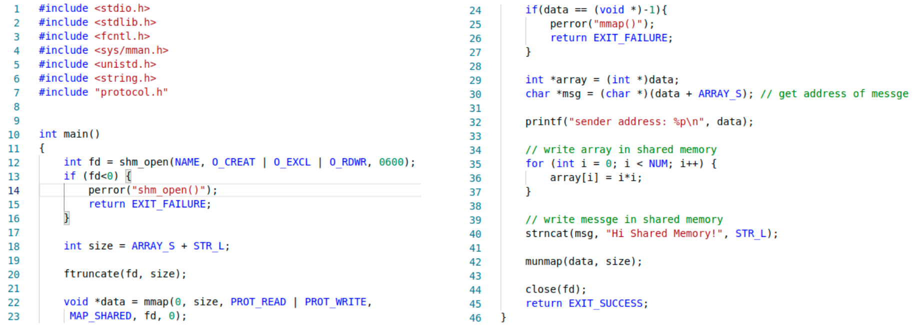
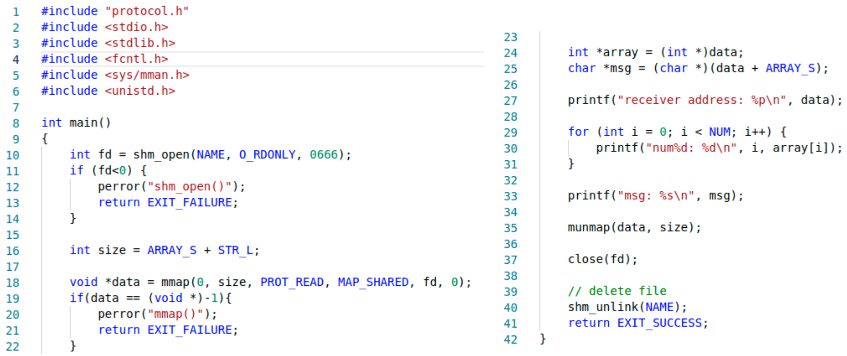
sender.c



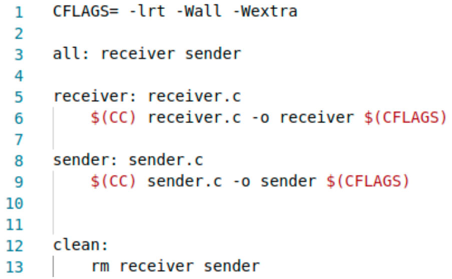
```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <string.h>
#include "protocol.h"


int main()
{
    int fd = shm_open(NAME, O_CREAT | O_EXCL | O_RDWR, 0600);
    if (fd<0) {
        perror("shm_open()");
        return EXIT_FAILURE;
    }

    int size = ARRAY_S + STR_L;

    ftruncate(fd, size);

    void *data = mmap(0, size, PROT_READ | PROT_WRITE,
     MAP_SHARED, fd, 0);
    if(data == (void *)-1){
        perror("mmap()");
        return EXIT_FAILURE;
    }

    int *array = (int *)data;
    char *msg = (char *)(data + ARRAY_S); // get address of messge

    printf("sender address: %p\n", data);

    // write array in shared memory
    for (int i = 0; i < NUM; i++) {
        array[i] = i*i;
    }

    // write messge in shared memory
    strncat(msg, "Hi Shared Memory!", STR_L);

    munmap(data, size);

    close(fd);
    return EXIT_SUCCESS;
}
```

receiver.c

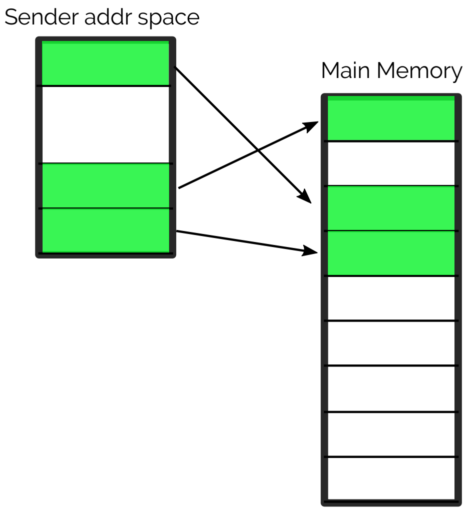


```c
#include "protocol.h"
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>

int main()
{
    int fd = shm_open(NAME, O_RDONLY, 0666);
    if (fd<0) {
        perror("shm_open()");
        return EXIT_FAILURE;
    }

    int size = ARRAY_S + STR_L;

    void *data = mmap(0, size, PROT_READ, MAP_SHARED, fd, 0);
    if(data == (void *)-1){
        perror("mmap()");
        return EXIT_FAILURE;
    }

    int *array = (int *)data;
    char *msg = (char *)(data + ARRAY_S);

    printf("receiver address: %p\n", data);

    for (int i = 0; i < NUM; i++) {
        printf("num%d: %d\n", i, array[i]);
    }

    printf("msg: %s\n", msg);

    munmap(data, size);

    close(fd);

    // delete file
    shm_unlink(NAME);
    return EXIT_SUCCESS;
}
```

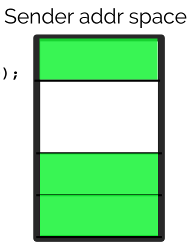
makefile

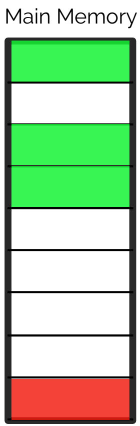


```
CFLAGS= -lrt -Wall -Wextra

all: receiver sender

receiver: receiver.c
	$(CC) receiver.c -o receiver $(CFLAGS)

sender: sender.c
	$(CC) sender.c -o sender $(CFLAGS)


clean:
	rm receiver sender
```

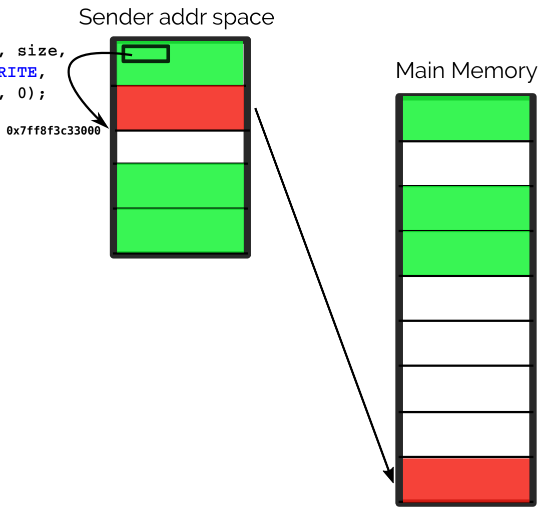

Sender addr space
Main Memory

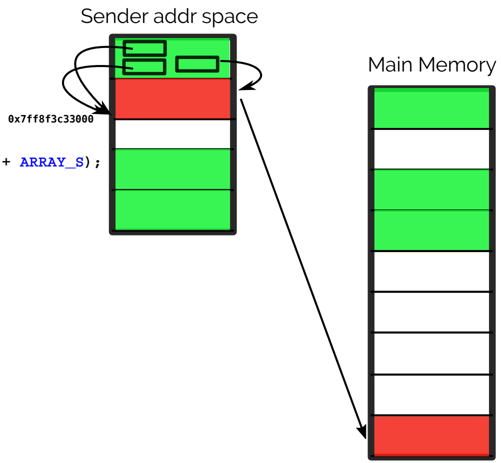
```
int fd = shm_open(NAME,
 O_CREAT | O_EXCL | O_RDWR, 0600);
    if (fd<0) {
        perror("shm_open()");
        return EXIT_FAILURE;
    }

    int size = ARRAY_S + STR_L;

    ftruncate(fd, size);
```





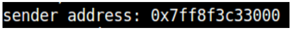
```
void *data = mmap(0, size,
 PROT_READ | PROT_WRITE,
     MAP_SHARED, fd, 0);
```





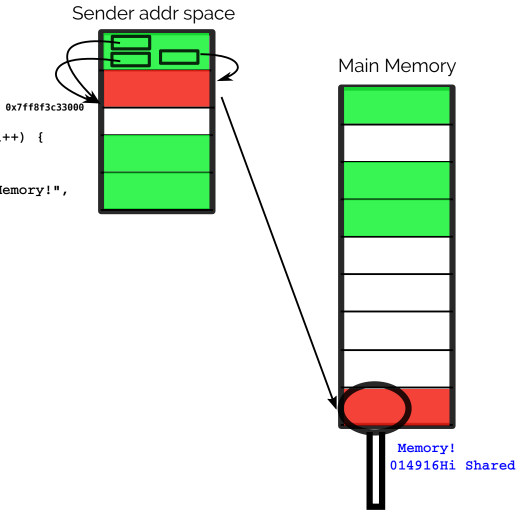
```
int *array = (int *)data;
char *msg = (char *)(data + ARRAY_S);
```

```c
printf("sender address: %p\n", data);
```

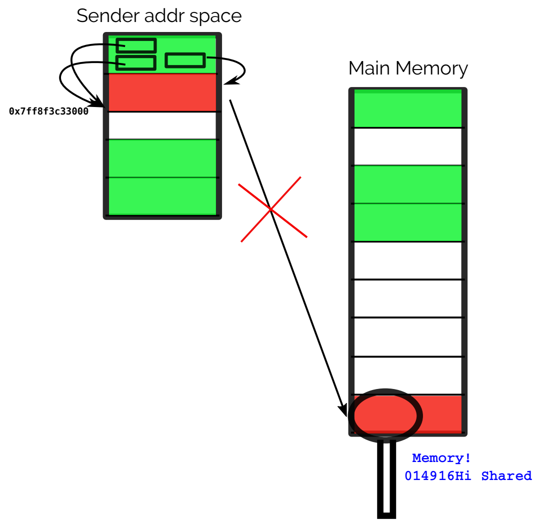


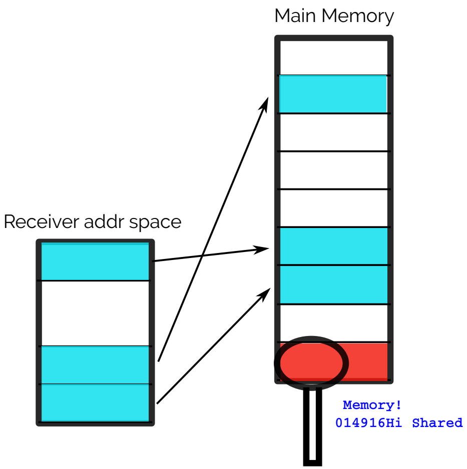
```
sender address: 0x7ff8f3c33000
```

```
for (int i = 0; i < NUM; i++) {
        array[i] = i*i;
    }
 strncat(msg, "Hi Shared Memory!",
 STR_L);
```

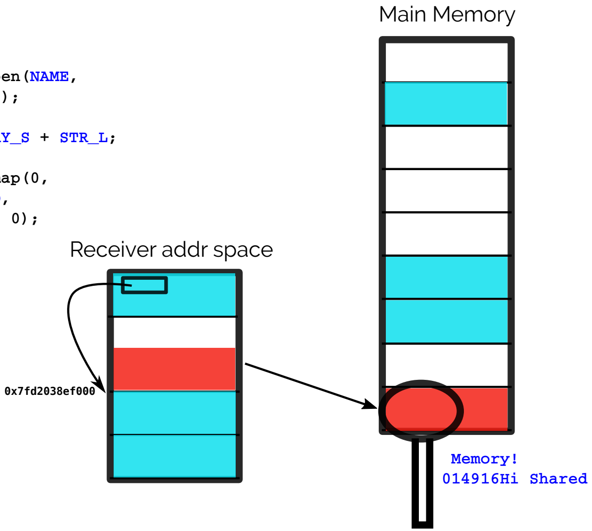


```
munmap(data, size);
```

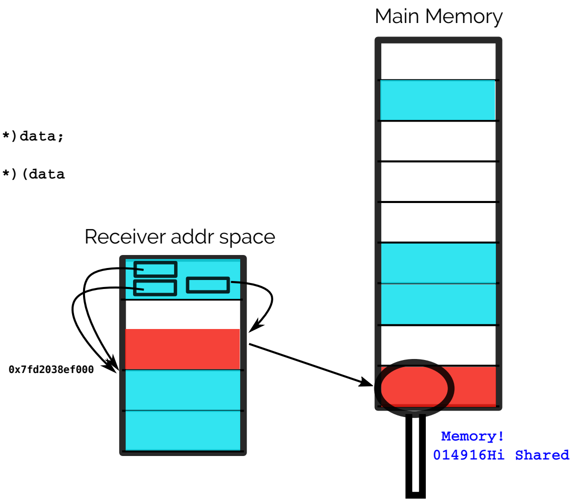



Main Memory
Receiver addr space
Memory!
014916Hi Shared

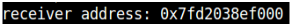
```
int fd = shm_open(NAME,
 O_RDONLY, 0666);

int size = ARRAY_S + STR_L;

void *data = mmap(0,
size, PROT_READ,
MAP_SHARED, fd, 0);
```



```
int *array = (int *)data;

char *msg = (char *)(data
 + ARRAY_S);
```

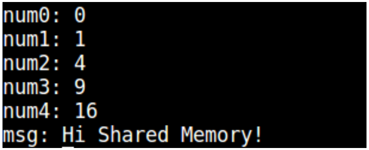


```
printf("receiver address: %p\n", data);
```



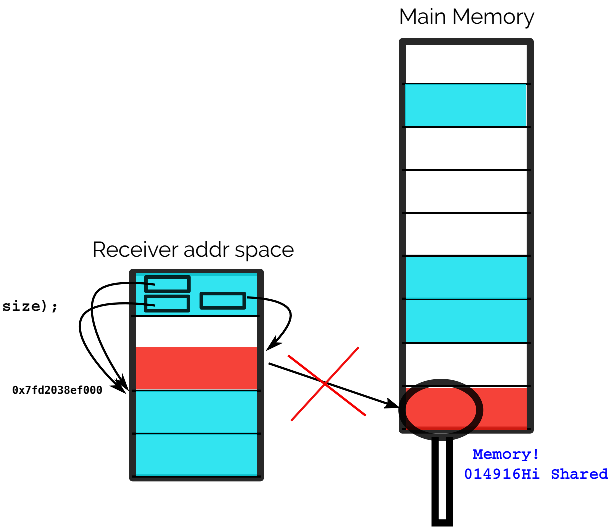
```
receiver address: 0x7fd2038ef000
```

```
for (int i = 0; i < NUM; i++)
{
 printf("num%d: %d\n", i,
array[i]);
}

printf("msg: %s\n", msg);
```



```
num0: 0
num1: 1
num2: 4
num3: 9
num4: 16
msg: Hi Shared Memory!
```

```
munmap(data, size);
```



What happens if we try to reference the red region in the address space now?

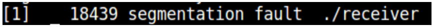
- for example:

```
printf("msg: %s\n", msg);
```

What happens if we try to reference the red region in the address space now?

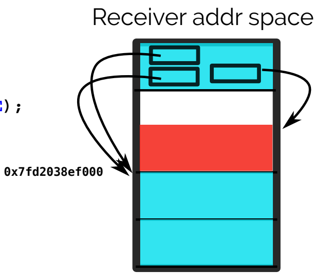
- for example:

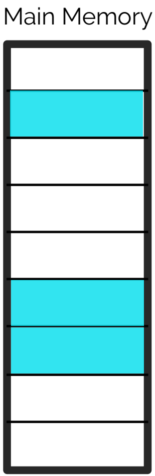
```
printf("msg: %s\n", msg);
```



```
[1]    18439 segmentation fault  ./receiver
```

```
close(fd);
shm_unlink(NAME);
```


Receiver addr space
0x7fd2038ef000


Main Memory