Swap Revisited

/run/media/lijunjie/资料/Learning Video/pointers-arrays-recursion/01_pointers/02_pointers-conceptually/04_swap-revisited_instructions.html

Now that we know about pointers, we can write functions that are able to access and manipulate the variables of another function—we can pass pointers to the locations we want the called function to operate on. We now have the tools to write a correct version of swap. The key is to pass the swap function pointers to the original variables a and b, as seen here:

```
void swap(int *x, int *y) {
  int temp = *x;
  *x = *y;
  *y = temp;
}

int main(void) {
  int a = 3;
  int b = 4;

  swap(&a, &b);
  printf("a = %d, b = %d\n", a, b);
  return EXIT_SUCCESS;
}
```

Because the variables *x* and *y* hold arrows pointing to the original variables a and b, the function can swap the caller function's variables and not simply its own local copies. The parameters now tell us the *locations* of data to manipulate, rather than the *values* to manipulate. However, it can manipulate the values at those locations by dereferencing the pointers.