Using Memory

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

struct foo_struct {
  int x;
  char a; char b; char c; char d;
};

int
main()
{
   int i;
   char a[40];
   int *iptr = (int *) a;
   struct foo_struct *sptr = (struct foo_struct *) a;
```


Using Memory

```
   for (i=0; i<40; i++) {
     a[i] = (char) i;
   }

   for (i=0; i<10; i++) {
     printf("%2d = 0x%08x\n", i, iptr[i]);
   }

   printf("x = 0x%08x  a = %d  b = %d c = %d d = %d\n",
     sptr[0].x, (int) sptr[0].a, (int) sptr[0].b,
     (int) sptr[0].c, (int) sptr[0].d);

   exit(0);
}
```

Using Memory: Example Output (x86)

```
 0 = 0x03020100
 1 = 0x07060504
 2 = 0x0b0a0908
 3 = 0x0f0e0d0c
 4 = 0x13121110
 5 = 0x17161514
 6 = 0x1b1a1918
 7 = 0x1f1e1d1c
 8 = 0x23222120
 9 = 0x27262524
x = 0x03020100  a = 4  b = 5 c = 6 d = 7
```


Using Memory: Example Output (Sparc)

```
 0 = 0x00010203
 1 = 0x04050607
 2 = 0x08090a0b
 3 = 0x0c0d0e0f
 4 = 0x10111213
 5 = 0x14151617
 6 = 0x18191a1b
 7 = 0x1c1d1e1f
 8 = 0x20212223
 9 = 0x24252627
x = 0x00010203  a = 4  b = 5 c = 6 d = 7
```

Writing to a File

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int
main()
{
   int i, rc, fd;
   int array[40];
```


Writing to a File

```
   for (i=0; i<40; i++) {
     array[i] = i;
   }

   fd = open("test-output", O_WRONLY | O_CREAT, S_IRWXU);

   if (fd < 0) {
     perror("open failed");
     exit(1);
   }
```

Writing to a File

```
    rc = write(fd, array, sizeof(array));

    if (rc < 0) {
      perror("write failed");
      exit(1);
    }

    close(fd);
    exit(0);
}
```


Writing to a File: Example Output

```
% cat test-output
#@u
   !$%
```

Writing to a File: Example Output (x86)

```
# Print offsets and values in Hex (x)
xxd test-output
0000000: 0000 0000 0100 0000 0200 0000 0300 0000  .........
0000010: 0400 0000 0500 0000 0600 0000 0700 0000  .........
0000020: 0800 0000 0900 0000 0a00 0000 0b00 0000  .........
0000030: 0c00 0000 0d00 0000 0e00 0000 0f00 0000  .........
0000040: 1000 0000 1100 0000 1200 0000 1300 0000  .........
0000050: 1400 0000 1500 0000 1600 0000 1700 0000  .........
0000060: 1800 0000 1900 0000 1a00 0000 1b00 0000  .........
0000070: 1c00 0000 1d00 0000 1e00 0000 1f00 0000  .........
0000080: 2000 0000 2100 0000 2200 0000 2300 0000   ...!..."
0000090: 2400 0000 2500 0000 2600 0000 2700 0000  $...%...&
```


Writing to a File: Example Output (Sparc)

```
# Print offsets and values in Hex (x)
 xxd test-output
0000000: 0000 0000 0000 0001 0000 0002 0000 0003  .........
0000010: 0000 0004 0000 0005 0000 0006 0000 0007  .........
0000020: 0000 0008 0000 0009 0000 000a 0000 000b  .........
0000030: 0000 000c 0000 000d 0000 000e 0000 000f  .........
0000040: 0000 0010 0000 0011 0000 0012 0000 0013  .........
0000050: 0000 0014 0000 0015 0000 0016 0000 0017  .........
0000060: 0000 0018 0000 0019 0000 001a 0000 001b  .........
0000070: 0000 001c 0000 001d 0000 001e 0000 001f  .........
0000080: 0000 0020 0000 0021 0000 0022 0000 0023  ... ...!.
0000090: 0000 0024 0000 0025 0000 0026 0000 0027  ...$...%.
```

Arrays and Addresses

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

static char *alpha = "abcdefghijklmnopqrstuvwxyz";

int
main()
{
   char a[12];
   char *b = 0;
   int i;
```


Arrays and Addresses

```
   for (i=0; i<12; i++) {
     a[i] = alpha[i];
   }

   printf("addr of a = %p\n", &a);
   printf("addr of a[0] = %p\n", &a[0]);
   printf("*a = %c\n", *a);
   printf("addr of b = %p\n", &b);
   printf("addr of b[0] = %p\n", &b[0]);
   printf("b = %p\n", b);
   printf("\n");
```

Arrays and Addresses

```
   b = a;
   printf("addr of b = %p\n", &b);
   printf("addr of b[0] = %p\n", &b[0]);
   printf("b = %p\n", b);
   printf("*b = %c\n", *b);
   printf("\n");

   b = &a[4];
   printf("addr of b = %p\n", &b);
   printf("addr of b[0] = %p\n", &b[0]);
   printf("b = %p\n", b);
   printf("*b = %c\n", *b);
   printf("\n");

   exit(0);
}
```


Arrays and Addresses: Example Output (x86)

```
addr of a = 0xbfe79b88
addr of a[0] = 0xbfe79b88
*a = a
addr of b = 0xbfe79b84
addr of b[0] = (nil)
b = (nil)

addr of b = 0xbfe79b84
addr of b[0] = 0xbfe79b88
b = 0xbfe79b88
*b = a

addr of b = 0xbfe79b84
addr of b[0] = 0xbfe79b8c
b = 0xbfe79b8c
*b = e
```