CS262 Lecture 05
Chapter 5 Pointers 2

Jyh-Ming Lien
Department of Computer Science

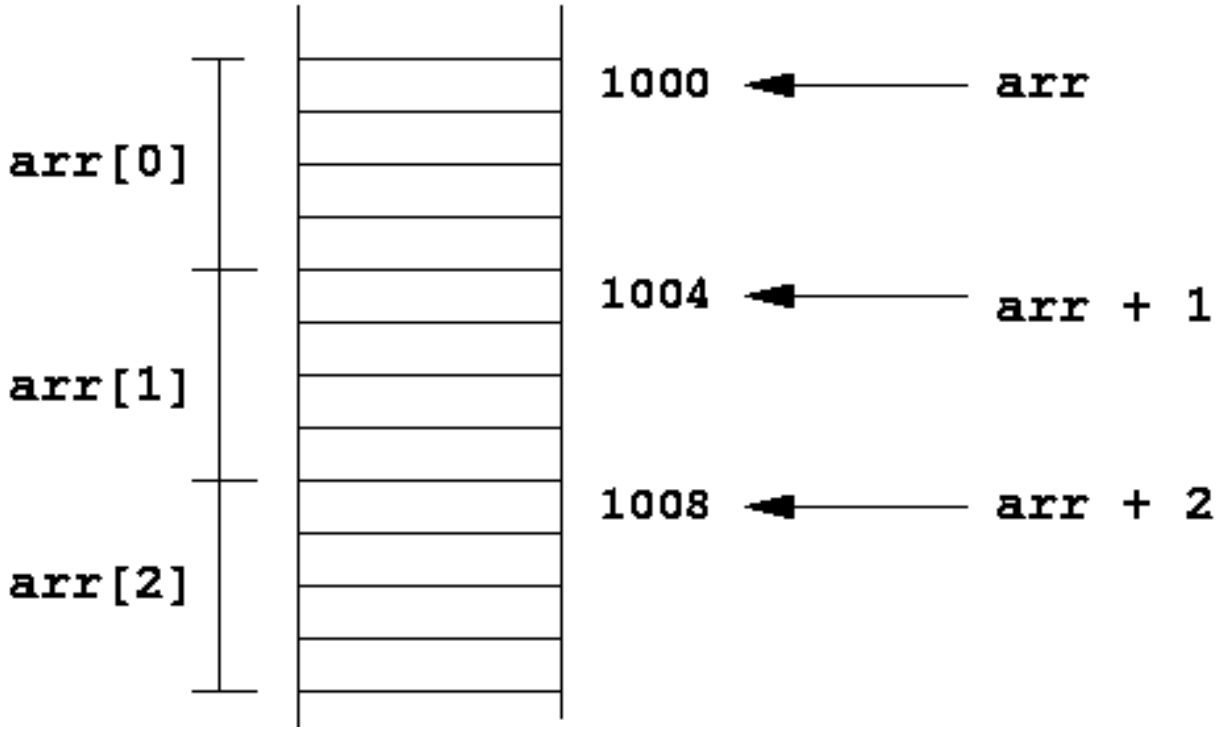
Address Arithmetic

- When p points to an element of an array
  - p++ moves the pointer to the next element
  - p-- moves the pointer to the previous element
- A[i]==*(A+i) **//i-th element of A**

- int * p1=0; p1++; **//what is the value of p1**
- long * p2=0; p2++; **//what is the value of p2**

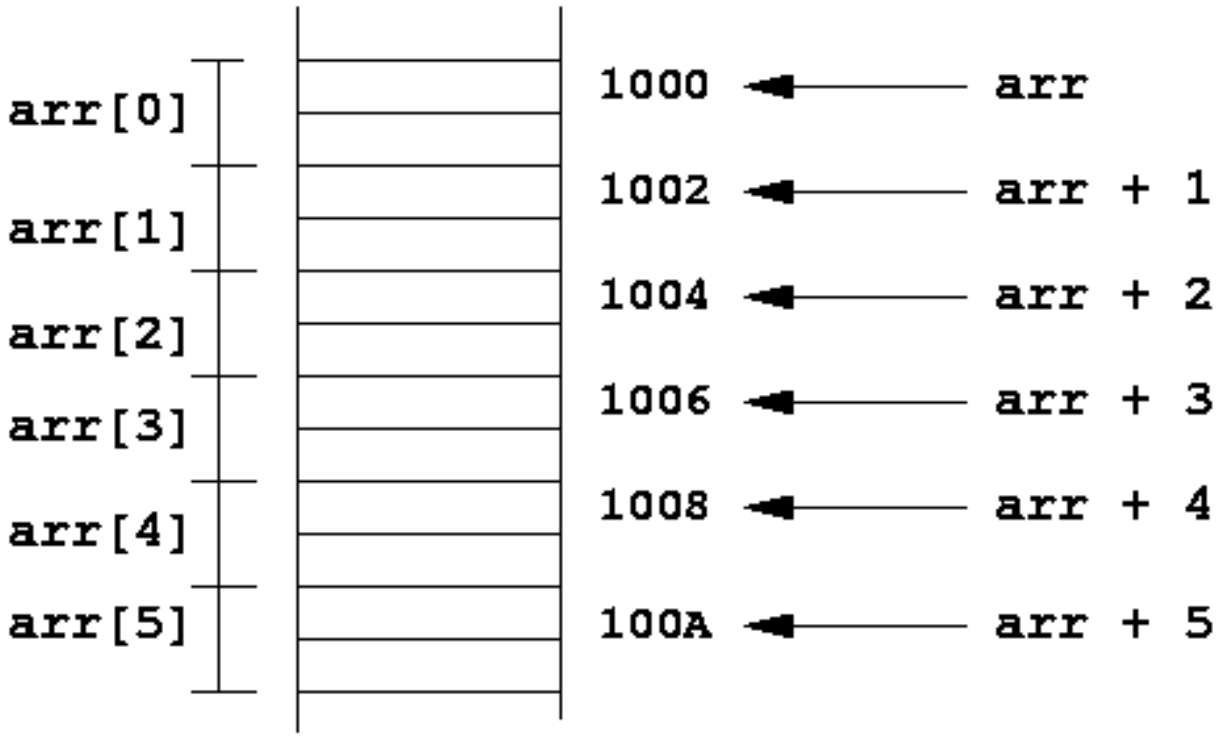
Address Arithmetic

- int A[100]
- A+i, contains the address &A[0]+i*sizeof(int)



Address Arithmetic

- short A[100]
- A+i, contains the address &A[0]+i*sizeof(short)



Address Arithmetic

- By reversing the process, we can retrieve the index
- int A[100];
- int * p=&A[10];
- int x=(p-A);
  - x is?

Address Arithmetic

- You can make pointer point to any address
- int A[10];
- int * p=A+10000; **//won't crash**
- *p=0; **//crash, because of dereferencing**

Bus Error

- bus error
  - access a physically impossible address
  - access an address that is not aligned

- see bus_err.c

Segmentation Fault

- segmentation fail
  - access read-only address
  - access private address
  - stack overflow (depends on the level of optimzation)

- See seg_fault.c, seg_fault2.c, overflow.c (try -O2 and without -O2)

Dynamic Memory Allocation

- void * malloc(# of bytes)
  - to create a dynamic array with ***n*** integer elements
  - the address to the first element is returned
  - when there is not enough space, null is returned
- int * A=(int *)malloc(sizeof(int)*10);
  - this creates an array with 10 integers
- when A is no longer used, deallocate A
  - free(A);

Dynamic Memory Allocation

- common errors
  - free(A+1); **//crash**
  - free(A); free(A); **//crash**
  - int A[10]; free(A);
  - int * p=malloc(4); int * q=p; free(q); free(p)
    - same as the second case

Dynamic Memory Allocation

- void * calloc(# of element, # of bytes)
  - equal to malloc((# of element)x(# of bytes))
- int * A=(int *)calloc(10,sizeof(int));
  - this creates an array with 10 integers

Other Related Functions

- void * **realloc** (void * **p**, long **n**);
  - expending or reducing allocated memory block pointed by **p** to **n** bytes
  - The content of the memory block is preserved
  - if p is null, realloc acts like malloc
  - if n==0, realloc acts like free
- void * **memset**(void * **p**, int **v**, long **n**)
  - Sets the first **n** bytes of the block of memory pointed by **p** to the specified value **v**
- void * **memcpy** (void * **B**, const void * **A**, long **n**);
  - Copy **n** bytes of the block of memory pointed by **A** to the memory block pointed by **B**
- memset and memcpy are usually faster than using for-loop