# CS-701 Lecture 6

February 20, 2001

Dr. Vickery

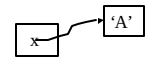
## Administrivia

- Need to limit the number of guests
- Send me a "codeword" if you want to be able to check your grades online.
  - Assignments 1 and 2 are being graded only "ok" or "not ok."
  - Submit future assignments as email attachments.
    - Do not send executables.
- What is ^M in a typescript?
  - Carriage return.

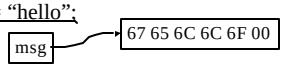
## Memory Structures

- int main(int argc, char * argv[], char * env[])
  - argv and env are arrays of pointers to *char*
  - A pointer to *char* normally means a pointer to the first char of a null-terminated string
  - Difference between
    - char * argv[]
    - char **argv

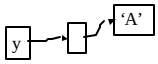
- Pointer: a variable that holds a memory address.
  - char *x; // x is a pointer to a char;
    // *x is a char.
  - *x = 'A';




- A C string is a sequence of character codes in successive memory bytes, terminated by an ASCII *nul* ('\0' or 0x00).
  - char *msg = "hello";




- char **y; // y is a pointer to a pointer to a char:




- Notes:
  - 'A' *might* be the beginning of a string
    - Or it might not be.
  - There has to be memory allocated for the two pointers as well as for the character(s).
    - When the program is compiled or
    - When the program executes
  - The middle pointer *might* be the beginning of an array of pointers
    - Or it might not be.

## Arrays

- Successive array elements occupy successive memory locations.
  - Think of strings as arrays of char, with an ASCII *nul* at the end.
  - Other arrays may or may not be terminated in some way. An array of ints would not be, but in many situations, arrays of pointers are terminated by *null*.
    - *nul* – an ASCII character code, 0x00.
    - *null* – a pointer with a numerical value of zero.

## Pointer Arithmetic

- If you add an integer to a pointer, the compiler multiplies the value of the integer by the size of the variable type in memory.
  - Useful for traversing the elements of an array.
  - Chars always occupy bytes; shorts usually occupy 2 bytes, ints 4, and longs 8. But this is not required by the C language, which says only:
    - sizeof(short) <= sizeof(int) <= sizeof(long)
    - sizeof(char) >= 1, sizeof(short) >= 2, and sizeof(long) >= 4

## Declaring Arrays and Pointers

- char *msg = "hello";
  - How big is msg?
  - What does msg = "good-bye"; do?
- char msg[] = "hello";
  - How big is msg?
  - What does msg = "good-bye"; do?
- So what is char *argv[]?
  - A *null* terminated array of strings.
    - Unix, not the C language, guarantees the *null*.

```
$ cat lecture.c
#include <stdio.h>
#include <string.h>
int
main( int argc, char *argv[], char *envp[] )
{
  char msg[] = "hello";
//  msg = "good-bye";
  strcpy(msg, "good-bye\n");  // Copies 10 bytes into 6-byte array!
  printf("msg: %d\n", sizeof( msg ) );
  printf("&msg: %d\n", sizeof( &msg ) );
  printf("*msg: %d\n", sizeof( *msg ) );
  printf( msg );
  return 0;
}

$ lecture
msg: 6
&msg: 8
*msg: 1
good-bye
```

*Does not compile*

## Some Points About Pointers

- char *msg = "hello";
  - Declares msg to be a pointer to char, creates a string, and assigns a pointer to the string to msg.
- char msg[] = "hello";
  - Declares msg to be an array of char, determines that the initializer is 6 bytes long, makes the array that large, and compiles the string into the array.
- char msg[6]; msg = "hello";
  - Syntax error because "hello" gives a pointer on the right side, but msg is an array of char on the left.
- char *msg; msg = "hello";
  - Okay; msg gets a pointer to the literal string in memory
- char *msg; strcpy(msg, "hello");
  - Runtime error because msg hasn't been initialized with a pointer to anything.

## More Points About Pointers

- char msg[] = "hello";
  - Now, msg and &msg[0] are both pointers to the first byte of the array.
  - Both *msg = 'J'; and msg[0] = 'J'; change the string to "Jello".
  - *(msg + 4) = 'a'; is the same as msg[4] = 'a';
  - msg[2] = '\0'; changes the string to "Je".
    - Memory ends up with 4A 65 00 6c 6f 00
      J e \0 l o \0
    - printf("%s\n", &msg[3]); // would print "lo"