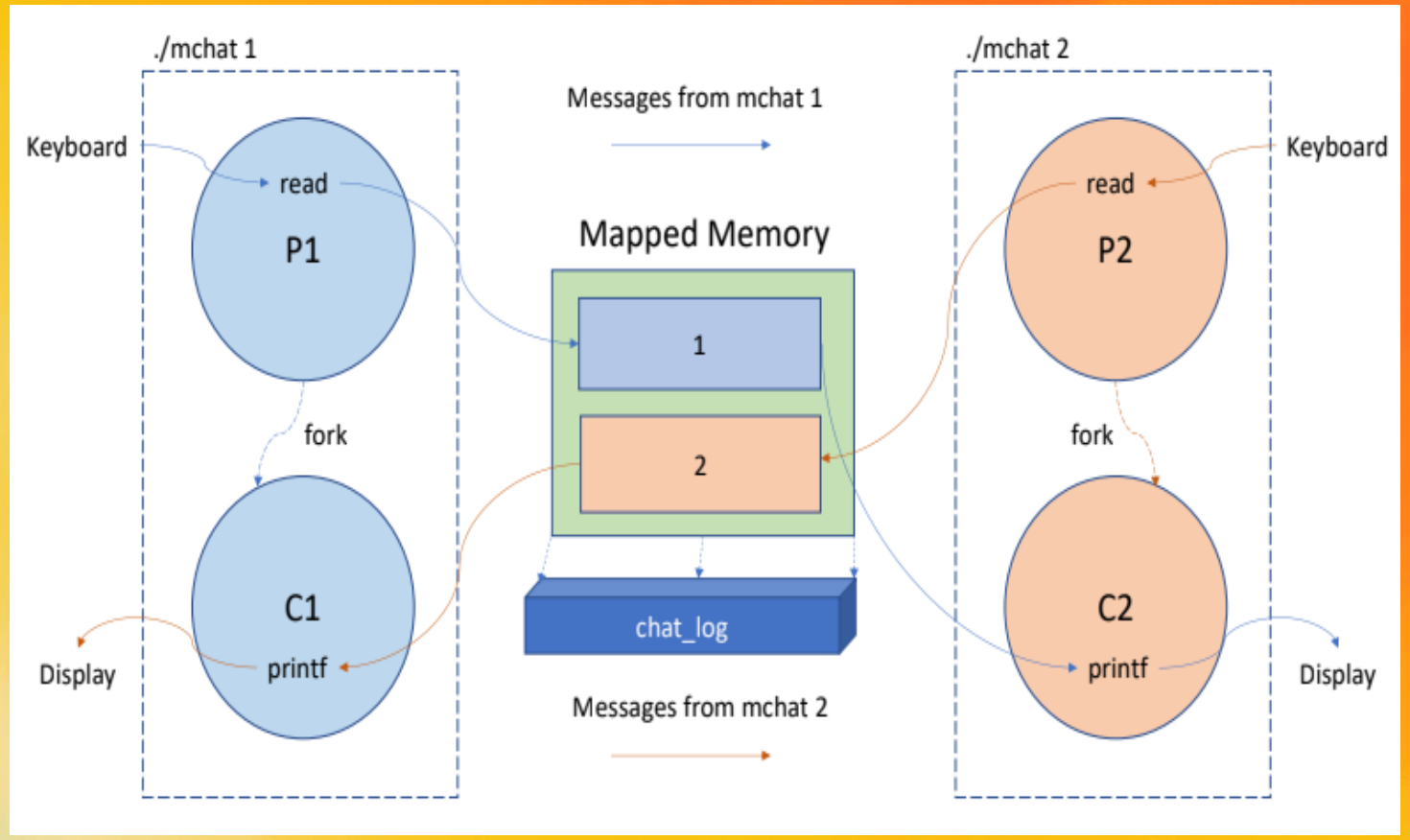
Mapped Memory



```c
static void forceKill()
{
  wait(NULL);
  exit(EXIT_SUCCESS);
}
```

จะถูกเรียกโดย signal ใช้ในการแก้ zombie

```c
struct mm_st
{
  int nbytes_0;
  int written_0;
  char data_0[BUFSIZ];
  int nbytes_1;
  int written_1;
  char data_1[BUFSIZ];
};
```

สร้าง struct เก็บค่า written แสดงการ
ถูกเขียน(1) ไม่ถูกเขียน(0) data เอาไว้
เก็บข้อมูลของแต่ละ user n
nbytes ใช้เก็บจำนวนข้อมูลใน data

```c
fd = open ("Chat log", O_RDWR | O_CREAT , S_IRUSR | S_IWUSR);
if(fd<0)
{
  perror("Open file error");
  exit(1);
}
```

เปิด File แบบ Readwrite ถ้าไม่มีให้สร้าง
ให้ permission Read Write กับ user

```
lseek (fd, FILE_LENGTH, SEEK_SET);
write (fd, "", 1);
lseek (fd, 0, SEEK_SET);
```

เติม \0 ที่ท้ายไฟล์

```
file_mem = mmap(NULL, FILE_LENGTH, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
if(file_mem==MAP_FAILED)
{
  perror("mmap() failed\n");
  exit(EXIT_FAILURE);
}
```

ทำการ map โดยให้ Linux
จัดการ address(NULL)
ให้ Readwrite
หลังจากที่มีการเขียน buffer
ลงบน File fd ตั้งแต่ตัวที่ 0

```
struct mm_st* mm_area;
mm_area = (struct mm_st*)file_mem;
memset(mm_area, 0, FILE_LENGTH);
mm_area->written_0=0;
mm_area->written_1=0;
mm_area->data_0[0]='\n';
mm_area->data_1[0]='\n';
mm_area->nbytes_0=0;
mm_area->nbytes_1=0;
```

สร้าง mm_area มาเก็บ address
ที่แปลงเป็น type pointer struct mm_st
เซ็ทค่าเริ่มต้นของแต่ละตัว

```
argv++;
if(strcmp(*argv, "1") == 0)
{
  child = fork();
  switch(child)
  {
    case -1: perror("Forking failed\n");
             exit(EXIT_FAILURE);
    case 0 : while(strncmp(buffer,"end chat",8))
             {
               if(mm_area->written_0)
               {
                 memset(buffer, 0, BUFSIZ);

                 int i = mm_area->nbytes_0;
                 int j;
                 do
                 {
                   i--;
                 }
                 while(mm_area->data_0[i]!='\n');

                 write(1,"User 2:", 7);
                 for(j=0, i=i+1;i<mm_area->nbytes_0+1;i++,j++)
                 {
                   buffer[j] = mm_area->data_0[i];
                 }
                 write(1, buffer, j);
                 mm_area->written_0 = 0;
               }
             }
             kill(child, SIGALRM);
             break;
```

ทำจนเจอ end chat
และ written_0 ต้องเป็น 1
เคลียร์ค่าใน buffer ก่อนรับ

คำนวณขนาดของคำล่าสุด
ที่อีก user ส่งมา

อ่านค่าจาก data_0
แล้วแสดงผลคำทางหน้าจอ

```
default : while(strncmp(buffer, "end chat", 8))
        {
          memset(buffer, 0, BUFSIZ);
          int nbytes = read(0, buffer, BUFSIZ);

          if(nbytes>1)
          {
            mm_area->nbytes_1+=nbytes;
            strcat(mm_area->data_1, buffer);
            mm_area->written_1 = 1;
          }
        }
        kill(child, SIGKILL);
        wait(NULL);
        break;
```

เขียนลงไฟล์เมื่อแก้ค่า string ใน struct

ทำจนกว่าจะ end chat

เคลียร์ค่าใน buffer

ตัด case \n ตัวเดียว

เพิ่มจำนวน nbytes

เปลี่ยนค่า data_1 โดยเขียนต่อไปเรื่อยๆ

แก้ written_1 เป็น 1

ส่งสัญญาณให้ parent เมื่อออกลูป

```
else if (strcmp(*argv, "2") == 0)
{
  child = fork();
  switch(child)
  {
    case -1: perror("Forking failed\n");
             exit(EXIT_FAILURE);
    case 0 :
             while(strncmp(buffer,"end chat",8))
             {
               if(mm_area->written_1)
               {
                 memset(buffer, 0, BUFSIZ);

                 int i = mm_area->nbytes_1;
                 int j;
                 do
                 {
                   i--;
                 }
                 while(mm_area->data_1[i]!='\n');

                 write(1,"User 1:", 7);
                 for(j=0, i=i+1;i<mm_area->nbytes_1+1;i++,j++)
                 {
                   buffer[j] = mm_area->data_1[i];
                 }
                 write(1, buffer, j);
                 mm_area->written_1 = 0;
               }
             }
             kill(child, SIGALRM);
             break;
    default : while(strncmp(buffer, "end chat", 8))
             {
               memset(buffer, 0, BUFSIZ);
               int nbytes = read(0, buffer, BUFSIZ);

               if(nbytes>1)
               {
                 mm_area->nbytes_0+=nbytes;
                 strcat(mm_area->data_0, buffer);
                 mm_area->written_0 = 1;
               }
             }
             kill(child, SIGKILL);
             wait(NULL);
             break;
```

ทำงานเหมือนกับแชท 1 แต่

ให้ลูกอ่านจาก data_1

ให้พ่อแม่เขียนโดยเป็น

data_0

```
munmap (file_mem, FILE_LENGTH);
close(fd);
return 0;
}
```

ยกเลิกการ map

กับปิด File