                     *
*       Louse  = user flag                                               *
*       Nrgflk = particle region number                                  *
*       Nlattc = particle lattice cell number                            *
```

(TRACKR)

Transport of particles:
particles are taken from the Stack and
info for the particle during tracking are kept here

TRACK Recording

Ntrack = number of track segments
Mtrack = number of energy deposition events along the track
0 < i < Ntrack
Xtrack = end x-point of the ith track segment
Ytrack = end y-point of the ith track segment
Ztrack = end z-point of the ith track segment
1 < i < Ntrack
Ttrack = length of the ith track segment
1 < j < Mtrack
Dtrack = energy deposition of the jth deposition event
Dptrck = momentum loss of the jth deposition event

Ntrack > 0, Mtrack > 0 : energy loss distributed along the track
Ntrack > 0, Mtrack = 0 : no energy loss along the track
Ntrack = 0, Mtrack = 0 : local energy deposition (the value and the point are not recorded in Trackr)

```
 COMMON / TRACKR /  XTRACK ( 0:MXTRCK ), YTRACK ( 0:MXTRCK ),
&                   ZTRACK ( 0:MXTRCK ), TTRACK  ( MXTRCK ),
&                   DTRACK  ( MXTRCK ), DPTRCK ( 3,MXTRCK ),
```

(TRACKR) : 2nd part

Jtrack = identity number of the particle: for recoils or kerma deposition it can be outside the allowed particle id range, assuming values like:
208: "heavy" recoil
211: EM below threshold
308: low energy neutron kerma
in those cases the id of the particle originating the interaction is saved inside J0trck (which otherwise is zero)

J0trck = see above

Etrack = total energy of the particle

Ptrack = momentum of the particle (not always defined, if < 0 must be obtained from Etrack)

Cx,y,ztrck = direction cosines of the current particle

Cx,y,ztrpl = polarization cosines of the current particle

Wtrack = weight of the particle

Wscrng = scoring weight: it can differ from Wtrack if some biasing techniques are used (for example inelastic interaction length biasing)

Ctrack = total curved path

Cmtrck = cumulative curved path since particle birth

(TRACKR) : 3rd part

Zfftrk = <Z_eff> of the particle
Zfrttk = actual Z_eff of the particle
Atrack = age of the particle
Wninou = neutron algebraic balance of interactions (both for "high" energy particles and "low" energy neutrons)
Wcinou = charge algebraic balance of interactions (for all interactions)
Spausr = user defined spare variables for the current particle
Ktrack = if > 0 neutron group of the particle (neutron)

Lt1trk = initial lattice cell of the current track (or lattice cell for a point energy deposition)
Lt2trk = final lattice cell of the current track
Iprodc = flag for prompt(=1)/radioactive products(=2)
Ltrack = flag recording the generation number
Llouse = user defined flag for the current particle
Ispusr = user defined spare flags for the current particle

```
...
     &          SPAUSR(MKBMX1), STTRCK, SATRCK, TKNIEL, TKEDPA,
     &          WCINOU,
...
     &          IPRODC, ISPUSR(MKBMX2), LFSSSC, LPKILL
```

(FHEAVY)

```
*       npheav     = number of heavy  secondaries                    *
*       kheavy(ip) = type of the secondary ip                        *
*                ( 3 = deuteron, 4 = 3-H, 5 = 3-He, 6 = 4-He,       *
*                  7-12 = "Heavy" fragment specified by Ibheav and Icheav ) *
*       cxheav(ip) = direction cosine of the secondary ip with respect to x-axis *
*       cyheav(ip) = direction cosine of the secondary ip with respect to y-axis *
*       czheav(ip) = direction cosine of the secondary ip with respect to z-axis *
*       tkheav(ip) = kinetic energy of secondary ip                  *
*       pheavy(ip) = momentum of the secondary ip                    *
*       wheavy(ip) = weight of the secondary ip                      *
*       agheav(ip) = "age" of the secondary ip with respect to the interaction time *

*       amheav(kp) = atomic masses of the twelve types of evaporated  *
*                  or fragmented or fissioned particles              *
*       amnhea(kp) = nuclear masses of the twelve types of evaporated *
*                  or fragmented or fissioned particles              *
*       anheav(kp) = name of the kp-type heavy particle              *
*       icheav(kp) = charge of the kp-type heavy particle            *
*       ibheav(kp) = mass number of the kp-type heavy particle       *
```

Note that kp = kheavy(ip) !!!

(PAPROP)

intrinsic PArticle PROPerties

```
*       am    (i) = i_th particle mass (GeV)                          *
*       ichrge(i) = electric charge of the i_th particle               *
*       ibarch(i) = baryonic charge of the i_th particle               *
*       ijdisc(i) = flag for discarding the i_th particle type         *
*       tmnlf (i) = mean (not half!) life of the i_th particle (s)     *
*       biasdc(i) = decay biasing factor for the i_th particle         *
*       biasin(i) = inelastic interaction biasing factor for the i_th particle *
*       lhadro(i) = True if the i_th particle type is a hadron         *
*       jspinp(i) = i_th particle spin (in units of 1/2)               *
*       iparty(i) = i_th particle parity (when meaningful)             *
```

(FLKMAT)

FLuKa MATerials

```
*      Amss(i) = Atomic weight (g/mole) of the i_th material          *
*       Rho(i) = Density of the i_th material                         *
*      Ztar(i) = Atomic number of the i_th material                   *
*    Ainlng(i) = Inelastic scattering length of the i_th material    *
*                for beam particles at the average beam energy in cm *
*    Aellng(i) = Elastic scattering length of the i_th material for  *
*                beam particles at average beam energy in cm         *
*     X0rad(i) = Radiation length of the i_th material in cm         *
*    Dmgene(i) = Damage energy of the i_th material (GeV)            *
*    Ainnth(i) = Inelastic scattering length of the i_th material    *
*                for neutrons at threshold energy in cm              *
*    Medium(k) = Material number of the k_th region                  *
*    Mssnum(i) = Mass number of the target nucleus for the i_th material *
*                if =< 0 it means that it is in the natural isotopic composition *
*    Libsnm(i) = flag whether inelastic interaction biasing must be done for this medium *
*    Matnam(i) = Alphabetical name of the i_th material number       *
*    Aocmbm(i) = Atomic density of the i_th material in barn^-1 cm^-1 *
*                (Atoms Over Cm times Barn for Materials)            *
*    Eocmbm(i) = Electron density of the i_th material in barn^-1cm^-1*
*                (Atoms Over Cm times Barn for Materials)            *
```

(EVTFLG)

EVenT FLaGs:

Flags indicating the event interaction type:

LELEVT = Elastic interaction
LINEVT = Inelastic interaction
LDECAY = Particle decay
LDLTRY = Delta ray production (Moller and Bhabha included)
LPAIRP = Pair production
LBRMSP = Bremsstrahlung
LANNRS = Annihilation at rest
LANNFL = Annihilation in flight
LPHOEL = Photoelectric effect
LCMPTN = Compton effect
LCOHSC = Rayleigh scattering
LLENSC = Low energy neutron scattering
LOPPSC = Optical photon scattering
LELDIS = Electromagnetic dissociation
LRDCAY = Radioactive decay

All **LOGICAL** variables!!!

mgdraw.f [1]

general event interface

Argument list (all variables are input only)

ICODE : FLUKA physical compartment originating the call
- = 1: call from subroutine KASKAD (hadrons and muons)
- = 2: call from subroutine EMFSCO (e^-, e^+ and photons)
- = 3: call from subroutine KASNEU (low-energy neutrons)
- = 4: call from subroutine KASHEA (heavy ions)
- = 5: call from subroutine KASOPH (optical photons)

MREG : current region

Subroutine mgdraw is activated by option USERDUMP with WHAT(1) ≥ 100.0, usually writes a "collision tape", i.e., a file where all or selected transport events are recorded. The default version (unmodified by the user) offers several possibilities, selected by WHAT(3)

mgdraw.f [2]

The different ENTRY points of mgdraw

MGDRAW called at each step, for trajectory drawing and recording dE/dx energy deposition events
BXDRAW called at boundary crossings (no record)
EEDRAW called at event end (no record)
ENDRAW for recording point energy deposition events
SODRAW for recording source particles

One can remove their default writing and/or customize them.

Additional flexibility is offered by the user entry USDRAW, interfaced with the most important physical events happening during particle transport.

mgdraw.f [3]

All six entries can be activated at the same time by setting USERDUMP WHAT(3) = 0.0 and WHAT(4) ≥ 1.0.

They constitute a complete interface to the entire FLUKA transport. Therefore, mgdraw can be used not only to write a collision tape, but to do any kind of complex analysis (*e.g.*, event by event output as in HEP applications).

When mgdraw should better not be used

- When biasing is requested (non-analogue run)
- Whenever low-energy neutrons (E<20 MeV) are involved, unless one has a deep knowledge of the peculiarities of their transport and quantities (*i.e.*, kerma, etc)

mgdraw.f: the MGDRAW entry

MTRACK:	number of energy deposition events along the track
JTRACK:	type of particle
ETRACK:	total energy of the particle
WTRACK:	weight of the particle
NTRACK:	values of XTRACK, YTRACK, ZTRACK: end of each track segment
MTRACK:	values of DTRACK: energy deposited at each deposition event
CTRACK:	total length of the curved path

Other variables are available in TRACKR (but not written by MGDRAW unless the latter is modified by the user: particle momentum, direction cosines, cosines of the polarisation vector, age, generation, etc. see a full list in the comment in the INCLUDE file).

mgdraw.f: the BXDRAW entry

called at *boundary crossing*

Argument list (all variables are input only)

ICODE	:	physical compartment originating the call, as in the MGDRAW entry
MREG	:	region from which the particle is exiting
NEWREG	:	region the particle is entering
XSCO, YSCO, ZSCO	:	point where the boundary crossing occurs

mgdraw.f: the EEDRAW entry

called at the *event end*

Argument list (all variables are input only)		
ICODE	:	physical compartment originating the call, as in the MGDRAW entry

mgdraw.f: the ENDRAW entry

called at point-like energy deposition
(for example: stopping particles, photoelectric effect, ...)

Argument list (all variables are input only)

ICODE : type of event originating energy deposition
ICODE = 1x: call from subroutine KASKAD (hadrons and muons);
= 10: elastic interaction recoil
= 11: inelastic interaction recoil
= 12: stopping particle
= 14: particle escaping (energy deposited in blackhole)
ICODE = 2x: call from subroutine EMFSCO (electrons, positrons and photons)
= 20: local energy deposition (i.e. photoelectric)
= 21 or 22: particle below threshold
= 23: particle escaping (energy deposited in blackhole)
ICODE = 3x: call from subroutine KASNEU (low-energy neutrons)
= 30: target recoil
= 31: neutron below threshold
= 32: neutron escaping (energy deposited in blackhole)
ICODE = 4x: call from subroutine KASHEA (heavy ions)
= 40: ion escaping (energy deposited in blackhole)
ICODE = 5x: call from subroutine KASOPH (optical photons)
= 50: optical photon absorption
= 51: optical photon escaping (energy deposited in blackhole)
MREG : current region
RULL : energy amount deposited
XSCO, YSCO, ZSCO : point where energy is deposited

mgdraw.f: the SODRAW entry

Argument list
No arguments

It writes by default, for each source particle:

NCASE: number of primaries followed so far (with a minus sign to identify SODRAW output), from **COMMON CASLIM**

NPFLKA: stack pointer, in **COMMON FLKSTK**

NSTMAX: highest value of the stack pointer encountered so far, in **COMMON FLKSTK**

TKESUM: total kinetic energy of the primaries of a user written source, in **COMMON SOURCM**, if applicable. Otherwise = 0.0

WEIPRI: total weight of the primaries handled so far, in **COMMON SOURCM**

NPFLKA times: (all variables in COMMON FLKSTK)		
	ILOFLK:	type of source particle
	TKEFLK + AM:	total particle energy (kinetic+mass)
	WTFLK:	source particle weight
	XFLK, YFLK, ZFLK:	source particle position
	TXFLK, TYFLK, TZFLK:	source particle direction cosines

mgdraw.f: the USDRAW entry

called *after each particle interaction*
(requested by USERDUMP WHAT(4) ≥ 1.0)

Argument list (all variables are input only)

ICODE : type of event

ICODE = 10x: call from subroutine KASKAD (hadron and muon interactions);
- = 100: elastic interaction secondaries
- = 101: inelastic interaction secondaries
- = 102: particle decay secondaries
- = 103: delta ray generation secondaries
- = 104: pair production secondaries
- = 105: bremsstrahlung secondaries

ICODE = 20x: call from subroutine EMFSCO (electron, positron and photon interactions)
- = 208: bremsstrahlung secondaries
- = 210: Møller secondaries
- = 212: Bhabha secondaries
- = 214: in-flight annihilation secondaries
- = 215: annihilation at rest secondaries
- = 217: pair production secondaries
- = 219: Compton scattering secondaries
- = 221: photoelectric secondaries
- = 225: Rayleigh scattering secondaries

ICODE = 30x: call from subroutine KASNEU (low-energy neutron interactions)
- = 300: neutron interaction secondaries

ICODE = 40x: call from subroutine KASHEA (heavy ion interactions)
- = 400: delta ray generation secondaries

MREG : current region

XSCO, YSCO, ZSCO : interaction point

Mathematical library

FLUKA contains many mathematical routines of general utility, so in general it should not be necessary to call external mathematical libraries (many taken from SLATEC):

flgaus:	Gaussian adaptive integration
erffun:	Error function
expin1:	E1 exponential function
besi0d:	Bessel function I0 (also I1, J0, J1, K0, K1)
dawsni:	Dawson function
gamfun:	Gamma function
radcub:	Real solutions of 3rd order algebraic equation
flgndr:	Legendre polynomials
yinter, d..intp:	interpolation routines
rordin, rordde:	Sorting of vector values

.....

Also: expansion in Laguerre and Chebyshev polynomials, Bezier fit, and many others...

For users who access the FLUKA source: they are in mathmvax directory

At some time it will be possible to have a short-writeup for their use.

A few examples (I)

```
 EXTERNAL FINTEG
 DOUBLE PRECISION FUNCTION FLGAUS ( FINTEG, XA, XB, EPSEPS, IOPT,
&                                                   NXEXP )
* Adaptive Gaussian quadrature routine
```

It gives the integral over the (XA,XB) interval of the product between X**NXEXP and the FINTEG function, to be coded by the user as a separate

```
 DOUBLE PRECISION FUNCTION FINTEG (X)
```

```
SUBROUTINE RADCUB ( AA0, AA1, AA2, AA3, X, X0, NRAD )
* Real solutions of 3rd order algebric equation
```

It computes real solutions of the equation:

A0*X^3++A1*X^2+A2*X+A3=0

The solutions are put in the array X; if there is only one real solution it is put into X(1), while X(2) and X(3) are set to 1.d32. If A0=0 the routine computes standard solutions of a second or first degree equation. If it doesn't exist any real solution the whole array X is set to 1.d32. It is possible to compute solutions with a scale factor X0, to avoid loss of significancy with very large or very small numbers. The flag NRAD records the number of real solutions found.

A few examples (II)

DOUBLE PRECISION FUNCTION GAMFUN (X)

It calculates the double precision complete **Gamma function** for double precision argument X

SUBROUTINE RORDIN (RVECT, ICORR, LEN)

It rearranges a real array in increasing order

SUBROUTINE RORDDE (RVECT, ICORR, LEN)

It rearranges a real array in decreasing order

DOUBLE PRECISION FUNCTION FLGNDR (X, LMAX, PLGNDR)

* Function for **LeGeNDRe polynomials**

It computes P_{lmax} (x) and stores all values P_i (x) for i=0,lmax into the PLGNDR array