

```
int x;
int main()
{
    x = 5;
    printf("%d\n", x);
    mysub();
}

void mysub(void)
{
    int x;
    printf("%d\n", x);
}

main.c" 26L, 220C written
pod1-1 14 % gcc main.c
pod1-1 15 % ./a.out
5
32647
pod1-1 16 % !vim
```

uninitialized  
variable  
will not always  
be 0.

```

// static variables: persistence of memory across function calls
#include <stdio.h>

int func1(void);
int main() {
    int x;
    x = func1();
    printf("%d\n", x);
    x = func1();
    printf("%d\n", x);
    x = func1();
    printf("%d\n", x);
    return 0;
}

int func1() {
    static int a = 1;
    return a++;
}

```

static variable  
can only be  
initialized once! [L]

```

int x;
x = func1();
printf("%d\n", x);
x = func1();
printf("%d\n", x);
x = func1();
printf("%d\n", x);
return 0;
}

int func1() {
    static int a = 1;
    return a++;
}

```

`static int a = 1;`  
 is located in data (global variable)  
 it's always be there  
 but, it's still local to his own  
 scope within his curly bracket

```

// Pointers of functions and passing them as
#include <stdio.h>

int twice(int);
int runfunc(int (*g)(), int);

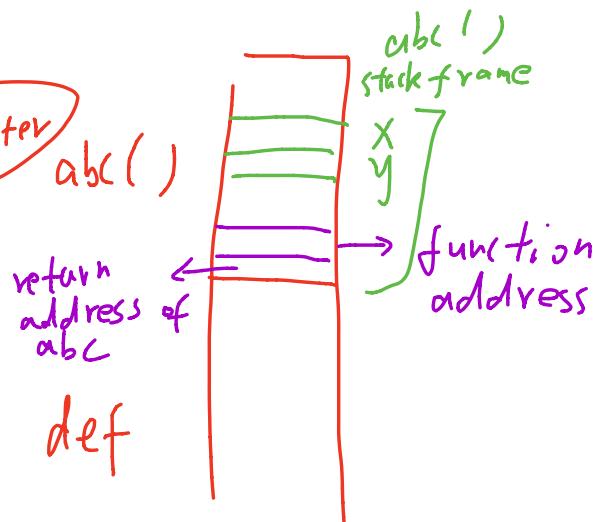
int main()
{
    int x, y;
    int (*f)(int);
    f = &twice; // can write as
    x = 3;
    y = twice(x);
    printf("result = %d\n", y);

    f = &twice; // can write as
    x = 4;
    y = (*f)(x);
    printf("result = %d\n", y);

    x = 5;
    y = runfunc(&twice, x);
    printf("result = %d\n", y);
}

// multiply by 2
int twice(int a)
{
    return 2*a;
}

```



```

printf("result = %d\n", y);

f = &twice;
x = 4;
y = (*f)(x);
printf("result = %d\n", y);

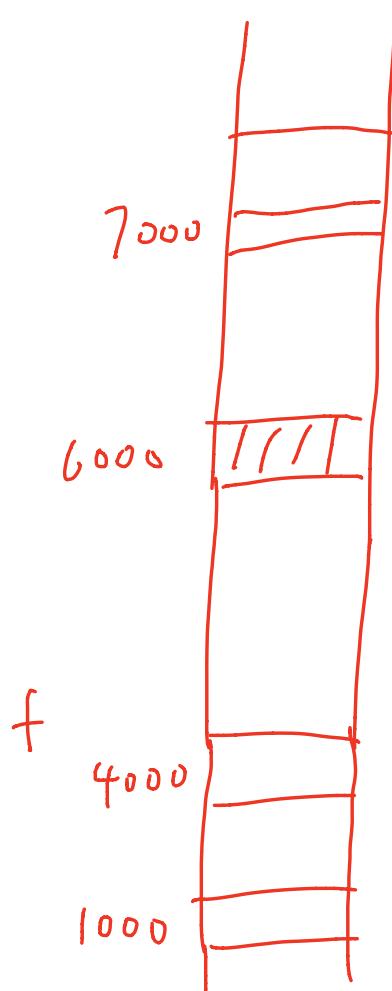
x = 5;
y = runfunc(&twice, x);
printf("result = %d\n", y);

}

// multiply by 2
int twice(int a)
{
    return 2*a;
}

// run function with a single argument
int runfunc(int (*h)(), int a)
{
    int b;
    b = (*h)(a); // can write as
    return b;
}

```

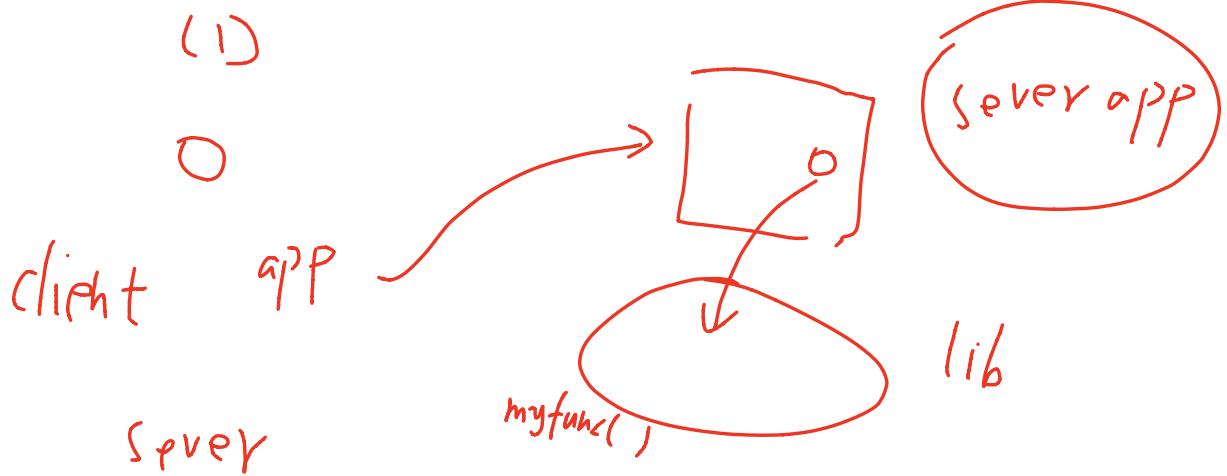


```

// run function with a single argument
int runfunc(int (*h)(int), int a)
{
    int b;
    b = (*h)(a);
    return b;
}

```

## Callback function



If the client connect me, the tell  
the client to run this function in lib

## (2) Segmentation fault

If seg fault happen, don't kill me,  
tell linux,  
run this function. signal handler.

## Lab 4

128.10.22.27      dotted decimal  
↓  
purdue

32 bit = 4 byte

(ls) legacy

Address      |      int X → variable  
Label :      function name  
              main()  
              abc()  
              def()

abc( ) {

    def(); → jump to def  
    }

abc( ) {

    xyz(&def)  
        ↓  
        optional

}

xyz(int (\*fp)())

int def();

# Bit Processing

```

// Program to inspect the 32 bits of unsigned int number
#include <stdio.h>
int main()
{
    int i;
    unsigned int x, y, m;
    // read input
    scanf("%u", &x);
    // set mask to 00 ... 01 0000...0001
    m = ~(-0 << 1);
    // loop over all 32 bits from lsb to msb
    for(i=0; i<32; i++) {
        // zero out all bits but for lsb
        y = x & m;
        printf("%u\n", y);
        // right shift to inspect the next significant bit
        x = x >> 1;
    }
}
"main1.c" 26L, 411C

```

most significant

positive 0 ... 0 & 0 = 0  
negative 1 ... 0 & 0 = 0  
 $2^{31} \sim -2^{31}$  0 & 1 = 0  
 $2^{32}$ , all positive 1 & 0 = 0  
 $\sim(0 << 1)$  1 & 1 = 1

kill any bit except the first bit (least significant bit)

0 OR 0 = 0  
0 OR 1 = 1  
1 OR 0 = 1  
1 OR 1 = 1

EXOR

$$0 \quad 0 = 0$$

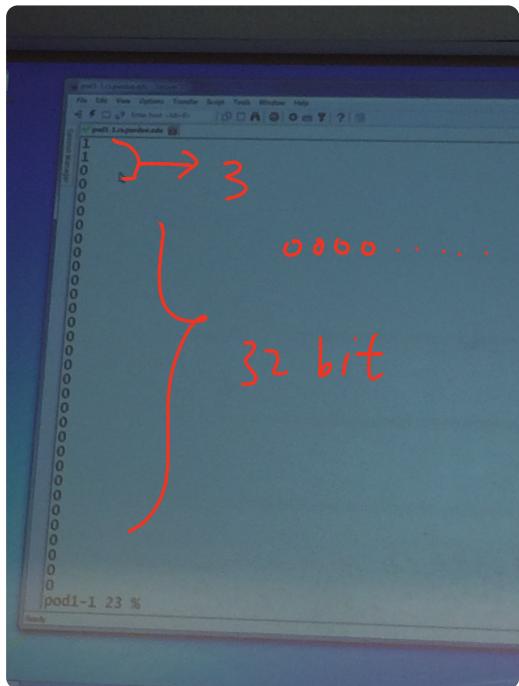
$$\text{Not } 0 = 1$$

$$0 \quad 1 = 1$$

$$\text{Not } 1 = 0$$

$$1 \quad 0 = 1$$

$$1 \quad 1 = 0$$



Input data      int  $X$  ;

mask      int  $m$  ;

Output data      int  $y$  ;

most significant



32 bit

least significant



$$X_0 = ?$$

$X$ :	$x_{31}$	$x_{30}$	$x_{29}$	...	$x_2$	$x_1$	$x_0$
$m$	0	0	0		0	0	1
$y$ :	0	0	0		0	0	$x_0$

$x_0$  And 1 is what  $x_0$  is  
left

what's  $x_1 = ?$

1. shift  $X$  to right for 1 bit

You will last for ever

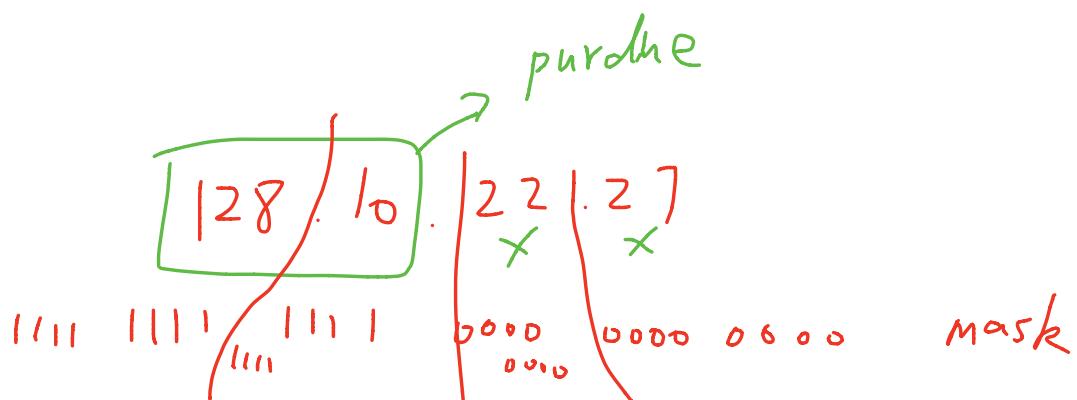
$$Z : \rightarrow \cup\ X_{31}\ X_{30}\ X_{29}\ \dots\ X_3\ X_2\ X_1$$

mask: 0 0 0 0 ... 0 0 |

$$y: \quad 6 \quad 0 \quad 0 \quad 0 \quad \quad \quad 0 \quad 0 \quad x,$$

→ with the same mask

$$z = x \gg 1$$



$$= 128,10,0,0$$

