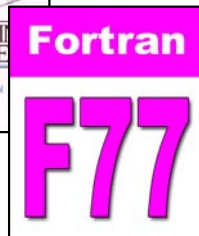
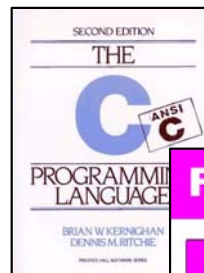
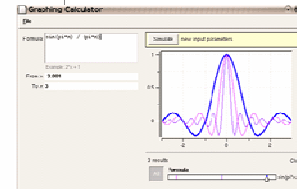


Rappture with C and Fortran



```

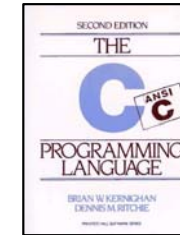
<?xml version="1.0"?>
<run>
  <code>
    <title>Graphing Calculator</title>
    <about>Press Simulate to view results.</about>
    <command>python $root/graph.py $driver</command>
  </code>
  <input>
    <string id="formula">
      <about>
        <label>Formula</label>
        <hint>Example: 2*X + 1</hint>
      </about>
      <size>20x</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
Software Architect
HUBzero™ Platform for Scientific Collaboration

Example: app-fermi

```
<?xml version="1.0"?>
<run>
  <tool >
    <about>Press Simulate to view results.</about>
    <command>@tool/fermi @driver</command>
  </tool >
  <input>
    <number id="temperature">...</number>
    <number id="Ef">...</number>
  </input>
  <output>
    <curve id="f12">...</curve>
  </output>
</run>
```



Ambient temperature:
Fermi Level:

See [example code](#)

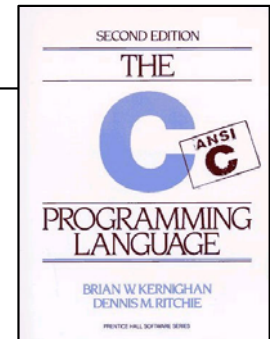
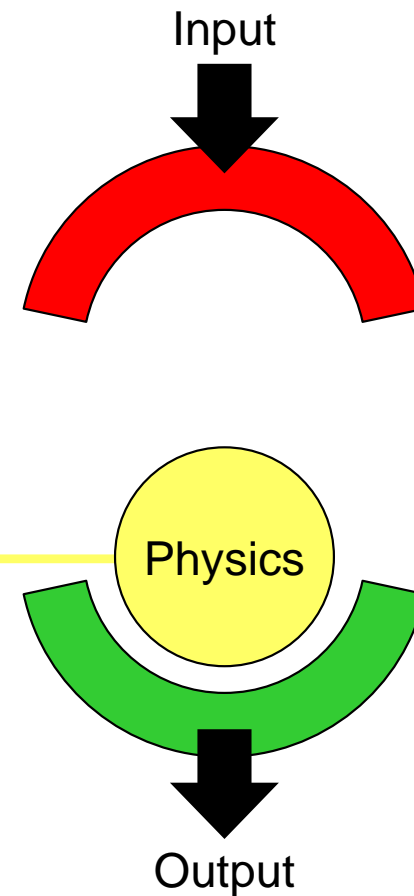
Example: app-fermi

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

Starting Point

fermi.c (simulator written in C)

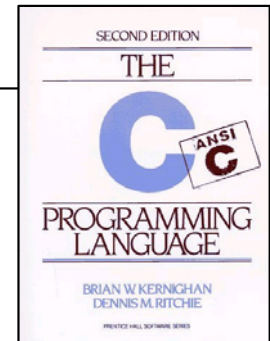
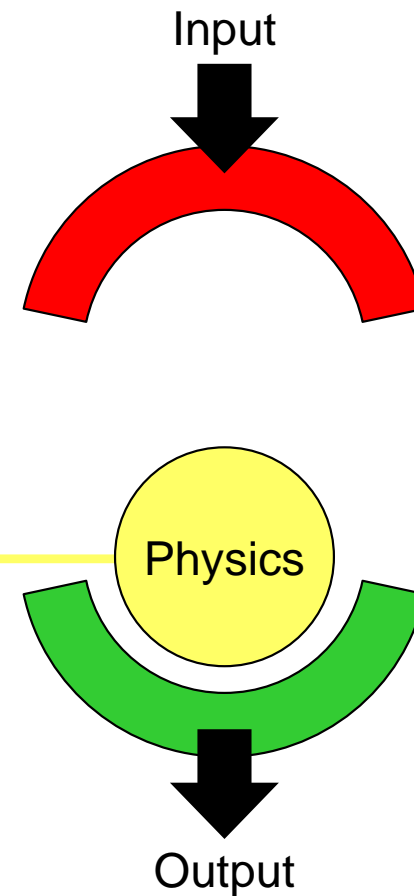
```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[]) {
    double T, Ef, E, dE, kT, Emin, Emax, f;
    printf("Enter the Fermi energy in eV: \n");
    scanf("%lg", &Ef);
    printf("Enter the Temperature in K: \n");
    scanf("%lg", &T);
    kT = 8.61734e-5 * T;
    Emin = Ef - 10*kT;
    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;
}
```



Starting Point

fermi.c (simulator written in C)

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



Rapture Inputs

fermi.c (simulator written in C)

```

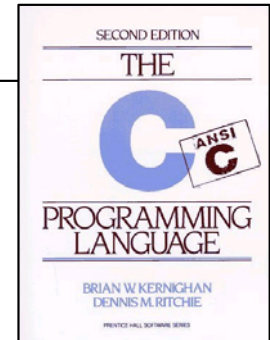
#include "rappture.h"
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[]) {
    double T, Ef, E, dE, kT, Emin, Emax, f; int err;
    RpLibrary *lib;
    const char* data;

    lib = rpLibrary(argv[1]);

    rpGetString(lib, "input. (temperature). current", &data);
    T = rpConvertDbl (data, "K", &err);

    rpGetString(lib, "input. (Ef). current", &data);
    Ef = rpConvertDbl (data, "eV", &err);

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

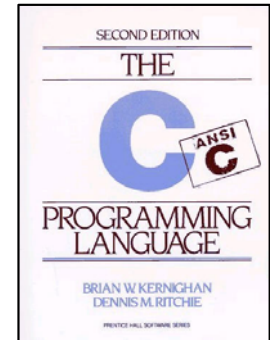


"72F"
295.22
"2.4eV"
2.4

Rappture Outputs

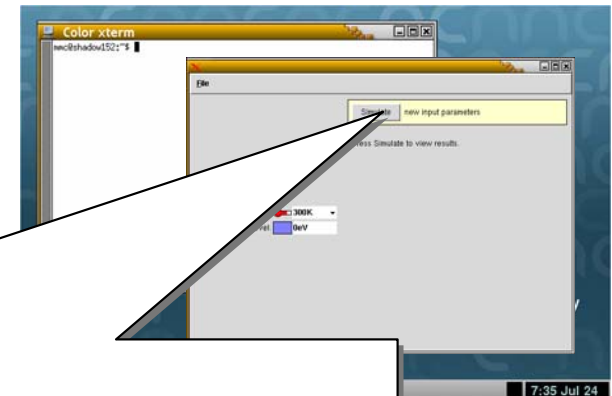
fermi.c (simulator written in C)

```
...
E = E_min;
dE = 0.005*(E_max-E_min);
while (E < E_max) {
    f = 1.0/(1.0 + exp((E - E_f)/kT));
    sprintf(line, "%f %f\n", f, E);      "x y \n"
    rpPutString(lib, "output.curve(f12).component.xy", line,
                RPLIB_APPEND);
    E = E + dE;
}
rpResult(lib);
return 0;
}
```



Compiling the Rappturized Code

From your workspace on nanoHUB...

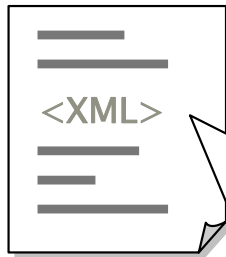


```
% cd tooldir
% ls
tool.xml fermi.c

% gcc -g fermi.c -o fermi -I/apps/rappture/current/include
-L/apps/rappture/current/lib -lrappture -lm
% ls
tool.xml fermi.c fermi

% rappture
```

How does the program get invoked?



tool.xml

Replaced with directory
containing tool.xml file

Ex: /apps/yourtool /current

Replaced with name of driver file

Ex: dri ver327. xml

```
<?xml version="1.0"?>
<run>
  <tool >
    <about>Press Simulate to view results.</about>
    <command>@tool /fermi @driver </command>
  </tool >
  <i nput>
    ...
  </i nput>
  <output>
    ...
  </output>
</run>
```

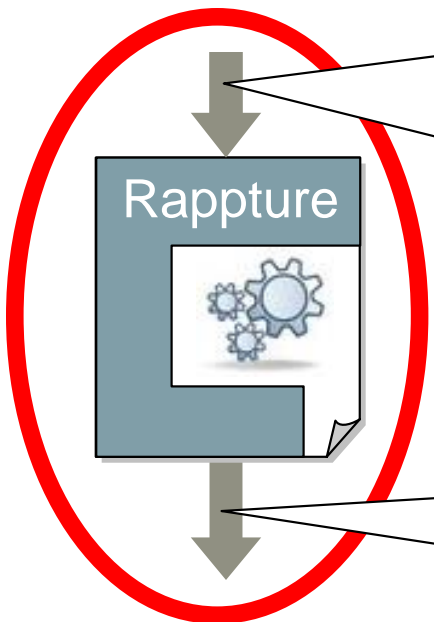
...with the driver file as the first argument

Invoke the C program...

How does the program get invoked?

```
@tool /fermi @driver
```

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



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

```
...
```

```
int main(int argc, char *argv[]) {
    double T, Ef, E, dE, kT, Emin, Emax, f;
    RpLibrary *lib;
    const char* data;
    lib = rpLibrary(argv[1]);
    ...
```

```
...
```

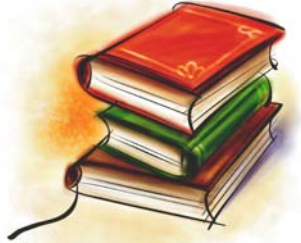
```
rpResult(lib);
return 0;
```

```
}
```

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);
```

Example: app-fermi

```
program fermi

  read(5, *) Ef
  read(5, *) T

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

  dE = 0.005*(Emax - Emi n)

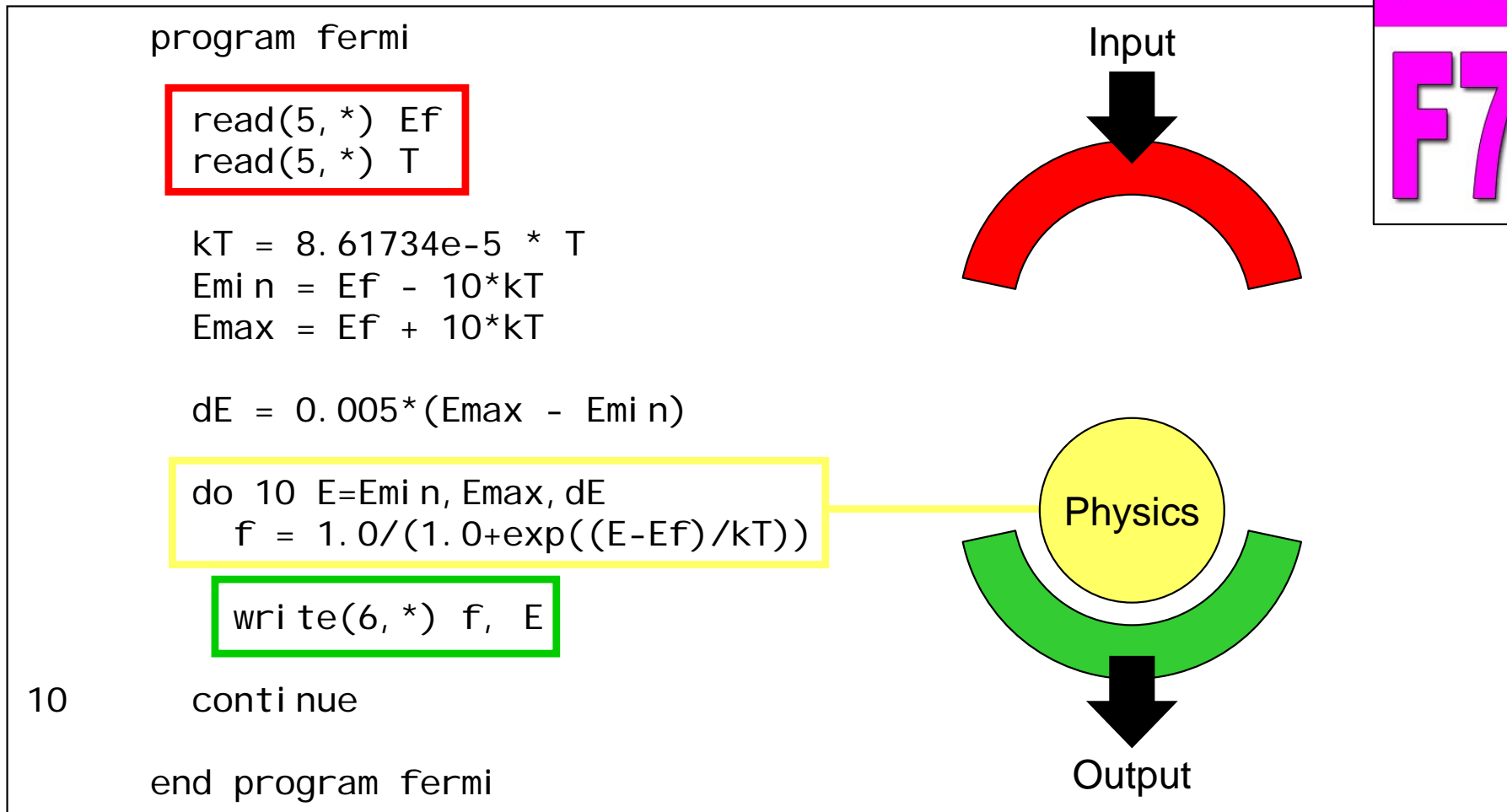
  do 10 E=Emi n, Emax, dE
    f = 1.0/(1.0+exp((E-Ef)/kT))

    write(6, *) f, E
  10  continue

end program fermi
```

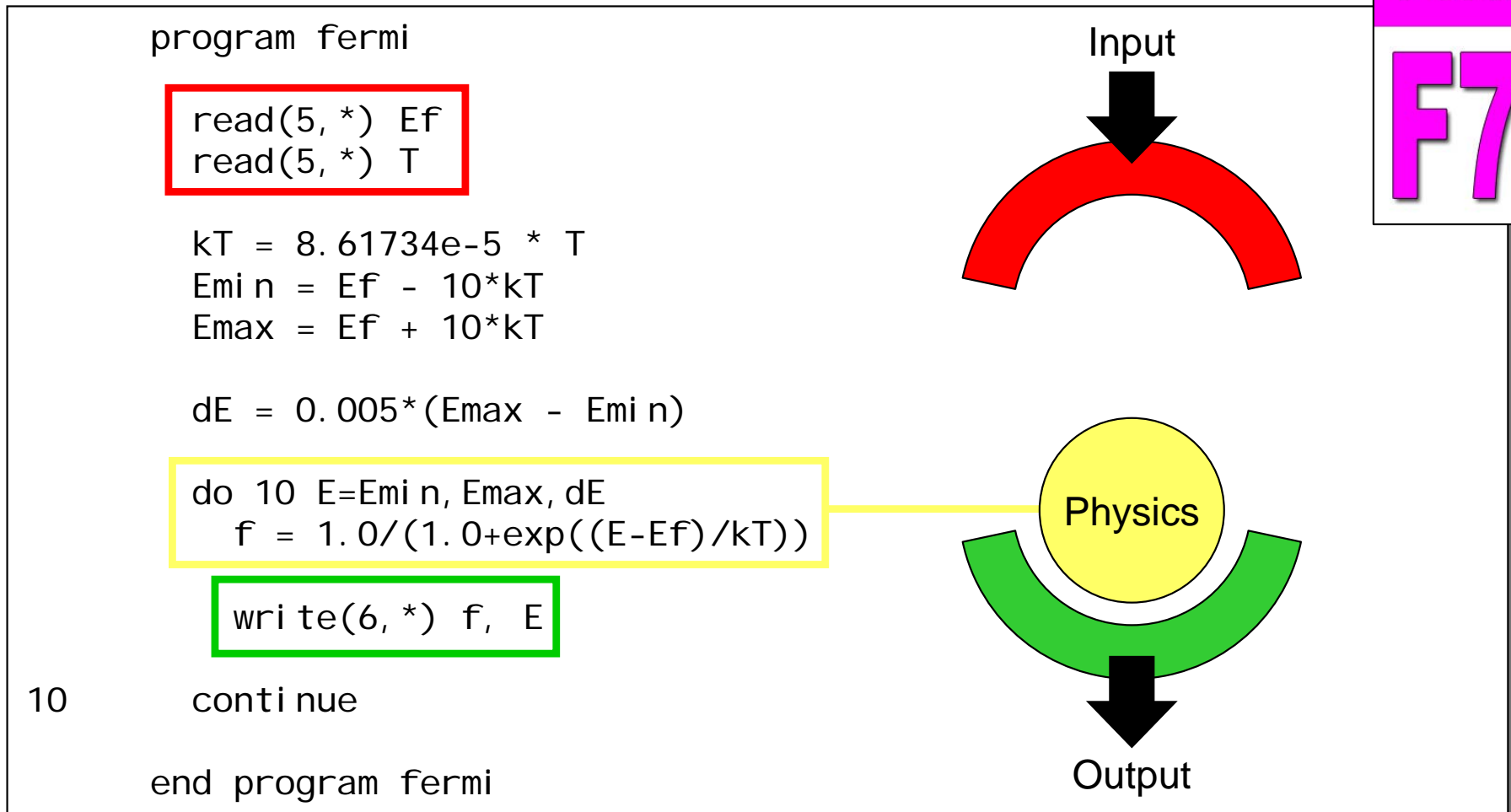
Starting Point

fermi.f (simulator written in Fortran)



Starting Point

fermi.f (simulator written in Fortran)



Rapture Inputs

fermi.f (simulator written in Fortran)

```
program fermi
```

```
call getarg(1,inFile)
driver = rp_lib(inFile)
```

```
+ call rp_lib_get(driver,  
+   "input.number(temperature).current", str)
```

```
okay = rp_units_convert_dbl(str, "K", T)
```

```
+ call rp_lib_get(driver,  
+   "input.number(Ef).current", str)
```

```
okay = rp_units_convert_dbl(str, "eV", Ef)
```

```
kT = 8.61734e-5 * T
```

```
Emi n = Ef - 10*kT
```

```
Emax = Ef + 10*kT
```

```
dE = 0.005*(Emax - Emi n)
```

...

Fortran
F77

"72F"

295.22

"2.4eV"

2.4

Rappture Outputs

fermi.f (simulator written in Fortran)

Fortran

F77

```
...
dE = 0.005*(Emax - Emin)

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

  wri te(xy, ' (E20.12, F13.9, A)' ) f, E, char(10)
  + call rp_lib_put_str(driver,
    "output.curve(f12).component.xy", xy, 1)
10 continue

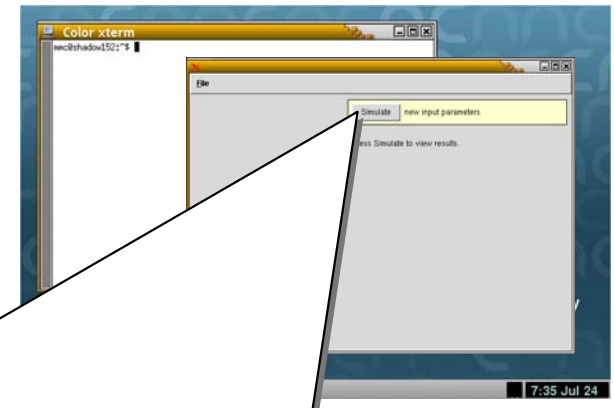
  call rp_result(driver)

end program fermi
```

" x y \n"

Compiling the Rappturized Code

From your workspace on nanoHUB...

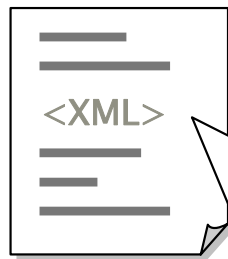


```
% cd tooldir
% ls
tool.xml fermi.f

% g77 -g fermi.f -o fermi -L/apps/rappture/current/lib -lrappture
% ls
tool.xml fermi.f fermi

% rappture
```

How does the program get invoked?



tool.xml

Replaced with directory
containing tool.xml file

Ex: /apps/yourtool /current

Replaced with name of driver file

Ex: dri ver327. xml

```
<?xml version="1.0"?>
<run>
  <tool >
    <about>Press Simulate to view results.</about>
    <command>@tool /fermi @driver </command>
  </tool >
  <i nput>
    ...
  </i nput>
  <output>
    ...
  </output>
</run>
```

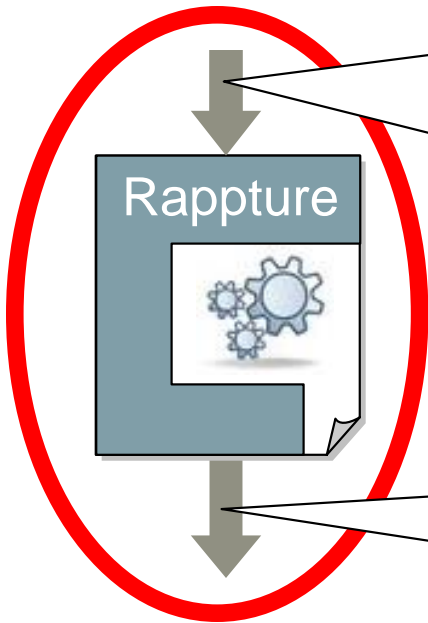
...with the driver file as the first argument

Invoke the Fortran program...

How does the program get invoked?

@tool /fermi @driver

/apps/yourtool /current/fermi driver327.xml



```

program fermi
...
call getarg(1,inFile)
driver = rp_lib(inFile)

call rp_lib_get(driver,
+ "input.number(temperature).current", str)
...

```

```

...
call rp_result(driver)

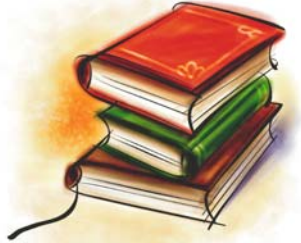
end program fermi

```

Not Much Overhead

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

Instead of this...



Documentation

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

```
wri te(6, *) Ef
```

Do this...

```
<number id="Ef">  
  <about>  
    <l a b e l >Fermi  Level </l a b e l >  
  </about>  
  <uni t s>eV</uni t s>  
  <defaul t>0eV</defaul t>  
</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](#)

