Pointer Arithmetic and Arrays

Stanley Yao
Computer Science Department
University of Arizona

Pointer Review
- Starting address
- Base type
  - void *
  - * operator
- Size of the value
- Size of the pointer
- Size of the value

Pointer Arithmetic
- Depend on base type of the pointer

Array Declaration
- char c[2];
- short s[2];
- int i[2];

Array Initialization
- int a[3];
  - The size is required by compiler to decide the amount of memory to allocate
  - For local array, all elements’ default initial values are undefined values
  - For global array, all elements’ default initial values are 0’s
- int a[] = {10, 20, 30};
  - When initialization is given, size can be omitted

Char Array - String
- char str[] = “hello”;
- char *str = “hello”;
- char str[6];
- strcpy(str, “hello”);
- int len = strlen(str);
- printf(“%s”, str);
- printf(“%d”, str[2]);

- Stores the ASCII code of ‘o’ (111)
Example: strlen()

/* strlen: return length of string s */
int strlen(char *s) {
    int n;
    for (n=0; *s != '\0'; s++)
        n++;
    return n;
}

Array Argument

- As formal argument, char *s and char s[] are equivalent
- int foo(char *s), foo can regard s as:
  - A pointer, or
  - An char array (string)
- How does foo know the size of array s?
- Passing partial array: foo(&str[2]);
- Passing subarray of str starting from str[2]
- In foo, s[-1] is actually str[1]

Array Argument (cont.)

- In int foo(char *s), the size of array s is unknown
  - s is only a pointer
  - Pass another parameter indicating size if needed
  - Use s[0]
  - etc.

Example: strcpy()

/* strcpy(): array version */
void strcpy(char *s, char *t) {
    int i=0;
    while ((s[i]=t[i]) != '\0')
        i++;
}
/* strcpy(): pointer version */
void strcpy(char *s, char *t) {
    while (*s=*t)
        s++; t++;
}
/* strcpy(): pointer version 2 */
void strcpy(char *s, char *t) {
    while (*s++ = *t++);
}

Pointer Array

- char *lineptr[3];
- lineptr[0] = "abc";
- lineptr[1] = "cdef";
- lineptr[2] = "bcd";

Multi-dimensional Arrays

- Array of arrays
  - One dimensional array whose element is also array
  - int a[4][3] = {
      {2, 3, 3},
      {3, 1, 2},
      {3, 1, 1},
      {2, 2, 5}
    };
  - int *p = (int *)a;
  - Size of various pointers
Multi-dimensional Array as Arguments

- int foo(int a[4][3]); -- OK
- int foo(int a[][3]); -- OK
- int foo(int a[][[]]); -- Error

- Which is a[1][0]???
- Only the first dimension can be omitted

2-D Array .vs. Pointer Array

```c
char *a[3];

char a[][2];
```

String Array

- char *msg[] = {“OK”, “Wrong argument”, “NULL pointer”, “Out of boundry”}
- printf(“%s”, msg[1]);

String Array (cont.)

- char msg[][20] = {“OK”, “Wrong argument”, “NULL pointer”, “Out of boundry”}
- printf(“%s”, msg[1]);

Acknowledgement