Heap-stack Diagram

Sanzheng Qiao

McMaster University

November, 2011

Outline

Sizes of objects

1 byte	`char`, `bool`
2 bytes	`short`
4 bytes	`int`, `float`
8 bytes	`long`, `double`
16 bytes	`long double`

1 byte = 8 bits

Sizes of objects

- enumerated type (`enum`): 4 bytes
- structure type (`struct`): enough space for all individual fields, no overlap
- array: enough space for all elements, assigned consecutive locations
- pointer: 4 bytes

Memory is byte addressable.

Outline

Generating heap-stack diagrams

- Start with an empty diagram, heap and stack side by side.
- Heap expands towards larger memory addresses, starting say 1000, grows downward on page, allocate heap memory when new operator appears
- Stack builds towards smaller memory addresses, starting say FFFF, grows upward on page

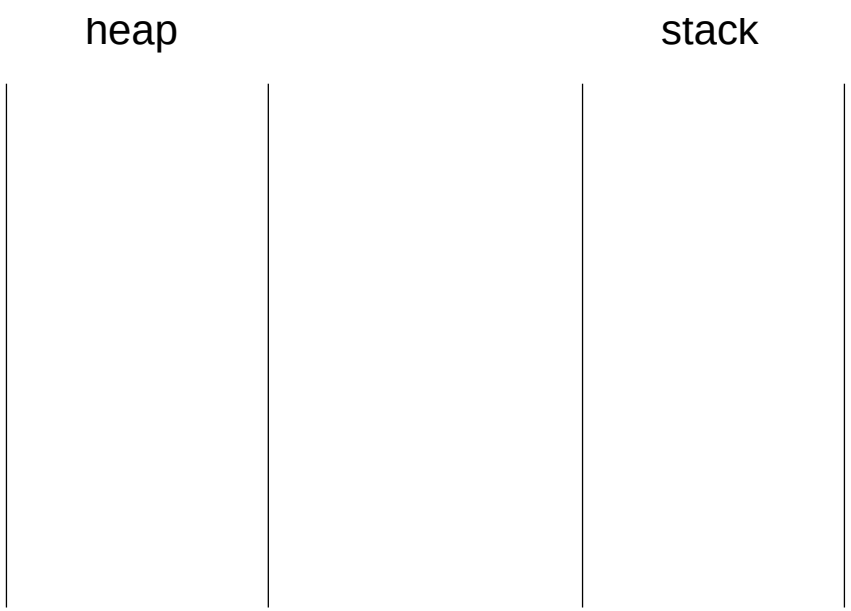
Generating heap-stack diagrams

- Manually trace the program, allocating memory as you go
- Add a new stack frame for each function call
  - Use a shaded block as overhead to separate stack frames
  - Spaces for all local variables (arguments, variables declared in the body, loop index), label the spaces
  - Initialize the arguments
  - Pop the stack frame when the function returns

Outline

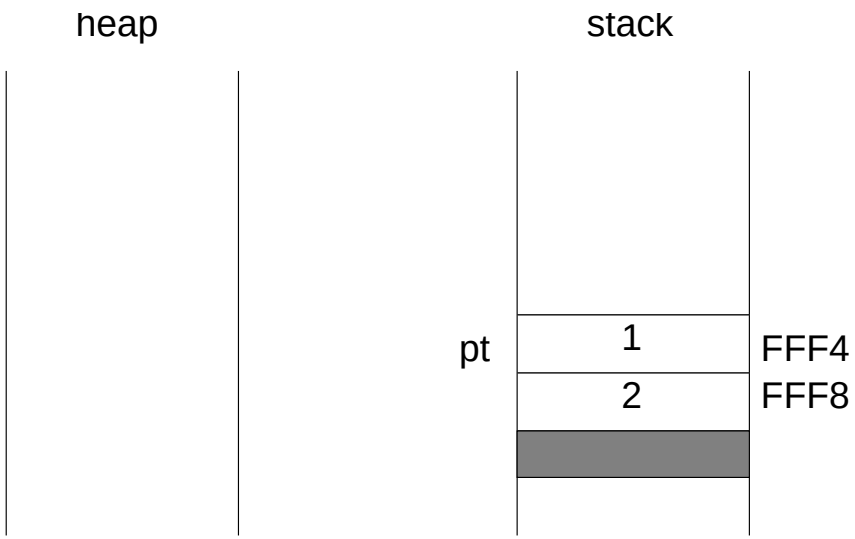
Example

```
int main() {
    pointT pt;
    pt.x = 1;
    pt.y = 2;
    int *array = new int[3];
    Nonsense(array, pt);
    return 0;
}

void Nonsense(int list[], pointT &ptr) {
    list[1] = ptr->x;
}
```



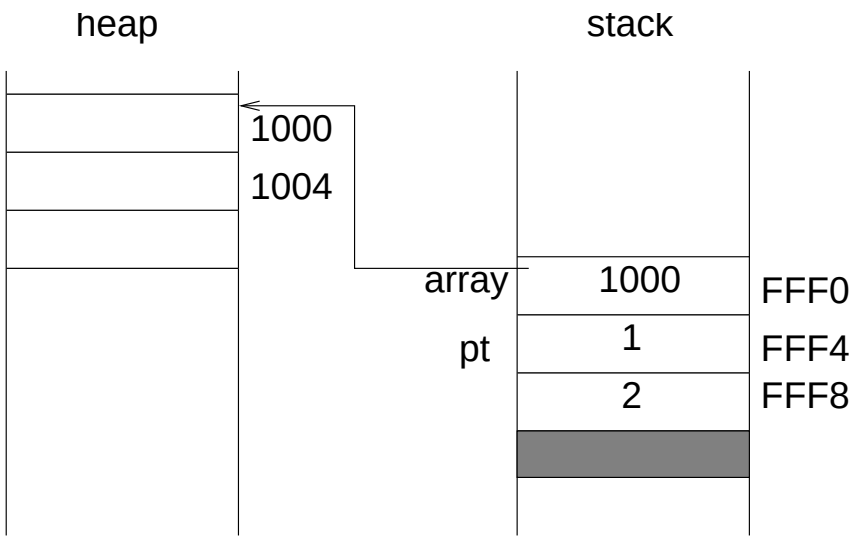
Example

```
int main() {
    pointT pt;
    pt.x = 1;
    pt.y = 2;
                                        <----- diagram
    int *array = new int[3];
    Nonsense(array, pt);
    return 0;
}

void Nonsense(int list[], pointT &ptr) {
    list[1] = ptr->x;
}
```



Example

```
int main() {
    pointT pt;
    pt.x = 1;
    pt.y = 2;
    int *array = new int[3];
                                        <----- diagram
    Nonsense(array, pt);
    return 0;
}

void Nonsense(int list[], pointT &ptr) {
    list[1] = ptr->x;
}
```



Example

```
int main() {
    pointT pt;
    pt.x = 1;
    pt.y = 2;
    int *array = new int[3];
    Nonsense(array, pt);
    return 0;
}

void Nonsense(int list[], pointT &ptr) {
    list[1] = ptr->x;
                                        <----- diagram
}
```