Quick Sort and Function Pointers

Amir Modarresi

Dr. Heechul Yun

Introduction

- In this lab, we review void pointers and function pointers in C, which are used in your second project.
- Generally a pointer refers to the address of a variable. The type of the pointer indicates the type of the variable which is stored in that address. It also helps operators on pointers work correctly.
- For example:

```
int * ptr;     /* defines an integer pointer with general
                  format <type> * var_name */
int num=5;
ptr= &num;     /* assigns the address of number to ptr */
printf (“The value of the variable num is %d”, *ptr);
```

Void Pointers

- Void pointers are general purpose pointers in C. They are useful, when you don’t know the type of data in advance, or you want to refer to different type of data with one pointer.
- For example:

```
void * ptr;   /* defines a void pointer with the format of
                 void * var_name */
char myChar;
int  myInt;
float myFloat;
ptr= &myChar; /* ptr keeps the address of character data
*/
ptr= &myInt;  /* ptr keeps the address of integer data */
ptr= &myfloat;/* ptr keeps the address of float data */
```

Void Pointers

- Generally, we can say that when a void pointer is assigned to a specific type pointer, it changes to that type of pointer.
- For example:

```
void * ptr;
int num=10;
char ch='c';
ptr = & num;
printf("%d", *(int *)ptr;
ptr = &ch;
printf("%c", *(char *)ptr;
```

Function Pointers

- Function pointers are similar to data pointers except they point to a function.
- It declares as:

```
type (* POINTER_name) (arg 1, arg2, … )
```

Function Pointer Declaration

- One easy way for declaration:
  - write your normal function declaration like:
    - `int myFunc (int a, int b)`
    - this is a function with two int arguments and returns int value.
  - wrap function name with the pointer syntax:
    - `int (*myFunc) (int a, int b)`
  - change the function name to a pointer name:
    - `int  (*comparer) (int a, int b)`
    - it points to a function with two integer arguments, where that function returns an integer value

Function Pointers – Similarity and Differences

- Differences with data pointers:
  - they point to code instead of data
  - we don’t allocate or deallocate memory for this type of pointers
  - you can use, either function name or &function name to assign its address to a function pointer.
- Similarity to data pointers:
  - we can define array of function pointers, where each elements refer to one function.
  - a function pointer can be passed as an argument to a function or be return from a function.

bubble sort with Function Pointer

```
typedef int (*comparer) (int a, int b)
void bubbleSort ( int * numbers, int count, comparer cmp)
{
  int temp=0, i=0, j=0;
  for(i=0; i<count; i++) {
       for (j=0; j<count -1; j++) {
            if (cmp(numbers[j], numbers[j+i])> 0{
                 temp=numbers[j+1];
                 numbers[j+1]=numbers[j];
                 numbers[j]=temp;
}}}}
int ascending_order( int a, int b){
       return a-b; }
```

```
int descending_order(int a, int b){
       return b-a; }
```

bubble sort with Function Pointer-cont.

```
int main ()
{
        int count;
        int * numbers;
        …
        int *numbers=malloc (count * sizeof(int));
        …
                bubble_sort(numbers, count, ascending_order);
…
}
```

qsort

- qsort is a sorting function in ‘C’ standard library. You can find more information about it in man qsort.
- It is in ***<stdlib.h>*** standard library and has the following format:
- `void qsort(void *base, size_t nmemb, size_t size, int (*compar) (const void *, const void *))`
- It sorts an array with starting address `base`, `nmemb` elements of size `size`.
- The content of array is sorted based on a comparison function pointed to by `compar`.
- The comparison function should return an integer value less than, equal to, or greater than zero, if the first argument is less than, equal to, or greater than the second argument. However, you can change comparison function based on your need.

Assignment

- Write a program that sorts the content of a input file called process.txt with qsort and consider the following conditions in your program:
  - process.txt has the following data

Pid	Arrival time	priority
5	10	3
2	4	0
7	14	0
3	6	1
1	2	1
4	8	2
6	12	3

Assignment-cont.

- Your program should be able to sort the content of the file based on arrival time and priority.
  - if it is sorted based on arrival time, it should be ascending order.
  - if it is sorted based on priority, it should be descending order. However, if two priority values are equal, arrival time should be consider as the second condition. The smaller arrival time with equal priority comes first.
- Your program should print the content of the sorted file based on arrival time and priority.

Conclusion

- Although this lab is not directly related to the concept of the operating systems that you learn in the class, it is used in system programming like operating system.
- Function pointers and void pointers add extra flexibility to programs and make a function or a pointer usable for different conditions.