

CMSC 313

COMPUTER ORGANIZATION

&

ASSEMBLY LANGUAGE

PROGRAMMING

LECTURE 17, SPRING 2013

C FUNCTION CALL CONVENTIONS



Linux/gcc/i386 Function Call Convention

- **Parameters pushed right to left on the stack**
 - ◇ first parameter on top of the stack
- **Caller saves EAX, ECX, EDX if needed**
 - ◇ these registers will probably be used by the callee
- **Callee saves EBX, ESI, EDI**
 - ◇ there is a good chance that the callee does not need these
- **EBP used as index register for parameters, local variables, and temporary storage**
- **Callee must restore caller's ESP and EBP**
- **Return value placed in EAX**

A typical stack frame for the function call:

```
int foo (int arg1, int arg2, int arg3) ;
```

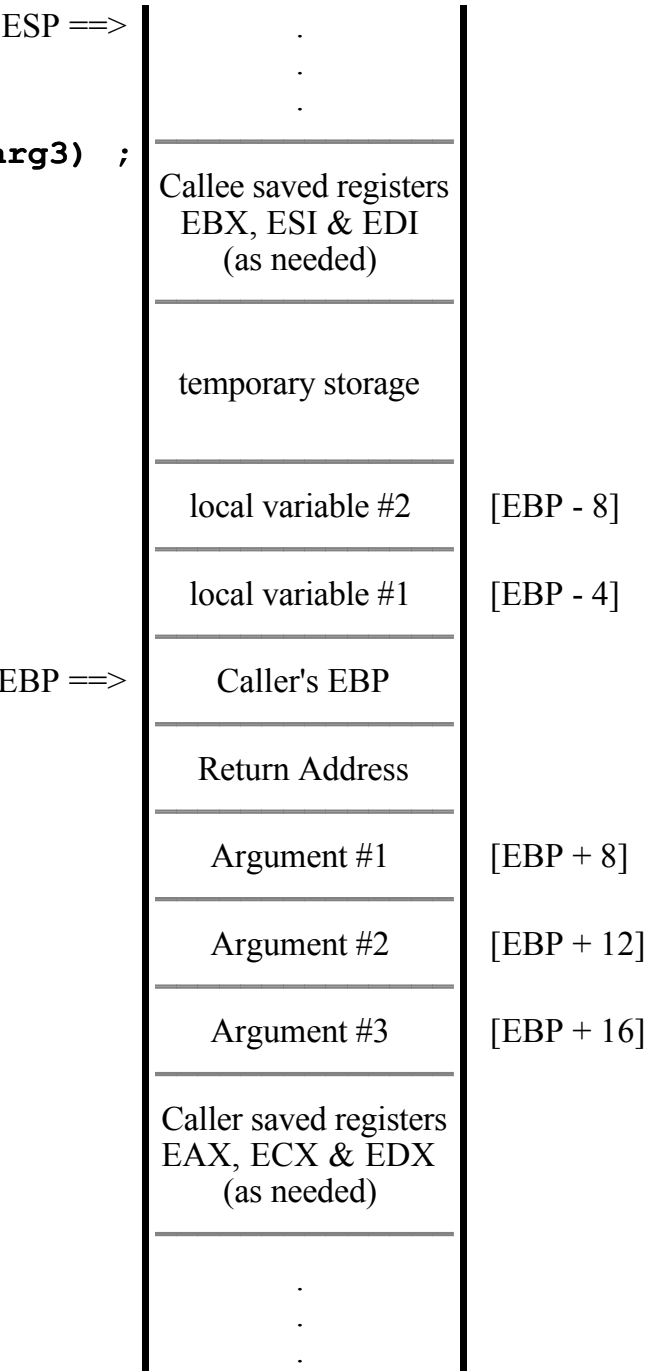


Fig. 1

```
// File: cfunc.c
//
// Example of C function calls disassembled
//
```

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

```
// a silly function
//
```

```
int foo(int x, int y) {
```

```
    int z ;
```

```
    z = x + y ;
```

```
    return z ;
```

```
}
```

```
int main () {
```

```
    int b ;
```

```
    b = foo(35, 64) ;
```

```
    b = b + b ;
```

```
    printf ("b = %d\n", b) ;
```

```
}
```

```
linux3% gcc cfunc.c
```

```
linux3% a.out
```

```
b = 198
```

```
linux3%
```

```
linux3% gcc -S cfunc.c
```

```
linux3% i2g -g cfunc.s >cfunc.asm
```

```
linux3%
```

```

        .file    "cfunc.c"
        .version    "01.01"
gcc2_compiled.:
.text
        .align 4
.globl foo
        .type    foo,@function
foo:
        pushl %ebp
        movl %esp,%ebp
        subl $4,%esp
        movl 8(%ebp),%eax
        movl 12(%ebp),%edx
        leal (%edx,%eax),%ecx
        movl %ecx,-4(%ebp)
        movl -4(%ebp),%edx
        movl %edx,%eax
        jmp .L1
        .p2align 4,,7
.L1:
        leave
        ret

```

```

.Lfe1:
        .size      foo, .Lfe1-foo
.section      .rodata
.LC0:
        .string "b = %d\n"
.text
        .align 4
.globl main
        .type      main, @function
main:
        pushl %ebp
        movl %esp, %ebp
        subl $4, %esp
        pushl $64
        pushl $35
        call foo
        addl $8, %esp
        movl %eax, %eax
        movl %eax, -4(%ebp)
        movl -4(%ebp), %eax
        addl %eax, -4(%ebp)
        movl -4(%ebp), %eax
        pushl %eax
        pushl $.LC0
        call printf
        addl $8, %esp

.L2:
        leave
        ret

.Lfe2:
        .size      main, .Lfe2-main
        .ident      "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"

```

```
        ;FILE "cfunc.c"
gcc2_compiled.:
SECTION .text
        ALIGN 4
GLOBAL foo
        GLOBAL foo:function
foo:
        push    ebp
        mov     ebp,esp
        sub     esp,4
        mov     eax,[ebp+8]
        mov     edx,[ebp+12]
        lea     ecx,[edx+eax]
        mov     [ebp-4],ecx
        mov     edx,[ebp-4]
        mov     eax,edx
        jmp     L1
        ;ALIGN 1<<4 ; IF < 7
L1:
        leave
        ret
```

```

.Lfe1:
        GLOBAL    foo:function (.Lfe1-foo)
SECTION .rodata
.LC0:
        db        'b = %d',10,''
SECTION .text
        ALIGN 4
GLOBAL main
        GLOBAL main:function
main:
        push    ebp
        mov     ebp,esp
        sub     esp,4
        push    dword 64
        push    dword 35
        call    foo
        add     esp,8
        mov     eax,eax
        mov     [ebp-4],eax
        mov     eax,[ebp-4]
        add     [ebp-4],eax
        mov     eax,[ebp-4]
        push    eax
        push    dword .LC0
        call    printf
        add     esp,8

L2:
        leave
        ret

.Lfe2:
        GLOBAL    main:function (.Lfe2-main)
        ;IDENT "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"

```

```

; File: printf1.asm
;
; Using C printf function to print
;
; Assemble using NASM:  nasm -f elf printf1.asm
;
; C-style main function.
; Link with gcc:  gcc printf1.o
;

; Declare some external functions
;
extern printf                                ; the C function, we'll call

SECTION .data                                ; Data section

msg:    db "Hello, world: %c", 10, 0         ; The string to print.

SECTION .text                                ; Code section.

global main

main:
    push    ebp                                ; set up stack frame
    mov     ebp, esp

    push    dword 97                          ; an 'a'
    push    dword msg                        ; address of ctrl string
    call    printf                            ; Call C function
    add     esp, 8                            ; pop stack

    mov     esp, ebp                          ; takedown stack frame
    pop     ebp                              ; same as "leave" op

    ret

```

```

linux3% nasm -f elf printf1.asm
linux3% gcc printf1.o

```

```

linux3% a.out
Hello, world: a
linux3% exit

```

```

; File: printf2.asm
;
; Using C printf function to print
;
; Assemble using NASM:  nasm -f elf printf2.asm
;
; Assembler style main function.
; Link with gcc: gcc -nostartfiles printf2.asm
;

%define SYSCALL_EXIT 1

; Declare some external functions
;
extern printf                ; the C function, we'll call

SECTION .data                ; Data section

msg:    db "Hello, world: %c", 10, 0    ; The string to print.

SECTION .text                ; Code section.

global _start
_start:
    push    dword 97          ; an 'a'
    push    dword msg         ; address of ctrl string
    call    printf            ; Call C function
    add     esp, 8            ; pop stack

    mov     eax, SYSCALL_EXIT ; Exit.
    mov     ebx, 0            ; exit code, 0=normal
    int     080H              ; ask kernel to take over

```

```

linux3% nasm -f elf printf2.asm
linux3% gcc -nostartfiles printf2.o
linux3%

```

```

linux3% a.out
Hello, world: a
linux3%

```

```
// File: arraytest.c
//
// C program to test arrayinc.asm
//

void arrayinc(int A[], int n) ;

main() {

int A[7] = {2, 7, 19, 45, 3, 42, 9} ;
int i ;

    printf ("sizeof(int) = %d\n", sizeof(int)) ;

    printf("\nOriginal array:\n") ;
    for (i = 0 ; i < 7 ; i++) {
        printf("A[%d] = %d  ", i, A[i]) ;
    }
    printf("\n") ;

    arrayinc(A,7) ;

    printf("\nModified array:\n") ;
    for (i = 0 ; i < 7 ; i++) {
        printf("A[%d] = %d  ", i, A[i]) ;
    }
    printf("\n") ;

}
```

```
linux3% gcc -c arraytest.c
linux3% nasm -f elf arrayinc.asm
linux3% gcc arraytest.o arrayinc.o
linux3%
linux3% a.out
sizeof(int) = 4
```

Original array:

A[0] = 2 A[1] = 7 A[2] = 19 A[3] = 45 A[4] = 3 A[5] = 42 A[6] = 9

Modified array:

A[0] = 3 A[1] = 8 A[2] = 20 A[3] = 46 A[4] = 4 A[5] = 43 A[6] = 10

linux3%

```
; File: arrayinc.asm
;
; A subroutine to be called from C programs.
; Parameters: int A[], int n
; Result: A[0], ... A[n-1] are each incremented by 1
```

```
SECTION .text
global arrayinc
```

```
arrayinc:
```

```
    push    ebp                ; set up stack frame
    mov     ebp, esp
```

```
    ; registers ebx, esi and edi must be saved, if used
    push    ebx
    push    edi
```

```
    mov     edi, [ebp+8]       ; get address of A
    mov     ecx, [ebp+12]      ; get num of elts
    mov     ebx, 0             ; initialize count
```

```
for_loop:
```

```
    mov     eax, [edi+4*ebx]    ; get array element
    inc     eax                 ; add 1
    mov     [edi+4*ebx], eax    ; put it back
    inc     ebx                 ; update counter
    loop    for_loop
```

```
    pop     edi                ; restore registers
    pop     ebx
```

```
    mov     esp, ebp           ; take down stack frame
    pop     ebp
```

```
    ret
```

```

// File: cfunc3.c
//
// Example of C function calls disassembled
// Return values with more than 4 bytes
//

#include <stdio.h>

typedef struct {
    int part1, part2 ;
} stype ;

// a silly function
//
stype foo(stype r) {

    r.part1 += 4;
    r.part2 += 3 ;
    return r ;
}

int main () {
    stype r1, r2, r3 ;
    int n ;

    n = 17 ;
    r1.part1 = 74 ;
    r1.part2 = 75 ;
    r2.part1 = 84 ;
    r2.part2 = 85 ;
    r3.part1 = 93 ;
    r3.part2 = 99 ;

    r2 = foo(r1) ;

    printf ("r2.part1 = %d, r2.part2 = %d\n",
        r1.part1, r2.part2 ) ;

    n = foo(r3).part2 ;
}

```

```

        ;FILE "cfunc3.c"
gcc2_compiled.:
SECTION .text
        ALIGN 4
GLOBAL foo
        GLOBAL foo:function

foo:
                                ; comments & spacing added
        push    ebp                ; set up stack frame
        mov     ebp,esp

        mov     eax,[ebp+8]        ; addr to store return value
        add     dword [ebp+12],4   ; r.part1 = [ebp+12]
        add     dword [ebp+16],3   ; r.part2 = [ebp+16]

        ; return value
        ;
        mov     edx,[ebp+12]       ; get r.part1
        mov     ecx,[ebp+16]       ; get r.part2
        mov     [eax],edx          ; put r.part1 in return value
        mov     [eax+4],ecx        ; put r.part2 in return value
        jmp     L1

L1:
        mov     eax,eax            ; does nothing
        leave   ; bye-bye
        ret     4                  ; pop 4 bytes after return

.Lfe1:

```

```

GLOBAL    foo:function (.Lfe1-foo)
SECTION   .rodata
.LC0:
    db      'r2.part1 = %d, r2.part2 = %d',10,''
SECTION   .text
    ALIGN 4
GLOBAL main
GLOBAL main:function
main:
    ; comments & spacing added
    push    ebp                ; set up stack frame
    mov     ebp,esp
    sub     esp,36             ; space for local variables

    ; initialize variables
    ;
    mov     dword [ebp-28],17   ; n = [ebp-28]
    mov     dword [ebp-8],74    ; r1 = [ebp-8]
    mov     dword [ebp-4],75
    mov     dword [ebp-16],84   ; r2 = [ebp-16]
    mov     dword [ebp-12],85
    mov     dword [ebp-24],93   ; r3 = [ebp-24]
    mov     dword [ebp-20],99

    ; call foo
    ;
    lea     eax, [ebp-16]       ; get addr of r2
    mov     edx, [ebp-8]        ; get r1.part1
    mov     ecx, [ebp-4]        ; get r1.part2
    push    ecx                 ; push r1.part2
    push    edx                 ; push r1.part1
    push    eax                 ; push addr of r2
    call    foo
    add     esp,8               ; pop r1
                                ; ret 4 popped r2's addr

    ; call printf
    ;
    mov     eax, [ebp-12]       ; get r2.part2
    push    eax                 ; push it
    mov     eax, [ebp-8]        ; get r2.part1
    push    eax                 ; push it
    push    dword .LC0          ; string constant's addr
    call    printf
    add     esp,12              ; pop off arguments

```

```

; call foo again
;
lea  eax, [ebp-36]          ; addr of temp variable
mov  edx, [ebp-24]          ; get r3.part1
mov  ecx, [ebp-20]          ; get r3.part2
push ecx                    ; push r3.part2
push edx                    ; push r3.part1
push eax                    ; push addr of temp var
call foo
add  esp,8                  ; pop off arguments

; assign to n
;
mov  eax, [ebp-32]          ; get part2 of temp var
mov  [ebp-28],eax           ; store in n

```

L2:

```

leave          ; bye-bye
ret

```

.Lfe2:

```

GLOBAL  main:function (.Lfe2-main)
;IDENT "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)"

```

FUNCTION POINTERS



DECLARING FUNCTION POINTERS

- Declaring a pointer to `int` value:

```
int *iptr ;
```

- Declaring a pointer to a function that takes an `int` parameter and returns an `int` value:

```
int (*fptr1) (int) ;
```

- Declaring a pointer to a function that takes two `int` parameters and returns an `int` value:

```
int (*fptr2) (int, int) ;
```

ASSIGNING VALUES TO FUNCTION POINTERS

- Assigning a value to an `int` pointer:

```
int a ;  
int *iptr = &a ;
```

- Assigning a value to a function pointer:

```
int add3(int) ;  
int sum (int, int) ;  
int (*fptr1) (int) ;  
int (*fptr2) (int, int) ;  
  
fptr1 = &add3 ;  
fptr2 = &sum ;
```

INVOKING FUNCTION POINTERS

```
int add3(int) ;  
int sum (int, int) ;  
int b ;
```

```
fptr1 = &add3 ;  
b = (*fptr1) (5) ;
```

```
fptr2 = &sum ;  
b = (*fptr2) (6, b) ;
```

```
/* File: funcptr1.c

   Demonstrating function pointers.
*/

#include <stdio.h>

int add3 (int n) {
    return n + 3 ;
}

int add5 (int n) {
    return n + 5 ;
}

int main() {

    int a, b ;
    int (* fptr) (int) ;

    a = 7 ;
    printf("a = %d\n", a) ;

    fptr = &add3 ;
    b = (*fptr) (a) ;
    printf("fptr(a) = %d\n", b ) ;

    fptr = &add5 ;
    b = (*fptr) (a) ;
    printf("fptr(a) = %d\n", b ) ;

    return 0 ;
}
```

Script started on Wed Oct 17 23:08:55 2012

River[1]% gcc -Wall funcptr1.c

River[2]% ./a.out

a = 7

fptr(a) = 10

fptr(a) = 12

River[3]%

River[3]% exit

exit

Script done on Wed Oct 17 23:09:11 2012

```

/* File: funcptr2.c

   Demonstrating function pointers.

*/

#include <stdio.h>

int add3 (int n) {
    return n + 3 ;
}

int add5 (int n) {
    return n + 5 ;
}

typedef int (* INT_INT_FPTR) (int) ;

void do_array(int A[], int size, INT_INT_FPTR fptr) {
    int i ;

    for ( i = 0 ; i < size ; i++) {
        A[i] = (* fptr) (A[i]) ;
    }
}

int main() {
    int A[10], i ;

    for (i = 0 ; i < 10 ; i++) {
        A[i] = i * i ;
    }

    printf("Original array A[]:\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d\n", i, A[i]) ;
    }

    do_array(A, 10, &add3) ;

    printf("\n\n After calling do_array(A, 10, &add3):\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d\n", i, A[i]) ;
    }

    do_array(A, 10, &add5) ;

    printf("\n\n After calling do_array(A, 10, &add5):\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d\n", i, A[i]) ;
    }

    return 0 ;
}

```

Script started on Wed Oct 17 23:09:20 2012

River[4]% gcc -Wall funcptr2.c

River[5]% ./a.out

Original array A[]:

A[0] = 0
A[1] = 1
A[2] = 4
A[3] = 9
A[4] = 16
A[5] = 25
A[6] = 36
A[7] = 49
A[8] = 64
A[9] = 81

After calling do_array(A, 10, &add3):

A[0] = 3
A[1] = 4
A[2] = 7
A[3] = 12
A[4] = 19
A[5] = 28
A[6] = 39
A[7] = 52
A[8] = 67
A[9] = 84

After calling do_array(A, 10, &add5):

A[0] = 8
A[1] = 9
A[2] = 12
A[3] = 17
A[4] = 24
A[5] = 33
A[6] = 44
A[7] = 57
A[8] = 72
A[9] = 89
River[6]%
River[6]% exit
exit

Script done on Wed Oct 17 23:09:33 2012

```

/* File: funcptr3.c

   Demonstrating function pointers.

*/

#include <stdio.h>

int diff (int m, int n) {
    return m - n ;
}

int sum (int m, int n) {
    return m + n ;
}

typedef int (* FPTR2) (int, int) ;

void do_array(int A[], int B[], int size, FPTR2 fptr) {
    int i ;

    for ( i = 0 ; i < size ; i++) {
        A[i] = (* fptr) (A[i], B[i]) ;
    }
}

int main() {

    int A[10], B[10], i ;

    for (i = 0 ; i < 10 ; i++) {
        A[i] = i * i ;
        B[i] = 2 * i ;
    }

    printf("Original arrays:\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d,    B[%d] = %d\n", i, A[i], i, B[i]) ;
    }

    do_array(A, B, 10, &diff) ;

    printf("\n\n After calling do_array(A, B, 10, &diff):\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d,    B[%d] = %d\n", i, A[i], i, B[i]) ;
    }

    do_array(A, B, 10, &sum) ;

    printf("\n\n After calling do_array(A, 10, &sum):\n") ;
    for (i = 0 ; i < 10 ; i++) {
        printf("A[%d] = %d,    B[%d] = %d\n", i, A[i], i, B[i]) ;
    }

    return 0 ;
}

```

```
Script started on Wed Oct 17 23:09:41 2012
River[7]% gcc -Wall funcptr3.c
River[8]%
```

```
River[8]% ./a.out
```

```
Original arrays:
```

```
A[0] = 0,    B[0] = 0
A[1] = 1,    B[1] = 2
A[2] = 4,    B[2] = 4
A[3] = 9,    B[3] = 6
A[4] = 16,   B[4] = 8
A[5] = 25,   B[5] = 10
A[6] = 36,   B[6] = 12
A[7] = 49,   B[7] = 14
A[8] = 64,   B[8] = 16
A[9] = 81,   B[9] = 18
```

```
After calling do_array(A, B, 10, &diff):
```

```
A[0] = 0,    B[0] = 0
A[1] = -1,   B[1] = 2
A[2] = 0,    B[2] = 4
A[3] = 3,    B[3] = 6
A[4] = 8,    B[4] = 8
A[5] = 15,   B[5] = 10
A[6] = 24,   B[6] = 12
A[7] = 35,   B[7] = 14
A[8] = 48,   B[8] = 16
A[9] = 63,   B[9] = 18
```

```
After calling do_array(A, 10, &sum):
```

```
A[0] = 0,    B[0] = 0
A[1] = 1,    B[1] = 2
A[2] = 4,    B[2] = 4
A[3] = 9,    B[3] = 6
A[4] = 16,   B[4] = 8
A[5] = 25,   B[5] = 10
A[6] = 36,   B[6] = 12
A[7] = 49,   B[7] = 14
A[8] = 64,   B[8] = 16
A[9] = 81,   B[9] = 18
```

```
River[9]%
```

```
River[9]% exit
exit
```

```
Script done on Wed Oct 17 23:09:55 2012
```

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.

CONFORMING TO

SVr4, 4.3BSD, C89, C99.

NOTES

Library routines suitable for use as the *compar* argument include **alphasort(3)** and **versionsort(3)**. To compare C strings, the comparison function can call **strcmp(3)**, as shown in the example below.

EXAMPLE

For one example of use, see the example under **bsearch(3)**.

Another example is the following program, which sorts the strings given in its command-line arguments:

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

static int
cmpstringp(const void *p1, const void *p2)
{
    /* The actual arguments to this function are "pointers to
       pointers to char", but strcmp(3) arguments are "pointers
       to char", hence the following cast plus dereference */

    return strcmp(*(char * const *) p1, *(char * const *) p2);
}

int
main(int argc, char *argv[])
{
    int j;

    assert(argc > 1);

    qsort(&argv[1], argc - 1, sizeof(char *), cmpstringp);
```

```
    for (j = 1; j < argc; j++)  
        puts(argv[j]);  
    exit(EXIT_SUCCESS);  
}
```

SEE ALSO

sort(1), alphasort(3), strcmp(3), versionsort(3)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at <http://www.kernel.org/doc/man-pages/>.

```

/* File: qsort1.c

   Demonstrating use of callback functions in qsort().
*/

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

typedef int (* COMP_FUNC_PTR) (const void *, const void *) ;

int lessthan (int *ptr1, int *ptr2) {
    return *ptr1 - *ptr2 ;
}

int main() {
    int A[25] = { 2, 7, 91, 23, 14, 72, 19, 31, 44, 62,
                  34, 11, 51, 62, 22, 81, 45, 54, 67, 74,
                  24, 15, 41, 83, 88} ;

    int i ;

    printf("Original array:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        printf("  A[%d] = %d\n", i, A[i]) ;
    }

    qsort(A, 25, sizeof(int), (COMP_FUNC_PTR) &lessthan) ;

    printf("\n\nSorted array:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        printf("  A[%d] = %d\n", i, A[i]) ;
    }

    return 0 ;
}

```

Script started on Wed Oct 17 23:14:12 2012

River[10]% gcc qsort1.c

River[11]%

River[11]% ./a.out

Original array:

```
A[0] = 2
A[1] = 7
A[2] = 91
A[3] = 23
A[4] = 14
A[5] = 72
A[6] = 19
A[7] = 31
A[8] = 44
A[9] = 62
A[10] = 34
A[11] = 11
A[12] = 51
A[13] = 62
A[14] = 22
A[15] = 81
A[16] = 45
A[17] = 54
A[18] = 67
A[19] = 74
A[20] = 24
A[21] = 15
A[22] = 41
A[23] = 83
A[24] = 88
```

Sorted array:

```
A[0] = 2
A[1] = 7
A[2] = 11
A[3] = 14
A[4] = 15
A[5] = 19
A[6] = 22
A[7] = 23
A[8] = 24
A[9] = 31
A[10] = 34
A[11] = 41
A[12] = 44
A[13] = 45
A[14] = 51
A[15] = 54
A[16] = 62
A[17] = 62
A[18] = 67
A[19] = 72
A[20] = 74
A[21] = 81
A[22] = 83
A[23] = 88
A[24] = 91
```

River[12]%

```
/* File: qsort2.c
```

```
    Demonstrating use of callback functions in qsort().  
    This time use bigger data sizes.
```

```
*/
```

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

```
typedef int (* COMP_FUNC_PTR) (const void *, const void *) ;
```

```
typedef struct {  
    char    realname[32] ;  
    char    nickname[16] ;  
    char    alignment[20] ;  
    char    role[20] ;  
    int     points ;  
    int     level ;
```

```
} Player ;
```

```
Player records[25] = {  
    {"Allen, Kevin P.", "kallen1", "neutral good", "Paladin", 6721, 6},  
    {"Baker, Matthew R.", "mbaker5", "true neutral", "Thief", 4313, 4},  
    {"Chandlee, Richard C.", "rchand1", "lawful evil", "Ranger", 3196, 3},  
    {"Cloud, Clinton E.", "ccloud2", "chaotic good", "Magician", 9583, 9},  
    {"Cobb, Milton T.", "cobb", "lawful neutral", "Fighter", 9169, 9},  
    {"Frankle, Alan E.", "frankle1", "chaotic good", "Fighter", 8924, 9},  
    {"Fu, David E.", "dfu1", "chaotic good", "Ranger", 10011, 12},  
    {"Helton, Robert L.", "helton1", "true neutral", "Paladin", 3034, 3},  
    {"Joshua, Aimee", "ajoshu1", "chaotic evil", "Thief", 3733, 4},  
    {"Macdonald, Matthew S.", "matmacd1", "chaotic good", "Fighter", 3797, 5},  
    {"Mcelvaney, Erin L.", "emcelv1", "lawful neutral", "Cleric", 9896, 11},  
    {"Mitchell, Susan M.", "smitchel", "neutral good", "Fighter", 9740, 10},  
    {"Mittal, Sandeep K.", "smittal", "chaotic good", "Cleric", 2706, 2},  
    {"Moore, Brian A.", "bmoore1", "lawful evil", "Ranger", 4016, 4},  
    {"Napier, Matthew A.", "mnapie1", "true neutral", "Magician", 4539, 5},  
    {"Nguyen, Michael D.", "mnguyen1", "chaotic good", "Ranger", 9490, 9},  
    {"Orticke, Alecia G.", "aortic1", "chaotic good", "Cleric", 2048, 4},  
    {"Palewicz, David E.", "dpalew1", "chaotic good", "Thief", 765, 3},  
    {"Raby, Adam M.", "araby1", "lawful neutral", "Magician", 8814, 9},  
    {"Roberts, Eric J.", "erob3", "neutral good", "Fighter", 7765, 7},  
    {"Rusinko, Nicholas D.", "nrusin1", "lawful good", "Thief", 5095, 5},  
    {"Seo, Maximilian", "mseol", "chaotic evil", "Ranger", 3205, 3},  
    {"Shenvi, Shilpa P.", "shenvil", "neutral good", "Thief", 9573, 10},  
    {"Szrom, Michelle L.", "mszrom1", "chaotic good", "Cleric", 9660, 9},  
    {"Tuveson, Eric A.", "etuves1", "chaotic evil", "Thief", 9245, 7}  
} ;
```

```
void PrintPlayer(Player *p) {
```

```
    printf("%32s (%16s)  %20s  %20s(%3d)  %10d\n",  
        p->realname, p->nickname, p->alignment, p->role, p->level,  
        p->points) ;
```

```
}
```

```

int byPoints(Player *p1, Player *p2) {
    return p1->points - p2->points ;
}

int byNickname(Player *p1, Player *p2) {
    return strcmp(p1->nickname, p2->nickname) ;
}

int byAlignment(Player *p1, Player *p2) {
    return strcmp(p1->alignment, p2->alignment) ;
}

int main () {

    int i ;

    printf("Original list:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byPoints) ;

    printf("\n\nSorted by points:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byNickname) ;

    printf("\n\nSorted by nickname:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byAlignment) ;

    printf("\n\nSorted by alignment:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    return 0 ;
}

```

```
/* File: qsort2.c
```

```
    Demonstrating use of callback functions in qsort().  
    This time use bigger data sizes.
```

```
*/
```

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

```
typedef int (* COMP_FUNC_PTR) (const void *, const void *) ;
```

```
typedef struct {  
    char    realname[32] ;  
    char    nickname[16] ;  
    char    alignment[20] ;  
    char    role[20] ;  
    int     points ;  
    int     level ;
```

```
} Player ;
```

```
Player records[25] = {  
    {"Allen, Kevin P.", "kallen1", "neutral good", "Paladin", 6721, 6},  
    {"Baker, Matthew R.", "mbaker5", "true neutral", "Thief", 4313, 4},  
    {"Chandlee, Richard C.", "rchand1", "lawful evil", "Ranger", 3196, 3},  
    {"Cloud, Clinton E.", "ccloud2", "chaotic good", "Magician", 9583, 9},  
    {"Cobb, Milton T.", "cobb", "lawful neutral", "Fighter", 9169, 9},  
    {"Frankle, Alan E.", "frankle1", "chaotic good", "Fighter", 8924, 9},  
    {"Fu, David E.", "dfu1", "chaotic good", "Ranger", 10011, 12},  
    {"Helton, Robert L.", "helton1", "true neutral", "Paladin", 3034, 3},  
    {"Joshua, Aimee", "ajoshu1", "chaotic evil", "Thief", 3733, 4},  
    {"Macdonald, Matthew S.", "matmacd1", "chaotic good", "Fighter", 3797, 5},  
    {"Mcelvaney, Erin L.", "emcelv1", "lawful neutral", "Cleric", 9896, 11},  
    {"Mitchell, Susan M.", "smitchel", "neutral good", "Fighter", 9740, 10},  
    {"Mittal, Sandeep K.", "smittal", "chaotic good", "Cleric", 2706, 2},  
    {"Moore, Brian A.", "bmoore1", "lawful evil", "Ranger", 4016, 4},  
    {"Napier, Matthew A.", "mnapie1", "true neutral", "Magician", 4539, 5},  
    {"Nguyen, Michael D.", "mnguyen1", "chaotic good", "Ranger", 9490, 9},  
    {"Orticke, Alecia G.", "aortic1", "chaotic good", "Cleric", 2048, 4},  
    {"Palewicz, David E.", "dpalew1", "chaotic good", "Thief", 765, 3},  
    {"Raby, Adam M.", "araby1", "lawful neutral", "Magician", 8814, 9},  
    {"Roberts, Eric J.", "erober3", "neutral good", "Fighter", 7765, 7},  
    {"Rusinko, Nicholas D.", "nrusin1", "lawful good", "Thief", 5095, 5},  
    {"Seo, Maximilian", "mseol", "chaotic evil", "Ranger", 3205, 3},  
    {"Shenvi, Shilpa P.", "shenvil", "neutral good", "Thief", 9573, 10},  
    {"Szrom, Michelle L.", "mszrom1", "chaotic good", "Cleric", 9660, 9},  
    {"Tuveson, Eric A.", "etuves1", "chaotic evil", "Thief", 9245, 7}  
} ;
```

```
void PrintPlayer(Player *p) {
```

```
    printf("%32s (%16s)  %20s  %20s(%3d)  %10d\n",  
        p->realname, p->nickname, p->alignment, p->role, p->level,  
        p->points) ;
```

```
}
```

```

int byPoints(Player *p1, Player *p2) {
    return p1->points - p2->points ;
}

int byNickname(Player *p1, Player *p2) {
    return strcmp(p1->nickname, p2->nickname) ;
}

int byAlignment(Player *p1, Player *p2) {
    return strcmp(p1->alignment, p2->alignment) ;
}

int main () {

    int i ;

    printf("Original list:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byPoints) ;

    printf("\n\nSorted by points:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byNickname) ;

    printf("\n\nSorted by nickname:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    qsort(records, 25, sizeof(Player), (COMP_FUNC_PTR) &byAlignment) ;

    printf("\n\nSorted by alignment:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    return 0 ;
}

```

Script started on Wed Oct 17 23:14:26 2012

River[13]% gcc -Wall qsort2.c

River[14]%

River[14]% ./a.out

Original list:

Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Joshua, Aimee (	ajoshu1)	chaotic evil	Thief(4)	3733
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245

Sorted by points:

Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Joshua, Aimee (	ajoshu1)	chaotic evil	Thief(4)	3733
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011

Sorted by nickname:

Joshua, Aimee (	ajoshul)	chaotic evil	Thief(4)	3733
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Napier, Matthew A. (	mnapiel)	true neutral	Magician(5)	4539
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706

Sorted by alignment:

Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Joshua, Aimee (	ajoshul)	chaotic evil	Thief(4)	3733
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Napier, Matthew A. (	mnapiel)	true neutral	Magician(5)	4539

River[15]%

River[15]% exit
exit

```
/* File: qsort3.c
```

```
    Demonstrating use of callback functions in qsort().  
    This time we sort an array of pointers.
```

```
*/
```

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

```
typedef int (* COMP_FUNC_PTR) (const void *, const void *) ;
```

```
typedef struct {  
    char    realname[32] ;  
    char    nickname[16] ;  
    char    alignment[20] ;  
    char    role[20] ;  
    int     points ;  
    int     level ;
```

```
} Player ;
```

```
Player records[25] = {  
    {"Allen, Kevin P.", "kallen1", "neutral good", "Paladin", 6721, 6},  
    {"Baker, Matthew R.", "mbaker5", "true neutral", "Thief", 4313, 4},  
    {"Chandlee, Richard C.", "rchand1", "lawful evil", "Ranger", 3196, 3},  
    {"Cloud, Clinton E.", "ccloud2", "chaotic good", "Magician", 9583, 9},  
    {"Cobb, Milton T.", "cobb", "lawful neutral", "Fighter", 9169, 9},  
    {"Frankle, Alan E.", "frankle1", "chaotic good", "Fighter", 8924, 9},  
    {"Fu, David E.", "dful", "chaotic good", "Ranger", 10011, 12},  
    {"Helton, Robert L.", "helton1", "true neutral", "Paladin", 3034, 3},  
    {"Joshua, Aimee", "ajoshu1", "chaotic evil", "Thief", 3733, 4},  
    {"Macdonald, Matthew S.", "matmacd1", "chaotic good", "Fighter", 3797, 5},  
    {"Mcelvaney, Erin L.", "emcelv1", "lawful neutral", "Cleric", 9896, 11},  
    {"Mitchell, Susan M.", "smitchel", "neutral good", "Fighter", 9740, 10},  
    {"Mittal, Sandeep K.", "smittal", "chaotic good", "Cleric", 2706, 2},  
    {"Moore, Brian A.", "bmoore1", "lawful evil", "Ranger", 4016, 4},  
    {"Napier, Matthew A.", "mnapie1", "true neutral", "Magician", 4539, 5},  
    {"Nguyen, Michael D.", "mnguyen1", "chaotic good", "Ranger", 9490, 9},  
    {"Orticke, Alecia G.", "aortic1", "chaotic good", "Cleric", 2048, 4},  
    {"Palewicz, David E.", "dpalew1", "chaotic good", "Thief", 765, 3},  
    {"Raby, Adam M.", "araby1", "lawful neutral", "Magician", 8814, 9},  
    {"Roberts, Eric J.", "erober3", "neutral good", "Fighter", 7765, 7},  
    {"Rusinko, Nicholas D.", "nrusin1", "lawful good", "Thief", 5095, 5},  
    {"Seo, Maximilian", "mseol", "chaotic evil", "Ranger", 3205, 3},  
    {"Shenvi, Shilpa P.", "shenvil", "neutral good", "Thief", 9573, 10},  
    {"Szrom, Michelle L.", "mszrom1", "chaotic good", "Cleric", 9660, 9},  
    {"Tuveson, Eric A.", "etuves1", "chaotic evil", "Thief", 9245, 7}  
} ;
```

```
void PrintPlayer(Player *p) {
```

```
    printf("%32s (%16s)  %20s  %20s(%3d)  %10d\n",  
        p->realname, p->nickname, p->alignment, p->role, p->level,  
        p->points) ;
```

```
}
```

```

int byPoints(Player **p1, Player **p2) {
    return (*p1)->points - (*p2)->points ;
}

int byNickname(Player **p1, Player **p2) {
    return strcmp((*p1)->nickname, (*p2)->nickname) ;
}

int byAlignment(Player **p1, Player **p2) {
    return strcmp((*p1)->alignment, (*p2)->alignment) ;
}

int main () {

    int i ;
    Player *rec_ptrs[25] ;

    for (i = 0 ; i < 25 ; i++) {
        rec_ptrs[i] = &records[i] ;
    }

    printf("Original list:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(rec_ptrs[i]) ;
    }

    qsort(rec_ptrs, 25, sizeof(Player *), (COMP_FUNC_PTR) &byPoints) ;

    printf("\n\nSorted by points:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(rec_ptrs[i]) ;
    }

    qsort(rec_ptrs, 25, sizeof(Player *), (COMP_FUNC_PTR) &byNickname) ;

    printf("\n\nSorted by nickname:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(rec_ptrs[i]) ;
    }

    qsort(rec_ptrs, 25, sizeof(Player *), (COMP_FUNC_PTR) &byAlignment) ;

    printf("\n\nSorted by alignment:\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(rec_ptrs[i]) ;
    }

    printf("\n\nOriginal list (again):\n") ;
    for (i = 0 ; i < 25 ; i++) {
        PrintPlayer(&records[i]) ;
    }

    return 0 ;
}

```

Script started on Wed Oct 17 23:15:52 2012

River[16]% gcc -Wall qsort3.c

River[17]%

River[17]% ./a.out

Original list:

Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Joshua, Aimee (	ajoshu1)	chaotic evil	Thief(4)	3733
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245

Sorted by points:

Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Joshua, Aimee (	ajoshu1)	chaotic evil	Thief(4)	3733
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011

Sorted by nickname:

Joshua, Aimee (	ajoshul)	chaotic evil	Thief(4)	3733
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706

Sorted by alignment:

Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245
Seo, Maximilian (	mseol)	chaotic evil	Ranger(3)	3205
Joshua, Aimee (	ajoshul)	chaotic evil	Thief(4)	3733
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Napier, Matthew A. (	mnapie1)	true neutral	Magician(5)	4539

Original list (again):

Allen, Kevin P. (	kallen1)	neutral good	Paladin(6)	6721
Baker, Matthew R. (	mbaker5)	true neutral	Thief(4)	4313
Chandlee, Richard C. (	rchand1)	lawful evil	Ranger(3)	3196
Cloud, Clinton E. (	ccloud2)	chaotic good	Magician(9)	9583
Cobb, Milton T. (	cobb)	lawful neutral	Fighter(9)	9169
Frankle, Alan E. (	frankle1)	chaotic good	Fighter(9)	8924
Fu, David E. (	dful)	chaotic good	Ranger(12)	10011
Helton, Robert L. (	helton1)	true neutral	Paladin(3)	3034
Joshua, Aimee (	ajoshu1)	chaotic evil	Thief(4)	3733
Macdonald, Matthew S. (	matmacd1)	chaotic good	Fighter(5)	3797
Mcelvaney, Erin L. (	emcelv1)	lawful neutral	Cleric(11)	9896
Mitchell, Susan M. (	smitchel)	neutral good	Fighter(10)	9740
Mittal, Sandeep K. (	smittal)	chaotic good	Cleric(2)	2706
Moore, Brian A. (	bmoore1)	lawful evil	Ranger(4)	4016
Napier, Matthew A. (	mnapiel)	true neutral	Magician(5)	4539
Nguyen, Michael D. (	mnguyen1)	chaotic good	Ranger(9)	9490
Orticke, Alecia G. (	aortic1)	chaotic good	Cleric(4)	2048
Palewicz, David E. (	dpalew1)	chaotic good	Thief(3)	765
Raby, Adam M. (	araby1)	lawful neutral	Magician(9)	8814
Roberts, Eric J. (	erober3)	neutral good	Fighter(7)	7765
Rusinko, Nicholas D. (	nrusin1)	lawful good	Thief(5)	5095
Seo, Maximilian (	mseo1)	chaotic evil	Ranger(3)	3205
Shenvi, Shilpa P. (	shenvil)	neutral good	Thief(10)	9573
Szrom, Michelle L. (	mszrom1)	chaotic good	Cleric(9)	9660
Tuveson, Eric A. (	etuves1)	chaotic evil	Thief(7)	9245

River[18]%
River[18]% exit
exit

Script done on Wed Oct 17 23:16:03 2012

NEXT TIME

- Polymorphism in C?

