Plain Object

```
#include <iostream>
class A
{
public:
  A() : w(1), v(2) { q = 80; ch1 = 'U'; ch2 = 'V'; ch3 = 'a';}
  ~A() {}

  void foo() {
    std::cout << "A::foo " << this << " " << w << std::endl;
  }

  void quux() {};

  void quuux(int a) {
    std::cout << "A::quuux " << this << " " << w << " " << a << std::endl;
  }

private:
  int w, v;
  char ch1, ch2, ch3;
  double q;
};
int main(int argc, char **argv)
{
  A *a = new A();
// private members are inaccessible outside the class...
  // std::cout << a->w << " " << a->v << std::endl;
// ...but they're still there if you know where to look
  // beginning of an instance of A looks the same an array of two ints
  std::cout << ((int *)a)[0] << " " << ((int *)a)[1] << std::endl;
  // for other fields we do the pointer arithmetic in bytes an coerce the
  // compiler into reading the right type from the resulting address
  std::cout << *((double *)((char *)a + 16)) << std::endl;

  delete a;
}
```

fields consecutively in memory
in declaration order

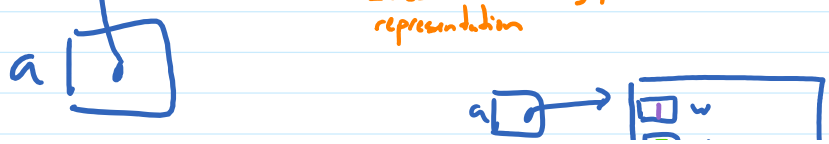
platform is little-endian
so ints have least significant
byte first
(and we're displaying as if this is an int)

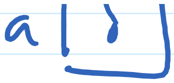
platform-dependent
padding

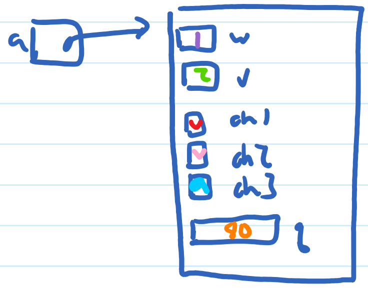
```
/*
(gdb) x/6x a
0x613e70:   0x00000001    0x00000002    0x00615655    0x00000000
0x613e80:   0x00000000    0x40540000
 */
```

w v ch2 ch3 ch1 q

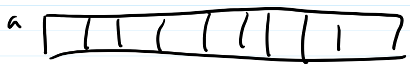
IEEE 64-bit floating point
representation



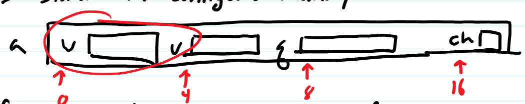

a


a
1
w
2
v
ch1
ch2
ch3
90
i

addr of a[i] is addr of a + i * (size of individual element)



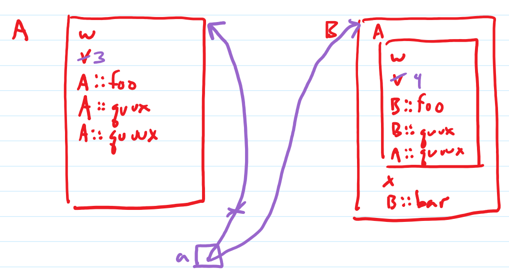
objects : fields stored in contiguous memory



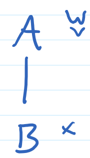
addr of a.whatever is addr of a + some fixed offset

arithmetic in bytes (not C code)

&(a.w) = &a + 0
&(a.v) = &a + 4



The A part of a B must be laid out like any other instance of A



A *a = new A();
a->v = 3;
a = new B();
a->v = 4;

same process for finding v: add sizeof(w) to a

A *b = new B();

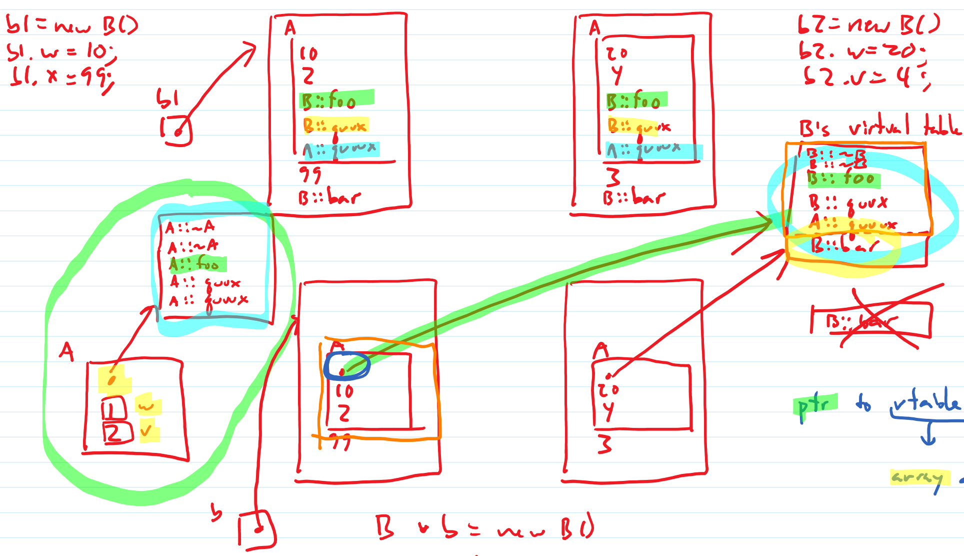
b->x illegal - b declared type is A * x not declared in A

b->foo() legal - foo is declared in A executes B::foo since b points to obj created w/ new B (dynamic/run-time type is B)

b1 = new B()
b1.w = 10;
b1.x = 99;

b2 = new B()
b2.w = 20;
b2.v = 4;

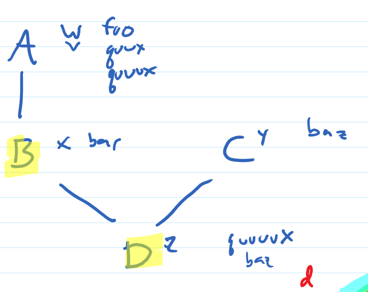
B's virtual table (vtable)



ptr to vtable = ptr to ptr to ptr to method

array of ptrs to methods

B *b = new B()
or
A *b = new B()

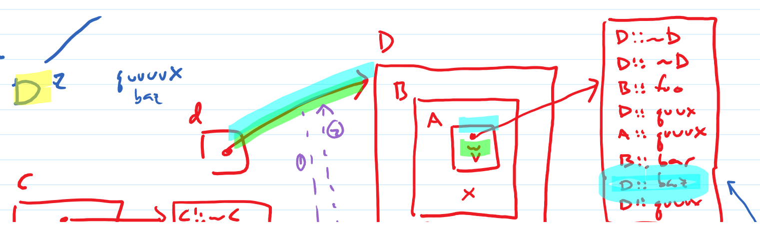


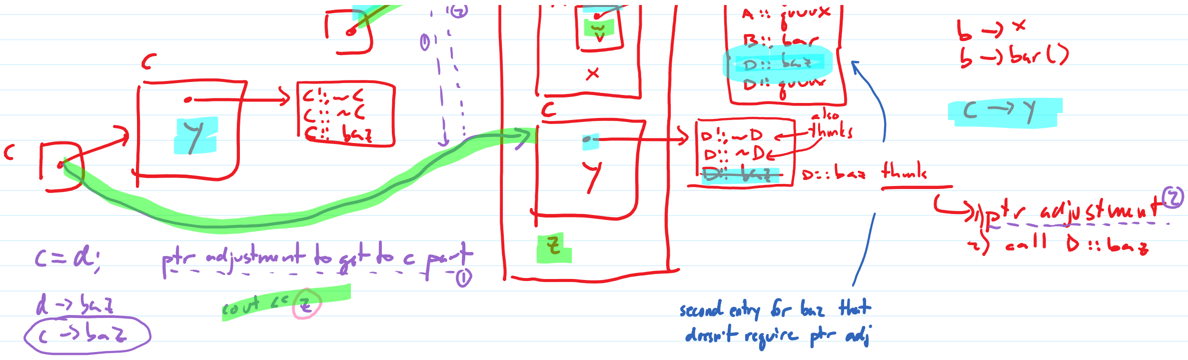
D *d = new D()

B * b = d;
A * a = d;
C * c = d;

a->v
a->foo()

b->x
b->bar()

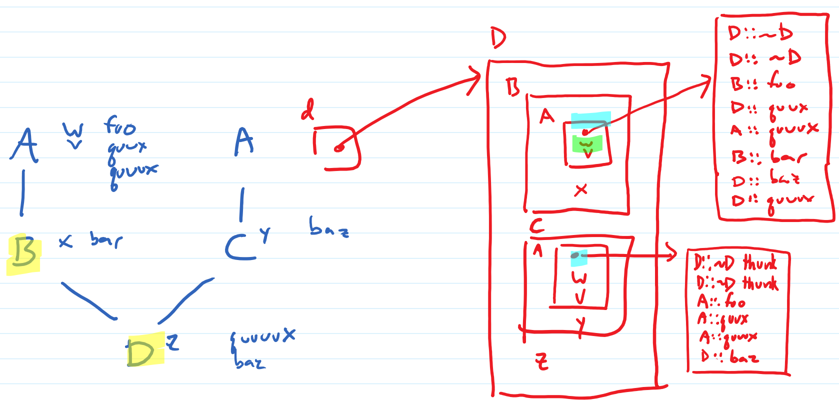


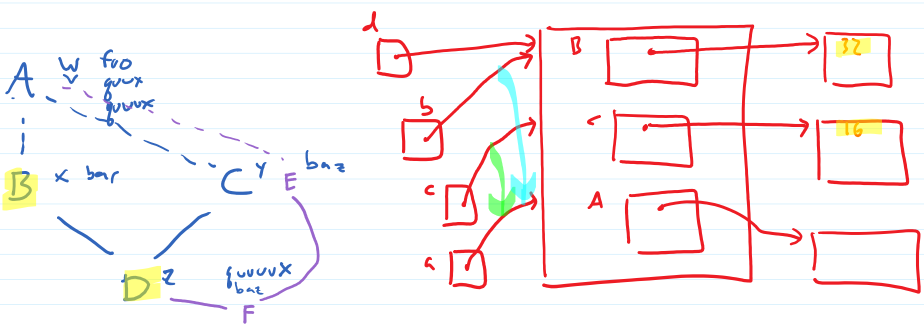


c = d;   ptr adjustment to get to c part

d -> baz   cout << z

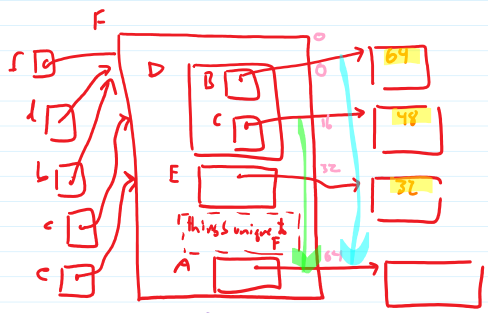
c -> baz





D *d = new D();   D *d = new E ;

A *a = d;
B *b = d;
C *c = d;



a = c;   adj ptr by 16
or
a = b;   adj ptr by 32

different adjustments based on dynamic (run-time) type of object

F *f = new F();
D *d = f;
C *c = f;
B *b = f;

different adjustments based
on dynamic (run-time) type of object

store necessary adjustment
in vtable

to access member of virtual base:
get offset of base part from vtable
add offset to ptr to object
proceed as usual
(for field, add field's fixed offset,
for virtual fxn, follow ptr to
vtable and get entry at fixed location)

```
F *f = new F();
D *d = f;
C *c = f;
B *b = f;
E *e = f
```

A * a = f
or
a = c
or
a = b
or
a = d
or
a = e

adjust by 48

adjust by 64

virtual

```
class A
{
public:
  A() : w(1), v(2) {}
  virtual ~A() {}

  virtual void foo() {
    std::cout << "A::foo " << this << " " << w << std::endl;
  }

  virtual void quux() = 0;

  virtual void quuux(int a) {
    std::cout << "A::quuux " << this << " " << w << " " << a << std::endl;
  }

private:
  int w, v;
};

class B : public virtual A
{
public:
  B() { x = 3; }
  ~B() {}

  void foo() {
    std::cout << "B::foo " << this << " " << x << std::endl;
  }

  virtual void bar() {
    std::cout << "B::bar "  << this << " " << x << std::endl;
  }

  void quux() {
    std::cout << "B::quux "  << this << " " << x << std::endl;
  }

private:
  int x;
};


class C : public virtual A
{
public:
  C() { y = 4; }

  virtual void baz() {
    std::cout << "C::baz "  << this << " " << y << std::endl;
    foo();
    quux();
  }
private:
  int y;
};

class D : public B, public C
{
public:
  D() { z = 5; }
```

```
    virtual void quux() {
      std::cout << "D::quux "  << this << " " << z << std::endl;
    }

  private:
    int z;
  };

  int main(int argc, char **argv)
  {
    D *d = new D();

    // no pointer adjustment
    d->bar();
    d->quux();

    // pointer adjustment
    d->baz();
    d->quuux(1);

    delete d;
  }
```

```
An instance of D
(gdb) x/12xw d
0x614ea0:    B 0x00401308 vtable  0x00000000    0x00000003 x    0x00000000

0x614eb0:    C 0x00401348 vtable  0x00000000    0x00000004 y    0x00000005 z

0x614ec0:    A 0x00401390 vtable  0x00000000    0x00000001 w    0x00000002 v
Its B vtable
(gdb) x/8xg 0x4012f0
                               B->A adjustment      B->whole object adj
0x4012f0 <_ZTV1D>:             0x0000000000000020   0x0000000000000000
0x401300 <_ZTV1D+16>:  RTTI    0x00000000004015f0   0x0000000000400d30 B::foo
0x401310 <_ZTV1D+32>:  B::bar  0x0000000000400d98   0x00000000004010fe D::quux
0x401320 <_ZTV1D+48>:  D::~D   0x0000000000401166   0x00000000004011ec D::~D
Its C vtable
(gdb) x/8xg 0x401330
                               C->A                 C->whole
0x401330 <_ZTV1D+64>:          0x0000000000000010   0xfffffffffffffff0
0x401340 <_ZTV1D+80>:  RTTI    0x00000000004015f0   0x0000000000400eaa C::baz
0x401350 <_ZTV1D+96>:          0x00000000004011da D::~D thunk   0x0000000000401217 D::~D thunk
0x401360 <_ZTV1D+112>:         0x0000000000000000   0xffffffffffffffe0 (foo/quux not overridden in C; look up as member of A)
Its A vtable
(gdb) x/8xg 0x401378
                                                    A->whole
0x401378 <_ZTV1D+136>:         0xffffffffffffffe0   0xffffffffffffffe0
0x401388 <_ZTV1D+152>: RTTI    0x00000000004015f0   0x00000000004011e0 D::~D thunk
0x401398 <_ZTV1D+168>:         0x000000000040121d D::~D thunk   0x0000000000400d8e B::foo thunk
0x4013a8 <_ZTV1D+184>:         0x000000000040115c D::quux thunk   0x0000000000400c20 A::quuux
(gdb) x/1i  0x00000000004015f0
   0x4015f0 <_ZTI1D>:   test   $0x1d,%al
(gdb) x/1i  0x0000000000400d30
   0x400d30 <B::foo()>: push   %rbp
(gdb) x/1i  0x0000000000400d98
   0x400d98 <B::bar()>: push   %rbp
(gdb) x/1i  0x00000000004010fe
   0x4010fe <D::quux()>:        push   %rbp
```

```
(gdb) x/1i  0x0000000000401166
   0x401166 <D::~D()>:  push   %rbp
(gdb) x/1i  0x00000000004011ec
   0x4011ec <D::~D()>:  push   %rbp
(gdb) x/1i  0x0000000000400eaa
   0x400eaa <C::baz()>: push   %rbp
(gdb) x/2i  0x00000000004011da
   0x4011da <_ZThn16_N1DD1Ev>:   sub    $0x10,%rdi
   0x4011de <_ZThn16_N1DD1Ev+4>:        jmp    0x401166 <D::~D()>
(gdb) x/2i  0x0000000000401217
   0x401217 <_ZThn16_N1DD0Ev>:   sub    $0x10,%rdi
   0x40121b <_ZThn16_N1DD0Ev+4>:        jmp    0x4011ec <D::~D()>
(gdb) x/3i  0x00000000004011e0
   0x4011e0 <_ZTv0_n24_N1DD1Ev>:        mov    (%rdi),%r10
   0x4011e3 <_ZTv0_n24_N1DD1Ev+3>:      add    -0x18(%r10),%rdi
   0x4011e7 <_ZTv0_n24_N1DD1Ev+7>:      jmpq   0x401166 <D::~D()>
(gdb) x/3i  0x000000000040121d
   0x40121d <_ZTv0_n24_N1DD0Ev>:        mov    (%rdi),%r10
   0x401220 <_ZTv0_n24_N1DD0Ev+3>:      add    -0x18(%r10),%rdi
   0x401224 <_ZTv0_n24_N1DD0Ev+7>:      jmp    0x4011ec <D::~D()>
(gdb) x/3i  0x0000000000400d8e
   0x400d8e <_ZTv0_n32_N1B3fooEv>:      mov    (%rdi),%r10
   0x400d91 <_ZTv0_n32_N1B3fooEv+3>:    add    -0x20(%r10),%rdi
   0x400d95 <_ZTv0_n32_N1B3fooEv+7>:    jmp    0x400d30 <B::foo()>
(gdb) x/3i  0x000000000040115c
   0x40115c <_ZTv0_n40_N1D4quuxEv>:     mov    (%rdi),%r10
   0x40115f <_ZTv0_n40_N1D4quuxEv+3>:   add    -0x28(%r10),%rdi
   0x401163 <_ZTv0_n40_N1D4quuxEv+7>:   jmp    0x4010fe <D::quux()>
(gdb) x/1i  0x0000000000400c20
   0x400c20 <A::quuux(int)>:    push   %rbp
(gdb) x/80i main
...
   0x400a48 <main(int, char**)+66>:     mov    -0x20(%rbp),%rdx
   0x400a4c <main(int, char**)+70>:     mov    -0x20(%rbp),%rax
   0x400a50 <main(int, char**)+74>:     mov    (%rax),%rax          d -> bar()
   0x400a53 <main(int, char**)+77>:     add    $0x8,%rax
   0x400a57 <main(int, char**)+81>:     mov    (%rax),%rax
   0x400a5a <main(int, char**)+84>:     mov    %rdx,%rdi
   0x400a5d <main(int, char**)+87>:     callq  *%rax
   0x400a5f <main(int, char**)+89>:     mov    -0x20(%rbp),%rax
   0x400a63 <main(int, char**)+93>:     mov    (%rax),%rax
   0x400a66 <main(int, char**)+96>:     add    $0x10,%rax
   0x400a6a <main(int, char**)+100>:    mov    (%rax),%rax          d -> quux()
   0x400a6d <main(int, char**)+103>:    mov    -0x20(%rbp),%rdx
   0x400a71 <main(int, char**)+107>:    mov    %rdx,%rdi
   0x400a74 <main(int, char**)+110>:    callq  *%rax
   0x400a76 <main(int, char**)+112>:    mov    -0x20(%rbp),%rax
   0x400a7a <main(int, char**)+116>:    lea    0x10(%rax),%rdx      <- add 16 to adjust D->C
   0x400a7e <main(int, char**)+120>:    mov    -0x20(%rbp),%rax
   0x400a82 <main(int, char**)+124>:    mov    0x10(%rax),%rax      d -> baz()
   0x400a86 <main(int, char**)+128>:    mov    (%rax),%rax
   0x400a89 <main(int, char**)+131>:    mov    %rdx,%rdi
   0x400a8c <main(int, char**)+134>:    callq  *%rax
   0x400a8e <main(int, char**)+136>:    mov    -0x20(%rbp),%rax     get d
   0x400a92 <main(int, char**)+140>:    mov    (%rax),%rax          follow vtable ptr
   0x400a95 <main(int, char**)+143>:    sub    $0x18,%rax           addr of offset to A
   0x400a99 <main(int, char**)+147>:    mov    (%rax),%rax          get offset of A      ptr adjust
   0x400a9c <main(int, char**)+150>:    mov    %rax,%rdx            save                 D->A via entry in vtable
   0x400a9f <main(int, char**)+153>:    mov    -0x20(%rbp),%rax     get d
   0x400aa3 <main(int, char**)+157>:    add    %rax,%rdx            add saved offset to d
   0x400aa6 <main(int, char**)+160>:    mov    -0x20(%rbp),%rax
   0x400aaa <main(int, char**)+164>:    mov    (%rax),%rax          d -> quuux(1)
   0x400aad <main(int, char**)+167>:    sub    $0x18,%rax
   0x400ab1 <main(int, char**)+171>:    mov    (%rax),%rax
   0x400ab4 <main(int, char**)+174>:    mov    %rax,%rcx
   0x400ab7 <main(int, char**)+177>:    mov    -0x20(%rbp),%rax
```

```
   0x400aad <main(int, char**)+167>:     sub    $0x18,%rax
   0x400ab1 <main(int, char**)+171>:     mov    (%rax),%rax
   0x400ab4 <main(int, char**)+174>:     mov    %rax,%rcx
   0x400ab7 <main(int, char**)+177>:     mov    -0x20(%rbp),%rax
   0x400abb <main(int, char**)+181>:     add    %rcx,%rax
   0x400abe <main(int, char**)+184>:     mov    (%rax),%rax
   0x400ac1 <main(int, char**)+187>:     add    $0x20,%rax
   0x400ac5 <main(int, char**)+191>:     mov    (%rax),%rax
   0x400ac8 <main(int, char**)+194>:     mov    $0x1,%esi
   0x400acd <main(int, char**)+199>:     mov    %rdx,%rdi
   0x400ad0 <main(int, char**)+202>:     callq  *%rax
...
*/
```