NWEN 241
Systems Programming

Week 4 Tutorial

Introducing GDB

- GDB: GNU Debugger

- A much better way to debug your programs
  - No need to rely on `printf()` to see the values of the variables
  - You can step through your code
  - You can even change variable values!!!

- You will learn more about GDB in Exercise 2 (out on Tues, 19 March)

Pointers and Arrays

- **Array decays into a pointer: an array is just a fixed pointer**
- You cannot re-assign an array to point to another location
- You can let another pointer point to the array

```
int *p;
```

- p can point to an `int`
- p can point to an array of `int`

Clarification on Pointer & Arrays

- Consider:

```
int arr[10] = {1, 2, 3};
```

- Since arrays decay to fixed pointer:
  - `arr` is a (fixed) pointer
  - `arr` is the address of the array
  - `&arr` is also the address of the array
- Hence, to let a pointer p point to arr, we can write in 3 ways:

```
int *p;
p = arr;
```

```
int *p;
p = &arr[0];
```

```
int *p;
p = &arr;
```

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

Structures

```
// declare "struct person"  type
struct person {
    char name[100];
    int age;
};

// give it an alias person_t
typedef struct person person_t;
```

- Struct is just a collection of variables (which can have different types) under a single name
- You can access members with the '.' operator or through a pointer with the '->' operator
- A struct can be referenced, copied, and assigned to
- The size of a struct is guaranteed to be as large as the sum as the size of its members

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