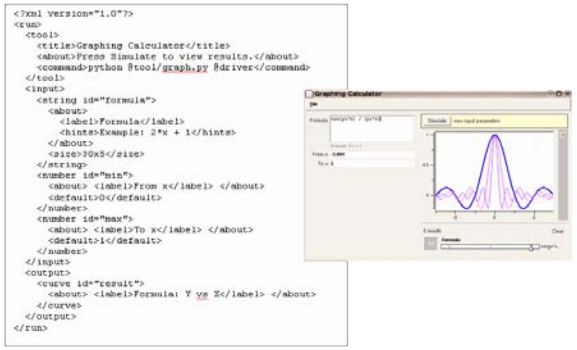
Rappture with C and Fortran



```
<?xml version="1.0"?>
<run>
  <tool>
    <title>Graphing Calculator</title>
    <about>Press Simulate to view results.</about>
    <command>python @tool/graph.py @driver</command>
  </tool>
  <input>
    <string id="formula">
      <about>
        <label>Formula</label>
        <hints>Example: 2*x + 1</hints>
      </about>
      <size>30x5</size>
    </string>
    <number id="min">
      <about> <label>From x</label> </about>
      <default>0</default>
    </number>
    <number id="max">
      <about> <label>To x</label> </about>
      <default>1</default>
    </number>
  </input>
  <output>
    <curve id="result">
      <about> <label>Formula: Y vs X</label> </about>
    </curve>
  </output>
</run>
```

Michael McLennan

HUBzero® Platform for Scientific Collaboration

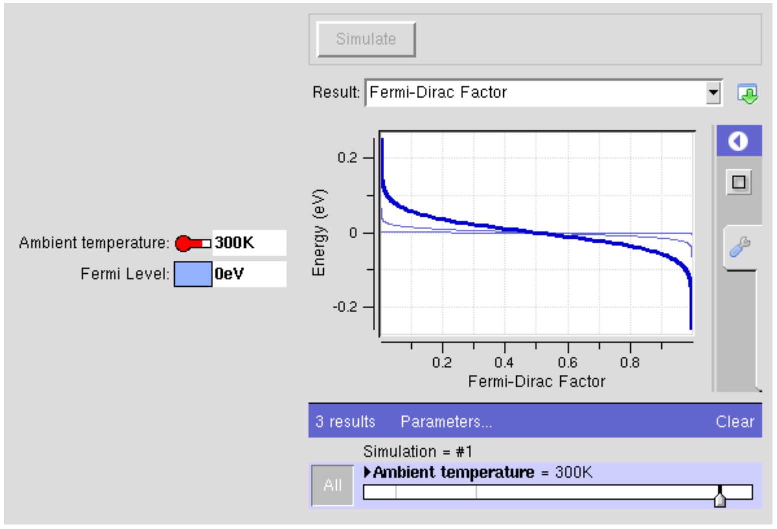
Purdue University

Example: app-fermi in C

The usual way…

```
% gcc fermi.c -o fermi -lm
% ./fermi
Enter the Fermi level (eV):
 Ef = 2.4
Enter the temperature (K):
 T = 77

% more out.dat
FERMI-DIRAC FUNCTION F1/2

f1/2           Energy (eV)
------------   ------------
    0.999955        2.33365
     0.99995        2.33431
    0.999944        2.33498
```

The Rappture way…



https://nanohub.org/infrastructure/rappture

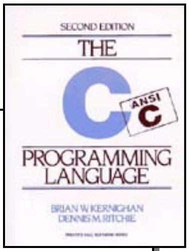
source code: rappture/examples/app-fermi

What is the interface?

fermi.c

```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[]) {
    double T, Ef, E, dE, kT, Emin, Emax, f;

    printf("Enter Fermi energy in eV:\n");
    scanf("%lg", &Ef);

    printf("Enter the Temperature in K:\n");
    scanf("%lg", &T);

    kT = 8.61734e-5 * T;
    Emin = Ef - 10*kT;
    Emax = Ef + 10*kT;
    E = Emin;
    dE = 0.005*(Emax-Emin);
    while (E < Emax) {
        f = 1.0/(1.0 + exp((E - Ef)/kT));

        printf("%f %f\n",f, E);
        E = E + dE;
    }
    return 0;
}
```

physics

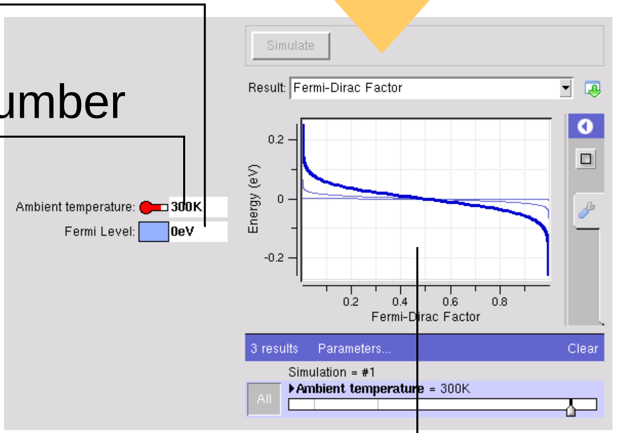


number

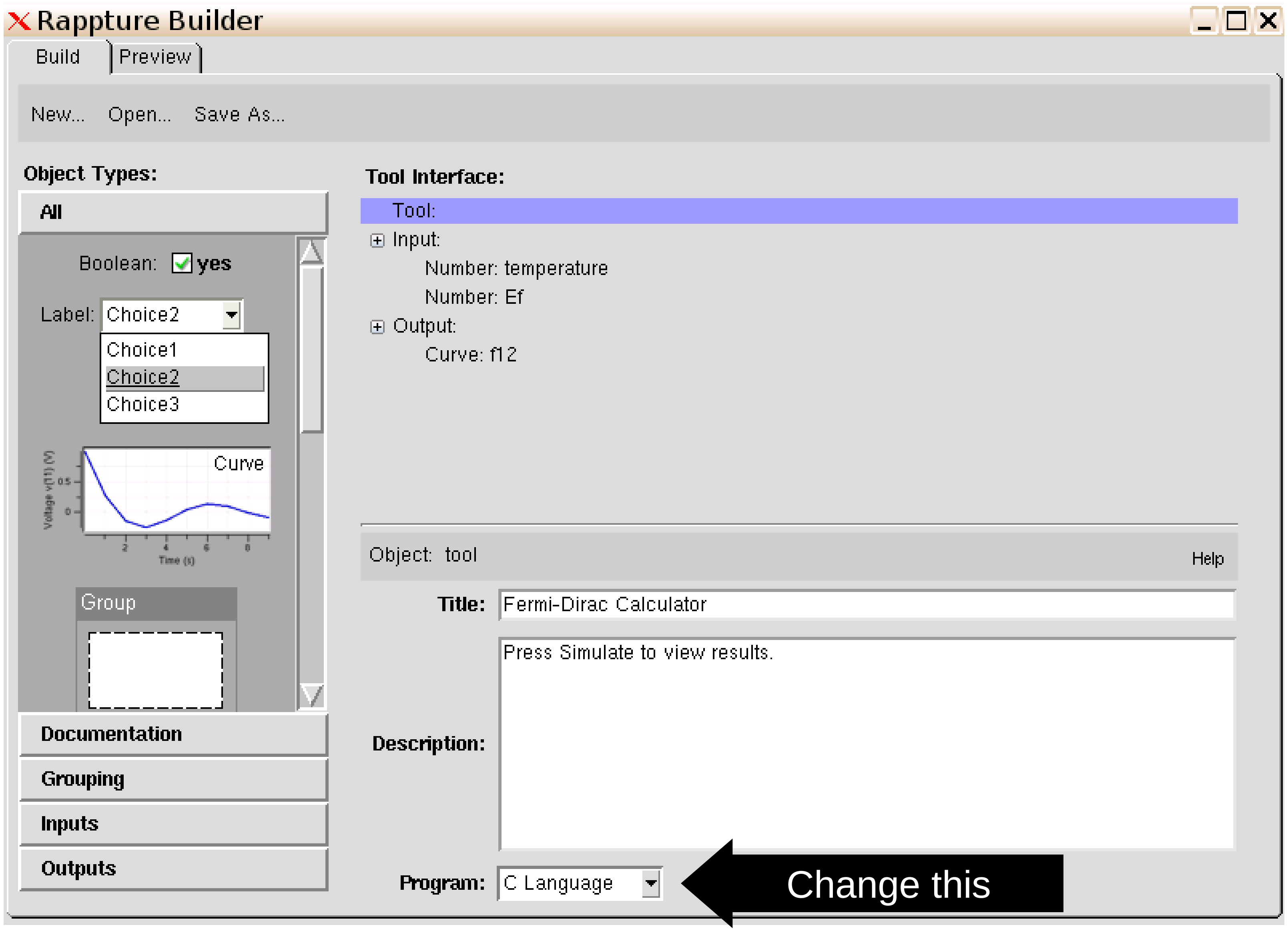
number

curve

Inputs/outputs are identical to the MATLAB version



Same interface - Use C Language



Tool Interface:

Tool:
- Input:
  - Number: temperature
  - Number: Ef
- Output:
  - Curve: f12

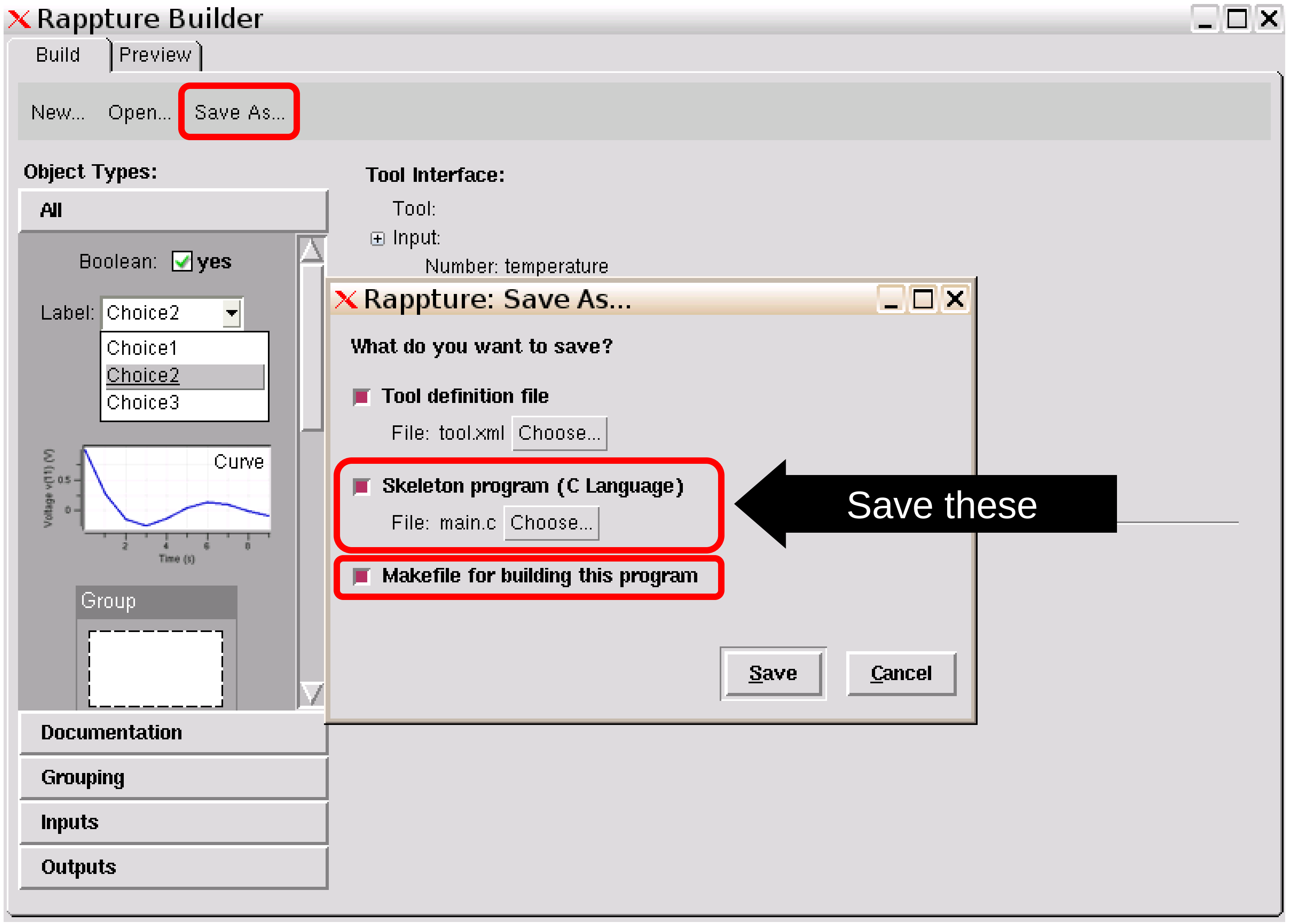
Object: tool

Title: Fermi-Dirac Calculator

Description: Press Simulate to view results.

Program: C Language

Generate program and Makefile



Add physics to the generated code

Generated program: main.c

…

```
    /* get input value for input.number(temperature) and convert to K */
    rpGetString(io,"input.number(temperature).current", &data);
    temperature = rpConvertDbl(data, "K", &err);

    /* get input value for input.number(Ef) and convert to eV */
    rpGetString(io,"input.number(Ef).current", &data);
    Ef = rpConvertDbl(d

    /*
     *****************
     *  Add your code h
     *****************
     */


    /* save output valu
    /* this shows just
    sprintf(line, "%g %g\n", x, y);
    rpPutString(io,"output.curve(f12).component.xy", line, RPLIB_APPEND);

    rpResult(io);
    exit(0);
}
```

physics

```
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;
E = Emin;
dE = 0.005*(Emax-Emin);
while (E < Emax) {
    f = 1.0/(1.0 + exp((E - Ef)/kT));
    E = E + dE;
}
```

Add physics to the generated code

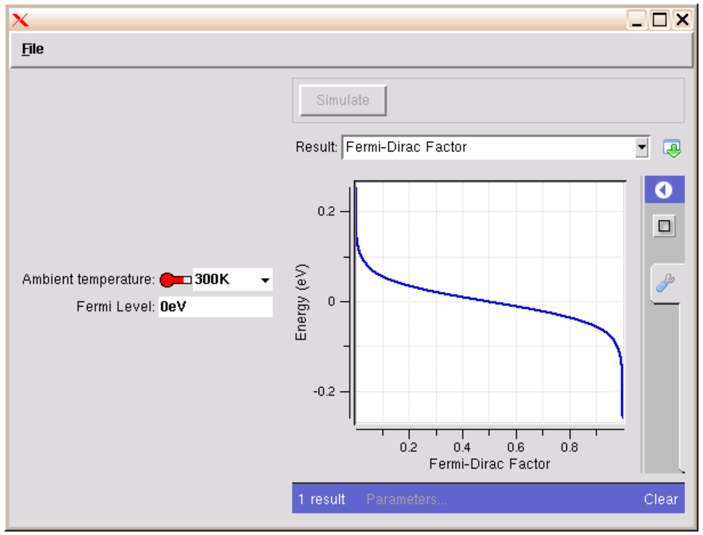
Generated program: main.c

```
…
    /* get input value for input.number(temperature) and convert to K */
    rpGetString(io,"input.number(temperature).current", &data);
    temperature = rpConvertDbl(data, "K", &err);

    /* get input value for input.number(Ef) and convert to eV */
    rpGetString(io,"input.number(Ef).current", &data);
    Ef = rpConvertDbl(data, "eV", &err);

    kT = 8.61734e-5 * temperature;
    Emin = Ef - 10*kT;
    Emax = Ef + 10*kT;
    E = Emin;
    dE = 0.005*(Emax-Emin);
    while (E < Emax) {
        f = 1.0/(1.0 + exp((E - Ef)/kT));
        /* save output value for output.curve(f12) */
        sprintf(line, "%g %g\n", f, E);
        rpPutString(io,"output.curve(f12).component.xy", line,
                    RPLIB_APPEND);
        E = E + dE;
    }
    rpResult(io);
    exit(0);
}
```

Build the program

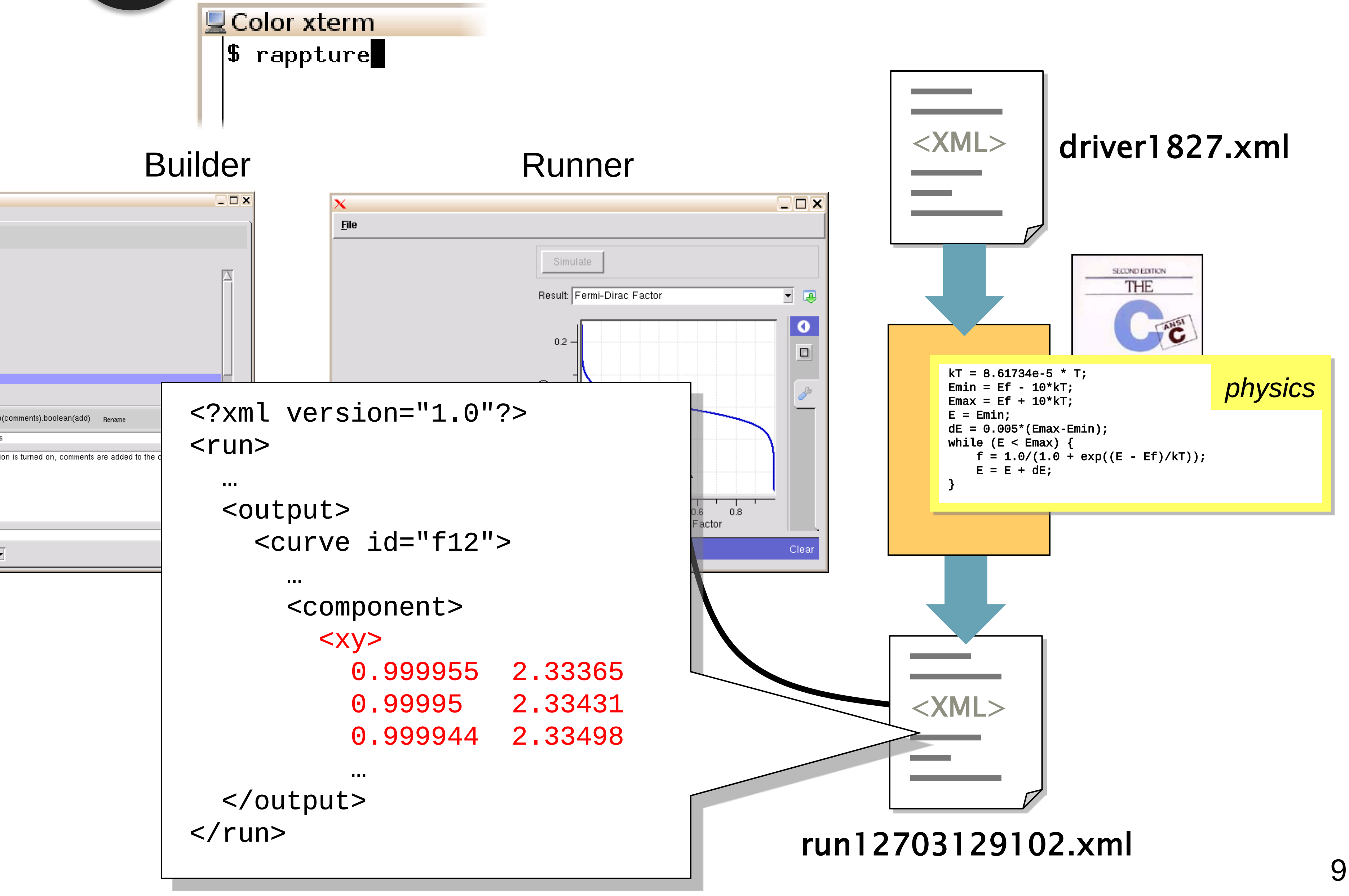
Don’t forget to “make” the program!

```
% make
gcc -g -Wall -I/apps/rappture/20110405/include
main.c -o mainc -L/apps/rappture/20110405/lib
-lrappture -lm

% ls
Makefile  tool.xml  main.c  mainc

% rappture
```



Ambient temperature: 300K
Fermi Level: 0eV

Running the code

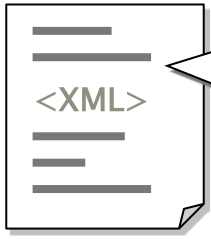


```
$ rappture
```

```
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;
E = Emin;
dE = 0.005*(Emax-Emin);
while (E < Emax) {
    f = 1.0/(1.0 + exp((E - Ef)/kT));
    E = E + dE;
}
```

```
<?xml version="1.0"?>
<run>
  …
  <output>
    <curve id="f12">
      …
      <component>
        <xy>
          0.999955   2.33365
          0.99995    2.33431
          0.999944   2.33498
          …
  </output>
</run>
```

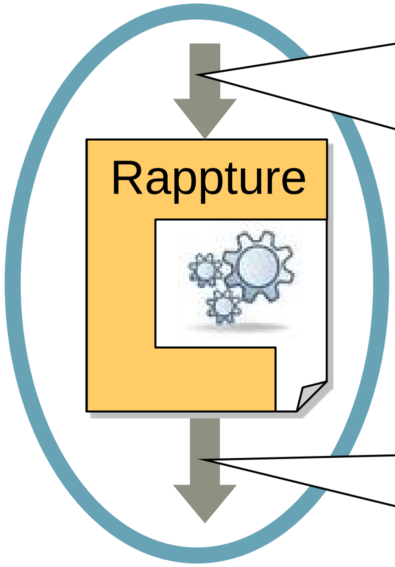
How your program gets invoked



tool.xml

```
<?xml version="1.0"?>
<run>
  <tool>
    <about>Press Simulate to view results.</about>
    <command>@tool/mainc @driver</command>
  </tool>
  <input>
    …
```

```
/apps/yourtool/current/mainc driver327.xml
```



```
int main(int argc, char * argv[]) {
    …
    /* open the XML file */
    io = rpLibrary(argv[1]);
    …
    rpGetString(io,"input.number(temperature).current",
&data);
    temperature = rpConvertDbl(data, "K", &err);
    …
```

```
    rpPutString(io,"output.curve(f12).component.xy",
line, RPLIB_APPEND);
    rpResult(io);
    exit(0);
}
```

Not Much Overhead

It’s easy to add Rappture to your new tools:

Instead of this…

Documentation

```
scanf("%lg",&Ef);
```

```
printf(“%g”,Ef);
```

Do this…

```
<number id="Ef">
  <about>
    <label>Fermi Level</label>
  </about>
  <units>eV</units>
  <default>0eV</default>
</number>
```

```
rpGetString(lib, path, &data);
Ef = rpConvertDbl(data, “eV", &err);
```

```
rpPutString(lib, path, val, RPLIB_APPEND);
…
rpResult(lib);
```

Fortran version

fermi.f

Fortran F77

```
 program fermi

       write(6,*) 'Fermi energy in eV:'
       read(5,*) Ef

       write(6,*) 'Temperature in K:'
       read(5,*) T

       kT = 8.61734e-5 * T
       Emin = Ef - 10*kT
       Emax = Ef + 10*kT

       dE = 0.005*(Emax - Emin)

       do 10 E=Emin,Emax,dE
         f = 1.0/(1.0+exp((E-Ef)/kT))

         write(6,*) f, E

10     continue

     end program fermi
```

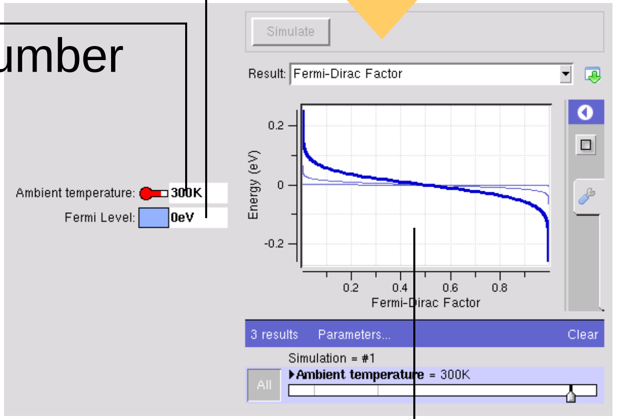
physics

number

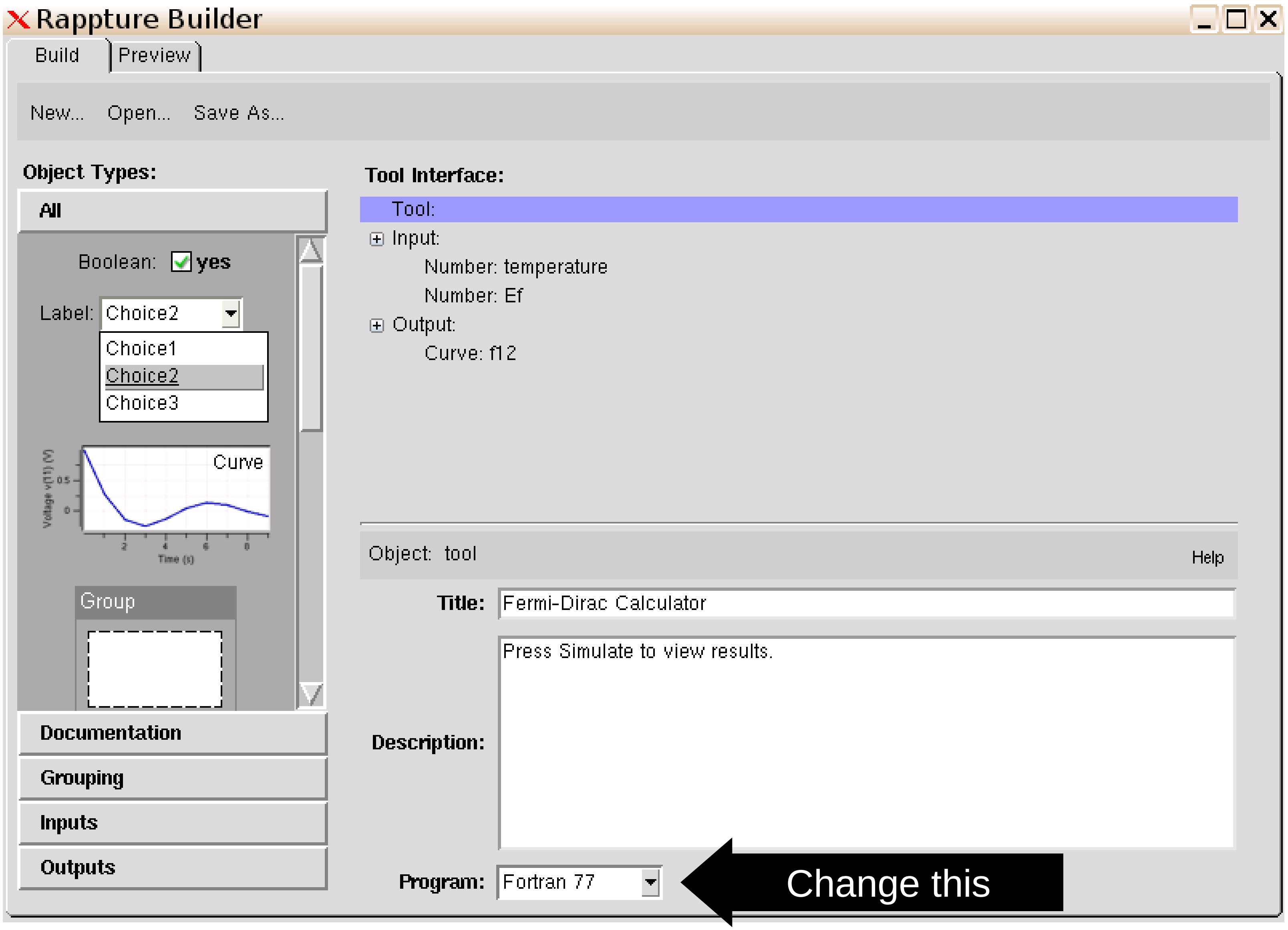
number

curve

Inputs/outputs are identical to the other versions



Same interface - Use C Language


Rappture Builder

Build | Preview

New... Open... Save As...

Object Types:

Tool Interface:

Tool:
⊞ Input:
Number: temperature
Number: Ef
⊞ Output:
Curve: f12

Object: tool Help

Title: Fermi-Dirac Calculator

Description: Press Simulate to view results.

Program: Fortran 77

Change this

Add physics to the generated code

Generated program: main.f

```
…
c        get input value for input.number(temperature) and convert to K
         call rp_lib_get(io,
     +     "input.number(temperature).current", strVal)
         ok = rp_units_convert_dbl(strVal,"K", temperature)

c        get input value for input.number(Ef) and convert to eV
         call rp_lib_get
     +     "input.number
         ok = rp_units_c

c        ---------------
c           Add your cod
c        ----------

c        save output val
c        this shows just one (x,y) point -- modify as needed
         write(strVal,'(E20.12,E20.12,A)') x, y, char(10)
         call rp_lib_put_str(io,
     +     "output.curve(f12).component.xy",strVal,1)

         call rp_result(io)
       end program main
```

physics

```
kT = 8.61734e-5 * T
Emin = Ef - 10*kT
Emax = Ef + 10*kT

dE = 0.005*(Emax - Emin)

do 10 E=Emin,Emax,dE
  f = 1.0/(1.0+exp((E-Ef)/kT))
```

Add physics to the generated code

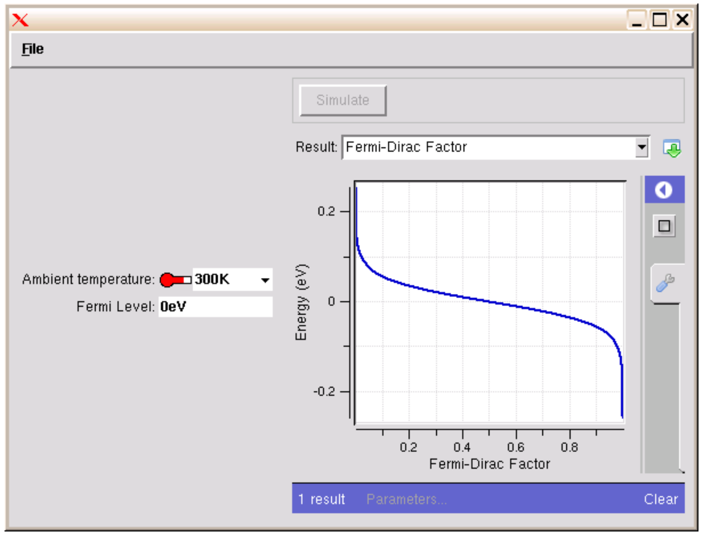
Generated program: main.f

```
…
c       get input value for input.number(temperature) and convert to K
        call rp_lib_get(io,
     +    "input.number(temperature).current", strVal)
        ok = rp_units_convert_dbl(strVal,"K",temperature)

c       get input value for input.number(Ef) and convert to eV
        call rp_lib_get(io,
     +    "input.number(Ef).current", strVal)
        ok = rp_units_convert_dbl(strVal,"eV",Ef)

        kT = 8.61734e-5 * temperature
        Emin = Ef - 10*kT
        Emax = Ef + 10*kT
        dE = 0.005*(Emax - Emin)

        do 10 E=Emin,Emax,dE
          f = 1.0/(1.0+exp((E-Ef)/kT))
c         save output value for output.curve(f12)
          write(strVal,'(E20.12,E20.12,A)') f, E, char(10)
          call rp_lib_put_str(io,
     +      "output.curve(f12).component.xy",strVal,1)
10      continue
…
```

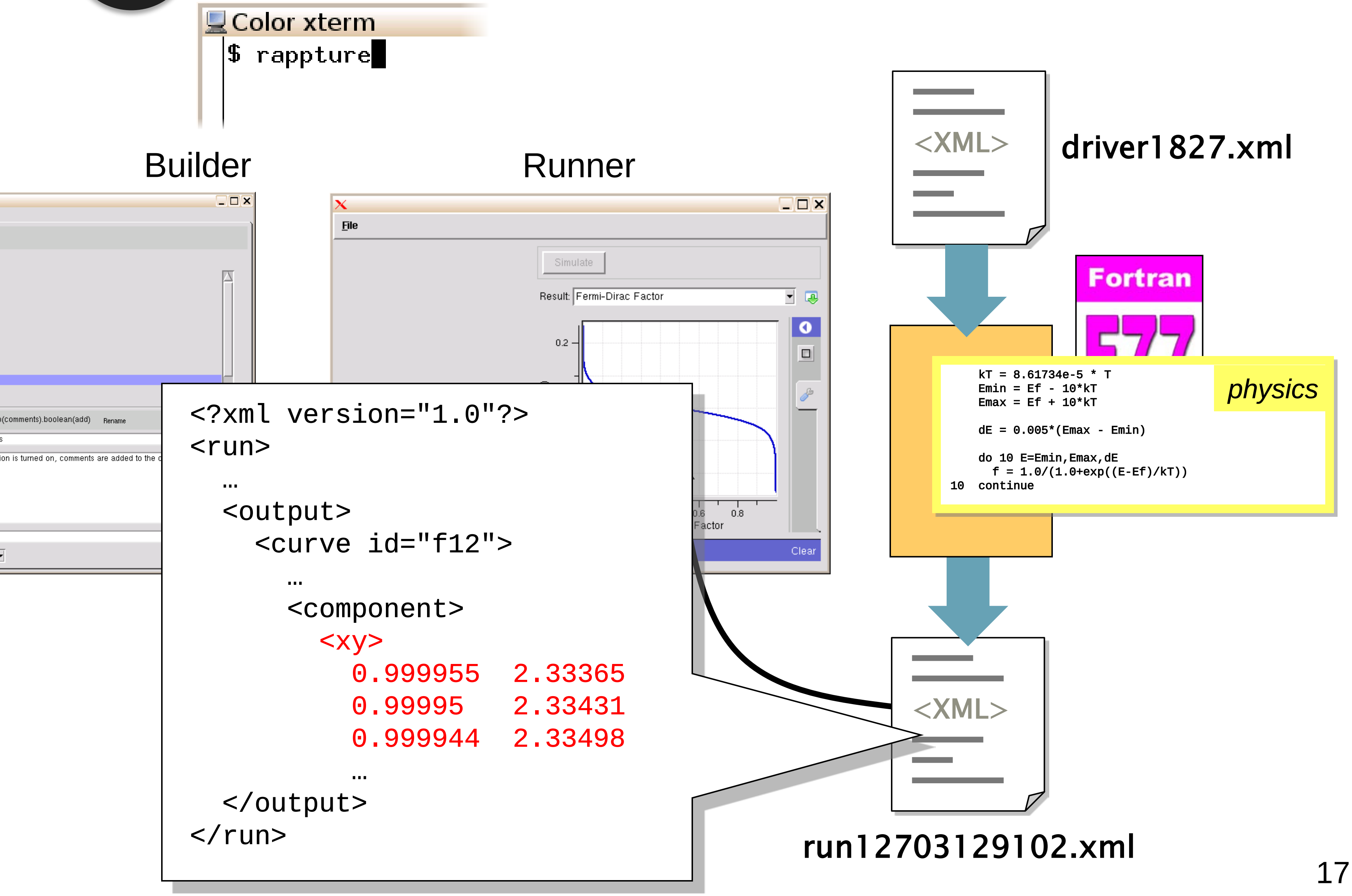
Build the program

Don’t forget to “make” the program!

```
% make
gfortran -g -Wall -I/apps/rappture/20110405
/include main.f -o mainf77 -L/apps/rappture
/20110405/lib -lrappture -lm

% ls
Makefile  tool.xml  main.f  mainf77

% rappture
```


Ambient temperature: 300K
Fermi Level: 0eV
Result: Fermi-Dirac Factor

Running the code

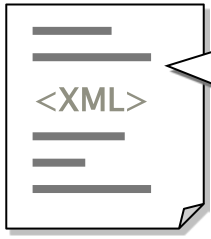


```
$ rappture
```

```
<?xml version="1.0"?>
<run>
  …
  <output>
    <curve id="f12">
      …
      <component>
        <xy>
          0.999955   2.33365
          0.99995    2.33431
          0.999944   2.33498
          …
  </output>
</run>
```

```
      kT = 8.61734e-5 * T
      Emin = Ef - 10*kT
      Emax = Ef + 10*kT

      dE = 0.005*(Emax - Emin)

      do 10 E=Emin,Emax,dE
        f = 1.0/(1.0+exp((E-Ef)/kT))
10    continue
```

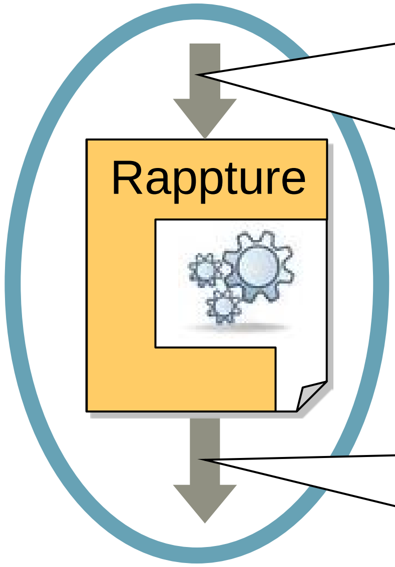
How your program gets invoked



tool.xml

```
<?xml version="1.0"?>
<run>
  <tool>
    <about>Press Simulate to view results.</about>
    <command>@tool/mainf77 @driver</command>
  </tool>
  <input>
    …
```

```
/apps/yourtool/current/mainf77 driver327.xml
```



```
 program main
        IMPLICIT NONE
        …
c       open the XML file containing the run parameters
        call getarg(1,inFile)
        io = rp_lib(inFile)
        if (io .eq. 0) then
            write(6,*) "FAILED loading Rappture data”
        …
```

```
        call rp_lib_put_str(io,
    +       "output.curve(f12).component.xy",strVal,1)
10      continue
        call rp_result(io)
      end program main
```

Not Much Overhead

It’s easy to add Rappture to your new tools:

Instead of this…

Documentation

```
read(5,*) Ef
```

```
write(6,*) Ef
```

Do this…

```
<number id="Ef">
  <about>
    <label>Fermi Level</label>
  </about>
  <units>eV</units>
  <default>0eV</default>
</number>
```

```
call rp_lib_get(driver, path, str)
ok = rp_units_convert_dbl(str, “eV”, Ef)
```

```
call rp_lib_put_str(driver, path, val, 0)
…
rp_result(driver)
```

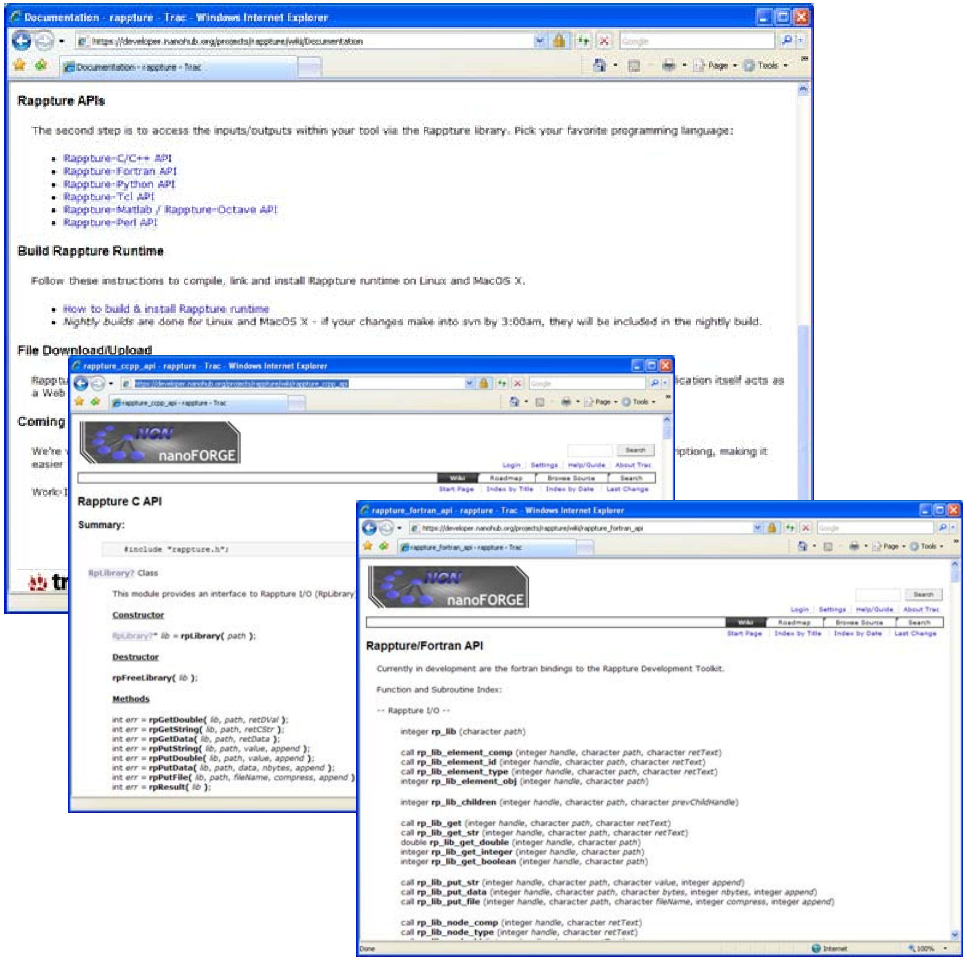
Reference Documentation

http://rappture.org

- Documentation
- Rappture-C/C++ API
- Rappture-Fortran API

Examples:

- app-fermi in C
- app-fermi in Fortran


Documentation - rappture - Trac - Windows Internet Explorer
https://developer.nanohub.org/projects/rappture/wiki/Documentation
Page
Tools

Rappture APIs

The second step is to access the inputs/outputs within your tool via the Rappture library. Pick your favorite programming language:

- Rappture-C/C++ API
- Rappture-Fortran API
- Rappture-Python API
- Rappture-Tcl API
- Rappture-Matlab / Rappture-Octave API
- Rappture-Perl API

Build Rappture Runtime

Follow these instructions to compile, link and install Rappture runtime on Linux and MacOS X.

- How to build & install Rappture runtime
- *Nightly builds* are done for Linux and MacOS X - if your changes make into svn by 3:00am, they will be included in the nightly build.

File Download/Upload

Rappt … lication itself acts as a Web

Coming

We're … ptiong, making it easier

Work: I

rappture_ccpp_api - rappture - Trac - Windows Internet Explorer
https://developer.nanohub.org/projects/rappture/wiki/rappture_ccpp_api
nanoFORGE
Search
Login | Settings | Help/Guide | About Trac
Wiki | Roadmap | Browse Source | Search
Start Page | Index by Title | Index by Date | Last Change

Rappture C API

Summary:

```
#include "rappture.h";
```

RpLibrary? Class

This module provides an interface to Rappture I/O (RpLibrary)

Constructor

RpLibrary?* *lib* = **rpLibrary(** *path* **);**

Destructor

rpFreeLibrary(*lib* **);**

Methods

int *err* = **rpGetDouble(** *lib, path, retDVal* **);**
int *err* = **rpGetString(** *lib, path, retCStr* **);**
int *err* = **rpGetData(** *lib, path, retData* **);**
int *err* = **rpPutString(** *lib, path, value, append* **);**
int *err* = **rpPutDouble(** *lib, path, value, append* **);**
int *err* = **rpPutData(** *lib, path, data, nbytes, append* **);**
int *err* = **rpPutFile(** *lib, path, fileName, compress, append* **);**
int *err* = **rpResult(** *lib* **);**

rappture_fortran_api - rappture - Trac - Windows Internet Explorer
https://developer.nanohub.org/projects/rappture/wiki/rappture_fortran_api
nanoFORGE
Search
Login | Settings | Help/Guide | About Trac
Wiki | Roadmap | Browse Source | Search
Start Page | Index by Title | Index by Date | Last Change
Done
Internet
100%

Rappture/Fortran API

Currently in development are the fortran bindings to the Rappture Development Toolkit.

Function and Subroutine Index:

-- Rappture I/O --

integer **rp_lib** (character *path*)

call **rp_lib_element_comp** (integer *handle*, character *path*, character *retText*)
call **rp_lib_element_id** (integer *handle*, character *path*, character *retText*)
call **rp_lib_element_type** (integer *handle*, character *path*, character *retText*)
integer **rp_lib_element_obj** (integer *handle*, character *path*)

integer **rp_lib_children** (integer *handle*, character *path*, character *prevChildHandle*)

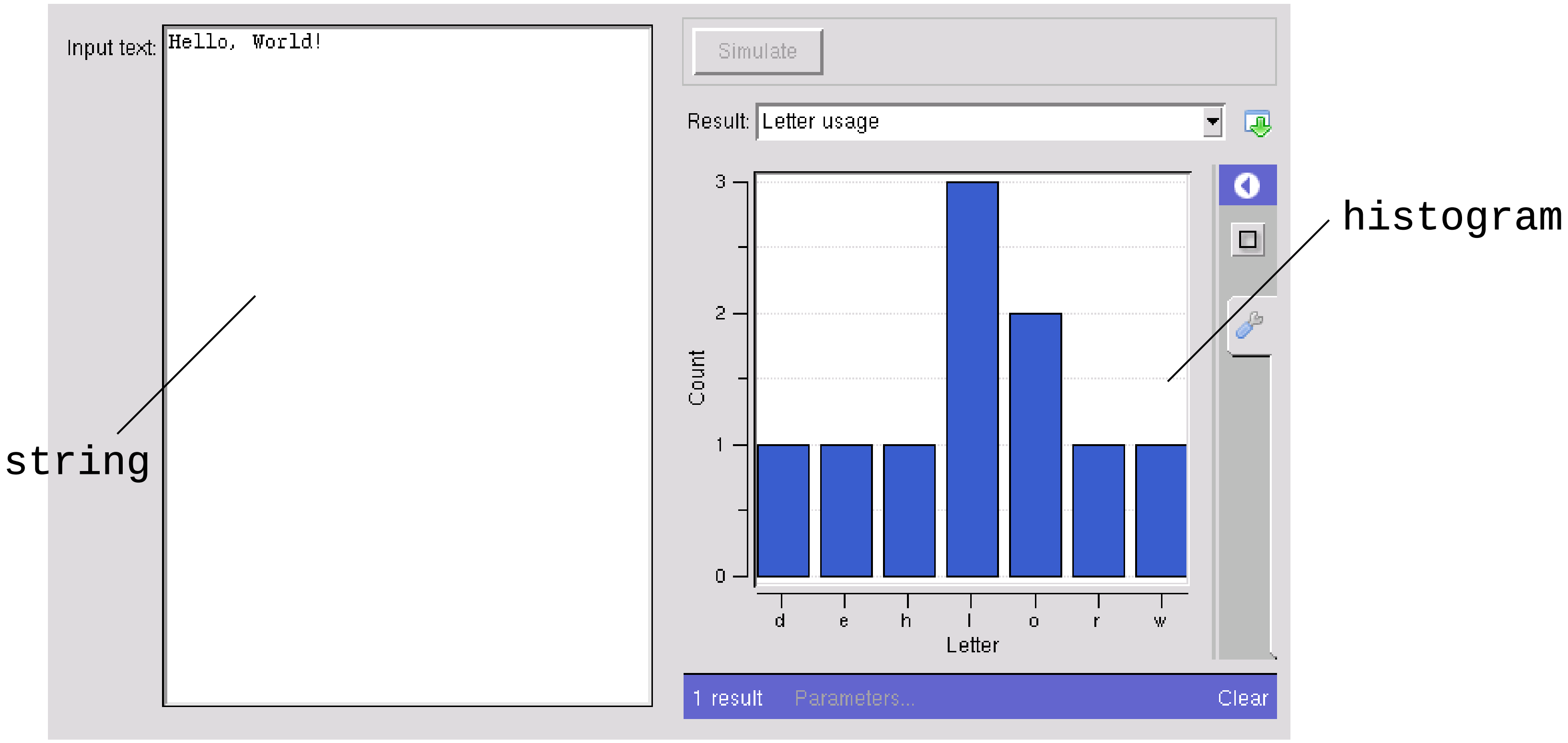
call **rp_lib_get** (integer *handle*, character *path*, character *retText*)
call **rp_lib_get_str** (integer *handle*, character *path*, character *retText*)
double **rp_lib_get_double** (integer *handle*, character *path*)
integer **rp_lib_get_integer** (integer *handle*, character *path*)
integer **rp_lib_get_boolean** (integer *handle*, character *path*)

call **rp_lib_put_str** (integer *handle*, character *path*, character *value*, integer *append*)
call **rp_lib_put_data** (integer *handle*, character *path*, character *bytes*, integer *nbytes*, integer *append*)
call **rp_lib_put_file** (integer *handle*, character *path*, character *fileName*, integer *compress*, integer *append*)

call **rp_lib_node_comp** (integer *handle*, character *retText*)
call **rp_lib_node_type** (integer *handle*, character *retText*)

Assignment #6: Rappture interface for C or Fortran

Create a Rappture interface for your letter counting program:



Extra Credit: Add a `number` output to report the number of words