Computer Systems Organization

CS-UH 2010

Recitation 1

Khalid Mengal

Instructor

Khalid Mengal

Email: kqm1@nyu.edu

Office Hours: Wednesday 2:00-3:00pm

Desk: A2-186B

Agenda

- C strings
- Pointers and Arrays
- Dynamic Memory Allocation

3

Strings

- A string is a one-dimensional array of type char.
- The general form of a string is:

  char *s_name*[*size*];

  where
  - *s_name* is the name of the string
  - *size* is the number of characters in the string (-1)

String input

- The format %s is used to read in a string:

```
char name[100];
scanf("%s", name);
```

- **gets(s)** raads a line from *stdin* into the buffer pointed to by *s* until either a terminating newline or **EOF**, which it replaces with a null byte.
- **fgets(s, size, stream)** reads in at most one less than *size* characters from *stream* and stores them into the buffer pointed to by *s*. Reading stops after an **EOF** or a newline. If a newline is read, it is stored into the buffer. A terminating null byte is stored after the last character in the buffer.

Initializing Strings

- Strings can also be initialized like arrays.

  char s[ ] = {'N','Y','U','\0'};

  – The \0 or null character is used to terminate the string also called string terminator

- There is a second equivalent syntax for initializing character arrays.

  char s[ ]= "NYU";

String Handling Functions

- The standard library string.h contains many useful string handling functions.
  - strcat()
  - strcmp()
  - strcnmp()
  - strcpy()
  - strncpy()
  - strlen()
  - …

String Concatenation

- strcat(s1,s2)
  - Appends s2 to s1.
  - The string s1 is returned.

- strncat(s1,s2,n)
  - Appends exactly n characters of s2 to s1.

String Comparison

- **strcmp(s1,s2)**
  - Compare lexicographically s1 with s2
  - returns 0 if s1 is equal to s2
  - returns <0 if s1 is less than s2
  - returns >0 if s1 is greater than s2

- **strncmp(s1,s2,n)**
  - Like strcmp(), but only compares the first n characters of the strings.

String Copying

- strcpy(s1,s2)
  - s2 is copied into s1
  - Whatever exists in s1 is overwritten.
  - s1 is returned.

- strncpy(s1,s2,n)
  - only copies the first n characters of string s2 into s1

String Length

- strlen(str)
  - counts and returns the number of characters in str

String Conversions

- There are three functions, defined in the standard library stdlib.h which convert strings to numbers.
- atof(s)
  - Converts a string s to double.
- atoi(s)
  - Converts a string s to int.
- atol(s)
  - Converts a string s to long int.

String Conversions

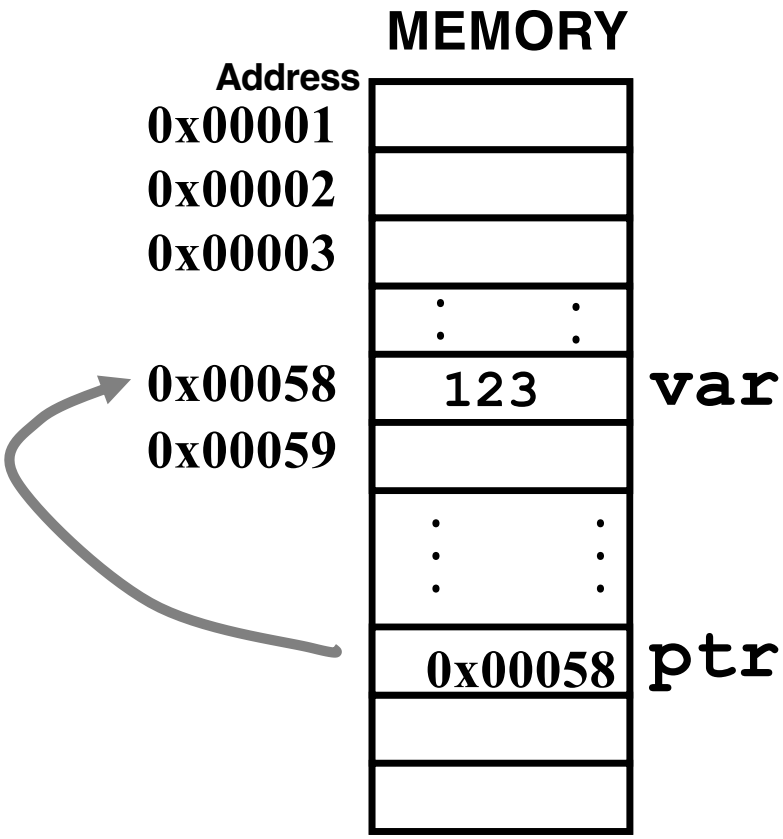
- For example:

```
char pi[ ] = “3.14159”
double PI;
PI = atof(pi);
printf(“%f\n”, PI);
```

Pointers

- A pointer is a variable that holds the *address* of something else.

```
int var;
int *ptr;
var = 123;
ptr = &var;
```



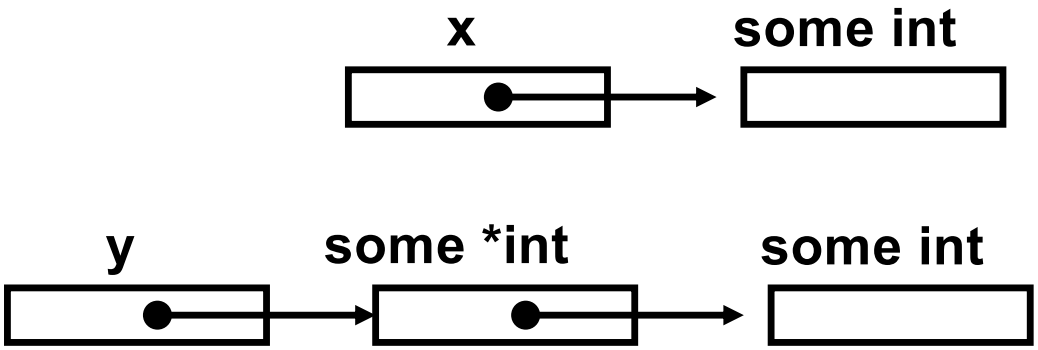
14

Advantages of using Pointers

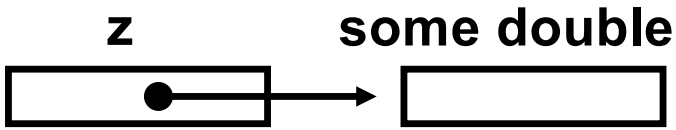
- Provide direct access to memory
- Make the program simple and efficient
- Allocate / deallocate memory during the execution of the program
- Pass arrays and c-strings to functions
- Return more than one value from a function

Pointers to anything

```
int *x;
int **y;
```




```
double *z;
```



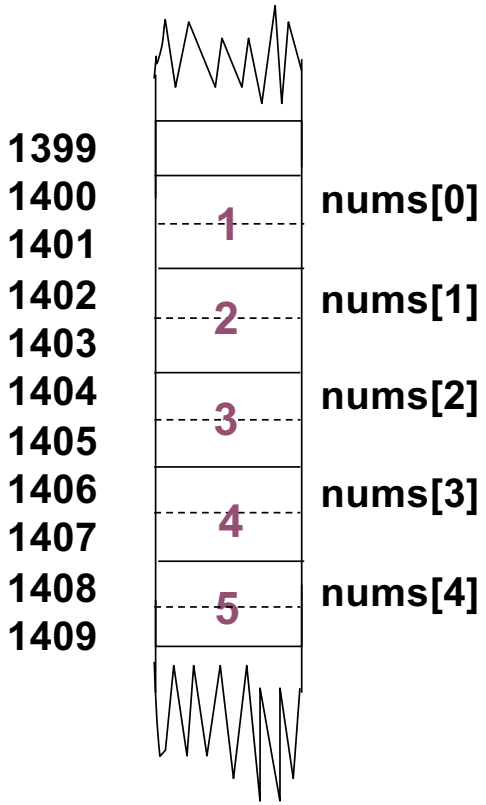
16

Array Notation

There is a close association between pointers and arrays.

```
short nums[ ]= { 1, 2, 3, 4, 5 };

for(int index=0; index<5; index++)
    printf("%i \n",nums[index]);
```

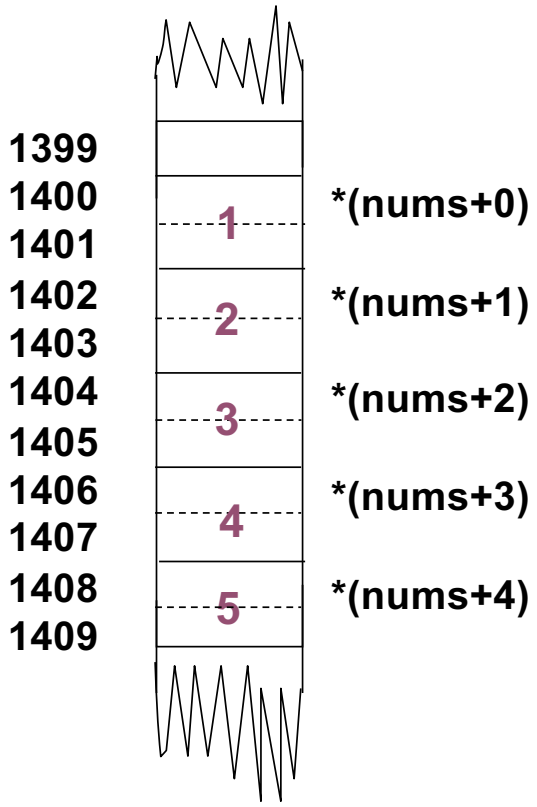


Pointer Notation

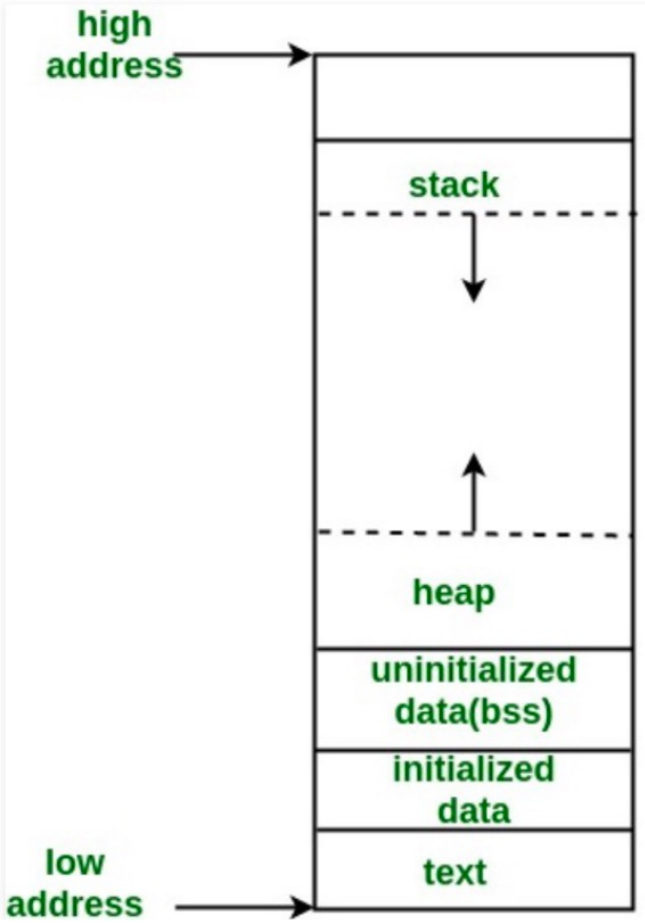
An array name is basically a const pointer.

```
short nums[10 ]= { 1, 2, 3, 4, 5 };

for(int index=0; index<5; index++)
    printf("%i \n",*(nums+index);
```



Program Address Space



- A ***program's address space*** is the range of ***logical*** addresses a program can operate on.
- A program address space is divided into three main areas:
  1) **Text/Code area -** near start of space
  2) **Initialized Data -** global and static variables
  3) **Un-initialized Data -** global and static variables
  4) **Heap** - middle of address space
  5) **Stack** - near top of address space

  stack grows, but direction of stack growth is OS dependent

Heap vs Stack Memory

- Stack
  - Fast Access
  - Contiguous
  - Automatic allocation/deallocation
  - Variables can not be resized
  - Limited Size (e.g. 8.192 MB)
    - ulimit –s (command to check stack size)
- Heap
  - Slow Access
  - Fragmented
  - Manual allocation/deallocation
  - Variables can be resized
  - Unlimited Size (determined by Physical RAM)

Allocating Variables Using malloc()

- **Malloc and realloc functions can be used to allocate dynamic memory in the heap**

  ***PointerType* * *PointerName* = (cast-type*) *malloc* (size-in-bytes);**

- **For example:**

```
ptr = (int*) malloc(100 * sizeof(int));

char *strPtr;                    // Two lines
strPtr = (char*)malloc (256* sizeof (char))
```

realloc() function

- Tries to enlarge the already allocated memory, if enough room available
- Otherwise create memory somewhere else and copies data from previous memory location to the new location
- New blocks will be uninitialized (garbage value).

```
ptr = realloc(ptr, newSize);
```

The free *function*

- Unlike Java, you as a programmer in C are responsible for deleting any dynamic memory objects you create.
- Deletion is accomplished by using the free() function and passing the pointer as an argument:

  free(*ptrName*);

- For example:

```
int *intArray = malloc(35*sizeof(int));          //Create an array of 35 ints in Heap
double *dPtr = malloc(sizeof(double));           //Create a double in Heap
free(intArray);                                  // Delete the array from Heap
free(dPtr);                                      // Delete double from Heap
```

Pointers & Functions: swap()

```
void swap(int *p, int *q)
  {
      int tmp;
      tmp = *p;
      *p = *q;
      *q = tmp;
    }
    .
    .
    .
    swap(&a, &b)            //Call swap function
```

Linux VM required for Assignment 3, 4

- Download Linux (CentOS) virtual machine image from following URL:
  - https://drive.google.com/file/d/1QLhvcIoK5nrkv40PnfHBb1ZKcPINibBG/view
  - Login ID: user
  - Password: user
  - root password is ”root” (in case if you want to install additional tools)
- Oracle Virtual Box can be downloaded from
  - https://www.virtualbox.org/wiki/Downloads

CentOS

Guest Additions for VM

- For better performace install guest additions for VM
  - Devices-> Insert Guest Additions CD image