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
int foo() {
    // ... 
}

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

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

```c
#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;
}
```
Defining a function pointer

- \textit{return\_type (* identifier) ( argument\_type1, argument\_type2, ...)};
  - \texttt{int (*f)(int, int);}
  - \texttt{f} is a variable

- Type Definition
  - \texttt{typedef int (*Ptr)(int, int)}
  - \texttt{Ptr} is a type

```c
#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));
}
```
#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;
}

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)`

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

<table>
<thead>
<tr>
<th>return value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative integer</td>
<td>The element pointed by <code>p1</code> goes before the element pointed by <code>p2</code></td>
</tr>
<tr>
<td>0</td>
<td>The element pointed by <code>p1</code> is equivalent to the element pointed by <code>p2</code></td>
</tr>
<tr>
<td>positive integer</td>
<td>The element pointed by <code>p1</code> goes after the element pointed by <code>p2</code></td>
</tr>
</tbody>
</table>
```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]);
}
```