Advanced language features

October 2018

- Macros and tricks for your instrument...

DECLARE / INITIALIZE

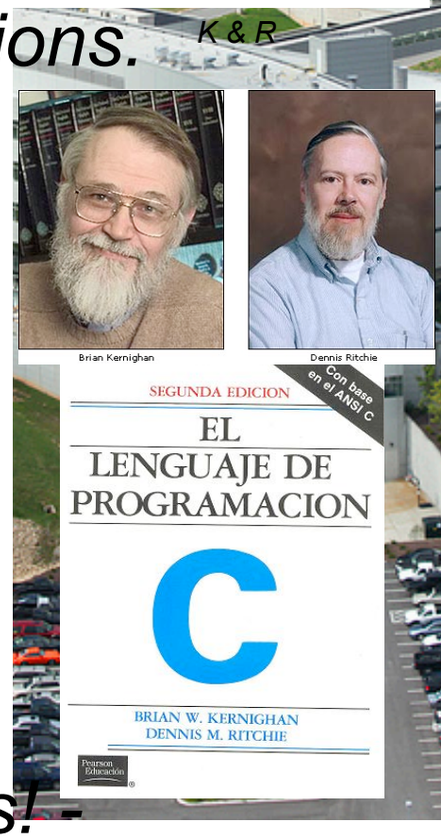
October 2018

- Use the DECLARE section define user variables and functions
  - DECLARE %{
  - double myvar;
  - %}

- Use INITIALIZE for initialization of user variables and calculations.
  - INITIALIZE %{
  - myvar = sqrt(PI*input_var)*rand01();
  - %}

- Both use normal c-syntax.

- BEWARE: (example) What you do in the c-style areas is c-standard, e.g. trigonometric functions from math.h use radians! - McStas placement specifiers work in degrees, etc...

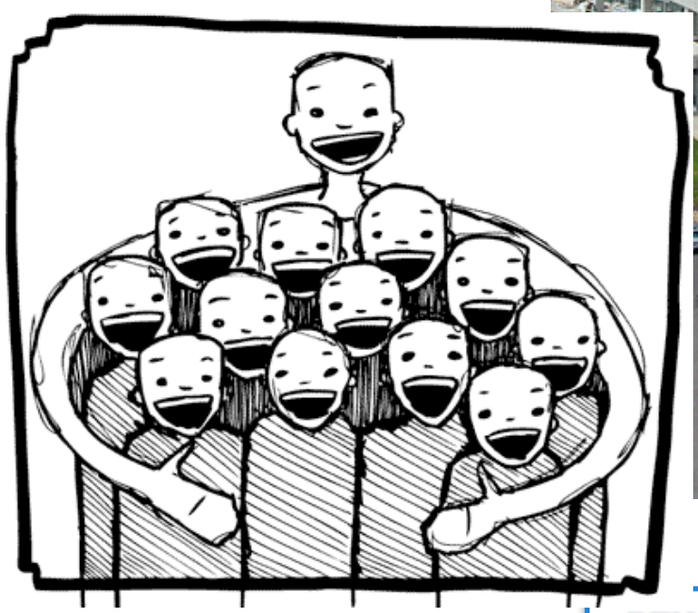


%include

- *Instrumentfiles can include external c-code or other instrumentfiles... See the examples:*

- *ILL_H15_IN6.instr:%include "monitor_nd-lib"*
- *ILL_H16_IN5.instr:%include "ILL_H16.instr"*
- *ILL_H25_IN22.instr:%include "ILL_H25.instr"*
- *ILL_H25_IN22.instr:%include "templateTAS.instr"*

- *Used in the DECLARE section*



Syntax in one, complex view...

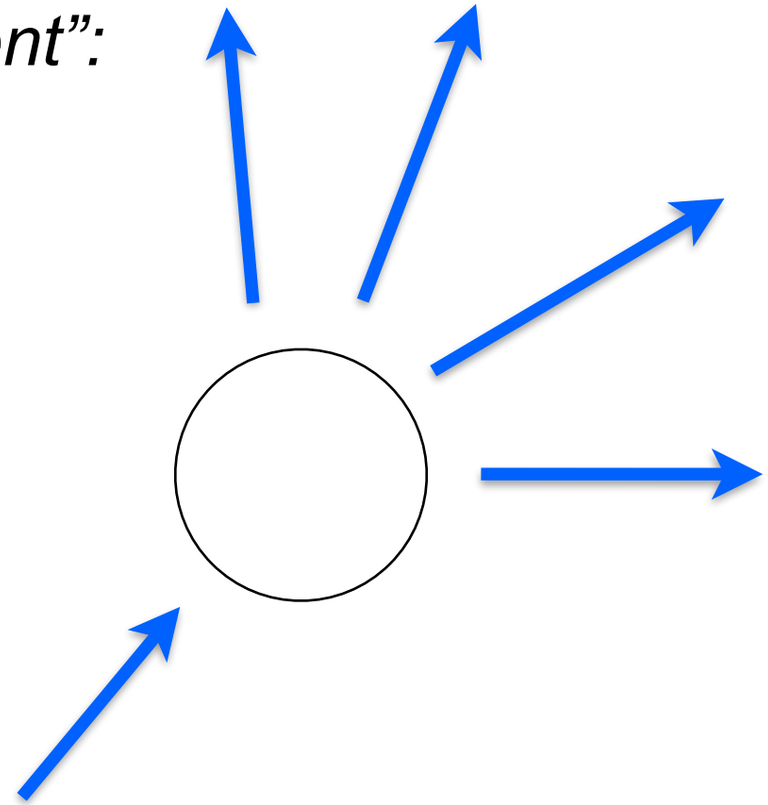
{SPLIT} COMPONENT name = comp(parameters) {WHEN condition}
 AT (...) [RELATIVE [reference|PREVIOUS] | ABSOLUTE]
 {ROTATED {RELATIVE [reference|PREVIOUS] | ABSOLUTE} }
 {GROUP group_name}
 {EXTEND C_code}
 {JUMP [reference|PREVIOUS|MYSELF|NEXT] [ITERATE number_of_times | WHEN condition] }

SPLIT

- *Increase statistics beyond this point in the instrumentfile*

- *SPLIT n MyArm = Arm()*
- *AT somewhere*

- *will “formulate an if-statement”:*



- *for j=1:n*
  - *comp1*
  - *comp2*
  - *comp3*
  - *...*
- *end (of instrument)*

- *ONLY meaningful in case of Monte Carlo choices after SPLIT point...*

WHEN

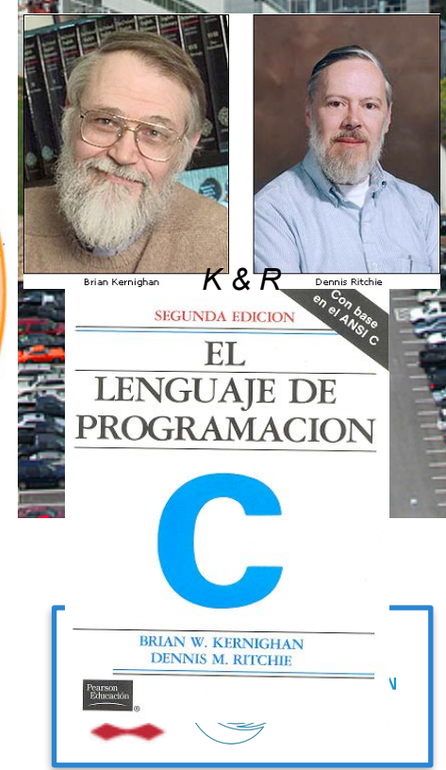
- *Syntax:*

- *COMPONENT Mine = Yours(blah, blah)*
- *WHEN (c-expression) AT (....)*

- *Is very powerful when combined with EXTEND and user variables, or as a method to let input parameters select if certain components are active.*

- *Example: Use EXTEND to flag if neutron was scattered on one monochromator blade or another. Then later use WHEN to only show contribution from blade N at sample position?*

- *COMPONENT Mon = PSD_monitor(...)*
- *WHEN (myvar==1) AT (0,0,0) RELATIVE Sample*



GROUP - components working in parallel

October 2018

COMPONENT Mono1 = Monochromator_curved(...)
AT (0,0, -LMM) RELATIVE Cradle ROTATED (0,A1/2,0) RELATIVE Cradle
GROUP IN6Monoks

COMPONENT Mono2 = Monochromator_curved(...)
AT (0,0, 0) RELATIVE Cradle ROTATED (0,A2/2,0) RELATIVE Cradle
GROUP IN6Monoks

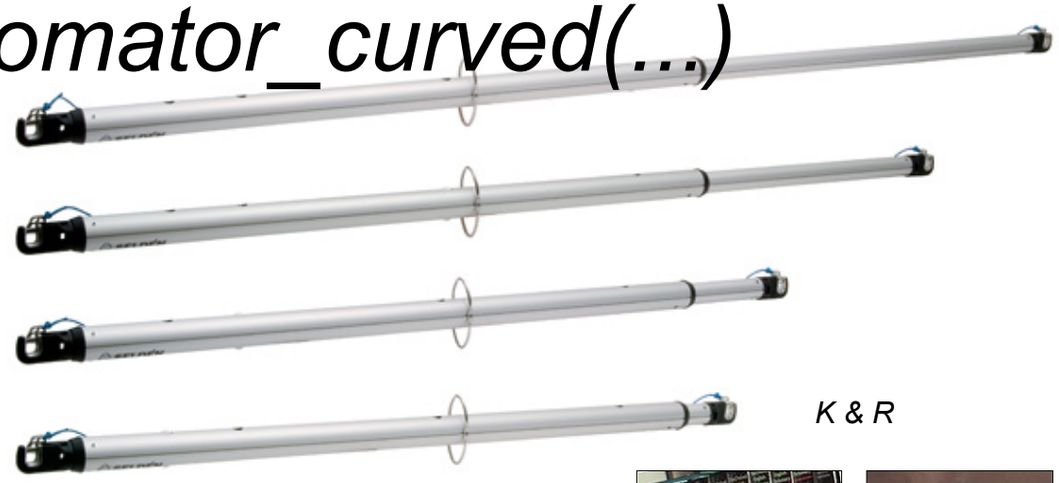
- One comp after the other is "tried" in sequential order until the neutron was SCATTERED.

EXTEND

October 2018

- *Enrich component behaviour using EXTEND:*

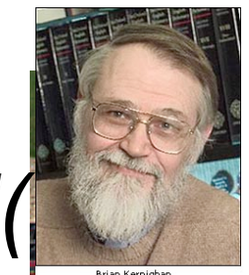
- *COMPONENT Mono1 = Monochromator_curved(...)*



  - *AT (0,0, -LMM) RELATIVE Cradle ROTATED (0,A1/2,0) RELATIVE Cradle*
  - *GROUP IN6Monoks*
  - *EXTEND*
  - *%{*
  - *if (SCATTERED) { myvar = ;1 }*
  - *%}*

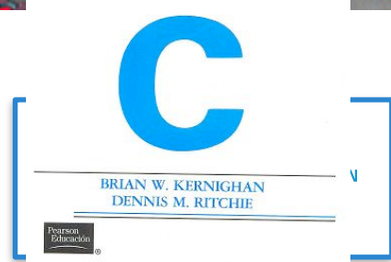
K & R

- *...*
- *COMPONENT Mono2 = Monochromator_curved(*



Brian Kernighan

Dennis Ritchie



  - *AT (0,0, 0) RELATIVE Cradle ROTATED (0,A2/2,0) RELATIVE Cradle*
  - *GROUP IN6Monoks*
  - *%{*
  - *if (SCATTERED) { myvar = 2 }*
  - *%}*

JUMP

- *A goto. Be careful. Can be used in two situations:*
- *JUMP to myself*
- *JUMP to an Arm*

- *No coordinate transformations are applied... (Meaning that if the Arms you JUMP between do not coincide you will “move” / “reorient” the neutrons...)*

- *Syntaxes:*
- *COMPONENT a=b(...)*
- *WHEN (expr) AT (...) JUMP somewhere*

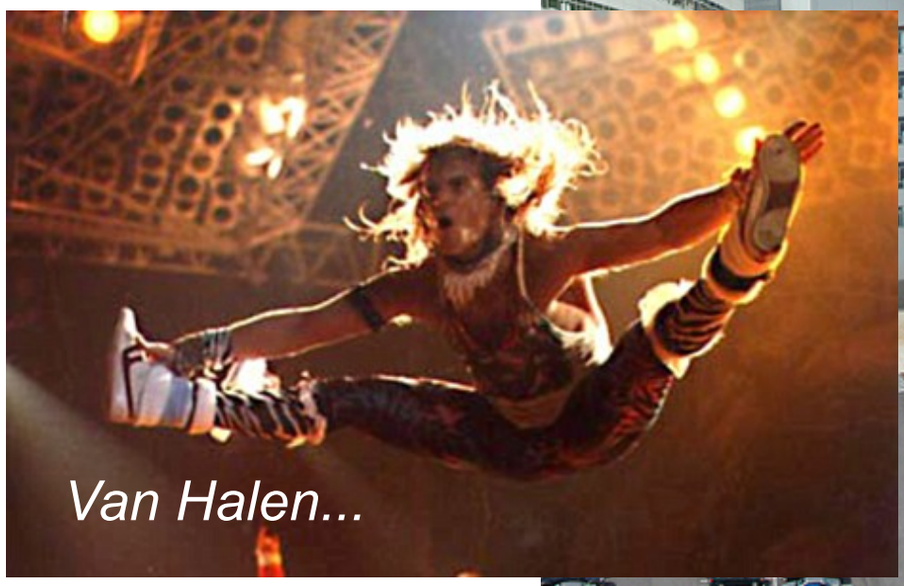
- *COMPONENT a=b(...)*
- *WHEN (expr) AT (...) JUMP somewhere*

JUMP

- A goto. Be careful. Can be used in two situations:
- JUMP to myself
- JU **BEWARE - This IS a GOTO!**

- No coordinate transformations are applied... (Meaning that if the Arms you JUMP between do not coincide you will “move” / “reorient” the neutrons...)

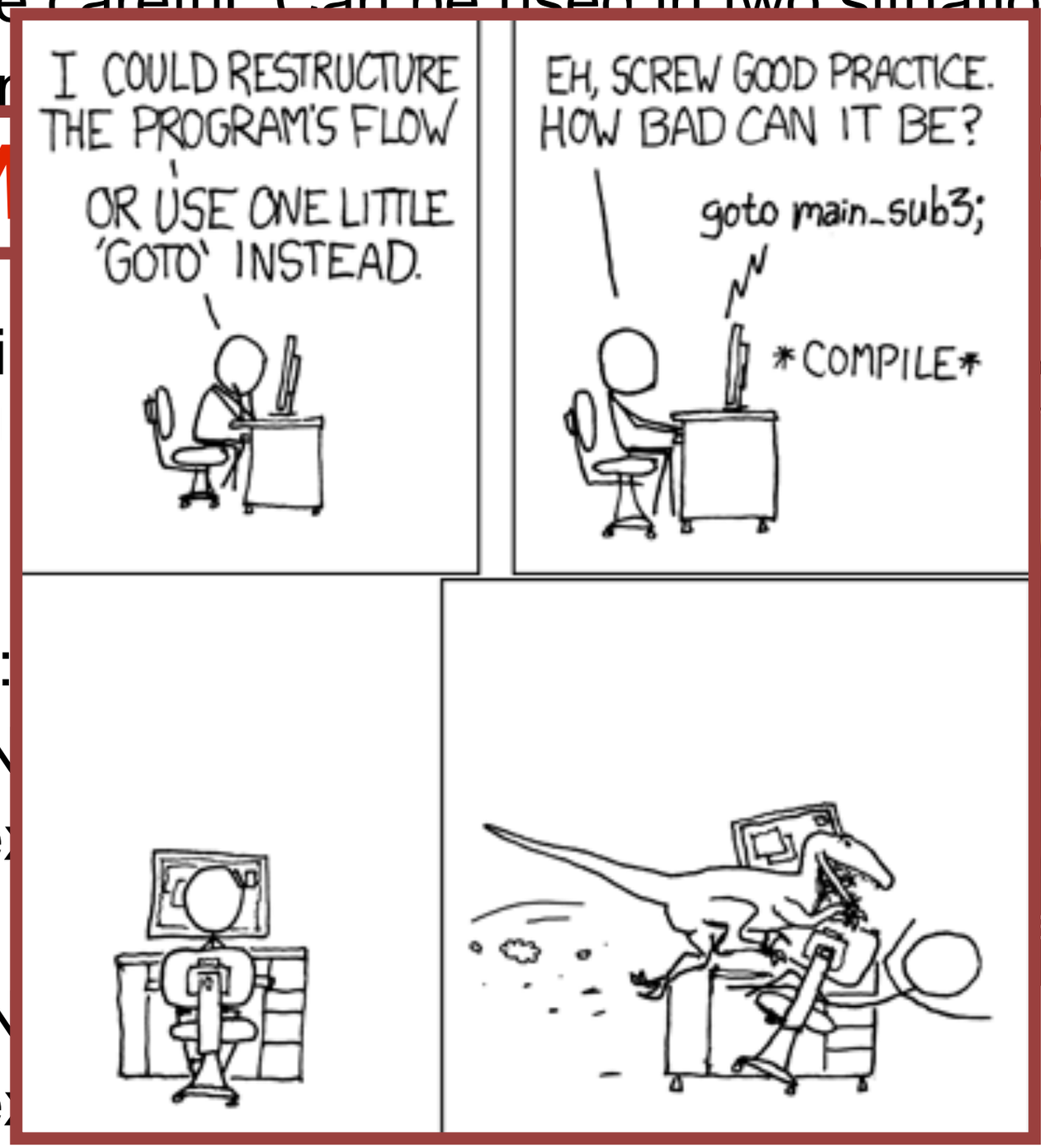
- Syntaxes:
- COMPONENT a=b(...)
- WHEN (expr) AT (...) JUMP somewhere



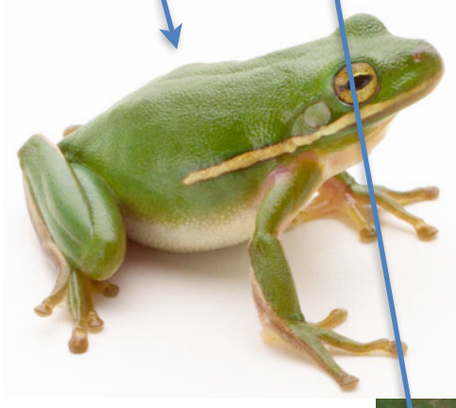
- COMPONENT a=b(...)
- WHEN (expr) AT (...) JUMP somewhere

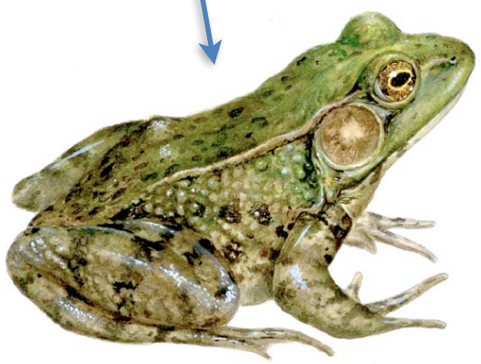
JUMP

- A goto. Be careful. Can be used in two situations:
- JUMP to r
- JU **BEW** **O!**
- No coordi Meaning that if the Arms you will “move” / “reorient”
- Syntaxes:
- COMPON
- WHEN (ex
- COMPON
- WHEN (ex



COPY- inside instruments





- *In instruments: (see ILL_H25.instr)*

- *COMPONENT H25_1 = Guide_gravity(*
- *w1=0.03, h1=0.2, w2=0.03, h2=0.2, l=L_H25_1,*
- *R0=gR0, Qc=gQc, alpha=gAlpha, m=m, W=gW)*
- *AT (0,0,Al_Thickness+gGap) RELATIVE PREVIOUS*
- *ROTATED (0,Rh_H25_1,0) RELATIVE PREVIOUS*

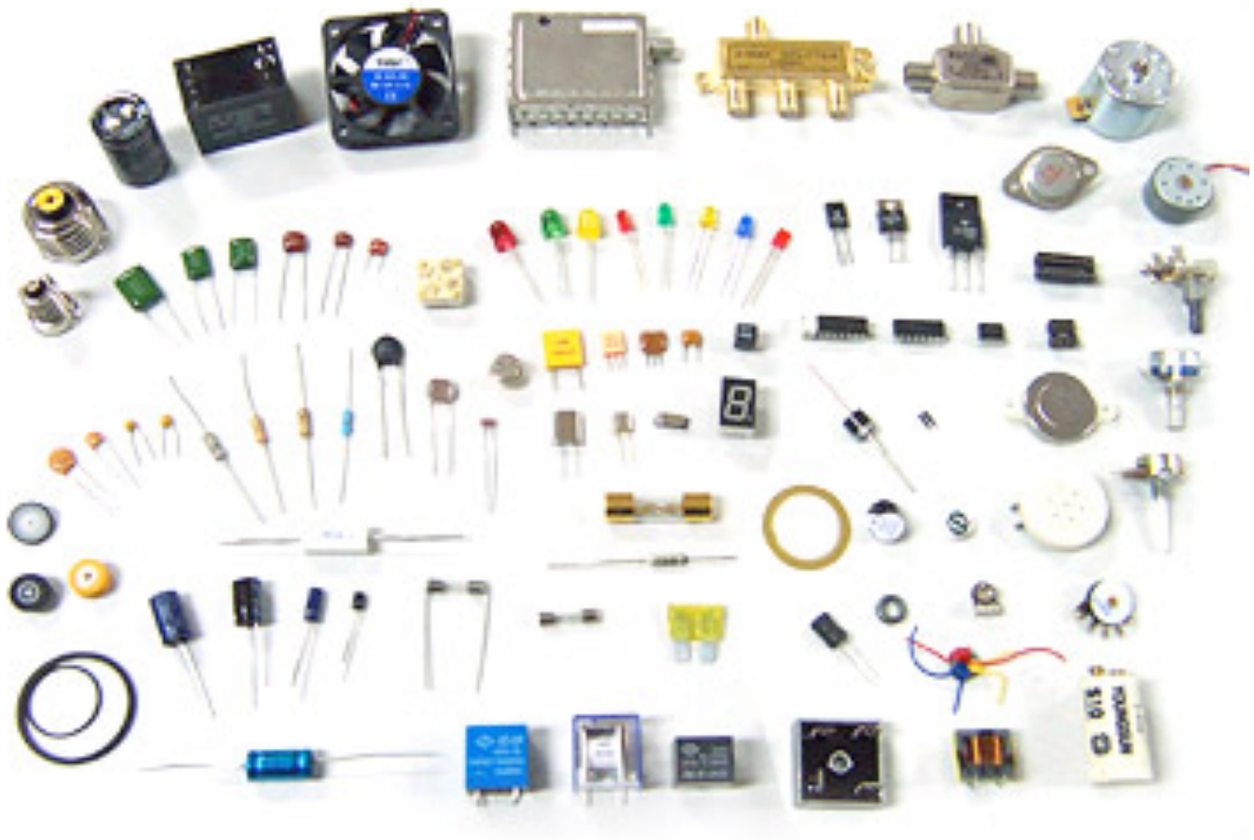
- *COMPONENT COPY(H25_1) = COPY(H25_1)*
- *AT (0,0,L_H25_1+gGap) RELATIVE PREVIOUS*
- *ROTATED (0,Rh_H25_1,0) RELATIVE PREVIOUS*

- *COMPONENT COPY(H25_1) = COPY(H25_1)(W=2*gW)*
- *AT (0,0,L_H25_1+gGap) RELATIVE PREVIOUS*
- *ROTATED (0,Rh_H25_1,0) RELATIVE PREVIOUS*

Other advanced topics

October 2018

- *Tricks, macros and functions for your components*

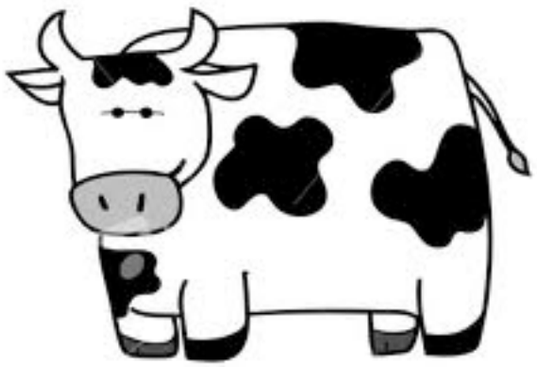


COPY In components:

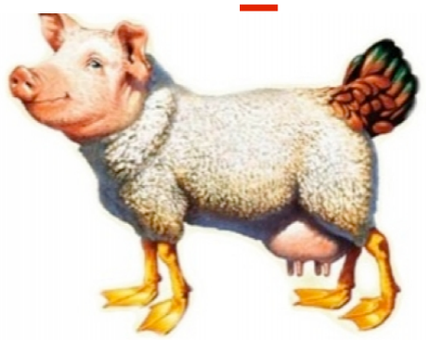
October 2018

+

+



=



There is a heritage mechanism to create childs of existing components. These are exact duplicates of the parent component, but one may override/extend original definitions of any section.

The syntax for a full component child is

```
DEFINE COMPONENT child_name COPY parent_name
```

This single line will copy all parts of the *parent* into the *child*, except for the documentation header.

As for normal component definitions, you may add other parameters, DECLARE, TRACE, ... sections. Each of them will replace or extend (be catenated to, with the COPY/EXTEND words, see example below) the corresponding *parent* definition. In practice, you could ...y a component and only rewrite some of it, as in the following example:

```
DEFINE COMPONENT child_name COPY parent_name


SETTING PARAMETERS (newpar1, newpar2)
INITIALIZE COPY parent_name EXTEND
%{
        ...C code to be catenated to the parent_name INITIALIZE ...
%}
SAVE
%{
        ...C code to replace the parent_name SAVE ...
%}
```