Chapter 11. Function Pointers

Byoung-Tak Zhang
TA: Hanock Kwak
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

http://bi.snu.ac.kr

Code and Data

```
int foo( ){
   ......
}

int main (int argc, char* argv[]) {
     foo() ;
     ......
}
```

Assembly code

```
main:
.LFB6:
      pushq   %rbp
.LCFI2:
      movq    %rsp, %rbp
.LCFI3:
      subq    $32, %rsp
.LCFI4:
      movl    %edi, -20(%rbp)
      movq    %rsi, -32(%rbp)
      movl    $10, -4(%rbp)
      movl    -4(%rbp), %edi
      call    foo
      leave
      ret
```

Function as Pointers

- Function code is stored in memory
- Start of the function code or the address of a function is a “function pointer”
- Function pointer is “different” from other pointers since you do not allocate or deallocate memory with them
- Function pointers can be passed as arguments to other functions or return from functions

Why use function pointers?

- Efficiency
- Elegance
- Runtime binding
  - e.g. Determine sorting function based on type of data at run time.

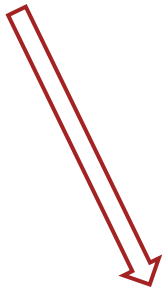
Why use function pointers?

```
#include <stdio.h>
#include <stdlib.h>

int values[] = { 40, 10, 100, 90, 20, 25 };

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

int main ()
{
  int n;
  qsort (values, 6, sizeof(int), compare);
  for (n=0; n<6; n++)
     printf ("%d ",values[n]);
  return 0;
}
```



- The function qsort sorts any type of arrays with a user defined comparison function.

Defining a function pointer

- *return_type* (* *identifier*) (*argument_type1*, *argument_type2*, ...);
  - **int (*f)(int, int);**
  - **f** is a variable

- Type Definition
  - **typedef int (*Ptr)(int, int)**
  - **Ptr** is a type

```
#include <stdio.h>

int sum(int a, int b)
{
    return a + b;
}

int main()
{
    int (*fn)(int, int);

    fn = sum;
    printf("%d\n", fn(1, 2));
}
```

Defining a function pointer

```c
#include <stdio.h>

typedef int (*Ptr)(int, int);

int sum(int a, int b)
{
    return a + b;
}

int mul(int a, int b)
{
    return a * b;
}
```

```c
int mystery(int a, int b, Ptr f)
{
    return f(a,b);
}

int main()
{
    printf("%d\n", mystery(1, 2, sum));
    printf("%d\n", mystery(1, 2, mul));
}
```

qsort

- A utility function that can be used to sort an array with any type.
- **void qsort(void* base, size_t num, size_t size,**
  int (*compare)(const void*,const void*));
  - **base**
    - Pointer to the first object of the array to be sorted, converted to a void*.
  - **num**
    - Number of elements in the array pointed by *base*.
  - **size**
    - Size in bytes of each element in the array.
  - **compare**
    - Pointer to a function that compares two elements.

qsort

- Compare Function
  - int compare (const void* p1, const void* p2)

```
int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}
```

return value	meaning
negative integer	The element pointed by *p1* goes befor e the element pointed by *p2*
0	The element pointed by *p1* is equivale nt to the element pointed by *p2*
positive integer	The element pointed by *p1* goes after the element pointed by *p2*

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int compare(const void * a, const void * b)
{
    return strcmp(*(char**)a, *(char**)b);
}


int main()
{
    char *arr[] = {"bird", "zibra", "donkey", "cobra", "mouse"};
    int i;

    qsort(arr, 5, sizeof(char*), compare);

    for (i = 0; i < 5; i++)
        printf("%s\n", arr[i]);
}
```