NWEN 241
Systems Programming

Week 4 Tutorial

Pointer Application 2: Passing Function Parameters (1)

```
void swap( int a, int b )
{
    int temp = a;
    a = b;
    b = temp;
}
```

Why pass pointer as function input parameter?

- That is the only way to make the function work

Pointer Application 2: Passing Function Parameters (2)

```
typedef struct student_info {
        char name[40];
        int student_id;
        int age;
} StudentInfo;

void print_student(StudentInfo *s)
{
        printf("Name: %s\n", s->name);
        printf("Student ID: %d\n", s->id);
        printf("Age: %d\n", s->age);
}

…

StudentInfo s1 = {"John", 12345, 20};
print_student(&s1);
```

Why pass pointer as function input parameter?

- That is the only way to make the function work
- To make program more efficient

Storage Classes at a Glance

C storage class	Declaration	Default init value	Init frequency	Stored in	Scope	Lifetime
`auto`	Inside block	Garbage	Every time block is entered	Memory	Local	Automatic
`static`	Inside block	0	Once at program start	Memory	Local	Static
	Outside any block	0	Once at program start	Memory	Global	Static
`extern`	Outside any block	0	Once at program start	Memory	External	Static
`register`	Inside block	Garbage	Every time block is entered	Maybe in register	Local	Automatic

Pointer Application 3: Dynamic Memory Allocation

1. Calculate how much memory you want to allocate
   - Number of elements
   - Size of each element
2. Request for memory using either `malloc()` or `calloc()`
   - Assign return value to an appropriate pointer
   - Check if request is granted by checking return value
3. Use memory as intended
4. Release memory by using `free()`