

Passing Pointers

C is pass by value

I want a called function to change a variable in the calling function

- ▶ I can't pass the variable itself, pass by value *always* gives the called function a copy
- ▶ So we resort to pointers
 - Pass the address of the variable we want to change
 - The type of the parameter has one more * than the variable we want to change
 - The called function puts one * on the parameter when it wants to use or change the variable in the calling function

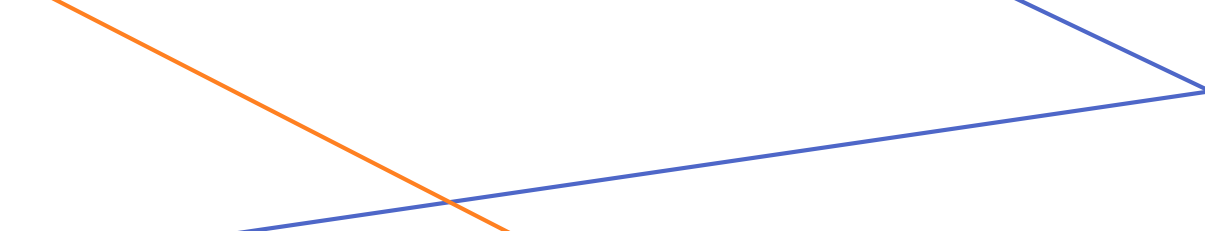
Example: int

```
void change_int( int *ip)
{
    *ip = 16;
}
```

```
void driver()
{
    int i = 0;
```

```
    change_int(&i);
}
```

The type of the parameter has one more * than the variable to be changed



Use one * to change the variable in the calling function



Pass the address of the variable to be changed using &



Example: an existing pointer

```
void change_p( char **xp)
{
    *xp = "Go bucks!";
}
```

```
void driver()
{
```

```
    char *ptr;
```

```
    change_p(&ptr);
```

```
}
```

The type of the parameter has one more * than the variable to be changed

Use one * to change the variable in the calling function

Pass the address of the variable to be changed using &

The same pattern holds for any type

- ▶ The parameter type has one more * than the variable to be changed
- ▶ Pass the address of the variable to be changed using &
- ▶ In the called function put one more * on the parameter to mean the same thing as the variable in the calling function
- ▶ If the variable I want to change has 3 * in the type, take the address and pass it to a parameter with 4 * in the type