CS480

Ch 7
Handling Data

March 30, 2009
Chapter 7
Do as much as we can, come back to it later

- Handling data at runtime
  - static: pages 395-429
  - dynamic: pages 440-446
  - how does garbage collection work?
  - what is reference counting?
  - dynamic vs static binding

- Handling data at compile time (Symbol Table)
  - pages 429-440
Process Layout
Binding

• Static Binding

• Dynamic Binding
class animal {
    private:
        int weight;
    public:
        animal() {}
        animal(int w): weight(w) {};
        virtual void makeSound() = 0 ;
        int getWeight() {return weight;};
        virtual int getWeight2() =0
        virtual void hi() { printf("HI");};
    };

class cat : public animal
{
    private:
        int weight;
    public:
        cat(int w) : weight(w) {};
        void makeSound()
        {
            weight > 20 ? roar() : meow();
        }
        int getWeight2() {return weight;};
        void bye() { printf("bye"); };
};

int main()
{
    animal *pAn = new cat(93);
    pAn->makeSound();
    pAn->showWeight();
    pAn->showWeight2();
    pAn->hi();
    pAn->bye();     // ????
}
Vtable for animal
animal::_ZTV6animal: 5u entries
0     (int (*)(...))0
8     (int (*)(...))(& _ZTI6animal)
16    __cxa_pure_virtual
24    __cxa_pure_virtual
32    animal::hi

Vtable for cat
cat::_ZTV3cat: 5u entries
0     (int (*)(...))0
8     (int (*)(...))(& _ZTI3cat)
16    cat::makeSound
24    cat::showWeight2
32    animal::hi
Data Layout

• Alignment

• Padding

• Packing
Data Allocation

• Static

• Stack

• Heap
  – dangling references
Parameter Passing

• Call by Value

• Call by Reference

• Copy Restore

• Call by Name
Dynamic Memory

• Heap
  – malloc()

• Garbage Collection
  – reference counting
  – marking
  – dangling references