Function Pointers

15-123

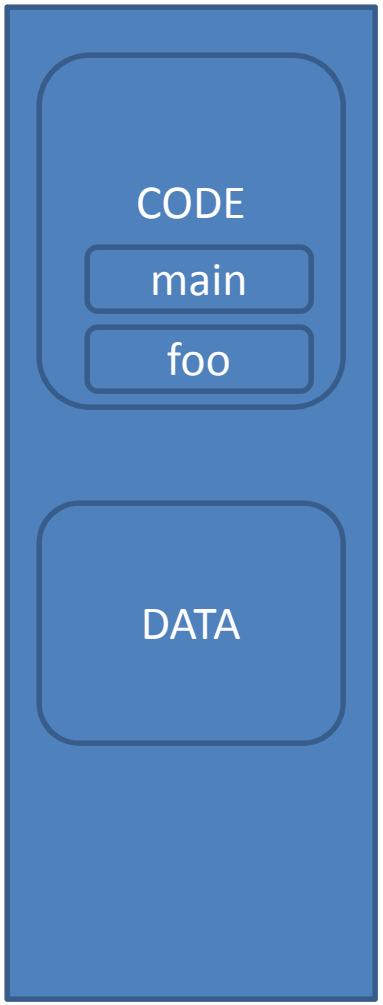
Systems Skills in C and Unix

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
```



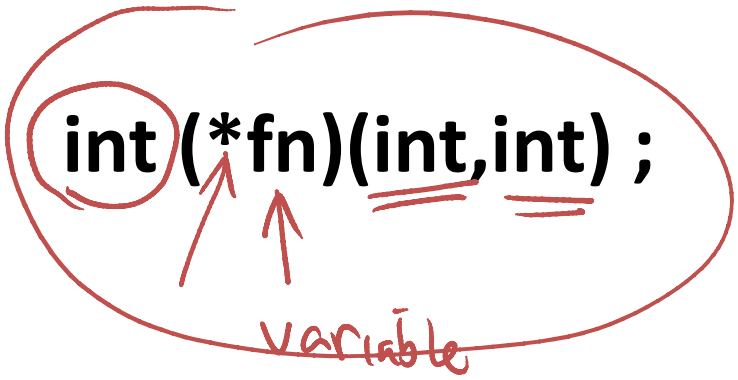
Functions as pointers

- 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
  - Determine sorting function based on type of data at run time
    - Eg: insertion sort for smaller data sets (n <100)
    - Eg: Quicksort for large data sets (n > 100000)
    - Other sorting algorithms based on type of data set

Defining a function pointer




```
int GCD(int a, int b) {
    ...
    return gcd;
}
fn = &GCD;
```

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

```
Ptr fn;
```

```
printf("%d", fn(10,15));
```

Using function pointers

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

Now let us define two functions as follows.

```
int gcd(int a, int b) { ….}
int sumofsquares(int x, int y) { return (x*x + y*y);}
```

Now let us call mystery

```
fn = &gcd;
printf("%d", mystery(10, 15, fn))
```

Functions that return function pointers

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

```
Ptr Convert(const char code) {
    if (code = = '+') return &Sum;
    if (code = = '-') return &Difference;
}
```

Use the Convert function as follows

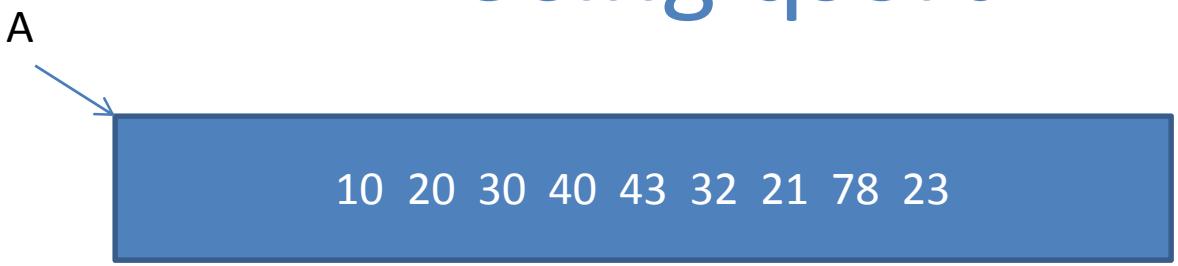
```
int main ( ) {
   int (*ptr)(int,int); /* or:  Ptr ptr; if you have a typedef */
   ptr = Convert('+');
   printf( "%d \n", ptr(2,4));
}
```

qsort

- A unix utility function that can be used to sort any data set stored in an array (really cool!!)

```
NAME
       qsort - sorts an array

SYNOPSIS
       #include <stdlib.h>

       void qsort(void *base, size_t nmemb, size_t size,
                  int(*compar)(const void *, const void *));

DESCRIPTION
       The  qsort()  function  sorts  an  array with nmemb elements of size
       size.  The base argument points to the start of the array.

       The contents of the array are sorted in ascending order according to
       a comparison function pointed to by compar, which is called with two
       arguments that point to the objects being compared.

       The comparison function must return an integer less than, equal  to,
       or  greater  than  zero  if  the  first argument is considered to be
       respectively less than, equal to, or greater than  the  second.   If
       two  members  compare  as  equal, their order in the sorted array is
       undefined.

RETURN VALUE
       The qsort() function returns no value.
```

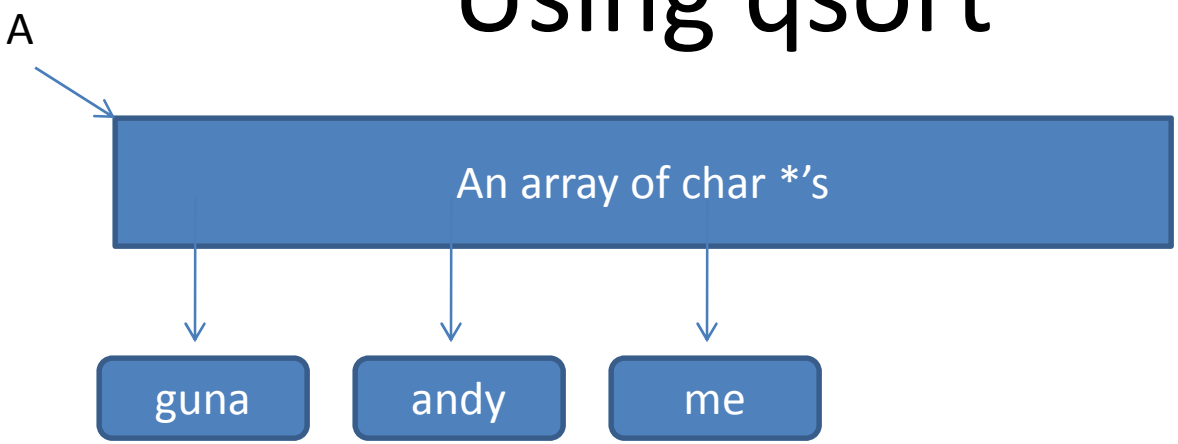
Using qsort



A is an array of integers. Sort it using qsort with natural integer order

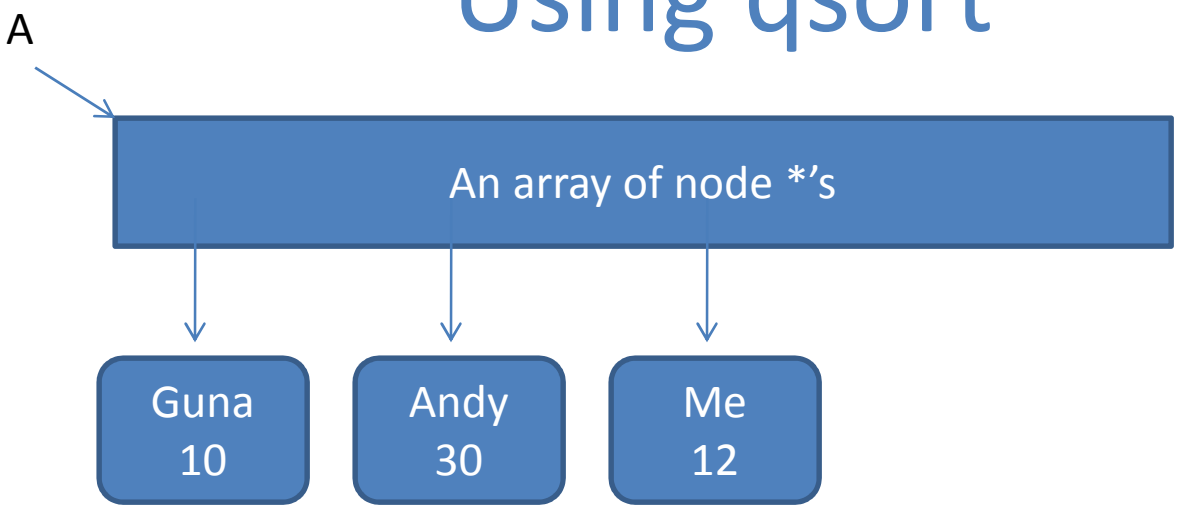
Write the compare function:
int (*intcomp)(const void * a, const void * b))

Using qsort



Write the compare function to sort by alpha order
Int (*strcomp)(const void * a, const void * b))

Using qsort



Write the compare function to sort by name:
int (*nodecomp)(const void * a, const void * b))

Example

- Write a compare function to sort by first character of name

Write the compare function to sort by alpha order
Int (*firstnamecharcompar)(const void * a, const void * b))

Sorting an 2D array of char's

```
char B[][10]={"guna","mccain","obama","paul","barr"};
qsort(*B,5,10,namecmp);
for (i=0;i<5;i++)
    printf("%s \n",B[i]);
```

Generic Bubble sort with function pointers

```
typedef  int  (*cmp)(const void*, int , int);
typedef  int  (*swap)(void*, int, int);

int bubblesort(void* A, int n, cmp cfn, swap sfn){
    int i, j;
    for  (i=0;i<n-1; i++)
      for (j=0; j<n-i-1; j++)
        if (cfn(A, j, j+1)>0)
            sfn(A, j, j+1);
}
```

Using Bubble Sort

```
int main(int argc, char* argv[]){
  int A[] = {31,52,21,45,18}, i;
  bubblesort(A,5,intcmpr, intswap);
  for (i=0; i<5; i++)
    printf("%s ",A[i]);
  printf("\n");
  return EXIT_SUCCESS;
}
```

- Homework: Write the intcmpr and intswap functions

Coding Examples