C++ Jezgro za Jupyter Notebook

Interaktivno kuckanje u jeziku C++

G. Vinterhalter M. Ranković

Matematički fakultet, Beogradski univerzitet
(Metodologija stručnog i naučnog rada)

Prezentacija radova, 2016

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije
- Svaka ćelija se može izvršiti (osvežiti) zasebno, pri čemu ćelije koda imaju izlaz.

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije
- Svaka ćelija se može izvršiti (osvežiti) zasebno, pri čemu ćelije koda imaju izlaz.
- Dokument se prikazuje renderuje kao html stranica. Zato izlaz može da predstavlja html kod.

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije
- Svaka ćelija se može izvršiti (osvežiti) zasebno, pri čemu ćelije koda imaju izlaz.
- Dokument se prikazuje renderuje kao html stranica. Zato izlaz može da predstavlja html kod.
  - Lepo formatiran kod
  - slike, grafikoni (svg, png, jpg, ...)
  - interaktivni sadržaj (java script)

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije
- Svaka ćelija se može izvršiti (osvežiti) zasebno, pri čemu ćelije koda imaju izlaz.
- Dokument se prikazuje renderuje kao html stranica. Zato izlaz može da predstavlja html kod.
  - Lepo formatiran kod
  - slike, grafikoni (svg, png, jpg, ...)
  - interaktivni sadržaj (java script)
- Ćelije podržavaju interaktivnu potržnju za korisničkim unosom.

Šta je to Jupyter Notebook ???

Jupyter Notebook je web aplikacija za kreiranje i deljenje dokumenata koji pored uobičajenog tekstualnog sadržaja sadrže i interaktivni kod.

- Dokument se sastoji iz vertikalnih ćelija:
  - ćelije teksta (markdown + latex math)
  - ćelije koda (kod koji Kernel izvršava)
  - sirove ćelije
- Svaka ćelija se može izvršiti (osvežiti) zasebno, pri čemu ćelije koda imaju izlaz.
- Dokument se prikazuje renderuje kao html stranica. Zato izlaz može da predstavlja html kod.
  - Lepo formatiran kod
  - slike, grafikoni (svg, png, jpg, ...)
  - interaktivni sadržaj (java script)
- Ćelije podržavaju interaktivnu potržnju za korisničkim unosom.

Format i konverzije

- Interno dokument se čuva kao ".json"fajl. (.ipynb format)

Format i konverzije

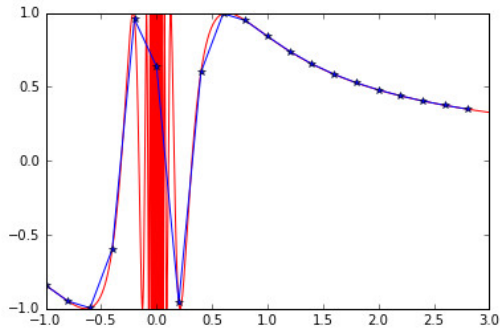
- Interno dokument se čuva kao ".json" fajl. (.ipynb format)

Sad demonstriramo matplotlib

- napravicemo tacke sa bibliotekom numpy
- plotovacemo tacke pomocu funkcije plot

```
with np.errstate(divide='ignore', invalid='ignore'):
    x = np.linspace(-1,3,10000)
    y = (np.sin(1/x))

plt.plot(x,y, 'r-')
plt.plot(x[::500], y[::500], 'b*-')
plt.show()
```



- Grafik predstavlja formulu $f(x) = sin(1/x)$ na intervalu $x \in [-1, 3]$


```
{
 "cell_type": "markdown", "metadata": {},
 "source": ["### Sad  demonstriramo  matplotlib\n",
            "- napravicemo tacke sa bibliotekom numpy\n",
            "- plotovacemo tacke pomocu funkcije plot" ] },
{
 "cell_type": "code", "execution_count": 18,
 "metadata": { "collapsed": false },
 "outputs": [
   {
     "data": {
       "image/png": "........ovde ide rezultujuca slika.....
       "text/plain": [ "<matplotlib.figure.Figure at 0x7f9f3
     },
     "metadata": {},
     "output_type": "display_data"
   }
 ],
 "source": [ "with np.errstate(divide='ignore', invalid='ign
             "    x = np.linspace(-1,3,10000)\n",
             "    y = (np.sin(1/x))\n",
             "\n",
             "plt.plot(x,y, 'r-')\n",
             "plt.plot(x[::500], y[::500], 'b*-')\n",
             "plt.show()\n"
           ]
},
{
  "cell_type": "markdown", "metadata": {},
  "source": [ "$f(x) = sin(1/x)$ na  intervalu $x\\in[-1,3]$
}
],
"metadata": { ...razne informacije...  }, "nbformat": 4, "nbf
}
```

Format i konverzije

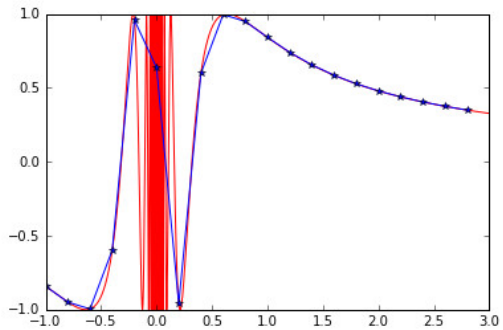
- Interno dokument se čuva kao ".json" fajl. (.ipynb format)

Sad demonstriramo matplotlib

- napravicemo tacke sa bibliotekom numpy
- plotovacemo tacke pomocu funkcije plot

```
with np.errstate(divide='ignore', invalid='ignore'):
    x = np.linspace(-1,3,10000)
    y = (np.sin(1/x))

plt.plot(x,y, 'r-')
plt.plot(x[::500], y[::500], 'b*-')
plt.show()
```



- Grafik predstavlja formulu $f(x) = sin(1/x)$ na intervalu $x \in [-1, 3]$


```
{
 "cell_type": "markdown", "metadata": {},
 "source": ["### Sad  demonstriramo  matplotlib\n",
            "- napravicemo tacke sa bibliotekom numpy\n",
            "- plotovacemo tacke pomocu funkcije plot" ] },
{
 "cell_type": "code", "execution_count": 18,
 "metadata": { "collapsed": false },
 "outputs": [
   {
     "data": {
       "image/png": "........ovde ide rezultujuca slika.....
       "text/plain": [ "<matplotlib.figure.Figure at 0x7f9f3
     },
     "metadata": {},
     "output_type": "display_data"
   }
 ],
 "source": [ "with np.errstate(divide='ignore', invalid='ign
             "    x = np.linspace(-1,3,10000)\n",
             "    y = (np.sin(1/x))\n",
             "\n",
             "plt.plot(x,y, 'r-')\n",
             "plt.plot(x[::500], y[::500], 'b*-')\n",
             "plt.show()\n"
           ]
},
{
  "cell_type": "markdown", "metadata": {},
  "source": [ "$f(x) = sin(1/x)$ na  intervalu $x\\in[-1,3]$
}
],
"metadata": { ...razne informacije...  }, "nbformat": 4, "nbf
}
```


- Čuvaju se i izlazi ćelija

Format i konverzije

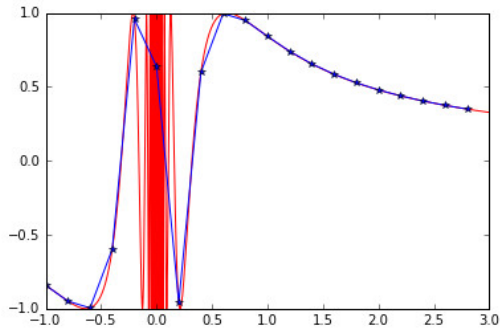
- Interno dokument se čuva kao ".json" fajl. (.ipynb format)

Sad demonstriramo matplotlib

- napravicemo tacke sa bibliotekom numpy
- plotovacemo tacke pomocu funkcije plot

```
with np.errstate(divide='ignore', invalid='ignore'):
    x = np.linspace(-1,3,10000)
    y = (np.sin(1/x))

plt.plot(x,y, 'r-')
plt.plot(x[::500], y[::500], 'b*-')
plt.show()
```



- Grafik predstavlja formulu $f(x) = sin(1/x)$ na intervalu $x \in [-1, 3]$


```
{
 "cell_type": "markdown", "metadata": {},
 "source": ["### Sad  demonstriramo  matplotlib\n",
            "- napravicemo tacke sa bibliotekom numpy\n",
            "- plotovacemo tacke pomocu funkcije plot" ] },
{
 "cell_type": "code", "execution_count": 18,
 "metadata": { "collapsed": false },
 "outputs": [
   {
     "data": {
       "image/png": "........ovde ide rezultujuca slika.....
       "text/plain": [ "<matplotlib.figure.Figure at 0x7f9f3
     },
     "metadata": {},
     "output_type": "display_data"
   }
 ],
 "source": [ "with np.errstate(divide='ignore', invalid='ign
             "    x = np.linspace(-1,3,10000)\n",
             "    y = (np.sin(1/x))\n",
             "\n",
             "plt.plot(x,y, 'r-')\n",
             "plt.plot(x[::500], y[::500], 'b*-')\n",
             "plt.show()\n"
           ]
},
{
  "cell_type": "markdown", "metadata": {},
  "source": [ "$f(x) = sin(1/x)$ na  intervalu $x\\in[-1,3]$
}
],
"metadata": { ...razne informacije...  }, "nbformat": 4, "nbf
}
```

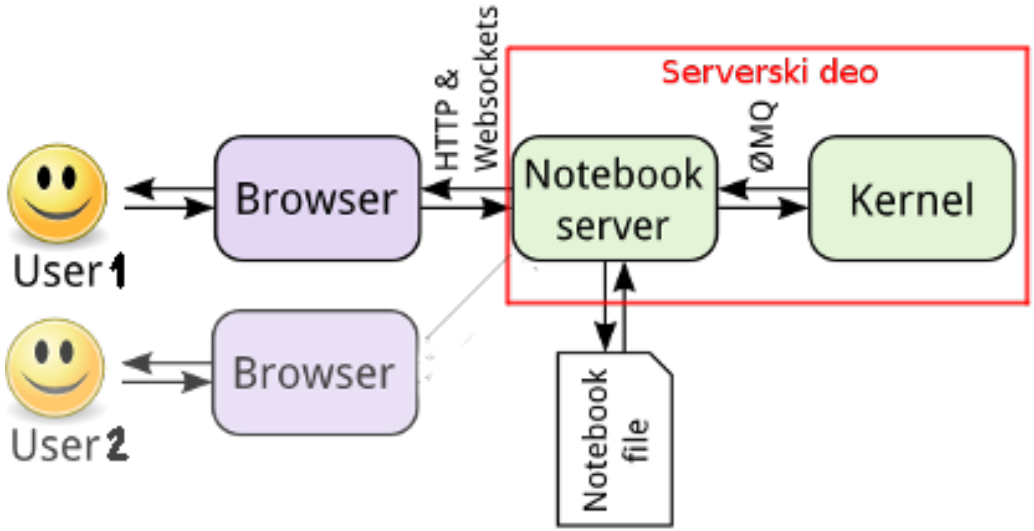

- Čuvaju se i izlazi ćelija
- Dokument se može eksportovati : pdf (preko latexa), html, html kao prezentacija, markdown, izvrni kod)

Arhitektura

- arhitektura je klijent server.

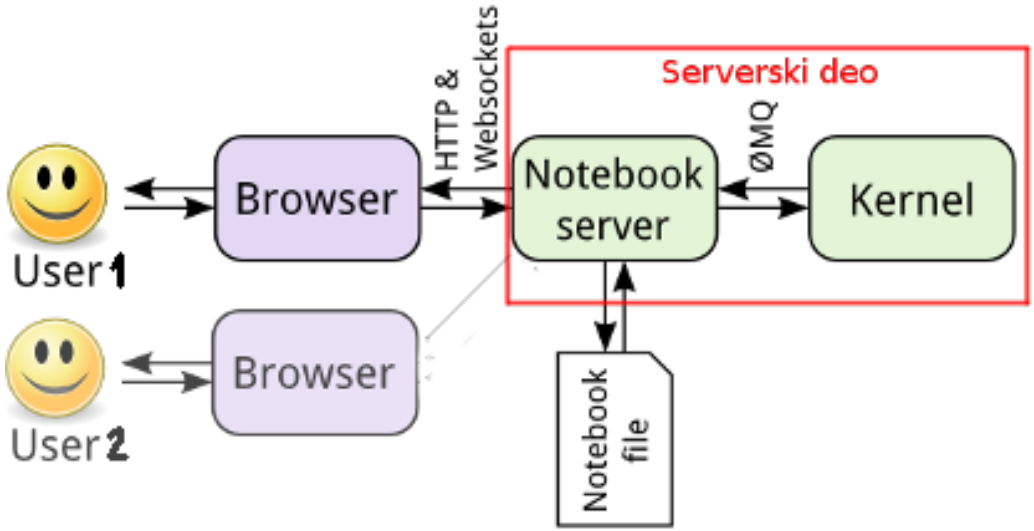
Arhitektura

- arhitektura je klijent server.



Arhitektura

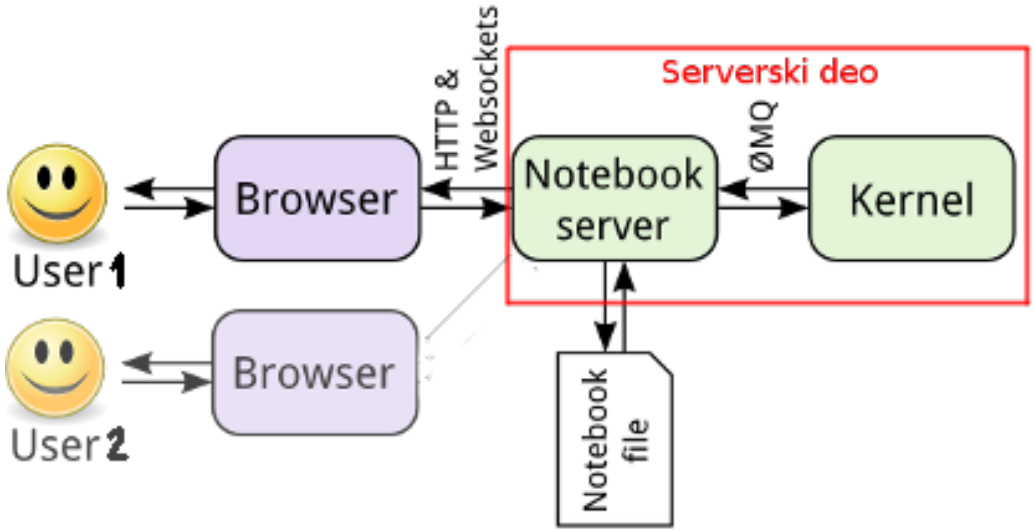
- arhitektura je klijent server.



- Notebook server i Kernel (Jezgro) čine serverski deo aplikacije

Arhitektura

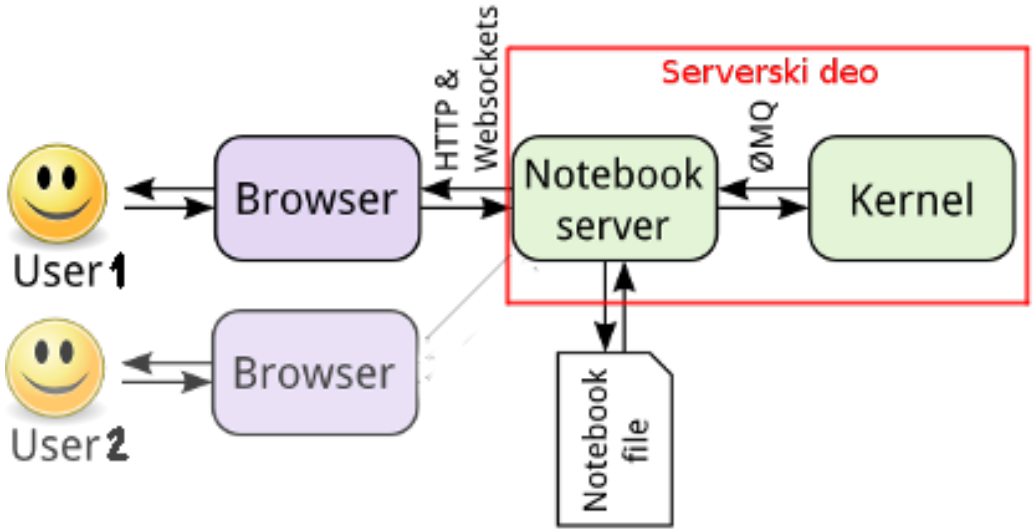
- arhitektura je klijent server.



- Notebook server i Kernel (Jezgro) čine serverski deo aplikacije
- Jezgro izvršava sadržaj ćelija koda. Jezgro obično podržava samo jedan jezik (Python, Haskell, Jullia, R, ...)

Arhitektura

- arhitektura je klijent server.



- Notebook server i Kernel (Jezgro) čine serverski deo aplikacije
- Jezgro izvršava sadržaj ćelija koda. Jezgro obično podržava samo jedan jezik (Python, Haskell, Jullia, R, ...)
- Mi želimo Jezgro koje će izvršavati C++

Dužnost Jezgra

- jezgro komunicira sa Notebook Server applikacijom i mora da ispoštuje određen format komunikacije (propisane poruke)

Dužnost Jezgra

- jezgro komunicira sa Notebook Server applikacijom i mora da ispoštuje određen format komunikacije (propisane poruke)
- Struktura poruke:

```
{
  'header' :{
    'msg' : uuid,
    'username' : str,
    'session' : uuid,
    'msg_type' : str,
    'version' : 5.0
  }

  'parent_header' : dict,
  'metadata' : dict,
  'content' : dict
}
```

Bitne poruke

- shell ROUTER/DEALER soket
  - execute_request / reply
  - introspect_request / reply
  - completion_request / reply
  - History_request / reply
  - KernelInfo_request / reply
  - ...
- PUB/SUB soket
  - stream (stdout, stderr, ...)
  - display (html, svg, png, ..)
  - clearOutput
  - ...
- stdin ROUTER/DEALER soket
  - input_request
- Dozvoljeno je dodavanje drugih vrsta poruka...

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:
  - Kompjaliramo: `g++ -shared -fpic`

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:
  - Kompjaliramo: `g++ -shared -fpic`
  - U pythonu učitavamo .so preko CDLL f-je. (dlopen u pozadini) flagovi: `RTLD_GLOBAL` `RTLD_DEEPBIND`

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:
  - Kompjaliramo: `g++ -shared -fpic`
  - U pythonu učitavamo .so preko CDLL f-je. (dlopen u pozadini) flagovi: `RTLD_GLOBAL` `RTLD_DEEPBIND`
  - pozivamo funkciju void __run__() koju smo prethodno ubacili u izvorni kod

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:
  - Kompjaliramo: `g++ -shared -fpic`
  - U pythonu učitavamo .so preko CDLL f-je. (dlopen u pozadini) flagovi: `RTLD_GLOBAL` `RTLD_DEEPBIND`
  - pozivamo funkciju void __run__() koju smo prethodno ubacili u izvorni kod
- `void __run__()` se automatski generiše od strne C++ Jezgra i sadrži izraze koji nisu deklaracije/definicije. Koristmo specijalnu sintaksu koja je inače standardna za jupyter notebook jezgra

Ideja za C++

- Šta želimo?
  - Korisnik ima osećaj kao da radi sa skript jezikom
  - Tačnije, promene u jednoj ćeliju utiču na ostale
- Koristimo dinamičko učitavanje. Ćeliju izvršavamo tako što:
  - Kompjaliramo: `g++ -shared -fpic`
  - U pythonu učitavamo .so preko CDLL f-je. (dlopen u pozadini) flagovi: `RTLD_GLOBAL` `RTLD_DEEPBIND`
  - pozivamo funkciju void __run__() koju smo prethodno ubacili u izvorni kod
- `void __run__()` se automatski generiše od strne C++ Jezgra i sadrži izraze koji nisu deklaracije/definicije. Koristmo specijalnu sintaksu koja je inače standardna za jupyter notebook jezgra
  - %r statment; line magic, statment je trenutno jednoliniski, U duhu c++ to bi trebalo prepraviti.
  - %%r statment; cell magic, cela ćelija se smešta u `void __run__()`

Primer 1

1.
```
int a = 10;
%r cout << a;
void f() {cout << a+1 << endl;}
```
```
int a = 10;
void f() {cout << a+1 << endl;}
void__run__(){ cout << a; }
```
```
result:
10
```

2.
```
%rr
a = 25;
cout << a << endl;
f();
```
```
extern int a;
void f();
void__run__(){
  a = 25;
  cout << a << endl;
  f();
}
```
```
result:
25
26
```

3.
```
float a = 3.14;
%r cout << a << endl;
%r f();
```
```
void f();
float a = 3.14;
void__run__(){
  cout << a << endl;
  f();
}
```

Primer 1

1.
```
int a = 10;
%r cout << a;
void f() {cout << a+1 << endl;}
```
```
int a = 10;
void f() {cout << a+1 << endl;}
void__run__(){ cout << a; }
```
```
result:
10
```

2.
```
%rr
a = 25;
cout << a << endl;
f();
```
```
extern int a;
void f();
void__run__(){
  a = 25;
  cout << a << endl;
  f();
}
```
```
result:
25
26
```

3.
```
float a = 3.14;
%r cout << a << endl;
%r f();
```
```
void f();
float a = 3.14;
void__run__(){
  cout << a << endl;
  f();
}
```
```
result:
3.14
26
```

Primer 1

1.
```
int a = 10;
%r cout << a;
void f() {cout << a+1 << endl;}
```
```
int a = 10;
void f() {cout << a+1 << endl;}
void__run__(){ cout << a; }
```
```
result:
10
```

2.
```
%rr
a = 25;
cout << a << endl;
f();
```
```
extern int a;
void f();
void__run__(){
  a = 25;
  cout << a << endl;
  f();
}
```
```
result:
25
26
```

3.
```
float a = 3.14;
%r cout << a << endl;
%r f();
```
```
void f();
float a = 3.14;
void__run__(){
  cout << a << endl;
  f();
}
```
```
result:
3.14
26
```

4.
```
%rr
cout << a << endl;
f();
```
```
extern float a;
void f();
void__run__(){
  cout << a << endl;
  f();
}
```

Primer 1

1.
```
int a = 10;
%r cout << a;
void f() {cout << a+1 << endl;}
```
```
int a = 10;
void f() {cout << a+1 << endl;}
void__run__(){ cout << a; }
```
```
result:
10
```

2.
```
%rr
a = 25;
cout << a << endl;
f();
```
```
extern int a;
void f();
void__run__(){
  a = 25;
  cout << a << endl;
  f();
}
```
```
result:
25
26
```

3.
```
float a = 3.14;
%r cout << a << endl;
%r f();
```
```
void f();
float a = 3.14;
void__run__(){
  cout << a << endl;
  f();
}
```
```
result:
3.14
26
```

4.
```
%rr
cout << a << endl;
f();
```
```
extern float a;
void f();
void__run__(){
  cout << a << endl;
  f();
}
```
```
result:
3.50325e-44
26
```

Rešenje

- Funkcije menjamo pokazivačima na funkcije.

Rešenje

- Funkcije menjamo pokazivačima na funkcije.
  - funkciju `void f() {...}` zamenjujemo pokazivačem tj.
    ```
    void f_1() {...}
    void (*f)() = f_1;
    ```

Rešenje

- Funkcije menjamo pokazivačima na funkcije.
  - funkciju `void f() {...}` zamenjujemo pokazivačem tj.
    ```
    void f_1() {...}
    void (*f)() = f_1;
    ```
  - Promena ponašanja funkcije (bez menjana potpisa) se svodi na promenu vrednosti pokazivača.

Rešenje

- Funkcije menjamo pokazivačima na funkcije.
  - funkciju `void f() {...}` zamenjujemo pokazivačem tj.
    ```
    void f_1() {...}
    void (*f)() = f_1;
    ```
  - Promena ponašanja funkcije (bez menjana potpisa) se svodi na promenu vrednosti pokazivača.
- Redefinisanje tipova ili potpisa.
  - Pravo ime je sakrivamo od korisnika.

Rešenje

- Funkcije menjamo pokazivačima na funkcije.
  - funkciju `void f() {...}` zamenjujemo pokazivačem tj.
    ```
    void f_1() {...}
    void (*f)() = f_1;
    ```
  - Promena ponašanja funkcije (bez menjana potpisa) se svodi na promenu vrednosti pokazivača.
- Redefinisanje tipova ili potpisa.
  - Pravo ime je sakrivamo od korisnika.
  - ako dođe do redefinicije menja se ime svim simbolima. Za sad jednostavno inkrementujem broj na kraju pravog imena.

Rešenje

- Funkcije menjamo pokazivačima na funkcije.
  - funkciju `void f() {...}` zamenjujemo pokazivačem tj.
    ```
    void f_1() {...}
    void (*f)() = f_1;
    ```
  - Promena ponašanja funkcije (bez menjana potpisa) se svodi na promenu vrednosti pokazivača.
- Redefinisanje tipova ili potpisa.
  - Pravo ime je sakrivamo od korisnika.
  - ako dođe do redefinicije menja se ime svim simbolima. Za sad jednostavno inkrementujem broj na kraju pravog imena.
  - Ime je promenjeno samo u toj ćeliji. U drugim ćelijma se idalje referiše staro ime jer nisu osvežene. Osvežavanje može da prozivede grešku jer se potpis više ne poklapa ili tip više ne podržava traženo ponašanje.

libclang

- libclang je C biblioteka koju koristimo za parsiranje C++ koda.
- Biblioteka je napravljena da korisnik može da obilazi apstraktno stablo sintakse (eng. AST) krećući se kroz čvorove tako što registruje handler funkcije koje sam definiše.
- Osnovni element stabla je CXCursor i sadrži informacije:
  - tipu:
    - Razne vrste deklaracije
    - Razne vrste izraza
    - Direktive
    - i još mnogo toga ...
  - datoteka u kojoj se nalazi (izbegavanje include fajlova)
  - Lokacija u izvornom kodu (pamtimo za modifikaciju koda)
  - tip podataka (potipis, povratna vrednost) (pamtimo)
  - ime simbola (pamtimo)
  - kvalifikacioni atributi (pamtimo)
  - referenca na definiciju (prepoznavanje lokalnih promenljivih)

libclang

Auto ?

- auto je predstavljen Kursorom 'unresolved'
- Međutim već naredni Cursor sadrži potreban tip
- U slučaju const auto &, prvi kursor je &

```
int * * a = nullptr;
auto b = 3.14;
auto f(int  c) {return nullptr};
  ---------------------------------------
TranslationUnitDecl <<invalid sloc>> <invalid sloc>
|-VarDecl <t1.cpp:2:1, col:13> col:9 a 'int **' cinit
| `-ImplicitCastExpr <col:13> 'int **' <NullToPointer>
|   `-CXXNullPtrLiteralExpr <col:13> 'nullptr_t'
|-VarDecl <line:4:1, col:10> col:6 b 'double':'double' cinit
| `-FloatingLiteral 0x25b0c58 <col:10> 'double' 3.140000e+00
|-FunctionDecl <line:5:1, col:31> col:6 f 'void* (int)'
| |-ParmVarDecl <col:8, col:13> col:13 c 'int'
| `-CompoundStmt <col:16, col:31>
|   `-ReturnStmt <col:17, col:24>
|     `-CXXNullPtrLiteralExpr <col:24> 'nullptr_t'
`-EmptyDecl <col:32> col:32
```

Kraj

- Nedostaci:
  - Trenutno ne podržavamo klase/strukture, unije i šablone.
  - Segmentation Fault ubija jezgro.
  - Interaktivni unos korisnika fali.
  - Nisu sve poruke implementirane.
- Literatura:
  - Ipython Developer documentation
    `https://ipython.org/ipython-doc/3/development/`
  - Ipython magic documentation
    `https://ipython.org/ipython-doc/3/interactive/magics.html`
  - dlopen man page `http://linux.die.net/man/3/dlopen`
  - Jupyter project `http://jupyter.org/`
  - Clang documentation `http://clang.llvm.org/`
  - Linkers and Loaders, John R. Levine