CSE 31
Computer Organization

Lecture 3 – C Pointers (cont.)

Announcements

- Lab
  - Lab 1 out this week
    - » Due at 11:59pm on the same day of your lab during week after next (with 7 days grace period after due date)
    - » You must demo your submission to your TA within 21 days
    - » Demo is REQUIRED to receive full credit
- Reading assignment
  - Chapter 4-6 of K&R (C book) to review C/C++ programming

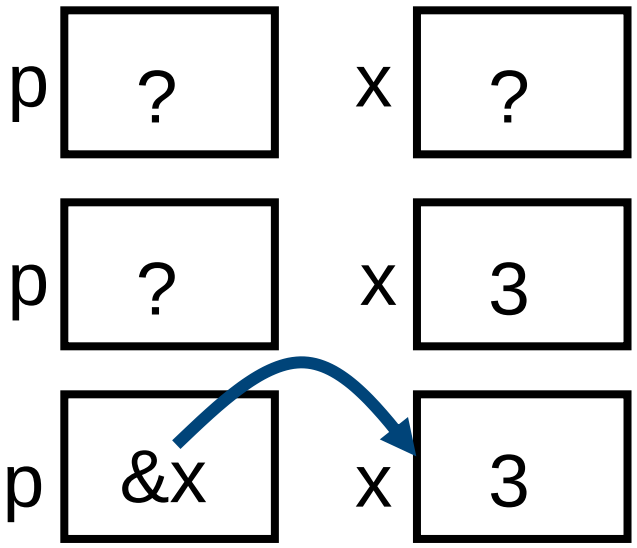
Pointers (review)

- How to create a pointer:
  & operator: get address of a variable

```
int *p, x;

x = 3;

p =&x;
```



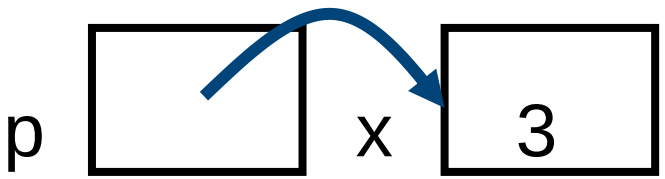
Note the "*" gets used 2 different ways in this example. In the declaration to indicate that p is going to be a pointer, and in the printf to get the value pointed to by p.

- How to get a value pointed to?
  - * "dereference operator": get value pointed to

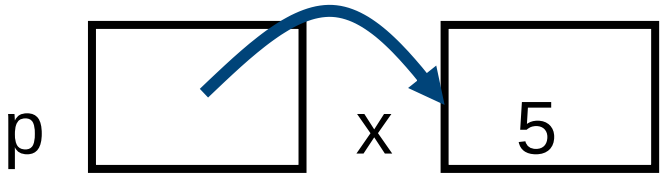
```
printf("p points to %d\n",*p);
```

Pointers

- How to change a variable pointed to?
  - Use dereference * operator on left of =




```
*p = 5;
```



Pointers and Parameter Passing

- Java and C pass parameters “by value”
  - procedure/function/method gets a copy of the parameter, so changing the copy cannot change the original

```
void addOne (int x) {
    x = x + 1;
}
int y = 3;
addOne(y);
```

`y` is still = 3 after the program ends!

Pointers and Parameter Passing

- How to get a function to change a value?

```
void addOne (int *p) {
   *p = *p + 1;
}

int y = 3;


addOne(&y);
```

`int *p = &y` when addOne() is called.

Passing the reference of y

`y` is now = 4 after the program ends.

Pointers

- Pointers are used to point to any data type (`int`, `char`, a `struct`, etc.).

- Normally a pointer can only point to one type (`int`, `char`, a `struct`, etc.).
  - `void *` is a type that can point to anything (generic pointer)
  - Use sparingly to help avoid program bugs… and security issues… and a lot of other bad things!

Pointers & Allocation (1/2)

- After declaring a pointer:

  `int *ptr;`

  `ptr` doesn't actually point to anything yet (it actually points somewhere - but we don't know where!).

- We can either:
  - make it point to something that already exists, or
  - allocate room in memory for something new that it will point to… (we will talk about it later)

C Pointer Dangers

- Unlike Java, C lets you cast a value of any type to any other type without performing any checking.

```
int x = 1000;
int *p = x;             /* invalid?? */
int *q = (int *) x; /* valid */
```

- The first pointer declaration should be invalid since the types do not match. (unsigned vs signed)
  - Does C allow this?

- The second declaration is valid in C but is almost certainly wrong
  - Is it ever correct?

More C Pointer Dangers

- Declaring a pointer just allocates space to hold the pointer – it does not *necessarily* allocate anything to be pointed to!
- Local variables in C are not initialized, they may contain anything.
- What does the following code do?

```
void f() {
    int *ptr;
    *ptr = 5;
}
```

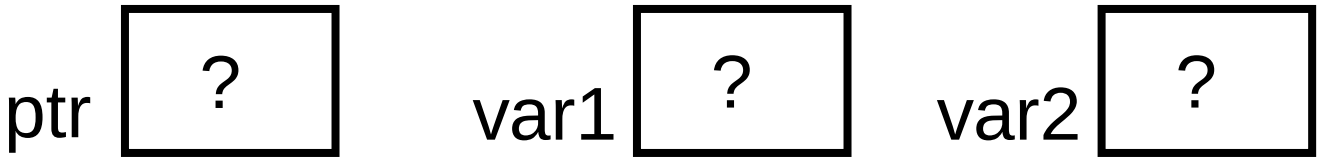
Where does it store the "5"?

Pointers & Allocation (2/2)

- Pointing to something that already exists:

```
int *ptr, var1, var2;
```

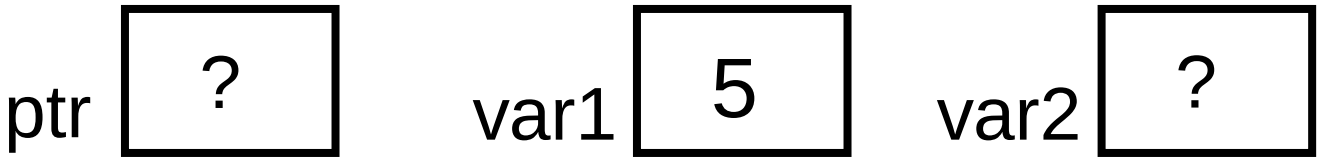
- `var1` and `var2` have room implicitly allocated for them.



Pointers & Allocation (2/2)

- Pointing to something that already exists:

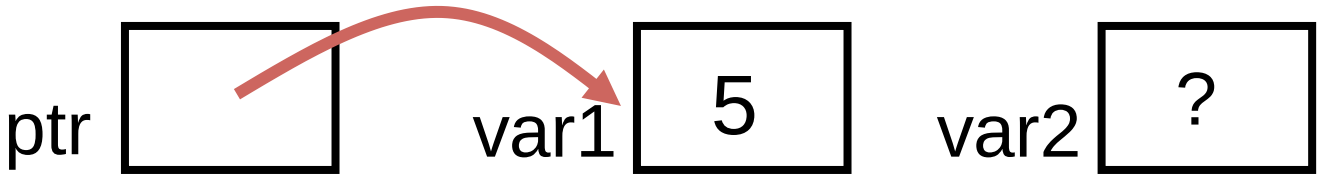
```
int *ptr, var1, var2;
var1 = 5;
```



Pointers & Allocation (2/2)

- Pointing to something that already exists:

```
int *ptr, var1, var2;
var1 = 5;
ptr  = &var1;
```



Pointers & Allocation (2/2)

- Pointing to something that already exists:

```
int *ptr, var1, var2;
var1 = 5;
ptr  = &var1;
var2 = *ptr;
```